



Reconstructing graphs with subgraph compositions^{☆,☆☆}

Antoine Dailly^a, Tuomo Lehtilä^{b,*}

^a Université Clermont Auvergne, INRAE, UR TSCF, 63000, Clermont-Ferrand, France

^b Department of Mathematics and Statistics, University of Turku, FI-20014, Turku, Finland



ARTICLE INFO

Article history:

Received 31 March 2025

Received in revised form 28 November 2025

Accepted 16 April 2026

Keywords:

Graph reconstruction

Mass spectrometry

Polymer-based data storage

Trees

ABSTRACT

We generalize the problem of reconstructing strings from their substring compositions first introduced by Acharya et al. in 2015 motivated by polymer-based advanced data storage systems utilizing mass spectrometry. Namely, we see strings as labeled path graphs, and as such try to reconstruct labeled graphs. For a given integer t , the subgraph compositions contain either vectors of labels for each connected subgraph of order t (t -multiset-compositions) or the sum of all labels of all connected subgraphs of order t (t -sum-composition). We ask whether, given a graph with a known structure and an oracle that can be queried for compositions, one can reconstruct the labeling of the graph. If it is possible, then the graph is reconstructible; otherwise, it is confusable, and two labeled graphs with the same compositions are called equicomposable. We prove that reconstructing through a brute-force algorithm is wildly inefficient, before giving methods for reconstructing several graph classes using as few compositions as possible. We also give negative results, finding the smallest confusable graphs and trees, as well as families with a large number of equicomposable non-isomorphic graphs. An interesting result occurs when twinning one leaf of a path: some paths are confusable, creating a twin out of a leaf sees the graph alternating between reconstructible and confusable depending on the parity of the path, and creating a false twin out of a leaf makes the graph reconstructible using only sum-compositions in all cases.

© 2026 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC license (<http://creativecommons.org/licenses/by-nc/4.0/>).

1. Introduction

The problem of reconstructing a graph from incomplete information has been studied for quite some time, being introduced at least in the 1950s [15]. Much of the research on this topic is about finding the original graph by asking oracle questions about, e.g., connectivity of sets of vertices. In particular, we usually know the vertex set and wish to find the edge set. We are interested in a vertex-labeled version of the reconstruction problem: given a graph (of which we know the structure) and an oracle, can we find the labels of the vertices? Note that we are in a wholly deterministic setting, so the oracle answers truthfully and correctly, and our goal is to find the correct labels exactly (up to graph isomorphisms). Furthermore, the problem is trivial if only one label can be used, so we are going to use alphabets of size at least 2.

There are several ways to define an oracle to query about the graph, and we are going to use the one used by Acharya et al. in the context of strings [1] and reused by Bartha et al. for edge-labeled graphs [5,6]. The oracle can be queried about *subgraph compositions*: for a given integer t , it gives information about all connected subgraphs of order t . This information is not separated, so we do not know which piece of information relates to any given subgraph.

[☆] The research of the second author was partially funded by the Research Council of Finland grants 338797 and 358718.

^{☆☆} A shorter conference version of this paper has been presented in IEEE Symposium on Information Theory 2025 (Dailly and Lehtilä, 2025 [8]).

* Corresponding author.

E-mail address: tualeh@utu.fi (T. Lehtilä).

Given a vertex-labeled graph \mathcal{G} where each label belongs to k -letter alphabet Σ_k for $k \geq 2$, we give the total number each label occurs with an integer valued vector of length k . For example, if the graph has two vertices labeled with A and one with B over alphabet $\Sigma_2 = (A, B)$, then the vector would be $(2, 1)$ where 2 corresponds to the number of A 's and 1 corresponds to the number of B 's. We are going to use two types of compositions. First is the *multiset-composition* $\mathcal{M}_t(\mathcal{G})$, which is a multiset of vectors such that each vector corresponds to a connected subgraph of order t , and contains all of its labels (note that we do not know which vector is connected to which subgraph, so we do not know the labels of vertices but only how many labels there are in different types of subgraphs). Next, the *sum-composition* $S_t(\mathcal{G})$, which is a vector of length k , where each element corresponds to a symbol of the alphabet Σ_k , and contains the number of times the symbol appears in all subgraphs of order t . It is clear that we can find a sum-composition from a multiset-composition, but the reverse is not true, hence sum-compositions contain less information, and being able to reconstruct a graph using only sum-compositions is generally stronger.

If it is possible to connect all labels of a graph G with its vertices by querying compositions, then we say that G is *reconstructible*. There is a straightforward and wildly inefficient algorithm to find the labels of a reconstructible graph: enumerate all its multiset-compositions, and do so for every possible non-isomorphic labeling until the multiset-combinations coincide. Obviously, our goal is to have more efficient algorithms for reconstructing graphs, which we will generally evaluate by number of queries (note that, in a classical complexity framework, we consider a query as a singular operation, having access to the oracle). Since sum-compositions contain less information than multiset-compositions, we also consider algorithms reconstructing a graph using only sum-compositions to be better than those also using multiset-compositions. If G can be reconstructed with only sum-compositions, then we say that G is *sum-reconstructible*.

Some graphs cannot be reconstructed, even with the brute-force method, those are called *confusable*. Two labeled graphs that yield the exact same multiset-compositions are called *equicomposable*, and a simple way to prove that a graph G is confusable is to exhibit two equicomposable labeled copies of G .

Formal definitions will be given in Section 1.2. Before this, we explain in the following subsection how the problem of reconstructing graphs can be related to recovering data from polymers.

1.1. Polymer-based memory systems

Recently, polymer-based data storage systems have been considered from both theoretical and experimental points-of-views, see for example [1,2,17]. In these types of memory systems, the information is stored into the (synthesized) polymer using molecules with large weight differences. Then, for example, when we use molecules with weights A and B , we can deduce the number of A 's and B 's based on the total weight of the polymer but this does not allow us to immediately know their ordering. For example, when we know that there are at most nine molecules in total and we have $A = 1$ and $B = 10$, then total mass 72 implies that there are two A s and seven B s.

For reading the mass of the polymer, tandem mass (MS/MS) spectrometers are used [1,11,17]. The tandem mass spectrometers break the polymers into smaller fragments and measure the masses of these fragments. The goal is to reconstruct the original polymer using the masses of these fragments which we obtain from multiple original copies of the polymer.

We consider this problem from a somewhat idealized combinatorial perspective for mathematical modeling, as has been previously done for example in [1,10,11,20,21]. The two assumptions originally presented in [1] which we use are:

- A1: We can uniquely deduce the composition, that is, the number of each symbol of a fragment string from its mass.
- A2: We observe every fragment of a polymer with equal frequency.

In previous work, it has been assumed that the underlying polymer structure is a string (although in [4] generalizing it to other structures, such as cycles, has been suggested).

1.2. Problem definitions and first lemmas

In this section, we give formal definitions, explain our notation and present some general lemmas.

Definition 1. For an integer $k \geq 2$, let $\mathcal{G} = (G, \lambda)$ be a connected labeled graph with labeling $\lambda : V(G) \rightarrow \Sigma_k$, where Σ_k contains symbols (A_1, \dots, A_k) . We denote $n_i(\mathcal{G}) = |\{v \in V(G) \mid \lambda(v) = A_i\}|$ and $c(\mathcal{G}) = \{A_1^{n_1(\mathcal{G})}, A_2^{n_2(\mathcal{G})}, \dots, A_k^{n_k(\mathcal{G})}\}$.

We denote the *subgraph composition multiset* of \mathcal{G} by:

$$\mathcal{M}(\mathcal{G}) = \{\{c(\mathcal{G}') \mid \mathcal{G}' \text{ is connected induced subgraph of } \mathcal{G}\}\}$$

and call the elements of $\mathcal{M}(\mathcal{G})$ *compositions*.

Let \mathcal{G}_t be the set of t -vertex connected induced subgraphs of \mathcal{G} . We denote by $\mathcal{M}_t(\mathcal{G})$ the restriction of $\mathcal{M}(\mathcal{G})$ to connected t -vertex subgraphs of \mathcal{G} , that is, $\mathcal{M}_t(\mathcal{G}) = \{\{c(\mathcal{G}') \mid \mathcal{G}' \in \mathcal{G}_t\}\}$. We define the *t -sum-composition* of \mathcal{G} as $S_t(\mathcal{G}) = \sum_{\mathcal{G}' \in \mathcal{G}_t} (n_1(\mathcal{G}'), n_2(\mathcal{G}'), \dots, n_k(\mathcal{G}'))$ and the *sum-composition* of \mathcal{G} by $S(\mathcal{G}) = \bigcup_{t=1}^{|V(\mathcal{G})|} S_t(\mathcal{G})$.

Notation. For readability, when $c(\mathcal{G}) = \{A_1^{n_1(\mathcal{G})}, A_2^{n_2(\mathcal{G})}, \dots, A_k^{n_k(\mathcal{G})}\}$, we may denote it as a sequence, that is, $c(\mathcal{G}) = A_1^{n_1(\mathcal{G})} A_2^{n_2(\mathcal{G})} \dots A_k^{n_k(\mathcal{G})}$.

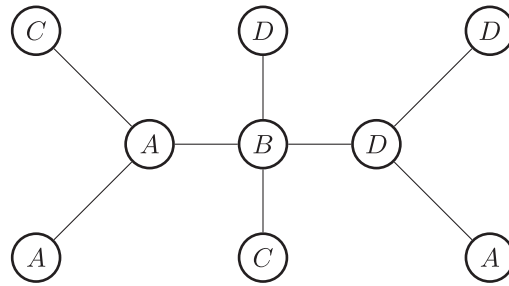


Fig. 1. A labeled tree \mathcal{T} on nine vertices.

Notation. For ease of use, we will sometimes denote the t -sum-composition of \mathcal{G} as:

$$S_t(\mathcal{G}) = \sum_{i=1}^k n_i^t(\mathcal{G})A_i$$

where $n_i^t(\mathcal{G}) = \sum_{\mathcal{G}' \in \mathcal{G}_t} n_i(\mathcal{G}')$. That is, $n_i^t(\mathcal{G})$ is the number of times the label A_i appears in total in every connected induced subgraph of order t .

Note that $c(\mathcal{G})$ can be understood as a *Parikh vector* of \mathcal{G} , [5]. Hence we mean with, for example, alphabet $\Sigma_3 = (A, B, C)$ that composition A is the same as vector $(1, 0, 0)$ and $(2, 1, 0) - AB = (1, 0, 0) = A$ (the subtraction $X - Y$ is only allowed if it does not result in any negative coordinates). We consider t -subgraph composition multisets and t -sum compositions as *queries* asked to an oracle. We call the former a *multiset-query* and the latter a *sum-query*.

As an example, consider the labeled tree \mathcal{T} from Fig. 1. The multiset-query $\mathcal{M}_3(\mathcal{T})$ outputs $\{\{A^2C, ABC, ABC, A^2B, ABD, ABD, ABD, BD^2, BD^2, BCD, BCD, AD^2\}\} = \{\{A^2C, 2ABC, A^2B, 3ABD, 2BD^2, 2BCD, AD^2\}\}$ where we write A^2 for AA and $2ABC$ for two copies of ABC . Note that in our notation ABC and ACB are the same as we are only interested in the compositions. The sum-query $S_3(\mathcal{T})$ outputs $(10, 10, 5, 11)$, which can also be written as $S_3(\mathcal{T}) = 10A + 10B + 5C + 11D$.

In the following definition, we define what we mean by *reconstructible*. Informally, the question is: can we (or how can we) obtain the original graph labeling of a graph (up to graph isomorphisms), when we know the structure of the underlying graph and the multiset $\mathcal{M}(\mathcal{G})$. Furthermore, we also define a stronger notion of *sum-reconstructibility* which asks whether the labeling of \mathcal{G} can be deduced based on $S(\mathcal{G})$ instead of $\mathcal{M}(\mathcal{G})$.

Definition 2. Let G be a graph and λ be a labeling of G . We say that $\mathcal{G} = (G, \lambda)$ is *reconstructible* if any labeled graph $\mathcal{G}' = (G, \lambda')$ such that \mathcal{G} and \mathcal{G}' have equal composition multisets (i.e., $\mathcal{M}(\mathcal{G}) = \mathcal{M}(\mathcal{G}')$) is isomorphic to \mathcal{G} . Similarly, we say that \mathcal{G} is *sum-reconstructible* if any labeled graph $\mathcal{G}' = (G, \lambda')$ with equal sum-composition (i.e., $S(\mathcal{G}) = S(\mathcal{G}')$) is isomorphic to \mathcal{G} .

Furthermore, we say that a graph G is (sum-)reconstructible if all of its labeled graphs are (sum-)reconstructible, and that a class of (resp. labeled) graphs is (sum-)reconstructible if all of its (resp. labeled) graphs are (sum-)reconstructible.

If G or \mathcal{G} is not reconstructible, then we say that it is *confusable*. Furthermore, we say that \mathcal{G}_1 and \mathcal{G}_2 are *equicomposable* if $\mathcal{M}(\mathcal{G}_1) = \mathcal{M}(\mathcal{G}_2)$, denoted by $\mathcal{G}_1 \sim \mathcal{G}_2$.

As a first, easy result, consider the complete graph K_n on n vertices, where every possible edge exists. Since all vertices are symmetrically positioned, two labelings with the same subgraph composition multiset are isomorphic, and thus it suffices to query $\mathcal{M}_n(K_n)$ to reconstruct K_n .

We also show the following natural lemmas:

Lemma 3. *If G is sum-reconstructible, then G is reconstructible.*

Proof. Let G be sum-reconstructible and suppose to the contrary that $\mathcal{G} = (G, \lambda)$ and $\mathcal{G}' = (G, \lambda')$ are equicomposable where λ and λ' are labelings over Σ_k . Hence, we have $\mathcal{M}(\mathcal{G}) = \mathcal{M}(\mathcal{G}')$, where $\mathcal{M}(\mathcal{G}), \mathcal{M}(\mathcal{G}') \in \mathbb{Z}^k$, (recall that $\mathcal{M}(\mathcal{G})$ and $\mathcal{M}(\mathcal{G}')$ are computed over connected subgraphs) and in particular, the subset $\mathcal{M}_t(\mathcal{G})$ of $\mathcal{M}(\mathcal{G})$ containing exactly the t symbol compositions of $\mathcal{M}(\mathcal{G})$ is equal to the subset $\mathcal{M}_t(\mathcal{G}')$ of $\mathcal{M}(\mathcal{G}')$. Hence, we have $S_t(\mathcal{G}) = S_t(\mathcal{G}')$. Furthermore, this is true for any $t \geq 1$. Hence, we have $S(\mathcal{G}) = S(\mathcal{G}')$, a contradiction. Thus, the graph G is reconstructible. \square

Lemma 4. *For $\mathcal{G} = (G, \lambda)$, we have $S_2(\mathcal{G}) = \sum_{v \in V(G)} d(v)\lambda(v)$.*

Proof. In $S_2(\mathcal{G})$, every edge is counted. Hence, the label of each vertex is counted exactly in as many edges as the vertex appears, which is its degree. \square

1.3. Graph notations

The graphs we consider are all simple, connected, and unweighted. We are going to study specific graph families, focusing mainly on *trees*, that is, acyclic graphs. In a tree, degree 1 vertices are called *leaves* while the other vertices are called *internal vertices*. The simplest subfamily of trees is *paths*: the path P_n on n vertices has two leaves (or one if $n = 1$). Another simple family is *stars*: the star S_n on $n + 1$ vertices has n leaves connected to the *center*. The *subdivided star* S_{n_1, \dots, n_m} is the star S_m where each edge has been subdivided $n_1 - 1, \dots, n_m - 1$ times, that is, it consists of a center to which are attached paths of n_1, \dots, n_m vertices, called *branches*. In particular, a star is *subdivided at most once* if $n_1, \dots, n_m \leq 2$ (that is, every edge is subdivided at most once). Finally, a *bistar* $B_{m,n}$ has two adjacent vertices each being adjacent with m and n leaves, respectively; those two non-leaf vertices are called the centers of the bistar.

The *open neighborhood* of a vertex is the set of all the vertices it is adjacent to, and its *closed neighborhood* is its open neighborhood plus itself; we call *twins* (resp. *false twins*) two vertices with the same closed (resp. open) neighborhood (for instance, in a star, all leaves are false twins, and in the complete graph all vertices are twins). Note that, if a set of vertices are all twins or false twins, then they are not distinguishable in the setting we are studying, hence, we consider isomorphic labelings to be the same. We denote by \sqcup the additive union of (multi)sets, that is, $A \sqcup A = \{\{A, A\}\}$.

1.4. Related research

The related research consists of two topics: graph theoretical reconstruction problems and mass spectrometry related string reconstruction problems as well as DNA-based data storages. We first discuss graph theoretic reconstruction problems and then continue with mass spectrometry and DNA related research.

Probably the best known and still open graph reconstruction conjecture from Kelly and Ulam states that any (unlabeled) graph G on at least $n \geq 3$ vertices can be reconstructed (up to isomorphisms) based on the multiset containing every induced $(n - 1)$ -vertex subgraph of G [15,26]. While this conjecture is still open, it has been proved for some graph classes such as trees [15]. Interested readers may read more about the reconstruction conjecture in a survey article by Harary [12]. For some variants of graph reconstruction, we refer interested readers to the survey [3] or to [7,16,24].

Bartha et al. have considered an edge version of our reconstruction problem in [5,6]. In their variant, the edges are labeled (instead of vertices) and we obtain a composition multiset containing the information about the edge labels. However, their results on the edge-variant do not seem to give us much information about the vertex-labeled case, although they do note some minor results for vertex-labeled case in their discussions. In particular, they state (without a proof or a construction) that the smallest confusable vertex-labeled tree has seven vertices. We obtained the mentioned graph through personal communications with Burcsi and Lipták (see Fig. 6) and show its unicity (see Theorem 22).

From the application side, the research related to advanced polymer-based memory storages is interesting. While especially DNA-based memory storage systems are widely studied, our research is related more strongly to polymer-based data storage systems using mass spectrometry. This has also been the motivation of Acharya et al. (and the related work it has generated) in [1] who initiated the study of string reconstruction using substring compositions, which we generalize to graphs: while they construct strings, it is possible to interpret their work as research on graph reconstruction focusing on labeled paths. Real-world mass spectrometry experiments on polymer-based data storage have been considered, for example in [2,17].

In particular, Acharya et al. have shown that the path P_n is reconstructible (from $\mathcal{M}(\mathcal{P}_n)$) if and only if $n + 1 \leq 8$ or $n + 1 \in \{p, 2p\}$ where p is a prime. The case of paths is natural from the point of view of memories, as information is often considered as strings of symbols. Banerjee et al. [4] have suggested generalizing the problem of string reconstruction to more complicated structures. In particular, they suggested cycles. The research of Acharya et al. has sprung up multiple studies and variations, see for example [1,10,11,20,21,28,29].

Another related widely studied polymer-based advanced information storage system is DNA-based data storage system where the information is directly stored into DNA-strands. These systems have also been studied from a similar combinatorial reconstruction perspective, where a strand is reconstructed from multiple erroneous copies, see for example [13,14,18,19,22,27]. It has been pointed out in [2] that the advantage of polymer-based systems over the DNA-based is their lower cost.

Some variants of the reconstruction problem include reconstructing strings/paths from composition multisets containing one of the two leaves of the path, that is, reconstructions of strings from prefixes and suffixes [11,28,29] (however, this variant seems most natural in the special case of paths). Another variant suggested by Bartha et al. in [5,6] for the edge-labeled reconstruction is to consider instead of any induced subgraph, only subgraphs with specific structure. For example, only subgraphs which are paths. Finally, one more variant, considered by Bartha et al. in [5,6] for the edge-labeled case, is to omit the structure of the underlying graph (or to restrict it to some graph class such as trees) and to also reconstruct it.

1.5. Structure of the paper

We begin in Section 2 by showing that the brute-force reconstruction can be extremely inefficient, since the graphs we are studying have a large number of non-isomorphic labelings. This also relates to the application to information storage. Then, in Section 3, we show how to reconstruct some classes of trees. An interesting case arises when creating a twin from a leaf of a path: this simple operation has a significant impact on reconstructibility. The choice of creating a twin or a false twin also has an impact, as is shown in Section 4, where we also enumerate the smallest confusable graphs and trees. Finally, in Section 5, we construct large families of equicomposable graphs that are more complex than paths, with the help of constructions from [1].

2. Number of labelings

Reconstructing a non-confusable graph G can be done by simply enumerating all its compositions and comparing those to the possible labelings of G over an alphabet of size k . However, doing so requires the enumeration of those labelings. In this section, we will see that the number of such labelings is extremely high (even with the condition that they should be non-isomorphic), and as such this method is generally inefficient, justifying the specific analysis we do in later sections. Furthermore, if we consider a labeling of a given graph as an encoding of information, then we are interested in graphs which have as many labelings as possible.

Definition 5. For a given graph G , we denote by $\chi_k(G)$ the number of non-isomorphic labelings of G over an alphabet of size k .

We have included in this section proofs for the value $\chi_k(G)$ over the graph classes studied in this article. Although some of these values may be known in the literature (such as $\chi_k(P_n)$), we have decided to include the proofs as they are short and of independent interest, as well as for the sake of completeness. We begin by giving the value of $\chi_k(G)$ for paths and then continue with complete graphs.

Theorem 6. Given a path P_n on n vertices, we have

$$\chi_k(P_n) = \frac{k^n + k^{\lceil n/2 \rceil}}{2}.$$

Proof. Given a path P_n , the total number of labelings is k^n . However, each labeling is isomorphic with its reverse. Note that, while they are isomorphic, a reverse is different from the original labeling unless the original labeling is a palindrome. Furthermore, the $\lceil n/2 \rceil$ first labels of a path fix the latter $\lfloor n/2 \rfloor$ labels of a palindrome. Hence, we have $\chi_k(P_n) = \frac{k^n - k^{\lceil n/2 \rceil}}{2} + k^{\lceil n/2 \rceil} = \frac{k^n + k^{\lceil n/2 \rceil}}{2}$. \square

Theorem 7. Given a complete graph K_n on n vertices, we have $\chi_k(K_n) = \binom{n+k-1}{n}$.

Proof. As no vertex is distinguishable from others, the problem can be translated as: in how many ways can we place n balls into k bins? Hence, by [25], the value $\chi_k(K_n)$ can be solved using the “stars and bars-technique”, which gives $\chi_k(K_n) = \binom{n+k-1}{n}$. \square

We now determine the value of $\chi_k(G)$ for the graph classes that we will study in Section 3.

Theorem 8. Given a subdivided star S with maximum degree $\Delta \geq 3$, maximum branch length m , and n_i branches of length i for $i \in \{1, \dots, m\}$, we have $\chi_k(S) = k \prod_{i=1}^m \binom{k^i + n_i - 1}{n_i}$.

Proof. As we can identify the ends of a branch from each other (only one of them is a leaf), there are k^i non-isomorphic labelings inside a single branch of length i . However, the n_i branches of length i are in symmetric positions and hence we can consider labeling them as labeling vertices of the complete graph K_{n_i} using an alphabet containing k^i symbols. Since we also have k options for the center, by Theorem 7, we have

$$\chi_k(S) = k \prod_{i=1}^m \chi_{k^i}(K_{n_i}) = k \prod_{i=1}^m \binom{k^i + n_i - 1}{n_i}. \quad \square$$

Corollary 9. Given a star S_n on $n + 1$ vertices, we have $\chi_k(S_n) = k \binom{n+k-1}{n}$.

Proof. Let S_n be a star on $n + 1$ vertices. Using the terminology of Theorem 8, we have $m = 1$ and $n_1 = n$. Hence, we have $\chi_k(S_n) = k \prod_{i=1}^1 \binom{k^i + n_i - 1}{n_i} = k \binom{n+k-1}{n}$. \square

Corollary 10. Given a subdivided star $S_{1,1,m}$ on $m + 3$ vertices, we have $\chi_k(S_{1,1,m}) = \frac{k^{m+3} + k^{m+2}}{2}$.

Proof. By Theorem 8, we have $\chi_k(S_{1,1,m}) = k \binom{k+1}{2} \binom{k^m}{1} = \frac{k^{m+3} + k^{m+2}}{2}$. \square

Theorem 11. Given a bistar $B_{n,m}$, we have

$$\chi_k(B_{n,m}) = \begin{cases} k^2 \binom{n+k-1}{n} \binom{m+k-1}{m}, & \text{if } n \neq m; \\ \frac{k^2 \binom{n+k-1}{n}^2 + k \binom{n+k-1}{n}}, & \text{if } n = m. \end{cases}$$

Proof. Let us first consider the case with $n \neq m$. Note that in this case the two stars around the centers are not in symmetrical positions. Hence, we may choose both centers in k ways, each leaf around one of the centers in $\binom{n+k-1}{n}$ ways and the leaves around the other center in $\binom{m+k-1}{m}$ ways. Hence, we have $\chi_k(B_{n,m}) = k^2 \binom{n+k-1}{n} \binom{m+k-1}{m}$.

Let us next consider the case with $n = m$. In this case, the two stars around the centers are in symmetrical positions. By Corollary 9, there are $k \binom{n+k-1}{n}$ ways to label one half of the bistar. As there are two symmetric halves in the bistar, we may consider labeling a path P_2 using an alphabet of size $k \binom{n+k-1}{n}$. Hence, it follows from Theorem 6 that $\chi_k(B_{n,m}) = \chi_{k \binom{n+k-1}{n}}(P_2) = \frac{k^2 \binom{n+k-1}{n}^2 + k \binom{n+k-1}{n}}{2}$. \square

Note that for n -vertex graphs, Corollary 10 gives $\chi_k(S_{1,1,n-3}) = \frac{k^n + k^{n-1}}{2}$ while Theorem 6 yields $\chi_k(P_n) = \frac{k^n + k^{\lceil n/2 \rceil}}{2}$. This implies that we may store more information in $S_{1,1,n-3}$ than in P_n . When $k = 2$ and n is large, the ratio $\chi_k(S_{1,1,n-3}) : \chi_k(P_n)$ approaches 1.5.

3. Reconstructibility of some trees

In this section, we will show how to reconstruct several subclasses of trees. We begin with useful lemmas, then each subsection will be dedicated to a specific subclass.

Lemma 12. One can find the composition of leaves and internal vertices of a tree \mathcal{T} using two sum-queries.

Proof. Let $\mathcal{T} = (T, \lambda)$ be such that T is a tree of order n with ℓ leaves. The query $S_n(\mathcal{T})$ gives us the list of all labels. The query $S_{n-1}(\mathcal{T})$ gives us the sum of labels for all the subtrees of order $n - 1$, each of which corresponds to T minus exactly one leaf. Hence, ℓ subtrees are being summed in $S_{n-1}(\mathcal{T})$; each leaf is counted $\ell - 1$ times and each internal vertex is counted ℓ times. Thus, the composition I of internal vertices is given by $S_{n-1}(\mathcal{T}) - (\ell - 1)S_n(\mathcal{T})$, and the composition L of leaves is given by $S_n(\mathcal{T}) - I$. \square

Example 13. Consider the labeled tree \mathcal{T} on nine vertices presented in Fig. 1. The queries $S_9(\mathcal{T})$ and $S_8(\mathcal{T})$ output $(3, 1, 2, 3)$ and $(16, 6, 10, 16)$ for alphabet $\Sigma = (A, B, C, D)$, respectively. As there are six leaves, the composition of the internal vertices is given by $I = (16, 6, 10, 16) - (6 - 1) \times (3, 1, 2, 3) = (1, 1, 0, 1)$; the internal vertices have labels A, B and D . From this, we can extract the composition of leaves: $S_9(\mathcal{T}) - I = (3, 1, 2, 3) - (1, 1, 0, 1) = (2, 0, 2, 2)$, so two leaves have label A , two leaves have label C , and two leaves have label D .

Corollary 14. Stars are sum-reconstructible using two sum-queries.

Proof. Let S_n be a star with a center and n leaves. We compute the composition of leaves and of the internal vertex using Lemma 12, so the label of the internal vertex is known. Since the leaves are symmetrically positioned, all labeled stars with the same composition of leaves and identically labeled center vertex are isomorphic, and thus we can reconstruct the star. \square

Lemma 15. The label of the center of a subdivided star with maximum degree $\Delta \geq 3$ can be found using three sum-queries.

Proof. Let $S = (S, \lambda)$ be such that S is a subdivided star with maximum degree $\Delta \geq 3$. Use Lemma 12 to obtain L and I the composition of leaves and internal vertices, respectively. By Lemma 4, the query $S_2(S)$ counts each leaf once, each degree 2 vertex twice, and the center Δ times. Hence, the label of the center (which is the only internal vertex with degree $\Delta \geq 3$) is given by the only non-zero coordinate of $S_2(S) - L - 2I$. \square

3.1. Stars subdivided at most once

Theorem 16. The class of stars subdivided at most once labeled using an alphabet on two symbols is reconstructible with four multiset-queries.

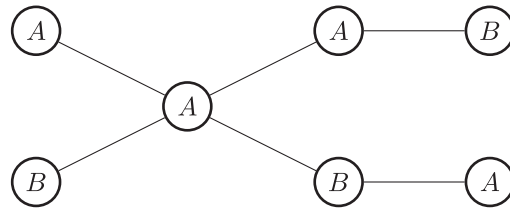


Fig. 2. A labeled star subdivided at most once used for illustrating the proof of Theorem 16 on alphabet of size 2.

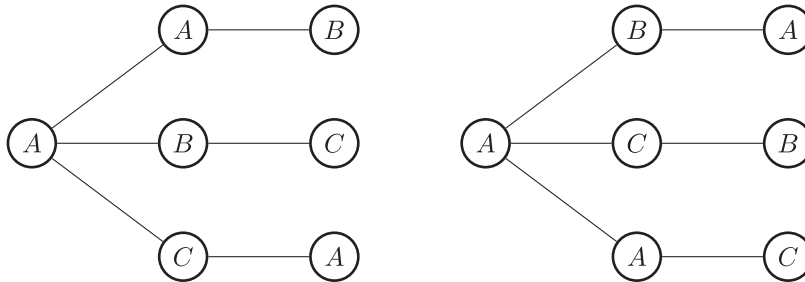


Fig. 3. Two non-sum-reconstructible labeled stars subdivided at most once with alphabet size 3.

Proof. Let $\Sigma = (A, B)$, and $S = (S, \lambda)$ be a labeled graph such that S is a star subdivided at most once. Note that, if S is a path, then it contains at most five vertices and we are done by [1], hence assume that there is a vertex of degree at least 3. First, use Lemma 12 to obtain the composition of leaves and internal vertices and the label of the center vertex with Lemma 15, for a total of three queries. Recall that we know, from $S_n(S)$, the multiset of all labels for vertices of S . The query $\mathcal{M}_{n-2}(S)$ gives the composition of all substars of order $n - 2$. Now, $R = \bigsqcup_{X \in \mathcal{M}_{n-2}(S)} (S_n(S) - X)$ gives the compositions of the following: all branches of length 2, and all pairs of leaves. Since we know the composition of leaves, we can remove all pairs of leaves from R and obtain the branches of length 2. Since we know the composition of branches of length 2 and the label of the center, we can deduce the labels of leaves adjacent to the center.

Now, all that remains is to decide, for each branch of length 2, which vertex is a leaf and which is internal. However, since we know the composition of internal vertices (from which we can remove the center), we can apply the following: for each branch with composition A^2 (resp. B^2), a leaf adjacent to an internal vertex with label A (resp. B) has label A (resp. B) too; the remaining branches have composition AB and thus leaves adjacent to an internal vertex with label A (resp. B) have label B (resp. A). This concludes the reconstruction of S . \square

Remark 17. We note that by [5, Theorem 2.2.4], we can identify the structures of subdivided stars from each others based on the numbers of their connected subgraphs and the number of vertices in these subgraphs. This implies that the proof of Theorem 16 (without restriction to 4 multiset-queries) can also be done without knowing the exact structure of the subdivided star (although we need to know that the underlying graph we are considering is a subdivided star).

Example. Consider the labeled star subdivided at most once S depicted in Fig. 2.

The queries $S_7(S)$ and $S_6(S)$ output $(4, 3)$ and $(14, 10)$, respectively. Since $(14, 10) - (4 - 1) \times (4, 3) = (2, 1)$, two (resp. one) internal vertices have label A (resp. B), and hence two leaves have label A and two leaves have label B . The query $S_2(S)$ outputs $(8, 4)$. Since $S_2(S) - L - 2I = (8, 4) - (2, 2) - 2 \times (2, 1) = (2, 0)$, the center vertex has label A , implying that the two degree 2 vertices have labels A and B , respectively. Finally, the query $\mathcal{M}_5(S)$ outputs $\{\{A^2B^3, 6A^3B^2, A^4B^1\}\}$, hence $R = \bigsqcup_{X \in \mathcal{M}_5(S)} (S_7(S) - X) = \{\{A^2, 6AB, B^2\}\}$. Since the combination of pairs of leaves is $\{\{A^2, 4AB, B^2\}\}$, the branches of length 2 both have a vertex labeled by A and a vertex labeled by B . Removing A^3B^2 (for the center and those branches) from A^4B^3 gives the labels of the leaves adjacent to the center: A and B . As for the branches of length 2, they have composition AB , so the leaf adjacent to the internal vertex labeled A (resp. B) has label B (resp. A), and we are done.

Remark 18. Note that the reconstruction method described in the proof of Theorem 16 does not work for larger alphabets. This is due to the fact that the branches of length 2 offer more choices, as depicted in Fig. 3: we may know the composition of branches of length 2 and of leaves, but this does not allow for an immediate reconstruction. However, note that those two subdivided stars are not confusable, hence they are reconstructible but not sum-reconstructible.

3.2. Bistars

Theorem 19. *The class of bistars labeled with an alphabet using two symbols is reconstructible with three multiset-queries.*

Proof. Let $\Sigma = (A, B)$, and $\mathcal{B} = (G, \lambda)$ be a labeled graph such that G is a bistar with center vertices u and v , vertex u (resp. v) having ℓ_u (resp. ℓ_v) adjacent leaves. First, use Lemma 12 to obtain the composition of leaves and of u and v . Now, there are two cases to consider.

Case 1: u and v have the same label. Assume without loss of generality that they both have label A . Let a_u and b_u (resp. a_v, b_v) be the (yet unknown) number of leaves attached to u (resp. v) labeled with A and B , respectively. Denote by α and β the total number of vertices labeled with A and B , respectively (which are known from $S_n(\mathcal{B})$). Note that we have $\alpha = a_u + a_v + 2$ and $\beta = b_u + b_v$, as well as $a_u + b_u = \ell_u$ and $a_v + b_v = \ell_v$.

The query $\mathcal{M}_3(\mathcal{B})$ outputs the composition of connected subgraphs of order 3, from which we can obtain x and y the number of subgraphs with compositions A^3 and A^2B , respectively.

We have $x = a_u + a_v + \binom{a_u}{2} + \binom{a_v}{2}$ since subgraphs with composition A^3 are exactly the subgraphs which either contain both centers together with a leaf labeled with A (leading to terms a_u and a_v) or the subgraphs containing one of the centers together with two of its adjacent leaves labeled with A (leading to terms $\binom{a_u}{2}$ and $\binom{a_v}{2}$). Similarly, we have $y = b_u + b_v + a_u b_u + a_v b_v$ since the subgraphs with composition A^2B are exactly the subgraphs which either contain both centers together with a leaf labeled with B (leading to terms b_u and b_v) or the subgraphs containing one of the centers together with two of its adjacent leaves each labeled differently (leading to terms $a_u b_u$ and $a_v b_v$).

Note that if $x = 0$ then we are done (all leaves have label B), and likewise if $y = 0$ (all leaves have label A). Hence, assume that $x, y > 0$, we have:

$$x = a_u + a_v + \binom{a_u}{2} + \binom{a_v}{2} = \frac{\alpha - 2 + a_u^2 + a_v^2}{2},$$

which implies that:

$$a_u^2 + a_v^2 = 2x - \alpha + 2. \tag{1}$$

Additionally, we have:

$$\begin{aligned} y &= b_u + b_v + a_u b_u + a_v b_v \\ &= \beta + a_u(\ell_u - a_u) + a_v(\ell_v - a_v) && \text{(by } a_u + b_u = \ell_u \text{ and } a_v + b_v = \ell_v) \\ &= \beta + \ell_u a_u + \ell_v a_v - (a_u^2 + a_v^2) \\ &= \beta + \ell_u a_u + \ell_v(\alpha - a_u - 2) - (a_u^2 + a_v^2) && \text{(by } \alpha = a_u + a_v + 2) \\ &= \beta + \ell_u a_u + \ell_v(\alpha - a_u - 2) - (2x - \alpha + 2) && \text{(by Eq. (1))} \\ &= \beta + a_u(\ell_u - \ell_v) + \alpha(\ell_v + 1) - 2(\ell_v + x + 1), \end{aligned}$$

which implies that:

$$a_u = \frac{y - \beta - \alpha(\ell_v + 1) + 2(\ell_v + x + 1)}{\ell_u - \ell_v}.$$

From this, we can now find a_v, b_u and b_v .

Case 2: u and v have different labels. Let a (resp. y) and x (resp. b) be the (yet unknown) number of leaves attached to the internal vertex with label A (resp. B) and themselves labeled with A and B , respectively. Denote by α and β the total number of vertices labeled with A and B , respectively (which are known from $S_n(\mathcal{B})$). Note that we have $\alpha = a + y + 1$ and $\beta = b + x + 1$.

The query $\mathcal{M}_2(\mathcal{B})$ outputs the composition of pairs of connected vertices, from which we can obtain α' and β' the number of such pairs with composition A^2 and B^2 , respectively. Note that we necessarily have $a = \alpha'$ and $b = \beta'$. This allows us to also obtain $x = \beta - \beta' - 1$ and $y = \alpha - \alpha' - 1$. We can now use those values to deduce whether $\ell_u = x + a$ and $\ell_v = b + y$, or $\ell_v = x + a$ and $\ell_u = y + b$; note that the two possibilities can be true simultaneously if and only if $\ell_u = \ell_v$ (in which case, we are trivially done). Hence, we can find the labels of u and v as well as the labels of the attached leaves. This concludes the case analysis; note that we use two queries, followed by one more query in both cases, so three queries in total. \square

Theorem 20. *The class of bistars labeled with an alphabet using at least three symbols is reconstructible. Furthermore, if the two centers have different labels, then, this can be done in three multiset-queries.*

Proof. Let $\mathcal{B} = (B, \lambda)$ be a labeled graph such that B is a bistar with center vertices u and v , vertex u (resp. v) having ℓ_u (resp. ℓ_v) adjacent leaves. First, use Lemma 12 to obtain the composition of leaves and of u and v . There are two cases to consider.

Case 1: u and v have different labels. We will apply a similar technique as in the proof of the same case in [Theorem 19](#). Assume without loss of generality that u and v have labels X and Y , denote the (yet unknown) number of leaves attached to the vertex with label X (resp. Y) by ℓ_x (resp. ℓ_y); we want to decide whether $\ell_u = \ell_x$ or $\ell_u = \ell_y$. Note that we also know x (resp. y) the number of vertices labeled with X (resp. Y).

The query $\mathcal{M}_2(\mathcal{B})$ outputs the composition of pairs of connected vertices. There are three types of combinations: XY , XZ for $Z \neq Y$ and YZ for $Z \neq X$. The combinations XZ for $Z \neq Y$ give us all leaves adjacent to the vertex with label X , denote by z_x their number. The combinations YZ for $Z \neq X$ give us all leaves adjacent to the vertex with label Y , denote by z_y their number. From those, we also know x_2 and y_2 , the numbers of combinations XX and YY , respectively. Finally, let α (resp. β) be the (yet unknown) number of vertices labeled with Y (resp. X) attached to the internal vertex with label X (resp. Y). We have $x = \beta + x_2 + 1$ and $y = \alpha + y_2 + 1$, from which we can deduce α and β . Furthermore, we have $\ell_x = z_x + \alpha$ and $\ell_y = z_y + \beta$, so we can decide whether $\ell_u = \ell_x$ or $\ell_u = \ell_y$ (if both are true simultaneously, then, $\ell_u = \ell_v$ and we are trivially done).

Case 2: u and v have the same label.

Let us denote by A the label of vertices u and v . Since u and v have the same label, we may consider the largest composition which contains only one A (this is found by exploring queries $\mathcal{M}_i(\mathcal{B})$ for decreasing values of i , starting from $i = \ell_u + \ell_v + 2$). The single A in this composition is either u or v (unless every vertex is labeled with A , in which case the reconstruction is trivially done). Hence, this composition gives every non- A label on one side of the bistar. Denote the total number of these non- A symbols by β_1 . As we can obtain the total number of each symbol with 1-vertex subgraphs, we can also obtain the number of non- A symbols on the other side of the bistar, denote this by β_2 . Hence, we only need to solve whether we have β_1 non- A symbols adjacent to u or adjacent to v . Note that when $\beta_1 = \beta_2$ or $\ell_u = \ell_v$, we are already done.

Assume without loss of generality that $\ell_v > \ell_u$ and let $\ell_v = \ell_u + \ell$. We denote by A_u (resp. A_v) and B_u (resp. B_v) the number of symbols A and non- A symbols adjacent to u (resp. v), respectively. Consider next the number of compositions A^3 (obtained using $\mathcal{M}_3(\mathcal{B})$) and denote it by a_3 . Since each A^3 contains either two leaves adjacent to the same center, or both centers together with a single leaf, we have

$$a_3 = \binom{A_u}{2} + \binom{A_v}{2} + A_u + A_v = \frac{A_u^2 + A_v^2 + A_u + A_v}{2}.$$

Furthermore, as we know the total number of vertices labeled with A , denote it by α , we also know that $A_u + A_v = \alpha - 2$. Hence, we have $2a_3 - \alpha + 2 = A_u^2 + A_v^2$. Furthermore, we can get

$$A_u^2 + A_v^2 = (\ell_u - B_u)^2 + (\ell_v - B_v)^2 = \ell_u^2 - 2\ell_u B_u + B_u^2 + \ell_v^2 - 2\ell_v B_v + B_v^2.$$

Note that we have $\beta_1^2 + \beta_2^2 = B_u^2 + B_v^2$. Hence, we have $\frac{-2a_3 + \alpha - 2 + \beta_1^2 + \beta_2^2 + \ell_u^2 + \ell_v^2}{2} = \ell_u B_u + \ell_v B_v = (B_u + B_v)\ell_u + \ell B_v$. Therefore, we can get

$$B_v = \frac{-2a_3 + \alpha - 2 + \beta_1^2 + \beta_2^2 + \ell_u^2 + \ell_v^2 - 2(\beta_1 + \beta_2)\ell_u}{2\ell}.$$

As we can compute everything on the right hand side, we can compute B_v . Furthermore, we have $\{B_v, B_u\} = \{\beta_1, \beta_2\}$. As we mentioned above, this is enough to reconstruct the labeling of \mathcal{B} . \square

3.3. Stars $S_{1,1,m}$

In this section, we study paths with one leaf false-twinned (that is, we create a false twin of one leaf). As shown with the proof of [Theorem 21](#), this very simple operation changes the status of the graph: recall that some paths are confusable [1]; we show that false-twinning one of the leaves makes the graph sum-reconstructible with a linear number of queries, and the proof is quite technical. However, we will see later ([Theorem 23](#) in [Section 4](#)) that true-twinning a leaf (that is, creating a twin of it) creates a graph class exhibiting another behavior (alternating between sum-reconstructible and confusable with the parity of m).

Recall that we have shown in [Section 2](#) that subdivided stars of type $S_{1,1,m}$ allow more labelings than paths using $m + 3$ labels. Furthermore, since it is relatively simple to construct $S_{1,1,m}$, these graphs seem suitable candidates for applications related to advanced information storages.

Theorem 21. *For any nonnegative integer m , the subdivided star $\mathcal{S} = (S_{1,1,m}, \lambda)$ is sum-reconstructible.*

Proof. Note that P_3 is sum-reconstructible by [Corollary 14](#), so let m be a positive integer. Let $\mathcal{S} = (S, \lambda)$ where $S = S_{1,1,m}$, denote the two leaves attached to the center of \mathcal{S} by u_1 and u_2 , and the vertices from the center to the end of the branch of length m by v_1, v_2, \dots, v_{m+1} (so the center is v_1 and v_{m+1} is a leaf).

For readability, we will use the notation of a given query $S_i(\mathcal{S})$ as a sum of labels, such that X corresponds to the label of vertex x . For example, [Lemma 4](#) implies that for every positive m :

$$S_2(\mathcal{S}) = U_1 + U_2 + V_{m+1} + 3V_1 + 2(V_2 + \dots + V_m). \tag{2}$$

Note that, from Lemma 15, we can compute the label of v_1 , and from Lemma 12 the composition of the three leaves $(U_1 + U_2 + V_{m+1})$ and of all internal vertices $(\sum_{i=2}^m V_i)$. We first manage small edge cases:

- Corollary 14 implies that the case $m \leq 1$ already holds.
- Let $m = 2$. Since we already know the label of v_1 , the other internal label is v_2 . Furthermore, we have $S_3(S) = 2U_1 + 2U_2 + 4V_1 + 3V_2 + V_3$, allowing us to reconstruct u_1 and u_2 , and thus v_3 , completing the reconstruction of S .
- Let $m = 3$ (so we are reconstructing $S = (S_{1,1,3}, \lambda)$), we compute $S_4(S) - S_3(S) = (2(U_1 + U_2) + 4V_1 + 4V_2 + 3V_3 + V_4) - (2(U_1 + U_2) + 4V_1 + 4V_2 + 2V_3 + V_4) = V_3$. We can then deduce the label of v_2 which is the last of the three internal vertices (recall that we have already obtained the label of the center vertex v_1). Next, we use Eq. (2) to compute $S_3(S) - S_2(S) = U_1 + U_2 + V_1 + 2V_2$, obtaining the labels of u_1 and u_2 , after which we can find the label of v_4 . This ends the reconstruction for $m = 3$.

Hence, for the rest of the proof, let us assume that $m \geq 4$.

There are three types of subgraphs of order i in S (for $3 \leq i \leq m + 1$): one subgraph containing both u_1 and u_2 as well as v_1, \dots, v_{i-2} , two subgraphs each containing one of either u_1 or u_2 as well as v_1, \dots, v_{i-1} , and $m + 2 - i$ paths containing vertices v_j, \dots, v_{j+i-1} for $j \in \{1, \dots, m + 2 - i\}$.

In the following, we consider two cases, depending on the parity of m .

Case 1: m is odd, so $m = 2p - 1$ for $p \geq 3$. We will consider $2p - 2$ queries, which will give the following results. First, for $i \in \{3, \dots, p\}$,

$$S_i(S) = 2(U_1 + U_2) + \sum_{j=1}^{i-2} ((j + 3)V_j) + (i + 1)V_{i-1} + \sum_{j=i}^{2p-i+1} (iV_j) + \sum_{j=1}^{i-1} (jV_{2p-j+1}). \tag{3}$$

Indeed, the connected subgraphs of order i are exactly the following: one subgraph containing the two leaves u_1 and u_2 together with the first $i - 2$ vertices of the branch of length m (v_1, \dots, v_{i-2}), two subgraphs containing the first $i - 1$ vertices of the branch of length m (v_1, \dots, v_{i-1}) and each one of the two leaves (either u_1 or u_2), and each of the $m - i + 2$ subpaths of length i of the branch of length m (the first one containing vertices v_1, \dots, v_i and the last one containing vertices $v_{m-i+2}, \dots, v_{m+1}$).

Then, for $i \in \{1, \dots, p\}$ (the counting is similar as above, and we omit the details),

$$S_{p+i}(S) = 2(U_1 + U_2) + \sum_{j=1}^{p-i} ((j + 3)V_j) + \sum_{j=p-i+1}^{p+i-2} ((p - i + 4)V_j) + (p - i + 3)V_{p+i-1} + \sum_{j=1}^{p-i+1} (jV_{2p-j+1}). \tag{4}$$

Claim 21.1. For $i \in \{1, \dots, p - 1\}$,

$$S_{p+i}(S) - S_{p+i+1}(S) = \sum_{j=p-i+1}^{p+i-2} (V_j) - V_{p+i}.$$

Proof of Claim 21.1. By applying Eq. (4) twice (once for $S_{p+i}(S)$ and once for $S_{p+i+1}(S)$), we obtain:

$$\begin{aligned} S_{p+i}(S) - S_{p+i+1}(S) &= \left(2(U_1 + U_2) + \sum_{j=1}^{p-i} ((j + 3)V_j) + \sum_{j=p-i+1}^{p+i-2} ((p - i + 4)V_j) \right. \\ &\quad \left. + (p - i + 3)V_{p+i-1} + \sum_{j=1}^{p-i+1} (jV_{2p-j+1}) \right) \\ &\quad - \left(2(U_1 + U_2) + \sum_{j=1}^{p-i-1} ((j + 3)V_j) + \sum_{j=p-i}^{p+i-1} ((p - i + 3)V_j) \right. \\ &\quad \left. + (p - i + 2)V_{p+i} + \sum_{j=1}^{p-i} (jV_{2p-j+1}) \right) \\ &= (p - i + 3 - (p - i + 3))V_{p-i} + \sum_{j=p-i+1}^{p+i-2} (V_j) \\ &\quad + (p - i + 3 - (p - i + 3))V_{p+i-1} + (p - i + 1 - (p - i + 2))V_{p+i} \\ &= \sum_{j=p-i+1}^{p+i-2} (V_j) - V_{p+i}. \quad \square \end{aligned}$$

Claim 21.2. For $i \in \{3, \dots, p\}$,

$$S_{i+1}(S) - S_i(S) = V_{i-1} + 2V_i + \sum_{j=i+1}^{2p-i} (V_j).$$

Proof of Claim 21.2. By applying Eq. (3) twice for $i \in \{3, \dots, p - 1\}$, we obtain:

$$\begin{aligned} S_{i+1}(S) - S_i(S) &= \left(2(U_1 + U_2) + \sum_{j=1}^{i-1} ((j + 3)V_j) + (i + 2)V_i \right. \\ &\quad \left. + \sum_{j=i+1}^{2p-i} ((i + 1)V_j) + \sum_{j=1}^i (jV_{2p-j+1}) \right) \\ &\quad - \left(2(U_1 + U_2) + \sum_{j=1}^{i-2} ((j + 3)V_j) + (i + 1)V_{i-1} \right. \\ &\quad \left. + \sum_{j=i}^{2p-i+1} (iV_j) + \sum_{j=1}^{i-1} (jV_{2p-j+1}) \right) \\ &= (i + 2 - (i + 1))V_{i-1} + (i + 2 - i)V_i + \sum_{j=i+1}^{2p-i} (V_j) + (i - i)V_{2p-i+1} \\ &= V_{i-1} + 2V_i + \sum_{j=i+1}^{2p-i} (V_j). \end{aligned}$$

We note that $S_{p+1}(S) - S_p(S)$ can be obtained with similar calculations by applying (Eq. (4)) for $S_{p+1}(S)$. \square

We next compute the labels of vertices in the following way: alternating between $S_{p+i+1}(S) - S_{p+i}(S)$ for $i \in \{1, \dots, p-1\}$ and $S_{i'+1}(S) - S_{i'}(S)$ for $i' \in \{p, \dots, 3\}$ (where i' is iterated from p to 3), we will be able to find the labels of v_{p+i} and both $v_{i'-1}$ and $v_{i'}$, respectively. We prove that we can do those computations by induction on $i \in \{1, \dots, p - 2\}$ and $i' \in \{p, \dots, 3\}$ (with i increasing and i' decreasing at each step; so $i + i' = p + 1$). First, consider the case where $(i, i') = (1, p)$ and $p > 2$. Using Claim 21.1 we compute:

$$S_{p+2}(S) - S_{p+1}(S) = V_{p+1} - \sum_{j=p-1+1}^{p+1-2} (V_j) = V_{p+1}$$

which gives the label of v_{p+1} . Then, using Claim 21.2, we compute:

$$S_{p+1}(S) - S_p(S) = V_{p-1} + 2V_p + \sum_{j=p+1}^{2p-p} (V_j) = V_{p-1} + 2V_p$$

from which we can derive the labels of v_{p-1} and v_p (if $V_p = V_{p-1}$, the same label appears thrice; otherwise, v_p has the label that appears twice and v_{p-1} the label that appears once).

Assume now that the statement holds for a pair $(i - 1, i' + 1)$, so we know the labels of $v_{p-i+1}, \dots, v_{p+i-1}$. By Claim 21.1, we compute:

$$\begin{aligned} S_{p+i+1}(S) - S_{p+i}(S) &= V_{p+i} - \sum_{j=p-i+1}^{p+i-2} (V_j) \\ &= V_{p+i} - \sum_{j=i'}^{p+i-2} (V_j) \\ &= V_{p+i} - Q, \end{aligned}$$

where Q is a known quantity by induction hypothesis, hence we obtain the label of v_{p+i} . Therefore, we obtain labels V_p, \dots, V_{p+i} . Using now Claim 21.2, we compute:

$$S_{i'+1}(S) - S_{i'}(S) = V_{i'-1} + 2V_{i'} + \sum_{j=i'+1}^{2p-i'} (V_j)$$

$$\begin{aligned}
 &= V_{i'-1} + 2V_{i'} + \sum_{j=i'+1}^{p+i-1} (V_j) \\
 &= V_{i'-1} + Q',
 \end{aligned}$$

where Q' is a known quantity by induction hypothesis, hence we obtain the label of $v_{i'-1}$. Hence, when i traverses from 1 to $p - 2$ and i' from p to 3, we are able to compute the labels of v_2, \dots, v_{2p-2} (and the label of v_1 was already known). Now, using one last time [Claim 21.1](#), we compute

$$S_{2p}(S) - S_{2p-1}(S) = V_{2p-1} - \sum_{j=2}^{2p-3} (V_j) = V_{2p-1} + Q'',$$

where Q'' is a known quantity, allowing us to find the label of v_{2p-1} . We now only have to find the label of v_{2p} . We observe that $S_{2p+1}(S) = 2(U_1 + U_2) + \sum_{j=1}^{2p-1} (3V_j) + 2V_{2p}$, which leads to:

$$\begin{aligned}
 S_{2p+1}(S) - S_{2p}(S) &= 2(U_1 + U_2) + \sum_{j=1}^{2p-1} (3V_j) + 2V_{2p} - 2(U_1 + U_2) - \sum_{j=1}^{2p-2} (4V_j) - 3V_{2p-1} - V_{2p} \\
 &= V_{2p} - \sum_{j=1}^{2p-2} V_j \\
 &= V_{2p} + Q''',
 \end{aligned}$$

where Q''' is a known quantity. The labels of u_1 and u_2 are now the two labels of leaves remaining. Recall that we already found the composition $U_1 + U_2 + V_{m+1}$. Hence, we can also deduce the labels of u_1 and u_2 .

This concludes the case where m is odd.

Case 2: m is even, so $m = 2p$ for $p \geq 2$. Let us first consider the case with $p = 2$ (so we are reconstructing $S = (S_{1,1,4}, \lambda)$), which will serve as a concrete example of the case disjunction that we are going to need for this case. This is necessary since one main equation needed for the proof ([Claim 21.3](#)) does not hold for this small graph. Denote by L the composition of leaves and by I the composition of the internal vertices. We have $S_5(S) - S_4(S) = V_4 - V_2$. We divide our considerations according to whether we have $V_2 \neq V_4$ or $V_2 = V_4$. If $V_2 \neq V_4$, then we can know V_2 and V_4 . Hence, we may deduce $V_3 = I - V_1 - V_2 - V_4$ and U_1, U_2 from $S_3(S) - S_2(S) = U_1 + U_2 + V_1 + 2V_2 + V_3$. Finally, we may finish with $V_5 = L - U_1 - U_2$. If $V_2 = V_4$, then we compute $S_4(S) - S_3(S) = V_2 + V_3$, so if $V_2 = V_3$ we are done (this gives their value, and we then conclude as above). Assume then that $V_2 \neq V_3$, we use [Eq. \(2\)](#) to compute $S_6(S) - S_2(S) = U_1 + U_2 + V_2 + V_3 + V_4 + V_5 = L + 2V_2 + V_3$ where L is the sum of the labels of leaves (which is known). This gives us the values of V_2 and V_3 , and we then conclude as above. For the rest of Case 2, we assume that $p \geq 3$.

We will consider $2p - 1$ queries, which will give the following results (the counting is similar as in Case 1, so we omit the details). First, for $i \in \{3, \dots, p + 1\}$,

$$S_i(S) = 2(U_1 + U_2) + \sum_{j=1}^{i-2} ((j + 3)V_j) + (i + 1)V_{i-1} + \sum_{j=i}^{2p-i+2} (iV_j) + \sum_{j=1}^{i-1} (jV_{2p-j+2}). \tag{5}$$

Then, for $i \in \{1, \dots, p\}$,

$$S_{p+1+i}(S) = 2(U_1 + U_2) + \sum_{j=1}^{p-i+1} ((j + 3)V_j) + \sum_{j=p-i+2}^{p-i-1} ((p + 4 - i)V_j) + (p + 3 - i)V_{p+i} + \sum_{j=1}^{p-i+1} (jV_{2p-j+2}). \tag{6}$$

Claim 21.3. For $i \in \{3, \dots, p\}$:

$$S_{i+1}(S) - S_i(S) = V_{i-1} + 2V_i + \sum_{j=i+1}^{2p-i+1} (V_j).$$

Proof of Claim 21.3. By applying [Equation \(5\)](#) twice, we obtain for $i \in \{3, \dots, p\}$,

$$\begin{aligned}
 S_{i+1}(S) - S_i(S) &= \left(2(U_1 + U_2) + \sum_{j=1}^{i-1} ((j + 3)V_j) + (i + 2)V_i + \sum_{j=i+1}^{2p-i+1} ((i + 1)V_j) + \sum_{j=1}^i (jV_{2p-j+2}) \right) \\
 &\quad - \left(2(U_1 + U_2) + \sum_{j=1}^{i-2} ((j + 3)V_j) + (i + 1)V_{i-1} + \sum_{j=i}^{2p-i+2} (iV_j) + \sum_{j=1}^{i-1} (jV_{2p-j+2}) \right)
 \end{aligned}$$

$$\begin{aligned}
 &= ((i - 1 + 3) - (i + 1)) V_{i-1} + (i + 2 - i) V_i + \sum_{j=i+1}^{2p-i+1} (V_j) + (i - i) V_{2p-i+2} \\
 &= V_{i-1} + 2V_i + \sum_{j=i+1}^{2p-i+1} (V_j). \quad \square
 \end{aligned}$$

Claim 21.4. For $i \in \{1, \dots, p - 1\}$:

$$S_{p+1+i}(S) - S_{p+2+i}(S) = V_{p-i+1} - V_{p+i+1} + \sum_{j=p-i+2}^{p+i-1} (V_j).$$

Proof of Claim 21.4. By applying Equation (6) twice, we obtain for $i \in \{1, \dots, p - 1\}$,

$$\begin{aligned}
 S_{p+1+i}(S) - S_{p+2+i}(S) &= \left(2(U_1 + U_2) + \sum_{j=1}^{p-i+1} ((j + 3)V_j) + \sum_{j=p-i+2}^{p+i-1} ((p + 4 - i)V_j) \right. \\
 &\quad \left. + (p + 3 - i)V_{p+i} + \sum_{j=1}^{p-i+1} (jV_{2p-j+2}) \right) \\
 &\quad - \left(2(U_1 + U_2) + \sum_{j=1}^{p-i} ((j + 3)V_j) + \sum_{j=p-i+1}^{p+i} ((p + 3 - i)V_j) \right. \\
 &\quad \left. + (p + 2 - i)V_{p+i+1} + \sum_{j=1}^{p-i} (jV_{2p-j+2}) \right) \\
 &= ((p - i + 1 + 3) - (p + 3 - i)) V_{p-i+1} + \sum_{j=p-i+2}^{p+i-1} (V_j) \\
 &\quad + ((p + 3 - i) - (p + 3 - i)) V_{p+i} + ((p - i + 1) - (p + 2 - i)) V_{p+i+1} \\
 &= V_{p-i+1} - V_{p+i+1} + \sum_{j=p-i+2}^{p+i-1} (V_j). \quad \square
 \end{aligned}$$

We now start computing the labels of vertices. First, note that by Claim 21.3 with $i = p$ we have:

$$S_{p+1}(S) - S_p(S) = V_{p-1} + 2V_p + V_{p+1}. \tag{7}$$

We can derive two subcases from Eq. (7).

Subcase 2.1: $V_{p-1} \neq V_{p+1}$, or $V_{p-1} = V_p = V_{p+1}$. In this case, $S_{p+1}(S) - S_p(S)$ clearly allows us to deduce V_p (it is the value that is repeated at least twice). From Eqs. (5) and (6) for $i = p + 1$ and $i = 1$, respectively, we obtain $S_{p+2}(S) - S_{p+1}(S) = V_p + V_{p+1}$. Since we already know V_p , we obtain the label of v_{p+1} , and thus the label of v_{p-1} from Eq. (7). Now, like in Case 1, we use induction and alternate between $S_{p+1+i}(S) - S_{p+2+i}(S)$ and $S_{i'+1}(S) - S_{i'}(S)$, for increasing $i \in \{1, \dots, p - 3\}$ and decreasing $i' \in \{p - 1, \dots, 3\}$, respectively. Doing so allows us to know the labels of $v_{i'-1}, \dots, v_{p+i+1}$ for the pair (i, i') where $i + i' = p$. Hence, our induction hypothesis is that after computing the pair (i, i') , we know the labels of $v_{i'-1}, \dots, v_{p+i+1}$. Once this is done, we finish with $i \in \{p - 2, p - 1\}$. By Claim 21.4 and induction hypothesis, we have

$$S_{p+1+i}(S) - S_{p+2+i}(S) = V_{p-i+1} - V_{p+i+1} + \sum_{j=p-i+2}^{p+i-1} (V_j), \tag{8}$$

which allows us to compute the label of v_{p+i+1} . By Claim 21.3, we also have

$$S_{i'+1}(S) - S_{i'}(S) = V_{i'-1} + 2V_{i'} + \sum_{j=i'+1}^{2p-i'+1} (V_j).$$

Hence, the induction hypothesis together with the label of v_{p+i+1} allows us to compute the label of $v_{i'-1}$. Note that as we have mentioned above, there are more potential values for i than for i' in order to compute all the internal labels. Hence, we first alternate between i and i' until the case $(p - 2, 3)$. At this point, we know the labels of v_1, \dots, v_{2p-2} . After this, we use Eq. (8) for $i = p - 2$ and $i = p - 1$ which gives the labels of v_{2p-1} and v_{2p} . We can again conclude by computing

the label of v_{2p+1} (from $S_{2p+1}(S) - S_{2p+2}(S) = \sum_{i=1}^{2p-1} (V_i) - V_{2p+1}$), allowing us to deduce the labels of u_1 and u_2 , and thus completing the reconstruction.

Subcase 2.2: $V_{p-1} = V_{p+1}$ and $V_{p-1} \neq V_p$. In this case, Eq. (7) gives $S_{p+1}(S) - S_p(S) = V_{p-1} + 2V_p + V_{p+1}$ and does not allow us to decide on the values of V_{p-1} and V_p . Now, like in Case 1 and Subcase 2.1, we will use induction and alternate operations. However, this time, we will prove by induction either that, for values of i that will be given later, $V_{2i} = V_{2i+2}$ and $V_{2i-1} = V_{2i+1}$ (depending on the parity of the induction step) and that $V_{2i-1} \neq V_{2i}$, or that we are able to determine the values of V_{p-i}, \dots, V_{p+i} (after which we start a second induction, exactly like in Case 1 and Subcase 2.1, which will allow us to compute the remaining labels V_j).

To start the induction, note that $V_{p-1} = V_{p+1} \neq V_p$, and let us denote for readability and without loss of generality that $V_{p-1} = V_{p+1} = A$ and $V_p = B$ (though we do not know these values yet). Our induction hypothesis is the following: For a given $i \in \{0, \dots, \lfloor \frac{p-3}{2} \rfloor\}$, we have $V_{p-2j} = V_{p+2j} = B$ and $V_{p-2j-1} = V_{p+2j+1} = A$ for each $j \in \{0, \dots, i\}$. Note that it holds for the base case $i = 0$.

Every full step of the induction is comprised of an *even* and an *odd* substep, each seeing two computations using Claims 21.3 and 21.4. At any point during those two substeps, we may actually “collapse” all values (that is, be able to know the values of V_{p-1} and V_p , and thus all the other values that are equal to those) and end this induction. During the even substep, we try to compute the values of V_{p+2i} and then of V_{p-2i} . If we are able to determine their actual value, then the induction ends. Otherwise, we get $V_{p-2i} = V_{p+2i} = V_p$. Then, during the odd substep, we try to compute the values of V_{p+2i+1} and then of V_{p-2i-1} . Again, either we are able to determine them and the induction ends, or we get $V_{p-2i-1} = V_{p+2i+1} = V_{p-1}$. In the latter case, we continue the induction with the next full step.

We first detail the *even* substep of the full induction step, where we try to compute the values of V_{p+2i} and then of V_{p-2i} . By induction hypothesis, we have $V_{p-2i+2} = V_{p-2i+4} = \dots = V_{p+2i-2} = B$ and $V_{p-2i+1} = V_{p-2i+3} = \dots = V_{p+2i-1} = A$. We compute, by Claim 21.4 and induction hypothesis (we have $1 \leq i \leq \frac{p-3}{2}$):

$$\begin{aligned} S_{p+2i}(S) - S_{p+1+2i}(S) &= V_{p-2i+2} - V_{p+2i} + \sum_{j=p-2i+3}^{p+2i-2} (V_j) \\ &= V_{p-2i+2} - V_{p+2i} + \sum_{\substack{j=p-2i+4 \\ j-p \text{ even}}}^{p+2i-2} (V_j) + \sum_{\substack{j=p-2i+3 \\ j-p \text{ odd}}}^{p+2i-3} (V_j) \\ &= B - V_{p+2i} + (2i - 2)B + (2i - 2)A. \end{aligned}$$

Now, if $V_{p+2i} = B$, then, we are still in the induction loop, since we cannot decide the value of B with the equation. Indeed, we can in this case only know that $V_{p+2i} = V_p$. Otherwise, the equation allows us to “collapse” the unknown values of $V_{p-2i+1}, \dots, V_{p+2i-1}$, as well as the value of V_{p+2i} , in which case we end the induction. Indeed, if $V_{p+2i} \neq B$, then there is an imbalance in the multiplicities of A 's and B 's in the above formula and we may assign to vertices $v_{p-2i+1}, \dots, v_{p+2i-1}$ the label A or B with the lesser multiplicity in the formula and to vertices $v_{p-2i+2}, v_{p-2i+4}, \dots, v_{p+2i-2}$ the label with larger multiplicity in the formula. Finally, either V_{p+2i} is equal to V_{p-2i+1} or v_{p+2i} has a third label distinct from A and B . Recall that by induction hypothesis, we have $V_{p-2i+2} = V_{p-2i+4} = \dots = V_{p+2i-2} = B$ and $V_{p-2i+1} = V_{p-2i+3} = \dots = V_{p+2i-1} = A$. Similarly, using Claim 21.3, the induction hypothesis, and the fact that $V_{p+2i} = B$ (we have $1 \leq i \leq \frac{p-3}{2}$):

$$\begin{aligned} S_{p+2-2i}(S) - S_{p+1-2i}(S) &= V_{p-2i} + 2V_{p-2i+1} + \sum_{j=p-2i+2}^{p+2i} (V_j) \\ &= V_{p-2i} + 2A + (2i - 1)A + (2i)B. \end{aligned}$$

Again, if $V_{p-2i} = B$, then, we cannot decide the values with the equation and thus, we remain in the induction loop; otherwise, we can deduce the value of B as well as the unknown values of $V_{2p-2i+1}, \dots, V_{p+2i-1}$. In the latter case, we end this induction.

Let us next detail the *odd* substep of the full induction step, where we try to compute the values of V_{p+2i+1} , and then of V_{p-2i-1} . Note that it is very similar to the even substep. Using both the induction hypothesis and the even substep, we have $V_{p-2i} = V_{p-2i+2} = \dots = V_{p+2i} = B$ and $V_{p-2i+1} = V_{p-2i+3} = \dots = V_{p+2i-1} = A$. We compute, by Claim 21.4 and induction hypothesis (we have $1 \leq i \leq \frac{p-3}{2}$):

$$\begin{aligned} S_{p+1+2i}(S) - S_{p+2+2i}(S) &= V_{p-2i+1} - V_{p+2i+1} + \sum_{j=p-2i+2}^{p+2i-1} (V_j) \\ &= V_{p-2i+1} - V_{p+2i+1} + \sum_{\substack{j=p-2i+2 \\ j-p \text{ even}}}^{p+2i-2} (V_j) + \sum_{\substack{j=p-2i+3 \\ j-p \text{ odd}}}^{p+2i-1} (V_j) \\ &= A - V_{p+2i+1} + (2i - 1)B + (2i - 1)A. \end{aligned}$$

Again, if $V_{p+2i+1} = A$, then, we are still in the induction loop, since we cannot decide the value of A with the equation. Indeed, we can in this case only know that $V_{p+2i+1} = V_{p-1} = V_{p+1}$. Otherwise, the equation allows us to “collapse” the unknown values of $V_{p-2i+1}, \dots, V_{p+2i-1}$, as well as the value of V_{p+2i+1} , in which case we end the induction. Indeed, if $V_{p+2i+1} \neq A$, then there is an imbalance in the multiplicities of A ’s and B ’s in the above formula and we may assign to vertices $v_{p-2i+1}, \dots, v_{p+2i-1}$ the label A or B with the lesser multiplicity in the formula and to vertices $v_{p-2i+2}, v_{p-2i+4}, \dots, v_{p+2i-2}$ the label with the larger multiplicity in the formula. Finally, either V_{p+2i+1} is equal to V_{p-2i+1} or v_{p+2i+1} has a third label distinct from A and B . Recall that by induction hypothesis and after the even substep, we have $V_{p-2i} = V_{p-2i+4} = \dots = V_{p+2i} = B$ and $V_{p-2i+1} = V_{p-2i+3} = \dots = V_{p+2i-1} = A$. Similarly, using Claim 21.3, the induction hypothesis, and the facts that $V_{p+2i+1} = A$ and $V_{p+2i} = V_{p-2i} = B$ (we have $1 \leq i \leq \lfloor \frac{p-3}{2} \rfloor$):

$$\begin{aligned} S_{p-2i+1}(S) - S_{p-2i}(S) &= V_{p-2i-1} + 2V_{p-2i} + \sum_{j=p-2i+1}^{p+2i+1} (V_j) \\ &= V_{p-2i-1} + 2B + (2i + 1)A + (2i)B. \end{aligned}$$

Again, if $V_{p-2i-1} = A$, then, we cannot decide their values with the equation and thus are still in the induction loop; if at least one of those conditions does not hold, then we can deduce their values as well as the unknown values of $V_{2p-2i+1}, \dots, V_{p+2i-1}$, in which case the induction ends.

Hence, either we are able to determine the unknown values (ending the induction at any point during the even or odd substep), after which we complete the reconstruction using the same method as in Case 1 and Subcase 2.1 (we do not give the details here, since this leads to the exactly same computations, so we refer the reader to Case 1), or the induction continues until we reach the last possible value for i .

If the induction does not stop early and we continue in this way until $i = \lfloor \frac{p-3}{2} \rfloor$, then, by (the already proved) induction hypothesis, for an odd p we have $V_2 = V_4 = \dots = V_{2p-2} = A$ and $V_3 = V_5 = \dots = V_{2p-3} = B$. We then compute $S_{2p-1}(S) - S_{2p}(S)$ to obtain V_{2p-1} and $S_{2p}(S) - S_{2p+1}(S)$ to obtain V_{2p} . As above, we check if these computations cause the collapse (as explained above) and allow us to finish the proof as in Case 1. If not, we have $V_2 = V_4 = \dots = V_{2p} = A$ and $V_3 = V_5 = \dots = V_{2p-1} = B$. Recall that we know the labels of the leaves, and let L be the sum of those three labels. Hence, we have $S_1(S) - L - V_1 = pA + (p - 1)B$ and therefore, we can deduce the values of both A and B .

Consider then an even p . We have, by (the already proved) induction hypothesis, $V_3 = V_5 = \dots = V_{2p-3} = A$ and $V_4 = V_6 = \dots = V_{2p-4} = B$. We then compute first $S_4(S) - S_3(S)$ to obtain V_2 . After this, we obtain values of V_{2p-2}, V_{2p-1} and V_{2p} by computing $S_{2p-2}(S) - S_{2p-1}(S)$, $S_{2p-1}(S) - S_{2p}(S)$ and $S_{2p}(S) - S_{2p+1}(S)$, respectively. As above, we check if these computations cause the collapse (as explained above) and allow us to finish the proof as in Case 1. If not, we have $V_2 = V_4 = \dots = V_{2p} = B$ and $V_3 = V_5 = \dots = V_{2p-1} = A$. Hence, we have $S_1(S) - L - V_1 = pB + (p - 1)A$ and therefore, we can again deduce the values of both A and B .

Hence, we know all the labels of v_1, \dots, v_{2p} . Note that knowing the value of V_1 was primordial for this last part, which will be important later in Theorem 23. Finally, by computing $S_{2p+1}(S) - S_{2p+2}(S) = \sum_{i=1}^{2p-1} (V_i) - V_{2p+1}$, we can compute the label of v_{2p+1} , and thus the labels of u_1 and u_2 which are the two other leaves, completing the reconstruction. \square

3.4. Summary

We considered several families of trees, which we were able to reconstruct. There are a few interesting remarks to draw from this. First, note that not all paths are reconstructible [1], but creating a false twin of one of the leaves of a path, that is appending a leaf to its second vertex, makes the graph always reconstructible (Theorem 21 for $S_{1,1,m}$). A natural question would be to further this by appending a leaf to other vertices of a path, studying the reconstructibility of the subdivided star $S_{1,m,t}$. As we will see later (Fig. 6), this family is not reconstructible in general, but it might be possible under some conditions (such as $m \leq \lfloor \frac{t}{4} \rfloor$).

However, the most interesting point is the diversity of algorithms for reconstruction that we found. As explained in Section 2, one can reconstruct a reconstructible graph by enumerating all subgraph composition multisets of all possible labelings and comparing it to the subgraph composition multiset of the graph, but doing so can be extremely costly. We are able to be way more fine-grained, using only two sum-queries to reconstruct a star (Corollary 14), and four queries (one multiset and three sums) for stars subdivided at most once when the alphabet contains two symbols (Theorem 16). For bistars, the alphabet size plays a role: with two symbols, three queries (two sums and one multiset) are enough (Theorem 19), which is also the case for a larger alphabet if the two non-leaves have different labels (Case 1 of Theorem 20). However, if those vertices have different labels, then, we may have to apply a brute-force method to find a specific composition (Case 2 of Theorem 20). Finally, for the star $S_{1,1,m}$, we use $m + 1$ sum-queries, and may have different methods depending on the parity of m (Theorem 21).

These very different methods of reconstruction are particularly interesting, and we conjecture that the methods used above are all optimal, meaning that no reconstruction algorithm for those classes can use fewer queries. This also begs the question of the existence of graphs for which, without using a brute-force reconstruction, the highest possible number of stronger queries (that is, a linear number of multiset-queries) would still be needed. Similarly, the classes we studied either can be reconstructed using a constant and small number of queries, or seem to require a linear number of queries; are there classes that require a logarithmic, or a sublinear but non-constant number of queries?

Another question that arises from the study of bistars and stars subdivided at most once is the part played by alphabet size. Which other classes require different algorithms for reconstruction, and in particular a difference between a constant and a linear number of queries when the alphabet size $k \geq 2$ passes certain thresholds?

4. Smallest confusable graphs

In this section, we enumerate all the smallest confusable graphs and trees, and construct a general family of graphs with a dichotomy between confusable and sum-reconstructible members depending on parity of their order. We also give the smallest non-sum-reconstructible graphs.

Theorem 22. *The smallest confusable graphs have order 5. The smallest confusable tree has order 7.*

Proof. We prove a stronger result by showcasing all the smallest confusable graphs (there are two of them) and the smallest confusable tree (it is unique). This is an expansion on results from [1,5,6]. As in the proof of [Theorem 21](#), for sum-queries, X denotes the label of vertex x .

Smallest confusable graphs. No graphs of order at most 3 can be confusable, since those graphs are exactly the paths and complete graphs.

There are six connected graphs of order 4, three of them (path P_4 , star S_3 and clique K_4) are clearly reconstructible. The cycle C_4 can easily be reconstructed: let A be a label of a vertex, we can immediately finish if either of $A^i \in \mathcal{M}_i(\mathcal{G})$ holds for $i \in \{2, 3, 4\}$; otherwise, if $\mathcal{M}_4(\mathcal{G}) = \{A^2XY\}$ with $X, Y \neq A$ we have two non-adjacent vertices with label A and we are done, and if $\mathcal{M}_4(\mathcal{G}) = \{AXYZ\}$ with $X, Y, Z \neq A$ we use $\mathcal{M}_2(\mathcal{G})$ to find the labels of the two vertices adjacent to the vertex with label A and conclude with the last label. Note however that C_4 is not sum-reconstructible since every vertex appears with the same multiplicity in every $S_i(\mathcal{G})$.

The diamond graph with vertices $V = \{u, v, w, x\}$ and edges $E = \{uv, vw, wx, xu, uw\}$, that is, complete graph K_4 without the edge vx , can easily be reconstructed. Indeed, since the query $S_2(\mathcal{G}) - 2S_1(\mathcal{G})$ gives the labels of u and w , after which we can know the labels of v and x . Vertices u and w , and v and x , respectively, are at symmetrical positions and hence, this is enough.

Finally, the paw (a triangle with a vertex adjacent to one vertex of the triangle) can be reconstructed since the degree 3 vertex is the only one in all three subgraphs of order 3 (the other ones appear in two) so its label can be computed with $S_3(\mathcal{G}) - 2S_1(\mathcal{G})$. After this, the label of the degree 1 vertex can be computed with $2S_1(\mathcal{G}) - S_2(\mathcal{G})$ together with the knowledge of the label of the degree 3 vertex. Finally, we can find the labels of the two degree 2 vertices which are at symmetrical positions. Hence, no graph of order 4 is confusable.

The line graph of the graph in Example 2 in [6] (depicted in [Fig. 1](#) in their paper) is confusable, as proved in [Theorem 23](#) and seen in [Fig. 4](#). Furthermore, the gem (a path plus one universal vertex) is confusable; consider the two labelings depicted in [Fig. 5](#), we have, for both of them, $\mathcal{M}_1(\mathcal{G}) = \{\{3A, 2B\}\}$ $\mathcal{M}_2(\mathcal{G}) = \{\{2A^2, B^2, 4AB\}\}$, $\mathcal{M}_3(\mathcal{G}) = \{\{A^3, 4A^2B, 3AB^2\}\}$ and $\mathcal{M}_4(\mathcal{G}) = \{\{2A^3B, 3A^2B^2\}\}$.

Let us next consider the remaining 19 connected graphs of order 5. Some can be managed easily: K_5 , P_5 , S_4 and $S_{1,1,2}$ are all reconstructible by [1], [Corollary 14](#) and [Theorem 21](#). In the following, we consider the remaining graphs of order 5, see [\[9\]](#) for a list of 5-vertex graphs.

5. K_5 minus an edge. For this graph, the query $S_2(\mathcal{G})$ counts three times the two degree 3 vertices and four times the three degree 4 vertices, which can be separated. Since they are symmetrical, this is enough for reconstruction.
6. K_4 plus one degree 2 vertex. Let u_1 and u_2 be the degree 3 vertices, v_1 and v_2 be the degree 4 vertices, and w be the degree 2 vertex. We have $4S_3(\mathcal{G}) - 6S_2(\mathcal{G}) = 2(U_1 + U_2) + 8W$, from which we can clearly deduce the label of w , and thus the labels of u_1 and u_2 , after which we can easily know the labels of v_1 and v_2 .
7. An edge to which three false twins are connected. The endpoints of the edge can be found using $S_2(\mathcal{G}) - 2S_1(\mathcal{G})$, after which we can deduce the labels of the three twins.
8. K_4 plus a leaf. For this graph, the query $S_2(\mathcal{G})$ counts three times the three degree 3 vertices, four times the degree 4 vertex, and the leaf once. Hence, we have $S_2(\mathcal{G}) - 3S_1(\mathcal{G})$ allows us to find the label of the leaf (which is “negative”) and of the degree 4 vertex. The last three vertices are twins, so this is enough.
9. The cycle C_4 plus a vertex of degree 3. Let u be the degree 2 vertex, v_1 and v_2 be its neighbors (those are false twins), and w_1 and w_2 be the other two vertices (those are true twins). We have $S_3(\mathcal{G}) = 5U + 6(V_1 + V_2) + 5(W_1 + W_2)$, so we can deduce the labels of v_1 and v_2 . Furthermore, $S_2(\mathcal{G})$ counts three times each vertex, except u which is counted twice, hence we can know its label too. This leaves the last two labels for w_1 and w_2 , completing the reconstruction.
10. The wheel W_4 (a cycle plus one universal vertex) can be reconstructed from $S_2(\mathcal{G})$: the four vertices of the cycle are symmetrical and counted three times each while the universal vertex is counted four times.
11. The cycle C_5 can be reconstructed. Assume that the symbol A is used in the labeling. Clearly, if A^5 , A^4 or A^3 appear in $\mathcal{M}_5(\mathcal{G})$, $\mathcal{M}_4(\mathcal{G})$ or $\mathcal{M}_3(\mathcal{G})$, respectively, then we are done. If none holds, but A^2 appears in $\mathcal{M}_2(\mathcal{G})$, then two adjacent vertices are labeled with A and their two neighbors are labeled with other symbols. If the number of A 's (found with $\mathcal{M}_5(\mathcal{G})$) is 3, then, the fifth vertex is labeled with A and we are done. Otherwise, the query $\mathcal{M}_2(\mathcal{G})$ contains exactly two compositions AX and AY , which give us the labels of the two neighbors, and the last vertex has the



Fig. 4. Two equicomposable 5-vertex labeled graphs.



Fig. 5. Two equicomposable labeled gems.

last label. Now, if there are no two consecutive A's, then, there are two possibilities. Assume first that two vertices are labeled with A, they are at distance 2 from each other, we can find the label X of their common neighbor with the composition A^2X in query $\mathcal{M}_3(\mathcal{G})$, and the last two vertices are symmetric, so we are done. Assume next that each vertex has a unique label. Let one of the vertices have label A. We can know the labels of its neighbors with compositions AX and AY in query $\mathcal{M}_2(\mathcal{G})$, after which the same query will contain compositions XW_1, YW_2 and W_1W_2 , from which we can clearly deduce the last two labels.

12. The butterfly (two triangles attached together through a vertex c) can be reconstructed: the label of c can be found with $S_2(\mathcal{G}) - 2S_1(\mathcal{G})$ since it is the only vertex appearing four times in $S_2(\mathcal{G})$ (all the other ones appear twice), and then the multiset-query $\mathcal{M}_2(\mathcal{G})$ can be used to eliminate all the compositions with c in it, leaving only two compositions corresponding to the pairs on each triangle.
13. The kite (a cycle vw_1xw_2 with an edge w_1w_2 , and a leaf u attached to v) can be reconstructed. We have $S_3(\mathcal{G}) - (S_2(\mathcal{G}) + S_5(\mathcal{G})) = V$, after which $2S_2(\mathcal{G}) - S_3(\mathcal{G}) = V + 2(W_1 + W_2) + X$ so we can find the labels of w_1 and w_2 (those are repeated twice, and are symmetric) and of x (it appears an odd number of times). Finding the label of u is then trivial.
14. The dart (a cycle vw_1xw_2 with an edge vx and a leaf u attached to v) can be reconstructed. We have $S_2(\mathcal{G}) + 2S_5(\mathcal{G}) - S_3(\mathcal{G}) = X$, after which $S_3(\mathcal{G}) - 2S_2(\mathcal{G}) = U - 2V - 2X$ so we can find the labels of u and v . Finding the labels of w_1 and w_2 is trivial (both are false twins and we do not need to separate them).
15. The house (a cycle $v_1w_1w_2v_2$ with a vertex u adjacent to both v_1 and v_2). We have $S_2(\mathcal{G}) - 2S_5(\mathcal{G}) = V_1 + V_2$ and $2S_2(\mathcal{G}) - S_3(\mathcal{G}) = U + V_1 + V_2$ so we know those three labels. All that is left is to decide which of the two remaining labels are W_1 and W_2 , which we can deduce using $\mathcal{M}_2(\mathcal{G})$ (the only possible confusion would be if $W_1 = V_2$ and $W_2 = V_1$, which is not a problem since the two possible labelings would be isomorphic).
16. The banner (a cycle vw_1xw_2 with a leaf u attached to v) can be reconstructed. We have $S_4(\mathcal{G}) - 3S_5(\mathcal{G}) = V$, after which $S_2(\mathcal{G}) - 2S_5(\mathcal{G}) = V - U$ so we know the label of u (whether $U = V$ or not). Furthermore, we have $2S_2(\mathcal{G}) - S_3(\mathcal{G}) = V + X$ from which we know the label of x , and we can find the labels of false twins w_1 and w_2 .
17. The cricket (a triangle vw_1w_2 with two leaves u_1 and u_2 attached to v) can be reconstructed. We have $S_2(\mathcal{G}) + 2S_5(\mathcal{G}) - S_3(\mathcal{G}) = W_1 + W_2$ (both w_1 and w_2 are twins, so we know their labels), after which $3S_2(\mathcal{G}) - S_3(\mathcal{G}) = 6V + 3(W_1 + W_2)$ so we can find the label of v . Finally, finding the labels of u_1 and u_2 is trivial (both are false twins).
18. The bull (a triangle v_1v_2w with two leaves u_1 and u_2 attached to v_1 and v_2 , respectively) can be reconstructed. We have $2S_2(\mathcal{G}) - S_3(\mathcal{G}) = 2(V_1 + V_2) + W$, so we know the label of w and the composition of labels of v_1 and v_2 , then the query $\mathcal{M}_2(\mathcal{G})$ will allow us to find the labels of u_1 and u_2 (the only possible confusion is if $U_1 = V_2$ and $U_2 = V_1$, which is not a problem since the two possible labelings would be isomorphic).
19. The complete bipartite graph $K_{2,3}$ can trivially be reconstructed from $S_2(\mathcal{G})$ (since all vertices in each part are equivalent to each other; the two parts can be separated and thus we are done).

Hence, the only confusable graphs of order 5 are the gem depicted on Fig. 5 and the graph depicted on Fig. 4, which we will call T_2 .

Smallest confusable trees. Trees of order at most 5 are not confusable per above. Trees of order 6 are $P_6, S_5, S_{1,1,3}$, the bistar $B_{2,2}$ (none of which are confusable, see [1], Corollary 14 and Theorems 20 and 21, respectively), and $S_{1,2,2}$ and $S_{1,1,1,2}$. For $S_{1,2,2}$, the proof of Theorem 16 works with any alphabet: the only possible confusion at the end of the proof (and identified in Remark 18) is if the two branches of length 2 are palindromes of each other, in which case the confusion does not matter due to isomorphisms. As for $S_{1,1,1,2}$, the proof of Theorem 16 clearly works too: once the label of the center vertex has been found, this gives the label of the only degree 2 vertex, after which we can remove pairs of leaves, hence having the composition of the branch of length 2, from which we can know the label of its leaf, giving the labels of the other three symmetrical leaves.

Trees of order 7 require more work. Some of them we can easily manage: P_7 [1], the star S_6 (Corollary 14), $S_{1,1,4}$ (Theorem 21) and the bistar $B_{2,3}$ (Theorem 20). The subdivided star $S_{1,2,3}$ is confusable, as can be seen in Fig. 6 (this is a result from [6], where they do not explicitly give the example, but Péter Burcsi and Zsuzsanna Lipták communicated the graph to us); this can be checked by enumerating the multiset compositions of both labelings. Since there are 11 trees with 7 vertices (see [23]), there are six remaining trees of order 7 to analyze. Note that in the following considerations, we obtain the composition of inner vertices I with Lemma 12 and in the cases of subdivided stars (that is, in Cases 1, 2, 3 and 4) Lemma 15 gives the label of the center vertex:

1. $S_{2,2,2}$. For this graph, the proof of Theorem 16 works, except in the case outlined in Remark 18: the branches of length 2 form a “cycle” (such as having compositions AB , BC and CA) and we can obtain the label X of the center vertex. However, in this case, the query $\mathcal{M}_4(\mathcal{S})$ will allow us to distinguish between the two possible labelings (if composition A^2BX is appearing, then the branch with composition AB has leaf A and degree 2 vertex B , allowing us to find the rest; otherwise, it is the reverse).
2. $S_{1,1,2,2}$. For this graph, the proof of Theorem 16 works: the only possible confusion at the end of the proof is if the two branches of length 2 are palindromes of each other, in which case the confusion does not matter since it leads to isomorphic labelings.
3. $S_{1,1,1,3}$. Denote the center by c , and the vertices of the branch of length 3 by v_1 , v_2 and v_3 where v_1 is adjacent to c and v_3 is a leaf. We have $S_5(\mathcal{S}) - (S_3(\mathcal{S}) + S_2(\mathcal{S})) = -4C + 2V_2 + V_3$. Since we know C , this allows us to deduce V_2 and V_3 , from which we can learn V_1 (the last internal vertex) and the three leaves adjacent to c .
4. $S_{1,1,1,1,2}$. For this graph, there are two inner vertices. As one of them is the center vertex, we can obtain labels of both of them. Hence, we only need to identify the label of the leaf at distance two from the center. We can do that by removing the pairs of leaves from compositions obtained with $\bigsqcup_{X \in \mathcal{M}_5(\mathcal{S})} (S_7(\mathcal{S}) - X)$, leaving only the pair with this leaf and the degree 2 vertex, whose label is known.
5. The labeled tree \mathcal{B} obtained by subdividing the middle edge of the bistar $B_{2,2}$. We can find the compositions L of leaves and I of internal vertices by Lemma 12, from which we can compute the label of the middle vertex with $L + 3I - S_2(\mathcal{B})$. Assume that the middle vertex has label A ; we will check the largest i such that the query $\mathcal{M}_i(\mathcal{B})$ contains the composition A^i , which will allow us to reconstruct the labeling. If $i = 7$, then, each label is A and we are done. If $i = 6$, then, every vertex but one leaf is labeled with A , so we are done. If $i = 5$, then, two leaves are not labeled with A , so we can check whether there is an AXY (for $X, Y \neq A$) in $\mathcal{M}_3(\mathcal{B})$; if true, then the two non- A leaves are adjacent to the same degree 3 vertex, otherwise they are adjacent to different degree 3 vertices, and in both cases we are done. If $i = 4$, then, either a degree 3 vertex is labeled with $X \neq A$, or exactly three leaves are labeled with $X, Y, Z \neq A$. In the first case the two leaves adjacent to the degree 3 vertex labeled with A are also labeled with A , and we know all other labels, so we are done. In the second case only one of AXY , AXZ and AYZ will be in $\mathcal{M}_3(\mathcal{B})$, allowing us to know which leaf is at distance 2 of the leaf labeled with A . In both cases we can reconstruct the labeling. If $i = 3$, then, either exactly one degree 3 vertex has label $X \neq A$ and a leaf adjacent to the degree 3 vertex with label A has itself label A and the other leaf has label $W \neq A$, or we have $I = 3A$. In the first case we know X , and we can use query $\mathcal{M}_3(\mathcal{B})$ to find all compositions A^2Y with $Y \neq A$: there can be from one to five A^2X depending on if zero, one or two leaves adjacent to the X -labeled degree 3 vertex are labeled with A and whether or not $W = X$; if there are one or two A^2X , then $W \neq X$ and we can find W from the only composition AY other than AX in $\mathcal{M}_2(\mathcal{B})$ after which we get the last two leaves; if there are three A^2X , then we have either $W = X$ and none of the last two leaves with label A or $W \neq X$ and the last two leaves have label A , we can distinguish between them by counting the number of AX compositions in $\mathcal{M}_2(\mathcal{B})$, which is two in the former case and three in the latter case, in both cases we can then easily conclude; if there are four A^2X , then we have $W = X$ and one of the last leaves has label A , we conclude immediately with the last leaf; if there are five A^2X , then $W = X$ and the last two leaves have label A . In the second case with $I = 3A$, all leaves have labels different from A , say W, X, Y and Z . In this case the query $\mathcal{M}_3(\mathcal{B})$ will allow us to check possible combinations among AWX , AWY , AWZ , AXY , AXZ and AYZ ; note that only two of those can exist unless there is some isomorphism, in both cases we can reconstruct the graph. If $i = 2$, then, one of the degree 3 vertices has label A and its two adjacent leaves have labels $X, Y \neq A$ and the other degree 3 vertex has label Z (which we know). If a leaf is labeled with A , then it is adjacent to the Z -labeled degree 3 vertex. We can use query $\mathcal{M}_2(\mathcal{B})$ to get all AX and AY combinations, allowing us to complete the reconstruction. Finally, if $i = 1$, then the two degree 3 vertices are labeled with $U, V \neq A$. Let W, X, Y and Z be the labels of the leaves. If $U = V$, then, we use query $\mathcal{M}_3(\mathcal{B})$. First, we count the number of A^2U compositions which there are at most six; if there are six A^2U , then all leaves have label A ; there cannot be exactly five A^2U ; if there are four A^2U , then three leaves have label A and we are done; if there are three A^2U , then two leaves attached to the same degree 3 vertex have label A ; if there are two A^2U , then two leaves attached to different degree 3 vertices have label A ; if there is one A^2U , then only one leaf has label A and we may query $\mathcal{M}_3(\mathcal{B})$ to find WZU where $W, Z \neq A$ to find how labels are distributed between the leaves. In all of cases, completing the reconstruction afterwards is easy. If no A^2U composition can be found, we remove all compositions containing A and are left with the following possible compositions: UWX , UWY , UWZ , UXY , UXZ and UYZ ; only two of these exist unless there is some isomorphism, in both cases we can reconstruct the graph. If $U \neq V$, then, leaves labeled with A can easily be found with compositions

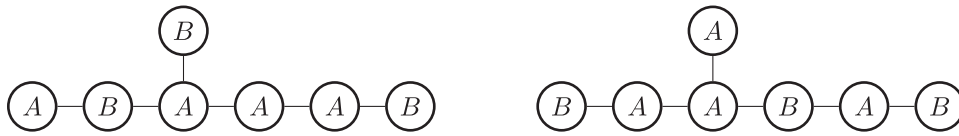


Fig. 6. Two equicomposable labelings of $S_{1,2,3}$.

AU and AV in $\mathcal{M}_2(\mathcal{B})$ (there is at least one of each using the middle vertex, any supplementary one gives us a leaf labeled with A), after which we use query $\mathcal{M}_3(\mathcal{B})$ to check for $AUW, AVW, \dots, AUZ, AVZ$, and again we can find which leaves are attached to each degree 3 vertex.

- The tree obtained by subdividing an edge incident with a leaf in the bistar $B_{2,2}$. Denote the vertices as following: x is the degree 2 vertex, y its adjacent leaf and w its adjacent degree 3 vertex, z is the leaf adjacent to w , v is the other degree 3 vertex, and u_1 and u_2 are the two leaves adjacent to v . Recall that we know L , the composition of leaves $(u_1, u_2, y$ and $z)$, and I , the composition of internal vertices $(v, w$ and $x)$, thanks to Lemma 12. We have $S_4(\mathcal{B}) - S_3(\mathcal{B}) - L - 2I = Z$. Furthermore, we have $S_5(\mathcal{B}) - S_3(\mathcal{B}) - 2L - I = V + 2X$, so we can deduce those two labels, which also gives the last internal label W . Finally, $S_3(\mathcal{B}) - 2L = 5V + 6W + 3X - Y$ allows us to know the value of Y , from which we finally deduce the labels of the last two leaves u_1 and u_2 , completing the reconstruction.

Hence, $S_{1,2,3}$ is the only confusable tree of order at most 7. \square

Interestingly, one of the smallest confusable graph (illustrated in Fig. 4) can be generalized into an infinite family of confusable graphs. We denote by T_m the graph constructed by attaching a path of length m to one vertex of a triangle (so the smallest confusable graph described above is T_2). We have the following:

Theorem 23. *Let m be a positive integer and $\mathcal{T} = (T_m, \lambda)$. If m is odd, then \mathcal{T} is sum-reconstructible. If m is even, then T_m is confusable.*

Proof. Let m be a positive integer and $\mathcal{T} = (T_m, \lambda)$. Denote the two degree 2 vertices in the triangle by u_1 and u_2 (note that they are twins, so if we find all the other labels then we can easily finish the reconstruction), the degree 3 vertex by v_1 , and the vertices of the path by v_2, \dots, v_{m+1} starting from the vertex attached to the triangle. Since T_m has $m + 3$ vertices, we have already shown above that the 4-vertex graph with $m = 1$ is not confusable.

Case 1: m is odd, so $m = 2p - 1$ for $p \geq 2$. We note that, for $j \neq 2$, we have $S_j(\mathcal{T}) = S_j(\mathcal{S})$ where $S_j(\mathcal{S})$ is as in Eqs. (3) and (4) when \mathcal{S} and \mathcal{T} have an equal number of vertices. We further note that $S_2(\mathcal{T}) \neq S_2(\mathcal{S})$. Since the proof of Lemma 15 used $S_2(\mathcal{T})$, we cannot obtain V_1 as in Theorem 21. First, if $p = 2$, then we can reconstruct \mathcal{T} in the following way: We find V_1 and V_2 with $S_3(\mathcal{T}) - S_2(\mathcal{T}) = V_1 + 2V_2$; we find V_3 with $S_4(\mathcal{T}) - S_3(\mathcal{T}) = V_3$; label V_4 with $S_5(\mathcal{T}) - S_4(\mathcal{T}) = V_4 - (V_1 + V_2)$; finally, we conclude with U_1 and U_2 as explained above.

Now, consider $p \geq 3$. We apply the same reasoning as in Case 1 of the proof of Theorem 21. Since $S_i(\mathcal{T})$ is the same for every $i \neq 2$, Eqs. (3) and (4) still hold. The two differences are that we do not know V_1 and $S_2(\mathcal{T}) \neq S_2(\mathcal{S})$. Hence, we can reuse Claims 21.1 and 21.2 to alternately compute V_{p+i} (using $S_{p+i+1}(\mathcal{T}) - S_{p+i}(\mathcal{T})$ for increasing values of $i \in \{1, \dots, p-1\}$), and labels V_{i-1} and V_i (using $S_{i+1}(\mathcal{T}) - S_i(\mathcal{T})$ for decreasing values of i for $i \in \{p, \dots, 3\}$). We refer the reader to the aforementioned proof for details; note that the two claims do not use V_1 so we can still apply the same reasoning. Hence, after this, we know the labels of v_2, \dots, v_{2p-1} . Now, by Lemma 4, $S_2(\mathcal{T}) = 3V_1 + V_{2p} + 2(U_1 + U_2 + \sum_{i=2}^{2p-1} (V_i))$, so we have $S_3(\mathcal{T}) - S_2(\mathcal{T}) = V_1 + 2V_2 + \sum_{i=3}^{2p-2} (V_i) = V_1 + Q$ where Q is a known quantity, so we find the label of v_1 . Finally, as in the aforementioned proof, $S_{2p+1}(\mathcal{T}) - S_{2p}(\mathcal{T}) = V_{2p} - \sum_{i=1}^{2p-2} (V_i)$, hence we obtain the label of v_{2p} . Finally, we conclude with the last two labels for u_1 and u_2 .

Case 2: m is even, so $m = 2p$ for $p \geq 1$. We prove that T_m is confusable by exhibiting two labelings λ_1 and λ_2 such that $\mathcal{M}((T_m, \lambda_1)) = \mathcal{M}((T_m, \lambda_2))$. The two binary labelings, illustrated in Fig. 7, are defined as follows:

- $\lambda_1(v_{2i+1}) = A$ for $i \in \{0, \dots, p\}$ and $\lambda_1(u_1) = \lambda_1(u_2) = \lambda_1(v_{2i}) = B$ for $i \in \{1, \dots, p\}$ (Fig. 7(a));
- $\lambda_2(u_1) = \lambda_2(v_{2i}) = A$ for $i \in \{1, \dots, p\}$ and $\lambda_2(u_2) = \lambda_2(v_{2i+1}) = B$ for $i \in \{0, \dots, p\}$ (Fig. 7(b)).

We next prove that $\mathcal{M}((T_m, \lambda_1)) = \mathcal{M}((T_m, \lambda_2))$. First, note that both graphs have the same numbers of symbols A and B . Furthermore, it is easy to see that, for both labeled graphs, all subgraphs of order 2 have set $\{A, B\}$ except one which has set $\{B^2\}$ (using Lemma 4).

Consider now the subgraphs of order $2i$ for $i \in \{2, \dots, p+1\}$. For (T_m, λ_1) , one subgraph has composition $A^{i-1}B^{i+1}$ (the one containing the triangle), and all other subgraphs have composition A^iB^i since they are all paths of vertices labeled alternately with symbols A and B . The same holds for (T_m, λ_2) , where the subgraph with composition $A^{i-1}B^{i+1}$ is the one containing vertices $u_2, v_1, \dots, v_{2i-1}$, and all other subgraphs have composition A^iB^i .

Finally, consider the subgraphs of order $2i + 1$ for $i \in \{1, \dots, p\}$. For (T_m, λ_1) , there are $p - i + 3$ subgraphs with composition A^iB^{i+1} (the $p - i$ subpaths starting from v_{2j} and ending at v_{2j+2i} for $j \in \{1, \dots, p - i\}$), as well as the three subgraphs containing u_1 and/or u_2) and $p - i + 1$ subgraphs with composition $A^{i+1}B^i$ (the subpaths starting from v_{2j+1}

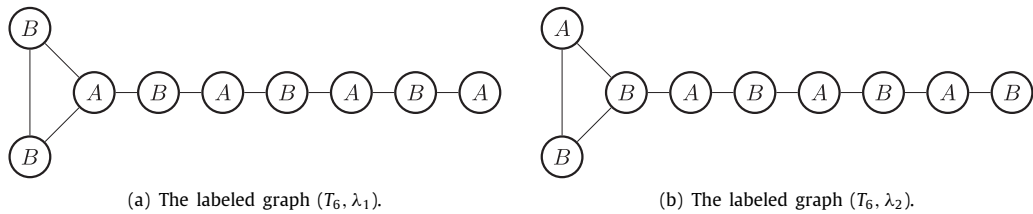


Fig. 7. Example of two equicomposable labelings for T_m with even m .

and ending at $v_{2j+1+2i}$ for $j \in \{0, \dots, p - i\}$). The same holds for (T_m, λ_2) : there are $p - i + 3$ subgraphs with composition $A^i B^{i+1}$ (the $p - i + 1$ subpaths starting from v_{2j+1} and ending at $v_{2j+1+2i}$ for $j \in \{0, \dots, p - i\}$, as well as the two subgraphs containing u_2) and $p - i + 1$ subgraphs with composition $A^{i+1} B^i$ (the $p - i$ subpaths starting from v_{2j} and ending at v_{2j+2i} for $j \in \{1, \dots, p - i\}$, as well as the subgraph containing u_1 and not u_2).

Hence, we have $\mathcal{M}((T_m, \lambda_1)) = \mathcal{M}((T_m, \lambda_2))$, and as such T_m is confusable. \square

Compare Theorem 23 with Theorem 21: adding one edge (alternately, creating a twin instead of a false twin) made half of the graph class confusable. This is due to the fact that, in a subdivided star, finding the label of the center vertex is easy. For T_m (as with $S_{1,1,m}$), we are starting the reconstruction from the middle of the path. In the odd case, this is possible since there are three middle labels V_{p-1}, V_p and V_{p+1} which we can reconstruct. However, in the even case, there might be some confusion with the labels of the middle vertices (even- and odd-indexed vertices alternating labels as in Fig. 7); but in the case of the subdivided star, this is solved by reaching an imbalance between even- and odd-indexed vertices by knowing the label of the center of the star, while T_m does not allow us to decide between the two possible labelings.

Finally, we show that while the smallest confusable trees have seven vertices, the path P_4 on four vertices is not sum-reconstructible.

Theorem 24. *The smallest non-sum-reconstructible graphs and trees have four vertices.*

Proof. There are two three vertex graphs, path P_3 and cycle C_3 . For P_3 , we may obtain the center vertex with sum-compositions of size 2 which is enough to reconstruct P_3 . Cycle C_3 is trivially reconstructible using sum-compositions of size 1.

Let us next consider path P_4 where vertices are labeled as V_1, V_2, V_3, V_4 from left to right. Let us denote by S_i the sum-composition of size i . We have $S_1 = S_4 = V_1 + V_2 + V_3 + V_4, S_2 = S_3 = V_1 + 2V_2 + 2V_3 + V_4$. Hence, we cannot distinguish for example between $V_1 = A, V_2 = A, V_3 = B, V_4 = B$ and $V_1 = A, V_2 = B, V_3 = A, V_4 = B$. \square

5. Enumerating confusable graphs

In [1], Acharya et al. have presented a way to construct equicomposable pairs of paths of length $n \geq 8$ where $n = m \cdot m' - 1$ for integers $m, m' \geq 3$. We use their construction to give more complex examples of equicomposable graphs. Consider two labeled paths \mathcal{P}_1 and \mathcal{P}_2 where vertices of \mathcal{P}_2 are denoted from left to right by p_1, p_2, \dots, p_m and they are labeled as $\lambda_2(p_i) = A_i$ for each i . We call the interleaving of a path \mathcal{P}_1 by bits of \mathcal{P}_2 , denoted by $\mathcal{P}_1 \circ \mathcal{P}_2$, a path where we start with a copy of path \mathcal{P}_1 , then continue with the first label of \mathcal{P}_2 which is then followed by another copy of \mathcal{P}_1 and the second label of \mathcal{P}_2 , and so on. In this way, we construct the labeled path with the following structure: $\mathcal{P}_1 A_1 \mathcal{P}_1 A_2 \mathcal{P}_1 \dots \mathcal{P}_1 A_m \mathcal{P}_1$.

We further denote by \mathcal{P}_2^* the reversal of path \mathcal{P}_2 , that is, the path labeled from left to right by A_m, A_{m-1}, \dots, A_1 . So, if $\lambda_2(p_i) = A_i$, then we denote $\lambda_2^*(p_i) = A_{m-i+1}$. Recall that we denote $\mathcal{G} \sim \mathcal{H}$ if \mathcal{G} and \mathcal{H} are equicomposable.

Lemma 25 ([1]). *For any labeled paths \mathcal{P}_1 and \mathcal{P}_2 , we have $\mathcal{P}_1 \circ \mathcal{P}_2 \sim \mathcal{P}_1 \circ \mathcal{P}_2^*$.*

In particular, Acharya et al. [1] proved that there exists a bijection g pairing the subgraphs of $\mathcal{P}_1 \circ \mathcal{P}_2$ and $\mathcal{P}_1 \circ \mathcal{P}_2^*$ which have identical composition multisets. Furthermore, when \mathcal{P}_2 contains an odd number of vertices $m = 2t - 1$, we can construct the bijection g so that any subgraph of $\mathcal{P}_1 \circ \mathcal{P}_2$ containing the center vertex p_t of \mathcal{P}_2 is paired with a subgraph $\mathcal{P}_1 \circ \mathcal{P}_2^*$ also containing the center vertex p_t . We call this bijection g the path bijection between $\mathcal{P}_1 \circ \mathcal{P}_2$ and $\mathcal{P}_1 \circ \mathcal{P}_2^*$.

Theorem 26. *Let \mathcal{G} be a labeled graph, \mathcal{P}_1 be a labeled path on at least two vertices, and \mathcal{P}_2 be a labeled path on $2t - 1$ vertices $p_1, p_2, \dots, p_{2t-1}$. Let \mathcal{G}_1 be constructed by joining \mathcal{G} with a single edge to vertex p_t in $\mathcal{P}_1 \circ \mathcal{P}_2$ and \mathcal{G}_2 be constructed by joining \mathcal{G} with a single edge to vertex p_t in $\mathcal{P}_1 \circ \mathcal{P}_2^*$. We have*

$$\mathcal{G}_1 \sim \mathcal{G}_2.$$

Proof. We construct a bijection f from the set of labeled induced subgraphs of \mathcal{G}_1 to the set of labeled induced subgraphs of \mathcal{G}_2 so that for any induced subgraph \mathcal{G}' of \mathcal{G}_1 we have $c(\mathcal{G}') = c(f(\mathcal{G}'))$. Let us denote by x the vertex of \mathcal{G} adjacent to the center vertex p_t of \mathcal{P}_2 .

When subgraph \mathcal{G}' contains only vertices from \mathcal{G} , we let $f(\mathcal{G}') = \mathcal{G}'$. When \mathcal{G}' contains vertices only from $\mathcal{P}_1 \circ \mathcal{P}_2$, we have $f(\mathcal{G}') = g(\mathcal{G}')$ where g is the path bijection between $\mathcal{P}_1 \circ \mathcal{P}_2$ and $\mathcal{P}_1 \circ \mathcal{P}_2^*$.

Consider finally the case where \mathcal{G}' contains vertices from both \mathcal{G} and $\mathcal{P}_1 \circ \mathcal{P}_2$. We partition the subgraph \mathcal{G}' into two parts \mathcal{G}'_g and \mathcal{G}'_p where \mathcal{G}'_g is a subgraph of \mathcal{G} and \mathcal{G}'_p is a subgraph of $\mathcal{P}_1 \circ \mathcal{P}_2$. We construct $f(\mathcal{G}')$ by separately having $f(\mathcal{G}'_g) = \mathcal{G}'_g$ and $f(\mathcal{G}'_p) = g(\mathcal{G}'_p)$ and after this, by adding an edge between $f(\mathcal{G}'_g)$ and $f(\mathcal{G}'_p)$ from p_t to x (note that this is always possible since, by construction, $x \in f(\mathcal{G}'_g)$ and $p_t \in f(\mathcal{G}'_p)$).

It is clear that this construction guarantees $c(f(\mathcal{G}')) = c(\mathcal{G}')$. Since this holds for any connected induced subgraph \mathcal{G}' of \mathcal{G}_1 , and furthermore $f(\mathcal{G}')$ is a connected induced subgraph of \mathcal{G}_2 , we have $\mathcal{M}(\mathcal{G}_1) = \mathcal{M}(\mathcal{G}_2)$. \square

In the following theorem we show that while some subdivided stars are reconstructible, it is also possible find subdivided stars which contain a large number of equicomposable labelings.

Theorem 27. For $k \geq 2$ and $m \geq 3$, there exist subdivided stars on $n = (2m + 1)(k^m - k^{\lceil m/2 \rceil}) + 1$ vertices which have $2^{\Theta(n/\log n)}$ equicomposable labeled graphs.

Proof. Let P^m be the set of non-isomorphic and non-palindromic labeled paths of length $m \geq 2$ over an alphabet of size k . We note that by Theorem 6, there are $\chi_k(P_m) = \frac{k^m + k^{\lceil m/2 \rceil}}{2}$ non-isomorphic labelings. Furthermore, as we have seen in the proof of Theorem 6, there are $k^{\lceil m/2 \rceil}$ palindromes of length m . Hence, we have $|P^m| = \frac{k^m - k^{\lceil m/2 \rceil}}{2}$. Let \mathcal{P} be a labeled path on three vertices labeled as A, A and B from left to right. For each $\mathcal{P}_m \in P^m$, we construct interleaved path $\mathcal{P}i = \mathcal{P}_m \circ \mathcal{P}$. After this, we identify the center vertex of each interleaved path $\mathcal{P}i$ leading to a subdivided star \mathcal{S} . Note that $\mathcal{P}i$ has $4m + 3$ vertices and hence \mathcal{S} has $(4m + 2)|P^m| + 1 = n$ vertices.

Observe that, by Theorem 26, \mathcal{S} is equicomposable with any subdivided star in which some $\mathcal{P}i = \mathcal{P}_m \circ \mathcal{P}$ is replaced by $\mathcal{P}_m \circ \mathcal{P}^*$. Note that since paths in P^m are chosen to be non-isomorphic and non-palindromic, the obtained subdivided star is non-isomorphic with \mathcal{S} . Furthermore, we may replace any number of $|P^m|$ paths in this way. In other words, we have $|P^m|$ binary choices and thus, there is a class of equicomposable graphs containing $2^{|P^m|}$ labeled graphs. Since $n = (4m + 2)|P^m| + 1$, we have $2^{|P^m|} \in 2^{\Theta(n/\log n)}$. \square

6. Conclusion

In this article, we have expanded the problem of reconstructing vertex-labeled graphs from their composition multisets from paths in [1] to general graphs (which has been briefly considered in [6]).

In particular, we have given multiple reconstructible graph classes and shown reconstruction algorithms using as few queries as possible. On the negative side, we also found some non-reconstructible graph classes, and studied smallest confusable graphs. While our focus has been on trees (and particularly on subdivided stars), we also studied other classes such as paths where one leaf has been twinned.

In general, the problem of deciding whether a graph class is reconstructible seems hard, even when we want the graph classes to be rather simple for possible applications in polymer-based information storage systems. From this perspective, the graph class of subdivided stars $S_{1,1,k}$ seems quite enticing, it is at the same time simple to construct, easy to reconstruct due to sum-reconstructibility (by Theorem 21) and allows for a larger number of labelings than simple paths (by Theorems 6 and 8). As an interesting open question, we ask: is any other subdivided star $S_{1,k,\ell}$ with $k > 1$ (sum-)reconstructible when ℓ can be large? We know that the answer is negative in some cases when $k = \ell$ (by Theorem 26), so it might be worth to restrict the value of k compared to ℓ . Another possible direction for future research would be finding some conditions for graphs to be confusable. We also refer the reader back to Section 3.4 for discussion specifically centered on reconstructible graph classes.

Acknowledgments

This paper is dedicated to the 60th birthday of Ralf Klasing. Antoine Dailly met Ralf in 2023, when he visited Clermont-Ferrand for one week; he was the one who introduced him to the topic of temporal graphs, which are now a subject of predilection for him. Tuomo Lehtilä worked with Ralf on the topic of Red-Blue separating sets in 2021 and 2022.

Ralf is a very nice and accessible person, as well as a great researcher, with a deep knowledge of graphs and algorithms. We are very pleased to have worked with him, and hope that more opportunities will arise in the future.

We thank Dénes Bartha, Péter Burcsi, and Zsuzsanna Lipták for personally communicating us the seven vertex confusable tree. We also thank the anonymous referees for their useful remarks and suggestions.

Data availability

No data was used for the research described in the article.

References

- [1] Jayadev Acharya, Hirakendu Das, Olgica Milenkovic, Alon Orlitsky, Shengjun Pan, String reconstruction from substring compositions, *SIAM J. Discrete Math.* 29 (3) (2015) 1340–1371.
- [2] Abdelaziz Al Ouahabi, Jean-Arthur Amalian, Laurence Charles, Jean-François Lutz, Mass spectrometry sequencing of long digital polymers facilitated by programmed inter-byte fragmentation, *Nat. Commun.* 8 (1) (2017) 967.
- [3] Kevin J. Asciak, M.A. Francalanza, Josef Lauri, W. Myrvold, A survey of some open questions in reconstruction numbers, *Ars Comb.* 97 (2010) 443–456.
- [4] Anisha Banerjee, Antonia Wachter-Zeh, Eitan Yaakobi, Insertion and deletion correction in polymer-based data storage, *IEEE Trans. Inform. Theory* 69 (7) (2023) 4384–4406.
- [5] Dénes Bartha, Reconstructibility of Tree Graphs (Ph.D. thesis), Eötvös Loránd University, 2021.
- [6] Dénes Bartha, Péter Burcsi, Zsuzsanna Lipták, Reconstruction of trees from jumbled and weighted subtrees, in: 27th Annual Symposium on Combinatorial Pattern Matching, CPM 2016, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.
- [7] Paul Brown, Trevor Fenner, A new approach to graph reconstruction using supercards, *J. Comb.* 9 (1) (2018) 95–118.
- [8] Antoine Dailly, Tuomo Lehtilä, Reconstructing graphs from subgraph compositions, in: 2025 IEEE International Symposium on Information Theory, ISIT, 2025, pp. 1–6.
- [9] H.N. de Ridder, et al., Information system on graph classes and their inclusions. List of small graphs, 2001–2024, <https://graphclasses.org/smallgraphs.html>. (Accessed 25 February 2025).
- [10] Ryan Gabrys, Srilakshmi Pattabiraman, Olgica Milenkovic, Mass error-correction codes for polymer-based data storage, in: 2020 IEEE International Symposium on Information Theory, ISIT, IEEE, 2020, pp. 25–30.
- [11] Ryan Gabrys, Srilakshmi Pattabiraman, Olgica Milenkovic, Reconstruction of sets of strings from prefix/suffix compositions, *IEEE Trans. Commun.* 71 (1) (2022) 3–12.
- [12] Frank Harary, A survey of the reconstruction conjecture, in: *Graphs and Combinatorics: Proceedings of the Capital Conference on Graph Theory and Combinatorics At the George Washington University June 18–22, 1973*, Springer, 1974, pp. 18–28.
- [13] Ville Junnila, Tero Laihonen, Tuomo Lehtilä, On Levenshtein's channel and list size in information retrieval, *IEEE Trans. Inform. Theory* 67 (6) (2021) 3322–3341.
- [14] Ville Junnila, Tero Laihonen, Tuomo Lehtilä, The Levenshtein's sequence reconstruction problem and the length of the list, *IEEE Trans. Inform. Theory* 70 (2) (2024) 1050–1066.
- [15] Paul Kelly, A congruence theorem for trees, *Pacific J. Math.* 7 (1) (1957) 961–968.
- [16] Alexandr V. Kostochka, Douglas B. West, On reconstruction of graphs from the multiset of subgraphs obtained by deleting ℓ vertices, *IEEE Trans. Inform. Theory* 67 (6) (2020) 3278–3286.
- [17] Kévin Launay, Jean-arthur Amalian, Eline Laurent, Laurence Oswald, Abdelaziz Al Ouahabi, Alexandre Burel, Florent Dufour, Christine Carapito, Jean-louis Clément, Jean-François Lutz, et al., Precise alkoxyamine design to enable automated tandem mass spectrometry sequencing of digital poly(phosphodiester)s, *Angew. Chem.* 133 (2) (2021) 930–939.
- [18] Vladimir I. Levenshtein, Efficient reconstruction of sequences, *IEEE Trans. Inform. Theory* 47 (1) (2001a) 2–22.
- [19] Vladimir I. Levenshtein, Efficient reconstruction of sequences from their subsequences or supersequences, *J. Combin. Theory Ser. A* 93 (2) (2001b) 310–332.
- [20] Srilakshmi Pattabiraman, Ryan Gabrys, Olgica Milenkovic, Reconstruction and error-correction codes for polymer-based data storage, in: 2019 IEEE Information Theory Workshop, ITW, IEEE, 2019, pp. 1–5.
- [21] Srilakshmi Pattabiraman, Ryan Gabrys, Olgica Milenkovic, Coding for polymer-based data storage, *IEEE Trans. Inform. Theory* 69 (8) (2023) 4812–4836.
- [22] Van Long Phuoc Pham, Keshav Goyal, Han Mao Kiah, Sequence reconstruction problem for deletion channels: A complete asymptotic solution, *J. Combin. Theory Ser. A* 211 (2025) 105980.
- [23] N.J.A. Sloane, et al., OEIS. Number of trees, 2001–2024, <https://oeis.org/A000055>. (Accessed 3 March 2025).
- [24] Hannah Spinoza, Douglas B. West, Reconstruction from the deck of k -vertex induced subgraphs, *J. Graph Theory* 90 (4) (2019) 497–522.
- [25] Richard P. Stanley, *Enumerative combinatorics volume 1* second edition, Cambridge University Press, 2011.
- [26] Stanislaw M. Ulam, *A collection of mathematical problems*, 1960.
- [27] Eitan Yaakobi, Jehoshua Bruck, On the uncertainty of information retrieval in associative memories, in: *Proceedings of 2012 IEEE International Symposium on Information Theory, ISIT, 2012*, pp. 106–110.
- [28] Yaoyu Yang, Zitan Chen, Reconstruction of multiple strings of constant weight from prefix-suffix compositions, *Entropy* 27 (1) (2025) 39.
- [29] Zuo Ye, Ohad Elishco, Reconstruction of a single string from a part of its composition multiset, *IEEE Trans. Inform. Theory* 70 (6) (2023) 3922–3940.