Wang *Algorithms for Molecular Biology* (2022) 17:13
https://doi.org/10.1186/s13015-022-00220-0

Algorithms for
Molecular Biology

RESEARCH

Open Access

Two metrics on rooted unordered trees with labels

Yue Wang[1,2*]

Abstract

Background: The early development of a zygote can be mathematically described by a developmental tree. To compare developmental trees of different species, we need to define distances on trees. If children cells after a division are not distinguishable, developmental trees are represented by the space $\mathcal{T}$ of rooted trees with possibly repeated labels, where all vertices are unordered. If children cells after a division are partially distinguishable, developmental trees are represented by the space $\mathcal{P}$ of rooted trees with possibly repeated labels, where vertices can be ordered or unordered.

Results: On $\mathcal{T}$, the space of rooted unordered trees with possibly repeated labels, we define two metrics: the best-match metric and the left-regular metric, which show some advantages over existing methods. On $\mathcal{P}$, the space of rooted labeled trees with ordered or unordered vertices, there is no metric, and we define a semimetric, which is a variant of the best-match metric. To compute the best-match distance between two trees, the expected time complexity and worst-case time complexity are both $\mathcal{O}(n^2)$, where n is the tree size. To compute the left-regular distance between two trees, the expected time complexity is $\mathcal{O}(n)$, and the worst-case time complexity is $\mathcal{O}(n \log n)$.

Conclusions: For rooted labeled trees with (fully/partially) unordered vertices, we define metrics (semimetric) that have fast algorithms to compute and have advantages over existing methods. Such trees also appear outside of developmental biology, and such metrics can be applied to other types of trees which have more extensive applications, especially in molecular biology.

Keywords: Metric, Unordered tree, Label, Semimetric

Background

In developmental biology, the early development of a zygote is a central topic. For most species, the zygote follows a highly deterministic process. For example, consider a zygote of *Arabidopsis thaliana*. In stage 1, the zygote divides asymmetrically along the apical-basal axis into two cells. In stage 2, the upper (apical) cell undergoes a symmetric horizontal (meridional) division, and the lower (basal) cell undergoes a vertical (equatorial) division. In stage 3, the upper two cells divide asymmetrically, and the lower two cells undergo symmetric vertical divisions. In stage 4, the upper four cells divide asymmetrically, the middle two cells do not divide, and the lower two cells undergo symmetric vertical divisions [1]. See Fig. 1 for illustrations of this process.

A mathematical representation of the zygote's early development is a developmental tree [2]. In this tree, each vertex represents a cell. Each cell has a label, representing the cell event it will perform, such as division (symmetric or asymmetric, horizontal or vertical), growth, and death. The root vertex is the zygote. Parent vertices (cells) and children vertices (cells) are linked by edges. Each level of this tree corresponds to all the cells at a given stage. See Fig. 2 for the developmental tree of *Arabidopsis thaliana*.

*Correspondence: yuew@g.ucla.edu

[1] Department of Computational Medicine, University of California, Los Angeles, USA
Full list of author information is available at the end of the article

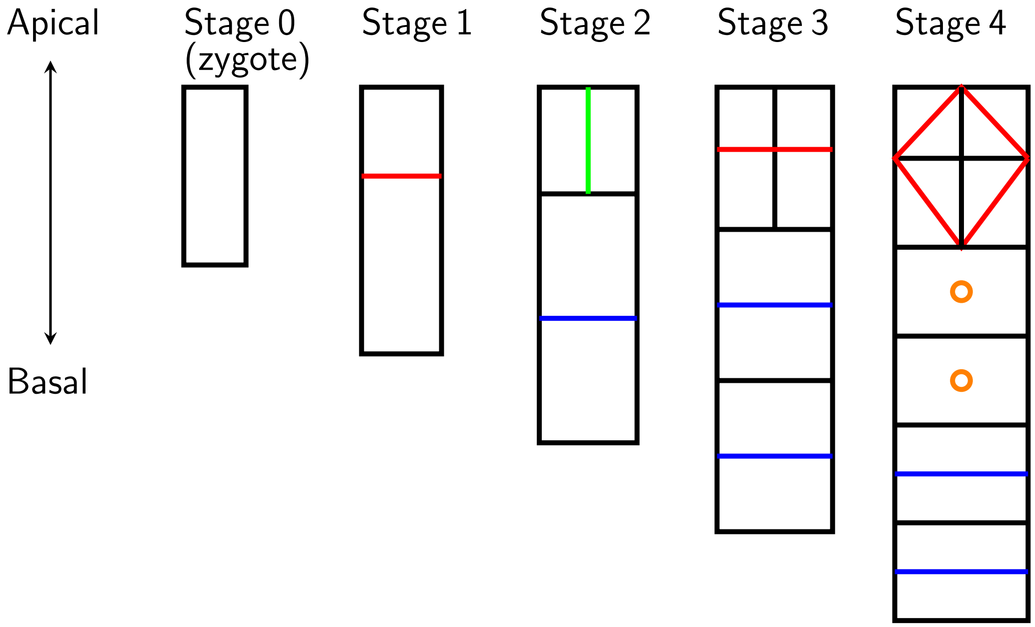

Fig. 1 Early development of an *Arabidopsis thaliana* zygote [1]. Each unit is a cell. A green line between two cells means these two cells were just generated by a symmetric horizontal division. A blue line between two cells means these two cells were just generated by a symmetric vertical division. A red line between two cells means these two cells were just generated by an asymmetric division. An orange circle in a cell means this cell did not divide during the last stage

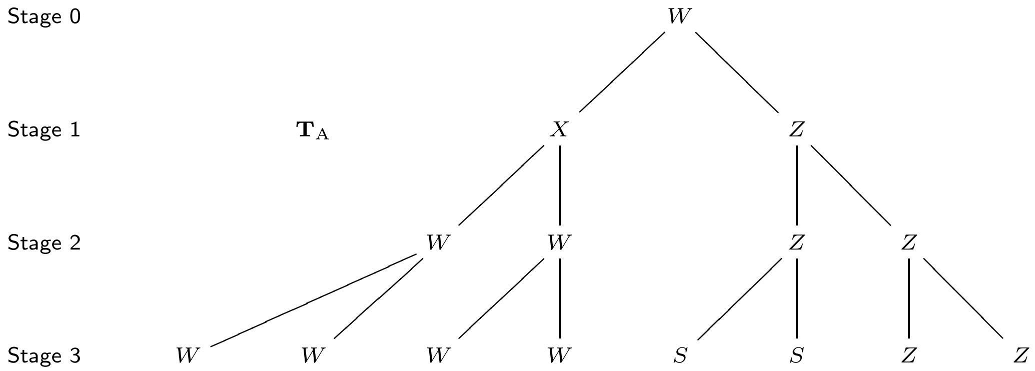

Fig. 2 The developmental tree of *Arabidopsis thaliana*, T_A, corresponding to Fig. 1. Each vertex represents a cell, and its label represents the cell event it performs. Label *X* means symmetric horizontal division; label *Z* means symmetric vertical division; label *W* means asymmetric division; label *S* means the cell stays still and does not divide

Zygotes of different species can have different early developments. See Figs. 3 and 4 for the early development of a sea urchin zygote and the corresponding developmental tree [3]. Starting from the zygote, sea urchin and *Arabidopsis thaliana* are different in division plane and division symmetry, and the cell numbers at stage 4 are already different (16 vs. 14). To quantitatively study the development of different organisms, we need a mathematical method to compare different developmental trees.

When we plot and compare developmental trees, we need to embed them in the plane, namely considering their planar embeddings. We put the zygote to the top, and its two children to the next lower level, and so on. An important question is: after a cell division, which child cell should be put to the left, and which to the right? In some situations, we cannot distinguish two children cells, and we can arbitrarily switch the position of these two children cells in the planar embedding. See Fig. 5 for equivalent planar embeddings of the same tree. Notice

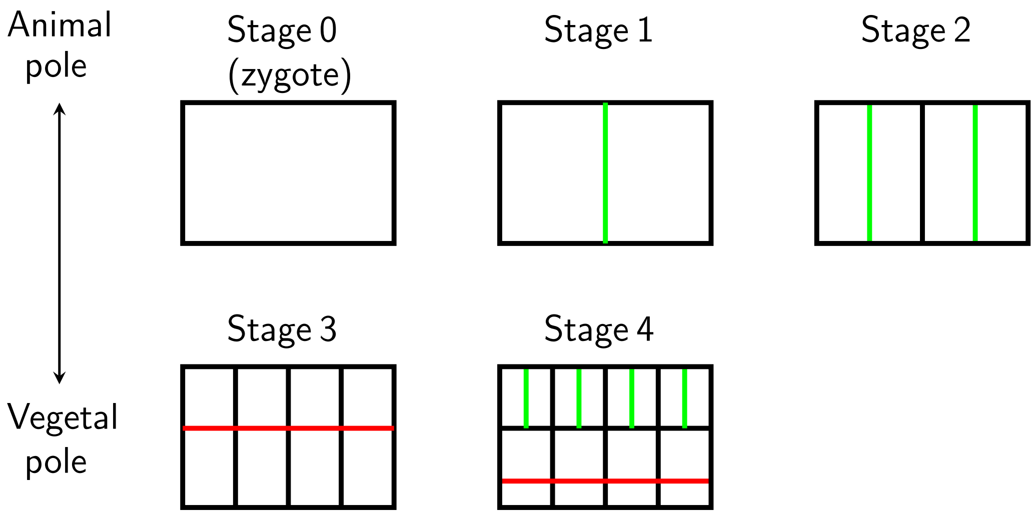

Fig. 3 Early development of a sea urchin zygote [3]. Each unit is a cell. A green line between two cells means these two cells were just generated by a symmetric horizontal division. A red line between two cells means these two cells were just generated by an asymmetric division

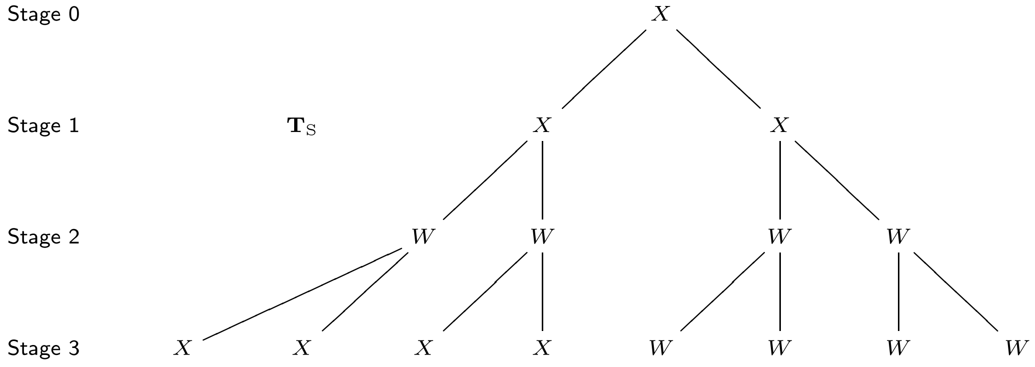

Fig. 4 The developmental tree of sea urchin, T_S, corresponding to Fig. 3. Each vertex represents a cell, and its label represents the cell event it performs. Label *X* means symmetric horizontal division; label *W* means asymmetric division

that when we switch cells in the planar embedding, the corresponding cell events are also switched. In some situations, we can distinguish two children cells from an asymmetric division or by which cell inherits the mother centriole [4]. Then we can set a rule to determine which child cell is the left child in the planar embedding, and we cannot switch these two children cells.

We start from the easier situation that we cannot distinguish children cells, so that in the planar embedding of the developmental tree, we can switch two subtrees for each vertex. Notice that a developmental tree has the zygote as its root, and different vertices can have the same label (cell event). The goal is to compare developmental trees.

In the language of graph theory, we need to define a metric on the space of rooted unordered trees with possibly repeated labels. Each tree has a root vertex, and each vertex has a label that is not necessarily unique. All vertices are unordered, meaning that we can switch left and right children in the planar embedding of each tree. Vertices and their labels are always associated, so that we do not distinguish a vertex and the label of a vertex. Therefore, when switching vertices, their labels are also switched. Such trees are not limited to developmental biology, but can be applied in various fields.

There are many metrics defined on trees, which can be roughly classified into three groups by their ideas: (1) Calculate the minimal operations needed to transform one tree into another, such as rearrangement distance [5], tree edit distance [6], edge rotation distance [7], and geodesic distance [8]. (2) Find the largest common structure of two trees, such as bottom-up distance [9] and subtree

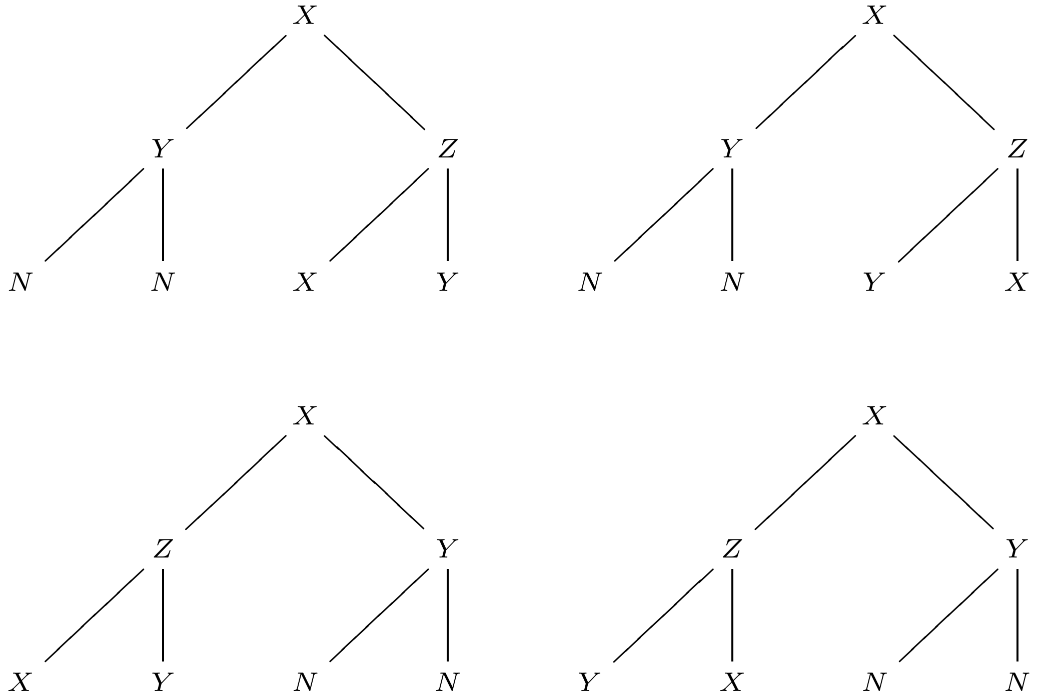

Fig. 5 An equivalent class of ordered trees, consisting of four equivalent ordered trees that differ by exchanging the left and right subtrees of some vertices

distance [10]. (3) Compare structures induced by the trees (e.g., splits or triple-vertices subtrees), such as Robinson-Foulds metric [11], matching cluster distance [12], and triples distance [13].

However, many existing methods have specific requirements on trees, so that they are not applicable in our case (rooted unordered trees with possibly repeated labels). Some methods require that different vertices have different labels, and different trees have the same label set [5, 7]. Some methods work for phylogenetic trees: only leaves vertices have labels; different vertices have different labels; different trees have the same label set [8, 11–13]. Some methods require that the trees are ordered [6].

In existing methods, the bottom-up distance [9] and the subtree distance [10] could work on rooted unordered trees with possibly repeated labels. The bottom-up distance between two trees T_1, T_2 is defined as $\mathrm{D}_{\mathrm{BU}}(T_1, T_2) = 1 - f / \max(n_1, n_2)$, where n_1, n_2 are the tree sizes, and f is the size of the largest common forest of two trees. The subtree distance $\mathrm{D}_{\mathrm{ST}}(T_1, T_2)$ is defined almost the same as the bottom-up distance, except that f is the size of the largest common subtree of two trees. Both distances could be calculated in linear time [9, 14]. These two methods have some disadvantages. For example, they are not robust under small perturbations on labels, and they do not compare non-common structures. See the next section for detailed discussions.

We develop two new metrics that apply for rooted unordered trees with possibly repeated labels: the best-match metric D_{BM} and the left-regular metric D_{LR}. For two unordered trees, the best-match metric searches all their planar embeddings, and compares the most similar pair. To calculate the left-regular metric for two unordered trees, we apply a procedure to fix one planar embedding for each unordered tree (its “regular form”), and compare the regular forms of these two unordered trees. These two metrics take into account different similarities between labels and different weights concerning their positions. These two metrics, especially the best-match metric, consider any common structures and compare non-common structures. To compute the best-match distance between two trees (binary or general k-ary), the expected time complexity and the worst-case time complexity are both $\mathcal{O}(n^2)$, where n is the tree size. To compute the left-regular distance between two trees (binary or not), the expected time complexity is $\mathcal{O}(n)$, and the worst-case time complexity is $\mathcal{O}(n \log n)$.

The above discussions are for unordered trees, where all vertices are unordered. In some cases, we can distinguish two children cells, so that certain vertices are ordered. Then the space we need to consider consists of rooted trees with possibly repeated labels, where vertices can be ordered or unordered. This larger space has complicated structures that do not allow the existence of a proper metric. Existing methods and the left-regular metric introduced in this paper are not applicable. Nevertheless, the best-match metric can be slightly modified to become a semimetric that works in this scenario.

The main text consists of the following contents: compare existing methods and our new methods; introduce related terminologies in graph theory; define two metrics on the space of rooted unordered trees with possibly repeated labels; define a semimetric on the space of rooted trees with possibly repeated labels, where vertices can be ordered or unordered.

Comparison of existing methods and new methods

In this section, we compare the performance of existing methods and new methods on rooted unordered trees with possibly repeated labels, so as to explain the motivation to develop new methods. The examples used are illustrated in Figs. 6, 7 and 8. See Table 1 for a summary of these comparisons.

Compared to the left-regular metric D_{LR}, especially to the best-match metric D_{BM} introduced in this paper, the bottom-up distance D_{BU} [9] and the subtree distance D_{ST} [10] have some disadvantages.

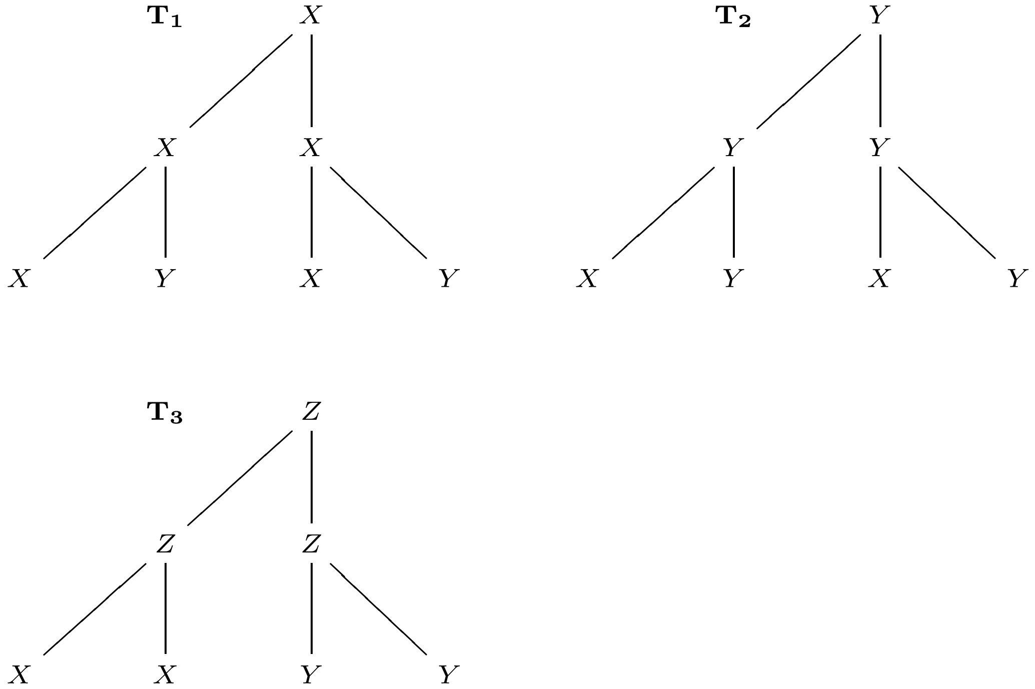

Fig. 6 Three trees T_1, T_2, T_3, used to compare the bottom-up distance D_{BU}, the subtree distance D_{ST}, the best-match metric D_{BM}, and the left-regular metric D_{LR}

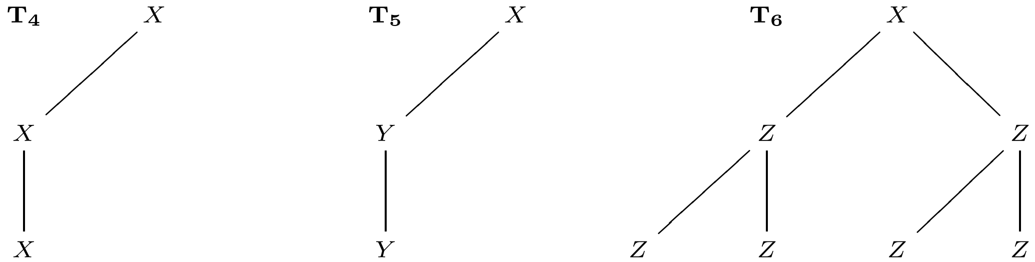

Fig. 7 Three trees T_4, T_5, T_6, used to compare the bottom-up distance D_{BU}, the subtree distance D_{ST}, the best-match metric D_{BM}, and the left-regular metric D_{LR}

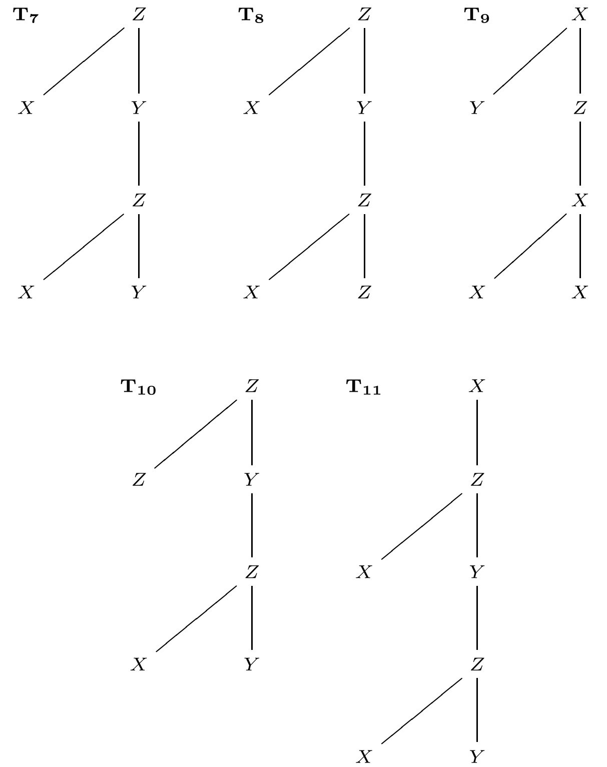

Fig. 8 Five trees $T_7, T_8, T_9, T_{10}, T_{11}$, used to compare the bottom-up distance $\mathrm{D_{BU}}$, the subtree distance $\mathrm{D_{ST}}$, the best-match metric $\mathrm{D_{BM}}$, and the left-regular metric $\mathrm{D_{LR}}$

In Fig. 6, T_1, T_2 have the same distribution of leaves labels, while T_1, T_3 have different distributions of leaves labels. However, $\mathrm{D_{BU}}(T_1, T_2) = \mathrm{D_{BU}}(T_1, T_3) = 3/7$, $\mathrm{D_{ST}}(T_1, T_2) = \mathrm{D_{ST}}(T_1, T_3) = 6/7$. The reason is that $\mathrm{D_{BU}}$ and $\mathrm{D_{ST}}$ only consider common structures, but not their detailed patterns. $\mathrm{D_{BM}}$ and $\mathrm{D_{LR}}$ can recognize the difference: $\mathrm{D_{BM}}(T_1, T_2) = 3$, $\mathrm{D_{BM}}(T_1, T_3) = 5$; $\mathrm{D_{LR}}(T_1, T_2) = 3, \mathrm{D_{LR}}(T_1, T_3) = 5$.

In Fig. 7, T_4, T_5 have the same tree topology, while T_4, T_6 have different tree topologies. However, $\mathrm{D_{BU}}(T_4, T_5) = \mathrm{D_{BU}}(T_4, T_6) = 1$, $\mathrm{D_{ST}}(T_4, T_6) = \mathrm{D_{ST}}(T_4, T_6) = 1$. The reason is that $\mathrm{D_{BU}}$ and $\mathrm{D_{ST}}$ do not compare non-common structures. $\mathrm{D_{BM}}$ and $\mathrm{D_{LR}}$ can recognize the difference: $\mathrm{D_{BM}}(T_4, T_5) = 2$, $\mathrm{D_{BM}}(T_4, T_6) = 6$; $\mathrm{D_{LR}}(T_4, T_5) = 2$, $\mathrm{D_{LR}}(T_4, T_6) = 6$.

In Fig. 8, T_7, T_8 only differ by a leaf label, while T_7, T_9 are much more different. However, $\mathrm{D_{BU}}(T_7, T_8) = 2/3 > 1/2 = \mathrm{D_{BU}}(T_7, T_9)$, $\mathrm{D_{ST}}(T_7, T_8) = \mathrm{D_{ST}}(T_7, T_9) = 5/6$. The reason is that $\mathrm{D_{BU}}$ and $\mathrm{D_{ST}}$ only consider certain common structures (sub-forest and subtree). $\mathrm{D_{BM}}$ and $\mathrm{D_{LR}}$ consider any common structures and recognize that T_7, T_8 are more similar: $\mathrm{D_{BM}}(T_7, T_8) = 1$, $\mathrm{D_{BM}}(T_7, T_9) = 5$; $\mathrm{D_{LR}}(T_7, T_8) = 1, \mathrm{D_{LR}}(T_7, T_9) = 5$.

Besides, for two vertices with different labels, $\mathrm{D_{BU}}$ and $\mathrm{D_{ST}}$ only know they are different, but not concerning how different they are. In reality, such as in comparing developmental trees, some labels are very different, while some

Table 1 Summary of the comparisons in the "Comparison of existing methods and new methods"

	$\mathrm{D_{BM}}$	$\mathrm{D_{LR}}$	Size of largest common forest	$\mathrm{D_{BU}}$	Size of largest common subtree	$\mathrm{D_{ST}}$
T_1 and T_2	3	3	4	3/7	1	6/7
T_1 and T_3	5	5	4	3/7	1	6/7
T_2 and T_3	5	5	4	3/7	1	6/7
T_4 and T_5	2	2	0	1	0	1
T_4 and T_6	6	6	0	1	0	1
T_5 and T_6	6	6	0	1	0	1
T_7 and T_8	1	1	2	2/3	1	5/6
T_7 and T_9	5	5	3	1/2	1	5/6
T_7 and T_{10}	1	8	4	1/3	4	1/3
T_7 and T_{11}	9	9	6	1/7	6	1/7
T_8 and T_9	5	5	2	2/3	1	5/6
T_8 and T_{10}	2	8	2	2/3	1	5/6
T_8 and T_{11}	8	8	2	5/7	1	6/7
T_9 and T_{10}	5	7	2	2/3	1	5/6
T_9 and T_{11}	7	7	3	4/7	1	6/7
T_{10} and T_{11}	9	10	4	3/7	4	3/7

Performance of $\mathrm{D_{BM}}$, $\mathrm{D_{LR}}$, $\mathrm{D_{BU}}$, and $\mathrm{D_{ST}}$ on trees in Figs. 6, 7 and 8 are illustrated

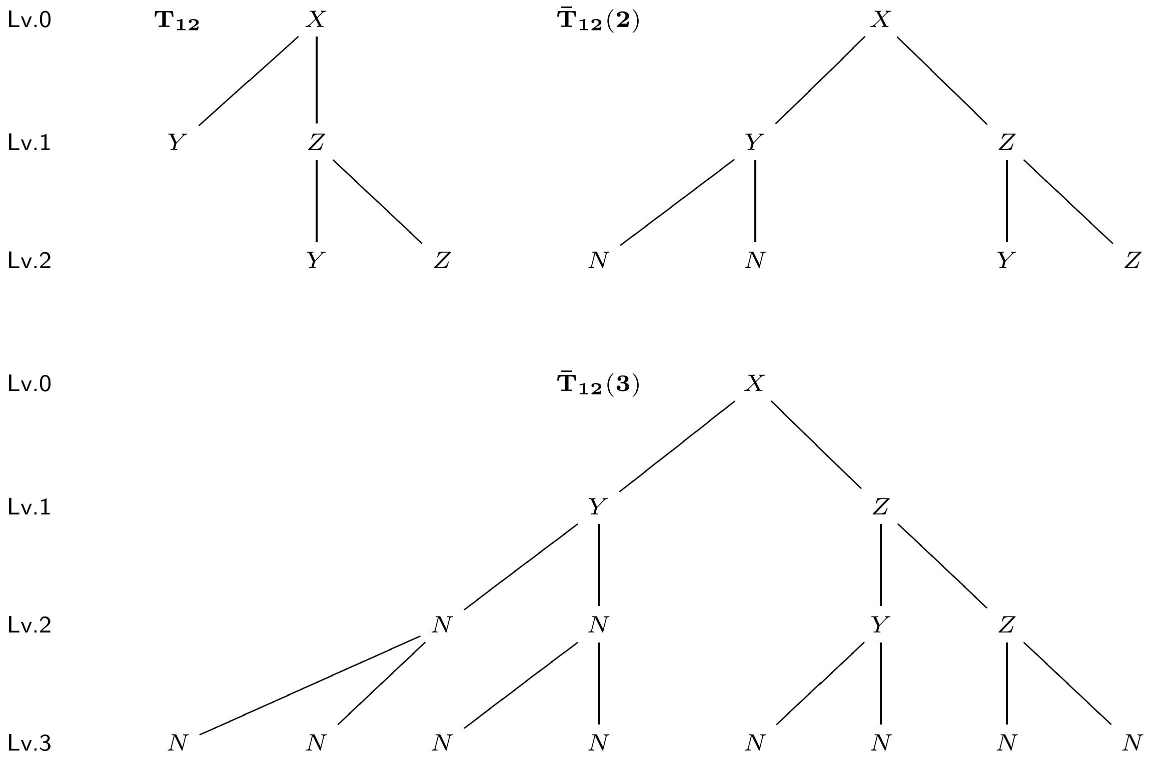
Lv.0
Lv.1
Lv.2
Lv.3
$\mathbf{T_{12}}$
$\bar{\mathbf{T}}_{\mathbf{12}}(\mathbf{2})$
$\bar{\mathbf{T}}_{\mathbf{12}}(\mathbf{3})$
X
Y
Z
N

Fig. 9 A 2-level tree T_{12} (upper left), its level-2 completion $\bar{T}_{12}(2)$ (upper right) and its level-3 completion $\bar{T}_{12}(3)$ (lower). The level (Lv.) of each vertex is marked on the left

labels are rather similar. The position of vertices can also be concerned. In general, a label difference closer to the root should be more crucial. In D_{BM} and D_{LR}, different distances between labels and different weights on vertices can be introduced naturally.

The above discussion explains our motivation to develop the best-match metric D_{BM} and the left-regular metric D_{LR}. However, D_{BM} and D_{LR} also have disadvantages.

In Fig. 8, T_7, T_{10} only differ by a leaf label. In this case, $D_{BU}(T_7, T_{10}) = 1/3$, $D_{ST}(T_7, T_{10}) = 1/3$, $D_{BM}(T_7, T_{10}) = 1$, but $D_{LR}(T_7, T_{10}) = 8$. The reason is that D_{LR} is not always robust under small perturbations on labels, similar to D_{BU} and D_{ST}. D_{BM} is robust under small perturbations on labels.

In Fig. 8, inserting one vertex to T_7 produces T_{11}. In this case, $D_{BU}(T_7, T_{11}) = 1/7$, $D_{ST}(T_7, T_{11}) = 1/7$, but $D_{BM}(T_7, T_{11}) = 9$, $D_{LR}(T_7, T_{11}) = 9$. The reason is that D_{BM} and D_{LR} are not robust under small perturbations on the tree topology, especially perturbations near the roots. D_{BU} and D_{ST} are more robust to the change of tree topology near the roots.

In summary, our methods outperform the existing methods in most cases. In general, we recommend the best-match metric D_{BM}. If time cost is a major concern, the left-regular metric D_{LR} can be applied.

Definitions and notations

Trees

In graph theory, a rooted tree is a connected acyclic undirected graph, where one vertex v_0 is designated as the root. Some vertices are linked by edges. For each vertex v_i, there is a unique path (edge sequence) that connects v_i and the root v_0. The number of edges in this path is called the depth of v_i. The depth of the root v_0 is stipulated as 0. The depth of a tree is the largest depth of its vertices. The kth level (or level k) of a tree consists of all vertices whose depths are k. If the depth of a tree is m, it is also called an m-level tree. If there is an edge between two vertices v_i, v_j, and the depth of v_i is smaller than the depth of v_j, then v_i is the parent vertex of v_j, and v_j is a child vertex of v_i. For v_i and its child vertex v_j, the tree with root v_j is called a subtree of v_i. A vertex without children vertices is called a leaf vertex [15].

In this paper, each vertex has a label, and different vertices might have the same label. The set of possible labels $\mathcal{L}$ can have infinite elements or even uncountable elements. In the following, we use $\mathcal{L} = \{X, Y, Z\}$ as an example.

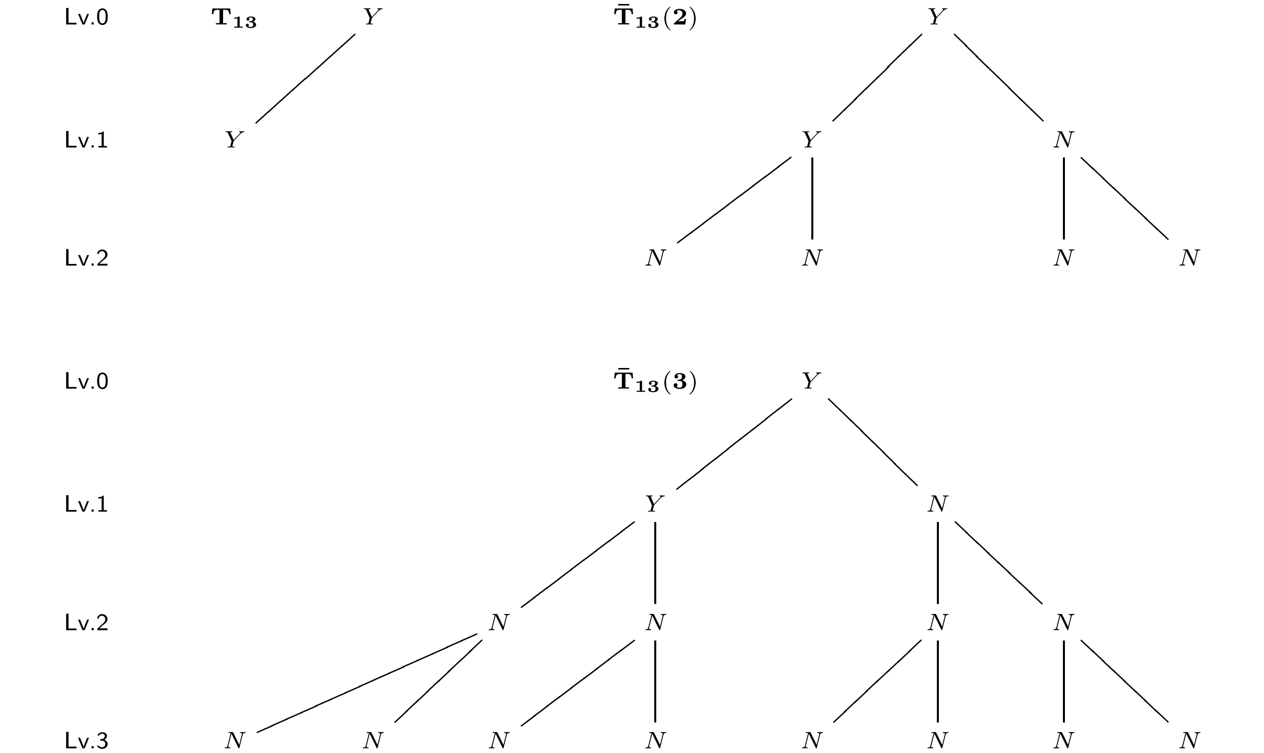
Lv.0
Lv.1
Lv.2
Lv.0
Lv.1
Lv.2
Lv.3
$\mathbf{T_{13}}$
$\bar{\mathbf{T}}_{\mathbf{13}}(\mathbf{2})$
$\bar{\mathbf{T}}_{\mathbf{13}}(\mathbf{3})$
Y
N

Fig. 10 A 1-level tree T_{13} (upper left), its level-2 completion $\bar{T}_{13}(2)$ (upper right) and its level-3 completion $\bar{T}_{13}(3)$ (lower). The level (Lv.) of each vertex is marked on the left

For simplicity, we only consider binary trees, meaning that each vertex has at most two children vertices. However, the methods in this paper also work for general k-ary trees.

For an l-level tree T and any $m \geq l$, we construct its **level-*m*** completion $\bar{T}(m)$ as the following: For a vertex not in level m, if it has less than two children vertices, add children vertices to it until it has two. Newly added vertices have the label “N” (means “null”). Repeat this procedure, until every vertex not in level m has two children vertices, and every vertex in level m has no children vertices. In other words, we construct a perfect binary m-level tree. See Figs. 9 and 10 for two trees and their completions with different levels.

For trees after completion, the label set is $\bar{\mathcal{L}} = \mathcal{L} \cup \{N\}$, which is $\{X, Y, Z, N\}$ in our examples. For now, we just require that there is a metric d on $\bar{\mathcal{L}}$. In this paper, for simplicity, we shall apply the trivial metric that different labels always have distance 1. Later, we will also need a total order on $\bar{\mathcal{L}}$.

A vertex is called ordered if in the planar embedding of this tree, we know which of its child vertex is the left child, and which is the right child. Otherwise, it is called unordered, and we can switch its two subtrees in the planar embedding. A tree is ordered if all its vertices are ordered. A tree is unordered if all its vertices are unordered.

Each ordered tree corresponds to a unique planar embedding. In the following, we do not distinguish an ordered tree and its planar embedding. For the space of rooted ordered trees with possibly repeated labels, we define that two trees are equivalent if one tree can transform into the other tree by switching subtrees of some vertices (labels are also switched along with the vertices). Here after transformations, two trees have the same tree topology, and corresponding vertices have the same label. The notation $T_1 \sim T_2$ means T_1, T_2 are equivalent, and $T_1 \not\sim T_2$ means T_1, T_2 are not equivalent. With this equivalence relationship, the space of ordered trees is divided into different equivalent classes. See Fig. 5 for an equivalent class of ordered trees, where four ordered trees are equivalent.

An unordered tree corresponds to different planar embeddings (ordered trees). Since we can switch two subtrees of an unordered vertex, equivalent ordered trees represent the same unordered tree. Besides, non-equivalent ordered trees represent different unordered trees. Therefore, the space of unordered trees is isomorphic to the space of equivalent classes of ordered trees. The four ordered trees in Fig. 5 represent the same unordered tree.

Metrics

To define a metric on unordered trees, we can switch to equivalent classes of ordered trees. A metric D on the space of equivalent classes of ordered trees maps a pair of such trees to a non-negative real number, and it satisfies the following criteria for any trees T_1, T_2, T_3:

(A1) $\mathrm{D}(T_1, T_2) = \mathrm{D}(T_2, T_1)$;
(A2) $\mathrm{D}(T_1, T_2) \geq 0$, and $\mathrm{D}(T_1, T_2) = 0$ if and only if $T_1 \sim T_2$;
(A3) $\mathrm{D}(T_1, T_2) + \mathrm{D}(T_1, T_3) \geq \mathrm{D}(T_2, T_3)$.

A metric that satisfies (A1)–(A3) also has another property: if $T_1 \sim T_2$, then $\mathrm{D}(T_1, T_3) = \mathrm{D}(T_2, T_3)$.

Before introducing metrics on unordered trees, we first need a metric on the space of ordered trees (not equivalent classes). For two ordered trees T_1 and T_2, consider their level-m completions, where m is no less than the depths of T_1 and T_2. For these two completed m-level trees $\bar{T}_1(m)$, $\bar{T}_2(m)$ with the same tree topology, there is a bijection between vertices. We define the ordered tree metric $\mathrm{D}_{\mathrm{OT}}(T_1, T_2)$ for such completed ordered trees:

$$\mathrm{D}_{\mathrm{OT}}(T_1, T_2) = \mathrm{D}_{\mathrm{OT}}(\bar{T}_1(m), \bar{T}_2(m)) = \sum_{i \in \bar{T}_1(m)} c(i)\mathrm{d}(i, i'),$$

where $i' \in \bar{T}_2(m)$ is the corresponding vertex of i, d is the metric on the label set $\bar{\mathcal{L}}$, and $c(i)$ is the weight coefficient that depends on the depth of i. In some scenarios, we want to emphasize the differences closer to the root (correspond to earlier developmental stages), meaning that we can assign a larger value to $c(i)$ with smaller depth of i. For simplicity, we use $c(i) = 1$ for all vertices in this paper. We can see that the value of D_{OT} does not depend on the choice of m. For tree T_{12} in Fig. 9 and tree T_{13} in Fig. 10, their D_{OT} distance is

$$\begin{aligned}\mathrm{D}_{\mathrm{OT}}(T_{12}, T_{13}) &= \mathrm{D}_{\mathrm{OT}}(\bar{T}_{12}(2), \bar{T}_{13}(2)) \\ &= \mathrm{D}_{\mathrm{OT}}(\bar{T}_{12}(3), \bar{T}_{13}(3)) = 4,\end{aligned}$$

since they have 4 pairs of corresponding vertices with different labels. In the rest of this paper, we always consider trees after completion of proper levels. Therefore, the number of vertices (tree size) n and the depth m satisfies $n = 2^{m+1} - 1$.

Best-match metric on unordered trees

Definition

We start to define metrics on the space $\mathcal{T}$ of unordered trees, namely the equivalent classes of ordered trees. For two ordered trees T_1, T_2 (representing their equivalent classes), we can check all pairs of ordered trees that one is equivalent with T_1, the other is equivalent with T_2, and choose the best-match pair with the minimal D_{OT} distance. We define D_{BM} on equivalent classes of ordered trees:

$$\mathrm{D}_{\mathrm{BM}}(T_1, T_2) = \min_{T_1 \sim T_1', T_2 \sim T_2'} \mathrm{D}_{\mathrm{OT}}(T_1', T_2').$$

This $\mathrm{D}_{\mathrm{BM}}(T_1, T_2)$ satisfies the criteria (A1)-(A3) for a metric, defined in the previous section. We name D_{BM} the best-match metric. For the tree T_{12} in Fig. 9 and the tree T_{13} in Fig. 10, $\mathrm{D}_{\mathrm{BM}}(T_{12}, T_{13}) = 4$.

From the definition of the best-match metric $\mathrm{D}_{\mathrm{BM}}(T_1, T_2)$, we can see that changing one label of T_1 will make $\mathrm{D}_{\mathrm{BM}}(T_1, T_2)$ change by at most 1. Therefore, the best-match metric is robust under small perturbations on labels. This property does not hold for the left-regular metric, the bottom-up distance, and the subtree distance.

A dynamic programming implementation

There are exponentially many trees being equivalent to a given tree. Thus brute-force searching is too expensive. Here we introduce a dynamic programming algorithm [16] for calculating the best-match metric $\mathrm{D}_{\mathrm{BM}}(T_a, T_b)$.

```
1   Input
        Two rooted unordered trees $T_a, T_b$ with possibly repeated labels
        A metric $\mathrm{d}$ on the label set
2   Replace $T_a, T_b$ by the level-$m$ completion of $T_a, T_b$
        % Here $m$ is no less than the depths of $T_a$ and $T_b$
3   Define a function $\mathrm{BM}(l, T_1, T_2)$
        % Here the input is two trees $T_1, T_2$ after completion with depth $l$,
        % and the output is $\mathrm{D_{BM}}(T_1, T_2)$
        If $l = 0$
            Return $\mathrm{d}(T_1, T_2)$
        Else
            Define $T_1^0$ to be $T_1$'s root, and $T_2^0$ to be $T_2$'s root
            Define $T_{1L}$ to be the left subtree of $T_1^0$, and $T_{1R}$ to be the right subtree of $T_1^0$
            Define $T_{2L}$ to be the left subtree of $T_2^0$, and $T_{2R}$ to be the right subtree of $T_2^0$
            Calculate $C_1 = \mathrm{BM}(l-1, T_{1L}, T_{2L}) + \mathrm{BM}(l-1, T_{1R}, T_{2R})$
            Calculate $C_2 = \mathrm{BM}(l-1, T_{1L}, T_{2R}) + \mathrm{BM}(l-1, T_{1R}, T_{2L})$
            Return $\mathrm{d}(T_1^0, T_2^0) + \min\{C_1, C_2\}$
            % Define $\mathrm{BM}(l, T_1, T_2)$ recursively
        End of if
4   Output $\mathrm{BM}(m, T_a, T_b)$, which is the best-match metric $\mathrm{D_{BM}}(T_a, T_b)$
```

Algorithm 1: Detailed workflow of calculating the best-match metric $\mathrm{D_{BM}}$.

See Algorithm 1 for the workflow of calculating the best-match metric $\mathrm{D_{BM}}$. The idea is simple: For the root, we only need to determine whether the left and right subtrees should be switched. In either case, the problem is reduced to minimizing the distance between subtrees. In other words, the vertex correspondence that minimizes the distance between two trees also minimizes the distance between two subtrees.

In the appendix, we illustrate the detailed procedure of calculating $\mathrm{D_{BM}}$ for the developmental trees of *Arabidopsis thaliana* and sea urchin. $\mathrm{D_{BM}}$ is also applied to other developmental trees with tree size ~ 100, and it is discovered that species with similar developmental trees (i.e., smaller $\mathrm{D_{BM}}$) are more likely to have the same anatomical traits [17]. For more examples, see Figs. 6, 7, 8, and Table 1. The Python code for calculating $\mathrm{D_{BM}}$ can be found online (https://github.com/YueWangMathbio/TreeMetric, DOI: https://doi.org/10.5281/zenodo.6400267).

Computational complexity

Assume we need $g(m)$ steps to calculate the best-match distance between two m-level trees. Then we have $g(0) = 1$, $g(m+1) = 4g(m) + 4$. Thus $g(m) = 4^{m+1} \times 7/12 - 4/3$. The number of vertices is $n = 2^{m+1} - 1$, thus the time complexity of computing the best-match metric is $\mathcal{O}(n^2)$. Here the worst-case time complexity and the expected time complexity are equal. The space complexity of computing the best-match metric is trivially $\mathcal{O}(n)$. When the trees are not binary, but k-ary, we have $g(m+1) = k^2 g(m) + k \cdot k!$. Here the $k^2 g(m)$ term means that there are k^2 pairs of subtrees to compare. The $k \cdot k!$ term means that for $k!$ possible subtree correspondences, we need $(k-1) \cdot k!$ steps to compute the sum of distances, $k! - 1$ steps to compare them, and 1 step to add $\mathrm{d}(T_1^0, T_2^0)$. Since $g(0) = 1$, we have $g(m) = [1/k^2 + (k-1)!/(k^2-1)](k^2)^{m+1} - (k \cdot k!)/(k^2-1)$. With $n = (k^{m+1} - 1)/(k-1)$, the time complexity is still $\mathcal{O}(n^2)$.

Left-regular metric on unordered trees

Preparation

Since the metric is defined on the equivalent classes of ordered trees, we need to guarantee that equivalent trees have the same behavior, namely $\mathrm{D}(T_1, T_3) = \mathrm{D}(T_2, T_3)$ for $T_1 \sim T_2$. One idea is to transform a given tree into some "regular form", which is unique to each equivalent class.

We define a total order on the label set $\bar{\mathcal{L}}$, such as $N > X > Y > Z$. Ideally, similar labels should be closer. With this total order on the label set (alphabet), there is an induced total order, namely the lexicographic order [18], for strings of labels with the same length: for two strings, compare the corresponding labels from the beginning, until there is a difference, and apply the total order for labels. For example, $XZN < XNY$, since $X = X$, and $Z < N$. For a tree after completion, we can write its labels as a string, in the order of up-down (root-leaf), left-right. This is named its label string. For example, the label string of $\bar{T}_{12}(2)$ in Fig. 11 is *XYZNNYZ*. We can reconstruct a tree from its label string.

Now we describe the procedure of left-regularization, through which a tree is transformed into its "regular form". Consider a tree T after level-l completion. For each vertex in level $l - 1$, if the label string of its left subtree is larger than the label string of its right subtree, switch its left and right subtrees. This procedure is called "left-regularization". After the left-regularization of level $l - 1$, repeat this procedure for level $l - 2, l - 3, \cdots, 1, 0$. When the procedure is finished, we obtain the fully "left-regularized" form of T. The procedure of left-regularization for the tree $\bar{T}_{12}(2)$ is shown in Fig. 11.

In a fully left-regularized tree, for each vertex, the label string of its left subtree is no larger than that of its right subtree. Thus each subtree is also fully left-regularized. By induction with the tree depth, we can see that two equivalent ordered trees have the same left-regularization. Two trees with the same left-regularization are obviously equivalent. Therefore, two ordered trees are equivalent if and only if their left-regularizations are the same. With this procedure, each unordered tree (or its corresponding equivalent class of ordered trees) corresponds to a unique left-regularized ordered tree.

Definition and properties

For a tree T, denote its fully left-regularized form as $\tilde{T}$. Now we can define D_{LR} on unordered trees:

$$\mathrm{D}_{\mathrm{LR}}(T_1, T_2) = \mathrm{D}_{\mathrm{OT}}(\tilde{T}_1, \tilde{T}_2).$$

This D_{LR} satisfies the criteria (A1)-(A3) for a metric, and we name it the left-regular metric. Notice that the choice of total order on $\bar{\mathcal{L}}$ might affect the value of $\mathrm{D}_{\mathrm{LR}}(T_1, T_2)$. See Algorithm 2 for the workflow of calculating the left-regular metric D_{LR}. The definition of the left-regular metric already implies how to calculate it.

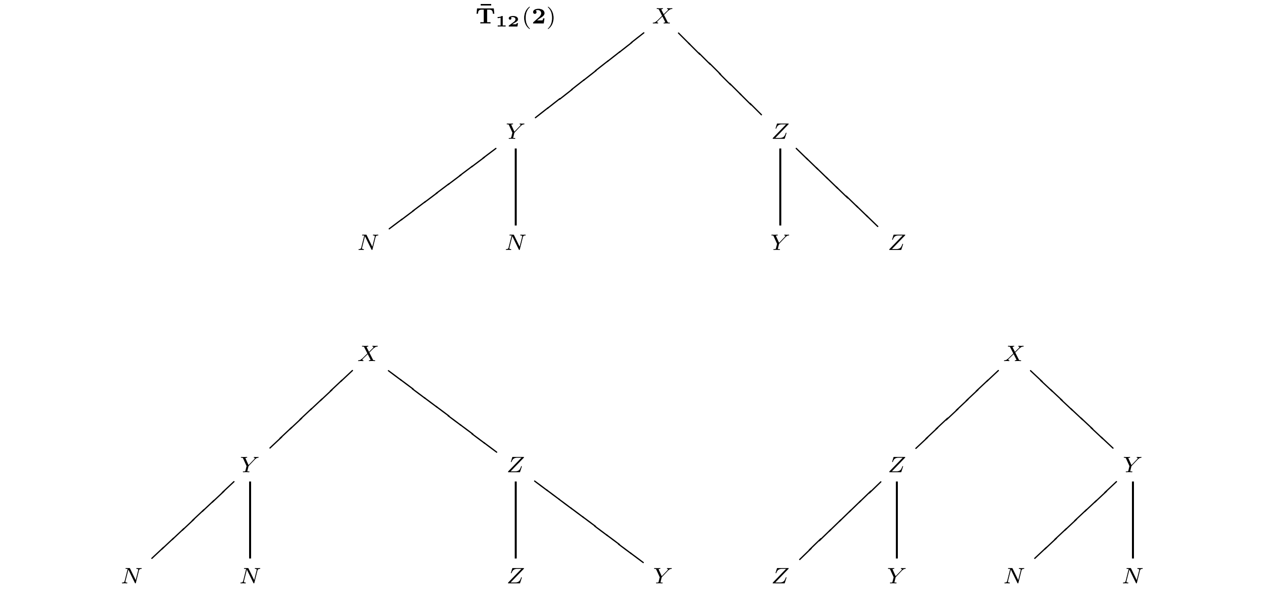

Fig. 11 Procedure of left-regularization. A 2-level tree $\bar{T}_{12}(2)$ (upper). After left-regularization on level 1 (lower left). Fully left-regularized (lower right)

```
1 Input
    Two rooted unordered trees $T_a, T_b$ with possibly repeated labels
    A function that calculates the ordered tree metric $\mathrm{D_{OT}}$
    A total order on the label set
2 Replace $T_a, T_b$ by the level-$m$ completion of $T_a, T_b$
    % Here $m$ is no less than the depths of $T_a$ and $T_b$
3 For $l$ from $m-1$ to $0$
    For each vertex $v$ in level $l$ of $T_a$,
      Set $S_\mathrm{L}(v)$ to be the label string of $v$'s left subtree
      Set $S_\mathrm{R}(v)$ to be the label string of $v$'s right subtree
      If $S_\mathrm{L}(v) > S_\mathrm{R}(v)$
        Switch left and right subtrees of $v$
      End of if
    End of for loop
  End of for loop
  Denote $T_a$ after this left-regularization by $\tilde{T}_a$
4 Repeat Step 3 for $T_b$ to obtain $\tilde{T}_b$
5 Output $\mathrm{D_{OT}}(\tilde{T}_a, \tilde{T}_b)$, which is the left-regular metric $\mathrm{D_{LR}}(T_a, T_b)$
```

Algorithm 2: Detailed workflow of calculating the left-regular metric $\mathrm{D_{LR}}$.

The tree $\bar{T}_{13}(2)$ in Fig. 10 is already left-regularized. Thus we can compare it with $\bar{T}_{12}(2)$ after left-regularization in Fig. 11 to find $\mathrm{D_{LR}}(T_{12}, T_{13}) = 5$. In the appendix, we illustrate the detailed procedure of calculating $\mathrm{D_{LR}}$ for the developmental trees of *Arabidopsis thaliana* and sea urchin. For more examples, see Fig. 6, 7, 8, and Table 1. The Python code for calculating $\mathrm{D_{LR}}$ can be found online (https://github.com/YueWangMathbio/TreeMetric, DOI: https://doi.org/10.5281/zenodo.6400267).

Consider an m-level tree. For each vertex in level l, to compare the label strings of its subtrees, we need at most 2^{m-l} steps. Therefore, the left-regularization on each level needs at most 2^m steps, and the total number of steps is no more than $(m+1)2^m$. Thus the worst-case time complexity of computing the left-regular metric is $\mathcal{O}(n \log n)$, where n is the vertex number. The space complexity of computing the left-regular metric is trivially $\mathcal{O}(n)$. If the trees are randomly generated, then the expectation of steps needed to compare two label strings is bounded by a constant C, regardless of string length. Thus the expected total number of steps is no more than $C2^{m+1}$, and the expected time complexity is $\mathcal{O}(n)$. When the trees are not binary, but k-ary, the orders of the worst-case time complexity and the expected time complexity are not changed.

Both the best-match metric and the left-regular metric transform two trees by switching subtrees, and compare the trees after transformation. The best-match metric switches subtrees for two trees cooperatively, so as to find the pair that has the minimal $\mathrm{D_{OT}}$ distance. The left-regular metric just switches subtrees independently, and the final pair might not be the best match. Thus we can see that for any two unordered trees T_1, T_2, $\mathrm{D_{LR}}(T_1, T_2) \geq \mathrm{D_{BM}}(T_1, T_2)$. Thus $\mathrm{D_{LR}}$ is an upper bound of $\mathrm{D_{BM}}$.

A semimetric on trees with ordered and unordered vertices

In some situations, for a developmental tree, we know the order of children cells for some cells, but not other cells. Therefore, in this section, we consider the space $\mathcal{P}$ of rooted trees with possibly repeated labels, where vertices can be ordered or unordered. This space contains all ordered trees and unordered trees. To represent this space, we consider ordered trees, where some non-leaves vertices have "lock marks", denoted by circles surrounding the labels. See Fig. 12 for some examples. For trees T_1^*, T_2^* possibly with lock marks, if we can switch subtrees of some vertices without lock marks in T_1^*, so that T_1^* is transformed into T_2^* (comparing both labels and lock marks), then we define that T_1^*, T_2^* are equivalent, and denote it by $T_1^* \sim T_2^*$. Under this equivalence relation, the space of ordered trees with lock marks is divided into different equivalent classes, and the space of such equivalent classes is isomorphic to the space of trees where vertices can be ordered or unordered. If a tree T^* with lock marks belongs to an equivalent class that represents T, then a vertex in T^* has a lock mark if and only if the corresponding vertex in T is ordered. We will define a distance on the equivalent classes of ordered trees with lock marks (not a metric, but a semimetric).

If we try to define a metric on the space of such equivalent classes, we shall meet a problem. Consider two ordered trees T_1^*, T_2^* with the same tree topology and labels, but different lock marks, then $T_1^* \not\sim T_2^*$. If we have a metric D, then $\mathrm{D}(T_1^*, T_2^*) > 0$. However, T_1^*, T_2^* have the same labels for corresponding vertices, and we argue that their distance should be 0. Due to this reason, we need to define another relation between ordered trees with lock marks.

For an ordered tree T^* that could have lock marks, and a normal ordered tree T without lock marks, if we can switch subtrees of some vertices without lock marks in T^*, so that T^* is transformed into T (only comparing labels, but not lock marks), then we say $T^* \to T$. For two ordered trees T_1^*, T_2^* that possibly have lock marks, if there exists an ordered tree T without lock marks, so that $T_1^* \to T$ and $T_2^* \to T$, then we define that T_1^*, T_2^* are semi-equivalent, denoted by $T_1^* \approx T_2^*$. $T_1^* \not\approx T_2^*$ means T_1^*, T_2^* are not semi-equivalent. The semi-equivalence relation is reflexive ($T^* \approx T^*$) and symmetric ($T_1^* \approx T_2^*$ means $T_2^* \approx T_1^*$), but not transitive: in Fig. 12, $T_{14}^* \approx T_{15}^*$, $T_{14}^* \approx T_{16}^*$, but $T_{15}^* \not\approx T_{16}^*$. Thus semi-equivalence is not an equivalence relation, and the space of ordered trees with lock marks cannot be divided into different equivalent classes by this relation. Equivalence is stronger than semi-equivalence: $T_1^* \sim T_2^*$ implies $T_1^* \approx T_2^*$, but not vice versa. Besides, if $T_1^* \sim T_3^*$, $T_2^* \sim T_4^*$, $T_1^* \approx T_2^*$, then $T_3^* \approx T_4^*$.

If $T_1^* \approx T_2^*$, then after certain transformations, they have the same labels, and their distance should be 0. If $T_1^* \not\approx T_2^*$, then they are essentially different, and their distance should be positive. Therefore, to define a distance D on the space of ordered trees that could have lock marks, we need to satisfy the following criteria for any trees T_1^*, T_2^*, T_3^*:

(B1) $\mathrm{D}(T_1^*, T_2^*) = \mathrm{D}(T_2^*, T_1^*)$;

(B2) $\mathrm{D}(T_1^*, T_2^*) \geq 0$, and $\mathrm{D}(T_1^*, T_2^*) = 0$ if and only if $T_1^* \approx T_2^*$.

However, the triangular inequality does not hold. Consider $T_{14}^*, T_{15}^*, T_{16}^*$ in Fig. 12. Since $T_{14}^* \approx T_{15}^*$, $T_{14}^* \approx T_{16}^*$, $T_{15}^* \not\approx T_{16}^*$, we have $\mathrm{D}(T_{14}^*, T_{15}^*) + \mathrm{D}(T_{14}^*, T_{16}^*) = 0 < \mathrm{D}(T_{15}^*, T_{16}^*)$. Therefore, we cannot define a metric with respect to the semi-equivalence relation. Besides, $T_1^* \approx T_2^*$ does not imply $\mathrm{D}(T_1^*, T_3^*) = \mathrm{D}(T_2^*, T_3^*)$. Nevertheless, we could require that

(B3): $T_1^* \sim T_2^*$ implies $\mathrm{D}(T_1^*, T_3^*) = \mathrm{D}(T_2^*, T_3^*)$.

We seek a distance that satisfies (B1)-(B3), which is a semimetric.

For the space of ordered trees that could have lock marks, existing methods and the left-regular metric are not applicable with respect to the semi-equivalence relation. Nevertheless, the best-match metric can be slightly modified to work in this scenario. On the space of ordered trees that could have lock marks, $\mathrm{D}_{\mathrm{BM}}^*$ is defined as

$$\mathrm{D}_{\mathrm{BM}}^*(T_1^*, T_2^*) = \min_{T_1^* \to T_1, T_2^* \to T_2} \mathrm{D}_{\mathrm{OT}}(T_1, T_2).$$

This $\mathrm{D}_{\mathrm{BM}}^*(T_1^*, T_2^*)$ satisfies the criteria (B1)-(B3) for a semimetric, and we name it the best-match semimetric. For the trees in Fig. 12, $\mathrm{D}_{\mathrm{BM}}^*(T_{14}^*, T_{15}^*) = 0$, $\mathrm{D}_{\mathrm{BM}}^*(T_{14}^*, T_{16}^*) = 0$, $\mathrm{D}_{\mathrm{BM}}^*(T_{15}^*, T_{16}^*) = 6$. See Algorithm 3 for the workflow of calculating the best-match semimetric $\mathrm{D}_{\mathrm{BM}}^*$. It is only slightly different from Algorithm 1 that calculates D_{BM}.

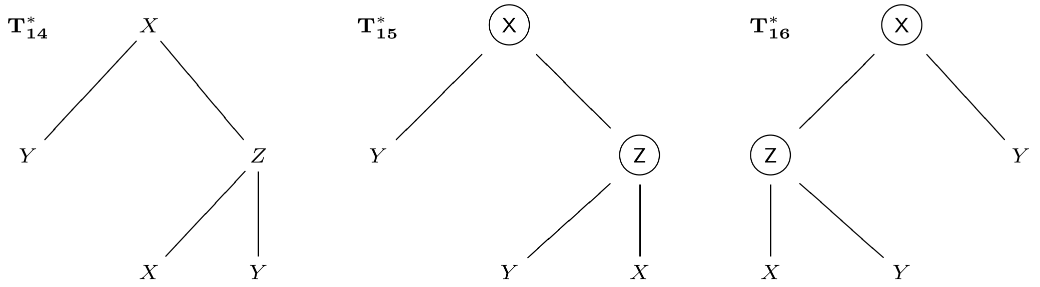

Fig. 12 Three ordered trees possibly with lock marks (circles): $T^*_{14} \approx T^*_{15}$, $T^*_{14} \approx T^*_{16}$, $T^*_{15} \not\approx T^*_{16}$

```
1   Input
      Two rooted trees $T^*_a, T^*_b$ with possibly repeated labels,
      where some vertices have lock marks
      A metric $\mathrm{d}$ on the label set
2   Replace $T^*_a, T^*_b$ by the level-$m$ completion of $T^*_a, T^*_b$
      % Here $m$ is no less than the depths of $T^*_a$ and $T^*_b$
3   Define a function $\mathrm{BM}(l, T^*_1, T^*_2)$
      % Here the input is two trees $T^*_1, T^*_2$ after completion with depth $l$,
      % and the output is $\mathrm{D}^*_{\mathrm{BM}}(T^*_1, T^*_2)$
      If $l = 0$
        Return $\mathrm{d}(T^*_1, T^*_2)$
      Else
        Define $T^{*0}_1$ to be $T^*_1$'s root, and $T^{*0}_2$ to be $T^*_2$'s root
        Define $T^*_{1L}$ to be the left subtree of $T^{*0}_1$, and $T^*_{1R}$ to be the right subtree of $T^{*0}_1$
        Define $T^*_{2L}$ to be the left subtree of $T^{*0}_2$, and $T^*_{2R}$ to be the right subtree of $T^{*0}_2$
        Calculate $C_1 = \mathrm{BM}(l-1, T^*_{1L}, T^*_{2L}) + \mathrm{BM}(l-1, T^*_{1R}, T^*_{2R})$
        Calculate $C_2 = \mathrm{BM}(l-1, T^*_{1L}, T^*_{2R}) + \mathrm{BM}(l-1, T^*_{1R}, T^*_{2L})$
        If $T^{*0}_1$ and $T^{*0}_2$ both have lock marks
          Return $\mathrm{d}(T^{*0}_1, T^{*0}_2) + C_1$
        Else
          Return $\mathrm{d}(T^{*0}_1, T^{*0}_2) + \min\{C_1, C_2\}$
        End of if
        % Define $\mathrm{BM}(l, T^*_1, T^*_2)$ recursively
      End of if
4   Output $\mathrm{BM}(m, T^*_a, T^*_b)$, which is the best-match semimetric $\mathrm{D}^*_{\mathrm{BM}}(T^*_a, T^*_b)$
```

Algorithm 3: Detailed workflow of calculating the best-match semimetric $\mathrm{D}^*_{\mathrm{BM}}$.

The time complexity of computing the best-match semimetric $\mathrm{D}^*_{\mathrm{BM}}$ is also $\mathcal{O}(n^2)$, where n is the number of vertices. The space complexity of computing $\mathrm{D}^*_{\mathrm{BM}}$ is $\mathcal{O}(n)$.

Conclusions

To study the early development of zygotes by comparing developmental trees, we define different distances on trees. On the space of rooted unordered trees with possibly repeated labels, we introduce two metrics: the best-match metric and the left-regular metric. For the same pair of trees, the best-match metric is no larger than the left-regular metric. They consider any common structures and compare non-common structures. Besides, different distances between labels and different weights on vertices can be introduced naturally. The best-match metric has an extra advantage: it is robust under small perturbations on labels. To compute the best-match metric, the time complexity is quadratic, and the left-regular metric has linear expected time complexity. In general, we recommend the best-match metric. If time cost is a major concern, the left-regular metric can be applied.

On the space of rooted trees with possibly repeated labels, where vertices might be ordered or unordered, most methods are not applicable, and we introduce the best-match semimetric. The properties of the best-match semimetric are almost the same as the best-match metric.

The methods introduced in this paper (possibly with modifications) can be applied to more commonly treated scenarios, e.g., unrooted trees, unlabeled or leaf-labeled trees, or trees with unique labels on vertices. Since our methods are not developed for such scenarios, the performance might not be as satisfactory as existing methods. Nevertheless, the ideas of our methods might inspire new methods in such scenarios.

Appendix

In this appendix, we present the detailed procedure of calculating the best-match metric D_{BM} and the left-regular metric D_{LR} for the developmental trees of *Arabidopsis thaliana* (T_{A}, Fig. 2) and sea urchin (T_{S}, Fig. 4).

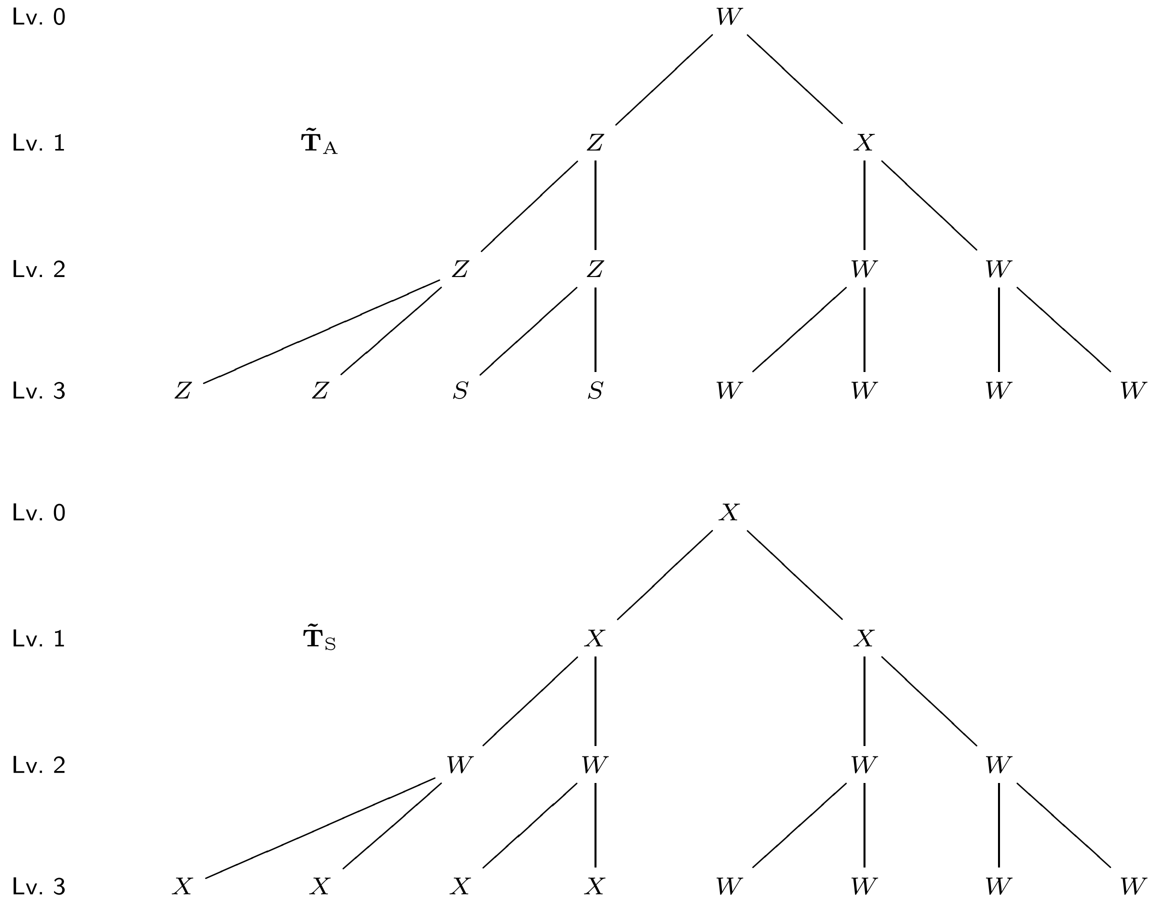

Fig. 13 The developmental trees of *Arabidopsis thaliana*, $\tilde{T}_{\mathrm{A}}$ and sea urchin, $\tilde{T}_{\mathrm{S}}$, after left-regularization

Best-match metric D_{BM}

The procedure is recursive. We need to determine the correspondence of subtrees rooted in level 1, which depends on the correspondence of subtrees rooted in level 2, which then depends on the correspondence of subtrees rooted in level 3.

Step 1:

$$\begin{aligned}\mathrm{D}_{\mathrm{BM}}(T_{\mathrm{A}}, T_{\mathrm{S}}) = &\mathrm{D}_{\mathrm{BM}}(WXZWWZZWWWWSSZZ,\\ &XXXWWWWXXXXWWWW)\\ =&\mathrm{d}(W,X) + \min\{\mathrm{D}_{\mathrm{BM}}(XWWWWWW, XWWXXXX)\\ &+ \mathrm{D}_{\mathrm{BM}}(ZZZSSZZ, XWWWWWW),\\ &\mathrm{D}_{\mathrm{BM}}(XWWWWWW, XWWWWWW)\\ &+ \mathrm{D}_{\mathrm{BM}}(ZZZSSZZ, XWWXXXX)\}.\end{aligned}$$

Step 2.1:

$$\begin{aligned}&\mathrm{D}_{\mathrm{BM}}(XWWWWWW, XWWXXXX)\\ &= \mathrm{d}(X,X) + \min\{\mathrm{D}_{\mathrm{BM}}(WWW, WXX) + \mathrm{D}_{\mathrm{BM}}(WWW, WXX),\\ &\quad \mathrm{D}_{\mathrm{BM}}(WWW, WXX) + \mathrm{D}_{\mathrm{BM}}(WWW, WXX)\}.\end{aligned}$$

Step 3.1.1 to Step 3.1.4 (the same procedure)

$$\begin{aligned}&\mathrm{D}_{\mathrm{BM}}(WWW, WXX)\\ &= \mathrm{d}(W,W) + \min\{\mathrm{d}(W,X) + \mathrm{d}(W,X), \mathrm{d}(W,X) + \mathrm{d}(W,X)\}\\ &=2.\end{aligned}$$

Back to Step 2.1

$$\mathrm{D}_{\mathrm{BM}}(XWWWWWW, XWWXXXX) = 4.$$

Step 2.2:

$$\begin{aligned}&\mathrm{D}_{\mathrm{BM}}(ZZZSSZZ, XWWWWWW)\\ &= \mathrm{d}(Z,X) + \min\{\mathrm{D}_{\mathrm{BM}}(ZSS, WWW) + \mathrm{D}_{\mathrm{BM}}(ZZZ, WWW),\\ &\quad \mathrm{D}_{\mathrm{BM}}(ZSS, WWW) + \mathrm{D}_{\mathrm{BM}}(ZZZ, WWW)\}.\end{aligned}$$

Step 3.2.1 and Step 3.2.3 (the same procedure)

$$\begin{aligned}&\mathrm{D}_{\mathrm{BM}}(ZSS, WWW)\\ &= \mathrm{d}(Z,W) + \min\{\mathrm{d}(S,W) + \mathrm{d}(S,W), \mathrm{d}(S,W) + \mathrm{d}(S,W)\}\\ &=3.\end{aligned}$$

Step 3.2.2 and Step 3.2.4 (the same procedure)

$$\begin{aligned}&\mathrm{D}_{\mathrm{BM}}(ZZZ, WWW)\\ &= \mathrm{d}(Z,W) + \min\{\mathrm{d}(Z,W) + \mathrm{d}(Z,W), \mathrm{d}(Z,W) + \mathrm{d}(Z,W)\}\\ &=3.\end{aligned}$$

Back to Step 2.2

$$\mathrm{D}_{\mathrm{BM}}(ZZZSSZZ, XWWWWWW) = 7.$$

Step 2.3:

$$\begin{aligned}&\mathrm{D}_{\mathrm{BM}}(XWWWWWW, XWWWWWW)\\ &= \mathrm{d}(X,X) + \min\{\mathrm{D}_{\mathrm{BM}}(WWW, WWW) + \mathrm{D}_{\mathrm{BM}}(WWW, WWW),\\ &\quad \mathrm{D}_{\mathrm{BM}}(WWW, WWW) + \mathrm{D}_{\mathrm{BM}}(WWW, WWW)\}.\end{aligned}$$

Step 3.3.1 to Step 3.3.4 (the same procedure)

$$\begin{aligned}&\mathrm{D}_{\mathrm{BM}}(WWW, WWW)\\ &= \mathrm{d}(W,W) + \min\{\mathrm{d}(W,W) + \mathrm{d}(W,W), \mathrm{d}(W,W) + \mathrm{d}(W,W)\}\\ &=0.\end{aligned}$$

Back to Step 2.3

$$\mathrm{D}_{\mathrm{BM}}(XWWWWWW, XWWWWWW) = 0.$$

Step 2.4:

$$\begin{aligned}&\mathrm{D}_{\mathrm{BM}}(ZZZSSZZ, XWWXXXX)\\ &= \mathrm{d}(Z,X) + \min\{\mathrm{D}_{\mathrm{BM}}(ZSS, WXX) + \mathrm{D}_{\mathrm{BM}}(ZZZ, WXX),\\ &\quad \mathrm{D}_{\mathrm{BM}}(ZSS, WXX) + \mathrm{D}_{\mathrm{BM}}(ZZZ, WXX)\}.\end{aligned}$$

Step 3.4.1 and Step 3.4.3 (the same procedure)

$$\begin{aligned}&\mathrm{D}_{\mathrm{BM}}(ZSS, WXX)\\ &= \mathrm{d}(Z,W) + \min\{\mathrm{d}(S,X) + \mathrm{d}(S,X), \mathrm{d}(S,X) + \mathrm{d}(S,X)\}\\ &=3.\end{aligned}$$

Step 3.4.2 and Step 3.4.4 (the same procedure)

$$\begin{aligned}&\mathrm{D}_{\mathrm{BM}}(ZZZ, WXX)\\ &= \mathrm{d}(Z,W) + \min\{\mathrm{d}(Z,X) + \mathrm{d}(Z,X), \mathrm{d}(Z,X) + \mathrm{d}(Z,X)\}\\ &= 3.\end{aligned}$$

Back to Step 2.4

$$\mathrm{D}_{\mathrm{BM}}(ZZZSSZZ, XWWXXXX) = 7.$$

Back to Step 1

$$\begin{aligned}\mathrm{D}_{\mathrm{BM}}(T_{\mathrm{A}}, T_{\mathrm{S}}) =& \mathrm{D}_{\mathrm{BM}}(WXZWWZZWWWWSSZZ,\\ & XXXWWWWXXXXWWWW)\\ =&8.\end{aligned}$$

Left-regularization metric D_{LR}

We use the total order $Z < X < W < S$ on the label set. We apply the left-regularization from level 2 to level 0.

Left-regularization on level 2:

For each vertex in level 2 of T_{A} and T_{S}, its left subtree and right subtree have the same label string, and we do not need to switch these subtrees. After this step, the label string of T_{A} is $WXZWWZZWWWWSSZZ$, and the label string of T_{S} is $XXXWWWWXXXXWWWW$.

Left-regularization on level 1:

For the vertex with label "Z" in level 1 of T_{A}, its left subtree label string is "ZSS", which is larger than that of its right subtree, "ZZZ". Thus we switch two subtrees of this vertex. For the other three vertices in level 1 of T_{A} and T_{S}, the left subtree and right subtree have the same label string, and we do not need to switch these subtrees. After this step, the label string of T_{A} is $WXZWWZZWWWWZZSS$, and the label string of T_{S} is $XXXWWWWXXXXWWWW$.

Left-regularization on level 0:

For the root vertex (level 0) of T_{A}, its left subtree label string is "XWWWWWW", which is larger than that of its right subtree, "ZZZZZSS". Thus we switch two subtrees of this vertex. For the root vertex of T_{S}, its left subtree label string is "XWWXXXX", which is smaller than that of its right subtree, "XWWWWWW". Thus we do not need to switch these subtrees. After this step, the label string of T_{A} is $WZXZZWWZZSSWWWW$, and the label string of T_{S} is $XXXWWWWXXXXWWWW$.

The left-regularization results of T_{A} and T_{S} are in Fig. 13. We can calculate the D_{OT} metric for these two trees. Since there are eight pairs of corresponding vertices with different labels, we have $\mathrm{D}_{\mathrm{LR}}(T_{\mathrm{A}}, T_{\mathrm{S}}) = 8$.

Acknowledgements
The author would like to thank Mikhail Gromov, Andrey Minarsky, Nadya Morozova, Robert Penner, and Alen Tosenberger for helpful discussions.

Authors contributions
YW is the only contributor of this work. The author read and approved the final manuscript.

Funding
This research was partially supported by NIH Grant R01HL146552.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The author declares that he has no competing interests.

Author details
[1] Department of Computational Medicine, University of California, Los Angeles, USA. [2] Institut des Hautes Études Scientifiques, Bures-sur-Yvette, France.

Received: 6 January 2022 Accepted: 16 May 2022
Published online: 06 June 2022

References

1. Wendrich JR, Weijers D. The Arabidopsis embryo as a miniature morphogenesis model. New Phytol. 2013;199(1):14–25.
2. Wang Y, Minarsky A, Penner R, Soulé C, Morozova N. Model of morphogenesis. J Comput Biol. 2020;27(9):1373–83.
3. Summers RG, Stricker SA, Cameron RA. Applications of confocal microscopy to studies of sea urchin embryogenesis. Methods Cell Biol. 1993;38:265–87.
4. Feldman JL, Geimer S, Marshall WF. The mother centriole plays an instructive role in defining cell geometry. PLoS Biol. 2007;5(6):149.
5. Bernardini G, Bonizzoni P, Della Vedova G, Patterson M. A rearrangement distance for fully-labelled trees. In: Proceedings of 30th Annual Symposium on Combinatorial Pattern Matching, pp. 28–12815; 2019.
6. Tai K-C. The tree-to-tree correction problem. J ACM. 1979;26(3):422–33.
7. Chartrand G, Saba F, Zou HB. Edge rotations and distance between graphs. Cas Pestovani Mat. 1985;110(1):87–91.
8. Billera LJ, Holmes SP, Vogtmann K. Geometry of the space of phylogenetic trees. Adv Appl Math. 2001;27(4):733–67.
9. Valiente G. An efficient bottom-up distance between trees. In: Proceedings of 8th Symposium on String Processing and Information Retrieval, pp. 212–219; 2001.
10. Zelinka B. Distances between rooted trees. Math Bohem. 1991;116(1):101–7.
11. Robinson DF, Foulds LR. Comparison of phylogenetic trees. Math Biosci. 1981;53(1–2):131–47.
12. Bogdanowicz D, Giaro K. On a matching distance between rooted phylogenetic trees. Int J Appl Math Comput Sci. 2013;23(3):669–84.
13. Critchlow DE, Pearl DK, Qian C. The triples distance for rooted bifurcating phylogenetic trees. Syst Biol. 1996;45(3):323–34.
14. Flajolet P, Sipala P, Steyaert J-M. Analytic variations on the common subexpression problem. In: International Colloquium on Automata, Languages, and Programming, Springer, pp. 220–234; 1990.
15. Diestel R. Graph theory. 5th ed. Berlin: Springer; 2017.
16. Ganapathy G, Goodson B, Jansen R, Le H-S, Ramachandran V, Warnow T. Pattern identification in biogeography. IEEE/ACM Trans Comput Biol Bioinform. 2006;3(4):334–46.
17. Butuzova O, Pakudin N, Minarsky A, Bessonov N, Morozova N. Developmental graphs comparison strategy for analysis of pattern formation and phylogeny. J Theor Biol. 2022;532:110925.
18. Harzheim E. Ordered sets. New York: Springer; 2006.

Publisher's Note