Infrastructure Optimization

January 2026

1 Problem Description

The problem proposed as *infrastructure optimization* can be modelled as a *Max Coverage / Max-k-Coverage* problem. The case consists of choosing the placement of the minimum number of objects in a given area such that at least $x\%$ of the area is covered.

In our setting, there are some fixed points where the objects can be placed, and the coverage of each single object is given as input data. Our task is to identify which objects should be activated and which ones should remain deactivated.

The area that we would like to cover is divided into a square grid, and the unit of coverage is single pixels.

In this setting, we assume that:

- All pixels have the same value.
- All objects have the same cost.
- A pixel is considered covered if at least one object is covering it.

Such a problem can be applied to EV charger station placement, antenna placement, and similar infrastructure planning tasks.

1.1 Max-k-Coverage

Given n elements and m sets

$$S = \{S_1, S_2, \ldots, S_m\}$$

consisting of various configurations of the n elements, we may pick at most k sets $S' \subset S$ at maximize the number of elements covered. This translates into infrastructure optimization in the following way.

Imagine the area we wish to cover being split into n pixels. Each of the m available antennas covers a set of pixels

$$\{S_1, S_2, \ldots, S_m\}.$$

We can then choose at most k antennas such that the number of covered pixels is maximized.

The Max-k-Coverage problem is NP-hard and can be solved approximately using both greedy algorithms and Integer Linear Programming.

Note that the initial setting of the problem was a Max-k-Coverage problem, but given the nature of the proposed problem, we will not fix k *a priori*. Instead, we search simultaneously for the optimal k that provides an 95% coverage.

1.2 Data

You will be working on a synthetic dataset generated based on the above description and in the shape of the city of Copenhagen[1] as shown in Fig.1.

[1]Map Attribution: H.Loper Link: `https://commons.wikimedia.org/wiki/File:Copenhagen_location_map.png`

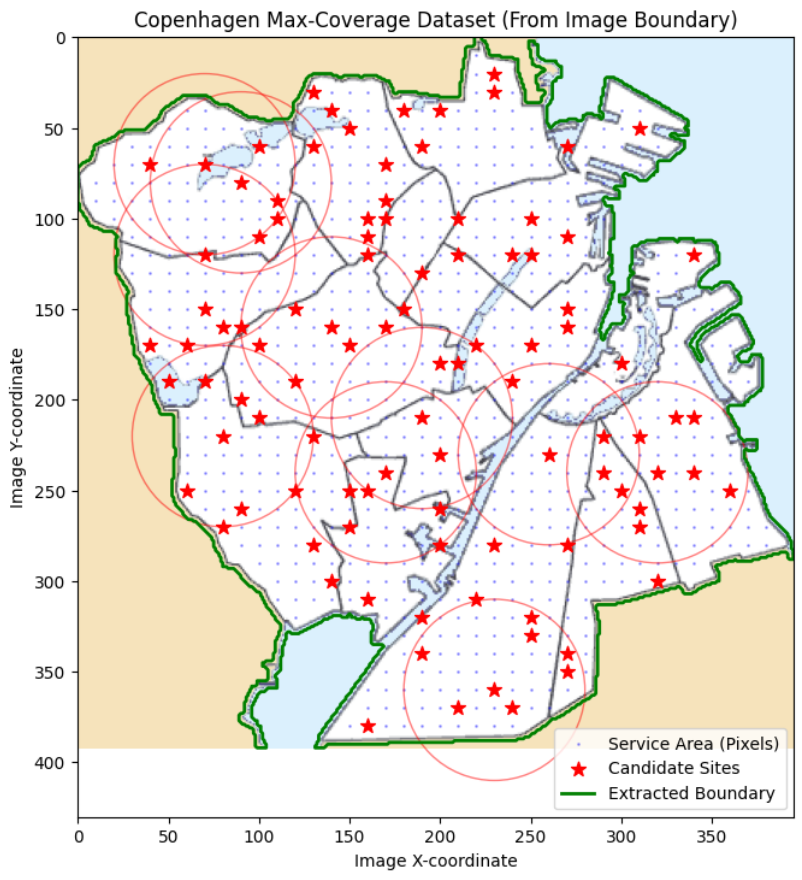

Figure 1: Visualization of dataset with 10 randomly selected sites and their coverage areas

The data set consists of 883 pixels and 100 candidate sites, at least 839 pixels need to be covered to reach 95% coverage, a pixel is considered covered if at least one selected site covers it, meaning the pixel lies within a predefined radius to the selected site.

You will be provided with "coverage.csv", a snippet is shown in Table 1. In this example, pixel 182, 206, 207, 208 are covered by site 0.

site_id	pixel_id
0	182
0	206
0	207
0	208

Table 1: An example of the coverage.csv file

You will also be provided with the pixels coordinates in "pixels.csv" and coordinates of the candidate sites in "candidates.csv" although they are not necessary.

2 Create a PUBO Model

The first step to apply a Quantum-Inspired Optimization (QIO) solution (e.g. Microsoft QIO Suite, D-Wave Quantum Annealer, Toshiba SQBM+, Fujitsu Digital Annealer) is to create a QUBO or PUBO model.

QUBO stands for *Quadratic Unconstrained Binary Optimization.* In PUBO, the *P* stands for *Polynomial.* These models can be directly fed into the corresponding solvers.

The goal of the hackathon is to create a model that tries to approximate the problem described above.

Some useful references include:

- F. Glover et al., *A Tutorial on Formulating and Using QUBO Models*, arXiv:1811.11538
- A. Lucas, *Ising formulations of many NP problems*, arXiv:1302.5843

3 Now What...

If you have a model with which you are satisfied, you can test the model with the data in Github. You can then submit your best solution using this link `https://forms.gle/ftNspDZPDaTwNdMF8`

Moreover, these models can be useful for other optimization problems and can be fully embedded into gate-based quantum computers via QAOA.