P1:10 P2:9 P3:40 P4:16 39

6.854 Dan Ports
Advanced Algorithms drkp@mit.edu
2004/10/20 Collaborators: {sarahl,pramook}

Problem Set 6

Problem 1:

We first write the linear program in terms of the flow vector $\vec{f}$, which is a column vector with each row representing the flow through one path, and the capacity vector $\vec{u}$, which in each row has the capacity through one edge. Then the linear program is

$$z = \max \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} \vec{f}$$
$$A\vec{f} \leq \vec{u}$$
$$\vec{f} \geq 0$$

The constraint matrix A has one column for each path, and one row for each edge, and contains a 1 in entry A_{ij} if edge i is contained in path j.

The dual of this problem is

$$w = \min \vec{u}^T \vec{y}$$
$$A^T \vec{y} \geq \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}$$
$$\vec{y} \geq \vec{0}$$

y is a vector which contains one element for each edge. For each edge e, we have

$$y_e \geq 0$$

and for each path P, we have

$$\sum_{e \in P} y_e \geq 1$$

We can consider each y_e to be the "length" of edge e, which cannot be negative. The objective is to minimize the total "volume" $\sum_{e \in E} u_e y_e$ subject to the constraints that the distance from the source to the sink (the total length along each $s \to t$ path) $\sum_{e \in p} y_e$ is at least 1. To ensure this, the minimum $\vec{y}$ will have $y_e = 1$ for each e in the minimum cut, and $y_e = 0$ for every other edge.

Problem 2:

a) Suppose we are given a LP optimization problem. Any LP can be written in standard form, i.e.

what if init LP is infeas?

$$z = \min \vec{c}^T \vec{x}$$
$$A\vec{x} \geq \vec{b}$$
$$\vec{x} \geq \vec{0}$$

This LP has a dual that can be written as

$$w = \max \vec{b}^T \vec{y}$$
$$A^T \vec{y} \leq \vec{c}$$
$$\vec{y} \geq \vec{0}$$

By strong duality, the optimum solutions of the primal and dual are the same:

$$\vec{c}^T\vec{x} = \vec{b}^T\vec{y}$$

Thus, if we define $\vec{x'} = \begin{pmatrix} \vec{x} \\ \vec{y} \end{pmatrix}$, we can express the constraints of both problems as

$$\begin{aligned} \begin{bmatrix} A & 0 \end{bmatrix}\vec{x'} &\geq b \\ \begin{bmatrix} 0 & A^T \end{bmatrix}\vec{x'} &\leq c \\ \begin{bmatrix} \vec{c}^T & -\vec{b}^T \end{bmatrix}\vec{x'} &= 0 \\ \vec{x'} &\geq 0 \end{aligned}$$

Since this captures the constraints that the value of the original primal and dual are the same, any feasible solution to this problem is an optimal solution to the original linear program. We can therefore choose to minimize any function over this space, e.g. $\vec{0}\vec{x}$. We can also express the linear program in canonical form. This gives a program of the desired form.

b) If the linear program is expressed in canonical form as

$$\begin{aligned} z &= \min \vec{0}\vec{x} \\ A\vec{x} &= \vec{b} \\ \vec{x} &\geq \vec{0} \end{aligned}$$

then the dual is

$$\begin{aligned} w &= \max \vec{b}^T\vec{y} \\ A^T\vec{y} &\leq \vec{0} \end{aligned}$$

$\vec{y}$ is unrestricted in sign.

c) Recall that if the primal is feasible, it has minimum value $w = 0$. Since the primal and dual have the same optimum values, the maximum value of the dual is also zero. So

$$\vec{b}^T\vec{y} = 0$$

The solution

$$\vec{y} = \vec{0}$$

satisfies this constraint, and also the constraint that

$$A^T\vec{y} \leq \vec{0}$$

d) Suppose we wish to solve any linear program (without knowing a dual solution). We can transform the linear program to the form in part a, and by part c, we know a solution to the dual. We can therefore use the $O\left((m+n)^k\right)$ algorithm to find a solution to the new linear program, since we have a solution to the dual. From this solution we can easily extract the solution to the original problem.

We need only show that the transformation of the problem in part a does not change the size of the constraint matrix beyond a constant factor. Observe that transforming a linear program into the standard or canonical increases the size of the problem by at most a constant factor: we need only

two inequality constraints to replace each equality constraint when converting to standard form, and we need to add only one slack variable to the program when converting an inequality constraint to canonical form, and all other transformations required do not change the size of the constraint matrix. Note also that combining the primal and the dual into a unified linear program as done in part a at most doubles the size of each dimension of the constraint matrix. Thus, our algorithm is guaranteed to run in $O\left((m+n)^k\right)$ time.

Problem 3:

a) We consider f_{ij} as the flow through the edge between i and j (the gross flow, since $f_{ij} \geq 0$). Note that f_{ij} defines a circulation, since we have the conservation constraint $\forall i \sum_j f_{ij} - f_{ji} = 0$, with no source or sink. The constraint $\sum f_{ij} = 1$ ensures that the total flow will be $1/e$, where e is the number of edges with non-zero flow in the network. Hence, minimizing the cost $\sum c_{ij} f_{ij}$ identifies a cycle with the minimum cost per edge, i.e. a minimum mean cycle.

b) The dual problem will have two types of variables, since there are two types of constraints in the primal. For each vertex v, we have a variable y_v, corresponding to the conservation constraint at that v, and we have a single variable λ resulting from the constraint $\sum f_{ij} = 1$.

Since the conservation constraint values are zero and the constraint value for the flow-sum constraint is 1, the objective for the dual is to maximize λ. $\vec{y}$ and λ are unrestricted in sign, since the constraints are equality constraints.

Note that the constraint matrix A for the conservation constraint has one column per edge and one row per vertex, plus one row for λ. It contains $+1$ for edge e and vertex v if e enters v, and -1 if e leaves v, and the λ row contains only ones. So the dual constraint $A^T \vec{y} \leq \vec{c}$ translates to

$$\forall e_{ij} \quad y_j - y_i + \lambda \leq c_{ij}$$

c) From above, we have the constraint $y_j - y_i + \lambda \leq c_{ij}$. Or equivalently,

$$c_{ij} - y_j + y_i - \lambda \geq 0$$

If $\vec{y}$ is interpreted as defining a price function on the vertices, then this is the reduced cost minus λ:

$$r_{ij} - \lambda \geq 0$$

This means that if λ is subtracted from every edge cost, there are no negative-cost cycles. So we are finding the value of λ that can be subtracted from every edge cost without creating a negative cost cycle. When this λ is subtracted from every edge, one cycle has cost zero. If the cycle has k edges, it originally had total cost λk; thus, λ is the mean cost of this cycle. Since this is the first zero-cost cycle created, λ is the minimum mean cycle cost, and $\vec{y}$ is a price function for a flow on the minimum mean cycle.

d) We can test whether a candidate value λ^* is greater or less than λ by subtracting λ^* from every edge, then using the Bellman-Ford algorithm to check whether there are any negative-cost cycles. If a negative-cost cycle exists, then $\lambda^* > \lambda$; otherwise, $\lambda^* \leq \lambda$. Note also that the minimum mean cycle cost λ is bounded by the largest and smallest values of the cost function.

We can therefore use a standard binary search algorithm to repeatedly narrow the possible range of values for λ, beginning with the interval between the largest and smallest values of the cost function. We test whether the median value of the interval is greater or smaller than λ, and then recursively search on the half of the interval that contains λ.

To show that this algorithm terminates, we need a lower bound on the difference between the minimum mean cycle and the next smallest mean cycle cost. We can find the smallest difference between two edge costs, then divide by the total number of edges in the graph. No two cycles can have a mean cost that differs by less than this amount, so when the search has reduced this interval to a length less than this amount, we have found a minimum mean cycle.

Problem 4:

a) Suppose Alice's mixed strategy $\vec{x}$ is known. Then Bob wishes to maximize $\vec{x}A\vec{y}$. But $\vec{x}$ is constant, so $\vec{x}A$ is a constant row vector. It must therefore have a maximal element. The elements y_i of Bob's strategy must sum to 1. So the optimal strategy $\vec{y}$ is the column vector that contains 1 as the element corresponding to the maximal element of $\vec{x}A$, and zero elsewhere. This is a pure strategy.

4/2

b) Alice wishes to choose x to minimize $\max_{\sum y_i=1} \vec{x}A\vec{y}$. For any $\vec{x}$, Bob's optimal strategy will be a pure strategy, i.e. y will contain one one element and the rest zeros. So Alice simply needs to minimize

$$\max\{\vec{x}A_1, \vec{x}A_2, \ldots, \vec{x}A_m\}$$

where A_i represents the ith column of A, i.e. $A\vec{y}$ for some pure strategy $\vec{y}$.

Let $z = \max\{\vec{x}A_1, \vec{x}A_2, \ldots, \vec{x}A_m\}$. Since z is the smallest number greater than or equal to each of the $\vec{x}A_i$, we can express this as the linear program

$$\begin{aligned} & w_a = \min z \\ \forall i \quad & \vec{x}A_i - z \leq 0 \\ & \sum_i x_i = 1 \\ & \vec{x} \geq 0 \\ & z \text{ unrestricted in sign} \end{aligned}$$

4/4

This linear program gives the optimum for Alice. For Bob, the equivalent is (letting A_i now be the ith *row* of A)

$$\begin{aligned} & w_b = \max z \\ \forall i \quad & A_i\vec{y} - z \geq 0 \\ & \sum_i y_i = 1 \\ & \vec{y} \geq 0 \end{aligned}$$

2/2

c) Alice's linear program corresponds to choosing $\vec{x}$ and v to minimize v. v is constrained to be larger than all $\vec{x}A_i$, so minimizing it chooses v as the maximum of the $\vec{x}A_i$. Since we showed above that for any fixed choice of $\vec{x}$, Bob's best response to maximize $\vec{x}A\vec{y}$ is to choose a pure strategy. Thus, $\vec{y}$ will be a unit vector, and so $A\vec{y}$ will be a column of A, i.e. one of the A_i. So the maximum of the $\vec{x}A_i$ will be the payoff for a particular $\vec{x}$. Minimizing this over all feasible $\vec{x}$ will give the optimum payoff for any choice of $\vec{x}$, which is Alice's best strategy. This is what the linear program finds.

d) The dual of Alice's minimization problem is a maximization problem. It has m variables corresponding to the $\vec{x}A_i - z \leq 0$ constraints (call these variables y_i), and one variable corresponding to the $\sum_i x_i = 1$ constraint (call it z).

Since the $\vec{x}A_i - z \leq 0$ constraints is a less-than constraint, the y_i are greater than zero. Since $\sum_i x_i = 1$ is an equality constraint, z is unrestricted in sign in the dual. Since the only non-zero constraint value in the primal is the one corresponding to z, the dual's objective is to maximize z.

We multiply y on the other side of the constraint matrix, and our constraint values are zero, from the objective of the primal:

$$\forall i \quad A_i\vec{y} - z \geq 0$$

The direction of the inequality comes from the fact that the x_i are non-negative.

The sum constraint now applies through the last column rather than the last row of the constraint matrix, so $\sum_i y_i = 1$. This is an equality constraint because z is unrestricted in sign in the primal, with constraint value 1 because z appears in the objective of the primal.

So the dual is

$$\begin{aligned} & D = \max z \\ \forall i \quad & A_i\vec{y} - z \geq 0 \\ & \sum_i y_i = 1 \\ & \vec{y} \geq 0 \end{aligned}$$

2/2

which is Bob's maximization problem. By strong duality, the primal and dual have the same solution, so Alice and Bob's optimization problems have the same optimum.