A Landscape function for Steiner trees

Tomas Andrade

August 3, 2020

Abstract

We consider a preprocessing heuristic for Steiner trees by means of a *landcape function*, which assigns values to all edges of the graph based on the properties of the terminals. If a given edge has a value higher than a specified threshold, the edge and associated nodes are discarded from the graph. This reduces the search space of the problem and sometimes leads to clustering, which allows for a divide-and-conquer approach. For a given (approximate) solver, we compare the performance of the algorithm with an without our preprocessing heuristic in various case studies in the context of urban network design. We find that the inclusion of our preprocessing strategy in the pipeline produces comparable results but significantly reduces computing time.

Contents

1 Intro

Consider a graph $G(V,E)$ with vertices V and edges E, and a proper subset of vertices called *terminals*, T. A Steiner tree is a tree graph which goes through all terminals and possibly other nodes in the graph, called *Steiner nodes*, minimizing a certain cost function. For concreteness we can assume that the target function is simply the length of the tree.

[picture]

In general this problem is hard – something about complexity. Simplifications arise in the planar case, where etc etc

A common heuristic is – networkx algoright about metric closure.

It is intuitively convenient to combine this or other heuristics with some type of preprocessing of the original graph which allows us to discard edges which will "obviously" not be part of the solution. Restrict to graphs embedded in $\mathbb{R}^2$ (possibly with Euclidean geometry). This gives rise to the idea of a landscape function f_T, which now describe.

The basic idea is to assign the value $f_T(x_i)$ to the node located at x_i, based on the properties of the terminals T, and to remove the node from the graph if $f_T(x_i) \geq \alpha$, where α is some threshold. There is of course an enormous freedom in the choice of f_T and α. We will explore some alternatives below and comment on possible extensions.

An interesting aspect of this procedure is the fact that the removal of certain edges can lead to *clustering* i.e. the separation of the original graph into disconnected pieces. This feature allows us to adopt a divide-and-conquer strategy in the construction of the solution, which once again can be approached in different ways depending on the problem at hand. We will implement a simple algorithm below an comment on plausible alternatives.

2 Methodology

Graph reduction and clustering

2.1 Landscape function

We now describe our choice for the landscape function. Consider a graph embedded in $\mathbb{R}^2$, and assign coordinates x_i to the nodes, while the terminals are labelled by $x_j^{(0)}$. Consider a Gaussian centred at terminal $x_j^{(0)}$, with amplitude A_j and width σ_j,

$$g_j(x) = A_j \exp\left[\frac{1}{\sigma_j^2}(x - x_j^{(0)})^2\right] \tag{1}$$

We construct the sum of the gaussians and denote it by

$$V(x) = \sum_j g_j(x) \tag{2}$$

This function is of course localized in the location where there is a terminal and decays exponentially away from it. We choose the landscape function

$$f_T(x) = \frac{\epsilon}{\epsilon + V(x)} \tag{3}$$

Note that f_T goes to 1 away from the terminals, while it approaches a small value $\sim \epsilon/A_j$ at terminal j. We depict the landscape function for a small graph in Fig 1.

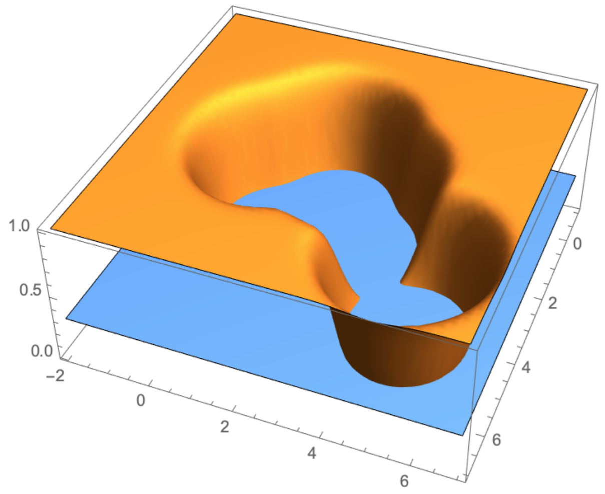

Figure 1: Landscape function for terminals at $(2,2),(3,3),(3,1),(5,5)$ and $A_0 = 1$, $\sigma_0^2 = 0.5$. We show the constant 0.3 which shows that the vertices and edges that survive are located inside the "container".

2.2 Threshold

Our choice for the landscape function naturally yields $\alpha \in (0,1)$. Intuitively, taking $\alpha = 0$ corresponds an aggressive reduction heuristic, since it leads to the removal of all edges. On the other hand, the choice $\alpha = 1$ does not remove any edges. Intermediate values remove some edges and keep others, and the details of this process in turn depend on how we choose the amplitude A_j and width σ_j of the Gaussians. Regardless of this details, it is indeed convenient to have α in a finite range for sampling purposes. For simplicity, we will assume that $\sigma_j = \sigma_0$, and $A_j = A_0$, which we expect to give good results when all terminals have similar importance (e.g. in a non-capacitated Steiner tree).

We expect there to be some value of α which is optimal for a given problem, we now describe a some approaches to find a good approximation. It is important to note that evaluating the landscape and checking which nodes and edges remain in the graph scales linearly with the size of the problem, so we ignore this step in evaluating the time complexity of the algorithm. We consider two approaches to estimate the best value of α:

- Sample various α's, estimate time complexity with given formula, and run algorithm with best value(s). Pros: captures nuances of particular cases; Cons: estimates ignore constants which can misrepresent best values.
- Parameter tuning, Bayesian optimization: [Pato?]

[Add figure]

2.3 Clustering

As mentioned above, removing certain edges can lead to clustering of the original graph, making it amenable to divide and conquer. In this circumstance, we will solve for the partial Steiner trees of each disconnected component, and connect the resulting trees with the following heuristic: i) join the smallest and largest trees by two of their terminals, by brute force computing distances between all pairs; ii) move to the next smallest tree. Note that this heuristic scales as k^2 where k is the number of terminals.

[Add figure]

2.3.1 Quotient graphs?

[Pato?]

2.4 Parallelization

The parts of the algorithm which seem amenable to parallelization are the following

- Solve the problem for the top ranked values of α
- Computation of each partial Steiner tree in case of clustering
- Compute the distance between all pair of terminals

2.5 Metrics

We will measure the performance of the algorithms by their computation (wall) time and the total length of the produced trees.

Baseline: networkx's algo without removing edges.

3 Case studies

Compete against Networkx and perhaps an algo from SteinLib

3.1 Steiner on a grid

3.2 Urban with real data