INTRODUCTION TO APX - PRACTICAL 1

intro to approximation, TSP

Exercise one

Decide which algorithms in the following list are actually approximation algoirthms. In the case they are, also guess their approximation ratio (no proof required today). Some of the lesser known problems are explained below the list.

1. **Maximum clique in the graph:** For every edge $e \in E(G)$ I greedily test if there is a third vertex which forms a triangle with this edge. If there is, I stop looking for triangles and start looking for a fourth vertex which forms a K_4 with the previous three. I continue until the greedy proces terminates. I do this for every edge, so on the output I list the largest of the cliques found.
2. **Minimum vertex cover in a bipartite graph:** I find the maximum matching in the bipartite graph. Then I take each edge of the matching and I select the vertex of the edge with the higher degree. On the output I return the set of selected vertices.
3. **Minimum vertex cover**: I start with finding the optimum solution of the following linear relaxation for minimum vertex cover:

$$\begin{aligned} & \min \sum_{v \in V} x_v \\ \forall uv \in E: \quad & x_u + x_v \geq 1 \\ \forall v \in V: \quad & x_v \geq 0 \end{aligned}$$

   Then I round every variable x_i of value at least $1/2$ up to 1. I return all variables with value 1.
4. **Minimum dominating set:** Again, I find the optimum solution of the following linear relaxation for the problem:

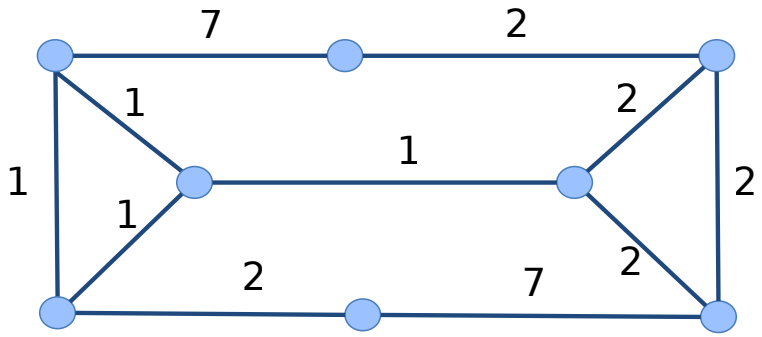
$$\begin{aligned} & \min \sum_{v \in V} x_v \\ \forall v \in V: \quad & x_v + \left(\sum_{vw \in E} x_w \right) \geq 1 \\ \forall v \in V: \quad & x_v \geq 0 \end{aligned}$$

   As before, I round every variable x_i with value at least $1/2$ up to 1 and I return the variables with value 1 as output.
5. **Knapsack problem (maximizng the number of objects in the bag with rational weights on the items:** I multiply with the least common multiple, solve with dynamic programming, and return the value of the dynamic program.

- A *vertex cover* is a set of vertices such that every edge of the graph is covered by it – at least one of the endpoints belongs to the vertex cover. We look for the vertex cover with the minimum number of vertices, or perhaps a weighted minimum.
- A *dominating set* is a set of vertices so that every vertex of the entire graph is either inside the dominating set or it has at least one of the neighbors inside the dominating set. Notice the

difference between a vertex cover and a dominating set. We minimize the number of vertices in the set, or perhaps a weighted minimum.

EXERCISE TWO Consider the following graph with distances on the edges:



1. Explain how one can compute the final metric that this graph describes.
2. What is the optimal TSP tour of this graph? What solution is found by the Christofides algorithm?
3. Suppose that I have a metric described by a graph similar to the one above. If in the complete metric I am looking for a Hamiltonian circuit, what structure I am looking for when I only consider the graph TSP?

EXERCISE THREE Why is it true that without triangle inequality nothing can be done with TSP? Let us try to answer that. The problem of *general TSP* is then a TSP problem on a vertex set V with distances between each pair of points given by an arbitrary (symmetric) length function $d\colon (V \times V) \to \mathbb{R}^+$.

Prove that if there exists a $\mathcal{O}(1)$-approximation algorithm for general TSP, then $P = NP$.

(*Hint:* Assume the contrary, take the $\mathcal{O}(1)$-approximation algorithm, and solve some NP-complete problem with it.)

EXERCISE FOUR We know from the lecture that the Christofides algorithm satisfies ALG $\leq \frac{3}{2}$OPT, where ALG is the value of the solution for the algorithm and OPT is the value of the minimum/optimum solution.

Let us refresh linear programming by proving that for Christofides algorithm, it is also true that ALG $\leq \frac{3}{2}\mathrm{OPT}_{\mathrm{LP}}$, where $\mathrm{OPT}_{\mathrm{LP}}$ is the optimum value of the following linear relaxation:

$$
\begin{aligned}
(P)\colon \qquad & \min \sum_{e \in E} c_e x_e \\
\forall v \in V\colon \qquad & \sum_{e = vx} c_e x_e = 2 \\
\forall S \subsetneq V, S \neq \emptyset\colon \qquad & \sum_{e \in E(S, V \setminus S)} c_e x_e \geq 2 \\
\forall e \in E\colon \qquad & 0 \leq x_e \leq 1
\end{aligned}
$$

The battle plan is as follows:

1. First verify that ALG $\leq \frac{3}{2}\mathrm{OPT}_{\mathrm{LP}}$ implies the original claim of ALG $\leq \frac{3}{2}$OPT.
2. Next, prove that for an optimum solution x^* of the LP (P) (that is precisely the point of value $\mathrm{OPT}_{\mathrm{LP}}$) it holds that $\frac{n-1}{n}x^*$ is a point inside the spanning tree polytope for the same graph.
3. Finally, use point 2 and finish the claim that ALG $\leq \frac{3}{2}\mathrm{OPT}_{\mathrm{LP}}$.

If you do not romember, the *spanning tree polytope* is the polytope given by these linear constraints:

$$\begin{aligned} & \sum_{e \in E} x_e = n - 1 \\ \forall S \subsetneq V, S \neq \emptyset: \quad & \sum_{e \in E(S, V \setminus S)} c_e x_e \geq 1 \\ \forall e \in E: \quad & x_e \geq 0 \end{aligned}$$

The *matching polytope* looks like this:

$$\begin{aligned} \forall v \in V: \quad & \sum_{e = vx} c_e x_e \leq 1 \\ \forall S \subsetneq V, S \neq \emptyset, |S| \text{odd}: \quad & \sum_{e \in E(S, V \setminus S)} c_e x_e \geq 1 \\ \forall e \in E: \quad & x_e \geq 0 \end{aligned}$$

Exercise five

The problem of *asymmetric TSP* tasks us with finding the minimum TSP of a directed graph where symmetry $d(u, v) = d(v, u)$ may not hold (but the triangle inequality still does).

Our first attempt could look like adapting Christofides:

1. Find an (undirected) minimum spanning tree.
2. Complete the spanning tree into an Eulerian graph using as few edges as possible.

Find an example of a directed graph with r really long edges (and the rest of edges with length 1) where step 1 succeeds at finding a minimum spanning tree and the tree can be completed to an Eulerian graph, but any such completion, even the optimal one, uses *all* the long edges, whereas the optimum uses just 1 long edge.

Such an example would show that different algorithms may be needed to approximate this problem.