CS41 Lab 11: approximation algorithms

(fall 2017)

1. **Traveling Salesman Problem.** In this exercise, you will examine and develop an approximation algorithm for the **Traveling Salesman Problem** (TSP).

   In this problem, a salesman travels the country making sales pitches. She must visit n cities and then return to her home city, all while doing so as cheaply as possible.

   The **input** is a complete graph $G = (V, E)$ along with nonnegative edge costs $\{c_e \mid e \in E\}$. A *tour* is a simple cycle $(v_{j_1}, \ldots, v_{j_n}, v_{j_1})$ that visits every vertex exactly once.[1] The goal is to **output** the minimum-cost tour.

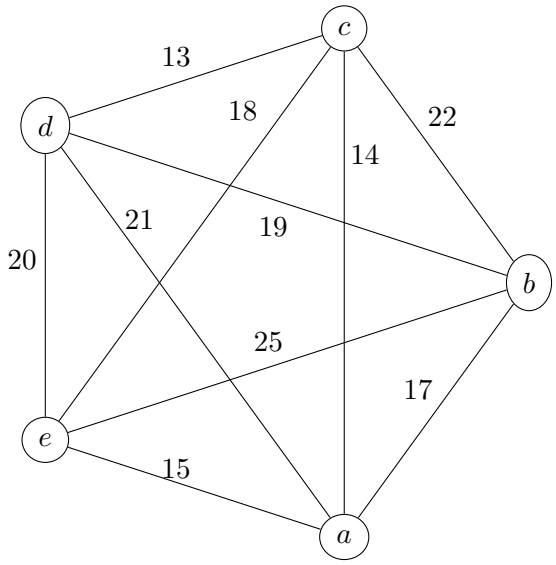
   For many TSP applications (such as when the cost is proportional to the distance between two cities), it makes sense for the edges to obey the *triangle inequality*: for every i, j, k, we have

   $$c_{(ik)} \leq c_{(ij)} + c_{(jk)}.$$

   This version is often called Metric-TSP.

   The (decision version of the) Traveling Salesman Problem is NP-Complete. For this problem, you will develop a 2-approximation algorithm for Metric-TSP.

   (a) First, to gain some intuition, consider the following graph:

   (b) *On your own* try to identify a cheap tour of the graph.

   (c) Build some more intuition by computing the minimum spanning tree (MST) of the graph. Let T be your minimum spanning tree.

[1] except for the start vertex which we visit again to complete the cycle

(d) Let OPT be the cheapest tour. Show that its cost is bounded below by the cost of the MST: $\text{cost}(T) \leq \text{cost}(OPT)$.

(e) Give an algorithm which returns a tour A which costs at most twice the cost of the MST: $\text{cost}(A) \leq 2\,\text{cost}(T)$.

(f) Conclude that your algorithm is a 2-approximation for METRIC-TSP.

2. **Solving Linear Programs.**

In this exercise, you'll optimally solve the linear program:

$$\begin{aligned} \text{Minimize} \quad & 2x_1 + 2x_2 + 3x_3 \\ \text{subject to:} \quad & x_1 + x_2 \geq 1 \\ & x_1 + x_3 \geq 1 \\ & x_2 + x_3 \geq 1 \end{aligned}$$

Note: All variables should be nonnegative. Throughout this lab writeup, we'll suppress writing these constraints for brevity.

(a) First, as a warm-up, solve the following linear program.

$$\begin{aligned} \text{Minimize} \quad & 7 + x_1 + 2x_2 \\ \text{subject to:} \quad & x_3 = 1 + 3x_1 - 4x_2 \\ & x_4 = 3 - 2x_1 + x_2 \\ & x_5 = 2 + x_1 + x_2 \end{aligned}$$

Hint: The answer should be *reallllllly* simple.

(b) The linear program in part (2a) is easy to solve for three reasons: (i) each constraint is an equality, (ii) the constants on the right-hand side of each constraint are positive, and (iii) the variables in the objective function all have positive coefficients. Our goal for the rest of the problem is to transform the LP we'd like to solve so it has all three conditions.

First, handle condition (i). It's possible to replace any inequality with an equality by adding an additional nonnegative variable called a *slack variable*. For example, if the LP has a constraint $x_1+3x_2 \geq 4$, then we could replace it with the equality $x_1+3x_2 = 4+s_1$.

Rewrite the LP by taking each constraint and replacing it with another constraint involving an equality and a new variable.

(c) Modify this LP by rearranging each inequality so that (i) there is a single variable to the left hand side, and (ii) the constants on the right hand side are all positive.

Note: the variable on the left-hand side of each constraint should appear no where else in the linear program. If it does, replace it by what's on the right hand side of the constraint. For example, taking the constraint $x_1 + 3x_2 = 4 + s_1$ from the previous subproblem, it is natural to transform it into:

$$s_1 = -4 + x_1 + 3x_2 \ .$$

If s_1 is a slack variable you just introduced, then it should appear only once in the linear program. On the other hand, the constant -4 is negative. To get a positive constant you could replace the constraint with

$$x_1 = 4 + s_1 - 3x_2$$

and replace any other occurence of x_1 in the linear program with $4 + s_1 - 3x_2$.

Modify the constraints of your linear program by making one or more transformations like the one above.

(d) **Removing negative coefficients in the objective function**. To complete the transformation of your linear program, you need to get rid of any negative coefficients in the objective function. It's possible to do this using what's called a *pivot operation*. In a pivot operation, you choose a variable in the objective function, select a constraint containing it, and make a substitution based on that constraint. For example, if your objective was to minimize $4 - x_1 + 2x_2$, and there was a constraint

$$s_1 = 3 - x_1 - x_2 \ ,$$

then you could transform the constraint into

$$x_1 = 3 - s_1 - x_2$$

and replace all other occurrences of x_1 in the LP with $3 - s_1 - x_2$.

Now, **perform pivot operations** until all variables in the objective function have positive coefficients.

(e) Solve the following minimization problem:

$$\begin{array}{ll} \text{Minimize} & 2x_1 + 2x_2 + 3x_3 \\ \text{subject to:} & x_1 + x_2 \geq 1 \\ & x_1 + x_3 \geq 1 \\ & x_2 + x_3 \geq 1 \end{array}$$

3. **(Kleinberg and Tardos, 11.9)**. Given disjoint sets A, B, C and a set $T \subseteq A \times B \times C$, a *3d matching* is a subset $M \subseteq T$ such that each element of $A \cup B \cup C$ appears at most once. The 3D-MATCHING problem is to find the largest 3d matching given sets A, B, C, and T.

   Give a deterministic (not randomized) polynomial-time 3-approximation algorithm for 3D-MATCHING.