Optimizing Network Locations

Mindaugas Kepalas[1], **Julius Žilinskas**[1]

[1]Institute of Data Science and Digital Technologies, Vilnius University

Faculty of Mathematics and Informatics

Problem

Suppose that in a (planar) network $\mathcal{N}$ we want to choose N point locations $P_1, P_2, \ldots, P_N \in \mathcal{N}$. (See Fig. 1)

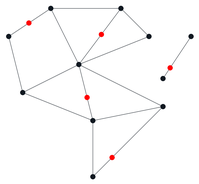

Fig. 1: Choosing 5 points (red) in a network $\mathcal{N}$ (black).

With these locations of the points, there is some associated cost (objective function) $F(P_1, P_2, \ldots, P_N)$.
The purpose of the research is to develop efficient algorithms which would find optimal point locations in a network, e.g., so that the objective function (cost) $F(P_1, P_2, \ldots, P_N)$ is minimized.

Problem Instance in Engineering

Problem formulation as in the previous section was inspired by the following problem from structural engineering [1]: suppose we are given a building foundation contour as in Fig. 2.

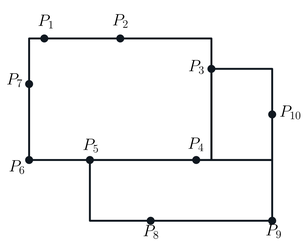

Fig. 2: Supporting a building with poles at positions $P_1, P_2, \ldots, P_{10}$.

On this contour, we choose N positions $P_1, P_2, \ldots, P_N$ of identical poles. Depending on this placement, each pole gets a load $F_i := F_i(P_1, \ldots, P_N),\ i = 1, \ldots, N$. The goal is to distribute the poles in such a way so that the load on each of them is as similar as possible, e.g., we want $F_1 \approx F_2 \approx \cdots \approx F_N$. We assume that we know the total (constant) weight W of the building; therefore the goal is $F_i \approx \frac{W}{N},\ i = 1, \ldots, N$. As an optimization problem, this could formulated as follows:

$$\min_{P_1,\ldots,P_N} \sum_{i=1}^{N} \left(F_i - \frac{W}{N}\right)^2 \quad \text{s.t.} \quad P_1, \ldots, P_N \in \mathcal{N} \qquad (1)$$

Related Geometrical Problem

For algorithm testing and analyzing purposes we present the following geometrical problem. Suppose, in the USA, we have a network of roads as in Fig. 3:

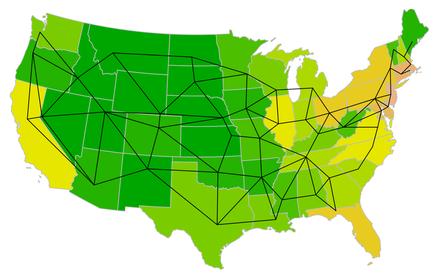

Fig. 3: Network in the USA. State color indicates how densely the state is inhabited.

In this road network, we want to open N facilities. Depending on facility positions, each of them gets a load $F_i(P_1, P_2, \ldots, P_N)$ of customers: see an example in Fig. 4. In the figure, you can see for example the smallest dot in the orange region and the largest dot in the pink region. This can be explained by looking at Fig. 3: the states contained in the orange region are sparsely inhabited, while the pink region completely contains Florida and some other densely populated states.

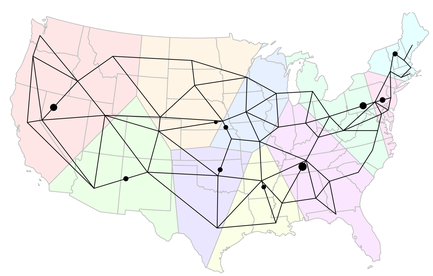

Fig. 4: Loads for facilities, opened at the positions of dots. Loads are proportional to dot sizes.

There is a bunch of computational geometry algorithms which are needed to output Fig. 4. One of them is the Voronoi diagram algorithm: for each facility, its region is actually an intersection of its Voronoi cell and USA map. Thanks to [R] sf package [2], we did not need to implement those algorithms ourselves.

First Optimization Algorithm

input :
- loss function F (depends on point locations),
- network $\mathcal{N}$,
- initial point locations $\mathcal{P} := \{P_1, \ldots, P_N\} \subset \mathcal{N}$,
- initial search window radius $r > 0$,
- shrinkage parameter $0 < \alpha \leq 1$,
- number of iterations I

output: best found point locations $\mathcal{P}^* := \{P_1^*, \ldots, P_N^*\}$

$\mathcal{P}^* := \mathcal{P}$.copy();
for $i := 1$ **to** I **do**
  $r := \alpha \cdot r$; /* shrink window radius */
  for $j := 1$ **to** N **do**
    /* 1st step: define a square with lower-left corner at $(P_j^*.x - r, P_j^*.y - r)$ and upper-right corner at $(P_j^*.x + r, P_j^*.y + r)$ */
    $\mathcal{S}_j := \texttt{GetSquare}(P_j^*, r)$;
    /* 2nd step: find the edges (or their parts) of network $\mathcal{N}$ which belong to the square $\mathcal{S}_j$ */
    $\mathcal{I}_j := \texttt{IntersectNetwork}(\mathcal{N}, \mathcal{S}_j)$;
    /* 3rd step: sample a point on edges in $\mathcal{I}_j$ */
    $P_j := \texttt{GetSamplePoint}(\mathcal{I}_j)$;
  end
  $\mathcal{P} := \{P_1, P_2, \ldots, P_N\}$;
  if $F(\mathcal{P}) < F(\mathcal{P}^*)$ **then** // if better locations found
    $\mathcal{P}^* := \mathcal{P}$.copy();
  end
end

Algorithm 1: Shrinking window random search.

You can see our very first algorithm to optimize network locations above, and an illustration of it in Fig. 5.

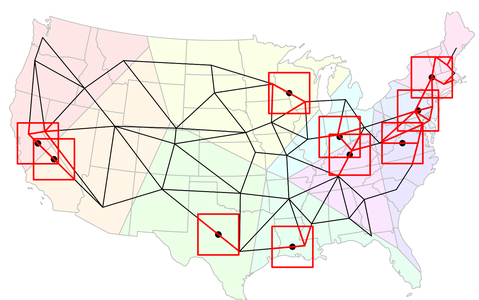

Fig. 5: **Algorithm illustration.** At every iteration, a sample of locations is drawn from within red squares around the best solution found so far. The squares are shrinking with every iteration.

Further Research

Finding optimal locations in a network is not an easy optimization problem. E.g., an usual network $\mathcal{N}$ is non-convex set, whereas most of classical optimization tools (like algorithms for Linear Programming, Quadratic Programming problems) work on convex constraint and objective functions [4]. Of course, one can use methods for non-convex optimization, e.g., Simulated Annealing or Genetic Evolution [1, 3]. However, locally, convex optimization is possible: if we restrict the points to belong to particular segments in $\mathcal{N}$, the (restricted) constraints become linear. If the objective is convex, then we obtain a rich set of efficient methods to solve the restricted optimization problem (at least locally). This is the nearest aim of the research: **to implement local optimization in network location problem**.

The next direction is to do a research on methods and tools available for mixed-integer optimization problems. Though the network is not a convex set, **it is possible to represent the constraint $P \in \mathcal{N}$ via linear constraints in higher-dimensional space, if we restrict some associated variables to be binary** (e.g., either 0 or 1). Then, if the objective function is linear or quadratic, tools for solving mixed-integer-programs (MIP) or mixed-integer-quadratic-programs (MIQP) could be used to solve network location problem optimally. Even if the function is not quadratic, it could be approximated locally by a quadratic model, and this approximation can be solved optimally.

Acknowledgements

The authors would like to mention that without **[R] sf package** [2] the geometrical problem presented in this poster would have needed to be implemented in other ways, possibly requiring much greater time resources.

Main References

[1] Rimantas Belevičius, Sergėjus Ivanikovas, Dmitrij Šešok, Saulius Valentinavičius, Julius Žilinskas. Optimal Placement of Piles in Real Grillages: Experimental Comparison of Optimization Algorithms. *Information Technology and Control*, 2011.

[2] Edzer Pebesma, Simple Features for R: Standardized Support for Spatial Vector Data, *The R Journal*, 2018.

[3] Mykel J. Kochenderfer, Tim A. Wheeler. Algorithms for optimization, MIT Press, 2019.

[4] Stephen Boyd, Lieven Vandenberghe. Convex optimization, Cambridge university press, 2004.