CMSC 451:Fall 2025 Dave Mount

Practice Problems 13

Problem 1. Suppose you have a sequence of points $X = \langle x_1, \ldots, x_n \rangle$ sorted from left to right along a line. The distance between two points x_i and x_j is just their absolute difference $|x_j - x_i|$. The *bottleneck TSP problem* is the following: Find a cycle that visits each point exactly once, such that *maximum* distance traveled between any two consecutive points is as small as possible.

Consider the following *alternating heuristic* for this problem: Travel from x_1 to x_n, skipping every other point. Then return from x_n to x_1 visiting the skipped points. (An example is shown in Fig. 1. The final cost is the longest segment traversed, which is the segment of length 7 between the points at positions 9 and 16.)

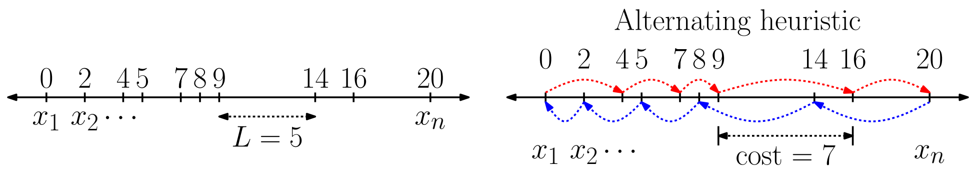


Figure 1: Alternating heuristic for bottleneck TSP.

Prove that this heuristic provides a factor-2 approximation to the bottleneck TSP problem (for points on a line). **Hint:** Let L be the maximum distance between any two consecutive points. Relate the costs of the optimum path and the heuristic path to L.

(**Note:** I believe that the alternating heuristic is actually optimal, but it is much easier to prove the factor-2 approximation bound.)

Problem 2. Given a graph $G = (V, E)$, a subset of vertices $V' \subseteq V$ is called a *dominating set* if every vertex of G is either in the set V' or is a neighbor of a vertex in V'. For example, in Fig. 2(a) the set $\{4, 8\}$ is a dominating set of size two. In the *dominating set problem* you are given a graph G and the objective is to compute a dominating set of *minimum size.* Consider the following greedy heuristic for this problem:

- Select the vertex of highest degree and add it to the dominating set. (Ties may be broken arbitrarily.)
- Remove this vertex, remove all its neighbors (since they have been covered), and remove all the edges incident to the removed vertices.
- Repeat this until no more vertices remain.

(For example, in Fig. 2(b) the heuristic would first select 7 since it has degree 6, higher than any other vertices. After deleting its neighbors, the vertices with the highest remaining degrees are 2 and 5 (each with degree 1). Suppose we take 5 for the dominating set and remove its neighbor 2. The last vertex remaining is 10. Thus, the greedy algorithm has taken 3 vertices, rather than the optimal 2.

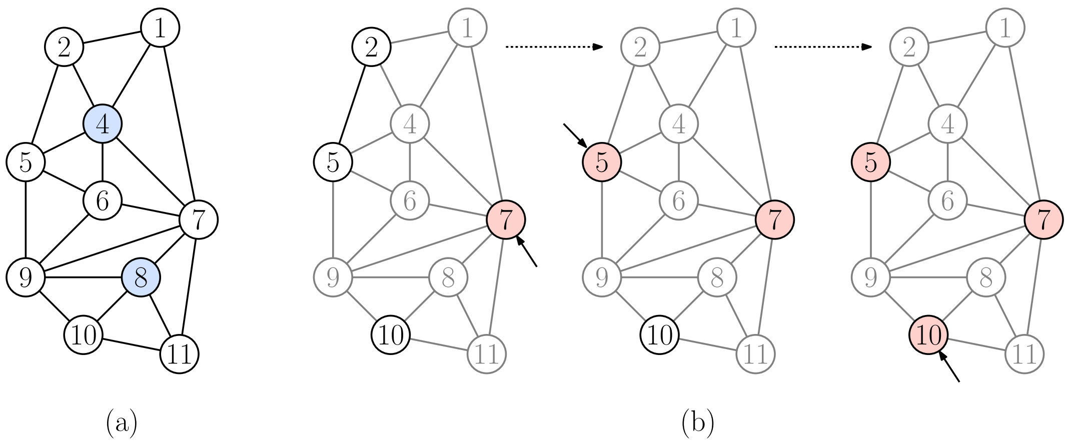


Figure 2: (a) Dominating set and (b) the greedy heuristic.

(a) Present an example to show that not only is the greedy heuristic not optimal, it can be off by a factor of at least 2 relative to the optimum solution. (That is, it builds a dominating set having at least twice as many vertices as the optimal solution.) If there are ties for the maximum degree, you may make whatever choice you like to make your counterexample work.

Hint: You might try to simulate the bad example for greedy set cover, shown in Figure 3 from Lecture 7. My example consisted of a graph with 18 vertices and an optimal dominating set of size 2.

(b) Show that the greedy heuristic achieves an approximation factor of $\ln n$, where $n = |V|$. That is, if there exists a dominating set of size k, then the greedy heuristic outputs a dominating set of size at most $k \cdot \ln n$.

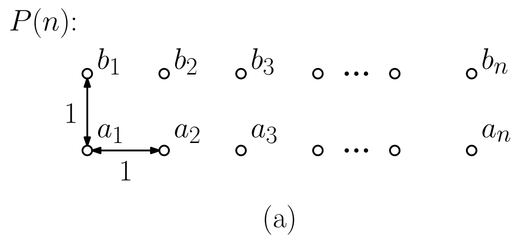
Problem 3. In this problem we will explore just how well the two TSP heuristics perform. The TSP heuristics presented in class (Twice-around and Christofides) work for a specific metric. Throughout this problem, assume that points are in the real plane $\mathbb{R}^2$, and all distances are measured by the L_1 metric, which is defined as follows. Given two points, $p = (x, y)$ and $p' = (x', y')$, the L_1 distance between these points, denoted $\|p' - p\|_1$, is defined to be the sum of the absolute differences in their x- and y-coordinates:

$$\|p' - p\|_1 \;=\; |x' - x| + |y' - y|.$$

(This is also known as the *Manhattan distance* or the *taxicab distance*, because it the distance between the points assuming that only horizontal and vertical motions are allowed.)

For any $n \geq 1$, consider the point set $P(n)$ shown in Fig. 3(a). It consists of $2n$ points $\{a_1, \ldots, a_n, b_1, \ldots, b_n\}$, where $a_i = (i, 0)$ and $b_i = (i, 1)$.

There are multiple minimum spanning trees for $P(n)$. For this problem, assume that the spanning tree is the comb-like structure shown in Fig. 3(b). (We could force this if we wanted by replacing the 1 in the y-coordinate by any value that is just a bit smaller than 1, say, 0.99999, but let's not do this to keep the math simple.)



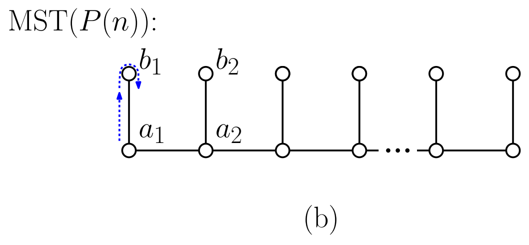


Figure 3: Approximation ratios for the metric TSP problem.

(a) For any $n \geq 1$, let TSP($P(n)$) denote the length of the optimum TSP tour for $P(n)$. Derive a formula for TSP($P(n)$) and justify its correctness.

(b) Let TA($P(n)$) denote the length of the TSP tour that results from the "twice-around" heuristic (assuming the MST shown in Fig. 3(b)). For consistency, start your twice-around tour at a_1 and go clockwise around the MST (so the first two vertices will be a_1 then b_1, as shown in Fig. 3(b)). This maximizes the length of the tour. Derive the asymptotic approximation bound for the twice-around tour on this example. That is, derive the limiting value of

$$\lim_{n\to\infty} \frac{\mathrm{TA}(P(n))}{\mathrm{TSP}(P(n))}.$$

Justify the correctness of your answer. (**Hint:** First, derive an exact formula for TA($P(n)$), and then take the ratio using the answer from (a).)

(c) Let Ch($P(n)$) denote the length of the TSP tour that results from Christofides algorithm (assuming the MST shown in Fig. 3(b)). As in (b), start your Eulerian tour at a_1 and start in a clockwise direction in order to make the tour length as long as possible. Derive the asymptotic approximation bound for the Christofides tour on this example. That is, derive the limiting value of

$$\lim_{n\to\infty} \frac{\mathrm{Ch}(P(n))}{\mathrm{TSP}(P(n))}.$$

Justify the correctness of your answer. (**Hint:** Same hint as part (b). For the sake of computing the weight of the minimum-weight perfect matching, it may simplify your calculations to assume that n is even.)

Problem 4. In this problem, we will consider an approximation algorithm for another NP-complete problem in metric spaces. To motivate this problem, suppose that you own a number of buildings in Manhattan, and you want to connect them all together using fiber-optic cables that run horizontally and vertically. To save on costs, you want to use the minimum amount of wire to connect your buildings. Throughout this problem, we will use the L_1 distance from Problem 3.

Let's represent your building as a set P of n points in the plane. Let us assume that distances are measured using the L_1 metric (see Problem 2). Given a point set P, define a *connector* to be a collection of horizontal and vertical line segments that form a connected graph containing all the points P. Define the *total weight* of a connector to be the sum of lengths of these line segments. A *minimum-weight connector* for a point set P is a connector of minimum total weight. Let MC(P) denote its weight. (Fig. 4(b) shows an example of a connector. It might the minimum connector, but I'm not sure since the problem is NP-hard).

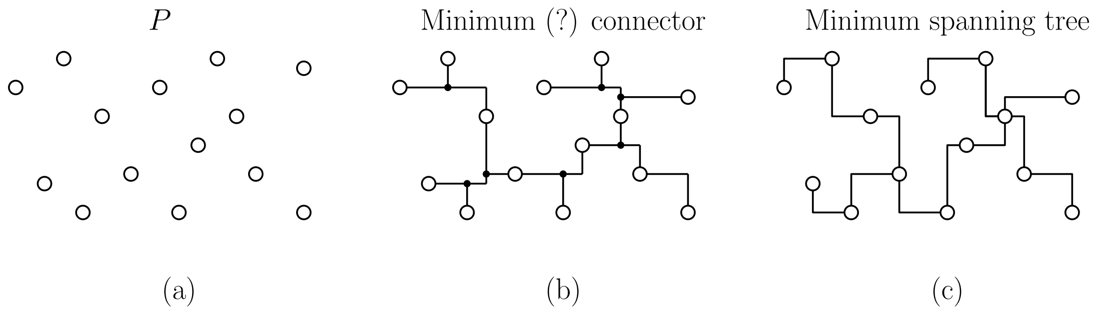


Figure 4: Approximation ratios for the metric TSP problem.

(a) Given a point set P, let MST(P) denote the weight of the minimum-spanning tree for P assuming edge weights are measured using the L_1 distance (see Fig. 4(c)). Give an (ideally small) example of a point set such that MC(P) is strictly smaller than MST(P). (**Hint:** This can be done with as few as three points. It will simplify grading if you can provide a figure, but specify the exact coordinates of the points in question and give the exact weights of the MST and MC.)

(b) The minimum-weight connector problem is NP-hard. Show that the minimum spanning tree can serve as an approximation, by showing that for any point set P in the plane

$$\frac{\text{MST}(P)}{\text{MC}(P)} \leq 2.$$

(**Hint:** The proof is similar to the proof given in class that TSP(P)/MST(P) ≤ 2.)

Problem 5. In this problem, we will consider a variant of the subset-sum (SS) approximation algorithm. Recall that we are given a set $S = \{x_1, \ldots, x_n\}$ of positive integers and a target value t. We are asked to find the subset $S' \subseteq S$ whose sum is as close as possible to t, but without going over. Recall that we can view this as trying to fill a knapsack of capacity t as full as possible, where the x_i's denote the sizes of the various objects.

Let's suppose that some designated subset $H \subseteq S$ of the items are declared to be *hazardous.* (E.g., They may be slightly radioactive.) We are given an integer m, which indicates the maximum number of hazardous items that we are allowed to take. The new optimization problem is: Given S and t as above, and given m, where $0 \leq m \leq n$, find the subset $S' \subseteq S$, containing at most m elements of H, whose sum is as close as possible to t, without going over. Let's call this *hazardous subset-sum* (HSS).

Present a polynomial-time approximation scheme (PTAS) for HSS. That is, given ε, such that $0 < \varepsilon < 1$, your algorithm should produce a valid answer z that satisfies $z \geq (1-\varepsilon)z^*$, where z^* is the optimum sum achievable by HSS. Justify the correctness of your algorithm and derive its running time.

(**Hint:** Modify the PTAS for SS that was given in class. I think that a reasonable running time to shoot for is a factor of $O(n)$ larger than the algorithm given in class, that is, $O((n^3 \log t)/\varepsilon)$.