602 Applied Algorithms **Jul-Nov 2023**

Endsem Solutions

Total Marks: 90 *Time: 180 minutes*

Instructions.

- Please write your answers to the point.
- You can directly use anything proved in the class.

Que 1 (Smallest ellipse) [5+4 marks]. Recall the following from the class: The smallest ellipse which contains the half-ellipse

$$\{(y_1, y_2) : y_1^2 + y_2^2 \leq 1, y_1 \geq 0\}$$

is given by $\{(y_1, y_2) : (3y_1 - 1)^2 + 3y_2^2 \leq 4\}$.

Find the smallest ellipse which contains the half-ellipse described by the following constraint

$$\begin{pmatrix} 1 - x_1 - x_2 & 3(x_2 - x_1) & 0 \\ 3(x_2 - x_1) & 1 + x_1 + x_2 & 0 \\ 0 & 0 & x_1 + x_2 \end{pmatrix} \succeq 0.$$

Describe the obtained ellipse using a PSD constraint.

Ans 1. Simplyfying the given PSD constraint for half-ellipse

$$E_1 := \{(x_1, x_2) : 1 - (x_1 + x_2)^2 - 9(x_2 - x_1)^2 \geq 0, x_1 + x_2 \geq 0\}.$$

Equivalently,

$$E_1 := \{(x_1, x_2) : (x_1 + x_2)^2 + 9(x_2 - x_1)^2 \leq 1, x_1 + x_2 \geq 0\}.$$

Observe that this can be obtained by applying a linear transformation $y_1 = x_1 + x_2$ and $y_2 = 3(x_2 - x_1)$ on the half-ellipse

$$E_0 := \{(y_1, y_2) : y_1^2 + y_2^2 \leq 1, y_1 \geq 0\}.$$

Hence, to get the smallest ellipse containing E_1, we can apply the same transformation on the smallest ellipse containing E_0. Smallest ellipse containing E_0:

$$S_0 := \{(y_1, y_2) : (3y_1 - 1)^2 + 3y_2^2 \leq 4\}.$$

Applying transformation $y_1 = x_1 + x_2$ and $y_2 = 3(x_2 - x_1)$, we get

$$S_1 := \{(x_1, x_2) : (3x_1 + 3x_2 - 1)^2 + 27(x_2 - x_1)^2 \leq 4\}.$$

This is the smallest ellipse containing E_1. Representing this using PSD constraint,

$$\begin{pmatrix} 3(1 - x_1 - x_2) & 3\sqrt{3}(x_2 - x_1) \\ 3\sqrt{3}(x_2 - x_1) & 1 + 3x_1 + 3x_2 \end{pmatrix} \succeq 0.$$

Alternatively,

$$\begin{pmatrix} 2 - 3\sqrt{3}(x_2 - x_1) & 3x_1 + 3x_2 - 1 \\ 3x_1 + 3x_2 - 1 & 2 + 3\sqrt{3}(x_2 - x_1) \end{pmatrix} \succeq 0.$$

Que 2 (Separation for PSD cone) [6 marks]. The below matrix A is not positive semidefinite.

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

Find a separating hyperplane between A and the set of 3×3 positive semidefinite matrices. In other words, find numbers $\{c_{i,j} : 1 \le i < j \le 3\}$ such that

$$c_{1,1}x_{1,1} + c_{1,2}x_{1,2} + c_{1,3}x_{1,3} + c_{2,2}x_{2,2} + c_{2,3}x_{2,3} + c_{3,3}x_{2,3} \ge 0$$

for every positive semidefinite matrix $\begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{1,2} & x_{2,2} & x_{2,3} \\ x_{1,3} & x_{2,3} & x_{3,3} \end{pmatrix}$ and

$$c_{1,1} \cdot 1 + c_{1,2} \cdot 1 + c_{1,3} \cdot 1 + c_{2,2} \cdot 0 + c_{2,3} \cdot 1 + c_{3,3} \cdot 3 < 0.$$

Ans 2. We will do both-sided Gaussian elimination.

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 3 \end{pmatrix} \xrightarrow{row_2 \leftarrow row_2 - row_1} \begin{pmatrix} 1 & 1 & 1 \\ 0 & -1 & 0 \\ 1 & 1 & 3 \end{pmatrix} \xrightarrow{col_2 \leftarrow col_2 - col_1} \begin{pmatrix} 1 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 3 \end{pmatrix} = B$$

The last matrix has diagonal entry -1, hence not PSD. We can take $v = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$. Then $v^T B v = -1$. On the other hand we know that $v^T X v \ge 0$ for every PSD matrix X. Representing the above transformations in matrix form we can write

$$B = \begin{pmatrix} 1 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 3 \end{pmatrix} \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thus, we can write $v^T B v = \begin{pmatrix} -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 3 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}$.

In other words $u^T A u < 0$ for $u = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}$. On the other hand we know that $u^T X u \ge 0$ for every PSD matrix X. From here, we get the desired numbers $c_{i,j}$.

$$\begin{aligned} c_{1,1} &= u_1 u_1 = 1 \\ c_{1,2} &= 2u_1 u_2 = -2 \\ c_{1,3} &= 2u_1 u_3 = 0 \\ c_{2,2} &= u_2 u_2 = 1 \\ c_{2,3} &= 2u_2 u_3 = 0 \\ c_{3,3} &= u_3 u_3 = 0 \end{aligned}$$

Que 3 (Barrier function) [6+4 marks]. Consider the following linear program.

$$\begin{aligned} \min -x_1 + x_2 + 2x_3 \text{ subject to} \\ x_1 \geq 0 \\ x_2 \geq 0 \\ x_3 \geq 0 \\ x_1 + x_2 + x_3 = 1 \end{aligned}$$

Suppose we want to solve this LP using the interior point method. First step is to convert this into an unconstrained optimization problem, where the objective function will be a combination of a barrier function and the given objective function with a parameter η. Describe this objective function.

When we take $\eta = 0$, what will be the optimizing point.

Ans 3. First we need to bring the LP in the standard form $Ax \leq b$. There are many ways of doing this. One is as follows.

$$\begin{aligned} \min -x_1 + x_2 + 2x_3 \text{ subject to} \\ x_1 \geq 0 \\ x_2 \geq 0 \\ x_3 \geq 0 \\ x_1 + x_2 + x_3 \geq 1 \\ -x_1 - x_2 - x_3 \geq -1 \end{aligned}$$

For this LP, we will get the barrier function $-\log(x_1) + -\log(x_2) - \log(x_3) - \log(x_1 + x_2 + x_3 - 1) - \log(1 - x_1 - x_2 - x_3)$. The problem with this is that the last two summands are not defined for any feasible solution, where we have $x_1 + x_2 + x_3 = 1$. Instead we will consider the following equivalent LP, by replacing x_3 with $1 - x_1 - x_2$.

$$\begin{aligned} \min -3x_1 - x_2 + 2 \text{ subject to} \\ x_1 \geq 0 \\ x_2 \geq 0 \\ 1 - x_1 - x_2 \geq 0 \end{aligned}$$

Now the barrier function will be $-\log(x_1) - \log(x_2) - \log(1 - x_1 - x_2)$. The combined objective function will be

$$\min \eta(-3x_1 - x_2 + 2) - \log(x_1) - \log(x_2) - \log(1 - x_1 - x_2).$$

This is equivalent to

$$\min \eta(-x_1 + x_2 + 2x_3) - \log(x_1) - \log(x_2) - \log(x_3) \text{ subject to } x_1 + x_2 + x_3 = 1.$$

Optimizing point: When we take $\eta = 0$, and we have minimize $-\log(x_1) - \log(x_2) - \log(1 - x_1 - x_2)$. We take derivative with respect to both x_1 and x_2 and put it equal to zero.

$$\begin{aligned} -1/x_1 + 1/(1 - x_1 - x_2) = 0 \\ -1/x_2 + 1/(1 - x_1 - x_2) = 0 \end{aligned}$$

Solving this, we get $x_1 = x_2 = 1/3$. And $x_3 = 1 - x_1 - x_2 = 1/3$.

Que 4 (Simplex algorithm optimal guarantee) [5 marks]. Consider an LP

$$\max w^T x \text{ subject to } Ax = b, x \geq 0,$$

where A is a $k \times n$ matrix. Assume non-degeneracy condition. Suppose we have a basic feasible solution α, where the basic variables are $x_1, x_2, \ldots, x_k$. That means $\alpha_{k+1} = \alpha_{k+2} = \cdots = \alpha_n = 0$.

Using some linear transformations, we can express the objective function as a function of $x_{k+1}, x_{k+2}, \ldots, x_n$. That is,

$$c_0 + c_{k+1}x_{k+1} + c_{k+2}x_{k+2} + \cdots + c_n x_n,$$

for some $c_0, c_{k+1}, \ldots, c_n$. Prove that if $c_{k+1} \leq 0, \quad c_{k+2} \leq 0, \ldots, c_n \leq 0$, then α is an optimal solution. What will be the LP optimal value?

Ans 4. In the simplex algorithm, we increase one of the non-basic variables. Starting from α, if we increase one of the non-basic variables, the objective function does not increase. This argument only shows that when we move along an edge, the objective function does not increase. This is not sufficient to prove that α is the optimal solution.

Since $\alpha_{k+1} = \alpha_{k+2} = \cdots = \alpha_n = 0$, the objective function value at α will be

$$c_0 + c_{k+1} \cdot 0 + c_{k+2} \cdot 0 + \cdots + c_n \cdot 0 = c_0.$$

We claim that c_0 is the optimal value for the LP. To see this consider any other feasible solution β. From LP constraints we know that $\beta_{k+1} \geq 0, \beta_{k+2} \geq 0, \ldots, \beta_n \geq 0$. Hence, The objective function value at β will be

$$c_0 + c_{k+1}\beta_{k+1} + c_{k+2}\beta_{k+2} + \cdots + c_n\beta_n \leq c_0.$$

The last inequality is true because $c_{k+1} \leq 0, c_{k+2} \leq 0, \ldots, c_n \leq 0$. We conclude that every feasible point has objective value at most c_0. Hence, α is an optimal solution with objective value c_0.

Que 5 (Simplex algorithm simulation) [10 marks]. Consider the following linear program.

$$\begin{aligned} \max 2x_1 + x_2 \text{ subject to} \\ x_1 \geq 0 \\ x_2 \geq 0 \\ x_1 \leq 1 \\ x_1 + x_2 \leq 2 \end{aligned}$$

- First write the LP in the standard form $Ax = b, x \geq 0$. The number of variables will become 4.
- Start with the basic feasible solution that has $x_1 = 0, x_2 = 0$. What will be the values of x_3, x_4?
- Run the iterations of the simplex algorithm till you get an optimal solution. After each iteration write down (i) the basic feasible solution and (ii) express the objective function in terms of the current non-basic variables. In an iteration, there may be multiple possible choices for which variable to increase. You can just make an arbitrary choice.

Below is how an iteration of simplex algorithm runs.

- Choose one of the non-basic variables to increase. It should be among those whose coefficient in the objective function is positive. If there is no such variable then output the current solution. If there is such a variable then increase it till the maximum possible value while maintaining feasibility.
- This gives you a new basic feasible solution and a new set of basic and non-basic variables (non-basic variables become zero).
- Express the objective function in terms of the non-basic variables.

Ans 5. In the desired standard form.

$$\begin{aligned} \max 2x_1 + x_2 \text{ subject to} \\ x_1 \geq 0 \\ x_2 \geq 0 \\ x_3 \geq 0 \\ x_4 \geq 0 \\ x_3 = 1 - x_1 \\ x_4 = 2 - x_1 - x_2 \end{aligned}$$

One possible run of the simplex:

- Initial basic feasible solution: $x_1 = 0$, $x_2 = 0$, $x_3 = 1$, $x_4 = 2$.
- Iteration 1: Increase x_1 till 1. $x_1 = 1$, $x_2 = 0$, $x_3 = 0$, $x_4 = 1$.
  Non-basic variables x_2 and x_3. Rewriting objective function: $x_2 + 2 - 2x_3$.
- Iteration 2: Increase x_2 till 1. $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 0$.
  Non-basic variables x_3 and x_4. Rewriting objective function: $3 - x_4 - x_3$.
- In the objective function all coefficients are negative. We cannot increase any non-basic variable now. We have the optimal solution.

Another possible run of the simplex:

- Initial basic feasible solution: $x_1 = 0$, $x_2 = 0$, $x_3 = 1$, $x_4 = 2$.
- Iteration 1: Increase x_2 till 2. $x_1 = 0$, $x_2 = 2$, $x_3 = 1$, $x_4 = 0$.
  Non-basic variables x_1 and x_4. Rewriting objective function: $x_1 + 2 - x_4$.
- Iteration 2: Increase x_1 till 1. $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 0$.
  Non-basic variables x_3 and x_4. Rewriting objective function: $3 - x_4 - x_3$.
- In the objective function all coefficients are negative. We cannot increase any non-basic variable now. We have the optimal solution.

Any of the above two solutions is fine.

Que 6 (Steiner tree primal dual simulation) [10 marks]. Run the Steiner tree primal dual algorithm (described below) on the following graph (see Figure 1) with three terminal vertices u, v, w. Edge weights are shown on the edges.

For each iteration of the algorithm, clearly write

1. for which sets C the dual variable y_C got updated and the updated values
2. and the edges which got included in the solution.

At the end, which edges will be pruned (if any)? What is the cost of Steiner tree you got? Is it optimal?

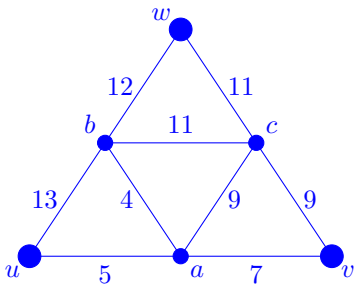

Figure 1: Graph with edge weights and three terminals

The primal dual algorithm:

Invariants: F will be maintained as a forest with tight edges.

Initialization: $F = \emptyset$. Set dual variables $y_C = 0$ for each separating set C of vertices. A set C of vertices is called separating if at least one terminal is inside C and at least one terminal is outside C.

1. Let $C_1, C_2, \ldots, C_q$ be those connected components in graph (V, F) which are separating.
2. Increase $y_{C_1}, y_{C_2}, \ldots, y_{C_q}$ simultaneously till some edge becomes tight.
3. Add newly tight edges to F (one by one, avoiding cycle formation)
4. Stop if all terminals u, v, w are connected to each other. Otherwise go to 1.
5. Pruning: delete any edges that are not used in connecting any terminals.

Ans 6. Below we show the four iterations and the pruning at the end.

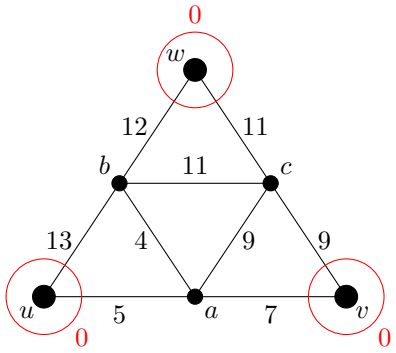

Figure 2: Initialization: $y_C = 0$ for all separating sets C.

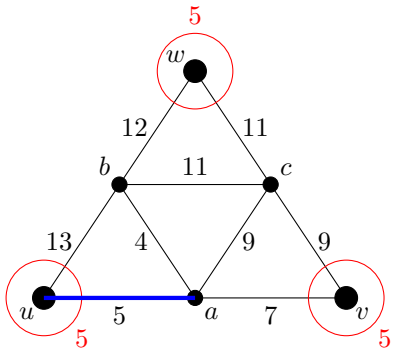

Figure 3: Iteration 1: Increase by 5. $y_{\{u\}} = y_{\{v\}} = y_{\{w\}} = 5$. Tight edges included $\{(u, a)\}$.

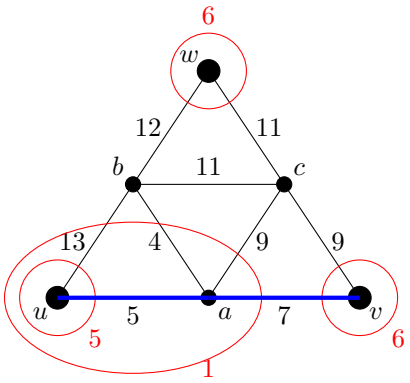

Figure 4: Iteration 2: Increase by 1. $y_{\{u,a\}} = 1$, $y_{\{v\}} = y_{\{w\}} = 6$. Tight edges included $\{(v, a)\}$.

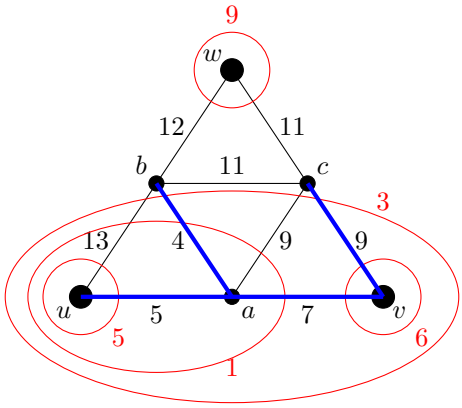

Figure 5: Iteration 3: Increase by 3. $y_{\{u,a,v\}} = 3$, $y_w = 9$. Tight edges included $\{(a, b), (v, c)\}$.

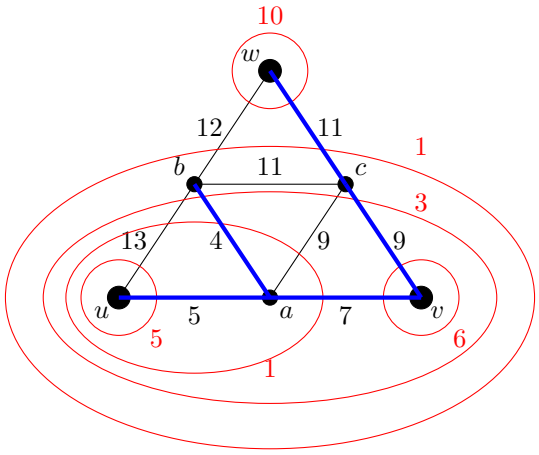

Figure 6: Iteration 4: Increase by 1. $y_{\{u,a,v,b,c\}} = 1$, $y_w = 10$. Tight edges included $\{(c, w)\}$.

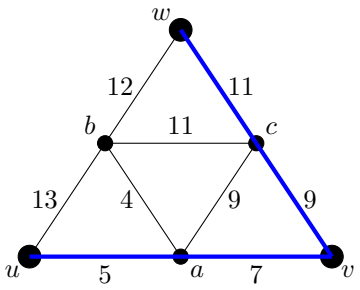

Figure 7: Pruning: remove edge (a, b), as it is not used in connecting any terminals.

The cost of the obtained Steiner tree is 32. It's not optimal. There is another better solution $\{(u, a), (a, b), (a, v), (b, w)\}$.

Que 7 (Hitting set via LP rounding) [10 marks]. We have n objects, $V = \{v_1, v_2, \ldots, v_n\}$, each with a cost, say $c_1, c_2, \ldots, c_n$. We are given some target subsets $T_1, T_2, \ldots, T_k \subseteq V$ each with 2 or 3 objects.

Example. $V = \{v_1, v_2, v_3, v_4, v_5\}$ with costs $\{20, 15, 10, 10, 20\}$. $T_1 = \{v_1, v_4\}, T_2 = \{v_2, v_4, v_5\}, T_3 = \{v_1, v_2, v_3\}$.

The goal is to select a *hitting-set* $H \subseteq V$ with minimum total cost, such that H contains at least one object from every target subset T_i. In the above example, $H = \{v_1, v_2\}$ is a hitting set with cost 35 and $H = \{v_3, v_4\}$ is a hitting-set with cost 20.

We write the following LP for minimum cost hitting-set, with variables $x_1, x_2, \ldots, x_n$ and k constraints.

$$\min \sum_{i=1}^{n} c_i x_i \text{ subject to}$$
$$x_i \geq 0 \text{ for } 1 \leq i \leq n$$
$$\sum_{v_i \in T_j} x_i \geq 1 \text{ for } 1 \leq j \leq k$$

Design a 3-approximation algorithm for the above problem via an appropriate rounding of an LP optimal solution. Recall that each target set has 2 or 3 objects.

Hint: choose an appropriate threshold θ. If $x_i^* \geq \theta$ then include ith object in the hitting-set.

Ans 7. Let $x^* \in \mathbb{R}^n$ be an optimal solution for the linear program. We choose threshold $\theta = 1/3$ and use the following rounding scheme: If $x_i^* \geq 1/3$ then include the object i in the set H.

Claim 1. *H is a hitting-set.*

Proof. For every target set T_j, we have the optimal solution satisfying the constraint $\sum_{i \in T_j} x_i^* \geq 1$. Since the set T_j has at most 3 objects, we get that for at least one object i in the set T_j it must be $x_i^* \geq 1/3$. Hence, at least one object will be included in H from every target subset T_j. □

Approximation guarantee: We will compare the cost of the obtained set H and the LP optimal cost. By construction, for every object i in H, $x_i^* \geq 1/3$.

$$\text{LP-OPT} = \sum_{i=1}^{n} c_i x_i^* \geq \sum_{i \in H} c_i x_i^* \geq \sum_{i \in H} c_i (1/3) = (1/3) \times \text{cost}(H)$$

Equivalently, $\text{cost}(H) \leq 3 \times \text{LP-OPT} \leq 3 \times$ OPT. The last inequality is true because LP optimal value is at most the cost of the optimal hitting-set. Hence, we have a 3-approximation algorithm.

Que 8 (MAXCUT integrality gap) [7 marks]. Recall the vector program we wrote for the MAXCUT problem. Here $w_{i,j}$ is the weight of the edge (i, j).

$$\max \sum_{i<j} w_{i,j} x_{i,j} \text{ subject to}$$
$$z_i \in \mathbb{R}^n \text{ for each } 1 \le i \le n$$
$$z_i^T z_i = 1 \text{ for each } 1 \le i \le n$$
$$x_{i,j} = \frac{1 - z_i^T z_j}{2} \text{ for each } 1 \le i < j \le n$$

Recall that the integrality gap for a given input, is the ratio of the maxcut value and the optimal value of the above program.

Find the integrality gap for the complete graph on 3 vertices, where all edge weights are equal. Since all weights are equal, you can assume that the optimal solution of the vector program will be symmetric. Is it larger or smaller than 0.878?

Ans 8. Let the edge weights are all w. Then the maxcut value for the complete graph on 3 vertices will be simply $2w$.

The optimal solution for the vector program will have three vectors z_1^*, z_2^*, z_3^* of unit length. In the objective function we are trying to maximize

$$\sum_{i<j} w x_{i,j} = \sum_{i<j} w \frac{1 - z_i^T z_j}{2} = \sum_{i<j} w \frac{1 - \cos\theta_{i,j}}{2},$$

where $\theta_{i,j}$ is the angle between z_i and z_j. That means we trying to maximize the angles between the three vectors. As mentioned in the question, the optimal solution will be symmetric. Hence, we can assume that the three angles will be equal. If θ is the angle between all three pairs of vectors, what can be the maximum possible value of θ? The maximum value will be when the three vectors are in a plane, and $\theta = 2\pi/3$.

Let's prove this formally. This part is not expected in the solution. Let us assume by symmetry, $z_i^T z_j = \alpha$ for every pair (i, j). We know that the corresponding matrix with these inner products must be PSD.

$$\max(1 - \alpha)/2 \text{ subject to } \begin{pmatrix} 1 & \alpha & \alpha \\ \alpha & 1 & \alpha \\ \alpha & \alpha & 1 \end{pmatrix} \succeq 0.$$

This means minimize α subject to $1 - \alpha^2 \ge 0$ and $2\alpha^3 - 3\alpha^2 + 1 = (\alpha - 1)^2(2\alpha + 1) \ge 0$. The minimum possible α is $-1/2$. That is $\theta = 2\pi/3$.

Then objective value for the vector program will be

$$\sum_{i<j} w \frac{1 - \cos(2\pi/3)}{2} = \sum_{i<j} w(3/4) = 9w/4.$$

Integrality gap = (max cut value)/(vector program optimal value) = (2w)/(9w/4) = 8/9 = 0.8888... .

8/9 is greater than 0.878. What this example shows that there is no rounding scheme that can achieve approximation factor larger than 0.888..... And the random hyperplane scheme already achieves 0.878, which is pretty close.

Que 9 (MAX k-cut) [8 marks]. For a given graph $G(V, E)$ with weights $\{w_e \geq 0 : e \in E\}$ on edges, max k-cut problem asks to partition the vertices into k parts, say $V_1, V_2, \ldots, V_k$, so as to maximize the sum of the weights of edges going across different parts. That is, we want to maximize the total weight of the set $\{(u, v) \in E : u \in V_i,\ v \in V_j \text{ and } i \neq j\}$.

Give an efficient randomized algorithm for max k-cut problem, where the expected weight of the obtained solution is at least $(k-1)/k$ times the optimal k-cut value. You need to prove the approximation guarantee.

Hint: we had seen a simple randomized approximation algorithm for maximum satisfiability in the class.

Ans 9. We randomly partition the set of vertices into k parts as follows: for each vertex $v \in V$, randomly and independently put it in $V_1, V_2, \ldots, V_k$, each with probability $1/k$. Let us compute the expected weight of the cut edges. Let w_e be the weight of any edge e. For an edge $e = (u, v)$, it contributes to the cut edges if only if u and v fall into different parts. Let us define a random variable X_e for each edge e as follows:

$$X_e = \begin{cases} 1 & \text{if the endpoints of } e \text{ fall into different parts,} \\ 0 & \text{otherwise.} \end{cases}$$

Let W be the total weight of the obtained cut edges. Observe that

$$W = \sum_{e \in E} w_e X_e.$$

Using linearity of expectation,

$$\mathbb{E}[W] = \sum_{e \in E} w_e \mathbb{E}[X_e] = \sum_{e \in E} w_e \Pr[\text{endpoints of } e \text{ fall into different parts}].$$

Now, we just need to compute Pr[endpoints of e fall into different parts] for each edge e. One can see that

$$\Pr[\text{endpoints of } e \text{ fall into different parts}] = 1 - (1/k \times 1/k) - (1/k \times 1/k) - \cdots - (1/k \times 1/k) = 1 - 1/k.$$

Here $1/k \times 1/k$ is the probability of the two endpoints falling into the same part V_i, and we subtracted this term for every for $i = 1, 2, \ldots, k$. Finally, we get

$$\mathbb{E}[W] = \sum_{e \in E} w_e (k-1)/k = (k-1)/k \sum_{e \in E} w_e.$$

Note that the optimal weight of a k-cut is trivially upper bounded by $\sum_{e \in E} w_e$. Hence, the expected weight of the obtained k-cut is at least (k-1)/k times the optimal k-cut value.

Que 10 (Separation oracle for job assignment) [15 marks]. Suppose we have k jobs and n (more than k) applicants for the jobs. Each applicant comes with a cost, say $c_1, c_2, \ldots, c_n$. Not every applicant is suitable for a job. For each job j, we are given the list $L_j \subseteq \{1, 2, \ldots, n\}$ of applicants suitable for it. We want to select the minimum cost set of k applicants, such that it is possible to do a one-to-one assignment of the k jobs to suitable selected applicants. We write the following LP.

$$\min \sum_{i=1}^{n} c_i x_i \text{ subject to}$$
$$x_i \geq 0 \text{ for } 1 \leq i \leq n$$
$$\sum_{i \in L_S} x_i \geq |S| \text{ for each non-empty subset } S \subseteq \{1, 2, \ldots, k\},$$

where

$$L_S = \bigcup_{j \in S} L_j.$$

The last constraint is saying that for any set S of *jobs*, we should look at the set of applicants who are suitable for any of these jobs (this set is L_S), and select at least $|S|$ applicants from it.

If we want to solve this LP using one of the LP algorithms, which algorithm seems most suitable? Design a polynomial time separation oracle for this LP.

Hint: you may use the s-t mincut subroutine.

Ans 10. Since there are 2^k constraints, ellipsoid method is most suitable. Ellipsoid method does not depend on the number of constraints.

Separation oracle: given a point $x \in \mathbb{R}^n$, we need to check whether all the LP constraints are satisfied, and if not then output one of the violated constraints. The number of constraints is exponential in k, so we cannot check each constraint one by one. Non-negativity constraints can be checked easily. To check the other constraints let us define a function $f(S)$ over sets $S \subseteq \{1, 2, \ldots, k\}$.

$$f(S) := \sum_{i \in L_S} x_i - |S|.$$

The LP constraints are equivalent to $f(S) \geq 0$ for every non-empty set $S \subseteq \{1, 2, \ldots, k\}$. Note that $f(\emptyset) = 0$, hence, we can remove the non-empty condition. So, the LP constraints are equivalent to

$$\min_{S \subseteq \{1,2,\ldots,k\}} f(S) \geq 0.$$

So, if we can compute this minimum efficiently, we will be done. Let us define a directed flow network. We have one vertex for each job, one vertex for each applicant, a source vertex s and a sink vertex t.

- Put an edge from the s to each job vertex with capacity one.
- Put an edge from applicant i to the sink with capacity x_i.
- Put an edge from each job j to its suitable applicants L_j with capacity ∞.

Now, we compute the s-t mincut in this network. Let the mincut be $s \cup S \cup A$, where S is a subset of job vertices and A is a subset of applicant vertices. Since it's a mincut, none of the ∞ capacity edges can be cut edges. That means all the vertices in the $L_S = \bigcup_{j \in S} L_j$ are inside A. If there are any vertices in A which are not in L_S, then we can remove them from A and the cut value can only decrease (or stay same). Hence, we can assume $A = L_S$. The capacity of this cut will be

$$\sum_{i \in L_S} x_i + k - |S| = f(S) + k$$

. If s-t mincut value is less than k then we have found a set S with $f(S) < 0$, that is a violated constraint.

Let us argue that if there is a violated constraint, we will find one. If at all there is a violated constraint that is $f(S) < 0$ for some S, then the cut value for $s \cup S \cup L_S$ will be $f(S) + k < k$. And hence, s-t mincut value is less than k.