Connecting Two Trees with Optimal Routing Cost *

Mong-Jen Kao[¶] Bastian Katz[§] Marcus Krug[§] D.T. Lee[¶†] Martin Nöllenburg[§‡] Ignaz Rutter[§]
Dorothea Wagner[§]

Abstract

We study the problem of finding the optimal connection between two disconnected vertex-weighted trees. We are given a distance function on the vertices and seek to minimize the routing cost of the tree resulting from adding one single edge between the two trees. The routing cost is defined as the sum of the weighted distances between all pairs of vertices in the induced tree-metric. The problem arises when augmenting and/or repairing communication networks or infrastructure networks.

We present an asymptotically optimal quadratic-time algorithm for the general case and show that the problem can be solved more efficiently for the Euclidean metric, when vertices are mapped to points in the plane, as well as for compactly representable graph metrics.

1 Introduction

In the construction of communication and infrastructure networks we often have to find a reasonable balance between the cost for establishing the links between the vertices in the network and the performance of the network in terms of various quality measures, such as routing cost, connectivity and diameter. While the cost should be minimized and increases with each established link, the performance of the network should be maximized and typically improves when more links are added. This tradeoff can be formalized in different ways. However, motivated by practical applications of this problem it is quite common to assume that we are given a limited budget for the construction cost and wish to optimize the performance of the network subject to this constraint.

The *Optimal Network Problem*, which has been introduced by Scott [8], addresses the problem of optimizing the *routing cost* of a network, defined by the sum of the shortest paths between all pairs of vertices in the graph. Due to its importance for communication networks, this problem has received considerable attention, among others by Dionne and Florian [1] and Wong [10].

If the budget for establishing the links in a network is rather tight, a tree is often the only affordable infrastructure. However, Johnson et al. [6] prove that the optimal network problem is NP-complete, even if all edges have the same length and the network must be a tree. This problem is also called the *Minimum Routing Cost Spanning Tree Problem (MRCST)*. More recently, Wu et al. presented an FPTAS for this problem [11] and Fischetti et al. [3] studied exact algorithms for computing the minimum routing cost spanning tree.

We consider the related problem of connecting a disconnected tree-network by adding a missing edge or repairing a broken network by removing the broken edge and establishing a new link.

1.1 Problem Definition

More formally, we consider the following problem. We are given a set of vertices V as well as some distance function d on V such that $d(v,v) = 0$ and $d(u,v) = d(v,u) \geq 0$ for all vertices $u, v \in V$. Further, we are given a partition of $V = V_1 \cup V_2$ and two disjoint trees $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ on V_1 and V_2, respectively. We write $n = |V|$, $m = |E|$, $n_i = |V_i|$ and $m_i = |E_i|$ for $i \in \{1, 2\}$. For each tree T on a subset $V' \subseteq V$, we consider the tree metric d_T, which is defined on V' such that the distance between $u, v \in V'$ is equal to the sum of the distances on the uniquely defined path between u and v. Further, we assume each vertex $v \in V$ has some non-negative demand $c(v)$. For $V' \subseteq V$ we write $c(V') := \sum_{v \in V'} c(v)$ as a shorthand. We define the *weighted routing cost* of T as

$$\mathrm{rc}(T) = \sum_{(u,v) \in V \times V} c(u) \cdot c(v) \cdot d_T(u,v) \; .$$

The demands can be considered to be an indicator for the importance of the vertices in the network. The amount of traffic between two vertices in the network is scaled by the product of the demands modeling the fact that important vertices are usually involved in more traffic than less important vertices and that the traffic

*Supported by NSC-DFG Projects NSC98-2221-E-001-007-MY3 and WA 654/18.

†Also supported by the Institute of Information Science, Academia Sinica, Taiwan.

‡Supported by the Concept for the Future of KIT under project YIG 10-209 within the framework of the German Excellence Initiative

§Faculty of Informatics, Karlsruhe Institute of Technology (KIT), firstname.lastname@kit.edu

¶Dep. of Computer Science and Information Engineering, National Taiwan University, Taiwan, d97021@csie.ntu.edu.tw, dtlee@csie.ntu.edu.tw

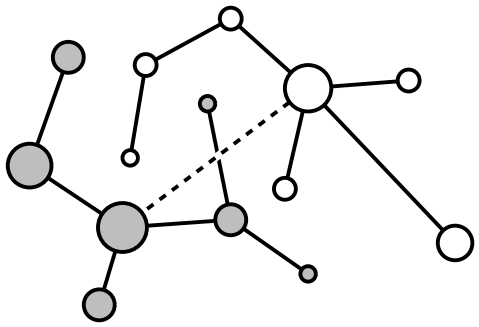

Figure 1: Two vertex-weighted trees and best connection (dashed).

between two important vertices is usually larger than that between an important and a less important vertex.

The *Optimal Routing Cost Augmentation Problem* is to find vertices $u \in V_1$ and $v \in V_2$ such that the routing cost of the tree $T_{uv} = (V, E_1 \cup E_2 \cup \{uv\})$ is minimized; see Figure 1 for an example.

For the *Optimal Routing Cost Replacement Problem* we are additionally given a pair of vertices $u \in V_1$ and $v \in V_2$ that should be excluded from the solution (since the corresponding edge must be replaced). We can solve this problem by simultaneously computing the best and second-best solution. If the best solution coincides with uv, then we return the second-best solution.

1.2 Contribution

In Section 2 we consider general distance functions. We show that both the optimal routing cost augmentation problem and the optimal routing cost replacement problem can be solved in $\Theta(n_1 \cdot n_2)$ time, which is optimal. In Section 3 we assume that vertices are points in the plane and that the distance between points is equal to the Euclidean distance. We show that both the augmentation problem and the replacement problem can be solved more efficiently in $\mathcal{O}(n \log \min\{n_1, n_2\})$ time by querying the additively weighted Voronoi diagram of a suitably chosen set of points. We adapt this idea to general graph metrics by computing the additively weighted Voronoi diagram on graphs in Section 4. This yields an $\mathcal{O}(n \log n)$-time algorithm for compactly representable metrics, i.e., metrics that are representable as sparse graphs. We conclude with some remarks and open problems in Section 5.

2 An optimal algorithm for the general case

In this section, we consider general distance functions on the vertex set. We show that the problem can be solved in $\Theta(n_1 \cdot n_2)$ time, which is optimal. For ease of notation we write $C_1 = c(V_1)$ and $C_2 = c(V_2)$ for the total demand in T_1 and T_2, respectively. Given two vertices $u \in V_1$ and $v \in V_2$, the routing cost of the tree T_{uv} resulting from joining T_1 and T_2 by the edge uv is given by

$$\begin{aligned} rc(T_{uv}) = rc(T_1) + rc(T_2) \\ + C_2 \cdot \sum_{u' \in V_1} c(u') \cdot d_{T_1}(u', u) \\ + C_1 \cdot \sum_{v' \in V_2} c(v') \cdot d_{T_2}(v, v') \\ + C_1 \cdot C_2 \cdot d(u, v) \ . \end{aligned} \tag{1}$$

It is composed of the routing cost inside the subtrees T_1 and T_2 of T_{uv}, respectively, and the routing cost effected by the shortest paths using the edge uv between the two trees. Since the total sum of demands for these paths equals $C_1 \cdot C_2$, the edge uv contributes a total amount of $C_1 \cdot C_2 \cdot d(u, v)$ to the routing cost. Furthermore, each shortest path starting at u' in T_1 and ending at u can be extended to a shortest path ending at some vertex v' in T_2. Hence, each shortest path of this kind contributes its length, weighted by its demand $c(u')$ and the total sum of the demands C_2 in T_2, to the routing cost. The situation is symmetrical for the paths starting in T_2 and ending at v.

Since the routing costs of T_1 and T_2 do not depend on the choice of the link between the two trees, our problem is equivalent to minimizing the remaining summands in equation (1).

We define the weight of a vertex $u \in V_1$, denoted by $w(u)$, as the sum of lengths of all shortest paths starting at $u' \in V_1$ and ending at u, weighted by the demand of u', i.e.,

$$w(u) = \sum_{u' \in V_1} c(u') \cdot d_{T_1}(u', u) \ .$$

We define the weight of a vertex $v \in V_2$, denoted by $w(v)$, analogously. Hence, we seek to minimize the term

$$rc'(T_{uv}) = C_2 w(u) + C_1 w(v) + C_1 \cdot C_2 \cdot d(u, v) \tag{2}$$

over all possible combinations of $u \in V_1$ and $v \in V_2$.

The weights of the trees can be computed in linear time as follows. First we compute the total demands in T_1 and T_2, respectively. We compute the weights in T_1 by rooting the tree in some vertex r and performing one bottom-up pass over the tree, followed by a top-down pass. For a vertex u in T_1 we denote the subtree rooted in u by T_u.

In the bottom-up pass, we compute two values for each vertex $u \in V_1$: the total demand $\gamma(u)$ of the vertices in T_u, and the sum $\lambda(u)$ of the shortest paths starting at some vertex u' in T_u and ending at u, weighted by the demand of u', i.e.,

$$\gamma(u) = \sum_{u' \in V(T_u)} c(u')$$

and

$$\lambda(u) = \sum_{u' \in V(T_u)} c(u')d(u', u) \ .$$

For a vertex u with children $u_1, \ldots, u_k$ these values can be computed in linear time as

$$\gamma(u) = c(u) + \sum_{i=1}^{k} \gamma(u_i)$$

and

$$\lambda(u) = \sum_{i=1}^{k} \big(\lambda(u_i) + \gamma(u_i) \cdot d(u_i, u)\big) \ ,$$

respectively. In the top-down pass, we compute the weight for each vertex $v \in V_1$. For the root r this weight is equal to $\lambda(r)$. For a vertex v with father $u \in V_1$ the weight can be computed by

$$w(v) = w(u) + (C_1 - 2\gamma(v))d(u, v) \ .$$

This equation is due to the fact that the weight of v is obtained from the weight of u by removing the demand $\gamma(v)$ in the subtree of v from the edge uv and adding the remaining demand $C_1 - \gamma(v)$ to the edge uv. For T_2 we proceed analogously.

Having this, we can compute the best and second-best connection between the two trees by enumerating all possible pairs uv such that $u \in V_1$ and $v \in V_2$, which yields a total running time of $\mathcal{O}(n_1 \cdot n_2)$. Note, that the described algorithm only finds the best or second-best solution, but does not compute the routing cost of this solution. If we have no restriction on the distance between the vertices, however, the algorithm is optimal.

Theorem 1 *The optimal routing cost augmentation problem and the optimal routing cost replacement problem can be solved in $\mathcal{O}(n_1 \cdot n_2)$ time for general distance function. This is optimal in the algebraic decision tree model.*

Proof. We have already outlined the algorithm and argued why it runs within the stated time complexity. It remains to show the lower bound on the running time. For this, we assume that we are given a set of integers $a_1, \ldots, a_N$. We construct an instance of the optimal routing cost augmentation problem such that finding the minimum routing cost connection between the two trees is equivalent to the minimum of the numbers $a_1, \ldots, a_N$. For this problem, we need at least $N - 1$ comparisons in the algebraic decision tree model of computation.

Let $N = n_1 n_2$ be any factorization of N and let V be a set of $n_1 + n_2$ vertices. Further, let $V_1, V_2 \subseteq V$ be a partition of V such that $|V_1| = n_1$ and $|V_2| = n_2$ and let T_1 and T_2 be two arbitrary trees on V_1 and V_2, respectively. We set the distance between two vertices in the same tree equal to one. Let $x : V_1 \times V_2 \to \{a_1, \ldots, a_N\}$ be a bijective mapping between the pairs of vertices in V_1 and V_2 and the numbers a_i. Then we choose the remaining distances as follows. Let W_1 and W_2 be the maximum weights of the vertices in T_1 and T_2, respectively. For $u \in V_1$ and $v \in V_2$ we define

$$d_0(u, v) = C_2 W_1 + C_1 W_2 - C_2 w(u) + C_1 w(v) \ .$$

Further, we set

$$d(u, v) = \frac{d_0(u, v) + x(u, v)}{C_1 C_2} \ .$$

Then $\mathrm{rc}'(T_{uv}) = C_2 W_1 + C_1 W_2 + x(u, v)$. For both the augmentation and the replacement problem we need to compute the minimum routing cost solution. However, minimizing the routing cost for the given instance is equivalent to computing the minimum over the values $x(u, v)$ for $u \in V_1$ and $v \in V_2$. Hence, in the algebraic decision tree model of computation, we need at least $n_1 \cdot n_2 - 1$ comparisons, which completes the proof. □

3 An Efficient Algorithm for the Euclidean Metric

The proof for the lower bound in the previous section crucially exploits the fact that we can choose distances between the vertices in an arbitrary fashion. If this is not the case, we can come up with more efficient algorithms.

In this section we consider the case that vertices are points in the plane and that the considered metric d is the Euclidean metric. In this case, we can compute the best connection between two trees in $\mathcal{O}((n_1 + n_2) \log \min\{n_1, n_2\})$ time. Throughout the section, we do not distinguish between vertices and points.

Theorem 2 *The optimal augmentation problem for the Euclidean metric can be solved in $\mathcal{O}((n_1 + n_2) \log \min\{n_1, n_2\})$ time.*

Proof. Without loss of generality we may assume that $n_2 \leq n_1$. Let $\sigma : \mathbb{R}^2 \to \mathbb{R}^2$ be an isotropic scaling with scale factor $s = C_1 \cdot C_2$, i.e., σ scales distances by a factor s and we thus have

$$d(\sigma u, \sigma v) = C_1 \cdot C_2 \cdot d(u, v) \ . \tag{3}$$

Let σV_1 and σV_2 denote the scaled sets of points.

For $x \in \mathbb{R}^2$ and $\widetilde{v} \in \sigma V_2$ we define a new distance function, defined by $d_+(x, \widetilde{v}) := d(x, \widetilde{v}) + C_1 \cdot w(v)$, where w is defined as in the previous section. The *additively weighted Voronoi cell of* $\widetilde{v}$ is the locus of points

$$\{x \in \mathbb{R}^2 \mid \forall \widetilde{u} \in \sigma V_2 \setminus \{\widetilde{v}\} : d_+(x, \widetilde{v}) < d_+(x, \widetilde{u})\} \tag{4}$$

The additively weighted Voronoi diagram $\mathcal{V}$ defined by d_+ consists of the additively weighted Voronoi cells of the points in σV_2 and can be computed in $\mathcal{O}(n_2 \log n_2)$ time [4].

For each point $u \in V_1$, we locate the nearest neighbor σv of σu in $\mathcal{V}$ using an algorithm with $\mathcal{O}(\log n_2)$ query time described by Kirkpatrick [7]. Then σv satisfies

$$d_+(\sigma u, \sigma v) = \min_{v' \in V_2} d_+(\sigma u, \sigma v') \tag{5}$$

and we have

$$\begin{aligned} d_+(\sigma u, \sigma v) &= d(\sigma u, \sigma v) + C_1 \cdot w(v) && (6)\\ &= C_1 \cdot C_2 \cdot d(u, v) + C_1 \cdot w(v)\ . && (7) \end{aligned}$$

Hence, $v \in V_2$ is the best endpoint of an edge starting at $u \in V_1$ with respect to routing cost. Minimizing $C_2 \cdot w(u) + d_+(\sigma u, \sigma v)$ over all vertices $\sigma u \in V_1$ and their respective nearest neighbor $\sigma v \in V_2$ will thus minimize the overall routing cost. The resulting overall running time is $\mathcal{O}(n_1 \log n_2 + n_2 \log n_2)$. □

In order to solve the replacement problem, we also need to compute the second-best solution. We can do this as follows. Let $u^* \in V_1$ and $v^* \in V_2$ be the best solution computed by the algorithm above. This algorithm can trivially be modified to simultaneously compute

$$\min_{u \in V_1 \setminus \{u^*\}, v \in V_2} \mathrm{rc}'(T_{uv})$$

in the same time complexity. By additionally computing the Voronoi diagram only for the points in $V_2 \setminus \{v^*\}$ and repeating the algorithm on this instance, we can also compute

$$\min_{u \in V_1, v \in V_2 \setminus \{v^*\}} \mathrm{rc}'(T_{uv})\ .$$

Clearly, the second-best solution is either of the two. Hence, we have the following corollary.

Corollary 1 *The optimal routing cost replacement problem for the Euclidean metric can be solved in time* $\mathcal{O}((n_1 + n_2) \log n_2)$.

Note that the same approach can also be used in a planar setting, i.e., when the newly introduced edge connecting the two trees may not intersect any other edge of the two trees. In this case we compute an additively weighted constrained Voronoi diagram, which can be done by adapting Fortune's sweepline algorithm [4] with $\mathcal{O}(n \log n)$ running time. In a constrained Voronoi diagram, we are given an additional set of line segments representing obstacles. Whenever the straight line connecting two points intersects one of the obstacles, the distance between the two points is assumed to be infinity, otherwise, it is equal to the (weighted) Euclidean distance between the points. In our application each edge defined by one of the trees is one such obstacle. Seidel shows how to adapt Fortune's algorithm to compute the constrained Voronoi diagram [9]. The adaption to additively weighted sites has been sketched in Fortune's original paper [4].

Corollary 2 *The planar augmentation problem for the Euclidean metric can be solved in* $\mathcal{O}((n_1 + n_2) \log n_2)$ *time.*

4 General Metrics

Every finite metric d can be encoded by a finite graph $M = (V, D)$ where each edge $e \in D$ has some length $\ell(e)$ and the distance d between two vertices in V is equal to the sum of the lengths of the shortest path between the vertices in the graph in terms of the edge lengths. We can directly translate our idea from the previous section to this setting by computing the additively weighted Voronoi diagram in M instead. Although the computation of various Voronoi diagrams on graphs has been considered by Hurtado et al. [5], among them a multiplicatively weighted Voronoi diagram, we are not aware of any investigation of the additively weighted Voronoi diagram on graphs. The following theorem is similar to the results by Hurtado et al. [5]. We assume that the additively weighted Voronoi diagram of a a set of sites $S \subseteq V$ on a metric graph $G = (V, E)$ is completely known if every vertex $v \in V \setminus S$ knows its nearest neighbor in S and we know the bisector point for each edge, if it exists.

Theorem 3 *The additively weighted Voronoi diagram of a set of sites* $S \subseteq V$ *on a graph* $G = (V, E)$ *has complexity* $\Theta(m)$ *and can be computed in time* $\mathcal{O}(m + n \log n)$.

Proof. Each edge of the graph contains at most one bisector point, since moving along the edge will alter the additively weighted distances by the same amount—either increasing or decreasing—for all distances. Hence we have at most m bisector points. On the other hand, we can have exactly m bisectors by setting $V' = V$. Hence, the complexity of the additively weighted Voronoi diagram is $\Theta(m)$.

To compute the additively weighted Voronoi diagram in G we use the parallel Dijkstra algorithm proposed by Erwig [2] with running time $\mathcal{O}(m + n \log n)$. To compute the diagram, we run Dijkstra's algorithm in parallel using the vertices in S as starting points. For a vertex $v \in V \setminus S$ and some vertex $s \in S$ the distance between v and s is $d_s(v, s) = d_G(v, s) + w(s)$. Whenever a vertex $v \in V \setminus S$ is settled, we update its closest neighbor in S. The bisector points can be computed in $\mathcal{O}(m)$ time from this information. □

Using this result, we can almost directly translate the technique for the Euclidean case to the general metric case studied in this section.

Theorem 4 *The optimal routing cost augmentation problem for general metrics can be solved in time $\mathcal{O}(m + n \log n)$ if the metric is given by a graph $M = (V, D)$ with edge length function ℓ.*

Proof. Instead of scaling the point set as in the Euclidean case, we scale the lengths of the edges in G by a factor C_1C_2, i.e., instead of using ℓ to assess the distance between two vertices in M, we use $C_1C_2\ell$. The rest of the proof is completely analogous. We compute the additively weighted Voronoi diagram on M for the set of sites V_2. Then we locate the vertex $u \in V_1$ that minimizes $C_2 \cdot w(u) + d_+(u, v)$ where $d_+(u, v)$ is the scaled and additively weighted distance between u and its closest neighbor v. The resulting time complexity is $\mathcal{O}(m + n \log n)$. □

Again we can proceed as in the Euclidean case in order to compute the second-best connection between the two trees.

Corollary 3 *The optimal routing cost replacement problem for general metrics can be solved in time $\mathcal{O}(m + n \log n)$ if the metric is given by a graph $M = (V, D)$ with edge length function ℓ.*

Although this result does not provide an asymptotic improvement in the worst-case, it does show that we can efficiently solve the augmentation problem for compactly representable metrics. If the graph representing the metric is sparse, then the above theorem states that we can solve the augmentation problem in $\mathcal{O}(n \log n)$ as in the Euclidean case.

5 Comments and Open Problems

We have studied a special class of augmentation problems, where the goal is to find the best connection between two disconnected trees in terms of routing cost. Although the problem can not be solved in subquadratic time for general distance functions in the algebraic decision tree model, it can be solved in $\mathcal{O}(n \log n)$ time for the Euclidean metric and sparse graph metrics.

It is an open question, for which graph metrics the problem can be solved in sub-quadratic time. Also, there are some interesting variants of the problem, for instance, when there are more than two disconnected trees. This problem arises, when a vertex of the network fails to work. Additionally, we could consider a Steiner-variant of the problem, in which we are allowed to introduce an additional vertex to which the disconnected components must be connected.

References

[1] R. Dionne and M. Florian. Exact and approximate algorithms for optimal network design. *Networks*, 9:37–60, 1979.

[2] Martin Erwig. The graph Voronoi diagram with applications. *Networks*, 36(3):156–163, 2000.

[3] Matteo Fischetti, Giuseppe Lancia, and Paolo Serafini. Exact algorithms for minimum routing cost trees. *Networks*, 39(3):161–173, 2002.

[4] Steven Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[5] Ferran Hurtado, Rolf Klein, Elmar Langetepe, and Vera Sacristán. The weighted farthest color Voronoi diagram on trees and graphs. *Computational Geometry*, 27(1):13 – 26, 2004.

[6] D. S. Johnson, J. K. Lenstra, and A. H. G. Rinnooy Kan. The complexity of the network design problem. *Networks*, 8(4):279–285, 1978.

[7] David Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12(1):28–35, 1983.

[8] A. J. Scott. The optimal network problem: Some computational procedures. *Transportation Research*, 3(2):201–210, 1969.

[9] R. Seidel. Constrained Delaunay triangulations and Voronoi diagrams with obstacles. Technical Report 260, IIG-TU Graz, Austria, 1988.

[10] Richard T. Wong. Worst-case analysis of network design problem heuristics. *SIAM Journal on Algebraic and Discrete Methods*, 1(1):51–63, 1980.

[11] Bang Ye Wu, Giuseppe Lancia, Vineet Bafna, Kun-Mao Chao, R. Ravi, and Chuan Yi Tang. A polynomial-time approximation scheme for minimum routing cost spanning trees. *SIAM J. Comput.*, 29:761–778, 1999.