# CS 6050/7050 Computational Geometry

Spring Semester, 2018

# Assignment 4: Linear Programming (Half-Plane Intersections)

**Due Day:** 8:00 p.m., Wednesday, April 4, 2018 (at the beginning of the class)

1. Let $H$ be a set of $n$ half-planes. Let $H_u$ be the subset of all upper half-planes of $H$ and let $H_l$ be the subset of all lower half-planes of $H$. Let $C_u$ be the upper envelope of the bounding lines of $H_u$ and let $C_l$ be the lower envelope of the bounding lines of $H_l$.

   Let $C$ be the common intersection of all half-planes of $H$. As discussed in class, we can compute $C$ by using $C_u$ and $C_l$. Suppose we have already computed $C_u$ and $C_l$ (for each of them, you are given a list of their vertices sorted from left to right). Design an $O(n)$ time algorithm to compute $C$ by using $C_u$ and $C_l$. **(20 points)**

2. On $n$ parallel railway tracks $n$ trains are going with the same direction and with constant speeds $v_1, v_2, \ldots, v_n$. At time $t = 0$ the trains are at positions $k_1, k_2, \ldots, k_n$. Design an $O(n \log n)$ time algorithm that detects all trains that at some moment in time are leading (i.e., in the front of all other trains).

   You may report your solution in the following manner: partition the time into many intervals $[t_i, t_{i+1}]$ and associate each interval with a train such that when $t \in [t_i, t_{i+1}]$ the associated train is leading. **(20 points)**

3. Let $S$ be a set of $k$ convex polygons. Let $n$ be the total number of vertices of all these convex polygons. Denote by $C$ the common intersection of all convex polygons of $S$.

   (a) Give an $O(n \log k)$ (not $O(n \log n)$) time algorithm to compute $C$. **(15 points)**

   (b) It is possible that $C$ is empty. Give an $O(n)$ time algorithm to determine whether $C$ is empty. **(10 points)**

4. In this exercise, we consider the following one-dimensional *one-center* problem.

   Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of points (not necessarily sorted) on a line $L$. Each point $p_i$ has a *weight* $w_i > 0$. For any point $q$ on $L$, the *weighted distance* between $q$ with any point $p_i$ of $P$ is defined as $d(q, p_i) = w_i \cdot |q - p_i|$, where $|q - p_i|$ is the distance between $q$ and $p_i$ on $L$.

   The one-center problem is to find a point $q$, called the *center*, such that the maximum weighted distance from $q$ to all points of $P$ (i.e., $\max_{1 \le i \le n} d(q, p_i)$) is minimized.

   (a) Suppose all points of $P$ have the same weights (i.e., $w_1 = w_2 = \cdots = w_n$). Design an $O(n)$ time algorithm to compute the center. **(5 points)**

   (b) Suppose the weights of the points of $P$ are not necessarily the same. Design an $O(n)$ time algorithm to compute the center. **(20 points)**

   **Hint:** Reduce the problem to finding the lowest point in a set of $O(n)$ upper half-planes and then simply apply the linear-time algorithm discussed in class.

*Remark:* In facility location theory in Operations Research, the points of $P$ are usually called *customers or clients* (the weights may represent their importance) and the center is called the *facility*. The weighted distance between the facility and each customer is called *service cost*. The one-center problem is thus to choose a location on the line $L$ to build a facility to serve all customers such that the maximum service cost is minimized.

**Total Points:** 90