CS602 Applied Algorithms **Jan 2022**

Assignment 1 Solution

Total Marks: 30 *Deadline:*

Note: The assignment needs to be done individually. Any kind of discussion with other students is not allowed. If you feel the need to discuss anything, you can reach out to the instructor.

You can directly use any result proved in the class.

Que 1 [10 marks]. Suppose r is a specified vertex in a directed graph G. A directed tree T rooted at r is a tree where every edge is directed away from the root. In other words, T has a unique path from r to v, for each vertex v in G. Figure 1 shows a directed graph where there does not exist any directed tree rooted at vertex r. Figure 2 shows a directed graph with a directed tree rooted at r shown in blue color.

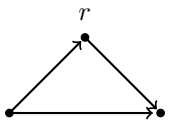

Figure 1: A directed graph with no directed trees rooted at r.

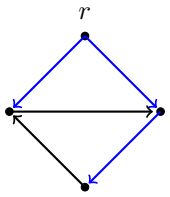

Figure 2: A directed graph with a directed treer rooted at r shown in blue.

In the weighted version of the problem, we are given weights on the edges of the graph and the goal is to find a minimum weight directed tree rooted at r. We can write the following linear program for this problem.

$$\begin{aligned} \min & \sum_{e \in E} w_e x_e \\ & \text{subject to} \\ & x_e \geq 0 \text{ for each } e \in E \\ & \sum_{e \in \delta^{in}(S)} x_e \geq 1 \text{ for each subset } S \subseteq V - \{r\} \end{aligned}$$

Here $\delta^{in}(S)$ means the set of edges incoming to the set S. And the last constraint is there for all subsets S of vertices which do not contain the root vertex r.

The dual of this LP is as follows.

$$\begin{aligned} \max & \sum_{S \subseteq V - \{r\}} y_S \\ & \text{subject to} \\ & y_S \geq 0 \text{ for each subset } S \subseteq V - \{r\} \\ & \sum_{S : e \in \delta^{in}(S)} y_S \leq w_e \text{ for each } e \in E \end{aligned}$$

Let us implement the following primal dual algorithm based on this LP.

1. Start with T as empty set. And set $y_S = 0$ for each S.
2. Pick any inclusionwise minimal subset S for which the primal constraint is not satisfied (that is, it has no incoming edges in T). Keep increasing the corresponding dual variable y_S till some dual constraint becomes tight.
3. If the dual constraint for an edge e becomes tight, set $x_e = 1$ and include e in T. If multiple constraints become tight, do this with all those edges one by one in an arbitrary order.
4. Stop if the T has a path from the root to every other vertex. Otherwise, go to 2.
5. **Pruning:** go over all the edges in T one by one in **reverse** order (the last edge that was included in T is to be considered first) and do the following. If after removal of an edge from T, it still has a path from the root to every other vertex, then we should remove that edge.
6. Output T.

Meaning of inclusionwise minimal in the algorithm: we mean that we should take that subset S for which the primal constraint is not satisfied, but for any proper subset of S the primal constraint is satisfied. For example, in the beginning, S can be any single vertex other than the root. For another example, if we have just one edge in T, say from u to v, then S cannot be taken as $\{v\}$ or $\{u, v\}$, but can be taken as $\{u\}$.

Prove that the tree T that is output by the above algorithm is a minimum weight directed tree rooted at r. If there is any doubt in the algorithm, please write to the instructor.

Hint: At the end of the algorithm we have a primal solution and a dual solution. One way to prove optimality is by showing complementary slackness (after the pruning phase).

Answer. First observe that every directed rooted tree gives a feasible integral solution for the primal LP. This is because in any directed rooted tree, every subset S of vertices not containing the root r will have at least one incoming from $V - S$. Thus, if we show that some directed rooted tree T gives an optimal solution for the primal LP, then T must be the optimal directed rooted tree.

Let us first argue that the output of the algorithm is a directed rooted tree. Line 4 in the algorithm ensures that we keep adding edges till the point that every vertex has a path from the root. A directed rooted tree requires that there no cycles and no vertex has two or more paths from the root. This is achieved in the pruning step. If there is a cycle, at least one of its edges can be removed while still maintaining the connectivity property. If a vertex has two paths from the root, again at least one of the edges can be removed safely. Hence, in whichever order we go over the edges, the pruning step will ensure that at the end, there are no cycles and no vertex has multiple paths from the root.

As said in the hint, optimality can be shown by complementary slackness. Complementary slackness has two conditions.

$$\text{for each } e \in E, x_e > 0 \implies \sum_{S: e \in \delta^{in}(S)} y_S = w_e$$

$$\text{for each subset } S \subseteq V - \{r\}, y_S > 0 \implies \sum_{e \in \delta^{in}(S)} x_e \geq 1$$

The first condition is true by the design of the algorithm. Note that in line 3, we set $x_e = 1$ only when the corresponding dual constraint is tight.

Now, we need to show the second condition. Another way to state the condition is, if the dual variable y_S is nonzero then there must be exactly one incoming edge to S in the tree T. First let us see how does S look like when the dual variable y_S is increased. The set S must be inclusionwise minimal set for which the primal constraint is not satisfied. That means, two things: (i) S has no incoming edge and (ii) every proper subset $R \subseteq S$ has an incoming edge. Observe that the incoming edge to R must come from within S, as S has no incoming edge. In other words, S is strongly connected at this point.

Observation: the first incoming edge for S is added only when S is already strongly connected.

The algorithm can add multiple incoming edges to S when y_S is increased. Moreover, further incoming edges to S can be added when $y_{S'}$ is increased for some superset $S' \supset S$. We need to argue that after the pruning step, only one incoming edge to S will remain. Here it is important that the order of pruning is the reverse of the order in which the edges were added. Reverse order of pruning and the above observation imply that during the phase in which the incoming edges to S are considered for pruning, the set S remains strongly connected. If S is strongly connected, then only one incoming edge to S will survive in pruning. This is because two or more incoming edges to a strongly connected component would mean more than one paths for each vertex in that component.

Hence, the complementary slackness conditions are satisfied and the output of the algorithm is an LP optimal solution. As discussed in the beginning a directed tree that is optimal for the LP is indeed a minimum weight directed tree.

Que 2 [10 marks]. Consider the Max 2-AND problem: a 2-AND clause is an AND of two literals (positive or negative). We are given a set of 2-AND clauses, and each clause has an associated weight. The goal is to find an assignment to the variables which maximizes the total weight of satisfied clauses.

For example, consider

$$(x_1 \wedge \overline{x}_2) \rightarrow \text{weight } 15,$$
$$(x_2 \wedge x_3) \rightarrow \text{weight } 10,$$
$$(\overline{x}_1 \wedge x_3) \rightarrow \text{weight } 10.$$

Here, if we set (x_1, x_2, x_3) =(True, False, True) then only the first clause is satisfied. If we set (x_1, x_2, x_3) =(False, True, True) then the last two clauses are satisfied. It is not possible to satisfy the first clause together with any other clause. Hence, the optimal value is 20.

Observe that if we set each variable to True with probability 1/2, then each clause is satisfied with probability 1/4. Hence, for an arbitrary input, we can get (1/4)-approximation simply by taking a random assignment.

Write an appropriate linear program for this problem. Then design a randomized (1/2)-approximation algorithm based on LP rounding. Note that the rounding scheme might have to be designed a bit more carefully than what we saw in the class. If t_i^* was the value of the i-th variable in the LP optimal solution, we had set the corresponding Boolean variable as true with probability t_i^*. Instead, we can try to take this probability as some linear function of t_i^*, for example, $t_i^*/2 + 1/4$.

Hint: When we wrote an LP in the class for the case of OR-clauses, we had one linear inequality for every clause. Here in case of AND clauses, you might need to put **two** linear inequalities for every clause.

Answer. Let's start with writing an appropriate LP for this. Let z_j be an LP variable for the jth clause. $z_j = 1$ is supposed to represent that the clause is satisfied and $z_j = 0$ is supposed to represent that the clause is not satisfied. Let t_i be an LP variable for the ith Boolean variable x_i. $t_i = 1$ is supposed to represent that x_i is set to True while $t_i = 0$ is supposed to represent that x_i is set to False.

Consider an example where the jth clause $x_1 \wedge x_2$. We would like to put linear constraints to ensure that if any one of t_1 and t_2 is set to 0 (i.e., one of x_1 and x_2 is set to False and so the jth clause is unsatisfied), then z_j must be 0. The following constraints ensure this.

$$\begin{aligned} z_j &\leq t_1 \\ z_j &\leq t_2 \end{aligned}$$

When both x_1 and x_2 are true, i.e., t_1 and t_2 are 1 then clear the above constraints allow z_j to go till 1. Along similar lines, one can add constraints for other kinds of clauses, for example, $x_1 \wedge \overline{x}_2$. Below is the final LP.

$$\max \sum_j w_j z_j$$

subject to

$$0 \le z_j \le 1 \quad \text{for each } j$$
$$0 \le t_i \le 1 \quad \text{for each } i$$
$$z_j \le t_i \quad \text{if clause } j \text{ has } x_i$$
$$z_j \le 1 - t_i \quad \text{if clause } j \text{ has } \overline{x}_i$$

An equivalent way of describing the above LP is

$$\max \sum_j w_j z_j$$

subject to

$$0 \le z_j \le 1 \quad \text{for each } j$$
$$0 \le t_i, f_i \le 1 \quad \text{for each } i$$
$$z_j \le t_i \quad \text{if clause } j \text{ has } x_i$$
$$z_j \le f_i \quad \text{if clause } j \text{ has } \overline{x}_i$$
$$t_i + f_i = 1 \quad \text{for each } i$$

Now, the plan is to find an optimal solution for this LP using any LP solver. Let (z^*, t^*) is an optimal solution for this LP. Below we discuss the randomized rounding scheme. As said in the question we will set x_i randomly with the following scheme.

$$x_i \leftarrow \begin{cases} \text{True} & \text{with probability } t_i^*/2 + 1/4 \\ \text{False} & \text{with probability } (1 - t_i^*)/2 + 1/4 \end{cases}$$

It's easy to see that the sum of the two probabilities is 1. Why we chose these probabilities will be clear in the analysis. To get an approximation guarantee, we need to compare the expected value of the satisfied clauses with the LP optimal value. The expected value of the satisfied clauses will be as follows.

$$\text{ALG} := \sum_j w_j \Pr[j\text{th clause is satisfied}] \quad = \quad \sum_j w_j \Pr[\text{ both the literals in the } j\text{th clause are set to True}]$$

Let p_j be the probability that the jth clause is satisfied. Depending on the clause type, the value of p_j will be as follows.

$$p_j = \begin{cases} (t_i^*/2 + 1/4) \times (t_k^*/2 + 1/4) & \text{if } j\text{th clause is } x_i \wedge x_k \\ (t_i^*/2 + 1/4) \times ((1 - t_k^*)/2 + 1/4) & \text{if } j\text{th clause is } x_i \wedge \overline{x}_k \\ ((1 - t_i^*)/2 + 1/4) \times ((1 - t_k^*)/2 + 1/4) & \text{if } j\text{th clause is } \overline{x}_i \wedge \overline{x}_k \end{cases}$$

The LP optimal value is given by

$$\text{LP-OPT} = \sum_j w_j z_j^*.$$

To show a 1/2-approximation guarantee, we would like to show that $\text{ALG} \ge (1/2) \times \text{LP-OPT}$. To show this, it will be sufficient to prove that

$$p_j \ge (1/2) \times z_j^*$$

We will prove this for the first case when jth clause is $x_i \wedge x_k$. The other two cases are similar, we just need to replace t_k^* with $1 - t_k^*$. In the first case, the following is implied from the LP constraints.

$$z_j^* \le t_i^*, \quad z_j^* \le t_k^*.$$

Then,

$$p_j = (t_i^*/2 + 1/4) \times (t_k^*/2 + 1/4) \ge (z_j^*/2 + 1/4)^2 = (z_j^*/2 - 1/4)^2 + z_j^*/2 \ge z_j^*/2.$$

Hence, $\text{ALG} \ge (1/2) \times \text{LP-OPT}$. This finishes the proof that the algorithm gives a 1/2-approximation.

Que 3 [10 marks]. Write an appropriate SDP for the above problem and design a randomized approximation algorithm based on rounding the optimal solution of the SDP. The approximation factor should be better than 1/2. Ideally, you can aim for 0.79.

As we had seen in the class, the starting point for writing the SDP would be to first write an integer program with variables constrained in {-1,1}, where the objective function has some degree two terms. When you try to write the objective function for this problem, you might get some degree two and some degree one terms. For example, $1 + y_3 + y_5 + y_3y_5$. But, to go to the 'inner product formulation', we would like only degree two terms. One way to fix this is to introduce an auxiliary variable y_0 and replace each y_i with y_0y_i. The previous expression becomes $1 + y_0y_3 + y_0y_5 + y_0^2y_3y_5$. Since $y_0 \in \{-1, 1\}$, we know $y_0^2 = 1$. Thus we get $1 + y_0y_3 + y_0y_5 + y_3y_5$. Finally, when you do the rounding by taking a random hyperplane, you can set the i-th variable to true if and only if it the corresponding vector falls on the same side as the vector for y_0. In particular, you will be setting y_0 to 1.

You will need to compute the probability of some event that involves three vectors. You can restrict your attention on 3 dimensional space. Whatever probability you get in 3 dimension, that applies to any dimension. To compute the desired probability, you might need the concept of *solid angle*. You can see this wikipedia page `https://en.wikipedia.org/wiki/Solid_angle` for the definition. The unit for solid angles is steradians. Just like, the angle of the whole circle is 2π radians, the solid angle of the whole sphere is 4π steredians. On the same wiki page, check the subsection on tetrahedron that gives a formula for the solid angle in terms of *dihedral angles* of the tetrahedron. This formula can be directly used. Another observation you might need is as follows: let u and v be two vectors with angle θ between them. Then the 2D planes orthogonal to u and v are given by $u^Tx = 0$ and $v^Tx = 0$ respectively. The dihedral angle between these planes in the region $u^Tx \leq 0$ and $v^Tx \leq 0$ is $\pi - \theta$.

The following inequality might be useful and you can directly use it. Let u_1, u_2, u_3 be three vectors in any dimension. Let θ_{12} be the angle between u_1 and u_2, let θ_{23} be the angle between u_2 and u_3, and let θ_{31} be the angle between u_3 and u_1. Then,

$$1 - \frac{\theta_{12} + \theta_{23} + \theta_{31}}{2\pi} \geq 0.79 \times \frac{1 + \cos\theta_{12} + \cos\theta_{23} + \cos\theta_{31}}{4}.$$

Answer. Let's first try to write an integer program with variables constrained in $\{-1, 1\}$. Let $y_i \in \{-1, 1\}$ be a variable which is supposed to be 1 if x_i is True and -1 otherwise. Let z_j be the clause variable for jth clause. Suppose jth clause is of the type $x_i \wedge x_k$. Consider the following equation.

$$z_j = (y_iy_k + y_i + y_k + 1)/4.$$

We can verify that z_j is 1 if both $y_i = 1$ and $y_k = 1$. Otherwise z_j is 0. Similarly, we can write an equation for z_j for all three types of clauses. We simply need to replace y_k with $-y_k$ and/or y_i with $-y_i$ in the above equation.

$$z_j = \begin{cases} (y_iy_k + y_i + y_k + 1)/4 & \text{if } j\text{th clause is } x_i \wedge x_k \\ (-y_iy_k + y_i - y_k + 1)/4 & \text{if } j\text{th clause is } x_i \wedge \overline{x}_k \\ (y_iy_k - y_i - y_k + 1)/4 & \text{if } j\text{th clause is } \overline{x}_i \wedge \overline{x}_k \end{cases}$$

We can verify that in the second case, z_j is 1 if and only if $y_i = 1$ and $y_j = -1$. In the third case, z_j is 1 if and only if $y_i = -1$ and $y_j = -1$. The objective function will be the usual one.

$$\max \sum_j w_j z_j.$$

To go to an SDP formulation, we would like to change the domain of y_is to be unit length vectors in some dimension. And the product y_iy_k is supposed to be replaced by the inner product of the two vectors. One issue is that there are linear terms like $+y_i$ in the above equations. To get rid of linear terms we will replace each y_i with y_0y_i, where y_0 is also constrained to be in $\{-1, 1\}$. Note that whatever be y_0, y_0^2 is always 1. The new equation we get is as follows.

$$z_j = \begin{cases} (y_iy_k + y_iy_0 + y_ky_0 + 1)/4 & \text{if } j\text{th clause is } x_i \wedge x_k \\ (-y_iy_k + y_iy_0 - y_ky_0 + 1)/4 & \text{if } j\text{th clause is } x_i \wedge \overline{x}_k \\ (y_iy_k - y_iy_0 - y_ky_0 + 1)/4 & \text{if } j\text{th clause is } \overline{x}_i \wedge \overline{x}_k \end{cases}$$

Now, the equation has only degree 2 terms and they can be replaced by inner product. Let n be the number of Boolean variables in the problem. Together with y_0, we will have $n+1$ y-variables. If we have $n+1$ vectors, their span can be at most $n+1$ dimensional. Hence, we will restrict these vectors in $\mathbb{R}^{n+1}$. The following is the vector program we will write.

$$\begin{aligned}
&\max \sum_j w_j z_j \\
&\text{subject to} \\
&y_i \in \mathbb{R}^{n+1} \text{ with } \|y_i\| = 1 \text{ for } 0 \le i \le n, \\
&z_j = \begin{cases} (y_i^T y_k + y_i^T y_0 + y_k^T y_0 + 1)/4 & \text{if } j\text{th clause is } x_i \wedge x_k \\ (-y_i^T y_k + y_i^T y_0 - y_k^T y_0 + 1)/4 & \text{if } j\text{th clause is } x_i \wedge \overline{x}_k \\ (y_i^T y_k - y_i^T y_0 - y_k^T y_0 + 1)/4 & \text{if } j\text{th clause is } \overline{x}_i \wedge \overline{x}_k \end{cases}
\end{aligned}$$

As discussed in the class, the above program can be converted to an SDP. Consider a matrix M whose (i,k) entry is supposed to contain the inner product $y_i^T y_k$. As discussed in the class, such an inner product matrix is always PSD.

$$\begin{aligned}
&\max \sum_j w_j z_j \\
&\text{subject to} \\
&M \in \mathbb{R}^{(n+1)\times(n+1)}, \\
&M_{i,i} = 1, \\
&M \succeq 0, \\
&z_j = \begin{cases} (M_{i,k} + M_{i,0} + M_{k,0} + 1)/4 & \text{if } j\text{th clause is } x_i \wedge x_k \\ (-M_{i,k} + M_{i,0} - M_{k,0} + 1)/4 & \text{if } j\text{th clause is } x_i \wedge \overline{x}_k \\ (M_{i,k} - M_{i,0} - M_{k,0} + 1)/4 & \text{if } j\text{th clause is } \overline{x}_i \wedge \overline{x}_k \end{cases}
\end{aligned}$$

Rounding. Now, we come to the rounding part. Suppose the optimal solution for the above vector program is $y*$. That is, in the optimal solution we have vector $y_i^* \in \mathbb{R}^{n+1}$ corresponding to Boolean variable x_i. We need to decide whether to set x_i to True or False. As said in the hint, we will take random hyperplane H in $\mathbb{R}^{n+1}$. If vector y_i^* is on the same side of H as the vector y_0^*, then we will set x_i to True otherwise False. To analyze the expected value, we need to compute the probability that the jth clause is satisfied. Suppose jth clause is $x_i \wedge x_k$. We will come to the other types of clauses later. We need to find the probability that both x_i and x_k are set to True. That is, both y_i^* and y_k^* fall on the same side of H as vector y_0^*. In the class we had discussed the probability that y_i^* falls on the same side as y_0^*. Note that the two events – (i) y_i^* and y_0^* are on the same side (ii) and y_k^* and y_0^* are on the same side – are not independent of each other. So the probabilities cannot be simply multiplied.

First let us see an easy way to compute the probabilities. This idea is from Mihir Vahanwala's submission. We had seen in the class that probability that two vectors will fall on the opposite sides of H is θ/π, where θ is the angle between the two vectors. When H is chosen randomly, there are only four possible events which are disjoint from each other:

1. E_1: all three y_k^*, y_i^*, and y_0^* are on the same side of H,
2. E_2: y_k^* and y_i^* are on the same side of H, but y_0^* is on the other side,
3. E_3: y_0^* and y_i^* are on the same side of H, but y_k^* is on the other side,
4. E_4: y_0^* and y_k^* are on the same side of H, but y_i^* is on the other side.

Let the probabilities of these four events be p_1, p_2, p_3, p_4 respectively. Observe that the event that y_k^* and y_i^* are on the opposite sides of H is exactly the union of events E_3 and E_4. Hence, we can write

$$p_3 + p_4 = \theta_{i,k}/\pi,$$

where $\theta_{i,k}$ is the angle between y_i^* and y_k^*. Similarly, we can write

$$p_2 + p_4 = \theta_{i,0}/\pi,$$

$$p_2 + p_3 = \theta_{k,0}/\pi,$$

where $\theta_{i,0}$ is the angle between y_i^* and y_0^* and $\theta_{k,0}$ is the angle between y_k^* and y_0^*. From the above three equations, we have

$$1 - p_1 = p_2 + p_3 + p_4 = \frac{\theta_{i,0} + \theta_{i,k} + \theta_{k,0}}{2\pi}.$$

Equivalently, the probability that all three y_k^*, y_i^*, and y_0^* are on the same side of H, is given by

$$p_1 = 1 - \frac{\theta_{i,0} + \theta_{i,k} + \theta_{k,0}}{2\pi}.$$

Alternate argument to get the same probability expression: Let the hyperplane H is described by $h^T x = 0$. Since y_i^*, y_k^*, y_0^* span a three dimensional space, without loss of generality we can assume that h is chosen randomly from the same three dimensional space. The event of our interest is that the signs of $h^T y_i^*$, $h^T y_k^*$, $h^T y_0^*$ are the same. The probability of this event is simply double of the probability that the three signs are all positive. This happens when h falls into the cone C defined by

$$C = \{u : u^T y_i^* \geq 0, u^T y_k^* \geq 0, u^T y_0^* \geq 0\}.$$

The probability of h falling into C will be $\phi_C/(4\pi)$, where ϕ_C is the solid angle of the cone C. Recall that the solid angle of the whole three dimensional space is 4π. It is known that the solid angle of cone C is given by the following (see `https://en.wikipedia.org/wiki/Solid_angle`)

$$\phi_{i,0} + \phi_{i,k} + \phi_{k,0} - \pi$$

where $\phi_{i,k}$ is the angle between the positive sides of the two planes $y_i^{*T} u = 0$ and $y_k^{*T} u = 0$. One can verify that $\phi_{i,k} = \pi - \theta_{i,k}$. Combining everything together, the solid angle

$$\phi_C = 2\pi - (\theta_{i,0} + \theta_{i,k} + \theta_{k,0}).$$

The probability that h falls into C is

$$\frac{2\pi - (\theta_{i,0} + \theta_{i,k} + \theta_{k,0})}{4\pi}.$$

The probability that the signs of $h^T y_i^*$, $h^T y_k^*$, $h^T y_0^*$ are the same is double of this:

$$\frac{2\pi - (\theta_{i,0} + \theta_{i,k} + \theta_{k,0})}{2\pi} = 1 - \frac{\theta_{i,0} + \theta_{i,k} + \theta_{k,0}}{2\pi}.$$

This is the desired probability.

Comparison with SDP optimal. We have computed the probability that the jth clause is satisfied, which is

$$1 - \frac{\theta_{i,0} + \theta_{i,k} + \theta_{k,0}}{2\pi}.$$

To compare the expected value with the SDP optimal value, we need to compare this probability with z_j^* which is equal to

$$z_j^* = (y_i^{*T} y_k^* + y_i^{*T} y_0^* + y_k^{*T} y_0^* + 1)/4 = (\cos\theta_{i,k} + \cos\theta_{i,0} + \cos\theta_{k,0} + 1)/4.$$

From the given inequality in the question,

$$1 - \frac{\theta_{i,0} + \theta_{i,k} + \theta_{k,0}}{2\pi} \geq 0.79 \times z_j^*.$$

The same bound can be derived for other kinds of clauses $x_i \wedge \overline{x}_k$ and $\overline{x}_i \wedge \overline{x}_k$. For that, we just need to replace vector y_k^* with $-y_k^*$ in the previous arguments.

Hence, we conclude that expected value of the satisfied clauses is at least 0.79 times the SDP optimal value. Hence, the algorithm gives 0.79-approximation.