A PRACTICAL NETWORK LAYOUT PLANNING SYSTEM USING GEOSPATIAL DATA

Chun-Ting Wu
Microsoft
Taipei, Taiwan
chunting.wu@microsoft.com

Jason Hong
Microsoft
Bellevue, WA
jason.hong@microsoft.com

Seshagiri Cherukuri
Microsoft
Bellevue, WA
seshagiri.cherukuri@microsoft.com

Wan Jia Kun Zhu
Microsoft
Bellevue, WA
Wan.Zhu@microsoft.com

Yi Zhang
Microsoft
Bellevue, WA
zhang.yi@microsoft.com

Chi-Wei Kao
Microsoft
Taipei, Taiwan
jjjjj19980806@gapp.nthu.edu.tw

ABSTRACT

Network construction cost optimization is a classic scenario for Steiner Minimum Tree (SMT) problem and a well known NP-hard question. In this paper, we demonstrate a practical planning system for calculating network construction cost, allowing users to generate a viable solution by simply selecting entities from the map interactively. One difference from the original SMT problem is that instead of connecting entities directly, constructing network along exiting road network is more cost-efficient. In this system, for simplicity, we adopt the 2-approximation algorithm for Steiner Tinimum Tree to solve this problem and showed how we integrate roads data into the result to fit real-world scenario.

***K*eywords** Network construction · Steiner Tree

1 Introduction

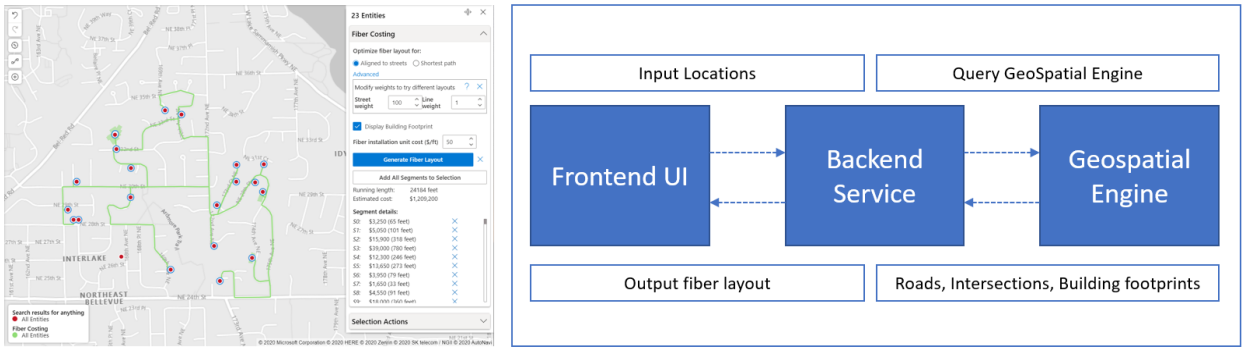
Fiber optic transmission is a fundamental technology for today's internet infrastructure. Internet service provider (ISP) companies need to construct the network in the most profitable way by minimizing the cost of fiber installation while maximizing the number of customers connected.

To optimize the fiber installation cost for a set of customer locations, the most intuitive way is to minimize the total length of the whole network. This can be modeled by a minimum spanning tree (MST) and can be easily solved. However, directly connecting the locations might not be cost-effective for a real-world scenario. For example, bodies of water could present impassible barriers and installation along roads is often cheaper as ISP companies will not need to seek permission to cross private lands or structures. In this way, fiber installation becomes a classic scenario of the well known Steiner Minimum Tree (SMT) defined as follows:

Definition 1 *Given a graph $G = (V, E)$ with edge weights d_e, and a set of required vertices $S \subseteq V$, find a subset $E' \subseteq E$ and $S \subseteq V' \subseteq V$, such that (i) the subgraph $G(V', E')$ is a spanning tree, and (ii) $\sum_{e \subseteq E'} d_e$ is minimized.*

However, SMT is a NP-complete problem, and unsolvable in polynomial time, unless P = NP. It's still an open question and on-going research continues to try to approximate the optimal solution in polynomial time.

In this work, for simplicity, we adopted the 2-approximation method for solving SMT, and proposed a way to incorporate urban road data into SMT computation process. We attempted to optimize the computational performance, aiming to provide our users with a responsive and interactive experience when using our system to plan their fiber construction layout.



(a) (b)

Figure 1: **System Overview** (a) Frontend interface to let users select input locations and view output result, (b) Architecture of our fiber layout planning system

2 Methods

2.1 System Overview

In this section, we give an overview of our system's architecture. The system consists of three parts as shown in Fig.1 (b), the first part is an interactive user interface Fig.1 (a) which allows users to select input locations and view the output layout and cost of fiber installation on a map view.

The second part leverages our existing geospatial engine, which can return nearby entities given an input coordinate with very low latency. It hosts building footprint and road data for runtime queries.

The last part is the back-end service that performs the main calculation. We created a Web API that takes a list of locations as input and returns the layout of connecting fiber layout along with the cost for each cable segment as output. In the following sections, we will describehow we integrate road data to a classic heuristic SMT algorithm to compute the network layout.

2.2 SMT 2-Approximation Algorithm

To solve this problem, we adopt the 2-approximation heuristic proposed by Kuo [1], which means the solutions are at most twice the cost of the Steiner Minimum Tree. The procedure is described as follows. Consider a connected, undirected graph $G = (V, E, d)$ and a set $S \subseteq V$, where V is the set of vertices in G, d is a cost function, and vertices in S are required in the output results. First we construct a complete graph G_1 from S with edges E_S, and weight d_S This algorithm first finds the MST T_1 on G_1, and then construct G_S by replacing tree edge in T_1 by its corresponding shortest path on G. Finally, we compute MST(T_S) again on G_S to remove cycles and delete optional Steiner points from the resulting Steiner Tree.

Algorithm 1: 2-Approximation algorithm for Steiner Tree problem [1]

1 Construct the graph $G_1 = (S, E_S, d_S)$ from G and S;
2 Find the MST T_1 of G_1;
3 Construct G_S by replacing edges in T_1 by its corresponding shortest path in G;
4 Find the MST T_S of G_S;
5 Construct a Steiner Tree T_H Deleting edges in T_S if the leaves are optional points;

2.3 Incorporating Roads

As we described before, installing fiber cable along streets is preferred to directly connecting the entities. Therefore, extending the original SMT problem, we need to construct the Steiner Tree on top of a road network. The definition of problem becomes:

Definition 2 *Given a set of required vertices S and road network R with road weight d_r and intersections I, find a subset of road $R' \subseteq R$ and $S \subseteq V \subseteq (S \cup I)$, such that (i) the graph $G(V, R')$ is a spanning tree, and (ii) $\sum_{r \subseteq R'} d_r$ is minimized.*

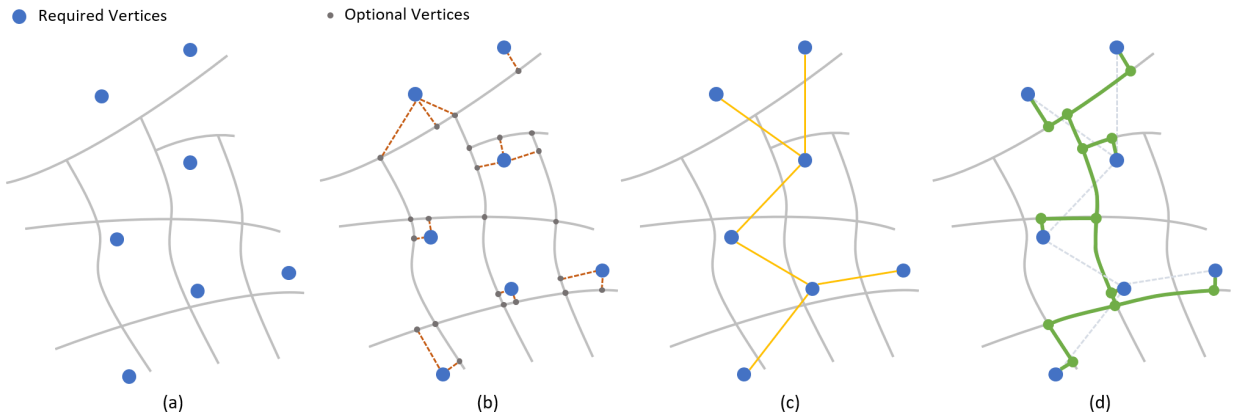


Figure 2: **Schematic of computing SMTs on road network** (a) Given a set of required point and a road network (b) Adding additional path from entities to their nearby roads and inserting optional points on intersections (c) Calculating MST on required points (d) Replacing MST edges by their shortest paths and removing cycles

Inspired by the heuristic algorithm we described in the previous section, we propose a way to deal with road network data in the process of computing SMTs. As shown in Figure 2, (a) Given a set of required points S, and a road network, and the cost of constructing fiber along roads d_r proportional to the road length. (b) Next, to connect points to roads, we add additional paths that connect required entities to its nearby roads. At each of the intersection sites, we break the roads into segments E, and insert additional vertices to become the optional Steiner points P to let $V = S + P$. We now have built the complete graph $G = (V, E, d)$. To compute SMT on G, (c) we first find the MST T_1 for S and replace every edge $e = (u, v)$ in T_1 by its corresponding shortest path from u to v on G, (d) and follow step 4 and 5 in algorithm 1 to remove cycles. Finally the Steiner Tree T_s of S on road network are constructed.

Algorithm 2: Computing SMT on road network [1]

1 Given required entities S and a Road Network R;
2 Constructing complete graph G_1 from S, and find its MST T_1;
3 For each of the entities, adding additional paths to nearby roads to R;
4 At each of the intersection sites, inserting optional vertices P to let $V = S + P$;
5 Breaking roads R into edge set E with cost d, and constructing graph $G = (V, E, d)$;
6 Constructing graph G_S by replacing edges in T_1 by corresponding shortest paths in G;
7 Finding the MST T_S on G_S, and deleting edges of optional points to form output Steiner Tree T_H;

2.4 Fiber Layout Computation API

This section describes the workflow of the fiber layout calculation in our backend API service. This API accepts requests containing a list of interested entities with their names, latitudes, longitudes, and the fiber construction cost factor of different types of connections. Before computing the network layout, we first conflate the entities using building footprint polygons [2]. In dense cities, multiple entities may share the same building and can be grouped into a single record for further computation. We call these resulting points V.

After, we construct a complete graph G_1 from V. Note that all the vertices in G_1 are required, and d is set to be the Euclidean distance between two points. and we follow the algorithm in the last section to compute its minimum spanning tree T_1 as shown in Figure 3(a). Continuing, for each point v in G_1 we retrieve the nearby roads segments and road intersections from the geospatial engine. We find each v's shortest path to nearby roads and add them to G, also inserting additional points to the corresponding road. After that, roads are split into line segments and vertices. We insert all of them into graph G, so that vertices become the optional Steiner points and the edge weight are set to be the distance multiplied by a cost factor f_{cost} to make the roads path more preferable for Steiner Tree computation.

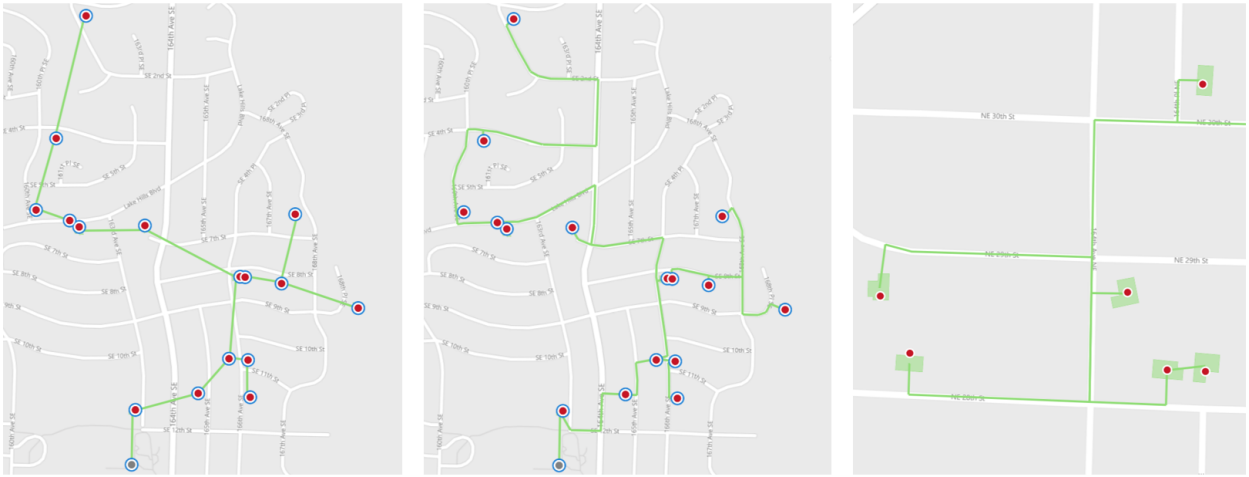


Figure 3: **Sample output from the system** (a) MST of required entities, (b) Fiber network layout using existing road network (c) Close-up of an output fiber network

Next, following the algorithm we describe in the last section, we compute MST T_1 for S and replace each of the edges by their corresponding shortest path on G_1. Finally, by removing duplicated edges and cycles, we can get the resulting Steiner Tree T_S as shown in Figure3(b). Note that here we replaced steps 7 in Algorithm 2 by simply removing cycles and duplicated edges from the graph, as from our experiments, the results are close and the total computation time can be reduced. Consider a set of input entities S_{Input}, the complete procedure is summarized in Algorithm 3.

Algorithm 3: Computing Fiber layout Using Road Network and Building Footprints

foreach *Input entities* $s_i \in S_{Input}$ **do**
  Query Geospatial Engine for building footprint f_i;
  if *Query Successful* **then**
    if $f_i \notin S$ **then**
      $S.insert(f_i)$
    end
  else
    $S.insert(s_i)$
  end
end
Construct the complete graph $G_1 = (S, E_S, d_S)$ from S;
Find the MST T_1 of G_1;
$G \leftarrow G_1$;
foreach *Input entities* $s_i \in S$ **do**
  Query Geospatial Engine for nearby roads R;
  foreach *road* $r_i \in R$ **do**
    $r_i \leftarrow DouglasPeucker(r_i)$;
    foreach *segment* $seg_i \in r_i$ **do**
      $w_i \leftarrow Length(seg_i) * f_{cost}$;
      $G.insert(seg_i, w_i)$
    end
  end
end
Construct G_S by replacing edges in T_1 by its corresponding shortest path in G;
Form a Steiner Tree T_H by removing cycles and duplicated edges in G_S.

Table 1: Execution time versus number of input entities.

Entities	100	200	300	400
Execution Time (sec)	2.54	3.70	5.31	9.81

2.5 Performance Optimization

To provide the service in real-time, we optimized the computational performance in two ways. (1) The raw input entities are clustered by DBSCAN [3] to reduce redundant calls to geospatial engine. (2) In our original road data, road geometries are lists of points to represent connected short line segments, however, some curved segments are too dense and can hurt the performance. As a result, we process each road by the Douglas-Peucker algorithm with distance tolerance $\epsilon = 10^{-4}$ to simplify their geometry before adding them to graph G. Figure 3(c) shows that in this way, the system not only can compute more efficiently but also get cleaner road geometries in the output Steiner Tree results. Besides, we also made the step of computing the shortest path parallel to leverage our multi-core machine, and the overall execution time can be further reduced.

In production, our backend API service is hosted on Azure Kubernetes Service (AKS) cluster, with Standard D16v3 virtual machines [4]. The deployed pod is configured with 3 CPU cores and 8G memory. The actual execution times are summarized in Table 1, and shows that our system can compute the fiber network for hundreds of entities within seconds.

3 Conclusion

In this paper, we demonstrate an interactive system for optimizing fiber network construction costs. Our proposed procedure successfully integrates road data to the classic Steiner Tree problem. With our optimization engineering work, our system can process hundreds of entities in seconds, which can provide our users with a smooth user experience, and a new way to do their work more efficiently. Although our proposed method is designed specifically for our scenario, the idea of incorporating additional knowledge or constraint in computing Steiner Tree is applicable to a large number of practical situations where the task is minimizing the cost of connecting important locations beyond simple direct connections, like storm water drain layout in municipal utility planning, VLSI circuit design and traffic or computer network designs.

References

[1] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for steiner trees. *Acta Inf.*, 15(2):141–145, June 1981.

[2] Microsoft. *US Building Footprint*. `https://github.com/Microsoft/USBuildingFootprints`.

[3] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[4] Microsoft. *Azure Virtual Machine Dv3 series*. `https://docs.microsoft.com/en-us/azure/virtual-machines/dv3-dsv3-series`.