Advanced Algorithms. Assignment 1

Exercise 1.
We extend the Load Balancing problem to instances that may contain both unsplittable and splittable jobs. For every splittable job, the work can be arbitrarily divided and done on different machines. Note that no discrete time slots are assumed, that is, the pieces of a splittable job may have arbitrary real lengths whose sum is the given length. As in the original problem, the goal is to minimize the makespan.

1.1. Argue why this extended Load Balancing problem with both types of jobs is also NP-complete.

1.2. Give an approximation algorithm for this extended Load Balancing problem and show that it still has an approximation ratio of 1.5. – You should not try and develop a new algorithm from scratch and even redo the entire analysis. Instead, use and adapt the already known result for Load Balancing with unsplittable jobs.

Exercise 2.
Here is a classic geometric problem, in the same application domain as the Center Selection problem:
Given n points with Cartesian coordinates (x_i, y_i) in the plane, and positive weights w_i, find one(!) point (x, y) that minimizes the weighted sum of the Euclidean distances to the given points. For formal clarity: We want to minimize

$$\sum_{i=1}^{n} w_i \cdot \sqrt{(x - x_i)^2 + (y - y_i)^2}.$$

This problem is at least not very easy to solve exactly. In the following we therefore propose a rough but rather quick and simple approximation algorithm: Instead of the Euclidean distance, take the Manhattan distance and minimize

$$\sum_{i=1}^{n} w_i \cdot (|x - x_i| + |y - y_i|).$$

(See reverse page.)

2.1. Explain how the point that minimizes the weigthed sum of Manhattan distances can be found in polynonial time. That is: Sketch an algorithm, argue why it is correct, and explain your time bound. Try to keep the time bound as low as possible.

2.2. Show that this algorithm has an approximation ratio $\sqrt{2}$. More specifically: Show that the point found by the algorithm has a weighted sum of Euclidean(!) distances that is at most $\sqrt{2}$ times the optimal sum.

Advice: In 2.1, split the problem in two "independent" one-dimensional problems, that is, work separately in x- and y-direction. To get a correct idea where to find the optimal coordinate x (and y), it might be good to study examples with very small n first. In 2.2, first write down in general terms what the approximation ratio of the proposed algorithm is, then do the necessary geometry. Finally check whether you have really achieved the goal of proving the claimed approximation ratio. (It is easy to forget this goal and stop half-way.)