Reconstruction of Trees from Jumbled and Weighted Subtrees

Dénes Bartha[1], Péter Burcsi[2], and Zsuzsanna Lipták[3]

1 Dept. of Computer Algebra, Eötvös Loránd University, Budapest, Hungary
denesb@gmail.com

2 Dept. of Computer Algebra, Eötvös Loránd University, Budapest, Hungary
bupe@compalg.inf.elte.hu

3 Dip. di Informatica, University of Verona, Italy
zsuzsanna.liptak@univr.it

Abstract

Let T be an edge-labeled graph, where the labels are from a finite alphabet Σ. For a subtree U of T, the *Parikh vector* of U is a vector of length $|\Sigma|$ which specifies the multiplicity of each label in U. We ask when T can be reconstructed from the multiset of Parikh vectors of all of its subtrees, or all of its paths, or all of its maximal paths. We consider the analogous problems for weighted trees. We show how several well-known reconstruction problems on labeled strings, weighted strings and point sets on a line can be included in this framework. We present reconstruction algorithms and non-reconstructibility results, and extend the polynomial method, previously applied to jumbled strings [Acharya *et al*, SIAM J on Discr. Math, 2015] and weighted strings [Bansal *et al*, CPM 2004], to deal with general trees and special tree classes.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Computations on discrete structures, G.2.2 [Graph Theory] Graph labeling, Trees

Keywords and phrases trees, paths, Parikh vectors, reconstruction problems, homometric sets, polynomial method, jumbled strings, weighted strings

Digital Object Identifier 10.4230/LIPIcs.CPM.2016.10

1 Introduction

Let T be an unrooted tree T with labeled edges, where the labels come from a finite ordered alphabet Σ. For a subtree U of T, the *Parikh vector* of U is a vector of length $|\Sigma|$ which specifies the multiplicity of each label in U. If the labels are positive reals or integers, we refer to them as *weights*, and define the weight of U as the sum of weights of the edges in U. (It is common to refer to a subtree as *jumbled* if only its Parikh vector is known, and as *weighted* if only its weight is known.) Given a subtree property $\mathcal{A}$, we refer to the multiset of Parikh vectors of all subtrees with property $\mathcal{A}$ as $MP_{\mathcal{A}}(T)$, and to the multiset of weights of all subtrees with property $\mathcal{A}$ as $MW_{\mathcal{A}}(T)$. For example, $MW_{\text{PATH}}(T)$ is the multiset of path weights in a weighted tree T.

Consider the two edge-labeled trees in Fig. 1, with labels from the alphabet $\Sigma = \{a, b\}$. The two trees are non-isomorphic, but the multisets of Parikh vectors of their *subtrees* are the same, $MP_{\text{SUBTREE}}(T_1) = MP_{\text{SUBTREE}}(T_2)$, as can be easily checked. At the same time, the multisets of Parikh vectors of their *paths* are not the same, $MP_{\text{PATH}}(T_1) \neq MP_{\text{PATH}}(T_2)$, since, for instance, T_2 has a path with Parikh vector $(1, 3)$ and T_1 does not.

These multisets can be described with the help of polynomials. Let variable x represent label a, and variable y label b. Then the polynomial describing the *subtrees* of both T_1 and T_2

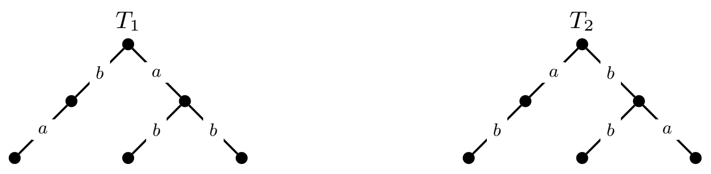

Figure 1 Two MP_{SUBTREE}-equivalent trees.

is $2x + 3y + 4xy + y^2 + x^2y + 3xy^2 + 2x^2y^2 + xy^3 + x^2y^3$, where the interpretation e.g. of the term $3xy^2$ is that there are 3 (the coefficient) subtrees that contain 1 letter a and 2 letters b (the exponents). In a similar way, polynomials can be used to describe the multisets of *paths* or of *maximal paths*. Moreover, they can be used to describe the *weights* of certain subtrees when the edges are labeled with positive integers. We will give more precise definitions later.

In this paper, we are interested in the following questions:

- **Computation:** How can we compute the polynomials describing the jumbled or weighted subtrees, paths, or maximal paths?
- **Reconstruction:** Can trees be uniquely reconstructed from the multiset of jumbled or weighted subtrees, paths, or maximal paths? I.e. are there non-isomorphic trees with the same multisets? – We split this problem into two sub-problems:
  1. **Large Unjumble:** Is the unlabeled tree (i.e., its topology) uniquely determined by the multiset of jumbled or weighted subtrees, paths, or maximal paths?
  2. **Small Unjumble:** Given the topology of the tree, is the labeling uniquely determined by the multiset of jumbled or weighted subtrees, paths, or maximal paths?

The method of using polynomials to describe multisets of Parikh vectors or of weights has been successfully applied in the past to strings. In [8] polynomials were used for representing the multiset of weights of substrings (there called *submasses*) of a weighted string, i.e. where each character is assigned a positive integer weight. Fast Fourier Transform was employed to compute this polynomial, and several algorithms were proposed for finding substrings with a given query weight, using this polynomial.

In [2] the authors describe a similar polynomial representation of the multiset of Parikh vectors of substrings, and study the class of strings having the same multiset (there called *confusable*), using algebraic methods based on this polynomial. The method was originally employed in [28] for the related *turnpike problem*: Given n unknown points on a line, reconstruct the positions of these points from the multiset of interpoint distances. Indeed, in [1], an algorithm was given for reconstruction of all confusable strings from the multiset of Parikh vectors of substrings, an adaptation of an algorithm given in [28]. Note that the turnpike problem itself can be viewed as a problem on an edge-weighted tree (a path), where the vertices are the points, the edges are weighted by the distances between consecutive points, and the input is the multiset of path weights.

In this paper, we show how the polynomial method can be extended to trees. But generalizing the substructure of *substring* to trees can result either in *subtrees*, or in *paths*. We show that the method works for both types of substructures, as well as for *maximal paths* (i.e. paths between leaves). Note that equivalence w.r.t. one does not imply equivalence w.r.t. the other.

In the case of strings, both for jumbled and for weighted substrings, the polynomial can be computed via convolution from a very easily computable polynomial with 0/1 coefficients (called *generating polynomial* in [2] and *prefix polynomial* in [8]), essentially using the fact that the Parikh vector (resp. weight) of a substring is the difference of the Parikh vectors (resp. weights) of two prefixes. We show how to compute the polynomials for trees in a similar manner, recursively from the polynomials of subtrees, but using both multiplication and addition of polynomials. Since strings can be represented as edge-labeled paths, our framework encompasses the known results on strings. Of course, if the tree T is a path, then the multisets of subtrees and of paths coincide.

The related problem of *jumbled pattern matching*, finding one or all occurrences of substructures with a given Parikh vector, has been studied recently extensively on strings, most recently in [13, 5, 25, 4, 15, 22, 24, 7, 21, 27, 29, 12, 11]; and also on vertex-colored graphs and trees [20, 14, 17]. On graphs, the problem is also called *motif search*, and it is NP-hard to decide whether a match exists, even when G is a tree [26]. When the number of colors is constant, the problem is fixed-parameter tractable w.r.t. treewidth [20].

Note that the variant of our problem where the subtrees are restricted to maximal paths is closely related to the problem of *distance-based phylogenetic reconstruction*, see e.g. [16], where a distance matrix between the leaves of a tree is given, and the task is to reconstruct the tree. The problem there is well-understood: such a tree exists if the input matrix has a certain property (called *additivity*), and an efficient algorithm exists for reconstructing the tree [30], which runs in cubic time in the number of leaves. The difference here is that we are given the input numbers without assignment to the pairs of leaves.

Following [28], we call two weighted trees T_1 and T_2 *homometric* if the multisets of pairwise distances between vertices is the same for both trees, or equivalently in our terms, $MW_{\text{PATH}}(T_1) = MW_{\text{PATH}}(T_2)$. We note that even though trees, and more generally, graphs, do appear in the literature on homometric sets [19, 6], those papers consider homometric vertex sets within one tree rather than homometric pairs of trees, while the papers [18, 3] treat quite different problems from the present ones.

Most proofs are omitted due to space limitations, and will be included in the full version.

2 The polynomial representation of Parikh multisets and weight multisets

Let Σ be a finite alphabet with elements $a_1, a_2, \ldots, a_\sigma$. Consider the polynomial ring over the integers in σ indeterminates, i.e. $\mathbb{Z}[x_1, x_2, \ldots, x_\sigma]$. When the alphabet is binary, we will denote the indeterminates by x and y.

If we interpret a Parikh vector $(k_1, k_2, \ldots, k_\sigma)$ as a multidegree, we can assign to it the monomial $x_1^{k_1} x_2^{k_2} \cdots x_\sigma^{k_\sigma}$. Note that the total degree of the polynomial equals the sum of entries of the Parikh vector. A *multiset* of Parikh vectors can then be represented as the sum of the monomials of its elements; multiplicities become coefficients. The power of this viewpoint is that disjoint union of (multi)sets corresponds to the product of the two monomials associated to the sets.

If we work with weights rather than arbitrary labels, then a single indeterminate suffices: to a weighted edge e with weight $w(e)$, we associate the polynomial $x^{w(e)}$. If we have a set U of edges and take the product of the monomials corresponding to the elements, then we get $x^{\sum_{e \in U} w(e)}$. The primary focus of the present paper are Parikh multisets and weight multisets of a tree T obtained by taking Parikh vectors or weights of subtrees of T satisfying some condition.

▶ **Definition 1.** Let T be a tree and $\mathcal{A}$ be a property of subtrees. Let the edges of T be labeled by an σ-element alphabet Σ. The $MP_{\mathcal{A}}$-polynomial of T, denoted by $f_{\mathcal{A}}(T)$ is the σ-variable polynomial associated to the multiset of Parikh vectors of all subtrees of T satisfying condition $\mathcal{A}$.

▶ **Definition 2.** Let T be a tree and $\mathcal{A}$ be a property of subtrees. Let the edges of T be weighted by positive integers. The $MW_{\mathcal{A}}$-polynomial of T, denoted by $g_{\mathcal{A}}(T)$ is the 1-variable polynomial associated to the multiset of weights of all subtrees of T satisfying condition $\mathcal{A}$.

The main reason for using polynomials to represent multisets is that we have additional algebraic structure, while all information about the multiset is still preserved. This is a crucial property used throughout (sometimes implicitly), so we state it as an observation.

▶ **Observation 3.** *Let T_1, T_2 be trees and $\mathcal{A}$ a subtree property. Then $MP_{\mathcal{A}}(T_1) = MP_{\mathcal{A}}(T_2)$ if and only if $f_{\mathcal{A}}(T_1) = f_{\mathcal{A}}(T_2)$. Similarly $MW_{\mathcal{A}}(T_1) = MW_{\mathcal{A}}(T_2)$ if and only if $g_{\mathcal{A}}(T_1) = g_{\mathcal{A}}(T_2)$.*

The following observation is also straightforward and means that $MP_{\mathcal{A}}(T)$ contains all the information for computing $MW_{\mathcal{A}}(T)$.

▶ **Observation 4.** *If the letters of the alphabets are positive integers, then they can be interpreted as weights. Then the $MW_{\mathcal{A}}$-polynomial of a tree can be calculated from the $MP_{\mathcal{A}}$-polynomial by substituting x^{a_i} into the variable x_i.*

▶ **Example 1.** *Let $\mathcal{A} =$ PATH. Let $\Sigma = \{a, b\}$ and let the indeterminate x correspond to a, and y to b. The tree T_1 in Figure 1 has the MP_{PATH}-polynomial $2x+3y+4xy+y^2+x^2y+2xy^2+2x^2y^2$, while the MP_{PATH}-polynomial of T_2 is $2x+3y+4xy+y^2+x^2y+2xy^2+x^2y^2+xy^3$. If we let $a = 3$ and $b = 2$, then the MW_{PATH}-polynomial of T_1 is $3t^2+2t^3+t^4+4t^5+2t^7+t^8+2t^{10}$. This is obtained from its MP_{PATH}-polynomial by letting $x = t^3$ and $y = t^2$. (We used a new letter t to avoid confusion.)*

In what follows, we discuss how $MP_{\mathcal{A}}$-polynomials and $MW_{\mathcal{A}}$-polynomials of a tree can be computed. We will restrict our attention to the case of $\mathcal{A} =$ SUBTREE, where all subtrees are considered, $\mathcal{A} =$ PATH, where only paths between pairs of vertices are considered and $\mathcal{A} =$ MAXPATH, where only maximal paths are considered. The theorems will be stated for $MP_{\mathcal{A}}$-polynomials, but are valid in the same form for $MW_{\mathcal{A}}$-polynomials.

Unless otherwise specified, the labels or weights are always on the edges rather than the vertices. The computation methods for vertex labeled and vertex weighted graphs are obtained by adapting the computations, which we will not state as separate theorems. Our examples of $MP_{\mathcal{A}}$-equivalent families are proved using the polynomial method. We present recursive computation methods for the three kinds of subtree properties in the following sections (the base cases for the recursion are left to the reader).

To conclude this section, we propose a new algorithmic application of $MP_{\mathcal{A}}$-polynomials (resp. $MW_{\mathcal{A}}$-polynomials) for randomized testing of $MP_{\mathcal{A}}$-equivalence (resp. $MW_{\mathcal{A}}$-equivalence) of trees. The method is based on randomized equality testing for polynomials using the Schwartz-Zippel lemma [32, 34]. The computation methods presented later all allow an efficient substitution into the polynomials, even in the case when we consider subtrees, where the size of the MP_{SUBTREE}-set and thus the number of coefficients of the polynomial can be exponential in the input. For the substitution we do not need the sequence of coefficients, we can use the recursive methods for evaluating the polynomial. Finally note that using modular arithmetic, calculations can be further sped up.

3 Subtrees

3.1 Computation of $f_{\text{SUBTREE}}(T)$

We first consider the case when all subtrees are considered in the Parikh multiset or the weight multiset. Although we work on free trees (i.e. unrooted trees), for the computations it is convenient to consider rooted trees. We root the tree T in an arbitrary vertex v and define an auxiliary polynomial $r(T, v)$, called the rooted MP-polynomial of T with root v, as the polynomial representing the Parikh multiset of all subtrees containing v. We have the following theorem.

▶ **Theorem 5.** *Let T be a rooted tree with root v. Let $v_1, v_2, \ldots, v_k$ be the children of v. Denote the subtrees rooted at v_i by T_i for $i = 1, \ldots, k$. Denote the index in Σ of the label on the edge connecting v and v_j by l_j. We have the following equations.*

$$r(T, v) = \prod_{j=1}^{k} (1 + x_{l_j} \cdot r(T_j, v_j)) \quad \text{and} \quad f(T) = r(T, v) + \sum_{j=1}^{k} f(T_j)$$

Note that Theorem 5 generalizes the computation of MP-polynomials or MW-polynomials of strings presented in e.g. [8, 28, 2] since a string can be interpreted as an edge-labeled path. The theorem also generalizes the subtree size multiset presented in [9].

3.2 Reconstructibility – Large Unjumble

For a general labeled tree T, one can ask if the unlabeled version of the tree (the topology) can be uniquely reconstructed from $MP_{\text{SUBTREE}}(T)$ or $MW_{\text{SUBTREE}}(T)$. This is already impossible from $MP_{\text{SUBTREE}}(T)$ for a trivial (i.e. one-element) alphabet, which also implies that $MW_{\text{SUBTREE}}(T)$ does not determine the isomorphism class of the unlabeled tree either.

If one puts the same label (resp. weight) on each edge, then the Parikh vector (resp. weight) of a subtree simply counts the number of edges in that subtree. It was proved in [9] that knowing the number of subtrees with k edges for all k, that is, in our terms, knowing $MP_{\text{SUBTREE}}(T)$ for one-letter alphabets is not generally enough for unique reconstruction of the tree up to isomorphism.

▶ **Proposition 6** ([9]). *Let $\Sigma = \{1\}$. There exist infinitely many pairs of trees T_1, T_2, such that if we label each edge with the only element of Σ, then $MP_{\text{SUBTREE}}(T_1) = MP_{\text{SUBTREE}}(T_2)$ and $MW_{\text{SUBTREE}}(T_1) = MW_{\text{SUBTREE}}(T_2)$.*

In the positive direction, we mention the following reconstructibility result from the same paper. A *spider* is a tree with one vertex of degree at least 3 and all others with degree at most 2 (called *star-like trees* in that paper).

▶ **Theorem 7** ([9]). *Let $|\Sigma| = 1$, and T_1, T_2 be two edge-labeled spiders with labels from Σ. If $MP_{\text{SUBTREE}}(T_1) = MP_{\text{SUBTREE}}(T_2)$, then T_1 and T_2 are isomorphic.*

3.3 Reconstructibility – Small Unjumble

When the alphabet is non-trivial, there are several non-isomorphic labelings of a typical tree. We consider reconstructibility of the labels for a fixed unlabeled tree. Note that the problem of reconstructing a string from its substring compositions [2] is a special case: a string can be represented as a path of equal length where the edge labels correspond to individual characters in the string.

The problem of reconstructing a 1-dimensional point set from interpoint distances considered in [28] is also a special case of the reconstruction of a tree from $MW(T)$: the weights are the distances between neighboring points on a line. Since every subtree of a path is a (sub)path, this remark also applies for reconstructibility from path Parikh vectors (resp weights), addressed in the following section. The above two problems can also be reduced to the case of vertex labeled paths.

We give non-reconstructibility examples for trees that are not a path. The smallest pair of non-isomorphic MP_{SUBTREE}-equivalent edge labeled trees are on six vertices.

▶ **Example 2.** *Let P be a path of length 4, whose vertices are called $v_1, v_2, \dots, v_5$ and the edges $v_1v_2, \dots$ are labeled with a, b, a, b. Construct T_1 by attaching a 6th vertex v_6 to v_4 with an edge labeled by b. Construct T_2 from P by attaching a 6th vertex to v_2 with an edge labeled by b. See the example in Fig. 1.*

It is also possible to attach a larger tree instead of the sixth vertex, which gives larger examples of MP_{SUBTREE}-equivalent pairs. We remark that the smallest such example for vertex labeled trees is on 7 vertices. We also give a construction that yields an infinite family of MP_{SUBTREE}-equivalent examples (similar constructions work for vertex labeled trees).

▶ **Proposition 8.** *Let s_1 and s_2 be two MP_{SUBTREE}-equivalent strings of length k over a binary alphabet Σ_1. Create two MP_{SUBTREE}-equivalent edge-labeled paths P_1, P_2 by using characters of the strings as labels. Let U be an edge labeled rooted tree with labels from a disjoint alphabet Σ_2. Create T_j $(j = 1, 2)$ from P_j by joining $k+1$ copies of U to each vertex of P_j, identifying the vertex on the path and the root of U. Then T_1 and T_2 are not isomorphic as labeled trees, but $MP_{\text{SUBTREE}}(T_1) = MP_{\text{SUBTREE}}(T_2)$.*

Finally, we present a result stating that, unsurprisingly, MP-equivalence does not generally follow from MW-equivalence, already for 2-letter alphabets. We have an infinite family already for paths.

▶ **Proposition 9.** *Let $k \leq n$ an integer, $\Sigma = \{1, 2\}$. Let P_1 be a path of length $14 + 5k$, edge labeled with elements of the sequence $s_1 = 21211112222122(12122)^k$. Let P_2 be a path of length 14, edge labeled with elements of the sequence $s_2 = 22111121222212(12212)^k$. Then $MP_{\text{SUBTREE}}(P_1) \neq MP_{\text{SUBTREE}}(P_2)$, but $MW_{\text{SUBTREE}}(P_1) = MW_{\text{SUBTREE}}(P_2)$.*

4 Paths

4.1 Computation of $f_{\text{PATH}}(T)$

Let $f(T) = f_{\text{PATH}}(T)$ be the MP_{PATH}-polynomial of T. Root T is an arbitrary vertex v. We denote by $r(T, v)$ the polynomial corresponding to all paths at least one of whose endpoints is v. We include 0-length paths in the computation, since it makes the formulae simpler, this adds a constant n (the number of vertices) to the polynomial.

▶ **Theorem 10.** *Let T be a rooted tree with root v. Let $v_1, v_2, \dots v_k$ be the children of v in T. Denote the subtrees rooted at v_1 (resp. v_2 etc.) by T_1 (resp. T_2 etc.). Denote the index in Σ of the label on the edge connecting v and v_j by l_j. We have the following equalities:*

$$r(T, v) = 1 + \sum (x_{l_j} \cdot r(T_j, v_j)), f(T) = r(T, v) + \sum_{j=1}^{k} f(T_j) + \sum_{1 \leq i < j \leq k} (x_{l_i} x_{l_j} r(T_i, v_i) r(T_j, v_j))$$

Proof. For the statement on r note that a path starting in v either stops there immediately, or contains exactly one of the v_j and thus a path from v_j to a vertex of T_j as a subpath. To understand the identity for f, observe that a path in T either contains v as one of its endpoints or is entirely contained in one of the T_j, or else it is the union of two paths which both have v as one endpoint and their respective other endpoints in distinct T_i and T_j. ◀

4.2 Reconstructibility – Large Unjumble

For a general labeled tree T, one can ask if the unlabeled version of the tree can be uniquely reconstructed from $MP_{\text{PATH}}(T)$ or $MW_{\text{PATH}}(T)$. We show that this is already impossible from $MP_{\text{PATH}}(T)$ for a one-element alphabet, which also implies that $MW_{\text{PATH}}(T)$ does not determine the isomorphism class of the unlabeled tree either.

In the following, we give infinitely many examples of pairs of unlabeled trees that are homometric. This can be considered as a special case of MP_{PATH}-equivalence (resp. MW_{PATH}-equivalence) when $|\Sigma| = 1$ (resp. we use the same weight everywhere).

▶ **Proposition 11.** *For $n \geq 11$ and odd, let T_1 be a tree constructed from a 5-star by adding respectively $1, 1, (n-5)/2$ and $(n-9)/2$ new vertices joined to the star's leaves, obtaining a tree on n vertices. Construct T_2 similarly, by adding 0, 2, $(n-7)/2$ and $(n-7)/2$ new vertices adjacent to the star's leaves. Then T_1 and T_2 are homometric but are not isomorphic.*

For $n \geq 12$ and even, let T_1 be a tree constructed from a 6-star by adding respectively $1, 1, 1$, $(n-6)/2$ and $(n-10)/2$ new vertices to the star's leaves, obtaining a tree on n vertices. Construct T_2 similarly, but add $0, 1, 2$, $(n-8)/2$ and $(n-8)/2$ new vertices. Then T_1 and T_2 are homometric but are not isomorphic.

We remark that all one has to do is check the number of paths of length $1, 2, 3$ and 4 since the constructed trees have diameter 4. The calculation is straightforward, and the idea behind it is that if we add k_1, k_2, k_3 and k_4 vertices to the 5-star as above, then the number of 1-paths (resp. 2-paths, 3-paths and 4-paths) is already determined by their sum and the sum of their squares, and these values are identical for the two trees. One can compose such trees by solving instances of the Prouhet-Tarry-Escott problem, see e.g. [10], Chap. 11.

4.3 Reconstructibility – Small Unjumble

We now turn to the problem of unique reconstructibility of the labeling, once the unlabeled version of the tree is known. Again, if we take the viewpoint of strings being (either edge or vertex) labeled graphs, then this problem contains as a special case the problem of string reconstructibility from MW_{PATH} or MP_{PATH}. We thus focus on reconstructibilty for other trees. First remark that the contruction in Proposition 8 also yields infinitely many MP_{PATH}-equivalent pairs of non-isomorphic trees.

We now give a family of pairs that are *vertex labeled* PM-equivalent trees.

▶ **Proposition 12.** *Let $k \geq 1$ an integer. Let P_{base} be a path of length 3 with alternating vertex labels $0, 1, 0, 1$, and P_{k-1} be a path on k vertices, al labeled by 0. Construct T_1 by attaching two copies of P_{k-1} to P_{base} with two edges: one is attached to the leaf with 0 label, and the other to the neighboring vertex on P_{base}. We get a tree on $2k+4$ vertices. The construction of T_2 is similar, but P_{base} is reversed. Then T_1 and T_2 are two different labelings of the same tree, and $MP_{\text{PATH}}(T_1) = MP_{\text{PATH}}(T_2)$.*

Class sizes for MP-equivalence. For paths, the size of an equivalence class of MW_{PATH}-equivalent paths (resp. MP_{PATH}-equivalent paths) is always a power of 2, as it was proved in [28] (resp. [2]). This result no longer holds for other classes of trees, as illustrated by the example below.

▶ **Example 3.** *Let T be a spider on 11 vertices with 5 legs of length 2. Then the following 3 weightings of T form an MP_{PATH}-equivalence class of size 3 (we give the weighting as 5-tuples of weight pairs on the legs from the center outwards).* T_1 : $[1,3],[2,3],[3,5],[4,1],[6,1]$, T_2 : $[1,3],[2,5],[3,1],[5,1],[5,3]$, T_3 : $[1,5],[2,1],[4,1],[4,3],[5,3]$.

Finally note that Proposition 9 also applies for $\mathcal{A} = \text{PATH}$.

5 Maximal paths

5.1 Computation of $f_{\text{MAXPATH}}(T)$

Let $f(T) = f_{\text{MAXPATH}}(T)$ be the MP_{MAXPATH}-polynomial of T. Let $r(T,v)$ denote the MP-polynomial corresponding to all paths with one endpoint in v and another one in a leaf. Finally, let $t(T,v)$ be the MP-polynomial for all maximal paths that have v as one of their endpoints. Note that $t(T,v) = 0$ if v is not a leaf in T.

▶ **Theorem 13.** *Let T be a rooted tree with root v, and let $v_1, v_2, \ldots v_k$ be the children of v in T. Denote the subtrees rooted at v_1 (resp. v_2 etc.) by T_1 (resp. T_2 etc.). Denote the index in Σ of the label on the edge connecting v and v_j by l_j. We have the following equalities:*

$$\begin{aligned}
r(T,v) &= \sum(x_{l_j} \cdot r(T_j, v_j)) \\
f(T) &= t(T,v) + \sum_{j=1}^{k}(f(T_j) - t(T_j, v_j)) + \sum_{i<j}(x_{l_i} x_{l_j} r(T_i, v_i) r(T_j, v_j)) \\
t(T,v) &= \begin{cases} r(T,v) & \text{if } k=1 \\ 0 & \text{if } k>1 \end{cases}
\end{aligned}$$

5.2 Reconstructibility – Small Unjumble

We only consider reconstructibility for weighted graphs. Let us fix the topology of T as an n-star. We have the following reconstructibility result for edge-weighted n-stars.

▶ **Theorem 14.** *Let T_1 and T_2 be two n stars s.t. $n-1$ is not a power of 2. Then $MW_{\text{MAXPATH}}(T_1) = MW_{\text{MAXPATH}}(T_2)$ implies that T_1 and T_2 are isomorphic as edge weighted trees. If $n = 2^k + 1$ for some $k \leq 0$, then there are non-isomorphic edge labeled n-stars that are MW_{MAXPATH}-equivalent.*

The theorem is an easy consequence of Theorem 1 and Theorem 2 from [33], about the reconstructibility of numbers from the multiset of their pairwise distances. These two theorems are also proved in [23, 31] using the polynomial representation of sumsets, which is more in the spirit of the present paper. To see how it follows, simply observe that the weights of maximal paths are the pairwise sums of edge labels.

6 Reconstruction Algorithms

In this section, we treat reconstruction of edge-labeled trees from *weighted paths*, where the topology of the tree is given (Small Unjumble). Note that we assume that $\mathcal{S}$ is given sorted.

First let us note that for the case where T is a star, a simple greedy algorithm will solve the problem exactly in time loglinear in the input size $|\mathcal{S}| = \binom{n}{2}$. Denote by X the multiset of weights of the edges, then $MW_{\text{PATH}}(T) = X \cup (X + X)$ (where by $X + X$ we denote the multiset of sums of two elements from the multiset X). Clearly, the two smallest numbers in $\mathcal{S}$ are necessarily in X, which means that their sum is necessarily in $X + X$. The algorithm starts with an empty X, iteratively chooses the smallest remaining number in $\mathcal{S}$, adds it to X, and eliminates it and its sums with those already in X from $\mathcal{S}$. We touch each of the $\binom{n}{2}$ input numbers exactly once; getting the next smallest one takes constant time, while finding the corresponding elements from $X + X$ takes $\log n$ time each.

▶ **Example 4.** *Let T be a 6-star, i.e. $|V(T)| = 6$ with one vertex of degree 5 and 5 leaves, and let $\mathcal{S} = \{2, 3, 5, 5, 7, 8, 9, 10, 11, 12, 12, 13, 14, 15, 19\}$. Necessarily $2, 3 \in X$, and this eliminates also $5 = 2 + 3$ from the input set. The next remaining smallest number is 5: this must again be an edge label, thus $5 \in X$, eliminating $7 = 5 + 2$ and $8 = 5 + 3$ from our input set. Continuing, we get that $9 \in X$, eliminating $11, 12, 14$, and finally, that $10 \in X$, eliminating $12, 13, 15, 19$. So we see that the 5 edges are labeled with $2, 3, 5, 9$, and 10 respectively.*

In particular, if T is a star, then if there is a solution, it is necessarily unique. Thus we have proved the following:

▶ **Proposition 15.** *If T is a star, then the Greedy Algorithm correctly reconstructs its labeling from $MW_{\text{PATH}}(T)$ in time $O(n^2 \log n)$. Moreover, for any instance $\mathcal{S}$, either $\mathcal{S}$ is uniquely reconstructable, or there is no solution.*

Now let's turn to a general tree topology. In the following we will generalize the algorithm given in [28] for the turnpike problem to any tree. To this end, we define the *path poset* of a tree T as the set of all paths in T, together with the inclusion order. We give an example below (Ex. 5). Note that the input $MW_{\text{PATH}}(T)$ consists precisely of the weights of all elements of the path poset. So the task is to fill in the values from $\mathcal{S}$ into the path poset. The following is immediate:

▶ **Lemma 16.** *For any tree T, the path poset of T is exactly the union of the path posets of its maximal paths.*

▶ **Example 5.** *Let T be as in Fig. 2, input $\mathcal{S} = \{1, 2, 2, 3, 3, 4, 5, 5, 5, 6, 6, 7, 8, 9, 11\}$. In the same figure, we show the three pyramids with the unique solution (up to exchanging the labels of d and e).*

Following [28], we will refer to the above representation of the values of the path poset of a maximal path as a *pyramid.* If $\pi = (v_1, \ldots, v_s)$ is a maximal path in T, then in its pyramid Δ, row k will hold all values of subpaths of π of length k. Let us refer to d_{ij} as the sum of the weights on the path from v_i to v_j. As was shown in [28], the following relationships hold within one pyramid:

▶ **Lemma 17** ([28])**.** *$d_{ij} + d_{k\ell} = d_{i\ell} + d_{kj}$ for $1 \leq i \leq k \leq \ell \leq j$.*

This property is then used in [28] for a backtracking algorithm which takes the next largest remaining value, guesses its position in the pyramid, and fills in all other values which are implied by it. When a choice implies a value not present in the input, the algorithm backtracks. We, however, need to fill in all pyramids concurrently. For this, the following lemma will be useful. We omit the proof for lack of space.

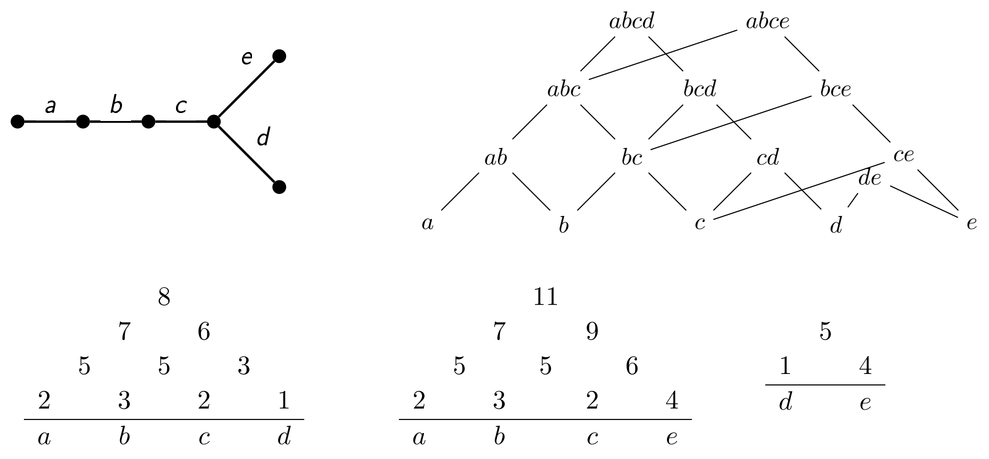

Figure 2 Example 5: A tree, its path poset, and the path posets of its three maximal paths (in the latter we omit the edges for clarity), with the values of the input set filled in.

▶ **Lemma 18.** *Let $\pi = (v_1, \ldots, v_r)$ and $\pi' = (u_1, \ldots, u_{r'})$ be two maximal paths in T with non-empty intersection ρ. Let Δ be the pyramid for π, with entries d_{ij}, and Δ' the pyramid for π', with entries d'_{ij}. If $\rho = (v_i, \ldots, v_{i+\ell}) = (u_{i'}, \ldots, u_{i'+\ell})$, then the following relationships hold between Δ and Δ':*

1. *for $k \leq i, k' \leq i'$: $d_{k,i+s} - d'_{k',i'+s} = d_{k,i+t} - d'_{k',i'+t}$ for all $0 \leq s,t \leq \ell$, and*
2. *for $k \geq i+\ell, k' \geq i'+\ell$: $d_{i+s,k} - d'_{i'+s,k'} = d_{i+t,k} - d'_{i'+t,k'}$ for all $0 \leq s,t \leq \ell$.*

Our algorithm proceeds as follows. In each step, it takes the next largest value still in $\mathcal{S}$ and places it in one of the maximal free places, i.e. in a free place that has no larger free place in any of the pyramids. It then fills in all implied positions according to Lemma 17 and 18. If at some point it encounters a value not present among the yet unused values, it backtracks. For example, in Example 5, for the first value 11 there are three possible choices, namely the tops of the three pyramids. Say we have already placed values 11 and 9 in their respective places as in the final solution. Now placing 8 on the top of the first pyramid will force the difference for all values on the right sides of the first and second pyramids to be 3, an application of Lemma 17.

▶ **Lemma 19.** *Every maximal free place is either on top of a pyramid, or on the side of a pyramid.*

▶ **Theorem 20.** *There is a $O((2\Gamma)^{(\Gamma+n)}n^2 \log n)$ algorithm for finding all possible labelings of a given tree T from the multiset of $\binom{n}{2}$ path weights, where n is the number of vertices of T, and Γ the number of maximal paths in T.*

Although the algorithm has exponential running time, it compares well to the simple exhaustive search if the number of leaves is small, since trying all possible labelings of the edges would give $O(n^{2n}/2^n)$ running time. Note that parameter Γ is quadratic in the number of leaves. So essentially the algorithm performs well on trees which are close to strings, and badly on trees that are close to stars, i.e. have many leaves. Indeed, as can be seen, the Greedy algorithm for stars applies the opposite strategy, namely filling in the path poset from below; this makes sense when the higher levels are more populous than the lower levels, while starting from above is appropriate when the form is pyramid-like. Moreover, the analysis is very pessimistic and does not so far take advantage of the improvements given by the pruning due to Lemmas 17 and 18. In practice, we expect that many branches will be pruned by these implications. For the special case of the turnpike problem, if we consider random instances then incorrect branches are pruned almost immediately, see [28].

7 Conclusion and Open Problems

Our reconstruction algorithm is purely combinatorial, and it seems a challenging problem to find a reconstruction algorithm based on MP-polynomials, similar to the ones presented in [28, 2]. We would also be interested in proving further results about unique reconstructibility with algebraic techniques.

Another intriguing task is connecting the Large Unjumble Problem for weighted maximal paths to the distance-based phylogeny problem: Note that if we had an assignment of the input numbers to the Γ leaf pairs, then a reconstruction, if it exists, is unique, and can be found in $O(\Gamma^{3/2})$ e.g. using the Neighbor Joining algorithm [30] (or it can be shown that no such reconstruction exists).

Further open problems include the complexity status of the reconstruction problems introduced, in particular in which of the cases Large Unjumble is computationally hard.

References

1 Jayadev Acharya, Hirakendu Das, Olgica Milenkovic, Alon Orlitsky, and Shengjun Pan. Quadratic-backtracking algorithm for string reconstruction from substring compositions. In *2014 IEEE Int. Symp. on Information Theory (ISIT 2014)*, pages 1296–1300, 2014. doi:10.1109/ISIT.2014.6875042.

2 Jayadev Acharya, Hirakendu Das, Olgica Milenkovic, Alon Orlitsky, and Shengjun Pan. String reconstruction from substring compositions. *SIAM J. Discrete Math.*, 29(3):1340–1371, 2015. doi:10.1137/140962486.

3 Tatsuya Akutsu, Daiji Fukagawa, Jesper Jansson, and Kunihiko Sadakane. Inferring a graph from path frequency. *Discrete Applied Mathematics*, 160(10-11):1416–1428, 2012. doi:10.1016/j.dam.2012.02.002.

4 Amihood Amir, Ayelet Butman, and Ely Porat. On the relationship between histogram indexing and block-mass indexing. *Philosophical Transactions of The Royal Society A: Mathematical Physical and Engineering Sciences*, 372(2016), 2014. doi:10.1098/rsta.2013.0132.

5 Amihood Amir, Timothy M. Chan, Moshe Lewenstein, and Noa Lewenstein. On hardness of jumbled indexing. In *41st Int. Coll. on Automata, Languages, and Programming (ICALP 2014)*, volume 8572 of *Lecture Notes in Computer Science*, pages 114–125. Springer, 2014. doi:10.1007/978-3-662-43948-7_10.

6 Maria Axenovich and Lale Özkahya. On homometric sets in graphs. *Electronic Notes in Discrete Mathematics*, 38:83–86, 2011. doi:10.1016/j.endm.2011.09.014.

7 Golnaz Badkobeh, Gabriele Fici, Steve Kroon, and Zsuzsanna Lipták. Binary Jumbled String Matching for Highly Run-Length Compressible Texts. *Information Processing Letters*, 113:604–608, 2013. doi:10.1016/j.ipl.2013.05.007.

8 Nikhil Bansal, Mark Cieliebak, and Zsuzsanna Lipták. Efficient algorithms for finding submasses in weighted strings. In *Proc. of the 15th Ann. Symp. on Combinatorial Pattern Matching (CPM 2004)*, volume 3109 of *Lecture Notes in Computer Science*, pages 194–204. Springer, 2004. doi:10.1007/978-3-540-27801-6_14.

9 Dénes Bartha and Péter Burcsi. Reconstructibility of trees from subtree size frequencies. *Stud. Univ. Babeş-Bolyai Math.*, 59:435–442, 2014.

10 Peter B. Borwein. *Computational excursions in analysis and number theory.* CMS books in mathematics. Springer, New York, Berlin, Heidelberg, 2002.

11 Péter Burcsi, Ferdinando Cicalese, Gabriele Fici, and Zsuzsanna Lipták. Algorithms for Jumbled Pattern Matching in Strings. *Int. Journal of Foundations of Computer Science*, 23:357–374, 2012. doi:10.1142/S0129054112400175.

12 Péter Burcsi, Ferdinando Cicalese, Gabriele Fici, and Zsuzsanna Lipták. On Approximate Jumbled Pattern Matching in Strings. *Theory of Computing Systems*, 50:35–51, 2012. doi:10.1007/s00224-011-9344-5.

13 Timothy M. Chan and Moshe Lewenstein. Clustered integer 3SUM via additive combinatorics. In *Proc. of 47th Annual ACM Symposium on Theory of Computing (STOC 2015)*, pages 31–40, 2015. doi:10.1145/2746539.2746568.

14 Ferdinando Cicalese, Travis Gagie, Emanuele Giaquinta, Eduardo Sany Laber, Zsuzsanna Lipták, Romeo Rizzi, and Alexandru I. Tomescu. Indexes for jumbled pattern matching in strings, trees and graphs. In *20th Int. Symp. on String Processing and Information Retrieval (SPIRE 2013)*, volume 8214 of *Lecture Notes in Computer Science*, pages 56–63. Springer, 2013. doi:10.1007/978-3-319-02432-5_10.

15 Ferdinando Cicalese, Eduardo Sany Laber, Oren Weimann, and Raphael Yuster. Approximating the maximum consecutive subsums of a sequence. *Theoret. Comput. Sci.*, 525:130–137, 2014. doi:10.1016/j.tcs.2013.05.032.

16 Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme J. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* CUP, 1998.

17 Stephane Durocher, Robert Fraser, Travis Gagie, Debajyoti Mondal, Matthew Skala, and Sharma V. Thankachan. Indexed geometric jumbled pattern matching. In *Proc. of the 25th Annual Symposium on Combinatorial Pattern Matching (CPM 2014)*, volume 8486 of *Lecture Notes in Computer Science*, pages 110–119. Springer, 2014. doi:10.1007/978-3-319-07566-2_12.

18 Tomás Feder and Rajeev Motwani. On the graph turnpike problem. *Inf. Process. Lett.*, 109(14):774–776, 2009. doi:10.1016/j.ipl.2009.03.024.

19 Radoslav Fulek and Slobodan Mitrovic. Homometric sets in trees. *Eur. J. Comb.*, 35:256–263, 2014. doi:10.1016/j.ejc.2013.06.008.

20 Travis Gagie, Danny Hermelin, Gad M. Landau, and Oren Weimann. Binary jumbled pattern matching on trees and tree-like structures. *Algorithmica*, 73(3):571–588, 2015. doi:10.1007/s00453-014-9957-6.

21 Emanuele Giaquinta and Szymon Grabowski. New algorithms for binary jumbled pattern matching. *Inf. Process. Lett.*, 113(14–16):538–542, 2013. doi:10.1016/j.ipl.2013.04.013.

22 Danny Hermelin, Gad M. Landau, Yuri Rabinovich, and Oren Weimann. Binary jumbled pattern matching via all-pairs shortest paths. *CoRR*, abs/1401.2065, 2014. URL: http://arxiv.org/abs/1401.2065.

23 Ross Honsberger. *In Polya's Footsteps: Miscellaneous Problems and Essays (Dolciani Mathematical Expositions).* The Mathematical Association of America, October 1997.

24 Tomasz Kociumaka, Jakub Radoszewski, and Wojciech Rytter. Efficient indexes for jumbled pattern matching with constant-sized alphabet. In *21st Annual European Symposium on Algorithms (ESA 2013)*, volume 8125 of *Lecture Notes in Computer Science*, pages 625–636. Springer, 2013. doi:10.1007/978-3-642-40450-4_53.

25 Eduardo Laber, Wilfredo Bardales, and Ferdinando Cicalese. On lower bounds for the maximum consecutive subsums problem and the $(min, +)$-convolution. In *Proceedings of the 2013 IEEE Int. Symp. on Information Theory (ISIT 2013)*. IEEE, 2014.

26 Vincent Lacroix, Cristina G. Fernandes, and Marie-France Sagot. Motif search in graphs: Application to metabolic networks. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 3(4):360–368, 2006. doi:10.1109/TCBB.2006.55.

27 Lap-Kei Lee, Moshe Lewenstein, and Qin Zhang. Parikh matching in the streaming model. In *19th Int. Symp. on String Processing and Information Retrieval (SPIRE 2012)*, volume 7608 of *Lecture Notes in Computer Science*, pages 336–341. Springer, 2012. doi:10.1007/978-3-642-34109-0_35.

28 Paul Lemke, Steven S. Skiena, and Warren D. Smith. *Discrete and Computational Geometry: The Goodman-Pollack Festschrift*, chapter Reconstructing Sets From Interpoint Distances, pages 597–631. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. doi:10.1007/978-3-642-55566-4_27.

29 Tanaeem M. Moosa and M. Sohel Rahman. Sub-quadratic time and linear space data structures for permutation matching in binary strings. *J. Discr. Algorithms*, 10:5–9, 2012. doi:10.1016/j.jda.2011.08.003.

30 N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol*, 4(4):406–425, 1987.

31 Svetoslav Savchev and Titu Andreescu. *Mathematical Miniatures*, volume 43 of *Anneli Lax New Mathematical Library*. The Mathematical Association of America, 2003.

32 Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980. doi:10.1145/322217.322225.

33 J. L. Selfridge and E. G. Straus. On the determination of numbers by their sums of a fixed order. *Pacific J. Math.*, 8(4):847–856, 1958.

34 Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proc. of Symbolic and Algebraic Computation (EUROSAM'79)*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979.