CS 170 Homework 10

Due **4/12/2023, at 10:00 pm (grace period until 11:59pm)**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write "none".

2 Flow vs LP

You play a middleman in a market of m suppliers and n purchasers. The i-th supplier can supply up to $s[i]$ products, and the j-th purchaser would like to buy up to $b[j]$ products.

However, due to legislation, supplier i can only sell to a purchaser j if they are situated at most 1000 miles apart. Assume that you're given a list L of all the pairs (i, j) such that supplier i is within 1000 miles of purchaser j. Given m, n, $s[1..m]$, $b[1..n]$, and L as input, your job is to compute the maximum number of products that can be sold. The runtime of your algorithm must be polynomial in m and n.

For parts (a) and (b), assume the product is divisible—that is, it's OK to sell a fraction of a product.

(a) Show how to solve this problem, using a network flow algorithm as a subroutine. Describe the graph and explain why the output from the network flow algorithm gives a valid solution to this problem.

(b) Formulate this as a linear program. Explain why this correctly solves the problem, and the LP can be solved in polynomial time.

(c) Now let's assume you *cannot* sell a fraction of a product. In other words, the number of products sold by each supplier to each purchaser must be an integer. Which formulation would be better, network flow or linear programming? Explain your answer.

Solution:

(a) *Algorithm:* We create a bipartite graph with $m + n + 2$ nodes. Label two of the nodes as a "source" and a "sink." Label m nodes as suppliers, and n nodes as purchasers. Now, we will create the following edges:

- Create an edge from the source to supplier i with capacity $s[i]$.
- For each pair (i, j) in L, create an edge from supplier i to purchaser j with infinite capacity.
- Create an edge from purchaser j to the sink with capacity $b[j]$. We then plug this graph into our network flow solver, and take the size of the max flow as the number of dollars we can make.

Proof of correctness: We claim that the value of the max flow is precisely the maximum amount transactions we can make. To show this, we can show that a strategy of selling products corresponds exactly to a flow in this graph and vice versa.

For any flow, let $x_{i,j}$ be the amount of flow that goes from the node of supplier i to the node of purchaser j. Then, we claim a product selling strategy will sell exactly $x_{i,j}$ products from supplier i to purchaser j. This is a feasible strategy, since the flow going out of the node of supplier i is already bounded by $s[i]$ by the capacity of the edge from the source to that node, and similarly for the purchasers. Similarly, for the other direction, one can observe that any feasible selling strategy leads to a feasible flow of the same value.

(b) We define a variable $x_{i,j}$, denoting the amount of products we take from supplier i and sell to purchaser j. Then, we have the following linear program

$$\begin{aligned} \max \sum_{i=1}^{m}\sum_{j=1}^{n} x_{i,j} \\ \sum_{i=1}^{m} x_{i,j'} \le b[j'],\ \text{for all } j' \in [1,n] \\ \sum_{j=1}^{n} x_{i',j} \le s[i'],\ \text{for all } i' \in [1,m] \\ x_{i,j} = 0,\ \text{for all } (i,j) \notin L \\ x_{i,j} \ge 0,\ \text{for all } (i,j) \end{aligned}$$

The linear program has mn variables and $O(mn)$ linear inequalities, so it can be solved in time polynomial in m and n.

(c) Network flow is better. Linear programming is not guaranteed to find an integer solution (not even if one exists), so the approach in part (b) might yield a solution that would involve selling fractional products. In contrast, since all the edge capacities in our graph in part (a) are integers, the Ford-Fulkerson algorithm for max flow will find an integer solution. Thus, max flow is the better choice, because there are algorithms for that formulation that will let us find an integer solution.

3 Reduction to 3-Coloring

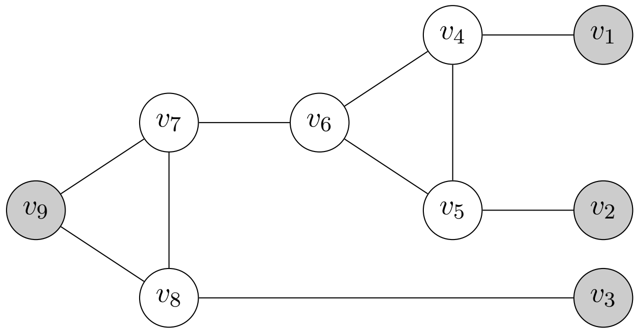
Given a graph $G = (V, E)$, a valid 3-coloring assigns each vertex in the graph a color from $\{\text{red, green, blue}\}$ such that for any edge (u, v), u and v have different colors. In the 3-coloring problem, our goal is to find a valid 3-coloring if one exists. In this problem, we will give a reduction from 3-SAT to the 3-coloring problem. Since we know that 3-SAT is NP-Hard (there is a reduction to 3-SAT from every NP problem), this will show that 3-coloring is NP-Hard (there is a reduction to 3-coloring from every NP problem).

In our reduction, the graph will start with three special vertices, labelled v_{TRUE}, v_{FALSE}, and v_{BASE}, as well as the edges $(v_{\mathsf{TRUE}}, v_{\mathsf{FALSE}})$, $(v_{\mathsf{TRUE}}, v_{\mathsf{BASE}})$, and $(v_{\mathsf{FALSE}}, v_{\mathsf{BASE}})$.

(a) For each variable x_i in a 3-SAT formula, we will create a pair of vertices labeled x_i and $\neg x_i$. How should we add edges to the graph such that in any valid 3-coloring, one of $x_i, \neg x_i$ is assigned the same color as v_{TRUE} and the other is assigned the same color as v_{FALSE}?

Hint: any vertex adjacent to v_{BASE} must have the same color as either v_{TRUE} or v_{FALSE}. Why is this?

(b) Consider the following graph, which we will call a "gadget":



Consider any valid 3-coloring of this graph that does *not* assign the color red to any of the gray vertices (v_1, v_2, v_3, v_9). Show that if v_9 is assigned the color blue, then at least one of $\{v_1, v_2, v_3\}$ is assigned the color blue.

Hint: it's easier to prove the contrapositive!

(c) We have now observed the following about the graph we are creating in the reduction:

(i) For any vertex, if we have the edges (u, v_{FALSE}) and (u, v_{BASE}) in the graph, then in any valid 3-coloring u will be assigned the same color as v_{TRUE}.

(ii) Through brute force one can also show that in a gadget, if all the following hold:

(1) All gray vertices are assigned the color green or blue.

(2) v_9 is assigned the color blue.

(3) At least one of $\{v_1, v_2, v_3\}$ is assigned the color blue.

Then there is a valid coloring for the white vertices in the gadget.

Using these observations and your answers to the previous parts, **give a reduction from 3-SAT to 3-coloring. Prove that your reduction is correct (you do not need to prove any of the observations above).**

Hint: create a new gadget per clause!

Solution:

(a) We add the edges $(x_i, \neg x_i)$, (x_i, v_{BASE}) and $(\neg x_i, v_{\mathsf{BASE}})$. Since $x_i, \neg x_i$ are both adjacent to v_{BASE} they must be assigned a different color than v_{BASE}, i.e. they both are assigned either the color of v_{TRUE} or the color of v_{FALSE}. Since we added an edge between x_i and $\neg x_i$, they can't be assigned the same color, i.e. one is assigned the same color as v_{TRUE} and one the same color as v_{FALSE}.

(b) It is easier to show the equivalent statement that if all the gray vertices on the right are assigned the color green, then the gray vertex on the left must be assigned the color green as well. Consider the triangle on the right. Since all the gray vertices are assigned green, the two right points must be assigned the colors red and blue, and so the left point in this triangle must be assigned green in any valid coloring. We can repeat this logic with the triangle on the left, to conclude that the gray vertex on the left must be assigned green in any valid coloring.

(c) Given a 3-SAT instance, we create the three special vertices and edges described in the problem statement. As in part a, we create vertices x_i and $\neg x_i$ for each variable x_i, and add the edges we gave in the answer to part a. For clause j, we add a vertex C_j and edges $(C_j, v_{\mathsf{FALSE}})$, (C_j, v_{BASE}). Lastly, for clause j we add vertices and edges to create a gadget where the three gray vertices on the right of the gadget are the vertices of three literals in the clause, and the gray vertex on the left is the vertex C_j (All white vertices in the gadget are only used in this clause's gadget).

If there is a satisfying 3-SAT assignment, then there is a valid 3-coloring in this graph as follows. Assign v_{FALSE} the color green, v_{TRUE} the color blue, and v_{BASE} the color red; assign x_i the color blue if x_i is v_{TRUE} and green if x_i is v_{FALSE} (vice-versa for $\neg x_i$). Assign each C_j the color blue. Lastly, fix any gadget. Since the 3-SAT assignment is satisfying, in each gadget at least one of the gray vertices on the right is assigned blue, so by the observation (ii) in the problem statement the gadget can be colored.

If there is a valid 3-coloring, then there is a satisfying 3-SAT assignment. By symmetry, we can assume v_{FALSE} is colored green, v_{TRUE} is colored blue, and v_{BASE} is colored red. Then for each literal where x_i is color blue, that literal is true in the satisfying assignment. By part a, we know that exactly one of $x_i, \neg x_i$ is colored blue, so this produces a valid assignment. By observation (i), we also know every node C_j must be colored blue. All literal nodes are colored green or blue, so by part b, this implies that for every clause, one of the gray literal nodes in the clause gadget is colored blue, i.e. the clause will be satisfied in the 3-SAT assignment.