6.854 Advanced Algorithms

Problem Set 7

John Wang
Collaborators:

Problem 1: Another way to formulate the maximum-flow problem as a linear program is via flow decomposition. Suppose we consider all $s-t$ paths P in the network G and let f_P be the amount of flow on path P. Then maximum flow says to find $z = \max \sum f_P$ subject to $\sum_{P \ni e} f_P \leq u_e$ for all edges e and $f_P \geq 0$ for all paths P. Take the dual of this LP and give an English explanation of the objective and constraints.

Solution: To take the dual of this problem, we must find corresponding variables for each of the constraints. Let y_e be the variables in the dual corresponding to the constraints $\sum_{P \ni e} f_P \leq u_e$ (one for each edge e). By formulating the dual's matrix, we see that the objective function for the dual becomes $min \sum_e u_e y_e$.

Using the coefficients on the objective from the primal, we can find some of the constraints for the dual. Namely, we know that $y_e \geq 0$. Moreover, we find the condition $\sum_{e \in P} y_e \geq 1$ for all paths P because of the objective function in the primal. Therefore, we have the following linear program:

$$\min \sum_e u_e y_e \tag{1}$$

$$\text{s.t.} \quad \sum_{e \in P} y_e \geq 1 \tag{2}$$

$$y_e \geq 0 \tag{3}$$

We can think of y_e as the length of each edge. Thus, the two constraints say that the total distance from s to t using the edges must be positive. The second constraint, $y_e \geq 0$ says that edge lengths must be greater than 0. If we use this interpretation to interpret the objective, we find that we are attempting to minimize the sum of the length times the capacity of each edge e.

In other words, we're attempting to assign lengths y_e which minimize the total cost of crossing over to t. Since we know that $z = \max \sum f_P = \min \sum_e u_e y_e$, we know that minimizing the lengths will be equal to the value of the min-cut, by LP duality. If we assign lengths of 1 over all edges that cross the minimum S, T cut, then we can be assured that $\sum f_P = \sum_e u_e y_e$ if we set all other edge lengths to 0. Thus, the dual of this problem is to find the minimum S,T cut and assign edges lengths of 1 to all edges crossing the cut. □

6.854 Advanced Algorithms

Problem Set 7

John Wang
Collaborators:

Problem 2-a: Explain why this captures the minimum mean cycle problem.

Solution: First, we know that f_{ij} represents a circulation. We see that f_{ij} is constrained to be non-negative by the fact that $f_{ij} \geq 0$. Moreover, we know that the flow across the entire graph must be equal to 1 because $\sum_{i,j} f_{ij} = 1$.

Now, we want to minimize $\sum c_{ij} f_{ij}$, which means that for each edge (i, j), we want to minimize $c_{ij} f_{ij}$. Suppose for the sake of contradiction that when we decompose $\{f_{ij}\}$ into cycles that there is some cycle cp which is not a minimum mean cycle. Let us say it is of length l and of cost c. Suppose the minimum mean cycle is of length l^* and cost c^*. Then we know that $c/l > c^*/l^*$. We also know that the cycle cp contribute cf/l to the objective function, where f is the circulation flowing through the cycle cp.

However, if we replace the cycle cp with the minimum mean cycle, then we can contributed $c^* f/l^*$ to the objective function. Notice that $c^* f/l^* < cf/l$, which means that our previous objective function was not minimal. This is a contradiction, so the circulation must go over minimum mean cycles. □

Problem 2-b: Give the dual of this linear program–it will involve maximizing a variable λ.

Solution: Let us create a variable λ which corresponds to the constraint $\sum_{i,j} f_{ij} = 1$. Since we have equality here, we know that λ will be unbounded in sign. Let us create a set of variables y_i which will correspond to the constraints $\sum_j f_{ij} - f_{ji} = 0$, one for each vertex i.

Based on the primal's objective function, we see that the new constraints in the dual will be given by $\lambda + y_i - y_j \leq c_{ij}$. Moreover, by the constraint matrix for the dual, we know that we want to maximize λ. Thus, we obtain the following problem:

$$\max \lambda \tag{4}$$

$$\text{s.t.} \qquad \lambda + y_i - y_j \leq c_{ij} \tag{5}$$

Where λ and each of the y_i are unbounded in sign, and each of the constraints are set for all i, j. □

Problem 2-c: Give an explanation (in terms of min-cost-flow reduced costs) for why this dual formulation also captures minimum mean cycles. (Hint: how much is added to the cost of a k-edge cycle).

Solution: If we rewrite the constraint $\lambda + y_i - y_j \leq c_{ij}$, we find that $\lambda \leq y_j + c_{ij} - y_i$. We know that c_{ij} are costs, and therefore, we can think of λ as the reduced cost, since we can think of y_i as a price function defined at each vertex i.

This means that we can define a valid price function y_i where all reduced costs are positive (if the solution to the dual is optimal). Moreover, we know that if λ is optimal, then we must have $0 = c_{ij} - \lambda + y_j - y_i$ for some cycle, since otherwise we could make λ marginally larger and obtain a better objective function while still satisfying the constraints.

This means that for some cycle K, we have $\sum_{(i,j)\in K} c_{ij} - \lambda + y_i - y_j = 0$, which simplifies to $\sum_{(i,j)\in K} c_{ij} - \lambda = 0$ by telescoping. Therefore, we see that:

$$\sum_{(i,j)\in C} c_{ij} = l\lambda \tag{6}$$

Where l is the length of the cycle. Now, let us set c to the total cost of the cycle, so that we know $c = \sum_{(i,j)\in C} c_{ij}$. Thus, we see that $c/l = \lambda$ is minimized, or in other words, we find a minimum mean cost cycle. □

Problem 2-d: Let's assume the costs c_{ij} are integers. Suggest a combinatorial algorithm (not based on linear programming) that uses binary search to find the right λ to solve the dual problem. Can you use this to find a minimum mean cycle? Note: to know when you can terminate the search, you will need to lower bound the difference between the smallest and the next smallest mean cost of a cycle.

Solution: First, we know that $\lambda \in [0, c_{max}]$ where c_{max} is the maximum cost for any edge in the graph. This follows since we know that $(c_{ij} - \lambda) + y_i - y_j = 0$, so that we must have $c_{ij} \geq \lambda$. This means that we can perform a binary search for λ inside of the interval $[0, c_{max}]$.

We search by taking examining the interval $[a, b]$ and taking $(b-a)/2$. Once we have a candidate for $\lambda = (b-a)/2$, then we can check if this is feasible. Feasibility is checked by looking for a negative-cost cycle using the reduced edge costs with $\lambda = (b-a)/2$. We can augment the graph with this new λ, then check for negative-cost cycles by using Bellman-Ford. If we find a negative reduced cost cycle, then we must check for λ which are smaller, so we recurse on the interval $[a, (b-a)/2]$. If there are no negative reduced cost cycles, then we can increase λ while still being feasible, so we recurse on the interval $[(b-a)/2, b]$.

We stop recursing when we have $b - a < \frac{1}{n^2}$, where n is the total number of vertices in the graph. This follows because each cycle can consist of at most n vertices each. We simply need to find a lower bound on the difference between the smallest and the second smallest mean cost of a cycle. However, we know that the mean costs $c_1 l_1 / k_1$ and $c_2 l_2 / k_2$ will have a difference $c_1 l_1 k_2 - c_2 l_2 k_1 >= 1$, since both of these values are integers. This implies that the mean costs differ by at least $\frac{1}{k_1 k_2} \geq \frac{1}{n^2}$.

This means that we can be sure that we are done recursing when $b-a$ comes within $\frac{1}{n^2}$. This is because, by the integrality of costs, we are assured that $c_1 l_1 k_2 - c_2 l_2 k_1$ is an integer, and so we can bound the difference between the smallest and second smallest mean costs by $\frac{1}{n^2}$. Therefore, we can terminate the search when this occurs.

We can then go and find the minimum mean cycle by using λ in the reduced costs, then we can simply look for the minimum mean cost cycles by looking for zero edge weight cycles, which can be found by using a simple BFS. $\square$

6.854 Advanced Algorithms

Problem Set 7

John Wang
Collaborators:

Problem 3-a: Argue that any LP optimization problem can be transformed into the form $min 0\dot{x}$ subject to $Ax = b$, $x \geq 0$. This LP has optimum 0 if it is feasible and ∞ if no.

Solution: We know that every LP optimization problem can be expressed in standard form. Therefore, we have the problem $\min cx$ such that $Ax = B, x \geq 0$. We derived in class that the dual of the standard form is $\max by$ such that $A^T y \geq c$. We p be the solution of the primal and let d be the solution of the dual.

We know that in the optimal solution, we must have $p = d$, so we need to check whether $p = cx = by = d$, subject to all the conditions in both the primal and dual. Therefore, to check optimality, we look for feasibility of the following LP:

$$by = cx \tag{7}$$

$$\text{s.t. } Ax = b \tag{8}$$

$$x \geq 0 \tag{9}$$

$$Ay \geq c \tag{10}$$

Now, to find the optimum of either the primal or the dual, we just need to check feasibility of the above program. Finding a feasible solution will automatically give us an optimum solution by looking at the $p = by$ such that $by = cx$. Moreover, if either the primal or the dual is unbounded or infeasible, there will be no feasible solution to this LP. Notice that the feasibility of this program is equivalent to the minimization of $0\dot{x}$, which is the objective function of the problem statement. If there are unbounded variables in the original primal, we can decompose $v = v^+ + v^-$ which represent the positive and negative portions of the unbounded variable. These can then enter into the variable set for the new LP.

Thus, we have created the problem $\min 0\dot{x}$ subject to $Ax = b$ where $x \geq 0$, where we have transformed any standard form LP. □

Problem 3-b: What is the dual of this linear program?

Solution: We can create a new variable y which will correspond to the constraint $Ax = b$. We see that the new objective function will be given by y amultiplied by its corresponding constraint, so we have $\max(Ax)^T y$. Since we know that $(Ax)^T = b^T$, we can rewrite this to $\max b^T y$. We are left to find the constraints of the dual.

We can use the coefficients from the primal's objective function to find the constant for the constraint. Moreover, we know that we want to set $A^T y$ as the left half of the constraint. Since we have $x \geq 0$, in the natural direction, we must have $A^T y$ in the other direction. We see that the constraint becomes $A^T y \leq 0$. Thus, the dual LP is:

$$\max b^T y \tag{11}$$

$$\text{s.t. } A^T y \leq 0 \tag{12}$$

□

Problem 3-c: Argue if the primal is feasible then the dual has an obvious optimum solution.

Solution: We know that all feasible solutions for the primal must have an objective of 0. Otherwise, we would have $0 \cdot x = \infty$, in which case the solution would not be feasible. Since we know that all feasible solutions to the primal have an objective of 0, we know by strong LP duality that all solutions of the dual must also have a solution of 0. We can also choose $y = 0$ and note that it satisfies $b^T y = 0$ and also the constraint $A^T y = 0$. Therefore, the obivous opimum solution for the dual is $y = 0$, leading to the objective function of value 0. □

Problem 3-d: Deduce that, given the hypothetical algorithm above, you can build an LP algorithm that will solve any LP without knowing beforehand a dual solution, in the same asymptotic time bounds as the algorithm above.

Solution: Let us use the magic algorithm given above to solve any LP without knowing a dual solution. First we know that any LP can be converted into standard form, then put in the form of problem 3-a, call this LP'. We know from problem 3-c that if the primal is feasible, then the dual has the optimum solution $y = 0$.

Thus, we can run the magic algorithm on LP', using the dual solution $y = 0$. We know that the algorithm finishes and gives an optimum solution in $O((m + n)^k)$ time if the dual solution is optimal. Since $y = 0$ might not be an optimal solution if LP' does not have any feasible solutions, then we must check to make sure the magic algorithm outputs a correct result. There exists some constant c such that after $c(m + n)^k$ steps, we can cut off the algorithm and check whether it has returned an optimum value for LP' yet.

If the value it outputs is feasible, so that $Ax = b$ and $x \geq 0$, then we know it must also be optimal for LP' because the objective simply takes $\min 0 \cdot x$. However, if the output is infeasbile, then LP' does not have any feasible solution. Notice that if LP' has a feasible solution then we are guaranteed to find it using the magic algorithm in this way, and if LP' does not have a feasible solution, then our check will discover this as well.

Now, we know that feasibility checking is equivalent to solving an LP. If we are given an LP in standard form, i.e. $\min cx$ such that $Ax = b$ and $x \geq 0$, then we just perform feasibility checking using the above algorithm to see if the problem $by = cx, Ax = b, x \geq 0, A^T y = c$ has any feasible solution. Our algorithm will return a feasible solution if it exists, and therefore, will also be able to provide an optimal solution to the original standard form LP. Thus, we ave been able to solve any LP in $O((m+n)^k)$ time using this magic algorithm. $\square$

6.854 Advanced Algorithms

Problem Set 7

John Wang
Collaborators:

Problem 4-a: Given a standard form LP $min\{cx|Ax = b, x \geq 0\}$, show how to define a different LP with an obvious feasible point, whose optima are precisely the feasible points of the original LP.

Solution: For each constraint in the constraint matrix defined by $Ax = b$, we will add a slack variable y_i to create a new constraint matrix. Basically, for each equation $a_i x = b_i$ we will create an equation $a_i x + y_i = b_i$. We know that the equations originally are in the form $a_i x = b_i$ because we have a standard form LP.

After we have defined these new variables y_i, we can create a vector $y = [y_1, y_2, \ldots, y_n]$. Now take the new LP:

$$\min \sum_i y_i \text{s.t} \qquad Ax + y = b \tag{13}$$

$$y_i \geq 0 \forall i x_i \geq 0 \forall i \tag{14}$$

Now, we have new variables y_i along with our previous variables x_i. However, we can trivially satisfy our constraints by setting $x_i = 0$ for all i, then setting $y_i = b_i$ for all i. This follows since $Ax = 0$, so we can be sure that $y = b$ will satisfy $Ax + y = b$. Moreover, we know that the optimum points of this LP are the x_i's when $\sum_i y_i = 0$. Since all y_i are positive, we must have $y = 0$ for this to occur.

However, if $y = 0$, then we can see that our original constraint $Ax = b$ is satisfied. Since we constrained $x_i \geq 0 \forall i$, we will have found a feasible solution to our standard form LP with the optimum of our new LP. □

Problem 4-b: Explain how this solves our problem, allowing us to use a simplex solver (which requires a feasible starting point) to solve arbitrary LPs without being given a feasible starting point.

Solution: We can use our new LP A from problem 4-a above to find a feasible starting point for simplex. First, we use the formulation above to create a problem whose optimum is an optimum for our original LP. Since we know that $x = 0$ and $y = b$ will suffice as a starting point in A, we can run simplex to find the optimum solution of A.

If in the optimum we find $y_i = 0 \forall i$, then we know that x will be a feasible starting point for our original LP. If this is not the case, then our original LP has no feasible points and thus is not feasible–this is because no variables x_i can be used to satisfy the constraints $Ax = b$ and $x_i \geq 0 \forall i$.

Now, once we have an feasible starting point for our original LP, we can run simplex to solve our LP. □

6.854 Advanced Algorithms

Problem Set 7

John Wang
Collaborators:

Problem 5-a: Explain how a network design problem can be solved using the ellipsoid algorithm. In particular, give a linear program involving exponentially many constraints and provide an algorithm that, given any proposed network, finds a violated constraint in polynomial time. (Hint: the demands must be satisfied by flows, which are intimately related to cuts).

Solution: □