CS4234: Optimization Algorithms

Problem Set 2

Due: August 25th, 11:59am

Instructions. This problem set covers material related to linear programming. Solve all the S-problems (standard). I have also included on challenge problem (i.e., an advanced problem). You do not need to hand in the A-problems (advanced).

- Start each problem on a separate page.
- Make sure your name and matric number is on each sheet of paper.
- Staple the pages together, and hand it in before class starts. Alternatively, if you submit it late, you can put it in the envelope next to my office door (and send me an e-mail). Alternatively, you may submit the problem set electronically on IVLE in the specified workbin.

Remember, that when a question asks for an algorithm, you should:

- First, given an overview of your answer. Think of this as the executive summary.
- Second, describe your algorithm in English, giving pseudocode if helpful.
- Third, give an example showing how your algorithm works. Draw a picture.

You may then give a proof of correctness, or explanation, of why your algorithm is correct, an analysis of the running time, and/or an analysis of the approximation ratio, depending on what the question is asking for.

Advice. Start the problem set early—some questions take time. Come talk to me about the questions. (Different students have different questions. Some have questions about how to write a good proof. Others need pointers of designing an algorithm. Still others want to understand the material from lecture more deeply before applying it to the problem sets.) I'm here for you to talk to.

Collaboration Policy. The submitted solution must be your own unique work. You may discuss your high-level approach and strategy with others, but you must then: (i) destroy any notes; (ii) spend 30 minutes on facebook or some other non-technical activity; (iii) write up the solution on your own; (iv) list all your collaborators. Similarly, you may use the internet to learn basic material, but do not search for answers to the problem set questions. Any similarity to other students' submissions will be treated as cheating.

Standard Problems (to be submitted)

S-1. Given a graph $G = (V, E)$, an independent set is a subset of vertices $I \subseteq V$ such that no two nodes in I are neighbors. There is a close connection between finding a minimum vertex coverand a maximum independent set. Assume that `OPT-VC` is an algorithm for finding a minimum vertex cover. Then the following is an algorithm for finding a maximum independent set:

```
OPT-IS(V, E)
    C = OPT-VC(V, E)
    I = V \ C
    return I
```

That is, if C is a minimum vertex cover, then $V \setminus C$ is a maximum independent set. Assume that `Greedy-Match-VC` is the 2-approximation algorithm for vertex cover that we studied in class. Consider the following algorithm for independent set:

```
Greedy-IS(V, E)
    C = Greedy-Match-VC(V, E)
    I = V \ C
    return I
```

Give an example graph where the approximation ratio for this algorithm is as bad as possible. That is, draw a graph where the maximum independent set is very large, while the independent set returned by the `Greedy-IS` algorithm is very small.

Notice what this demonstrates: even though it is easy to translate an *optimal* algorithm for vertex cover into an *optimal* algorithm for independent set, it is very hard to turn an *approximation* algorithm for vertex cover into an *approximation* algorithm for independent set.

S-2. Consider the following linear program:

$$\begin{aligned} \max 2x + 4y & \qquad \text{where:} \\ x - 2y &\geq 0 \\ x + y/2 &\leq 10 \\ x &\geq 0 \\ y &\geq 0 \end{aligned}$$

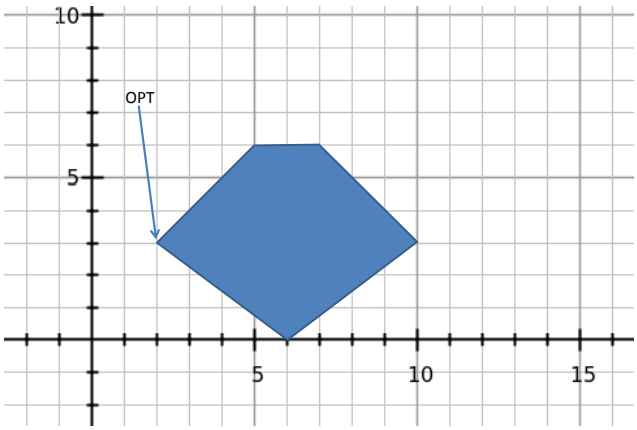
Problem 2.a. Draw a picture illustrating the feasible region and identify the optimal solution.

Problem 2.b. Give a *different* objective function (for the same constraints) that would have a different optimal solution.

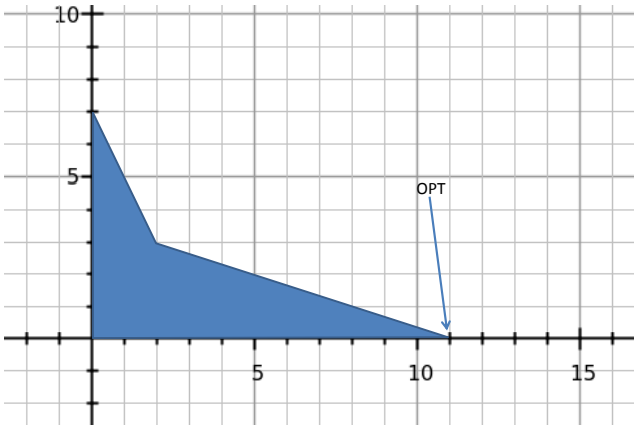
Problem 2.c. Give an *extra* constraint such that, if added to the linear program above, the feasible region is empty.

S-3. For each of the following two illustrations, give a linear program whose feasible region is exactly the dark region depicted and which is optimized at the point indicated by the arrow. If no such linear program exists, indicate *IMPOSSIBLE* and succinctly explain why. (You may assume that all the vertices on the pictures are supposed to be at integer coordinates.)

Problem 3.a.



Problem 3.b.



S-4. The two best places in the galaxy to find water are Earth and Mars (where it can be found frozen in the ice caps). Our two space stations, Argo (which orbits Saturn) and Volos (which orbits Jupiter) both need water. Argo requires 18 rockets to deliver water every month, while Volos requires 14 requires to deliver water every month.

Every month, we have the capability of launching 20 rockets of water from earth and 12 rockets of water from Mars. Each rocket of water from Earth to Argo costs 1.2 billion dollars, while a rocket of water from Earth to Volos costs 600 million dollars. Each rocket of water from Mars to Argo costs 300 million dollars, and each rocket of water from Mars to Volos costs 800 million dollars.

Problem 4.a. Write a linear program that determines the amount of water to be send from Earth and Mars to Argo and Volos. Your goal is to minimize the total cost. (You do not need to solve the linear program.) Comment on any issues that my arise in using the LP solution.

Problem 4.b. (*Optional, but recommended.*) Use the LP solver in Excel or Open Office to solve the linear program above. There are many tutorials on the web explaining how to use the LP solver. One example can be found at: `http://tinyurl.com/pu34qa8`.

What is the best solution?

S-5. (Triangles, Revisited) On the first problem set, we considered the problem of removing the minimum number of edges from a graph in order that it be triangle free. Now we consider the *weighted* version.

Assume you are given a weighted, undirected graph $G = (V, E)$ where each edge $e \in E$ is assigned weight $w(e) \geq 0$. The goal is to remove a set of edges $D \subseteq E$ with minimum weight such that the remaining graph $G = (V, E \setminus D)$ has no triangles. Give a polynomial time algorithm for solving this problem that is a 3-approximation of optimal, i.e., that returns a set of edges D that weight at most 3-times the optimal set of edges needed to produce a triangle-free graph.

Problem 5.a. Explain your algorithm.

Problem 5.b. Show that your algorithm is correct, i.e., returns a set D such that the graph $(V, E \setminus D)$ is triangle-free.

Problem 5.c. Show that your algorithm is a 3-approximation of optimal.

Advanced Problems

These do not need to be submitted. They are intended to be particularly challenging. Optionally, come talk to me about these during office hours.

A-1. Recall the classic NP-complete problem **SAT** in which you are given a formula and have to decide if it is satisfiable. Here we consider **MAX-SAT**, where the goal is to satisfy as many clauses as possible. We define the problem as follows:

- Let $v_1, v_2, \ldots, v_n$ be a set of n boolean variables (i.e., each is either *true* or *false*). The notation $\overline{v_i}$ implies the negation of v_i.
- A clause is a disjunction of variables or their negation, e.g., $(v_2 \vee \overline{v_7} \vee \overline{v_9} \vee v_{12})$. (Recall that the symbol '$\vee$' here indicates **or**.)
- The input to the problem is a sequence of clauses $C_1, C_2, \ldots, C_m$.
- The goal is to satisfy as many clauses as possible, i.e., to assign boolean values to the variables so as to maximize the number of satisfied clauses. Your algorithm your output the maximum number of satisfies clauses.

For example, one input might look like:

$$\begin{aligned}(v_1 \vee v_2 \vee \overline{v_3})\\ (v_3 \vee v_5)\\ (v_1)\\ (\overline{v_1} \vee \overline{v_3} \vee \overline{v_5})\end{aligned}$$

The answer to this problem instance is 3, because we can set $v_1 = \textit{true}$ and $v_3 = \textit{true}$, satisfying three clauses; but it is impossible to satisfy all four clauses.

The **MAX-SAT** problem is obviously NP-hard, as if you could solve **MAX-SAT**, you could also satisfy **SAT** (i.e., by checking whether it is possible to satisfy *all* the clauses.) Our goal is going to be to come up with an approximation algorithm.

We now give an integer linear program for solving **MAX-SAT**. Your job is to relax this ILP and show how to use this ILP to develp a $(1 - 1/e)$-approximation algorithm.

For the purpose of notation, define C_j^+ to be the non-negated variables in clause j, and c_j^- to be the negated variables in clause j. For example, in the clause $C_1 = (v_1 \vee v_2 \vee \overline{v_3} \vee \overline{v_4})$, we have $C_1^+ = \{v_1, v_2\}$ and $C_1^- = \{v_3, v_4\}$.

Consider the following integer linear program. The intent here is that each x_i indicates whether v_i is true or false, and each z_j indicates whether clause C_j is satisfied or not.

$$\begin{aligned}\max \left(\sum_{j=1}^{m} z_j \right) & \\ \forall j \in \{1 \ldots m\} : z_j &\leq \sum_{x_i \in C_j^+} x_i + \sum_{x_i \in C_j^-} (1 - x_i) \\ \forall j \in \{1 \ldots m\} : z_j &\in \{0, 1\} \\ \forall i \in \{1 \ldots n\} : x_i &\in \{0, 1\}\end{aligned}$$

Problem 1.a. Prove that the ILP yields a correct solution for **MAX-SAT**.

Problem 1.b. Explain how to relax this ILP to find a randomized algorithm for **MAX-SAT**. (*Hint: think of each* x_i *as a probability.*)

Problem 1.c. Prove that the expected number of satisfied clauses is at least $OPT/(1-1/e)$, i.e., it is a $(1-1/e)$ approximation algorithm. (*Hint:* Recall that for non-negative real numbers a, b, c, we know that $(abc) \leq [(a+b+c)/3]^3$. Also, recall that $(1-1/x)^x \leq 1/e$.)