# Steiner tree problem

Antoine Huchet

February 12, 2018

## 1 Problem description

We are going to study the Steiner tree problem. The problem is defined as follows: Given a graph $G = (V, E)$, a weight function $w : e \mapsto \mathbb{R}$ assigning a weight w(e) for every $e \in V$ and a set of terminal nodes $T \subset V$. The goal is to find a subset of edges $M \subset E$ of minimal weight that connects all terminals in $G$.

This problem is NP-COMPLETE. Here we are going to be presenting and comparing different heuristic approaches to the problem.

The basic framework I will use is the one seen in class. I will first initialise a solution from a known approximation algorithm for the problem. Then I will generate new solutions to the problem by modifying my current solution. Finally I will select a few solutions from which I will iterate, optimizing the gain function. I will stop after a fixed amount of iteration.

## 2 Admissible solution

### 2.1 Check admissibility

A solution needs to connect all terminal nodes. I will define a solution to be admissible when all terminal nodes $t \in T$ are in the same connected component.

### 2.2 Gain function

My gain function will simply be the sum of all edges plus the diameter of the graph. The diameter being the maximum eccentricity over all vertices. The eccentricity of a vertex being the distance (shortest path over the weighted edges) from that vertex to the furthest away vertex.

EXPLAIN:::::::Note that an optimal solution will always have its terminal nodes as leaves.

## 3 Initialisation

The problem will be initialised by the following 2-apprximation algorithm.

For all pair of terminals, compute the shortest path between all pair of terminal vertices. Then return the minimum spanning tree of this subgraph.

This algorithm is clearly correct as terminal nodes are never disconnected from the connected component. Is is a known 2-approximation REFERENCE.

# 4 Variation

Here I will present the different ways of generating a new solution based on an already computed solutions that I studied.

## 4.1 Mutation

I will refer as mutation all variations that modify a solution by only changing it a little bit.

### 4.1.1 Adding an edge

An straight-forward mutation that could be applied to an admissible solution would be adding an edge. The solution will obviously stay admissible as no terminal nodes can be disconnected from that operation.

This operation is quite limited as it doesn't change the solution much. We might want to modify the solution more to bypass potential local optimums.

### 4.1.2 Adding a path

Another mutation that would be worth considering is adding a whole path to our current solution. In order to do so, we would randomly select two nodes from out curent solution then add a shortest path based on the graph $G$ we started with.

### 4.1.3 Removing an edge

After either of the previous two mutation, it would be interesting to try to remove an edge. This operation is more time consuming as it would entail checking if the solution is still feasible.

## 4.2 Crossover

I will refer as crossover a variation that merges from two previous solutions.

4.3 Combining both

5 Selection

5.1 Elitist selection

5.2 Elitist selection on offsprings only

5.3 Fitness proportional selection

5.4 Boltzmann selection

5.5 Threshold selection

6 Testing it all together

7 Conclusion