The Inverse Voronoi Problem in Graphs

Édouard Bonnet∗ Sergio Cabello† Bojan Mohar‡ Hebert Pérez-Rosés§

December 3, 2018

Abstract

We introduce the inverse Voronoi diagram problem in graphs: given a graph \(G\) with positive edge-lengths and a collection \(U\) of subsets of vertices of \(V(G)\), decide whether \(U\) is a Voronoi diagram in \(G\) with respect to the shortest-path metric. We show that the problem is NP-hard, even for planar graphs where all the edges have unit length. We also study the parameterized complexity of the problem and show that the problem is \(W[1]\)-hard when parameterized by the number of Voronoi cells or by the pathwidth of the graph. For trees we show that the problem can be solved in near-linear time and provide a lower bound of \(\Omega(n\log n)\) time for trees with \(n\) vertices.

Keywords: distances in graphs, Voronoi diagram, inverse Voronoi problem, NP-complete, parameterized complexity.

1 Introduction

Let \((X,d)\) be a metric space, where \(d : X \times X \to \mathbb{R}_{\geq 0}\). Let \(S\) be a subset of \(X\). We refer to each element of \(S\) as a site, to distinguish it from an arbitrary point of \(X\). The Voronoi cell of each site \(s \in S\) is then defined by

\[
\text{cell}_{(X,d)}(s,S) = \{x \in X | \forall s' \in S : d(s,x) \leq d(s',x)\}.
\]

The Voronoi diagram of \(S\) in \((X,d)\) is

\[
\mathcal{V}_{(X,d)}(S) = \{\text{cell}_{(X,d)}(s,S) | s \in S\}.
\]

It is easy to see that, for each set \(S\) of sites, each element of \(X\) belongs to some Voronoi cell \(\text{cell}(s,S)\). Therefore, the sets in \(\mathcal{V}_{(X,d)}(S)\) cover \(X\). On the other hand, the Voronoi cells do not need to be pairwise disjoint. In particular, when some point \(x \in X\) is closest to two sites, then it is in both Voronoi cells.

∗Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France. Email address: edouard.bonnet@ens-lyon.fr. Supported by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program “Investissements d’Avenir” (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR).
†Department of Mathematics, IMFM, and Department of Mathematics, FMF, University of Ljubljana, Slovenia. Supported by the Slovenian Research Agency, program P1-0297 and projects J1-8130, J1-8155, J1-9109. Email address: sergio.cabello@fmf.uni-lj.si.
‡Department of Mathematics, Simon Fraser University, Burnaby, BC, Canada. Email address: mohar@sfu.ca. On leave from IMFM & FMF, Department of Mathematics, University of Ljubljana. Supported in part by the NSERC Discovery Grant R611450 (Canada), by the Canada Research Chairs program, and by the Research Project J1-8130 of ARRS (Slovenia).
§Departament d’Enginyeria Informàtica i Matemàtiques, Universitat Rovira i Virgili, Tarragona, Spain. Partially supported by Grant MTM2017-86767-R from the Spanish Ministry of Economy, Industry and Competitiveness.
In the inverse Voronoi problem, we are given a metric space \((X, d)\) and a sequence \(X_1, \ldots, X_k\) of subsets of \(X\) that cover \(X\). The task is to decide whether \(\{X_1, \ldots, X_k\}\) is a Voronoi diagram in \((X, d)\). This means that we have to decide whether there exists sites \(s_1, \ldots, s_k\) such that, for each index \(i\), we have \(X_i = \text{cell}_{(X,d)}(s_i, \{s_1, \ldots, s_k\})\).

The inverse Voronoi problem is closely related to problems in classification and clustering. In pattern recognition, a classic paradigm to classify is to use the nearest neighbor rule: given a learning set \(\mathcal{L}\) of objects that are already classified, each new object is classified into the same class as its closest object from \(\mathcal{L}\). To reduce the size of the learning set, Hart [17] introduced the concept of consistent subsets. A subset \(\mathcal{L}'\) of the learning set \(\mathcal{L}\) is a consistent subset if, for each object \(\ell\) from \(\mathcal{L}\), the object \(\ell\) and its closest neighbor in \(\mathcal{L}'\) are in the same class. An equivalent, alternative perspective of this is given by Voronoi diagrams: in the Voronoi diagram of a consistent subset \(\mathcal{L}'\), each object \(\ell\) of \(\mathcal{L}\) belongs to a Voronoi cell defined by a site \(s\) in \(\mathcal{L}'\) if and only if \(\ell\) and \(s\) belong to the same class. Ritter et al. [25] introduced the problem of finding consistent subsets of minimum size. Surveying the research in this applied area is beyond the scope of our research. We refer to Biniaz et al. [5] and Gottlieb et al. [16] for some of the latest algorithmic results on this topic. Considering each class as a Voronoi cell, the inverse Voronoi problem is asking precisely whether there exists a consistent subset with one element per class. Such consistent subset has of course to be of optimal size.

**Graphic version.** Let \(G\) be an undirected graph with \(n\) vertices and abstract, positive edge-lengths \(\lambda: E(G) \to \mathbb{R}_{\geq 0}\). The length of a path in \(G\) is the sum of the edge-lengths along the path. We define the (shortest-path) distance between two vertices \(x\) and \(y\) of \(G\), denoted by \(d_G(x, y)\), as the minimum length over all paths in \(G\) from \(x\) to \(y\).

Since \((V(G), d_G)\) is a metric space, we can consider the concepts of Voronoi cells and Voronoi diagrams for this space. We denote them by \(\text{cell}_G(s, S)\) and \(V_G(S)\) respectively. Moreover, when the graph is clear from the context, we remove the subindex and thus just talk about \(\text{cell}(s, S)\) and \(V(S)\).

In this paper we consider computational aspects of the inverse Voronoi problem when the metric space is the shortest-path metric in a graph. Thus, we consider the following problem.

**Graphic Inverse Voronoi**

Input: \((G, U)\), where \(G\) is a graph with positive edge-lengths and \(U = (U_1, \ldots, U_k)\) is a sequence of subsets of vertices of \(G\) that cover \(V(G)\).

Question: Are there sites \(s_1, \ldots, s_k \in V(G)\) such that \(\text{cell}_G(s_i, \{s_1, \ldots, s_k\}) = U_i\) for each \(i\)?

As far as the existence of polynomial-time algorithms is concerned, it is equivalent to consider a graph or a finite metric space. Indeed, for each finite metric space we can build a graph that encodes those distances by using a complete graph with edge-lengths, and, inversely, given a graph, we can compute the matrix of distances between all pairs of vertices in polynomial time. However, considering special classes of graphs may be useful to get more efficient algorithms.

**Our results.** First we show that the problem **Graphic Inverse Voronoi** is NP-hard even for planar graphs where the candidate Voronoi cell are pairwise disjoint and each has at most 3 vertices. The reduction is from a variant of **Planar 3-SAT**. The bound on the number of vertices per cells is tight: when each candidate Voronoi cell has 2 vertices, the problem can be solved using 2-SAT.

Many graph decision and optimization problems admit fixed-parameter tractable (FPT) algorithms with respect to additional parameters that quantify how complex is the input; see for instance [10]. Using the framework of parameterized complexity, we provide stronger lower
bounds when parameterized by the number $k$ of sites and the pathwidth $p(G)$ of $G$. More precisely, assuming the Exponential Time Hypothesis (ETH), we show that the problem cannot be solved in time $f(k)|V(G)|^{o(k/\log k)}$ nor in time $f(p(G))|V(G)|^{o(p(G))}$ for any computable function $f$. These hardness results hold for graphs where all the edges have unit length.

Then we consider efficient algorithms for the problem Graphic Inverse Voronoi when the underlying graph is a tree. One has to be careful with the size of the description of the input because the size of the Voronoi diagram may be quadratic in the size of the tree. For example, in a star with $2n$ leaves and sites in $n$ of the leaves, each Voronoi cell has size $\Theta(n)$, and thus an explicit description of the Voronoi diagram has size $\Theta(n^2)$. Motivated by this, we define the **description size** of an instance $I = (G, (U_1, \ldots, U_k))$ for the Graphic Inverse Voronoi to be $N = N(I) = |V(G)| + |E(G)| + \sum |U_i|$.

We show that the problem Graphic Inverse Voronoi for trees can be solved in $O(N \log^2 N)$ for arbitrary trees. We also show a lower bound of $\Omega(N \log N)$ in the algebraic computation tree model for trees with arbitrary edge-lengths.

One may be tempted to think that the problem is easy for trees. Our near-linear time algorithm for arbitrary trees is far from trivial. Of course we cannot exclude the existence of a simpler algorithm running in near-linear time, but we do think that the problem is more complex than it may seem at first glance. Figure 1 may help understanding that the interaction between different Voronoi cells may be more complex than it seems.

In our solution we first make a reduction to the same problem in which Voronoi cells are disjoint, and then we make another transformation to an instance having maximum degree 3. Finally, we employ a bottom-up dynamic programming procedure that, to achieve near-linear time, uses dynamic binary search trees to manipulate sets of intervals.

**Related work.** Voronoi diagrams on graphs were first investigated by Erwig [12], who showed that they can be efficiently computed. Subsequently, graph Voronoi diagrams have been used in a variety of applications. For instance, Okabe [24] describes several applications of graph Voronoi diagrams. More recent applications, many of them for planar graphs, can be found in [7, 9, 13, 14, 19, 22]. Voronoi diagrams in graphs have also been considered in the context of the so-called Voronoi game [3, 15] and in the context of topological data analysis [11].

On the other hand, the inverse Voronoi problem in the traditional, Euclidean setting has been studied since the mid 1980s, starting with the seminal paper by Ash and Bolker [2]. We are not aware of any previous work considering the graphic inverse Voronoi problem.
2 Basics

For a positive integer $k$ we use the notation $[k] = \{1, \ldots, k\}$.

Consider an instance $(G, (U_1, \ldots, U_k))$ to the GRAPHIC INVERSE VORONOI and a candidate solution $s_1, \ldots, s_k \in V(G)$. We say that $s_i$ and $s_j$ ($i \neq j$) are compatible if we have $d(s_i, u) = d(s_j, u)$ for each $u \in U_i \cap U_j$, $d(s_i, u) < d(s_j, u)$ for each $u \in U_i \setminus U_j$, and $d(s_i, u) < d(s_j, u)$ for each $u \in U_j \setminus U_i$. Consider a fixed index $i \in [k]$. It is straightforward from the definition that $\text{cell}_G(s_i, \{s_1, \ldots, s_k\}) = U_i$ if and only if $s_i$ and $s_j$ are compatible for all $j \neq i$. (Here it is relevant the assumption that $U_1 \cup \cdots \cup U_k$ is $V(G)$.)

The following results are folklore. In all cases we use the notation $X$ as the set of all vertices in $G$, $G$ as the ground graph that defines the metric. Note that in the following claims it is important that $G$ has positive edge-lengths.

We have remarked before that Voronoi cells need not be disjoint. A vertex belongs to various Voronoi cells if it is equidistant to different sites. An alternative is to define cells using strict inequalities. More precisely, for a set of sites $S$, the open Voronoi cell of each site $s \in S$ is then defined by

$$\text{cell}^< (s, S) = \{ x \in X \mid \forall s' \in S \setminus \{s\} : d(s, x) < d(s', x) \}.$$ 

In this case, the cells are disjoint but they do not necessarily form a partition of $X$. The following lemmas are straightforward and we omit their proofs.

**Lemma 1.** For each set $S$ of sites and each site $s \in S$ we have $s \in \text{cell}^< (s, S)$ and

$$\text{cell}^< (s, S) = \text{cell} (s, S) \setminus \bigcup_{s' \neq s} \text{cell} (s', S).$$

**Lemma 2.** For each set $S$ of sites, each site $s \in S$, and each vertex $v \in \text{cell}(s, S)$, every shortest path from $s$ to $v$ is contained in $G[\text{cell}(s, S)]$, the subgraph of $G$ induced by $\text{cell}(s, S)$. The same statement is true for $\text{cell}^< (s, S)$.

A consequence of this Lemma is that the shortest path from $s$ to $v \in \text{cell}(s, S) \setminus \text{cell}^< (s, S)$ has a path with vertices inside $\text{cell}^< (s, S)$ followed by a path with vertices of $\text{cell}(s, S) \setminus \text{cell}^< (s, S)$.

**Lemma 3.** Given an instance $I = (T, (U_1, \ldots, U_k))$ for the problem GRAPHIC INVERSE VORONOI, where $T$ is a tree, and a candidate solution $s_1, \ldots, s_k$, we can check in $O(N)$ time whether $s_1, \ldots, s_k$ is indeed a solution.

**Proof.** We add a new vertex $a$ (called the apex) to $T$ and connect it to each candidate site $s_1, \ldots, s_k$ with edges of the same positive length. See the left drawing in Figure 2. The resulting graph $T_a$ has treewidth 2, and thus we can compute shortest paths from $a$ to all vertices in linear time [8]. Let $d_a[v]$ be the distance in $T_a$ from $a$ to $v$.

Next we build a digraph $D_a$ describing the shortest paths from $a$ to all other vertices. The vertex set of $D_a$ is $V(T) \cup \{a\} = V(T_a)$. For each arc $u \rightarrow v$, where $uv \in E(T_a)$, we add $u \rightarrow v$ to $D_a$ if and only if $d_a[v] = d_a[u] + \lambda(uv)$. With this we obtain a directed acyclic graph $D_a$ that contains all shortest paths from $a$ to every $v \in V(T)$ and, moreover, each directed path in $D_a$ is indeed a shortest path in $T_a$. See Figure 2 right.

Now we label each vertex $v$ with the indices $i$ of those sites $s_i$, whose Voronoi cells contain $v$, as follows. We start setting $L(s_i) = \{i\}$ for each site $s_i$. Then we consider the vertices $v \in V(T)$ in topological order with respect to $D_a$. For each vertex $v$, we set $L(v)$ to be the union of $L(u)$, where $u$ iterates over the vertices of $V(T)$ with arcs in $D_a$ pointing to $v$. It is easy to see by induction that $L(v) = \{i \in [k] \mid v \in \text{cell}_T(s_i, \{s_1, \ldots, s_k\})\}$. During the process we keep a counter for $\sum_v |L(v)|$, and if at some moment we detect that the counter exceeds $N$, we stop and report that $s_1, \ldots, s_k$ is not a solution. Otherwise, we finish the process when we computed the sets $L(v)$.
Now we compute the sets $V_i = \{ v \in V(T) \mid s_i \in L(v) \}$ for $i = 1, \ldots, k$. This is done iterating over the vertices $v \in V(T)$ and adding $v$ to each site of $L(v)$. This takes $O(N + \sum_v |L(v)|) = O(N)$ time. Note that $V_i = \text{cell}_T(s_i, \{s_1, \ldots, s_k\})$. It remains to check that $U_i = V_i$ for all $i \in [k]$. For this we add flags to $V(T)$ that are initially set to false. Then, for each $i \in [k]$, we do the following: check that $|U_i| = |V_i|$, iterate over the vertices of $U_i$ setting the flags to true, iterate over the vertices of $V_i$ checking that the flags are true, iterate over the vertices of $U_i$ setting the flags back to false. The procedure takes $O(N + \sum_v |L(v)|) = O(N)$ time and, if all the checks were correct, we have $U_i = V_i = \text{cell}_T(s_i, \{s_1, \ldots, s_k\})$ for all $i \in [k]$.

3 Hardness of the Graphic Inverse Voronoi

In this Section we sow that the problem GRAPHIC INVERSE VORONOI is NP-hard, even for planar graphs. Stronger lower bounds are derived assuming the Exponential Time Hypothesis (ETH). We will reduce from a variant of the satisfiability (SAT) where each clause has 3 literals, all the literals are positive, and we want that each clause is satisfied at exactly one literal. The problem can be stated combinatorially as follows.

**POSITIVE 1-IN-3-SAT**

Input: $(\mathcal{Y}, \mathcal{C})$, where $\mathcal{Y}$ is a ground set and $\mathcal{C}$ is a family of subsets of $\mathcal{Y}$ of size 3.

Question: Is there a subset $T \subseteq \mathcal{Y}$ such that $|C \cap T| = 1$ for each $C \in \mathcal{C}$?

In this combinatorial setting, $\mathcal{Y}$ represents the variables, $\mathcal{C}$ represents the clauses with 3 positive literals each, and $T$ represents the variables that are set to true.

The incidence graph $I(\mathcal{Y}, \mathcal{C})$ of an instance $(\mathcal{Y}, \mathcal{C})$ has vertex set $\mathcal{Y} \cup \mathcal{C}$ and an edge between $v \in \mathcal{Y}$ and $C \in \mathcal{C}$ precisely when $v \in C$. The graph is bipartite.

As shown by Mulzer and Rote [23], the problem POSITIVE 1-IN-3-SAT is NP-complete even when the incidence graph is planar.

**Theorem 4.** The GRAPHIC INVERSE VORONOI problem is NP-hard on planar graphs with unit edge-lengths, even when the candidate Voronoi cells are disjoint sets of size at most 3.

**Proof.** We reduce from POSITIVE 1-IN-3-SAT with planar incidence graphs. Let $(\mathcal{Y} = \{x_1, \ldots, x_n\}, \mathcal{C} = \{C_1, \ldots, C_m\})$ be an instance of POSITIVE 1-IN-3-SAT with planar incidence graph. We produce an equivalent instance $(G, U)$ of GRAPHIC INVERSE VORONOI as follows. See Figure 3.
For each element \( x_i \in \mathcal{V} \), we add two vertices \( v(x_i) \) and \( \overline{v}(x_i) \) to the vertex set of \( G \), and we connect them by an edge. We add the set \( \{v(x_i), \overline{v}(x_i)\} \) to the candidate Voronoi cells \( \mathcal{U} \).

For each subset \( C_j = \{x_a, x_b, x_c\} \), we add three vertices \( v(j, a), v(j, b), \) and \( v(j, c) \) to \( V(G) \), and we connect the three pairs by an edge, forming a triangle. We add the set \( \Delta(j) = \{v(j, a), v(j, b), v(j, c)\} \) to \( \mathcal{U} \).

Finally, for each \( x_a \in \mathcal{V} \) and each \( C_j \in \mathcal{C} \) with \( x_a \in C_j \), we link \( v(j, a) \) to \( v(x_a) \) by an edge.

This finishes the construction of \((G, \mathcal{U})\). We observe that the sets of \( \mathcal{U} \) are indeed pairwise disjoint and of size 2 or 3. The graph \( G \) is planar since it is obtained from the planar incidence graph \( I(\mathcal{V}, \mathcal{C}) \) by adding pendant vertices and splitting each vertex representing a subset (with three neighbors) into a triangle in which each vertex is linked to one distinct neighbor.

If there is a solution \( T \) to the instance \((\mathcal{V}, \mathcal{C})\), we position the sites in the following way. For each \( x_i \in \mathcal{V} \), we place the site of \( \{v(x_i), \overline{v}(x_i)\} \) in \( v(x_i) \) if \( x_i \in T \), and in \( \overline{v}(x_i) \), otherwise. For each \( C_j = \{x_a, x_b, x_c\} \in \mathcal{C} \), we place the site of \( \Delta(j) \) in \( v(j, z) \), where \( x_z \) is the unique element of \( C_j \cap T \). We denote by \( S \) the obtained set of sites. We check that this placement defines the same Voronoi cells as specified by \( \mathcal{C} \).

For each \( \overline{v}(x_i) \in S \), we have \( \text{cell}_c(\overline{v}(x_i), S) \supseteq \{v(x_i), \overline{v}(x_i)\} \), since by construction there is no site in \( v(x_i) \). The only neighbors of \( \{v(x_i), \overline{v}(x_i)\} \) are vertices \( v(j, i) \) for some values of \( j \in [m] \). However, those neighbors do not contain a site of \( S \) by construction. On the other, there is always a site of \( S \) at distance at most 1 of \( v(j, i) \), whereas \( \overline{v}(x_i) \) is at distance 2 of \( v(j, i) \). Hence, \( \text{cell}_c(\overline{v}(x_i), S) = \{v(x_i), \overline{v}(x_i)\} \).

Similarly, for each \( v(x_i) \in S \), we have \( \text{cell}_c(v(x_i), S) \supseteq \{v(x_i), \overline{v}(x_i)\} \), since by construction there is no site in \( \overline{v}(x_i) \). The only neighbors of \( \{v(x_i), \overline{v}(x_i)\} \) are vertices \( v(j, i) \) for some values of \( j \in [m] \), but since \( v(x_i) \in S \), by construction, \( v(j, i) \) also belongs to \( S \). Therefore, \( \text{cell}_c(v(x_i), S) = \{v(x_i), \overline{v}(x_i)\} \).

Finally, consider some \( v(j, z) \in S \), where \( C_j = \{x_a, x_b, x_c\} \). We have \( \text{cell}_c(v(j, z), S) \supseteq \Delta(j) \) because \( v(j, z) \) is the only site in \( \Delta(j) \). The only other neighbor of \( v(j, z) \) is \( v(x_z) \), which is in \( S \). The only neighbor of \( v(j, z) \) with \( z' \in \{a, b, c\} \setminus z \) is \( v(x_{z'}) \) which is at distance 2 of \( v(j, z) \) and at distance 1 of the site \( \overline{v}(x_{z'}) \in S \). Thus, \( \text{cell}_c(v(j, z), S) = \Delta(j) \).

If there is no solution to the instance \((\mathcal{V}, \mathcal{C})\), we show that there is no solution to the Graphic Inverse Voronoi instance \((G, \mathcal{U})\). Fix a position of the sites. The set of sites \( S \) has to intersect
each \{v(x_i), \overline{v}(x_i)\} exactly once. Define the set

\[ T = \{ x_i \in \mathcal{V} \mid \text{the site chosen for } \{v(x_i), \overline{v}(x_i)\} \text{ is } v(x_i) \}. \]

As \( T \) is not a solution for the \textsc{POSITIVE 1-IN-3-SAT} instance, there is a \( C_j = \{x_a, x_b, x_c\} \in \mathcal{C} \) such that \( |C_j \cap T| \neq 1 \). We now turn our attention to the site chosen for \( \Delta(j) \). We distinguish two cases: \( |C_j \cap T| = 0 \) and \( |C_j \cap T| \geq 2 \). If \( |C_j \cap T| = 0 \), for every position of the site, say in \( v(j, z) \) (with \( z \in \{a, b, c\} \)), then \( \text{cell}_c(v(j, z), S) \) contains \( v(x_z) \), and therefore cannot be equal to \( \Delta(j) \). Now if \( |C_j \cap T| \geq 2 \), let \( v(x_r) \) and \( v(x_r') \) be two sites of S with \( z \neq z' \in \{a, b, c\} \). Since \( \Delta(j) \) contains precisely one site, we have \( \text{cell}_c(v(j, z), S) \) contains \( v(x_z) \), and therefore cannot be equal to \( \{v(x_z), \overline{v}(x_z)\} \). Similarly, if \( v(j, z') \notin S \), then \( \text{cell}_c(v(x_r'), S) \) contains \( v(x_r') \) in both cases, we reach the conclusion that there cannot be a solution for the instance \((G, \mathcal{U})\).

Note that in the argument we did not use that \( I(\mathcal{V}, \mathcal{C}) \) or \( G \) are planar.

Using additional properties of the reduction from \textsc{(PLANAR) 3-SAT} to \textsc{(PLANAR) POSITIVE 1-IN-3-SAT} given by Mulzer and Rote [23] and the Sparsification Lemma, we derive the following conditional lower bound.

\begin{corollary}
Unless the Exponential Time Hypothesis fails, the problem \textsc{GRAPHIC INVERSE Voronoi} cannot be solved in time \( 2^{o(n)} \) in general graphs and in time \( 2^{o(\sqrt{m})} \) in planar graphs, where \( n \) is the number of vertices, even when the potential Voronoi cells are disjoint and of size at most 3.
\end{corollary}

\begin{proof}
Applying the reduction of Mulzer and Rote [23] to a 3-SAT instance with \( n \) variables and \( m \) clauses gives an instance to \textsc{POSITIVE 1-IN-3-SAT} with \( O(n + m) \) variables and \( O(m) \) clauses. This is so because in their reduction each clause is replaced locally using \( O(1) \) new variables and clauses. The reduction and the proof used in Theorem 4 then gives an instance with \( O(n + m) \) vertices. (The reduction also works for non-planar instances, as mentioned at the end of the proof.) Therefore, if we could solve \textsc{GRAPHIC INVERSE Voronoi} in time \( 2^{o(\sqrt{|V(G)|})} \), we could solve any 3-SAT instance with \( n \) variables and \( m \) clauses in time \( 2^{o(\sqrt{|V(G)|})} \) = \( 2^{o(n + m)} \). However, the Sparsification Lemma [18] rules out, under the Exponential Time Hypothesis, a running time \( 2^{o(n + m)} \) for 3-SAT.

The reduction from 3-SAT to \textsc{PLANAR 3-SAT} given by Lichtenstein [20] increases quadratically the number of variables and clauses. Together with the reduction of Mulzer and Rote from \textsc{(PLANAR) 3-SAT} to \textsc{(PLANAR) POSITIVE 1-IN-3-SAT} and our reduction in the proof of Theorem 4, we conclude that each instance of 3-SAT with \( n \) variables and \( m \) clauses becomes an instance of \textsc{GRAPHIC INVERSE Voronoi} where the graph \( G \) is planar and has \( O((n + m)^2) \). Again, solving the problem in planar graphs in time \( 2^{o(\sqrt{|V(G)|})} \) time for planar graphs would contradict the Sparsification Lemma.
\end{proof}

This upper bound of 3 for the size of the potential Voronoi cells is sharp.

One can solve in polynomial time instances where \( \cdot \). We show that the problem can be solved in polynomial time when each potential Voronoi cell has at most two points not contained in other potential cells. For this, one uses a reduction to 2-SAT. Inspired by Lemma 1, we say that each \( U \in \mathcal{U} \) defines the \textit{potential open Voronoi cell}

\[ U \setminus \left( \bigcup_{U' \in \mathcal{U} \setminus \{U\}} U' \right). \]

\begin{theorem}
The \textsc{GRAPHIC INVERSE Voronoi} problem can be solved in polynomial time when all the potential open Voronoi cells are of size at most 2.
\end{theorem}
Figure 4: Left: An instance satisfying the hypothesis of Theorem 6. The vertices of each $U \in \mathcal{U}$ are enclosed by dashed curve. The crosses indicate the position when the variables are true. Of the $\binom{6}{2} = 15$ clauses for (in)compatibility, some of them are $x_3 \lor x_5 \lor x_4$, $x_5 \lor x_4$, $x_2 \lor x_3$.

Proof. We present a polynomial reduction to 2-SAT. See Figure 4 for an example. Let $(G, \mathcal{U} = \{U_1, \ldots, U_k\})$ be the GRAPHIC INVERSE VORONOI instance. We denote by $U'_i$ the open potential Voronoi cell of the potential Voronoi cell $U_i$. By assumption, $|U'_i| \leq 2$. Because of Lemma 1, if the instance has a solution, then $s_i \in U'_i$. For each open cell $U'_i$, we introduce a variable $x_i$. We interpret putting the site on one fixed but arbitrary vertex of $U'_i$ to setting $x_i$ to true, and putting the site on the other vertex (if it exists) to setting $x_i$ to false. Now, $\mathcal{V}_G(S) = \mathcal{U}$ if and only if for each pair of sites $s_i, s_j \in S$ with $s_i \in U'_i$ and $s_j \in U'_j$:

- every vertex of $U_i \setminus U_j$ is strictly closer to $s_i$ than to $s_j$, and
- every vertex of $U_j \setminus U_i$ is strictly closer to $s_j$ than to $s_i$, and
- every vertex of $U_i \cap U_j$ is equidistant to $s_i$ and $s_j$.

Therefore, one just needs to check that each pair of sites of $S$ is compatible, that is, satisfies those three conditions.

We define the following set of 2-SAT constraints. For each open cell $U'_i$ of size 1, we add the clause $x_i$, which forces to set $x_i$ to true. For each pair $s_i \in U'_i, s_j \in U'_j$ which is not compatible we add the clause $\ell_i \lor \ell_j$ where $\ell_i$ (resp. $\ell_j$) is the opposite literal to the one chosen by placing a site in $s_i$ (resp. $s_j$).

It is easy to check that the produced 2-SAT formula is satisfiable if and only if there is a pairwise compatible set of sites. This is in turn equivalent to the existence of a solution for the GRAPHIC INVERSE VORONOI instance.

\[ \square \]

4 Hardness parameterized by the number of Voronoi cells

In the previous section we showed that the problem GRAPHIC INVERSE VORONOI is NP-hard. Stronger lower bounds are derived under the assumption of the Exponential Time Hypothesis (ETH). We will prove the following result.

Theorem 7. The GRAPHIC INVERSE VORONOI problem is W[1]-hard parameterized by the number of candidate Voronoi cells. Furthermore, for $n$-vertex graphs and $k$ subsets to be candidate Voronoi
cells, for any computable function $f$, there is no algorithm to solve the Graphic Inverse Voronoi problem in $f(k)n^{O(k/\log k)}$ time, unless the Exponential Time Hypothesis fails. The claim holds even for graphs with unit edge-lengths.

Note that it is trivial to solve the problem in $n^O(k)$ time: just try each $n^k$ tuples of $k$ vertices as candidate sites and check each of them. The remaining of this section is devoted to prove Theorem 7. We will reduce from the following problem:

**Multicolored Subgraph Isomorphism**

**Input:** $(H, P)$, where $H$ is a graph whose vertex set $V(H)$ is partitioned into $\ell$ pairwise disjoint sets $V_1 \cup \ldots \cup V_\ell$, and a pattern graph $P$ with vertex set $V(P) = [\ell]$.

**Question:** Can we select vertices $v_i \in V_i$ for every $i \in [\ell]$ such that we have $v_i v_j \in E(H)$ for each $i j \in E(P)$?

When the answer is positive, we say that $P$ is isomorphic to a multicolored subgraph of $H$. It follows from the work of Marx [21] that, assuming the Exponential Time Hypothesis, the **Multicolored Subgraph Isomorphism** cannot be solved in time $f(\ell)n^{O(\ell/\log \ell)}$ for any computable function $f$, even when the pattern $P$ has $\Theta(\ell)$ edges. This lower bound is made explicit for example in [22, Corollary 5.5], where $P$ is assumed to be 3-regular.

Consider an instance $(H, P)$ to the **Multicolored Subgraph Isomorphism** problem, where $V_1, \ldots, V_\ell$ are the partite classes of $V(H)$ and $P$ has $\Theta(\ell)$ edges. We assume for simplicity that each vertex of $P$ has degree at least 2. For each $i, j \in [\ell]$, let $E_H(V_i, V_j)$ denote the edges of $H$ with one endpoint on $V_i$ and the other endpoint in $V_j$. We shall assume that $E_H(V_i, V_j)$ is empty whenever $i j \notin E(P)$ because those edges can be removed without affecting the instance. We build an instance $(G, U) = (G(H, P), U(H, P))$ for the **Graphic Inverse Voronoi** problem as follows.

Figures 5 and 6 may be helpful to follow the construction.

- We start with $V(G) = V(H)$ and $E(G) = E(H)$.
- For each $i \in [\ell]$, we add all edges between all the vertices in $V_i$.
- We subdivide each edge $e$ of $G$ with a new vertex, which we call $w(e)$.
- For each $i j \in E(P)$, let $W_{i,j}$ be the vertices $w(e)$ used to subdivide $E_H(V_i, V_j)$. We add all edges between all the vertices in $W_{i,j}$.

![Figure 5: A graph $H$ (left) and a pattern $P$ (right) for the Multicolored Subgraph Isomorphism.](image)
Figure 6: Example showing the construction for \((H,P)\) of the Figure 5. Left: the graph \(G\). The vertices inside a connected shaded region form a clique whose edges are not shown in the drawing. Right: three (of the five) candidate Voronoi cells of \(U\) are indicated by shaded regions of different colors. The cliques induced by \(V_i\) and \(W_{ij}\) are not shown in this figure.

- For each \(i j \in E(P)\), we add \(U_{ij} = W_{ij} \cup V_i \cup V_j\) to \(U\).

All the edges have unit length. This completes the construction of \(G = G(H,P)\) and \(U = U(H,P)\). Note that \(U\) has \(|E(P)| = \Theta(\ell)\) candidate Voronoi regions, while \(G\) has \(|V(H)| + |E(H)| = \Theta(|V(H)|^2)\) vertices and

\[
\sum_{i \in [\ell]} \left( \frac{|V_i|}{2} \right) + \sum_{ij \in E(P)} \left( \frac{|E_H(V_i, V_j)|}{2} \right)
\]

edges.

The next two lemmas show that the pair \((G,U)\) is a correct reduction from MULTICOLORED SUBGRAPH ISOMORPHISM to GRAPHIC INVERSE VORONOI. The intuition of the reduction is that selecting the site of each Voronoi cell corresponds to selecting an edge of \(E_H(V_i, V_j)\) for each \(ij \in E(P)\). Moreover, the selection of the edges we make need to have compatible endpoints in each partite set \(V_i\), as otherwise we do not get the correct Voronoi cells.

**Lemma 8.** If \(P\) is isomorphic to a multicolored subgraph of \(H\), then \(G\) has a set \(S\) of sites such that \(V(S) = U\).

**Proof.** Assume that \(P\) is isomorphic to a multicolored subgraph of \(H\). This means that there are vertices \(v_i \in V_i\), for every \(i \in [\ell]\), such that \(v_i v_j \in E(H)\) for every \(ij \in E(P)\). This means that, for every \(ij \in E(P)\), the vertex \(w(v_i, v_j)\) obtained when subdividing \(v_i v_j\) belongs to \(G\). We define \(s_{ij} = w(v_i, v_j)\) for every \(ij \in E(P)\) and \(S = \{w(v_i, v_j) | ij \in E(P)\}\).

We claim that \(S\) is a set of sites in \(G\) such that \(\text{cell}(s_{ij}, S) = U_{ij}\), for every for every \(ij \in E(P)\). This claim implies the lemma.
For each $s_{ij} \in S$ and for each vertex $u$ of $G$ we have the following distances

$$d_G(s_{ij}, u) = \begin{cases} 
0 & \text{if } u = s_{ij}, \\
1 & \text{if } u \in W_{ij} \setminus \{s_{ij}\}, \\
1 & \text{if } u = v_i \text{ or } v = u_j, \\
2 & \text{if } u \in (V_i \cup V_j) \setminus \{v_i, v_j\}, \\
\geq 2 & \text{if } u \in W_{i'j'} \text{ for some } i'j' \neq ij, \\
\geq 3 & \text{if } u \in V_t \text{ for some } t \neq i, j.
\end{cases}$$

Therefore, each vertex in $V_1 \cup \cdots \cup V_\ell = V(H)$ is at distance at most 2 from some vertex of $S$ and each vertex in $\cup_{i \in E(P)} W_{ij}$ is at distance at most 1 from some vertex of $S$.

Now we note that, for each $ij \in E(P)$, each vertex of $W_{ij}$ is strictly closer to $s_{ij}$ than to any other site. Furthermore, for each $i \in [\ell]$, each vertex of $V_i$ has the same distance to each site $s_{ij}$ with $ij' \in E(P)$, and a larger distance to each site $s_{i'j'}$ with $i'j' \in E(P - i)$. Therefore $cell(s_{ij}, S) = W_{ij} \cup V_i \cup V_j = U_{ij}$. The result follows.

\[ \square \]

**Lemma 9.** If $G$ has as set $S$ of sites such $\forall(S) = \emptyset$, then $P$ is isomorphic to a multicolored subgraph of $H$.

**Proof.** Let $S$ be a set of sites in $G$ such that $\forall(S) = \emptyset$. For each $ij \in E(P)$, let $s_{ij}$ be the site of $S$ with $cell(s_{ij}, S) = U_{ij}$.

Because of Lemma 1, each $s_{ij} \in S$ belongs to

$$cell^c(s_{ij}, S) = cell(s_{ij}, S) \setminus \left( \bigcup_{s \in S \setminus \{s_{ij}\}} cell(s, S) \right) = U_{ij} \setminus \left( \bigcup_{i'j' \in E(P) \setminus \{ij\}} U_{i'j'} \right) = W_{ij}.$$

In the last equality we have used that each vertex of $P$ has degree at least 2, which means that each $V_i$ is contained in at least 2 sets $U_{ij}$ of $S$. We conclude that, for each $ij \in E(P)$, the site $s_{ij}$ must be in $W_{ij}$.

Since each site $s_{ij}$ is in $W_{ij}$, for each for each $ij \in E(P)$, the construction of $G$ implies that there are unique vertices $v(i, ij) \in V_i$ and $v(j, ij) \in V_j$ such that $s_{ij}$ is the vertex obtained when subdividing the edge connecting $v(i, ij)$ and $v(j, ij)$. In particular, $v(i, ij)v(j, ij)$ is an edge of $H$.

Fix the index $i \in [\ell]$ and consider two edges $ij, ij' \in E(P)$ incident to $i$. We must have $v(i, ij) = v(i, ij')$, as otherwise we would have $d_G(s_{ij}, v(i, ij)) = 1 < 2 = d_G(s_{ij}, v(i, ij'))$, which would imply that $V_i \not\subseteq cell(s_{ij}, S) = U_{ij}$ and would contradict the definition of $U_{ij} = V_i \cup V_j \cup W_{ij}$. Therefore, each of the (three) edges $ij$ of $E(P)$ define the same vertex $v(i, ij) \in V_i$. We denote this vertex henceforth $v_i$.

We have found $\ell$ vertices $v_1, \ldots, v_\ell$ with the property that $v_i \in V_i$, for each $i \in [\ell]$, and such that the edge $v_iv_j = v(i, ij)v(j, ij)$ in $E(H)$, for each $ij \in E(P)$. This means that $P$ is isomorphic to the multicolored subgraph of $H$ defined by $\{v_1, \ldots, v_\ell\}$.

\[ \square \]

**Proof of Theorem 7.** As shown in Lemmas 8 and 9, $H$ has a multicolored subgraph isomorphic to $P$ if and only if $U$ is a valid Voronoi diagram of $G$. Thus, the answer to \textsc{Multicolored Subgraph Isomorphism}($H, P$) and \textsc{Graphical Inverse Voronoi}($G, U$) is the same.

Recall that $U$ has $|E(P)| = \Theta(\ell)$ candidate Voronoi regions. If we could solve each instance of the \textsc{Graphical Inverse Voronoi} problem with $n$ vertices and $k$ sites in time $f(k)n^{o(k/\log k)}$, for some computable function $f$, then we could solve the instance $(G, U)$ in

$$f(|U|) \cdot |V(G)|^{o(|U|/\log |U|)} \leq f(\Theta(\ell))(\Theta(|V(H)|^2))^{o(\Theta(\ell)/\log(\Theta(\ell)))} \leq g(\ell)|V(H)|^{o(\ell/\log \ell)}$$

time, for some computable function $g$. However, this also means that we could solve the \textsc{Multicolored Subgraph Isomorphism} in $H$ with pattern $P$ in $g(\ell)|V(H)|^{o(\ell/\log \ell)}$ time, and this contradicts the Exponential Time Hypothesis.

\[ \square \]
5 Hardness parameterized by the pathwidth and the treewidth

In this section we show that the Graphic Inverse Voronoi problem is unlikely to be fixed parameter tractable with respect to the pathwidth of the graph. Since the pathwidth is always smaller than the treewidth, this implies the same result for the treewidth. More precisely, in this section we will prove the following.

**Theorem 10.** The Graphic Inverse Voronoi problem is W[1]-hard parameterized by the pathwidth of the input graph. Furthermore, for n-vertex graphs with pathwidth p, there is no algorithm to solve the Graphic Inverse Voronoi problem in time $f(p)n^{o(p)}$ for any computable function $f$, unless the Exponential Time Hypothesis fails. The claims hold even for graphs with unit edge-lengths and disjoint candidate Voronoi cells.

In order to show that we will reduce from the following W[1]-hard problem.

**Multicolored Independent Set**

**Input:** A graph $H = (V, E)$ whose vertex set $V$ is partitioned into $\ell$ pairwise disjoint sets $V_1 \cup \ldots \cup V_\ell$.

**Question:** Is there an independent set $X$ of size $\ell$ in $H$ such that $|X \cap V_i| = 1$, $\forall i \in [\ell]$?

The Multicolored Independent Set problem is W[1]-hard with respect to $\ell$ and cannot be solved in time $f(\ell)n^{o(\ell)}$ for any computable function $f$, assuming the Exponential Time Hypothesis [10, Corollary 14.23]. The lower bounds still hold if all the partite sets $V_i$ have the same cardinality and the there are no edges connecting any two vertices within a set $V_i$.

Let $H = (V_1 \cup \ldots \cup V_\ell, E)$ be an instance of Multicolored Independent Set such that $|V_1| = |V_2| = \ldots = |V_\ell| = t$. Let $m$ be the number of edges in $H$. We build an equivalent Graphic Inverse Voronoi instance $(G, U)$ where the treewidth of $G$ is $\Theta(\ell)$. This instance will have unit edge-length edges and the sets in $U$ will be pairwise disjoint.

Our global strategy for the reduction is to propagate a vertex choice in each $V_i$ with a path-like structure with $\ell$ rows and $m = |E|$ columns. In each column, we introduce a single distinct edge of $E$ so that the pathwidth of the built graph stays in $\Theta(\ell)$. Figure 8 shows the whole reduction in the graph $H$ of Figure 7. (Seeing the details requires zooming in.) In Figure 9 we show a part of the construction in detail showing also the notation we employ. The detailed construction is as follows.

- For each $i \in [\ell]$ and $j \in [m]$, we add to $V(G)$ an independent set $I(i, j)$ of size $|V_i| = t$. The vertices of the independent set $I(i, j)$ are denoted by $v(i, j, 1)$ to $v(i, j, t)$, the third index being in one-to-one correspondence with the vertices of $V_i$.
- For each $i \in [\ell]$ and $j \in [m]$, we add two vertices $a(i, j)$ and $b(i, j)$. Furthermore, for each $h \in [\ell]$, we connect $a(i, j)$ to $v(i, j, h)$ by a private path $P_a(i, j, h)$ of length $t + h$, and we connect $b(i, j)$ to $v(i, j, h)$ by a private path $P_b(i, j, h)$ of length $t + h$. For each $i \in [\ell]$ and $j \in [m - 1]$, we connect $b(i, j)$ and $a(i, j + 1)$ by an edge.
- For each $i \in [\ell]$ and $j \in [m]$, we add three new vertices $c(i, j)$, $e(i, j)$, and $z(i, j)$. For each $h \in [\ell]$, we add a private path $P_c(i, j, h)$ of length $t$ between $v(i, j, h)$ and $c(i, j)$. Furthermore, we connect $e(i, j)$ and $z(i, j)$ with an edge and add a path $P_e(i, j)$ of length $t$ with one extreme on $e(i, j)$ and the other extreme connected through an edge to $c(i, j)$. (Thus $e(i, j)$ and $c(i, j)$ are connected with a path of length $t + 1$.)
- For each $i \in [\ell]$ and $j \in [m]$, we denote by $U(i, j)$ the set of vertices comprising $I(i, j)$ and all the paths going from this independent set to $a(i, j)$, $b(i, j)$, and $c(i, j)$, including those three vertices. We add $U(i, j)$, $V(P_e(i, j))$, and $Z(i, j) = \{z(i, j)\}$ to the candidate Voronoi cells $U$. 

12
Figure 7: A graph $H$ whose vertex set is partitioned into $\ell = 3$ partite sets, each with $t = 4$ vertices.

- We call \textbf{j-th column} the set $\bigcup_{i} U(i, j) \cup P_{e}(i, j) \cup Z(i, j)$ for a fixed $j \in [m]$. We introduce exactly one distinct edge of $E$ per column. Let $e_1, \ldots, e_m$ be any ordering of the edges of $E$. We put an edge gadget encoding $e_j$ in the $j$-th column, for every $j \in [m]$. Assume that $e_j$ is an edge between the $h$-th vertex of $V_i$ and the $h'$-th vertex of $V_{i'}$ where $i \neq i'$. We add a path $P(e_j)$ of length $2t + 2$ between $v(i, j, h)$ and $v(i', j, h')$. We add a path $Q(e_j)$ of length $t$ between the middle vertex of $P(e_j)$ and a new vertex, denoted $f(j)$. (The vertex $f(j)$ has degree 1 and it is at distance $2t + 1$ from $v(i, j, h)$ and from $v(i', j, h')$.) We add $R(j) = V(Q(e_j)) \cup V(P(e_j)) \setminus \{v(i, j, h), v(i', j, h')\}$ as a candidate Voronoi region to $\mathbb{U}$. The subgraph induced by $R(j)$ is the \textbf{edge gadget} of $e_j$.

That finishes the construction of $G = G(H)$ and of $\mathbb{U} = \mathbb{U}(H)$. All the edges of $G$ have unit length. One can observe that $\mathbb{U}$ is made of pairwise disjoint sets and it contains $(3\ell + 1)m$ candidate Voronoi cells. We first show that the pathwidth (and thus also the treewidth) of $G$ is at most $2\ell + 4$. For that, we use the pursuit-evasion game characterization of pathwidth.

\textbf{Lemma 11.} The pathwidth of $G$ is at most $2\ell + 4$.

\textbf{Proof.} In the pursuit-evasion game, searchers try to find a fugitive hidden at an edge of the graph. The searchers occupy vertices of the graph (at most one searcher per vertex). At each step, the searchers can change their position arbitrarily (they do not need to travel via edges), whereas the fugitive can move along any path that does not cross a searcher and occupy a new edge (or stay put). The fugitive is caught when both endpoints of her/his edge is occupied by a searcher. The minimum number of searchers needed to get a winning strategy for the searchers is equal to the pathwidth plus one.

We present a winning strategy for capturing a fugitive in $G$ using $2\ell + 5$ searchers. We make $m$ rounds where in the $j$-th round, $j = 1, \ldots, m$, we scan completely the $j$-th column and the gadget for $e_j$.

At the start of the $j$-th round we have $2\ell$ searchers placed at the vertices $a(i, j)$ and $b(i, j)$ for all $i \in [\ell]$. Assume that the edge $e_j \in E$ is between the $h$-th vertex of $V_i$ and the $h'$-th vertex of $V_{i'}$. We place two searchers at $v(i, j, h)$ and $v(i', j, h')$. Let $X_j$ be the set of $2\ell + 2$ vertices where we have searchers. They will stay there for most of the $j$-th round. We then search the whole first column plus the edge gadget of $e_j$ using the remaining three searchers. For this, we note that each connected component of $G - X_j$ contained in the $j$-th column and the connected
Figure 8: Whole graph showing the reduction used to proof Theorem 10 for the graph $H$ in Figure 7. Each connected gray area corresponds to one candidate Voronoi region. Figure 9 shows details for a part of the construction.
Figure 9: Left: Zoom into a part of the reduction shown in Figure 8 with some notation. Each connected gray area corresponds to one candidate Voronoi region. Some selection of sites marked with crosses that is locally correct (but globally would have a problem). This selection corresponds to selecting vertex 3 of $V_1$, vertex 2 of $V_2$ and vertex 1 of $V_3$. 
component induced by $R(j)$ has pathwidth 2, and thus it can be be searched with three searchers. At this point, the fugitive, if not captured yet, has to be to the right of the $\ell$ searchers placed at $b(1,j), \ldots, b(\ell,j)$, that is, on some edge incident to some vertex defined by $j' > j$. If $j = m$, we are done since there are no edges left to host the fugitive. Otherwise, we move the searchers from $a(1,j), \ldots, a(\ell,j)$ to $a(1,j+1), \ldots, a(\ell,j+1)$, then the searchers from $b(1,j), \ldots, b(\ell,j)$ to $b(1,j+1), \ldots, b(\ell,j+1)$, and start the next round.

We now show the correctness of the reduction.

**Lemma 12.** If $H$ has a multicolored independent set of size $\ell$, then there is a set $S \subseteq V(G)$ such that $V_G(S) = U$.

**Proof.** Assume there is a multicolored independent set $X$ of size $\ell$ in $H$. We define the set of sites $S$ as follows.

- For each $i \in [\ell]$, we place a site on the $m$ vertices $v(i,j,h_i)$ for all $j \in [m]$, where $h_i$ is the index of the vertex of $X$ in color class $i$.
- For every $i \in [k]$ and $j \in [m]$, we place two sites at $e(i,j)$ and $z(i,j)$.
- For each edge $e_j$ of $H$ with no endpoint in $X$, we place a site at $s_j = f(j)$.
- For each edge $e_j$ of $H$ with exactly one endpoint in $X$, we place a site on the vertex $s_j$ of $R(j)$ adjacent to $v(i,j,h_i)$.

Note that, since $X$ is an independent set, there cannot be an edge $e_j$ with two endpoints in $X$. Therefore we have covered all cases. This finishes the placement of the sites.

For every $i \in [\ell]$ and $j \in [m]$, we have $\text{cell}(z(i,j), T) = Z(i,j)$ because $e(i,j)$, the only neighbor of $z(i,j)$, is also a site. It also holds that $\text{cell}(e(i,j), T) = P_e(i,j)$ because $c(i,j)$ is at distance $t+1$ from $e(i,j)$ and at distance $t$ from the site $v(i,j,h_i)$. For every $i \in [k]$ and $j \in [m-1]$, the vertices $v(i,j,h_i)$ and $v(i,j+1,h_i)$ are compatible since $d_G(v(i,j,h_i), b(i,j)) = t+h_i = d_G(v(i,j+1,h_i), a(i,j))$ and $d_G(v(i,j,h_i), a(i,j+1)) = t+h_i+1 = d_G(v(i,j+1,h_i), b(i,j))$. Here it is relevant the choice of the lengths of the paths $P_e(i,j,h_i), P_i(i,j,h)$ and $P(i,j,h)$ to ensure that the shortest path from vertex $v(i,j,h)$ to $a(i,j)$ is $P_e(i,j,h_i)$, instead of passing through $b(i,j)$ or $c(i,j)$. (Similar statements hold for the shortest paths from $v(i,j,h)$ to $b(i,j)$ and to $c(i,j)$.)

We now only need to check that the site $s_j$ in the edge gadget of $e_j$ –the edge, say, between the $h$-th vertex of $V_j$ and the $h'$-th vertex of $V_{\ell}$– is compatible with the sites chosen in $S$ for $U(i,j)$ and $U(i',j)$. The nice property that makes everything work is that, for every $i \in [k], j \in [m], h \neq h' \in [t]$, $d_G(v(i,j,h), v(i,j,h'))$ is always equal to $2t$. Indeed the shortest path between $v(i,j,h)$ and $v(i,j,h')$ goes through $e(i,j)$, which is at distance $t$ of both vertices.

There are two cases: $s_j$ is the middle vertex of $P(e_j)$ or $s_j = f(j)$. If $s_j \in P(e_j)$, it means that one of the endpoints of $e_j$ is in the multicolored independent set $X$. Without loss of generality, we assume that it is the $h$-th vertex of $V_j$ (hence, $h = h_i$). In that case, the sites $s_j$ and $v(i,j,h_i)$ are adjacent vertices, and therefore they are compatible. The sites $s_j$ and $v(i',j,h_i)$ are also compatible since $d_G(s_j, v(i',j,h_i)) = 2t+1$ and $d_G(v(i',j,h'), v(i',j,h_i)) = 2t$.

Now, if $s_j = f(j)$, it means that $e_j$ does not touch any vertex of $S$. Hence, $h \neq h_i$ and $h' \neq h_i$. Then we have $d_G(s_j, v(i,j,h_i)) = d_G(s_j, v(i',j,h_i)) = 2t+1$, $d_G(v(i,j,h), v(i,j,h_i)) = 2t$ and $d_G(v(i',j,h'), v(i',j,h_i)) = 2t$. It follows that also in this case the site $s_j$ is compatible with $v(i,j,h_i)$ and $v(i',j,h_i)$.

Therefore, we showed that each site $v(i,j,h_i) \in S$ is compatible with every other site of $S$. This implies that for every $i \in [k]$ and $j \in [m]$, we have $\text{cell}(v(i,j,h_i), T) = U(i,j)$. In turn, it implies that $\text{cell}(s_j, T) = R(j)$ for each $j \in [m]$, and therefore $V_G(S) = U$. \qed
Lemma 13. If $H$ has no multicolored independent set of size $k$, then there is no set $S \subseteq V(G)$ such that $\mathcal{V}_G(S) = \emptyset$.

Proof. A solution for the Graphic Inverse Voronoi has to put sites on every $e(i, j)$ and $z(i, j)$, otherwise the Voronoi cell $Z(i, j)$ would not appear in the set of cells. As $e(i, j)$ is at distance $t + 1$ of $c(i, j)$, the site chosen for the cell $U(i, j)$ has to be at distance exactly $t$ of $c(i, j)$ (otherwise, this site would not be compatible with $e(i, j)$). So, the site chosen for $U(i, j)$ has to be in $I(i, j)$.

Then we prove that if a site is placed on $v(i, j, h)$, a site should be placed consistently on $v(i, j + 1, h)$. This is immediate by construction, since the only vertex of $U(i, j + 1)$ which has a distance to $a(i, j + 1)$ equal to $d_G(v(i, j, h), b(i, j)) = t + h$ is $v(i, j + 1, h)$. Here we are using again that the shortest path from $v(i, j, h)$ to $b(i, j)$ is indeed $P_b(i, j, h)$, and does not detour through $a(i, j)$ or $c(i, j)$. This implies that, for each $i \in [\ell]$, all the choices of sites for the cells $\{U(i, j)\}_{j \in [m]}$ have to be consistent to the same vertex, say of index $h_i$ in $V_i$. This defines a (consistent) set $X$ of $\ell$ vertices of $H$.

As by assumption $\ell$ cannot be an independent set, there is an edge $e_j$ with both endpoints in $X$. Say those endpoints are the vertices in color classes $i$ and $i'$. Then, the site for $R(j)$ cannot be closer to the two vertices of $R(j)$ that are adjacent to $v(i, j, h_i)$ and $v(i', j, h_{i'})$. Hence there is no $S \subseteq V(G)$ such that $\mathcal{V}_G(S) = \emptyset$.

Proof of Theorem 10. Because of Lemmas 12 and 13, solving Graphic Inverse Voronoi for $(G, U)$ also solves Multicolored Independent Set for $H$. The graph $G$ has $O(m |V(H)|) = O(|V(H)|^4)$ vertices and pathwidth $p \leq 2\ell + 5$ because of Lemma 11. An algorithm for the Graphic Inverse Voronoi with running time $f(p) |V(G)|^{o(p)}$ (for some computable function $f$) would imply that we can solve Multicolored Independent Set in time $f(2\ell + 5) (|V(H)|^4)^{o(2k + s)} = g(\ell)n^{o(\ell)}$ for a computable function $g$. This would disprove the Exponential Time Hypothesis.

We show an almost matching upper bound when the potential Voronoi cells form a partition of the vertex set.

Theorem 14. Instances $(G, U)$ of Graphic Inverse Voronoi can be solved in time $|V(G)|^{O(w \log k)}$, when the $k$ cells of $U$ are pairwise disjoint and $w$ is the treewidth of $G$.

Proof. We solve a more general problem where each potential Voronoi cell of $U$ comes with a prescribed subset, specifying where one can actually place its site. Let $H$ be the graph on $k$ vertices obtained by contracting each cell of $U$ into a single vertex. The treewidth of $H$ is at most $w$. We exhaustively guess in time $k^w$ a balanced vertex-separator $S$ of size $w$ in the graph $H$. Each connected component of $H - S$ has thus less than $2k/3$ vertices. We further guess in time at most $|V(G)|^w$ the $w$ corresponding sites in a fixed solution. For each guess, we remove the $w$ corresponding cells – say their union is $U$ – from $G$, update the prescribed subsets of the remaining cells to those placements compatible with the sites that are already fixed, and solve recursively each connected component of $G - U$. Thus, we get $|V(G)|^{2w}$ independent subproblems, each of them with at most $2k/3$ candidate Voronoi regions (and restricted subset of possible placements). Since the depth of the branching tree is $O(\log k)$, the total running time is bounded by $|V(G)|^{O(w \log k)}$. \qed

6 Arbitrary trees – Transforming to nicer instances

In this and the following section we consider the problem Graphic Inverse Voronoi for trees. In this section we provide a transformation to reduce the problem to trees of maximum degree 3 and disjoint Voronoi regions. In fact, we have to consider the following more general problem, where the input also specifies, for each Voronoi cell, a subset of vertices where the site has to be placed.
GENERALIZED GRAPHIC INVERSE VORONOI

Input: \((G, U)\), where \(G\) is a graph with positive edge-lengths and \(U = ((U_1, S_1), \ldots, (U_k, S_k))\) is a sequence of pairs of subsets of vertices of \(G\).

Question: are there sites \(s_1, \ldots, s_k \in V(G)\) such that \(s_i \in S_i\) and \(U_i = \text{cell}_G(s_i, \{s_1, \ldots, s_k\})\) for each \(i \in [k]\)?

Note that we may assume that \(S_i \subseteq U_i\) for each \(i \in [k]\). Following the analogy with GRAPHIC INVERSE VORONOI, we define the \textit{description size} of an instance \(I = (G, ((U_1, S_1), \ldots, (U_k, S_k)))\) to be \(N(I) = |V(G)| + |E(G)| + \sum_i |U_i| + \sum_i |S_i|\).

Clearly, the problem GRAPHIC INVERSE VORONOI can be reduced to the problem GENERALIZED GRAPHIC INVERSE VORONOI by taking \(S_i = U_i\) for each \(i \in [k]\). This transformation can be done in linear time (in the size of the instance) Thus we assume for the rest of this section that we are dealing with the problem GENERALIZED GRAPHIC INVERSE VORONOI, where the underlying graph is a tree \(T\).

6.1 Transforming to disjoint cells

In this section we explain how to decrease the overlap between different Voronoi regions by considering one edge of the tree at a time and transforming the instance. When there are no edges to process, we can conclude that the original instance has no solution or we can find a solution to the original instance.

Consider an instance \(I = (T, ((U_1, S_1), \ldots, (U_k, S_k)))\) for the problem GENERALIZED GRAPHIC INVERSE VORONOI, where \(T\) is a tree. See Figure 1 for an example of such an instance. For each index \(i \in [k]\) we define
\[
W_i = U_i \bigcup_{j \neq i} U_j,
E_i = \{uv \in E(T) \mid u \in W_i, v \in U_i \setminus W_i\}.
\]

The intuition is that each \(W_i\) should be the set of vertices in the interior and each \(E_i\) should be the set of edges within the cell with exactly one vertex in the interior. As a preprocessing step, we replace \(S_i\) by \(S_i \cap W_i\) for each \(i \in [k]\). Since a site cannot belong to two Voronoi regions, this replacement does not reduce the set of feasible solutions for \(I\). To simplify notation, we keep using \(I\) for the new instance. We also assume that \(S_i \neq \emptyset\) for \(i \in [k]\). The following result is then easy to prove; see Lemma 2.

**Lemma 15.** (a) If there is a solution to GENERALIZED GRAPHIC INVERSE VORONOI with input \(I\), then each set \(U_i\) (\(i \in [k]\)) induces a connected subgraph of \(T\).

(b) If all sets \(U_i\) induce connected subgraphs of \(T\) and two sets \(U_i\) and \(U_j\) (\(i \neq j\)) intersect, then \(E_i \neq \emptyset\) and \(E_j \neq \emptyset\).

If the sets \(U_1, \ldots, U_k\) are pairwise disjoint, we do not need to do anything. If at least two of them overlap but the sets \(E_1, \ldots, E_k\) are empty, then Lemma 15 implies that there is no solution. In the remaining case some \(E_i\) is nonempty, and we transform the instance as follows.

In the transformations we will need “short” edges. To quantify this, we introduce the \textit{resolution} \(\text{res}(I)\) of an instance \(I\), defined by
\[
\text{res}(I) = \min \left(\mathbb{R}_{>0} \cap \{d_T(s_i, u) - d_T(s_j, u) \mid u \in U_i \cap U_j, s_i \in S_i, s_j \in S_j, i, j \in [k]\} \right).
\]

Here we take the convention that \(\min(\emptyset) = +\infty\). From the definition we have the following property:
\[
\forall i, j \in [k], u \in U_i \cap U_j, s_i \in S_i, s_j \in S_j:
|d_T(s_i, u) - d_T(s_j, u)| < \text{res}(I) \implies d_T(s_i, u) = d_T(s_j, u).
\]
Consider any value \( \epsilon > 0 \). Fix any index \( i \in [k] \) such that \( E_i \neq \emptyset \) and consider an edge \( xy \in E_i \) with \( x \in W_i \) and \( y \in U_1 \setminus W_i \). By renaming the sets, if needed, we assume henceforth that \( i = 1 \), that is, \( E_1 \neq \emptyset \), and \( x \in W_1 \) and \( y \in U_1 \setminus W_1 \). We build a tree \( T' \) with edge-lengths \( \lambda' \) and a new set \( U'_1 \) as follows. We obtain \( T' \) from \( T \) by subdividing \( xy \) with a new vertex \( y' \). We define \( U'_1 \) to be the subset of vertices of \( U_1 \) that belong to the component of \( T - y \) that contains \( x \), and then we also add \( y' \) into \( U'_1 \). Note that \( u \in U_1 \) belongs to \( U'_1 \) if and only if \( d_T(u, x) < d_T(u, y) \). In particular \( y \notin U'_1 \). Finally, we set the edge-lengths \( \lambda'(xy') = \lambda(xy) \) and \( \lambda'(yy') = \epsilon \), and the remaining edges have the same length as in \( T \). This completes the description of the transformation. Note that \( T' \) is just a subdivision of \( T \) and, effectively, the edge \( xy \) became a 2-path \( xy'y' \) that is longer by \( \epsilon \). All distances in \( T' \) are larger or equal than in \( T \), and the difference is at most \( \epsilon \).

Let \( I' \) be the new instance, where we use \( T', \lambda' \) and \( U'_1 \), instead of \( T, \lambda \) and \( U_1 \), respectively. (We leave \( U_i \) unchanged for each \( i \in [k] \setminus \{1\} \) and we leave \( S_i \) unchanged for each \( i \in [k] \).) See Figure 10 for two examples of this transformation and Figure 11 for a schematic view. We call \( I' \) the instance obtained from \( I \) by expanding the edge \( xy \) from \( E_i \) by \( \epsilon \). Note that \( y' \) is not a valid placement for a site in \( I' \), since \( y' \notin S_1 \).

Our definition of \( \text{res}(I) \) is carefully chosen so that it does not decrease with the expansion of an edge. That is, \( \text{res}(I') \geq \text{res}(I) \). This is an important but subtle point needed to achieve efficiency. It will permit that all the short edges that are introduced during the transformations have the same small length \( \epsilon \), and we will be able to treat \( \epsilon \) symbolically.

The next two lemmas show the relation between solutions to the instances \( I \) and \( I' \).

**Lemma 16.** Suppose that \( \epsilon > 0 \). If \( S \) is a solution to \( \text{GENERALIZED GRAPHIC INVERSE VORONOI with input } I \), then \( S \) is also a solution to \( \text{GENERALIZED GRAPHIC INVERSE VORONOI with input } I' \).

**Proof.** Consider a solution \( s_1, \ldots, s_k \) to \( \text{GENERALIZED GRAPHIC INVERSE VORONOI with input } I \), and define \( S = \{ s_1, \ldots, s_k \} \). This means that, for all \( i \in [k] \), we have \( s_i \in S_i \) and \( U_i = \text{cell}_T(s_i, S) \). Our objective is to show that \( U'_1 = \text{cell}_{T'}(s_1, S) \) and \( U_i = \text{cell}_{T'}(s_i, S) \) for all \( i \in [k] \setminus \{1\} \).

Clearly, \( W_1 = \text{cell}_{T'}(s_1, S) \) and therefore \( s_1 \in W_1 \). Moreover, \( xy \) is an edge connecting \( x \in \text{cell}_{T'}(s_1, S) \) to \( y \in U_1 \setminus W_1 = \text{cell}_{T}(s_1, S) \setminus \text{cell}_{T'}(s_1, S) \). Because of Lemma 2, each path starting at \( W_1 \) consists of a path contained in \( W_1 \), followed by a path contained \( U_1 \setminus W_1 \), and followed by a path contained in \( V(T) \setminus U_1 \), where the last or the last two parts may be empty. From this structure, \( x \in W_1 \) and \( y \in U_1 \setminus W_1 \), we deduce that \( W_1 \) is contained in the component of \( T' - y \) that contains \( U'_1 \), and in particular \( W_1 \subseteq U'_1 \).

Let \( V_x \) be the vertex set of the component of \( T' - y' \) that contains \( x \) and let \( V_y \) be the vertex set of the component of \( T' - y' \) that contains \( y \). See Figure 11. Let \( S_x = S \cap V_x \) and \( S_y = S \cap V_y \).
We conclude that we obtain that $U \subseteq V_x \cap V_y$. From the definition of $U'$ we have $U' = \{ y' \} \cup (V_x \cap U_1)$. Moreover, for each $j \in [k] \setminus \{ 1 \}$, we have $x \notin U_j$ because $x \in W_1$, and Lemma 2 implies that the set $U_j$ is fully contained either in $V_x$ or in $V_y$. Now we have the following easy relations between distances in $T$ and $T'$; we will use them often without explicit reference.

\[
\forall u, v \in V_x : \quad d_T(u, v) = d_T(u, v) \\
\forall u, v \in V'_x : \quad d_T(u, v) = d_T(u, v) \\
\forall u \in V_x, v \in V'_y : \quad d_T(u, v) = d_T(u, v) + \epsilon \\
\forall u \in V'_x : \quad d_T(u, y') = d_T(u, y) \\
\forall u \in V'_y : \quad d_T(u, y') = d_T(u, y) + \epsilon.
\]

Consider any index $\ell \in [k] \setminus \{ 1 \}$ such that $y \in U_1 \cap U_\ell$. It cannot be that $U_\ell$ is contained in $V_x$ because $x \in \text{cell}_T(s_1, S)$ would imply $d_T(s_1, y) < d_T(s_1, y)$. This means that $U_\ell \subset V_y$ and, in particular, $s_\ell \in S_y$.

We first note that the sets $U'_1, U'_2, \ldots, U'_k$ cover $V(T')$. Indeed, since $y \in U_1 \cap U_\ell$ for some index $\ell \in [k] \setminus \{ 1 \}$, the sites $s_1$ and $s_\ell$ are closest sites to $y$ in $T$, and using that $s_1 \in V_x$ and $s_\ell \in V_y$, we obtain that $U_1 \setminus U'_1$ is contained in $U_\ell$. Since $U_1, \ldots, U_k$ cover $V(T)$, $y' \in U'_1$ by construction, and $V(T') = V(T) \cup \{ y' \}$, we conclude that indeed $U'_1, U'_2, \ldots, U'_k$ cover $V(T')$.

First we make the following two claims.

**Claim 16.1.** $y' \in \text{cell}_{T'}(s_1, S)$ and $y' \notin \text{cell}_{T'}(s_1, S)$ for any $i \in [k] \setminus \{ 1 \}$.

**Proof.** Fix any index $i \in [k] \setminus \{ 1 \}$. Consider first the case when $s_i \in S_x$. In this case the path from $s_i$ to $y'$ passes through $x$, which is a vertex in $\text{cell}_{T'}(s_1, S)$. If follows that $d_{T'}(s_1, x) < d_{T'}(s_1, x)$, which implies

\[
d_{T'}(s_1, y') = d_{T'}(s_1, y) < d_{T'}(s_1, y) = d_{T'}(s_1, y).
\]

Consider now the case when $s_i \in S_y$. Because $y \in U_1 = \text{cell}(s_1, S)$, we have $d_{T'}(s_1, y) \leq d_{T'}(s_1, y)$ and we conclude that

\[
d_{T'}(s_1, y') = d_{T'}(s_1, y) + \epsilon > d_{T'}(s_1, y) + \epsilon = d_{T'}(s_1, y') + \epsilon = d_{T'}(s_1, y') + \epsilon > d_{T'}(s_1, y').
\]

In each case we get $d_{T'}(s_1, y') < d_{T'}(s_1, y')$, and the claim follows. \hfill \Box

**Claim 16.2.** $y \notin \text{cell}_{T'}(s_1, S)$.

**Proof.** Since $y$ belongs to $U_1 \cap U_\ell$, for some index $\ell \in [k] \setminus \{ 1 \}$, we have $d_{T'}(s_1, y) = d_{T'}(s_1, y)$. Using that $U_\ell$ is contained in $V_y$, and thus $s_\ell \in V_y$, we have

\[
d_{T'}(s_1, y) = d_{T'}(s_1, y) = d_{T'}(s_1, y) = d_{T'}(s_1, y) = d_{T'}(s_1, y) + \epsilon < d_{T'}(s_1, y).
\]

We conclude that $y$ is not an element of $\text{cell}_{T'}(s_1, S)$. \hfill \Box

---

Figure 11: Notation in the proof of Lemma 16.
Claims 16.1 and 16.2 imply that \( y' \) belongs only to the Voronoi region \( \text{cell}_T(s_1, S) \) and \( y \) does not belong to \( \text{cell}_T(s_1, S) \). This means that each vertex of \( V_x \) belongs only to some regions \( \text{cell}_T(s_i, S) \) with \( s_i \in S_x \) and each vertex \( V_y \) belongs to some regions \( \text{cell}_T(s_i, S) \) with \( s_i \in S_y \). That is, it cannot be that some vertex \( u \in V_x \) belongs to \( \text{cell}_T(s_i, S) \) with \( s_i \in S_y \) and it cannot be that some vertex \( u \in V_y \) belongs to \( \text{cell}_T(s_i, S) \) with \( s_i \in S_x \). Effectively, this means that \( y' \) splits the Voronoi diagram \( V_T(S) \) into the part within \( T'[V_x] \) and the part within \( T'[V_y] \), with the gluing property that \( y' \in \text{cell}_T(s_1, S) \). Since \( U_1' \setminus \{y'\} = U_1 \cap V_x \) and the distances within \( T'[V_x] \) and within \( T'[V_y] \) are the same as in \( T \), the result follows.

The converse property is more complicated. We need \( \epsilon \) to be small enough and we also have to assume that \( I \) has a solution. It is this tiny technicality that makes the reduction nontrivial.

**Lemma 17.** Suppose that \( 0 < \epsilon < \text{res}(I) \) and the answer to \textbf{Generalized Graphic Inverse Voronoi} with input \( I \) is “yes”. If \( S' \) is a solution to \textbf{Generalized Graphic Inverse Voronoi} with input \( I' \), then \( S' \) is also a solution to \textbf{Generalized Graphic Inverse Voronoi} with input \( I \).

**Proof.** When the instance \( I \) has some solution, then the properties discussed in Lemmas 15 and 16 hold. We keep using the notation and the properties established earlier. In particular, each set \( U_i \) \((i \in [k] \setminus \{1\})\) is contained either in \( V_x \) or in \( V_y \), and the set \( U_1' \) is contained in \( V_y \).

Consider a solution \( s_1, \ldots, s_k \) to \textbf{Generalized Graphic Inverse Voronoi} with input \( I' \), and set \( S = \{s_1, \ldots, s_k\} \). This means that \( U_1' = \text{cell}_T(s_1, S) \) and, for all \( i \in [k] \setminus \{1\} \), we have \( U_i = \text{cell}_T(s_i, S) \). We have to show that, for all \( i \in [k] \), we have \( U_i = \text{cell}_T(s_i, S) \), which implies that \( S \) is a solution to input \( I \).

Like before, we split the proof into claims that show that \( S \) is a solution to \textbf{Generalized Graphic Inverse Voronoi} with input \( I \). We start with an auxiliary property that plays a key role.

**Claim 17.1.** For each \( i \in [k] \), we have \( y \in U_i \) if and only if \( y \in \text{cell}_{T'}(s_i, S) \).

**Proof.** Suppose first that \( y \in U_i \) and \( i \neq 1 \). Then \( U_i \subseteq V_y \). Since \( y \in U_i = \text{cell}_{T'}(s_i, S) \) and \( y \notin U_1' = \text{cell}_{T'}(s_1, S) \), we have

\[
d_T(s_i, y) = d_{T'}(s_i, y) < d_{T'}(s_1, y) = d_T(s_1, y) + \epsilon
\]

(1)

Since \( y' \notin U_i = \text{cell}_{T'}(s_i, S) \) and \( y' \in U_1' = \text{cell}_{T'}(s_1, S) \), we have

\[
d_T(s_1, y') = d_{T'}(s_1, y') < d_{T'}(s_i, y) = d_T(s_i, y) + \epsilon
\]

(2)

Joining (1) and (2) we get

\[
d_T(s_i, y) < d_{T'}(s_i, y) + \epsilon < d_{T'}(s_i, y) + 2\epsilon,
\]

or equivalently

\[
|d_T(s_i, y) - d_{T'}(s_i, y)| < \epsilon < \text{res}(I)
\]

From the definition of \( \text{res}(I) \) and since \( y \in U_1 \cap U_i \), we conclude that \( d_T(s_1, y) = d_T(s_i, y) \). For each \( s_j \in S_y \) we use that \( y \in U_i = \text{cell}_{T'}(s_i, S) \) to obtain

\[
d_T(s_j, y) = d_{T'}(s_j, y) \geq d_{T'}(s_i, y) = d_T(s_i, y).
\]

For each \( s_j \in S_x \) we use that the path from \( s_j \) to \( y \) goes through \( x \in \text{cell}_{T'}(s_1, S) \) to obtain

\[
d_T(s_j, y) \geq d_T(s_1, y) = d_T(s_i, y).
\]

We conclude that for each \( j \in [k] \) we have \( d_T(s_j, y) \geq d_T(s_i, y) \), and therefore \( y \in \text{cell}_{T'}(s_1, S) \).

Since \( d_T(s_1, y) = d_T(s_j, y) \) whenever \( y \in U_1 \cap U_i \), and \( y \in U_i \) for some \( i \in [k] \setminus \{1\} \), we also obtain \( y \in \text{cell}_{T'}(s_1, S) \). With this we have shown one direction of the implication.
We then have

\[ d_T(s_i, y) = d_T(s_i, y) = d_T(s_i, y) = d_T(s_i, y). \]

Since \( d_T(s_i, y) = d_T(s_i, y) \) and \( y \in cell_T(s_i, S) \), we conclude that \( y \in cell_T(s_i, S) = U_i. \)

**Claim 17.2.** \( x \in cell_T(s_i, S) \) and \( x \notin cell_T(s_i, S) \) for any \( i \in [k] \setminus \{1\}. \)

**Proof.** Since \( x \in U_1^* = cell_T(s_i, S) \) and \( x \notin U_i = cell_T(s_i, S) \) for any \( i \in [k] \setminus \{1\} \), we have

\[ \forall i \in [k] \setminus \{1\}: \quad d_T(s_i, x) < d_T(s_i, x). \]

We then have

\[ \forall s_i \in S_x, s_i \neq s_i: \quad d_T(s_i, x) = d_T(s_i, x) < d_T(s_i, x) = d_T(s_i, x). \quad (3) \]

For each \( s_i \in S_y \), note that the path from \( s_i \) to \( x \) passes through \( y \), and \( y \in cell_T(s_i, S) \) because of Claim 17.1. Using that \( s_i \in V_\ell \), we have

\[ \forall s_i \in S_y: \quad d_T(s_i, x) < d_T(s_i, x). \quad (4) \]

Joining (3) and (4), the claim follows.

**Claim 17.3.** For each \( u \in V_y \), we have \( u \in U_1 \) if and only if \( u \in cell_T(s_1, S) \).

**Proof.** Consider some solution \( s_1^*, \ldots, s_k^* \) to **Generalized Graphic Inverse Voronoi** with input \( I \), and set \( S^* = \{s_1^*, \ldots, s_k^*\} \). This means that \( U_i = cell_T(s_i^*, S^*) \) for each \( i \in [k] \). We also fix an index \( \ell \in [k] \setminus \{1\} \) such that \( y \in U_\ell \cap U_1 \). Recall that \( U_\ell \subseteq V_y \) because \( x \notin U_\ell \). Using Claim 17.1 and using that \( S^* \) is a solution to \( I \) we have

\[ d_T(s_1, y) = d_T(s_1, y) \quad \text{and} \quad d_T(s_1^*, y) = d_T(s_1^*, y). \quad (5) \]

Consider some \( u \in U_1 \cap V_y \). We will show that \( u \in cell_T(s_1, S) \). Consider the subtree \( \hat{T} \) defined by the paths connecting the vertices \( s_1^*, s_1, s_1^*, u \). See Figure 12. The path from \( u \in V_y \) to \( s_1 \) attaches to the path from \( s_1^* \) to \( y \) at the vertex \( y \). Indeed, if it would attach through another vertex \( a \neq y \), then we would have \( d_T(s_1^*, a) < d_T(s_1^*, a) \) because of (5), which would imply \( d_T(s_1^*, a) < d_T(s_1^*, u) \), contradicting the assumption that \( u \in cell_T(s_1^*, S^*) = U_1 \). Similarly, we see that the \((u, s_1)\)-path attaches to the \((s_1, y)\)-path at the vertex \( y \).

![Figure 12: Situation in the proof of Claim 17.3.](image-url)
Since each path from $s_1, s^*_1, s_\ell$ and $s^*_\ell$ to $u$ passes through $y$, from (5) we get

$$d_T(s_1, u) = d_T(s_\ell, u) \quad \text{and} \quad d_T(s^*_1, u) = d_T(s^*_\ell, u). \quad (6)$$

Together with $u \in U_1 = \text{cell}_T(s^*_1, S^*)$ we conclude that $u \in \text{cell}_T(s^*_\ell, S^*) = U_\ell$. Since $u \in U_\ell = \text{cell}_T(s_\ell, S)$ we have

$$\forall s_j \in S_y : \quad d_T(s_1, u) = d_T(s_\ell, u) \leq d_T(s_j, u).$$

Together with the fact that each $s_j \in S_y$ is farther from $u$ than $s_1$ because $x \in \text{cell}_T(s_1, S)$, we conclude that $u \in \text{cell}_T(s_1, S)$. This finishes the left-to-right direction of the implication.

Consider now a vertex $u \in V_y \cap \text{cell}_T(s_1, S)$. Since $y$ is on the path from $s_1$ to $u$, we obtain from (5) that $d_T(s_1, u) \leq d_T(s_\ell, u)$, and therefore $u \in \text{cell}_T(s_\ell, S)$. Because $u \in V_y$, $d_T(s_\ell, u) = d_T(s_\ell, u)$, and distances in $T'$ can be only larger than in $T$, we have $u \in \text{cell}_T(s^*_\ell, S^*) = U_\ell = \text{cell}_T(s^*_\ell, S^*)$. This means that

$$\forall i \in [k] : \quad d_T(s^*_i, u) \leq d_T(s^*_1, u). \quad (7)$$

It cannot be that $y$ lies on the path in $T$ from $s_\ell$ to $s^*_\ell$ because $y \in U_1$, while the path connecting $s_\ell$ and $s^*_\ell$ must be contained in $W_\ell$ (Lemma 2). This means that $y$ is on the path from $s^*_\ell$ to $u$. Since $y$ is also on the path from $s^*_1$ to $u$, we get from (5) and (7) that

$$\forall i \in [k] : \quad d_T(s^*_i, u) = d_T(s^*_1, u) \leq d_T(s^*_\ell, u).$$

Together with $u \in U_\ell = \text{cell}_T(s^*_1, S^*)$ we obtain that $u \in \text{cell}_T(s^*_1, S^*) = U_1$. This finishes the proof of the claim.

We are now ready to prove Lemma 17: for all $i \in [k]$ we have $U_i = \text{cell}_T(s_i, S)$. Because of Claim 17.2, $x$ belongs only to the Voronoi cell $\text{cell}_T(s_1, S)$, This means that the Voronoi cells restricted to $V_x$ are the same in $T$ and in $T'$. Therefore, we have $U_i = \text{cell}_T(s_i, S) = \text{cell}_T(s_1, S)$ for all $i$ with $U_i$ contained in $V_x$. Because of Claim 17.3, we have $U_1 = \text{cell}_T(s_1, S)$. Indeed, Claim 17.3 takes care for the vertices of $U_1 \cap V_y$, while $d_T|_{[V_x]} = d_T|_{V'_x}$ and $U'_1 \setminus \{y'\} = U_1 \cap V_x$, takes care of the vertices in $U_1 \cap V_y$. For the indices $i$ for which $U_i$ is contained in $V_y$, we have to consider the possibility that $s_1$ may affect $\text{cell}_T(s_i, S)$. However, since $d_T(s_1, y) = d_T(s_1, y)$ for some index $\ell \in [k] \setminus \{1\}$ with $y \in U_\ell \subseteq V_y$ (Claim 17.1), each vertex $u \in V_y$ that belongs to cell$_T(s_1, S)$ also belongs to cell$_T(s_\ell, S)$. This means that whether $u \in V_y$ belongs to a cell cell$_T(s_1, S)$ or not is not affected by $s_1$. It follows that, for all $i$ with $U_i$ contained in $V_y$, we have $U_i = \text{cell}_T(s_i, S) = \text{cell}_T(s_1, S)$. This finishes the proof of the Lemma.

It is important to note that the transformation described above only works for trees. A similar transformation for arbitrary graphs may have feasible solutions that do not correspond to solutions in the original problem. See Figure 13 for a simple example.

---

1. Here it is important that we replaced $S_T$ with $S_x \cap W_\ell$ in the preprocessing step. Without that replacement, the lemma is actually not true because it can happen that $s_\ell \in U_1 \cap U_1$. 

---

Figure 13: A similar transformation for arbitrary graphs does not work. On the right side we have the transformed instance with a feasible solution that does not correspond to a solution in the original setting.
Another important point is that we need the assumption that $I$ had a solution. This means that, any solution $S'$ we obtain after making a sequence of expansions, has to be tested in the original instance. However, if $S'$ is not a valid solution in $I$, then $I$ has no solution.

Consider an instance $I = (T, ((U_1, S_1), \ldots, (U_k, S_k)))$. Set $I_0 = I$ and define, for $t \geq 1$, the instance $I_t$ by transforming $I_{t-1}$ using an expansion of some edge. For all expansions we use the same parameter $\epsilon$. We finish the sequence when we obtain the first instance $\hat{I} = \left(\hat{T}, ((\hat{U}_1, \hat{S}_1), \ldots, (\hat{U}_k, \hat{S}_k))\right)$ such that the sets $\hat{U}_1, \ldots, \hat{U}_k$ are pairwise disjoint. Note that this procedure stops because the number of pairs $(i, j)$ with $U_i \cap U_j \neq \emptyset$ decreases with each expansion. This implies that the number of steps is at most $\binom{k}{2}$. In fact, the number of steps is even smaller.

**Lemma 18.** $\hat{I}$ is reached after at most $k - 1$ edge expansions.

**Proof.** We prove this by induction on $k$. There is nothing to show if $k = 1$. Otherwise, note that the sets $U_i$ in $V_x$ and those in $V_y$ (respectively) give rise to two independent subproblems with $k_x$ and $k_y$ sites (respectively), where $k_x + k_y = k$. By induction, the number of edge expansions is at most $1 + (k_x - 1) + (k_y - 1) = k - 1$.

The next lemma shows that using the same parameter $\epsilon$ for all edge expansions is a correct choice. This is due to our careful definition of resolution $\text{res}(\cdot)$.

**Lemma 19.** Assume that $0 < \epsilon < \text{res}(I)$ and the answer to Generalized Graphic Inverse Voronoi with input $I$ is “yes”. Then $S$ is a solution to Generalized Graphic Inverse Voronoi with input $I$ if and only if $S$ is also a solution to Generalized Graphic Inverse Voronoi with input $\hat{I}$.

**Proof.** Note that, by construction, $\text{res}(I_{t-1}) \leq \text{res}(I_t)$ for all $t \geq 1$. Indeed, when we expand the edge $xy$ inserting $y'$, then there is no set $U_i$ that is on both sides of $T' - y'$. This means that for all the parameters $s_i, s_j, u_i, u_j$ considered in the definition of $\text{res}(I_t)$ we have $d_T(s_i, u) - d_T(s_j, u) = d_T(s_i, u) - d_T(s_j, u)$. Therefore, $\epsilon < \text{res}(I_t)$ for all $t$. The claim now follows easily from Lemmas 16 and 17 by induction on $t$.

### 6.2 Transforming to maximum degree 3

Consider an instance $I = (T, ((U_1, S_1), \ldots, (U_k, S_k)))$ for the problem Generalized Graphic Inverse Voronoi, where $T$ is a tree and the sets $U_1, \ldots, U_k$ are pairwise disjoint. See Figure 14 for an example of such an instance viewed around a vertex of degree $> 3$. We want to transform it into another instance $I' = (T', ((U'_1, S'_1), \ldots, (U'_k, S'_k)))$ where the maximum degree of $T'$ is 3, the sets $U'_1, \ldots, U'_k$ are pairwise disjoint, and a solution to $I'$ corresponds to a solution of $I$.

In the transformations we will need “short” edges again and we use again the resolution of the instance $I$. Since the sets $U_i$ ($i \in [k]$) are pairwise disjoint, we need another version of the resolution:

$$\text{res}'(I) = \min\left(\mathbb{R}_{>0} \cap \{d_T(s_i, u) - d_T(s_j, u) \mid u \in V(T), s_i \in S_i, s_j \in S_j, i, j \in [k]\}\right).$$

From the definition we have the following property:

$$\forall i, j \in [k], u \in V(T), s_i \in S_i, s_j \in S_j :$$

$$d_T(s_i, u) - d_T(s_j, u) \leq \text{res}'(I) \implies d_T(s_i, u) = d_T(s_j, u).$$

We explain how to transform the instance into one where all vertices have maximum degree 3. We will use $T'$ and $\lambda'$ for the new graph and its edge-lengths. For each edge $uv$ of $T$ we place two vertices $a_{u,v}$ and $a_{v,u}$ in $T'$, and connect them with an edge. The length $\lambda'$ of such an edge
For an example of the whole process see Figure 14.

To recover the solutions, we define the projection map \( \pi(a_{uv}) = u \). Thus, \( \pi \) sends each vertex of \( T' \) to the corresponding vertex of \( T \) that was used to create it. Note that for each \( i \in [k] \) we have \( \pi(S'_i) = S_i \) and \( \pi(U'_i) = U_i \).

The distances in \( T' \) and \( T \) are closely related:

\[
\forall u, v \in V(T') : \ d_T(\pi(u), \pi(v)) \leq d_{T'}(u, v) \leq d_T(\pi(u), \pi(v)) + 2n\delta. \tag{8}
\]

In particular, if we take \( \delta < \text{res'}(I)/2n \), then

\[
\forall s'_1, s'_2, u \in V(T') : \ \pi(s'_1) \neq \pi(s'_2) : \ d_{T'}(s'_1, u) < d_{T'}(s'_2, u) \implies d_T(\pi(s'_1), \pi(u)) < d_T(\pi(s'_2), \pi(u)). \tag{9}
\]

**Lemma 20.** Suppose that \( 0 < \delta < \text{res'}(I)/2n \) and the sets \( U_1, \ldots, U_k \) are pairwise disjoint subsets of \( V(T) \). The answer to \( (T, (U_1, S_1), \ldots, (U_k, S_k)) \) is “yes” if and only if the answer to \( (T', ((U'_1, S'_1), \ldots, (U'_k, S'_k))) \) is “yes”.

**Proof.** The “if” part is easier. Suppose that the answer to \( I' \) is “yes”. Then, there exist \( s'_i, \ldots, s'_k \), with \( s'_i \in S'_i \), and \( U'_i = \text{cell}_T(s'_i, \{s'_1, \ldots, s'_k\}) \) for each \( i \in [k] \). Consider any fixed \( i \in [k] \) and any vertex \( u \in U_i \). There exists some vertex \( u' \in U'_i \) such that \( u = \pi(u') \). Set \( s_i = \pi(s'_i) \) for all \( j \in [k] \). By (9), \( d_T(s_i, u) < d_T(s_j, u) \) for each \( j \in [k] \setminus \{i\} \), which implies \( u \in \text{cell}_T(s_i, \{s_1, \ldots, s_k\}) \) and \( u \notin \text{cell}_T(s_j, \{s_1, \ldots, s_k\}) \) for all \( j \in [k] \setminus \{i\} \). It follows that \( U_i = \text{cell}_T(s_i, \{s_1, \ldots, s_k\}) \).

Now we turn to the “only if” part. Suppose that there exist \( s_1, \ldots, s_k \), with \( s_i \in S_i \), and \( U_i = \text{cell}_T(s_i, \{s_1, \ldots, s_k\}) \) for each \( i \in [k] \). Our claim is that if we take a vertex \( s'_i \in \pi^{-1}(s_i) \) for each \( i \in [k] \), then \( U'_i = \text{cell}_T(s'_i, \{s'_1, \ldots, s'_k\}) \). In order to prove that, consider any fixed index \( i \in [k] \) and any vertex \( u' \in U'_i \). Set \( u = \pi(u') \in U_i \), and suppose that the shortest \( (u, s_i) \)-path \( P \) is \( u = v_0, v_1, \ldots, v_r = s_i \). Then, \( d_T(s'_i, u') \) is at most \( \lambda(P) + \delta \sum_{j=0}^r \text{deg}(v_j) - 1 \). Since \( \sum_{j=0}^r \text{deg}(v_j) \leq 2|E(T)| < 2n \), we conclude that

\[
d_T(s'_i, u') < \lambda(P) + 2n\delta \leq \lambda(P) + \text{res'}(I). \tag{10}
\]
Now consider any $j \in [k] \setminus \{i\}$, and suppose the shortest $(u, s_j)$-path $Q$ in $T$ is $u = w_0, w_1, \ldots, w_t = s_j$. Since $s_1, \ldots, s_k$ is a solution for $I$, we have $\lambda(Q) > \lambda(P)$. Then, $d_{T'}(s_j, u')$ is at least $\lambda(Q) \geq \lambda(P) + \text{res}(I)$. From (10) we conclude that $d_{T'}(s'_j, u') < d_{T'}(s'_j, u')$ for all $j \in [k] \setminus \{i\}$. This implies that $u' \in \text{cell}_{T'}(s'_j, \{s'_1, \ldots, s'_k\})$ and $u' \notin \text{cell}_{T'}(s'_j, \{s'_1, \ldots, s'_k\})$ for all $j \in [k] \setminus \{i\}$. It follows that $U'_i = \text{cell}_{T'}(s'_j, \{s'_1, \ldots, s'_k\})$.

6.3 Algorithm to transform

We are now ready to explain algorithmic details of the whole transformation and explain its efficient implementation.

Suppose that we have an instance $I = (T, (U_1, \ldots, U_k))$ for the problem $\text{GRAPHIC INVERSE VORONOI}$, where $T$ is a tree. Let us use $N = N(I) = |V(T)| + \sum_i |U_i|$ for the description size of $I$. As mentioned earlier, we can convert in $O(N)$ time this to an equivalent instance $(T, ((U_1, S_1), \ldots, (U_k, S_k)))$ for the problem $\text{GENERALIZED GRAPHIC INVERSE VORONOI}$. Let $I'$ be this new instance and note that its description size is $O(N)$.

For each vertex $v \in V(T)$ we make a list $L(v)$ that contains the indices $i \in [k]$ with $v \in U_i$. The lists $L(v)$, for all $v \in V(T)$, can be computed in $O(N)$ time by scanning the sets $U_1, \ldots, U_k$: for each $v \in U_i$ we add $i$ to $L(v)$. Note that a vertex $v \in V(T)$ belongs to $W_i$ if and only if $i$ is the only index in the list $L(v)$. With this we can compute the sets $W_1, \ldots, W_k$. Scanning the sets $S_1, \ldots, S_k$, we can replace each set $S_i$ with the set $S_i \cap W_i$. Together we have spent $O(N)$ time and we have made the preprocessing step described before Lemma 15.

We can now mark each edge of $T$ that belongs to some set $E_i$ in linear time, as follows. First, we root the tree $T$ at an arbitrary vertex $r$ and store for each vertex $v$ of $T$ its parent node. (The parent of $r$ is set to null.) We add to each vertex a flag to indicate whether it belongs to the set $U_i$ under consideration. Initially all flags are set to false. This takes $O(N)$ time.

Then we iterate over $i \in [k]$; we describe the work for a fixed index $i \in [k]$. First we change the flag of each vertex $u \in U_i$ to true. Then we consider the edge $xy$ for each $x \in U_i$, where $y$ is the parent of $u$. If the flag of $x$ is true, $L(x)$ has a single element and $L(y)$ has more than one element, then $x \in W_i$, $y \in U_i \setminus W_i$, and $xy \in E_i$. Similarly, if the flag of $y$ is true, $L(y)$ has a single element and $L(x)$ has more than one element, then $y \in W_i$, $x \in U_i \setminus W_i$, and $xy \in E_i$. Note that each edge $xy \in E_i$ is detected in this way because $x$ and $y$ have to be in a parent-child relation. Finally, we set the flags of vertices of $U_i$ back to false, and proceed to the next iteration. It is clear that this procedure takes time $O(|U_i|)$ for each $i \in [k]$, and thus it takes $O(N)$ time in total.

Now we can make the expansions of the edges. Assume for the time being that $\epsilon$ is already known. We will discuss its choice below. We iterate over the indices $i \in [k]$ and consider $E_i$. We mark the vertices of $U_i$ in $T$ using the flags of each vertex. For each $u \in U_i$, we store a list with its children in $T[U_i]$, the subgraph of $T$ induced by $U_i$. For each $xy \in E_i$, with $x \in U_i$ and $y \in U_i \setminus W_i$, we make the expansion as follows: edit $T$ by inserting $y'$, set the new edge-lengths for the edges $yy'$ and $xy'$, remove from $U_i$ the subset $R_{xy}$ of elements of $U_i$ that are closer to $y$ than to $x$, and insert $y'$ in $U_i$. If $y$ is a child of $x$, the set $R_{xy}$ of elements to be removed from $U_i$ can be obtained as the descendants of $y$ in $T[U_i]$. If $x$ is a child of $y$, the set $R_{xy}$ of elements to be removed from $U_i$ can be obtained using the descendants of the ancestors of $y$ in $T[U_i]$. In both cases, we identify $R_{xy}$ in time $O(|R_{xy}|)$. We conclude that expanding an edge $xy \in E_i$ takes $O(1 + |R_{xy}|)$. Since each element of $U_i$ can be deleted at most once from $U_i$, and the elements $y'$ we insert cannot be deleted because they belong only to (the new) $U_i$, the expansions for the edges in $E_i$ take $O(1 + |U_i|)$ time all together. Thus, all the expansions required for Lemma 19 can be carried out in $O(N)$ time, assuming the value $\epsilon$ is available. Let $\tilde{I}$ be the resulting instance with the disjoint sets.

Now we can make the transformation from $\tilde{I}$ to an instance with maximum degree 3. Assume for the time being that we have the parameter $\Delta$ available. Then the transformation described
in Section 6.2 can be easily carried out in linear time. Thus, in $O(N)$ time we obtain the final instance with pairwise disjoint sets $U_1, \ldots, U_k$ and tree $T$ of maximum degree 3.

It remains to discuss how to choose the values of $\varepsilon$ and $\delta$ for the transformations. It is unclear whether $\varepsilon$ or $\delta$ can be computed in $O(N)$ time when the edges have arbitrary lengths. (If, for example, all edges have integral lengths, then we could take $\varepsilon = 1/4$ and $\delta = 1/10n$.) We will handle this using composite lengths. The length of each edge $e$ is going to be described by a triple $(a, b, c)$ that represents the number $a + b\varepsilon + c\delta$ for infinitesimals $\delta \ll \varepsilon$. Thus the length encoded by $(a, b, c)$ is smaller than the length encoded by $(a', b', c')$ if and only if $(a, b, c)$ is lexicographically smaller than $(a', b', c')$. In the original graph we replace the length of each edge $e$ by $(\lambda(e), 0, 0)$. In the expansion, the new edges $yy'$ get length $(0, 1, 0)$, and in converting the tree to maximum degree 3 we use edges of length $(0, 0, 1)$. The length of a path becomes a triple $(a, b, c)$ that is obtained as the vector sum of the triples over its edges. Each comparison and addition of edge-lengths costs $O(1)$ time. We summarize.

**Theorem 21.** Suppose that we are given an instance $I$ for the problem **Graphic Inverse Voronoi** or for the problem **Generalized Graphic Inverse Voronoi** of description size $N = N(I)$ over a tree $T$. In $O(N)$ time we can either detect that $I$ has no solutions, or construct another instance $I'$ for the problem **Generalized Graphic Inverse Voronoi** over a tree $T'$ with the following properties:

- the tree $T'$ in the instance $I'$ has maximum degree 3,
- the sets in the instance $I'$ are pairwise disjoint,
- if the answer to $I$ is “yes”, then any solution to $I'$ is also a solution to $I$.

## 7 Algorithm for subcubic trees with disjoint Voronoi cells

In this section we consider the problem **Generalized Graphic Inverse Voronoi** for an input $(T, U)$, with the following properties:

- $T$ is a tree of maximum degree 3
- $U$ is a sequence of pairs $(U_1, S_1), \ldots, (U_k, S_k)$ where the sets $U_1, \ldots, U_k$ are pairwise disjoint.

Our task is to find sites $s_1, \ldots, s_k$ such that, for each $i \in [k]$, we have $U_i = \text{cell}_T(s_i, \{s_1, \ldots, s_k\})$ and $s_i \in S_i$. We may assume that $V(T) = \bigcup_{i \in [k]} U_i$, that $T[U_i]$ is connected for each $i \in [k]$, and that $S_i \subseteq U_i$ for each $i \in [k]$, as otherwise it is clear that there is no solution. These conditions can easily be checked in linear time.

First, we describe an approach to decide whether there is a solution without paying much attention to the running time. Then, we describe its efficient implementation taking time $O(N \log^2 N)$, where $N$ is the description size of the instance.

### 7.1 Characterization

For each vertex $v$, let $i(v)$ be the unique index such that $v \in U_{i(v)}$. We choose a leaf $r$ of $T$ as a root and henceforth consider the tree $T$ rooted at $r$. We do this so that each vertex of $T$ has at most two children. For each vertex $v$ of $T$, let $T(v)$ be the subtree of $T$ rooted at $v$, and define also

$$J(v) = \{ j \in [k] | U_j \cap T(v) \neq \emptyset \}.$$

Note that $i(v) \in J(v)$. Since each $U_j$ defines a connected subset of $T(v)$, for each $j \in J(v)$, $j \neq i(v)$, we have $U_j \subset T(v)$ and therefore it must be that $s_j \in T(v)$.
Consider a fixed vertex \( v \) of \( T \) and the corresponding subtree \( T(v) \). We want to parameterize possible distances from \( v \) to the site \( s_{i(v)} \), that is, the site whose cell contains the vertex \( v \), that provide the desired Voronoi diagram restricted to \( T(v) \). A more careful description is below. We distinguish possible placements of \( s_{i(v)} \) within \( T(v) \), which we refer as “below” (or on) \( v \) and for which we use the notation \( B(v) \), and possible placements outside \( T(v) \), which we refer as “above” and for which we use the notation \( A(v) \).

First we deal with the placements where \( s_{i(v)} \) is “below” \( v \). In this case we start defining \( X(v) \) as the set of tuples \( (s_j)_{j \in J(v)} \) that satisfy the following two conditions:

\[
\begin{align*}
\forall j \in J(v) : & \quad s_j \in S_j, \\
\forall j \in J(v) : & \quad \text{cell}_{T(v)}(s_j, \{s_t \mid t \in J(v)\}) \cap T(v) = U_j \cap T(v).
\end{align*}
\]

Note that \( X(v) \subseteq \prod_{j \in J(v)} S_j \). Finally, we define

\[
B(v) = \{ d_T(s_{i(v)}, v) \mid (s_j)_{j \in J(v)} \in X(v) \}.
\]

The set \( B(v) \) represents the valid distances at which we can place \( s_{i(v)} \) inside \( T(v) \) such that \( s_{i(v)} \) is the closest site to \( v \), and still complete the rest of the placements of the sites to get the correct portion of \( U \) inside \( T(v) \).

Now we deal with the placements “above” \( v \). For \( \alpha > 0 \), let \( T_a^+(v) \) be the tree obtained from \( T(v) \) by adding an edge \( v v_{\text{new}} \), where \( v_{\text{new}} \) is a new vertex, and setting the length of \( v v_{\text{new}} \) to \( \alpha \). The role of \( v_{\text{new}} \) is the placement of the site closest to \( v \), when it is outside \( T(v) \). See Figure 15. In the following discussion we use also Voronoi diagrams with respect to \( T_a^+(v) \). Let \( Y_a(v) \) be the set of tuples \( (s_j)_{j \in J(v)} \) that satisfy all of the following conditions:

\[
\begin{align*}
s_{i(v)} = v_{\text{new}}, \\
\forall j \in J(v) \setminus \{i(v)\} : & \quad s_j \in S_j, \\
\forall j \in J(v) : & \quad \text{cell}_{T_a^+(v)}(s_j, \{s_t \mid t \in J(v)\}) \cap T(v) = U_j \cap T(v).
\end{align*}
\]

Finally we define

\[
A(v) = \{ \alpha \in \mathbb{R}_{>0} \mid Y_a(v) \neq \emptyset \}.
\]

We are interested in deciding whether \( B(r) \) is nonempty. Indeed, for the root \( r \) we have \( J(r) = [k] \) and \( T(r) = T \) by construction. The definition of \( X(v) \) implies that \( B(r) \) is nonempty if and only if there is some tuple \( (s_1, \ldots, s_k) \in S_1 \times \cdots \times S_k \) such that

\[
\forall i \in J(r) = [k] : \quad \text{cell}_{T}(s_i, \{s_1, \ldots, s_k\}) = \text{cell}_{T(r)}(s_i, \{s_1, \ldots, s_k\}) = U_i \cap T(r) = U_i.
\]
This is precisely the condition we have to check to solve \textsc{Generalized Graphic Inverse Voronoi}.

We are going to compute $A(v)$ and $B(v)$ bottom-up along the tree $T$. If $v$ is leaf of $T$, then $J(v) = \{i(v)\}$ and clearly we have

$$A(v) = \mathbb{R}_{>0} \quad \text{and} \quad B(v) = \begin{cases} \{0\} & \text{if } v \in S_{i(v)}, \\ \emptyset & \text{if } v \notin S_{i(v)}. \end{cases}$$

Consider now a vertex $v$ of $T$ that has two children $v_1$ and $v_2$. Assume that we already have $A(v_j)$ and $B(v_j)$ for $j = 1, 2$. For $j = 1, 2$ define the sets

$$A'(v_j) = \{x - \lambda(vv_j) \mid x \in A(v_j)\},$$

$$B'(v_j) = \{x + \lambda(vv_j) \mid x \in B(v_j)\},$$

$$C'(v_j) = \{\alpha \mid \exists x \in B(v_j) \text{ such that } x - \lambda(vv_j) < \alpha < x + \lambda(vv_j)\}.$$

This is the offset we obtain when we take into account the length of the edge $vv_j$. The set $C'(v_j)$ will be relevant for the case when $i(v) \neq i(v_j)$. The following lemmas show how to compute $A(v)$ and $B(v)$ from its children. Figure 16 is useful to understand the different cases.

**Lemma 22.** If the vertex $v$ has two children $v_1$ and $v_2$, then

$$A(v) = \mathbb{R}_{>0} \cap \left\{ A'(v_1) \cap A'(v_2) \mid \begin{array}{ll} A'(v_1) \cap A'(v_2) & \text{if } i(v) = i(v_1) = i(v_2), \\ A'(v_1) \cap C'(v_1) & \text{if } i(v) = i(v_1) \neq i(v_2), \\ A'(v_2) \cap C'(v_2) & \text{if } i(v) = i(v_2) \neq i(v_1), \\ C'(v_1) \cap C'(v_2) & \text{if } i(v) \neq i(v_1) \text{ and } i(v) \neq i(v_2). \end{array} \right\}$$

**Proof.** This is a standard proof in dynamic programming. We only point out the main insight showing the role of $A'(v_j)$ and $C'(v_j)$ for $j \in \{1, 2\}$.

When $i(v) = i(v_j)$, placing $s_{i(v)}$ at $v_{\text{new}}$ of the tree $T^{+}_{\alpha}(v)$ is the same as placing it at $v_{\text{new}}$ of $T^{+}_{\alpha + \lambda(vv_j)}(v_j)$. The valid values $\alpha$ for $T^{+}_{\alpha + \lambda(vv_j)}(v_j)$ are described by $A'(v_j)$, a shifted version of $A(v_j)$.

When $i(v) \neq i(v_j)$, there has to be a compatible placement of $s_{i(v_j)}$ inside $T(v_j)$ such that $v$ is closer to $s_{i(v)}$ than to $s_{i(v_j)}$, while $v_j$ is closer to $s_{i(v_j)}$ than to $s_{i(v)}$. That is, we must have

$$d_T(v_{\text{new}}, v) < d_T(s_{i(v_j)}, v) \quad \text{and} \quad d_T(s_{i(v_j)}, v_j) < d_T(v_{\text{new}}, v_j),$$

or equivalently, $\alpha$ must satisfy

$$\alpha < d_T(s_{i(v_j)}, v_j) + \lambda(vv_j) \quad \text{and} \quad d_T(s_{i(v_j)}, v_j) < \alpha + \lambda(vv_j).$$
Thus, each possible value \( x \) of \( d_T(s_{i(v)}, v_j) \), that is, each \( x \in B(v_j) \), gives the interval \( (x - \lambda(vv_j), x + \lambda(vv_j)) \) of possible values for \( \alpha \). The union of these intervals over \( x \in B(v_j) \) is precisely \( C'(v_j) \).

To construct \( B(v) \) it is useful to have a function that tells whether \( v \) is a valid placement for \( s_{i(v)} \). For this matter we define the following function:

\[
\chi(v) = \begin{cases} 
\{0\} & \text{if } i(v) = i(v_1) = i(v_2), v \in S_{i(v)} \text{, } 0 \in A'(v_1) \text{ and } 0 \in A'(v_2), \\
\{0\} & \text{if } i(v) = i(v_1) \neq i(v_2), v \in S_{i(v)} \text{, } 0 \in A'(v_1) \text{ and } 0 \in C'(v_2), \\
\{0\} & \text{if } i(v) = i(v_2) \neq i(v_1), v \in S_{i(v)} \text{, } 0 \in A'(v_2) \text{ and } 0 \in C'(v_1), \\
\{0\} & \text{if } i(v) \neq i(v_1), i(v) \neq i(v_2), v \in S_{i(v)} \text{, } 0 \in C'(v_1) \text{ and } 0 \in C'(v_2), \\
\emptyset & \text{otherwise.}
\end{cases}
\]

**Lemma 23.** If the vertex \( v \) has two children \( v_1 \) and \( v_2 \), then

\[
B(v) = \chi(v) \cup \begin{cases} 
(B'(v_1) \cap A'(v_2)) \cup (B'(v_2) \cap A'(v_1)) & \text{if } i(v) = i(v_1) = i(v_2), \\
B'(v_1) \cap C'(v_2) & \text{if } i(v) = i(v_1) \neq i(v_2), \\
B'(v_2) \cap C'(v_1) & \text{if } i(v) = i(v_2) \neq i(v_1), \\
\emptyset & \text{if } i(v) \neq i(v_1) \text{ and } i(v) \neq i(v_2).
\end{cases}
\]

**Proof.** First we note that \( \chi(v) = \{0\} \) if and only if \( v \) is a valid placement for \( s_{i(v)} \). Indeed, the formula is the same that was used for \( A(v) \), but for the value \( \alpha = 0 \), and it takes into account whether \( v \in S_{i(v)} \).

The proof for the correctness of \( B(v) \) is again based in standard dynamic programming. The case for \( s_{i(v)} \) being placed at \( v \) is covered by \( \chi(v) \). The main insight for the case when \( s_{i(v)} \) is placed in \( T(v_1) \) is that, from the perspective of the other child, \( v_2 \), the vertex is placed “above” \( v_2 \). That is, only the distance from \( s_{i(v)} \) to \( v_2 \) is relevant. Thus, we have to combine \( A'(v_1) \) and \( B'(v_2) \), with the appropriate shifts. More precisely, for \( v_2 \) we have to use \( B'(v_2) \) or \( C'(v_2) \) depending on whether \( i(v_2) = i(v) \) or \( i(v_2) \neq i(v) \).

When \( v \) has a unique child \( v' \), then the formulas are simpler and the argumentation is similar. We state them for the sake of completeness without discussing their proof.

\[
A(v) = \mathbb{R}_{>0} \cap \begin{cases} 
A'(v') & \text{if } i(v) = i(v'), \\
C'(v') & \text{if } i(v) \neq i(v').
\end{cases}
\]

\[
B(v) = \begin{cases} 
B'(v') \cup \{0\} & \text{if } i(v) = i(v'), v \in S_{i(v)} \text{, and } \lambda(vv') \in A(v'), \\
B'(v') & \text{if } i(v) = i(v') \text{ and } (v \notin S_{i(v)} \text{ or } \lambda(vv') \notin A(v')), \\
\{0\} & \text{if } i(v) \neq i(v'), v \in S_{i(v)} \text{ and } 0 \in C'(v'), \\
\emptyset & \text{if } i(v) \neq i(v') \text{ and } (v \notin S_{i(v)} \text{ or } 0 \notin C'(v')).
\end{cases}
\]

### 7.2 Algorithm

In this section we present an efficient algorithm based on the characterization of the previous section. We keep using the same notation. In particular, \( T \) keeps being a rooted tree and each vertex has at most two children. We use \( n \) for the number of vertices of \( T \).

There are two main ideas used in our approach. The first one is that, for each vertex of the tree with two children, we want to spend time (roughly) proportional to the size of the smaller
Thus, we want to bound $\sigma$. For each vertex $v$ with two children, let $v_1$ and $v_2$ be its two children. If $v$ has only one child, we denote it by $v_1$.

For each node $v$, let $n(v)$ be the number of vertices in the subtree $T(v)$. (Thus $n(r) = n$.)

**Lemma 24.** If $V_2$ denotes the vertices of $T$ with two children, then

$$
\sum_{v \in V_2} \min\{n(v_1), n(v_2)\} = O(n \log n).
$$

**Proof.** For each vertex $u$ of $T$ define

$$
\sigma(u) = \sum_{v \in V \cap V(T(u))} \min\{n(v_1), n(v_2)\}.
$$

Thus, we want to bound $\sigma(r)$. We show by induction on $n(u)$ that

$$
\sigma(u) \leq n(u) \log_2 n(u).
$$

For the base case note that, when $n(u) = 1$, the vertex $u$ is a leaf and $\sigma(u) = 0$, so the statement holds.

If $u$ has one child $u_1$, then we have $V_2 \cap T(u) = V_2 \cap T(u_1)$,

$$
\sigma(u) = \sigma(u_1) \leq n(u_1) \log_2 n(u_1) \leq n(u) \log_2 n(u),
$$

and the bound holds. If $u$ has two children $u_1$ and $u_2$, then we can assume without loss of generality that $n(u_1) \leq n(u_2)$, which implies that $n(u_1) < n(u)/2$. Using the induction hypothesis for $n(u_1)$ and $n(u_2)$, we obtain

$$
\sigma(u) = \sum_{v \in V \cap V(T(u))} \min\{n(v_1), n(v_2)\}
= \sigma(u_1) + \sigma(u_2) + n(u_1)
\leq n(u_1) \log_2 n(u_1) + n(u_2) \log_2 n(u_2) + n(u_1)
< n(u_1) \log_2 (n(u)/2) + n(u_2) \log_2 n(u) + n(u_1)
= n(u_1) (\log_2 n(u) - 1) + n(u_2) \log_2 n(u) + n(u_1)
= (n(u_1) + n(u_2)) \log_2 n(u)
= n(u) \log_2 n(u).
$$

**Representation of $A(\nu)$ and $B(\nu)$.** We are going to represent $A(\nu)$ and $B(\nu)$ using balanced search trees. A suitable dynamic balanced binary search tree can store a set $X$ of $m$ real values and support the following operations:

- make a copy of the tree storing $X$ in $O(m)$ time;
- report the elements of $X$ in $O(m)$ time;
- insert a new element in $O(\log m)$;
- find the successor/predecessor in $X$ for a query value $y$ in $O(\log m)$ time;
- for a given real value $y$, split $X$ into the representation for $X_{\leq} = \{x \in X \mid x \leq y\}$ and the representation for $X_{>} = \{x \in X \mid x > y\}$ in $O(\log m)$ time;
• join the trees for $X_1$ and $X_2$, assuming that $\max(X_1) < \min(X_2)$, to obtain the tree for $X = X_1 \cup X_2$ in $O(\log m)$ time;

• add the same given value $\alpha$ to all the elements of $X$ in $O(1)$ time.

These properties are explained, for example, in the book by Brass [6, Chapter 3]; see Section 3.11 of the book for the more complex operations of split and join. They are also achieved (with amortized time bounds) using the classical splay trees [26]. For adding a value to all the elements we just need to keep an offset value in each node to be added to all elements below it. The offset of an element is obtained by adding the offsets of all its ancestors. A consequence of these properties is that in time $O(\log m)$ we can also split $X$ into the elements inside a given interval and the elements outside the interval so that we get a tree representation for both subsets.

The set $A(v)$ is stored as the union of intervals that may intersect, but with the property that no interval contains another interval. See Figure 17. Thus, sorting the intervals by their left endpoints or their right endpoints gives the same result. We use a dynamic binary search tree for the intervals using the left endpoints of the intervals as keys. For each interval, we store its length. We also allow to store in a node $\mu$ the information that all the intervals stored under this node have the same length, which is also stored at $\mu$. Since the tree is always accessed in a top-to-bottom manner, each time we access a node, we know the length of the corresponding interval, even if defined by an ancestor.

The set $B(v)$ is stored like a set of zero-length intervals also using a dynamic search tree, as it was done for $A(v)$. The reason for this artificial approach is that in our algorithm sometimes we will have to reset the lengths of all the intervals. Thus, there is no real difference between the data structure to store the sets $A(\cdot)$ and the one to store the sets $B(\cdot)$.

The size of this representation is the size of the binary search tree, that is, the number of (possibly non-disjoint) intervals that define $A(v)$ or $B(v)$. This is potentially larger than the minimum number of intervals that is needed because the intervals can intersect. For each vertex $v$ of $T$, we use $m_A(v)$ and $m_B(v)$ to denote the size of the representations of $A(v)$ and $B(v)$, respectively. Although the values $m_A(v)$ and $m_B(v)$ actually depend on the representation, this relaxation of the notation will not lead to confusion.

There are a couple of properties of the representation of $A(v)$ that we will use. With this representation, we can decide whether $y \in A(v)$ for a query $y$ in time $O(\log m_A(v))$. Indeed, we search for the largest left endpoint $a$ to the left of $y$, and then check whether the interval with $a$ as its left endpoint contains $y$. Similarly, we can report the rightmost or the leftmost interval in the representation that contains a query value $y$ in $O(\log m_A(y))$ time. For a given interval $I$, we can extract the representations of $I \cap A(v)$ and $(\mathbb{R}^2 \setminus I) \cap A(v)$ in $O(\log m_A(v))$ time. To obtain $I \cap A(v)$, we search for the rightmost interval in the representation of $A(v)$ that contains the left endpoint of $I$ and the leftmost interval that contains the right endpoint of $I$. We collect the intervals between those two extreme intervals, and clip those extreme intervals with $I$. (The other intervals are completely contained in $I$ and can be left untouched.) This computes $I \cap A(v)$, and a similar clipping has to be performed for $(\mathbb{R}^2 \setminus I) \cap A(v)$.

A minimal representation of $A(v)$ is the set of maximal intervals (with respect to inclusion).
that are contained in $A(v)$. From our representation of $A(v)$ we can compute the minimal representation of $A(v)$ in linear time, that is, $O(m_A(v))$ time. For this we just need to report the intervals of the structure sorted by their left endpoints, and merge adjacent intervals that intersect.

It is clear that $B(v)$ has at most $n(v)$ values because each value corresponds to a vertex of $T(v)$.

**Lemma 25.** Consider a vertex $v$ of $T$ with two children $v_1$ and $v_2$, and assume that we have representations for $A(v_1)$, $B(v_1)$, $A(v_2)$ and $B(v_2)$. Set $m_1 = m_A(v_1) + m_B(v_1)$ and $m_2 = m_A(v_2) + m_B(v_2)$, and assume that $m_1 \leq m_2$. We can compute in $O(m_1 \log m_2)$ time the representation of $A(v)$ and $B(v)$.

Moreover, the obtained representation of $A(v)$ has at most

$$\max\{m_A(v_1) + m_A(v_2), m_A(v_1) + m_B(v_2), m_B(v_1) + m_A(v_2), m_B(v_1) + m_B(v_2)\}$$

intervals.

**Proof.** First we compute $\chi(v)$. This can be done in $O(\log m_2)$ time by making queries to the representation of $A(v_1)$, $A(v_2)$, $B(v_1)$, $B(v_2)$. For this we note that $0 \in C(v_j)$ (where $j \in \{1, 2\}$) if and only if $[-\lambda(vv), +\lambda(vv)]$ contains some element of $B(v_j)$.

Next, from the representation of $A(v_2)$ we compute $A'(v_2)$ by adding the offset $-\lambda(vv_2)$ and intersecting with $\mathbb{R}_{\geq 0}$. This takes $O(\log m_2)$ time. Similarly, we can compute $B'(v_2)$. Note that we cannot afford to make copies of the representations of $A(v_2)$ or $B(v_2)$ because this would take $\Theta(m_2)$ time, which may be too much. On the other hand, we can make copies of the sets $A(v_1)$ and $B(v_1)$ in $O(m_1)$ time. In particular, we can assume that we have representations of $A'(v_1)$ and $B'(v_1)$ as an explicit list of intervals and values, respectively. This also implies that we can assume that the intervals in the representation of of $A'(v_1)$ are the intervals of the minimal representation.

Consider the case when $i(v) = i(v_1) = i(v_2)$. We have two parts.

1. First we compute $B(v)$. Because of Lemma 23, we have to compute $(B'(v_1) \cap A'(v_2)) \cup (B'(v_2) \cap A'(v_1))$. For each element $y$ of $B'(v_1)$, we query the representation of $A'(v_2)$ to decide whether $y \in A'(v_2)$. Thus, we can compute $B'(v_1) \cap A'(v_2)$ in $O(m_1 \log m_2)$ time. For each interval $I$ in the minimal representation of $A'(v_1)$, we query $B'(v_2)$ to collect the values $I \cap B'(v_2)$ represented as a tree. Each such query takes $O(\log m_2)$ time, for a total of $O(m_1 \log m_2)$. We can merge the representations of $I \cap B'(v_2)$, over the $m_1$ maximal intervals $I$ of $A'(v_1)$, in $O(m_1 \log m_2)$ time. Inserting in this representation the values of $B'(v_1) \cap A'(v_2)$, we finally obtain $(B'(v_1) \cap A'(v_2)) \cup (B'(v_2) \cap A'(v_1))$. If $\chi(v)$ is nonempty, we also insert 0 in the result, so that we obtain $B(v)$. Note that in this computation we have destroyed the representation of $B'(v_2)$.

2. Next we compute $A(v)$, which is $A'(v_1) \cap A'(v_2)$ because of Lemma 22. For each interval $I$ in the minimal representation of $A'(v_1)$, we extract from $A'(v_2)$ a representation of $I \cap A'(v_2)$. Then we compute $\bigcup I \cap A'(v_2) = A'(v_1) \cap A'(v_2) = A(v)$ by joining these representations. Since we are joining $O(m_1)$ trees, this takes $O(m_1 \log m_2)$. Note that in this computation we have destroyed the representation of $A'(v_2)$, so this step has to be made after the computation of $B(v)$, because $A'(v_2)$ is also used there (but not changed).

Consider now the case when $i(v) = i(v_1) \neq i(v_2)$. We proceed as follows.

1. First we compute $B(v)$, which is $B'(v_1) \cap C'(v_2)$ because of Lemma 23. Note that, for each $y \in \mathbb{R}$, we have $y \in C'(v_2)$ if and only if the interval $[y - \lambda(vv_2), y + \lambda(vv_2)]$ contains some element of $B(v_2)$. Therefore, for each element $y \in B'(v_1)$, we can query the representation of $B(v_2)$ with the interval $[y - \lambda(vv_2), y + \lambda(vv_2)]$ to detect whether the intersection is nonempty. This takes $O(m_1 \log m_2)$ time and does not change $B(v_2)$. Finally, we build the

---

2In the process we destroy the representations for $A(v_2)$ and $B(v_2)$. 

33
We use Lemma 25. If $i = i(v_2) \neq i(v_1)$. We proceed as follows.

1. The computation of $B(v)$ is trivial, since $B(v) = \chi(v)$ by Lemma 23.

2. The computation of $A(v) = C'(v_1) \cap C'(v_2)$ is similar to the case when $i(v) = i(v_1) \neq i(v_2)$. We compute explicitly the minimal representation of $C'(v_1)$, and use it as it was done there (for $A'(v_1)$). This takes $O(m_1 \log m_2)$ time.

In each case we spent $O(m_1 \log m_2)$ time, and the time bound follows. For the upper bound on the representation of $A(v)$, we note that each left endpoint of each interval gives rise to at most one interval in the representation of $A(v)$. The four terms correspond to the four possible cases we considered for the indices $i(v)$, $i(v_1)$, and $i(v_2)$.

**Lemma 26.** The problem Generalized Graphic Inverse Voronoi for an input $(T, U)$ where $T$ is an $n$-vertex tree of maximum degree 3 and the candidate Voronoi cells are pairwise disjoint, can be solved in $O(n \log^2 n)$ time.

**Proof.** We root $T$ at a leaf so that each node has at most two descendants. For each vertex $v$ of $T$, we compute a representation of the sets $A(v)$ and $B(v)$. The computation is bottom-up: we compute $A(v)$ and $B(v)$ when this has been computed for all the children of $v$. If $v$ has two children, we use Lemma 25. If $v$ has one child, then the computation can be done in $O(\log m_d(v) + \log m_b(v))$ time in a straightforward manner. When we arrive to the root $r$, we just have to check whether $B(r)$ is nonempty.
We can see by induction that, for each vertex \( v \) of \( T \), \( m_A(v) \leq n(v) \). (We already mentioned earlier that \( B(v) \) has at most \( n(v) \) values, one per vertex of \( T(v) \).) This is clear for the leaves because \( A(\cdot) \) has only one interval. For the internal nodes \( v \) that have only one child \( u \) it follows because the representation of \( A(v) \) is obtained from the representation of \( A(u) \) by a shift. For the internal nodes \( v \) with two children \( v_1 \) and \( v_2 \), the bound on \( m_A(v) \) follows by induction from the bound in Lemma 25. In particular, we have \( O(\log m_A(v) + \log m_B(v)) = O(\log n) \).

For each vertex with one child we spend \( O(\log n) \) time. For each vertex \( v \) with two children \( v_1 \) and \( v_2 \) we spend \( O(\min(n(v_1), n(v_2)) \log n) \) time. Thus, if \( V_1 \) and \( V_2 \) denote the vertices with one and two children, respectively, we spend

\[
O(n) + \sum_{v \in V_1} O(\log n) + \sum_{v \in V_2} O(\min(n(v_1), n(v_2)) \log n)
\]

\[
= O(n \log n) + O(\log n) \sum_{v \in V_2} O(\min(n(v_1), n(v_2)))
\]

time. Using Lemma 24, this time is \( O(n \log^2 n) \). Standard adaptations can be used to recover an actual solution.

\[ \Box \]

**Theorem 27.** The problem \textsc{Generalized Graphic Inverse Voronoi} for instances \( I = (T, ((U_1, S_1), \ldots, (U_k, S_k))) \), where \( T \) is a tree, can be solved in time \( O(N \log^2 N) \), where \( N = |V(T)| + \sum_i |U_i| + |S_i| \).

**Proof.** Because of Theorem 21, we can transform in \( O(N) \) time the instance \( I \) to another instance \( I' = (T', ((U'_1, S'_1), \ldots, (U'_k, S'_k))) \), where \( T' \) has maximum degree 3, the sets \( U'_1, \ldots, U'_k \) are pairwise disjoint, and \( T' \) has \( O(N) \) vertices. We can compute a solution to instance \( I' \) in \( O(N \log^2 N) \) time using Lemma 26. Then, we have to check whether this solution is actually a solution for \( I \). For this we use Lemma 3.

\[ \Box \]

**Corollary 28.** The problem \textsc{Graphic Inverse Voronoi} for instances \( I = (T, (U_1, \ldots, U_k)) \), where \( T \) is a tree, can be solved in time \( O(N \log^2 N) \), where \( N = |V(T)| + \sum_i |U_i| \).

### 8 Lower bound for trees

We can show the following lower bound on any algorithm based on algebraic operations on the lengths of the edges.

**Theorem 29.** In the algebraic computation tree model, solving \textsc{Graphic Inverse Voronoi} for trees with \( n \) vertices takes \( \Omega(n \log n) \) operations, even when the lengths are integers.

**Proof.** Consider an instance \( X, Y \) for the decision problem \textsc{Set Intersection}, where \( X = \{x_1, \ldots, x_n\} \) and \( Y = \{y_1, \ldots, y_n\} \) are sets of integers. We task is to decide whether \( X \cap Y \) is nonempty. This problem has a lower bound of \( \Omega(n \log n) \) in the algebraic computation tree model [27]. (In particular, this implies the same lower bound for the bounded-degree algebraic decision tree model.) Adding a common value to all the numbers, we may assume that \( X \) and \( Y \) contain only positive integers.

We construct an instance to the \textsc{Graphic Inverse Voronoi} problem with trees, as follows. See Figure 18. We construct a star \( S_X \) with \( n + 1 \) leaves. The edges of \( S_X \) have lengths \( x_1, \ldots, x_n, 2 \). We construct also a star \( S_Y \) with \( n + 1 \) leaves whose edges have lengths \( y_1 + 1, \ldots, y_n + 1, 1 \). Finally, we identify the leaf of \( S_X \) incident to the edge of length 2 and the leaf of \( S_Y \) incident to the edge of length 1. Let \( T \) be the resulting tree. We take the sets \( U_1 \) and \( U_2 \) to be the vertex sets of \( S_X \) and \( S_Y \), respectively. Note that \( T \) has \( 2n + 3 \) vertices. The reduction makes \( O(n) \) operations.

Since placing the sites on the center of the stars does not produce a solution, it is straightforward to see that the answers to \textsc{Set Intersection}(\( X, Y \)) and to \textsc{Graphic Inverse Voronoi}((\( T, (U_1, U_2) \))
are the same. Thus, solving **Graphic Inverse Voronoi**($T,(U_1,U_2)$) in $o(n \log n)$ time would provide a solution to **Set Intersection**($X,Y$) in $o(n \log n)$ time, and contradict the lower bound.

The lower bound also extends to the problem **Generalized Graphic Inverse Voronoi** with disjoint regions because we can apply the transformation to make the cells disjoint.

### 9 Conclusions

We have introduced the inverse Voronoi problem for graphs and we have shown several different hardness results, also within the framework of parameterized complexity. We have presented an algorithm for the case of trees that works in near-linear time, and also have shown a lower bound indicating that the problem for arbitrary trees cannot be solved in linear time (in a certain computation model).

Here we list some possible directions for further research:

- Is there an algorithm to solve the problem in $n^{O(w)}$ time for graphs with $n$ vertices and treewidth $w$ when the candidate Voronoi cells intersect? Perhaps one can also use some treewidth associated to the candidate Voronoi regions. In particular, for planar graphs a running time of $n^{O(\sqrt{k})}$ seems plausible but challenging when the Voronoi cells overlap.

- Considering cells defined by additively weighted sites.

- Following the analogy to problems considered in the Euclidean case [1, 4], find the smallest set $S$ such that each $U_i$ is the union of some Voronoi cells in $V(S)$. Taking $S = V(G)$ gives a feasible solution, and our hardness implies that the problem is NP-hard. Can one get approximation algorithms?

### Acknowledgments

Part of this work was done at the 21st Korean Workshop on Computational Geometry, held in Rogla, Slovenia, in June 2018. We thank all workshop participants for their helpful comments.

### References

[1] G. Aloupis, H. Pérez-Rosés, G. Pineda-Villavicencio, P. Taslakian, and D. Trinchet-Almaguer. Fitting voronoi diagrams to planar tesselations. *Combinatorial Algorithms - 24th International Workshop, IWOCA*, pp. 349–361. Springer, Lecture Notes in Computer Science 8288, 2013, [https://doi.org/10.1007/978-3-642-45278-9_30](https://doi.org/10.1007/978-3-642-45278-9_30).
[2] P. F. Ash and E. D. Bolker. Recognizing dirichlet tessellations. *Geometriae Dedicata* 19(2):175–206, Nov 1985, [http://dx.doi.org/10.1007/BF00181470](http://dx.doi.org/10.1007/BF00181470).

[3] S. Bandypadhyay, A. Banik, S. Das, and H. Sarkar. Voronoi game on graphs. *Theor. Comput. Sci.* 562:270–282, 2015, [https://doi.org/10.1016/j.tcs.2014.10.003](https://doi.org/10.1016/j.tcs.2014.10.003).

[4] S. Banerjee, B. B. Bhattacharya, S. Das, A. Karmakar, A. Maheshwari, and S. Roy. On the construction of generalized voronoi inverse of a rectangular tessellation. *Trans. Computational Science*, pp. 22–38. Springer, Lecture Notes in Computer Science 8110, 2013, [https://doi.org/10.1007/978-3-642-41905-8_3](https://doi.org/10.1007/978-3-642-41905-8_3).

[5] A. Biniaz, S. Cabello, P. Carmi, J. De Carufel, A. Maheshwari, S. Mehrabi, and M. Smid. On the minimum consistent subset problem. *CoRR* abs/1810.09232, 2018, [arXiv:1810.09232](http://arxiv.org/abs/1810.09232).

[6] P. Brass. *Advanced Data Structures*. Cambridge University Press, 2008, [https://doi.org/10.1017/CBO9780511800191](https://doi.org/10.1017/CBO9780511800191).

[7] S. Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. *ACM Trans. Algorithms*, to appear. Preliminary version presented at SODA 2017. Full version available at [http://arxiv.org/abs/1702.07815](http://arxiv.org/abs/1702.07815).

[8] S. Chaudhuri and C. D. Zaroliagis. Shortest paths in digraphs of small treewidth. part I: sequential algorithms. *Algorithmica* 27(3):212–226, 2000, [https://doi.org/10.1007/s004530010016](https://doi.org/10.1007/s004530010016).

[9] É. Colin de Verdière. Shortest cut graph of a surface with prescribed vertex set. *Algorithms - ESA 2010, 18th Annual European Symposium, Part II*, pp. 100–111. Springer, Lecture Notes in Computer Science 6347, 2010, [http://dx.doi.org/10.1007/978-3-642-15781-3_9](http://dx.doi.org/10.1007/978-3-642-15781-3_9).

[10] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015, [http://dx.doi.org/10.1007/978-3-319-21275-3](http://dx.doi.org/10.1007/978-3-319-21275-3).

[11] T. K. Dey, F. Fan, and Y. Wang. Graph induced complex on point data. *Comput. Geom.* 48(8):575–588, 2015, [https://doi.org/10.1016/j.comgeo.2015.04.003](https://doi.org/10.1016/j.comgeo.2015.04.003).

[12] M. Erwig. The graph Voronoi diagram with applications. *Networks* 36(3):156–163, 2000, [http://dx.doi.org/10.1002/1097-0037(200010)36:3<156::AID-NET2>3.0.CO;2-L](http://dx.doi.org/10.1002/1097-0037(200010)36:3<156::AID-NET2>3.0.CO;2-L).

[13] P. Gawrychowski, H. Kaplan, S. Mozes, M. Sharir, and O. Weimann. Voronoi diagrams on planar graphs, and computing the diameter in deterministic $\tilde{o}(n^{5/3})$ time. *Proc. 29th ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pp. 495–514. SIAM, 2018, [http://dx.doi.org/10.1137/1.9781611975031.33](http://dx.doi.org/10.1137/1.9781611975031.33). Full version available at [http://arxiv.org/abs/1704.02793](http://arxiv.org/abs/1704.02793).

[14] P. Gawrychowski, S. Mozes, O. Weimann, and C. Wulff-Nilsen. Better tradeoffs for exact distance oracles in planar graphs. *Proc. 29th ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pp. 515–529, 2018, [http://dx.doi.org/10.1137/1.9781611975031.34](http://dx.doi.org/10.1137/1.9781611975031.34).

[15] D. Gerdner, V. Mészáros, D. Pálvölgyi, A. Pokrovskiy, and G. Rote. Advantage in the discrete voronoi game. *J. Graph Algorithms Appl.* 18(3):439–457, 2014, [https://doi.org/10.7155/jgaa.00331](https://doi.org/10.7155/jgaa.00331).
L. Gottlieb, A. Kontorovich, and P. Nisnevitch. Near-optimal sample compression for nearest neighbors. *IEEE Trans. Information Theory* 64(6):4120–4128, 2018, doi:10.1109/TIT.2018.2822267, https://doi.org/10.1109/TIT.2018.2822267.

P. E. Hart. The condensed nearest neighbor rule (corresp.). *IEEE Trans. Information Theory* 14(3):515–516, 1968, https://doi.org/10.1109/TIT.1968.1054155.

R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.* 63(4):512–530, 2001, https://doi.org/10.1006/jcss.2001.1774.

S. Kannan, C. Mathieu, and H. Zhou. Graph reconstruction and verification. *ACM Trans. Algorithms* 14(4):40:1–40:30, 2018, http://doi.acm.org/10.1145/3199606. Preliminary version in ICALP 2015.

D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.* 11(2):329–343, 1982, https://doi.org/10.1137/0211025.

D. Marx. Can you beat treewidth? *Theory of Computing* 6(1):85–112, 2010, https://doi.org/10.4086/toc.2010.v006a005.

D. Marx and M. Pilipczuk. Optimal parameterized algorithms for planar facility location problems using voronoi diagrams. *CoRR* abs/1504.05476, 2015, http://arxiv.org/abs/1504.05476. Preliminary version at ESA 2015.

W. Mulzer and G. Rote. Minimum-weight triangulation is NP-hard. *J. ACM* 55(2):11:1–11:29, 2008, http://doi.acm.org/10.1145/1346330.1346336.

A. Okabe and K. Sugihara. *Spatial Analysis along Networks. Statistical and Computational Methods*. John Wiley and Sons, 2012.

G. L. Ritter, H. B. Woodruff, S. R. Lowry, and T. L. Isenhour. An algorithm for a selective nearest neighbor decision rule (corresp.). *IEEE Trans. Information Theory* 21(6):665–669, 1975, https://doi.org/10.1109/TIT.1975.1055464.

D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *J. ACM* 32(3):652–686, 1985, http://doi.acm.org/10.1145/3828.3835.

A. C. Yao. Lower bounds for algebraic computation trees with integer inputs. *SIAM J. Comput.* 20(4):655–668, 1991, https://doi.org/10.1137/0220041.