Characterization of random walks on space of unordered trees using efficient metric simulation

Farah Ben-Naoum, Christophe Godin, Romain Azaïs

CHARACTERIZATION OF RANDOM WALKS ON SPACE OF UNORDERED TREES USING EFFICIENT METRIC SIMULATION

Farah Ben Naoum[1], Christophe Godin[2,†], and Romain Azaïs[2,†,*]

[1]EEDIS Laboratory, Computer Sciences Department, Djillali Liabes University, Sidi Bel-Abbes, Algeria.
[2]Laboratoire Reproduction et Développement des Plantes, Univ Lyon, ENS de Lyon, UCB Lyon 1, CNRS, INRAE, Inria, F-69342, Lyon, France.
[†]Romain Azaïs and Christophe Godin's work has been supported by the European Union's H2020 project ROMI.
[*]Corresponding author: Romain Azaïs, romain.azais@inria.fr

ABSTRACT

The simple random walk on $\mathbb{Z}^p$ shows two drastically different behaviours depending on the value of p: it is recurrent when $p \in \{1, 2\}$ while it escapes (with a rate increasing with p) as soon as $p \geq 3$. This classical example illustrates that the asymptotic properties of a random walk provides some information on the structure of its state space. This paper aims to explore analogous questions on space made up of combinatorial objects with no algebraic structure. We take as a model for this problem the space of unordered rooted trees endowed with Zhang edit distance. To this end, it defines the canonical unbiased isotropic random walk on the space of trees and provides an efficient algorithm to evaluate its escape rate. Compared to Zhang algorithm, it is incremental and computes the edit distance along the random walk approximately 100 times faster on trees of size 500 on average. The escape rate of the random walk on trees is precisely estimated using intensive numerical simulations, out of reasonable reach without the incremental algorithm.

1 Introduction

A random walk is a discrete trajectory, i.e., a sequence of locations in a space, where each location is obtained by a random elementary move from the previous one. When the space is endowed with a graph structure, the canonical random walk evolves along the edges of the graph. On a discrete metric space, the natural graph structure puts edges between neighbours, i.e., between elements x and y of $\mathcal{X}$ such that $d(x, y)$ is minimal nonnegative, which can be assumed to be 1 without loss of generality. In addition, the walk is said isotropic when all the elementary moves share the same probability. In other words, the isotropic random walk on $(\mathcal{X}, d)$ is a sequence $(X_h)_{h\geq 0}$ of elements of $\mathcal{X}$ such that the conditional probability of X_{h+1} given $X_h = x$ is the uniform distribution on $\{y \in \mathcal{X} : d(x, y) = 1\}$ (see [Göbel and Jagers, 1974] for finite graphs and [Woess, 2000, I.1.C. Random walks on graphs] for locally finite graphs).

The end-to-end distance, also called distance from origin, is defined as $d(X_0, X_h)$ and quantifies the remoteness to the initial value. The asymptotic behaviour of this function of the random walk is often referred to as escape rate and has been investigated in various settings and from different points of view (see [Gouëzel, 2017] for a theoretical analysis for random walks defined on hyperbolic groups, [Woess, 2000, II.8.A. The rate of escape] for a study in a general setting, and [Mazur, 1965] in the context of a polymer molecule model).

For example, on $\mathbb{Z}^p$ with $p \geq 3$, the random walk (almost surely) goes to infinity, and behaves at first order when h goes to infinity as

$$\sqrt{\frac{2h}{p}} \frac{\Gamma\left(\frac{p+1}{2}\right)}{\Gamma\left(\frac{p}{2}\right)},$$

where $\Gamma(z) = \int_0^\infty x^{z-1} \exp(-x) dx$. When $p \in \{1, 2\}$, the behaviour of the random walk is drastically different: the random walk is recurrent, meaning that it visits its origin location infinitely often with probability 1. In other words, when $p < 3$, the local connectivity of $\mathbb{Z}^p$ is small enough to avoid the escape of the random walk, while when $p \geq 3$, the process escapes with the rate above obeying to the following expected property: the larger the dimension (and thus the connectivity), the faster the escape rate.

By definition, the isotropic random walk is a canonical object associated with the structure of the space $\mathcal{X}$, i.e., with its metric through its local connectivity. This example shows that the properties of the random walk, in particular the asymptotic properties, can be strongly related to the structure of the state space (the dimension in the case of the random walk on $\mathbb{Z}^p$ but it could be a more complex dependency). As a consequence, one can build an understanding of the local structure of a discrete metric space through the study of the behaviour of the canonical random walks on it. In this paper, we aim to develop such considerations for unordered rooted trees. It should be noted that this space has no classical algebraic structure, which considerably complicates its study.

One can distinguish two main types of metrics on the space of unordered rooted trees: edit distances and alignment distances, which form a particular case of the former (see the survey [Bille, 2005] and the references therein). An edit distance is defined as the minimal cost of a sequence of allowed elementary edit operations that transform a tree into another (up to isomorphism). Typically, the edit operations consist in adding a node, deleting a node, and changing the label of a node. It has been shown that the problem of computing a general edit distance between two unordered labeled trees is NP-hard (see [Bille, 2005, Zhang, 1996]). Taking this into account, one line of research is to develop fixed-parameter polynomial algorithms, which aim to estimate a given parameter assuming that it is fixed in the evaluation of the time-complexity. For instance, [Akutsu et al., 2011] develops and investigates an exact algorithm for a general tree edit distance that runs in polynomial time but assuming the fact that the target distance is fixed. Another option is to add restrictions to the definition of the distance so that the underlying optimization problem becomes polynomial. [Zhang, 1996] has introduced a slight constraint on the set of edit operations which made reachable a polynomial algorithm. Interestingly, it can be noted that the less restrictive conditions imposed in [Lu, 1979] are only valid for ordered trees, while the problem remains NP-hard for unordered trees (see [Bille, 2005, 3.4 Constrained edit distance]). Zhang edit distance, denoted by δ in this paper, is one of the most general metrics known that remains polynomial without assuming that some parameters are fixed or bounded.

The local connectivity of trees endowed with Zhang edit distance is non-trivial. First of all, there is a minimal element: the tree composed of a unique node. This tree has a unique neighbour which is composed of two vertices, one being the unique child of the other. In addition, from a tree with n nodes, there are from n to $2n$ ways to construct a bigger tree that remains at distance 1. However, the number of nodes that can be deleted in Zhang distance is always less than n, meaning that there are less than n neighbours smaller than the original tree. This draws a complex funnel-like structure, where the number of neighbours of a tree linearly increases as one moves away from the tree with one node. This being established and recalling the asymptotic behaviour of random walks on $\mathbb{Z}^p$ for which the number of neighbours of any state is $2p$, one can expect a random walk on trees to escape fastly from its starting point. Nevertheless, the form of the escape rate function is far from obvious. In this paper, we investigate, from a numerical point of view, the escape rate of the canonical random walk on trees endowed with Zhang edit distance.

A natural way to proceed is to generate numerically a large number N of random walks within a large time window $[0, H]$ in order to estimate accurately the average escape rate, then evaluate its dependency on the number of elementary moves. In other words, we need to simulate N independent random walks on trees $(X_h^i)_{0 \leq h \leq H}$, $1 \leq i \leq N$, and keep track of the end-to-end distance processes $(\delta(X_0^i, X_h^i))_{0 \leq h \leq H}$. Consequently, $N \times H$ edit distances need to be evaluated. In light of the quadratic complexity of Zhang algorithm [Zhang, 1996], this approach clearly requires intensive computing resources, making it impractical as soon as the trees get large.

In the present paper, we remark that X_{h+1}^i is not completely independent of X_h^i but obtained from it by one of the elementary moves allowed in Zhang edit distance. In this context, we show that the edit distance $\delta(X_0^i, X_{h+1}^i)$ does not require to process Zhang algorithm in full, but can be evaluated fastly from $\delta(X_0^i, X_h^i)$ (and the computations made to evaluate it) and the knowledge of the edit operation from X_h^i to X_{h+1}^i. More precisely, we exhibit an incremental algorithm for Zhang edit distance, which achieves much shorter computation times than the original version: for instance, the computation time is divided by almost 100 for trees of size 500. Applying this idea from the initial value of the random walk, one can keep track of the distance process without any intensive computations. This notably makes possible to estimate accurately the escape rate function of the canonical random walk on trees on a commercial laptop (90M Zhang edit distances on trees up to several hundreds of nodes were evaluated to write this paper).

The paper is organized as follows. Section 2 is devoted to precise definitions of the concepts of interest: trees in Subsection 2.1, Zhang edit distance in Subsection 2.2, and isotropic random walk on trees in Subsection 2.3. Our incremental algorithm for Zhang edit distance is developed in Section 3. Pseudo-code is provided in Appendix A.

A simulation study is presented in Section 4. In particular, Subsection 4.2 presents our investigations on the sharp estimation of the escape rate of random walks on trees.

2 Random walk on trees endowed with Zhang distance

2.1 Definition of unordered trees

Rooted trees A rooted tree T is a connected acyclic digraph such that there is a unique vertex $\mathcal{R}(T)$ (the root of T) that has no parent, and any vertex different from the root has exactly one parent. The parent of a vertex w is denoted $\mathcal{P}(w)$ and the set of all its ancestors by $\mathcal{P}^+(w)$, while all the nodes that have v for parent form the set of children of v, denoted $\mathcal{C}(v)$. $\mathcal{L}(T)$ is the set of leaves of T that are all the vertices without children. The tree without any vertex represents the empty tree and is denoted Θ. The subtree $T[v]$ rooted in v is the subgraph of T composed of v and all its descendants in T with edges inheriting from T. $F[v]$ denotes the forest of subtrees emanating from v, i.e., the set of trees $T[w]$ where $w \in \mathcal{C}(v)$.

Unordered trees In this paper, we consider unordered rooted trees for which the left-to-right order among sibling vertices is not significant. Unordered rooted trees, simply called unordered trees or trees in the sequel, are defined from the definition of tree isomorphism. A one-to-one correspondence $\Phi : T \to T'$ is called a tree isomorphism if w is a child of v in T implies that $\Phi(w)$ is a child of $\Phi(v)$ in T'. T and T' are said isomorphic whenever there exists a tree isomorphism between them. The existence of a tree isomorphism defines an equivalence relation on the set of rooted trees. The class of unordered rooted trees is the quotient set of rooted trees by the existence of a tree isomorphism. Remarkably, one can determine if two trees of size n are isomorphic, which the most elementary operation required to handle unordered trees, in $O(n)$ [Aho et al., 1974, Example 3.2 and Theorem 3.3].

Characteristics of trees A tree T can be described by several integer-valued characteristics that will be used in this paper, namely:

- its size $\#T$ that counts the number of vertices of T;
- its height $\mathcal{H}(T)$ that is the length of the longest path from the root to the leaves;
- its outdegree $\mathcal{D}(T)$ that is the maximum number of children that can be found in T;
- its Strahler number, recursively defined on vertices v of T as $\mathcal{S}(v) = 1$ if $v \in \mathcal{L}(T)$, and

$$\mathcal{S}(v) = \max_{c \in \mathcal{C}(v)} \mathcal{S}(c) + \begin{cases} 0 & \text{if} \quad \#[\arg\max_{c \in \mathcal{C}(v)} \mathcal{S}(c)] = 1, \\ 1 & \text{if} \quad \#[\arg\max_{c \in \mathcal{C}(v)} \mathcal{S}(c)] \geq 2, \end{cases}$$

  else, the Strahler number of T being defined as the Strahler number of its root.

2.2 Definition of Zhang distance from edit operations

Editing distances consist in evaluating the number of elementary operations transforming one tree in another. General editing operations for unordered trees were introduced in [Zhang et al., 1992], but this paper also shows that computing the induced distance is NP-complete [Zhang et al., 1992, Theorem 11]. In [Zhang, 1996], the set of editing operations is slightly constrained so that a polynomial algorithm for computing the induced distance, refered to as Zhang distance in this paper, is exhibited. Zhang distance is defined from the following set of edit operations of a tree T.

- Add leaf under u: add a new vertex k in the set of children of u such that k is a leaf.
- Del leaf k: remove leaf k from the set of children of its parent.
- Add internal node under u: replace the set of children of u by a unique vertex k so that the set of children of k is the previous set of children of u.
- Del internal node k (only if k is the unique child of its parent u): remove k so that the set of children of k becomes the children of u.

For the sake of clarity, these 4 operations are presented in Fig. 1.

The tree obtained after an edit operation o applied to a tree T is denoted $o(T)$. An edit operation sequence is an ordered list of edit operations $s = (o_n, \ldots, o_1)$ applicable to T, i.e., such that, for any $1 \leq i \leq n-1$, operation o_{i+1} makes sense to be applied to $o_i \circ \cdots \circ o_1(T)$.

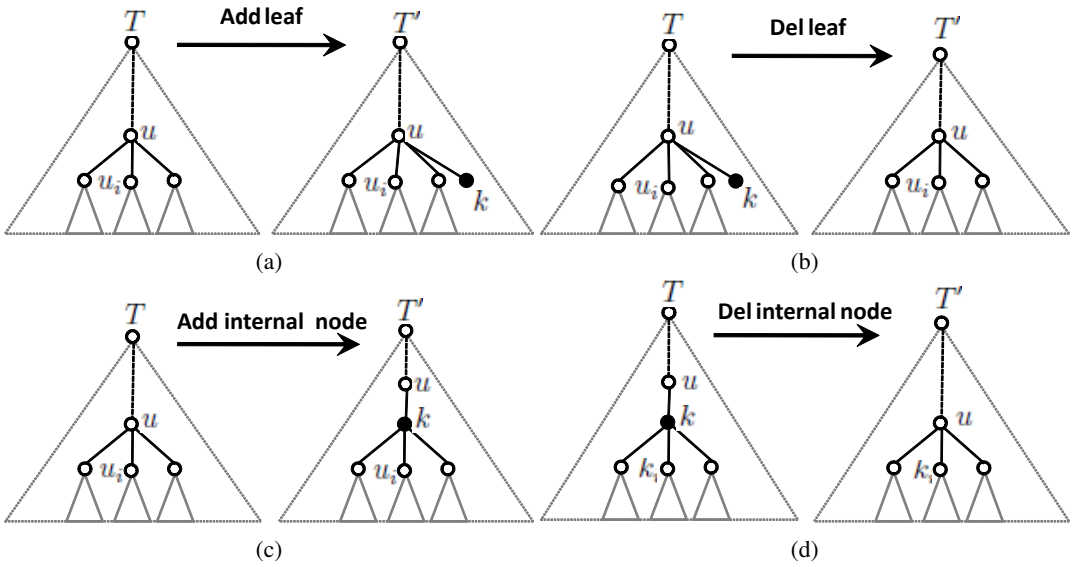


Figure 1: Elementary tree editing operations transforming a tree T into an other tree T', in such edition either: (a) a new leaf k is inserted under vertex u, (b) leaf k of a parent u in T is deleted from T', (c) a new vertex k is added under the vertex u such that k becomes the unique child of u and the children set of k is the previous u children set, (d) k is the unique child of its parent u then remove k so that the k children set becomes the children set of u.

The edit distance between two trees T and T' is the minimal length of edit operation sequences transforming T into a tree isomorphic to T', i.e.,

$$\delta(T, T') = \min_{s(T)=T'} \#s. \tag{1}$$

δ defines a distance metric on the space of unordered trees [Zhang et al., 1992, 2.1. Editing operations and editing distance between unordered labeled trees].

2.3 Isotropic random walk on tree space

A Markov chain on a discrete space $\mathcal{X}$ is defined from a transition kernel Q, that gives, for any $(x, y) \in \mathcal{X}^2$, the probability $Q(x, y)$ to go from x to y in one step. If the space $\mathcal{X}$ is augmented with a graph structure, i.e., if a set $\mathcal{E}$ of (undirected) edges on $\mathcal{X}$ is given, a random walk is defined as a Markov chain such that $Q(x, y) > 0$ if and only if $(x, y) \in \mathcal{E}$, i.e., if $y \in \mathcal{N}(x)$ where $\mathcal{N}(x)$ is the set of neighbors of x,

$$\mathcal{N}(x) = \{y \in \mathcal{X} \ : \ (x, y) \in \mathcal{E}\}.$$

(see [Woess, 2000, I.1.C. Random walks on graphs] where these random walks are referred to as of nearest neighbor type). The transitions of an isotropic random walk, also called simple random walk, should not privilege one specific direction: in general, one expects $Q(x, \cdot)$ to be the uniform distribution on the set $\mathcal{N}(x)$ (supposed to be finite for any x), i.e.,

$$Q(x, y) = \begin{cases} \frac{1}{\#\mathcal{N}(x)} & \text{if } y \in \mathcal{N}(x), \\ 0 & \text{else,} \end{cases}$$

(see [Göbel and Jagers, 1974] for finite graphs and [Woess, 2000, eq. (1.19)] for locally finite graphs).

Zhang distance induces a natural locally finite graph structure on the set of unordered trees, for which two trees form an edge if they are at distance 1. In this case, the set $\mathcal{N}(T)$ of neighbors of T is given by

$$\begin{aligned} \mathcal{N}(T) &= \{T' \ : \ \delta(T, T') = 1\}, \\ &= \{o(T) \ : \ o \in \mathcal{O}\}, \end{aligned}$$

where $\mathcal{O}$ denotes the set of all possible operations applicable to T. This set can be decomposed as

$$\mathcal{O} = \mathcal{O}_{\text{AL}} \cup \mathcal{O}_{\text{DL}} \cup \mathcal{O}_{\text{AIN}} \cup \mathcal{O}_{\text{DIN}},$$

where

- $\mathcal{O}_{\text{AL}}$ stands for the set of add leaf operations, which cardinality is $\#T$;
- $\mathcal{O}_{\text{DL}}$ stands for the set of del leaf operations, which cardinality is $\#\mathcal{L}(T)$;
- $\mathcal{O}_{\text{AIN}}$ stands for the set of add internal node operations, which cardinality is $\#T - \#\mathcal{L}(T)$;
- $\mathcal{O}_{\text{DIN}}$ stands for the set of del internal node operations, which cardinality is the number of single children that are not leaves, which is bounded by $\#T - \#\mathcal{L}(T) - 1$.

Consequently, the number of operations that make the tree increase, $2\#T - \#\mathcal{L}(T)$, is much larger than the number of operations that make the tree decrease, roughly upper-bounded by $\#T - 1$. Thus, choosing a random neighbor of T by selecting an edit operation with uniform distribution on $\mathcal{O}$ introduces a bias that tends to augment the size of the tree, which is not the expected behavior of an isotropic random walk.

In order to avoid this bias, we propose to select with probability $1/2$ an adding operation, chosen with uniform distribution on $\mathcal{O}_{\text{AL}} \cup \mathcal{O}_{\text{AIN}}$, and with probability $1/2$ a deleting operation, chosen with uniform distribution on $\mathcal{O}_{\text{DL}} \cup \mathcal{O}_{\text{DIN}}$, which leads to

$$Q(T, T') = \begin{cases} \frac{1}{2(\#\mathcal{O}_{\text{AL}} + \#\mathcal{O}_{\text{AIN}})} & \text{if } T' = o(T) \text{ with } o \in \mathcal{O}_{\text{AL}} \cup \mathcal{O}_{\text{AIN}}, \\ \frac{1}{2(\#\mathcal{O}_{\text{DL}} + \#\mathcal{O}_{\text{DIN}})} & \text{if } T' = o(T) \text{ with } o \in \mathcal{O}_{\text{DL}} \cup \mathcal{O}_{\text{DIN}}, \\ 0 & \text{else.} \end{cases} \tag{2}$$

In the sequel, $(T_h)_{h \geq 0}$ denotes the isotropic random walk defined from the transition kernel Q given in (2), i.e.,

$$\forall\, h \geq 0,\ \mathbb{P}(T_{h+1} = y | T_h = x) = Q(x, y).$$

A sample trajectory is presented in Fig. 2.

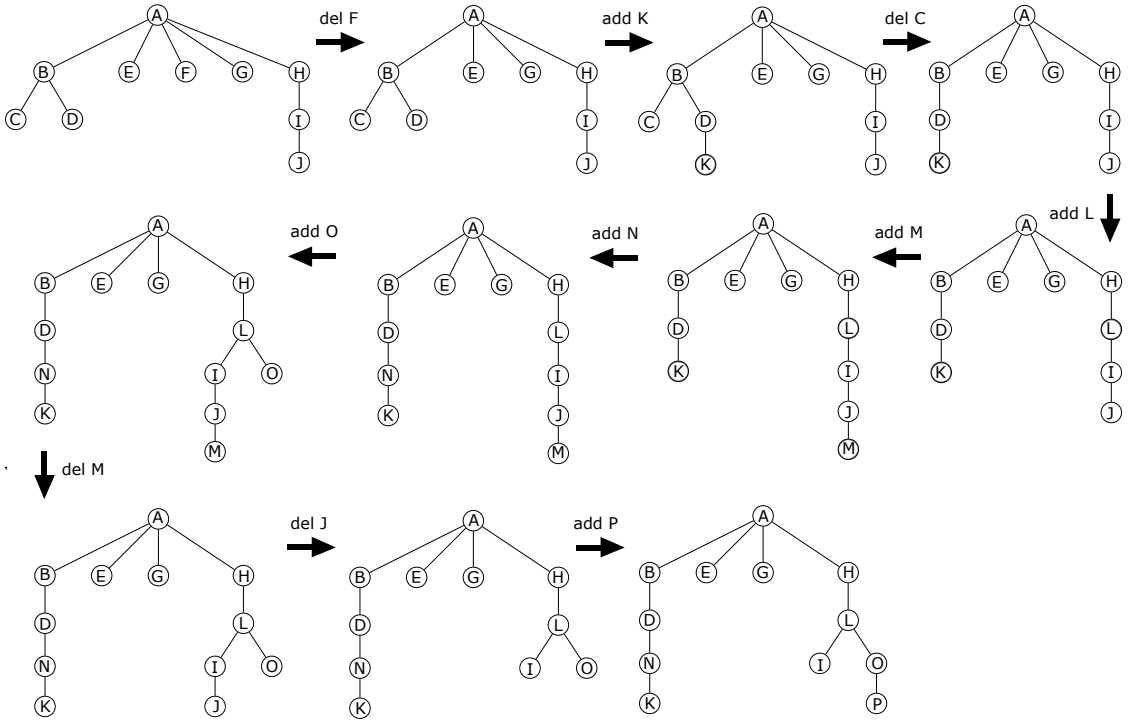


Figure 2: Example of the 10 first steps of a random walk starting from a tree of size 10. Colors indicate if a vertex was present in the original tree (white), if a vertex will be deleted at the next step (light gray), if a vertex has been previously added (dark gray), or if a vertex has been added at the last step (black).

The behavior of $(T_h)_{h \geq 0}$ can be described through integer-valued characteristics of trees, namely the size $(\#T_h)_{h \geq 0}$, the height $(\mathcal{H}(T_h))_{h \geq 0}$, the outdegree $(\mathcal{D}(T_h))_{h \geq 0}$, or the Strahler number $(\mathcal{S}(T_h))_{h \geq 0}$ (see Subsection 2.1 for definition of these quantities). However, one important feature of random walks is the end-to-end distance (see [Gouëzel, 2017, Mazur, 1965, Woess, 2000] for various contexts and objectives), which evaluates the remoteness to the initial value, given in the framework of Zhang distance by $(\delta(T_0, T_h))_{h \geq 0}$.

3 Efficient computation of the tree edit distance in a random walk context

During a random walk, our goal is to analyze the behaviour of the above tree-edit distance $\delta(T_0, T_h)$ given in (1) between the origin tree and the current tree. For unordered trees, this computation can be carried out using the *dynamic programming* principle in a recursive and bottom-up manner, from the leaves of the trees to their roots [Zhang, 1996]. In this section, we will show how this computation can be carried out efficiently, in an incremental manner during the random walk.

3.1 Recursive computation of tree edit distance

[Zhang, 1996] introduced an efficient algorithm to compute edit distances between unordered trees (here corresponding to T_0 and T_h), assuming simple edit operations (see Subsection 2.2). This algorithm relies on the following set of recursive equations that proceed bottom up from the leaves to the tree root,

$$\delta(v,w) = \min \begin{cases} \delta(\Theta, w) + \min\limits_{w_i \in \mathcal{C}(w)} \{\delta(v, w_i) - \delta(\Theta, w_i)\}, \\ \delta(v, \Theta) + \min\limits_{v_i \in \mathcal{C}(v)} \{\delta(v_i, w) - \delta(v_i, \Theta)\}, \\ d(v,w) + \delta_F(v,w), \end{cases}$$
$$\delta_F(v,w) = \min \begin{cases} \delta_F(\Theta, w) + \min\limits_{w_i \in \mathcal{C}(w)} \{\delta_F(v, w_i) - \delta_F(\Theta, w_i)\}, \\ \delta_F(v, \Theta) + \min\limits_{v_i \in \mathcal{C}(v)} \{\delta_F(v_i, w) - \delta_F(v_i, \Theta)\}, \\ \delta_{FM}(v,w) = \sum\limits_{(v_i, w_j) \in M^*(v,w)} \delta(v_i, w_j), \end{cases} \tag{3}$$

where $\delta(v,w)$ and $\delta_F(v,w)$ respectively represent the distance between subtrees $T_0[v]$ and $T_h[w]$ and the distance between the forests $F_0[v]$ and $F_h[w]$ (see Fig. 3). $\delta_{FM}(v,w)$ is the cost of $M^*(v,w) \subseteq \mathcal{C}(v) \times \mathcal{C}(w)$ the *optimal forest mapping* rooted in v and w. For this define $\mathcal{M}(v,w)$ the set of all mappings between $F[v]$ and $F[w]$. Based on the above edit operations, the only valid mappings between forests must verify [Zhang, 1996],

$$\forall\,(x,y) \in M(v,w),\ \forall\,(x',y') \in M(v,w),\ x = x' \Leftrightarrow y = y'.$$

An *optimal forest mapping* M^* between $F_0[v]$ and $F_h[w]$ must satisfy,

$$M^*(v,w) = \underset{M(v,w) \in \mathcal{M}(v,w)}{\arg\min} \left(\sum_{(x,y) \in M(v,w)} \delta(x,y) + \sum_{x \in M^-(v,w)} \delta(x,\Theta) + \sum_{y \in M^+(v,w)} \delta(\Theta,y) \right).$$

$M^-(v,w)$ and $M^+(v,w)$ represent respectively the subsets of $\mathcal{C}(v)$ and $\mathcal{C}(w)$ which are not included in the mapping $M(v,w)$.

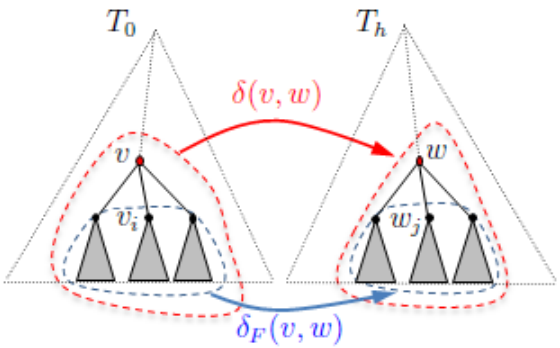


Figure 3: Intermediate distances involved in computation of the mapping distance of trees T_0 and T_h. Dashed lines surround the elements to be mapped for each pair (v,w): $\delta(v,w)$ is the mapping distance between the subtrees $T_0[v]$ and $T_h[w]$, and $\delta_F(v,w)$ is the mapping forest distance between $F_0[v]$ and $F_h[w]$.

[Zhang, 1996] modeled this problem as a minimum cost maximum bipartite matching problem. A weighted bipartite graph G is constructed, made of sets of vertices: a first set V_0 represents the trees of the forest $F_0[v]$ and a second set V_h represents the trees of $F_h[w]$. In addition, if the two forests have different sizes, a vertex θ, representing the empty

tree, is added to the set with lower number of vertices. An edge is created between each pair of vertices in $V_0 \times V_h$, and is attached a weight γ_{ij} corresponding to the tree edit distance between the subtrees $T_0[v_i]$ and $T_h[w_j]$ represented by the two connected vertices in G. Two nodes representing the source s and the sink t are added to G (see Fig. 4). Using this graph, the above forest mapping problem is solved as the *minimum cost maximum flow* (MCF) problem on G [Király and Kovács, 2012].

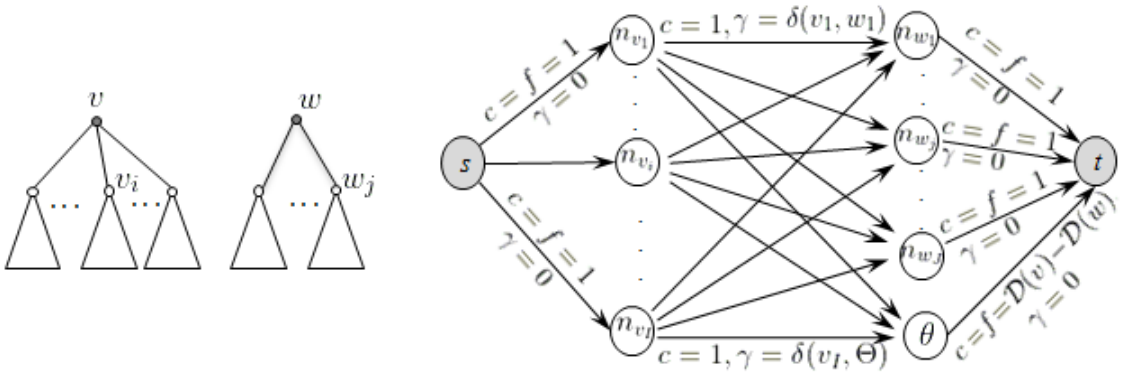


Figure 4: Two forests $F_0[v]$ and $F_h[w]$ (left) of mapping cost computed from the associated matching graph (right), where for each edge the triplet (f, c, γ) denotes the flow, the capacity and the cost. Note that for this example $\mathcal{D}(v) > \mathcal{D}(w)$.

MCF problem [Király and Kovács, 2012] Let $G = (V, E)$ be a weighted connected directed graph. Each edge $(i, j) \in E$ is associated with 2 non-negative integers f_{ij} and c_{ij} representing respectively the edge *flow* and *capacity*, and a real γ_{ij} defining the *cost* per unit flow f_{ij} on the arc. The *cost of a flow* f on G is defined as

$$\gamma(f) = \sum_{(i,j)\in E} \gamma_{ij} f_{ij}.$$

In addition, we assume that there exist two nodes in G, denoted s and t, respectively called the source and target nodes such that s has no entering edge, and t has no outgoing edge. All the other nodes are called *transit* nodes.

A flow f is *feasible*, if and only if it meets the following constraints:

1. Capacity: for any $(i, j) \in E$, $0 \leq f_{ij} \leq c_{ij}$.
2. Flow conservation: for any transit node i, $\sum_{(i,j)\in E} f_{ij} = \sum_{(j,i)\in E} f_{ji}$.
3. Supply-demand conservation: the source supply value $\sum_{(s,j)\in E} f_{sj}$ is equal to the sink demand value $\sum_{(i,t)\in E} f_{it}$.

A Minimum Cost Maximum Flow (MCF) problem consists in finding an *optimal* flow f^* representing the feasible maximal flow on the graph which minimizes the flow cost.

The search of the optimal forest mapping from $F_0[v]$ to $F_h[w]$ thus corresponds to solving the MCF problem associated with the graph G encoding forests $F_0[v]$ and $F_h[w]$. Interestingly, different algorithms can be used for the MCF resolution, such as [Edmonds and Karp, 1972, Edmonds and Karp, 2003], Network Simplex [Király and Kovács, 2012, Ahuja and Magnanti, 2018], and the Cost-Scaling algorithm [Goldberg and Tarjan, 1990]. In [Zhang, 1996], the former algorithm, improved by Tarjan [Tarjan, 1983], is used, which induces the final time complexity of the overall algorithm $O(\#T_0 \times \#T_h \times (\mathcal{D}(T_0) + \mathcal{D}(T_h)) \times \log_2(\mathcal{D}(T_0) + \mathcal{D}(T_h))$.

3.2 Incremental tree edit distance

The above set of recursive equations (3) makes it possible to compute the tree edit distance between 2 trees in a time essentially polynomial in the product of the input tree sizes. However, during a random walk $(T_h)_{h\geq 0}$, performing this computation at each step between the current tree T_h and the original tree T_0 would be prohibitive. In this section, we show that this computational cost can be drastically reduced by making use of the fact that each tree T_h is followed by a tree T_{h+1} that results from an elementary edit operation made on T_h.

As we have seen above, such an elementary operation can either be a vertex insertion or deletion in the tree T_h (see Fig. 1 in which T and T' are here respectively referenced as trees T_h and T_{h+1}). We remark that because the tree T_{h+1} differs only by one vertex from the tree T_h, the recursive computation of $\delta(T_0, T_{h+1})$ will differ only slightly from the recursive computation of $\delta(T_0, T_h)$: most of the intermediate distances computed during the recursion are actually the same for a majority of subtrees not affected by the edit operation. In the sequel, we make use of this property to reduce the complexity of the computation of $\delta(T_0, T_{h+1})$, by updating in (3) only the distances of $\delta(T_0, T_h)$ impacted by the edit operation at step $h+1$.

In the computation of the distance between two trees, we note that a number of intermediate quantities, $\delta(v, w)$, $\delta(v, \Theta)$, $\delta(\Theta, w)$, $\delta_F(v, w)$, $\delta_F(v, \Theta)$, $\delta_F(\Theta, w)$ and $M^*(v, w)$, are recursively computed for each pair (v, w) during the algorithm following (3). These quantities define the *distance information at* (v, w).

Let us assume that we already computed the distance between T_0 and T_h using (3) and that, at the next step, a vertex k is either inserted in or deleted from the tree T_h to produce T_{h+1}. Then, in the recursion of (3) not all the intermediate quantities need be recomputed to compute $\delta(T_0, T_{h+1})$. Indeed, all the distances $\delta(v, w)$ for w not an ancestor of k, are left unchanged by the edit operation changing T_h in T_{h+1}. Only the distances $\delta(v, w)$ for $w \in \mathcal{P}^+(k)$ actually need to be re-evaluated (see Fig. 5).

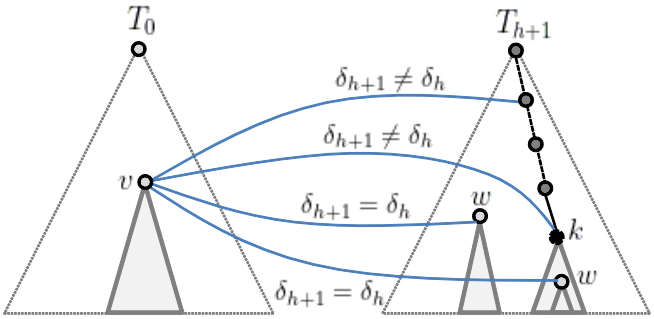


Figure 5: Tree T_{h+1} in which a node k is edited (just inserted or about to be deleted). In step $h+1$, the only distance informations to be updated are for the vertex k and for all k ancestors, while for all remaining T_{h+1} vertices the distance information is preserved compared with that computed in step h.

How the distance information at (v, w) used to compute $\delta(T_0, T_h)$ must be updated at step $h+1$ to compute $\delta(T_0, T_{h+1})$ depends on the type of edit operation (see Fig. 1) and on the position of the vertex w with respect to the edited vertex k (see Fig. 5). Let us consider the different situations leading to different updating strategies.

Leaf insertion (see Fig. 1a) T_{h+1} results from the insertion of a leaf k in T_h. Then, for all v in T_0 and for every vertex w in T_{h+1}, most of the distance information at (v, w) remains unchanged compared with what was computed at the previous step, except in three cases:

i. $w = k$: the distance information at (v, w) is set to that of (v, l), where l is any other leaf in T_h.

ii. $w = \mathcal{P}(k)$: at step $h+1$, the forest $F_{h+1}[w]$ is made of the forest $F_h[w]$ augmented with the subtree $T_{h+1}[k]$ (here reduced to only one leaf vertex). This means that the distance information at (v, w) at step $h+1$ must be computed using (3) to account for the fact that (see Fig. 6a):
   - The distance information at (v, k) is new.
   - For every $v_i \in \mathcal{C}(v)$, the distance informations at (v_i, w) and at (Θ, w) have potentially changed due to the leaf insertion.
   - The optimal forest mapping $M^*(v, w)$ obtained in step h is no longer valid at step $h+1$. We have to integrate the new subtree $T_{h+1}[k]$ in the mapping. See problem B in paragraph *Re-evaluation of forest mappings* below.

iii. w is an ancestor of $\mathcal{P}(k)$: at step $h+1$, the forest $F_h[w]$ undergoes an edition without affecting the w children list. Therefore, the distance information at (v, w) at step $h+1$ must be recomputed using (3) to account for the following changes with respect to the computation at step h (see Fig. 6b):
   - The distance information at (v_i, w) for every $v_i \in \mathcal{C}(v)$, the distance information at (v, w_j) for $w_j \in \mathcal{P}^+(k)$, and the distance information at (Θ, w) have potentially changed due to the insertion.
   - Here the forest mapping at step h remains a valid mapping at step $h+1$ but should be further optimized to account for potential changes in the distance between subtrees. See problem A in paragraph *Re-evaluation of forest mappings* below.

Leaf deletion (see Fig. 1b) The leaf k has just been deleted from T_h, and the only tree distance informations that are affected by this deletion are for the two cases:

i. $w = \mathcal{P}(k)$: at step $h+1$ the subtree $T_h[k]$ (here reduced to only one leaf vertex) is substracted from the forest $F_h[w]$. The distance informations at (v, w) in step $h+1$ must be recomputed using (3) to account for the following changes with respect to the distance computation in step h (see Fig. 6c):
   - The previous distance informations at (v, k) are excluded from the computation.
   - For every $v_i \in \mathcal{C}(v)$, the distance informations at (v_i, w) and at (Θ, w) have potentially changed due to the deletion.
   - The optimal forest mapping $M^*(v, w)$ computed in step h is no longer valid at step $h+1$, since the subtree $T_h[k]$ has to be excluded from the mapping. See problem B in paragraph *Re-evaluation of forest mappings* below.

ii. w is an ancestor of $\mathcal{P}(k)$: the distance information needs to be updated as in (iii) of leaf insertion (see above).

Internal node insertion (see Fig. 1c) T_{h+1} results from the insertion of an internal vertex k in T_h. Then, the only tree distance informations that are affected by this insertion are as follows:

i. $w = k$: let us denote u the vertex of T_h under which the insertion takes place (see Fig. 1c). In the new tree T_{h+1}, the insertion of an internal vertex w under u means that, by definition, the forests $F_h[u]$ and $F_{h+1}[w]$ are identical. As a consequence, the distance information at step $h+1$ at (v, w) is equal to that previously obtained for (v, u) at step h and does not need to be re-evaluated.

ii. $w = \mathcal{P}(k)$: if k is an internal vertex inserted under w in T_{h+1}, a first step of the edition at level of vertex w is to empty the forest $F_{h+1}[w]$ before adding to it the subtree $T_{h+1}[k]$. Meaning that, the distance information at (v, w) is at first set to that of (v, l), l any leaf in T_h. Next, to deal with the insertion of the subtree $T_{h+1}[k]$ in the temporarily empty forest $F_{h+1}[w]$, the obtained distance information at (v, w) is re-evaluated in a similar way as for case (ii) of leaf insertion (see above).

iii. w is an ancestor of $\mathcal{P}(k)$: see above leaf insertion (iii).

Internal node deletion (see Fig. 1d) Internal vertex k has just been deleted from T_h, and the only tree distance informations that are affected by this deletion are:

i. $w = \mathcal{P}(k)$: if k is an internal vertex deleted under w, then the forests $F_{h+1}[w]$ and $F_h[k]$ are identical, and the distance information at (v, w) in step $h+1$ is equal to the distance information at (v, k) of step h and does not need to be re-evaluated.

ii. w is an ancestor of $\mathcal{P}(k)$: see above leaf insertion (iii).

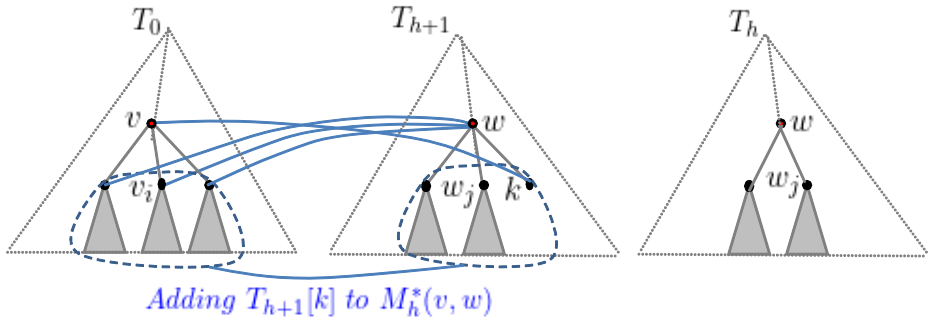
Re-evaluation of forest mappings One common problem to the 4 different cases enumerated above is linked with the re-evaluation of the forest mapping M^* at $h+1$ knowing the forest mapping at h. A careful analysis reveals that again, not all the computation needs be redone at step $h+1$. Two main cases arise, depending on whether the edition operation impacts the validity of the forest mapping, i.e., when $w = \mathcal{P}(k))$ at step h, or not, i.e., when w is an ancestor of $\mathcal{P}(k))$. This leads us to solve two different problems:

A. If the mapping at step h remains valid at step $h+1$, it may no longer be optimal. A new optimal valid forest mapping $M^*(v, w)$ needs to be computed;

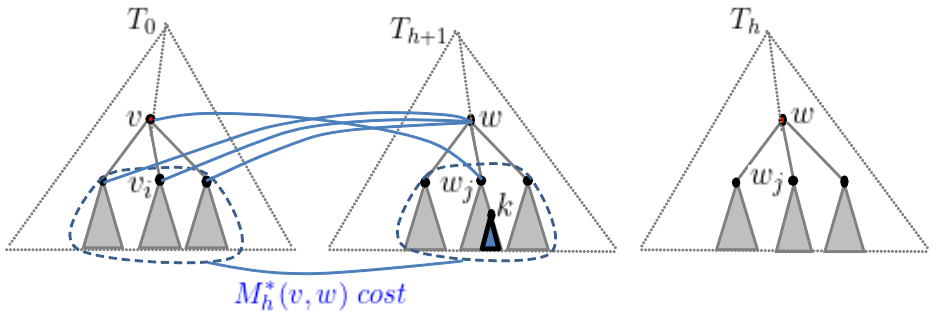
B. If not, we need to find an initial valid mapping. We will show that such a new valid mapping can efficiently be obtained from the optimal mapping at step h, and solve the optimal forest mapping, starting from this updated initial valid mapping.

The search of an optimal forest mapping can be solved as a MCF problem on a bipartite matching graph G representing the two forests [Zhang, 1996]. Let us study how the MCF problem can be solved in situation A, then B.

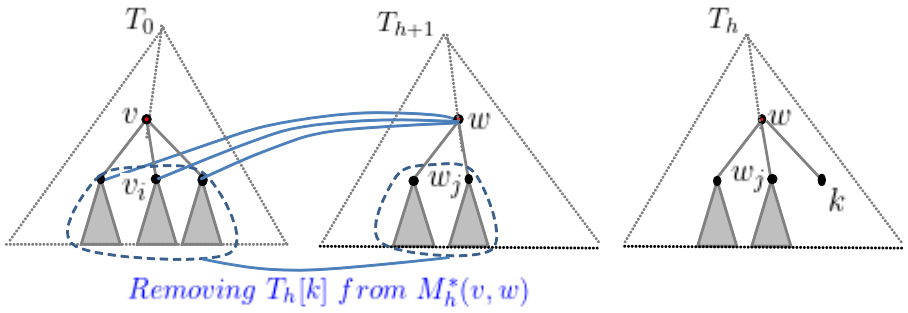
Resolution of problem A In this case, we already know a feasible flow f_h^* on G that corresponds to the optimal solution of the MCF problem at the previous step. However, this flow may have now lost its optimal character due to the cost changes in G. Given the limited change in the graph G between two consecutive steps, an obvious heuristics to optimize the search for a new optimal flow f_{h+1}^*, is to start the optimization from the previous optimal flow, that is feasible. For this, we chose to use the Network Simplex (NS) algorithm [Király and Kovács, 2012, Ahuja and Magnanti, 2018], that can be initialized by an arbitrary feasible flow, here chosen as f_h^* at step $h+1$.



(a)



(b)



(c)

Figure 6: The intermediate distance information -at connected vertices pairs- which are included in $\delta(v,w)$ computation and changing between step h and step $h+1$ after a vertex k edition. The variation of the (v,w) optimal forest mapping is depicted as follows: (a) After a leaf k insertion under vertex w, $T_{h+1}[k]$ has to be integrated in the previous mapping $M_h^*(v,w)$. (b) For any editing operation on k, if w is an ancestor of $\mathcal{P}(k)$ then the previous mapping $M_h^*(v,w)$ has to be updated according to the new costs $\delta(v_i,w_j)$ for w_j an ancestor of k and for any v_i a child of v. (c) After a leaf k deletion under vertex w, the previous mapping $M_h^*(v,w)$ has to be updated by excluding the subtree $T_h[k]$.

Resolution of problem B Here, edges costs are not changed in the graph G, but due to the edit operation (insertion or deletion of node k), the right-hand side of the graph G is modified (see Fig. 7). This may alter the feasibility of the previous flow f^* as well as its optimality. We must thus compute a new feasible flow (maximal and verifying the node law) from which we then derive an optimal flow. This can be simply done incrementally from the feasible flow f_h^* depending on the different cases illustrated in Fig. 7.

Case 1: n_k is inserted in previous graph G

All capacities on the new edges incident to of new node n_k are set to 1 while their initial flows are set to 0. Then the flow on edge (n_k,t) is obviously saturated. Two cases must be now considered:

- The node θ is on the right hand-side (see Fig. 7a). The previous capacity and flow on edge (θ,t) are decreased by one unit to match the demand value at t. Then, we select a node i, in the left-hand side of the bipartite graph G, among nodes for which the previous flow $f_{i\theta}^*$ was saturated and minimizing the cost $\gamma(i,n_k)$. And

by one unit we increase the flow on edge (i, n_k) while decreasing that on edge (i, θ), to verify the node law at nodes i, n_k and θ.

- The node θ is on the left hand-side (see Fig. 7b). The previous capacity and flow on edge (s, θ) are increased by one unit as well as the supply and demand values of respectively s and t. Then we augment the flow $f^*_{\theta n_k}$ by one unit, to verify the node law at n_k and θ.

Case 2: n_k is deleted from previous graph G

We model this situation by setting the flow from n_k to t and the capacities on all n_k incident edges to 0. We then consider the unique node i from the left-hand side that was previously connected to n_k with a unit flow. We must decrement the flow on this edge to verify the node law at n_k. Two cases must be now considered:

- The node θ is on the right hand-side (see Fig. 7c). The previous capacity and flow on edge (θ, t) are increased by one unit to match the t demand value. Then, we decrease the flow on edge (i, θ) to verify the node law at i and θ.
- The node θ is on the left hand-side (see Fig. 7d). The previous capacity and flow on edge (s, θ) are decreased by one unit as well as the supply and demand values of respectively s and t. If i is θ then the algorithm stops. Otherwise, we can pick a node j, in the right-hand side nodes, among those for which the previous flow $f^*_{\theta j}$ was saturated and minimizing the cost $\gamma(i, j)$. By one unit we increase the flow on edge (i, j) while decreasing that on edge (θ, j) to verify the node law at i and j.

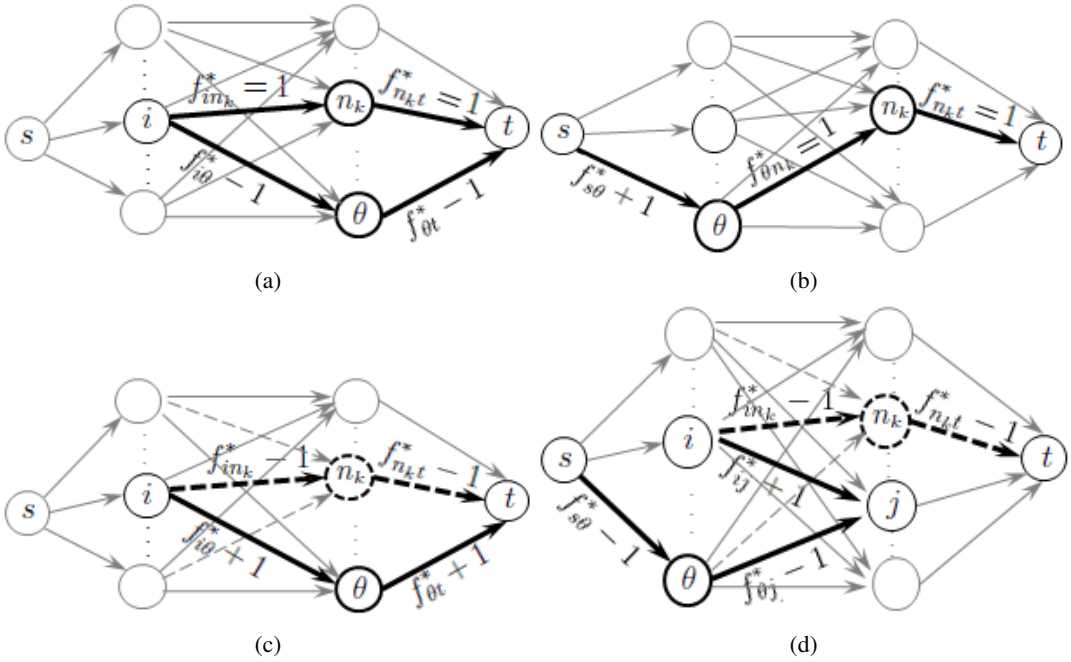


Figure 7: Examples of bipartite matching graphs G (mapping forests $F_0[v]$ and $F_{h+1}[w]$) of optimal flow f^* at step h and which are updated at level of a node n_k at step $h + 1$. Bold edges highlight edges on which the flow f^* is updated at step $h + 1$ to get a feasible one. Elements in dashed lines represent parts of G at step h to be deleted at step $h + 1$. The four examples differ in the θ node position (depending on the degrees of v and w) and on the operation performed on node n_k. (a) n_k is inserted in G and $\mathcal{D}(v) > \mathcal{D}(w)$. (b) n_k is inserted in G and either $\mathcal{D}(v) < \mathcal{D}(w)$ or $\mathcal{D}(v) = \mathcal{D}(w)$ and θ has just been inserted in G with incident edges of a null flow value. (c) n_k is about to be deleted from G and either $\mathcal{D}(v) > \mathcal{D}(w)$ or $\mathcal{D}(v) = \mathcal{D}(w)$ and θ has just been inserted in G with incident edges of a null flow value. (d) n_k is about to be deleted from G and $\mathcal{D}(v) < \mathcal{D}(w)$.

The resolutions of problems A and B complete the incremental computation of the edit-distance algorithm (a pseudo-code version of this algorithm is provided in Appendix A). We now can derive the overall computational complexity of the new incremental algorithm.

Proposition 1. *Let T_0 and T_h be two trees and T_{h+1} be a tree obtained from T_h using a single edit operation. Knowing the edit-distance between T_0 and T_h and the related distance information, the complexity of computing incrementally the edit distance between trees T_0 and T_{h+1} is in $O(\#T_0 \times \mathcal{H}(T_{h+1}) \times \mathcal{D}(T_{h+1}) \times \max(\mathcal{D}(T_0), \mathcal{D}(T_{h+1})))$.*

Proof. Altogether, the incremental edit distance algorithm browses the set of vertices in T_0 and updates the corresponding distance information for the edited node k and its parents. For this, the algorithm must repeatedly solve problems A and B for all the updated pairs (see Algorithm 1 in Appendix A). Let us first compute the complexities of algorithms solving problems A and B as defined above.

The complexity of problem A depends on the worst number of loops performed by the NS algorithm starting with a given initial feasible flow solution (the flow at previous step f^* that is feasible). Each NS loop transforms a feasible flow solution into another while decreasing its total cost until it becomes optimal. In the graph G mapping the forests rooted in v and w, let n be the G node belonging to the right hand side partition and representing a subtree rooted in the ancestor vertex of the edited vertex k in tree T_{h+1}, and let (i, n) be the unique saturated edge of the flow f_h^* at step h. According to the edit operations definition (given in Subsection 2.2), the costs on the n incoming edges are either incremented or decremented by one unit at step $h+1$ compared with step h. All other costs remain the same. Thus, at step $h+1$, either:

1. f_h^* was the unique minimal solution at step h:
   a. if the cost $\gamma(i, n)$ is decremented, it certainly remains minimal;
   b. if it is incremented, f_h^* remains minimal but it may no longer be the only one. Therefore, we can still write $f_{h+1}^* = f_h^*$.
2. f_h^* was not the unique minimal solution at step h. Let us denote $f_h'^*$ another solution for which the flow on the unique edge (i', n) is saturated, then either:
   a. the cost $\gamma(i, n)$ is decremented, thus f_h^* remains minimal and $f_{h+1}^* = f_h^*$;
   b. the cost $\gamma(i, n)$ is incremented and either:
      i. the cost $\gamma(i', n)$ is incremented, thus f_h^* remains the minimum and $f_{h+1}^* = f_h^*$;
      ii. the cost $\gamma(i', n)$ is decremented, and the flow $f_h'^*$ is the new optimal solution, $f_{h+1}^* = f_h'^*$.

As a consequence, we notice that for all these situations, the minimum flow remains unchanged between step h and $h+1$, except for the unique case 2.b.ii where the NS algorithm will perform a number of loops to find the flow $f_h'^*$ from f_h^*. Each loop swaps a subset of f_h^* saturated flows solutions by an other subset of saturated flows in $f_h'^*$ while decreasing the total cost on G. In the worst case the number of loops is bounded by the supply-demand value on G, namely $\max(\mathcal{D}(v), \mathcal{D}(w))$.

According to [Király and Kovács, 2012, Ahuja and Magnanti, 2018], each loop takes a worst time of $O(\mathcal{D}(v) \times \mathcal{D}(w))$. This gives an over all complexity of problem A of $O(\mathcal{D}(v) \times \mathcal{D}(w) \times \max(\mathcal{D}(v), \mathcal{D}(w)))$.

For problem B, recovering the flow feasibility on graph G mapping the forests rooted in v and w, as we describe below, takes a worst time of $O(\mathcal{D}(v) + \mathcal{D}(w))$.

Finally, the total complexity of the incremental edit distance algorithm is obtain by assembling these complexities together following Algorithm 1:

$$
\begin{aligned}
& \sum_{v \in T_0} O\left(\mathcal{D}(v) + \mathcal{D}(\mathcal{P}(k)) + \sum_{w \in \mathcal{P}^+(k)} \mathcal{D}(v) \times \mathcal{D}(w) \times \max(\mathcal{D}(v), \mathcal{D}(w)) \right) \\
= \; & O\left(\#T_0 + \#T_0 \times \mathcal{D}(T_{h+1}) + \sum_{v \in T_0} \sum_{w \in \mathcal{P}^+(k)} \mathcal{D}(v) \times \mathcal{D}(w) \times \max(\mathcal{D}(T_0), \mathcal{D}(T_{h+1})) \right) \\
= \; & O\left(\#T_0 + \#T_0 \times \mathcal{D}(T_{h+1}) + \max(\mathcal{D}(T_0), \mathcal{D}(T_{h+1})) \times \sum_{v \in T_0} \mathcal{D}(v) \times \sum_{w \in \mathcal{P}^+(k)} \mathcal{D}(w) \right) \\
= \; & O\left(\#T_0 + \#T_0 \times \mathcal{D}(T_{h+1}) + \max(\mathcal{D}(T_0), \mathcal{D}(T_{h+1})) \times \sum_{v \in T_0} \mathcal{D}(v) \times \mathcal{H}(T_{h+1}) \times \mathcal{D}(T_{h+1}) \right) \\
= \; & O\left(\#T_0 + \#T_0 \times \mathcal{D}(T_{h+1}) + \mathcal{H}(T_{h+1}) \times \mathcal{D}(T_{h+1}) \times \max(\mathcal{D}(T_0), \mathcal{D}(T_{h+1})) \times \#T_0 \right) \\
= \; & O(\#T_0 \times \mathcal{H}(T_{h+1}) \times \mathcal{D}(T_{h+1}) \times \max(\mathcal{D}(T_0), \mathcal{D}(T_{h+1}))),
\end{aligned}
$$

which states the result. □

4 Simulation study

The goal of this simulation study is twofold: first, we aim at proving the interest of the incremental algorithm in terms of computation time to evaluate the edit distance; second, we exploit the efficiency of this incremental algorithm to investigate in depth the escape rate of the isotropic random walk on unordered trees from intensive numerical experiments.

We ran all the simulations presented in this section in `Python3` on a Macbook Pro laptop running OSX High Sierra, with 2.9 GHz Intel Core i7 processors and 16 GB of RAM. The classical and incremental edit distance algorithms, as well as the random walk, have been implemented as a module of the `Python` library `treex` [Azaïs et al., 2019].

4.1 Computation time improvement

Starting from two random trees T and T' for which the edit distance has been pre-evaluated, we have computed $\delta(T, o(T'))$ using both Zhang algorithm and the incremental algorithm developed in this paper, for o picked at random in the set of all possible operations applicable to T', distinguishing the type of operation: add leaf, del leaf, add internal node, or del internal node. The boxplots of computations times required to evaluate $\delta(T, o(T'))$ are presented in Fig. 8 for different trees sizes. They have been estimated from $1\,000$ independent experiments for each size. The average computation times (over all the possible operations) as well as the average rates of improvement are given in Tab. 1.

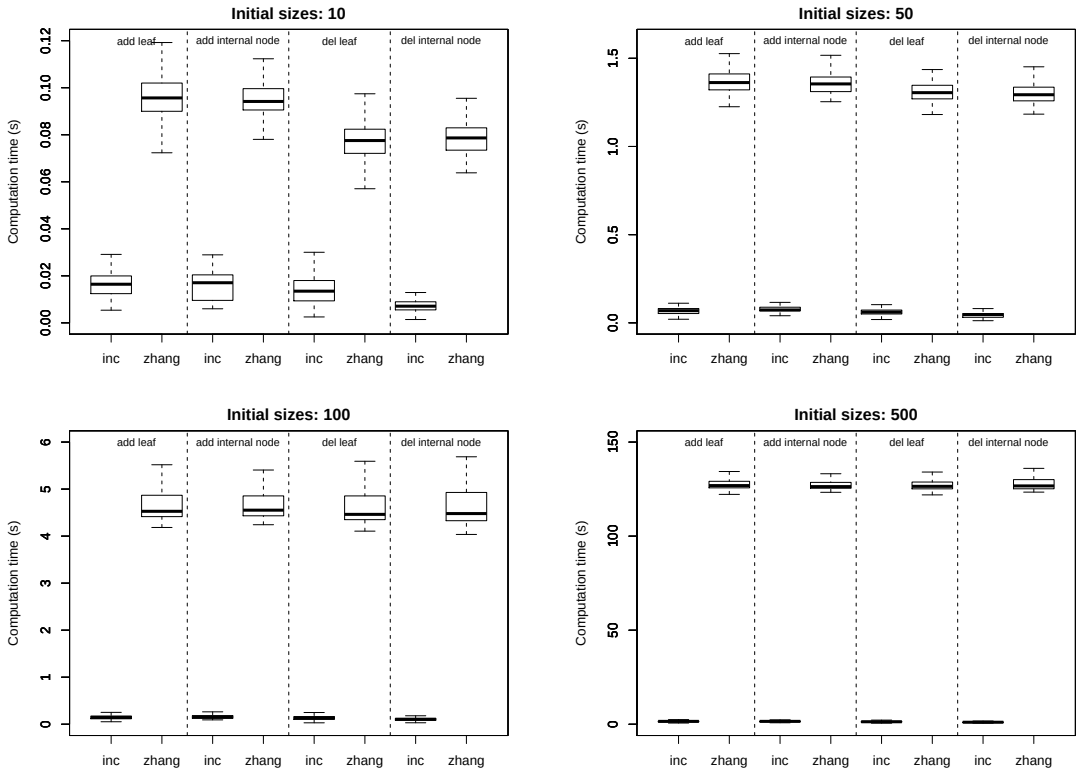


Figure 8: Computation times required for computing the edit distance between two trees of different sizes (top left: 10, top right: 50, bottom left: 100, and bottom right: 500) after editing the second one with one of the 4 possible operations, with the algorithm from the literature and the incremental one developed in this paper.

From Fig. 8, one can observe that the incremental algorithm is faster than Zhang algorithm, and the larger the trees, the (relatively) faster it is. One also remarks that the computation times are more predictable (in the sense that the boxplots are narrow) for the incremental algorithm. The gain in terms of computation time is already significant for trees of size 10, since the rate of improvement is of order 7 (from 15 to 84 ms on average). For trees of size 500, one sees a gain of almost two orders of magnitude (see Tab. 1). These numerical results show the great efficiency of our approach. In particular, simulating the trajectory of a random walk while evaluating the end-to-end distance is much

Trees size	10	50	100	500
Computation time (s) Zhang algorithm	0.08378765	1.337326	4.90168	127.9237
Computation time (s) Incremental algorithm	0.01455773	0.06493733	0.1449104	1.425516
Ratio of computation times "zhang/incremental"	6.99916	23.1193	38.00966	96.71445

Table 1: Average computation time (s) and rate of improvement for the incremental algorithm compared to Zhang algorithm for trees of different sizes. Estimates from 1 000 independent replicates. This table compiles the data of Fig. 8, which present them for the different edit operations.

faster using the incremental algorithm: for example, 14.9 (Zhang algorithm) vs. 1.8 s (incremental algorithm) for 200 steps starting from a tree of size 10. See Tab. 2 for average computation times of random walk simulations.

	25 steps	50 steps	100 steps	200 steps
Computation time (s) Zhang algorithm	1.591511	3.199535	5.190579	14.883755
Computation time (s) Incremental algorithm	0.2946835	0.4631565	0.7088750	1.7541514

Table 2: Average computation time (s) to simulate and evaluate the end-to-end distance of a random walk starting from a tree of size 10. Estimates from 100 independent replicates.

4.2 Sharp estimation of average escape rate

Fig. 9 presents a sample trajectory starting from an initial tree of size 10 within a time window of 10 000 steps. The 5 characteristics introduced in Subsection 2.1, namely the end-to-end distance, the size, the height, the outdegree, and the Strahler number, as well as their average behavior estimated from 1 000 replicates, are given. For all of them, one observes a slowing escape towards infinity, which results in concave curves. The aim of this subsection is to estimate, as precisely as possible, the average escape rate of the end-to-end distance thanks to intensive simulations.

The end-to-end distance can be studied through the evoluation of the size. Given any trees T and T', the distance between them can be bounded from their sizes as

$$|\#T' - \#T| \leq \delta(T, T') \leq \#T' + \#T. \tag{4}$$

The upper-bound corresponds to the worst case where all the vertices of T have been deleted before adding all the vertices of T'. The lower bound is obtained in the very favorable case where T' can be obtained from T by only adding (or deleting) vertices. From (4) and given a random walk on trees $(T_h)_{h\geq 0}$, we obtain the following relation between the size and end-to-end distance processes,

$$1 - \frac{\#T_0}{\#T_t} \leq \frac{\delta(T_0, T_t)}{\#T_t} \leq 1 + \frac{\#T_0}{\#T_t}. \tag{5}$$

However, it should be noted that the size process $(\#T_t)_{t\geq 0}$ is a simple random walk on integers reflected at the barrier 1, i.e., a $\mathbb{N}^*$-valued process such that,

$$\forall\, n \geq 2,\ \mathbb{P}(\#T_{h+1} = n+1 | \#T_h = n) = \mathbb{P}(\#T_{h+1} = n-1 | \#T_h = n) = \frac{1}{2},$$

and $\mathbb{P}(\#T_{h+1} = 2 | \#T_h = 1) = 1$. The average escape rate of a (reflected in 0) random walk $(W_h)_{h\geq 0}$ has been studied in the literature. Taking $p_1 = 1$ in [Katriel, 2011, eq. (1.2)], we obtain that, when h goes to infinity, $\mathbb{E}[W_h | W_0 = x] \sim \rho(x, h)$, where

$$\rho(x, h) = \sqrt{\frac{2h}{\pi}} + \frac{x^2}{\sqrt{2\pi h}}. \tag{6}$$

Together with (5), and using the fact that $\#T_h$ can be expressed as $W_h + 1$, this proves that, when h goes to infinity,

$$\mathbb{E}[\delta(T_0, T_h) | \#T_0 = x] \sim \rho(x, h). \tag{7}$$

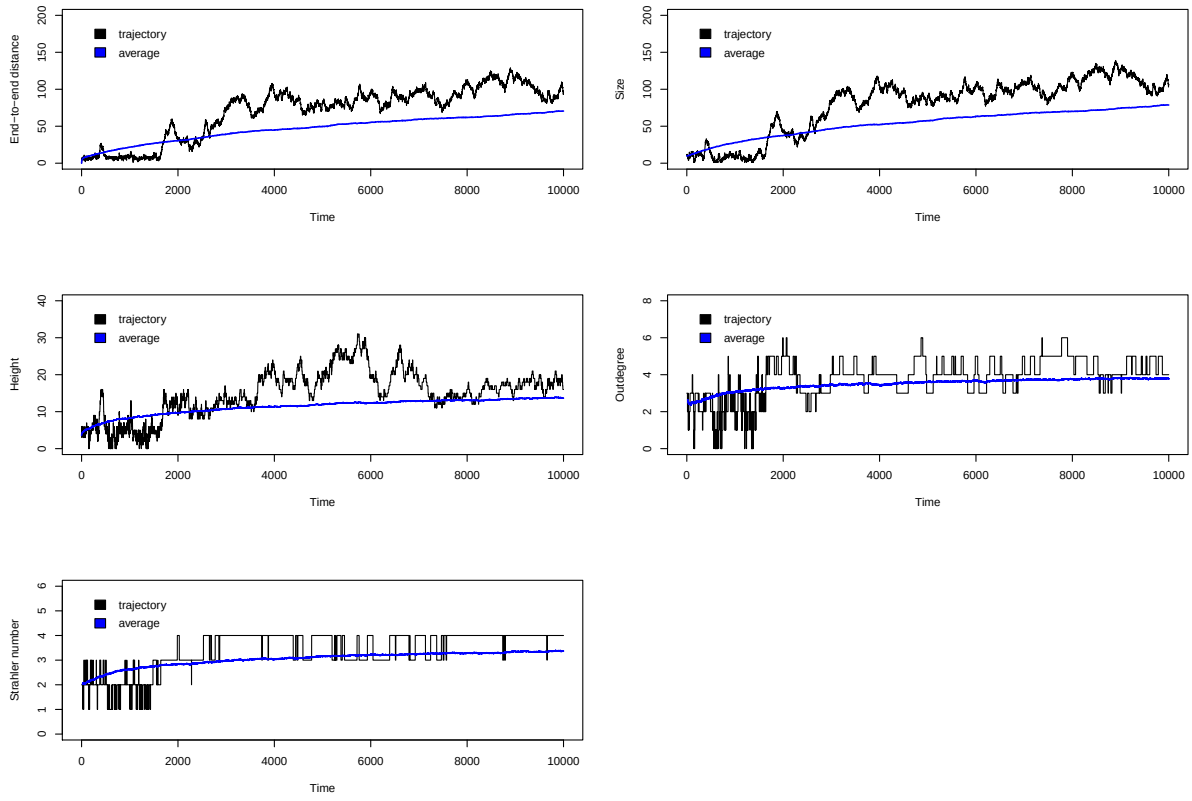


Figure 9: 5 characteristics of a sample trajectory of the random walk starting from an initial tree of size 10 up to time 10 000 (in black) and average curves (in blue, estimated from 1 000 independent replicates): end-to-end distance (top left), size (top right), height (middle left), outdegree (middle right), and Strahler number (bottom left). See Subsection 2.1 for definitions.

We aim to improve our empirical knowledge of this rate thanks to intensive simulations of random walks, in particular through the study of the dependency on the initial condition x.

In order to refine the escape rate (7), we have performed intensive simulations of random walks, within a time window of 10 000 steps repeated 1 000 times to achieve accurate average behaviors, starting from 9 possible initial sizes (from 2 to 10 nodes). Consequently, 90M evaluations of distances have been completed to conduct this study.

Fig. 10 presents the average size and end-to-end distance processes starting from initial trees of different sizes. We remark that the theoretical rate (6) is accurate for the size process as expected, but not for the end-to-end distance, in particular when the size of the initial condition $\#T_0$ increases. It is not surprising in regards to (5) that introduces an error of order $O(\#T_0/\#T_h)$.

Instead of comparing $\delta(T_0, T_h)$ with $\#T_h$, we propose in Fig. 11 to investigate the approximation accuracy of the two bounds given in (4), i.e., compare $\delta(T_0, T_h)$ with the lower-bound $\#T_h - \#T_0$ and with the upper-bound $\#T_h + \#T_0$. The numerical simulations empirically show that $\#T_h - \#T_0$ is the best approximation of the the distance process, which tends to prove that the random walk keeps a long-term memory of the starting tree. Indeed and as aforementioned, $\#T_h - \#T_0$ is equal to the distance when nodes have been only added to the initial structure, without removing any part of it.

We also remark from Fig. 11 that $\delta(T_0, T_h)$ is less accurately estimated by $\#T_h - \#T_0$ when $\#T_0$ increases, which says that the memory of the starting tree is less preserved when it is larger. We propose to estimate this approximation error through a polynomial fitting presented in Fig. 12: (i) at each time step h, we observe that the ratio $\mathbb{E}[\delta(T_0, T_h)|\#T_0 = x]/(\mathbb{E}[\#T_h|\#T_0 = x] - \#T_0)$ is surprisingly well-captured by a function of the form $1 + \alpha_t \#T_0^2$ (3 examples are given in Fig. 12 top left, top right, and bottom left); (ii) we show that the regression coefficients α_h behave as $\beta/h^{5/4}$,

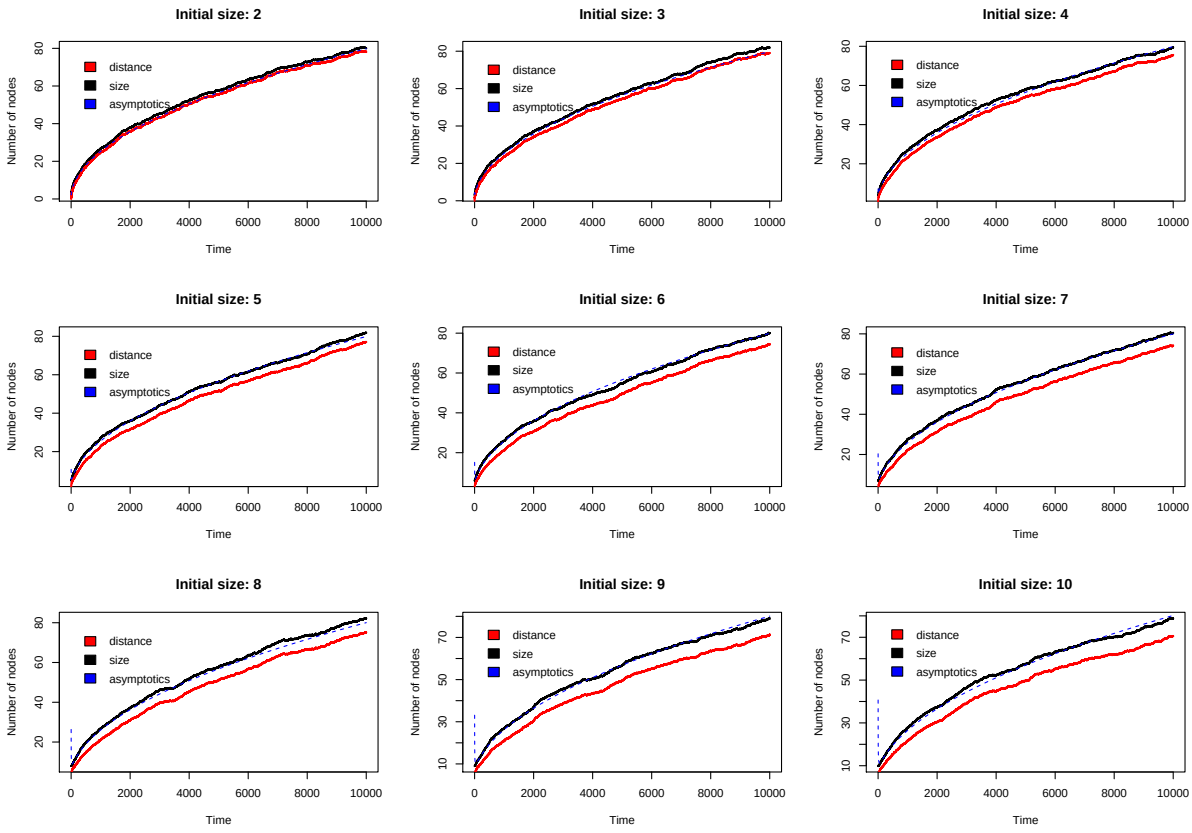


Figure 10: Average size process $(\#T_h)_{h\geq 0}$ (in black) and average end-to-end distance process $(\delta(T_0, T_h))_{h\geq 0}$ (in red) starting from trees of size between 2 and 10. Each curve has been estimated from 1 000 independent replicates of the random walk. The theoretical asymptotics $\rho(\#T_0, h)$, see (6), of the size process has been added in blue dashed lines.

with $\beta = 17.83953$ (see Fig. 12 bottom right). As a consequence, we obtain this estimated but sharp rate,

$$\frac{\mathbb{E}[\delta(T_0, T_h)|\#T_0 = x]}{\mathbb{E}[\#T_h|\#T_0 = x] - \#T_0} \sim 1 + \frac{\beta\,\#T_0^2}{h^{5/4}}.$$

Finally, together with all of the above, our estimated escape rate of the end-to-end distance process is given by

$$\mathbb{E}[\delta(T_0, T_h)|\#T_0 = x] \sim \sqrt{\frac{2h}{\pi}} - \#T_0 + \frac{\#T_0^2}{\sqrt{2\pi h}} + \frac{\beta\,\#T_0^2}{h^{3/4}}\sqrt{\frac{2}{\pi}} - \frac{\beta\,\#T_0^3}{h^{5/4}} + \frac{\beta\,\#T_0^4}{\sqrt{2\pi}\,h^{7/4}}, \tag{8}$$

which accuracy is illustrated in Fig. 13.

References

[Aho et al., 1974] Aho, A. V., Hopcroft, J. E., and Ullman, J. D. (1974). *The Design and Analysis of Computer Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

[Ahuja and Magnanti, 2018] Ahuja, R. K. and Magnanti, T. L. (2018). *Network flows*. Franklin Classics.

[Akutsu et al., 2011] Akutsu, T., Fukagawa, D., Takasu, A., and Tamura, T. (2011). Exact algorithms for computing the tree edit distance between unordered trees. *Theoretical Computer Science*, 412(4):352–364.

[Azaïs et al., 2019] Azaïs, R., Cerutti, G., Gemmerle;, D., and Ingels, F. (2019). treex: a python package for manipulating rooted trees. *Journal of Open Source Software*, 4(38):1351.

[Bille, 2005] Bille, P. (2005). A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1-3):217–239.

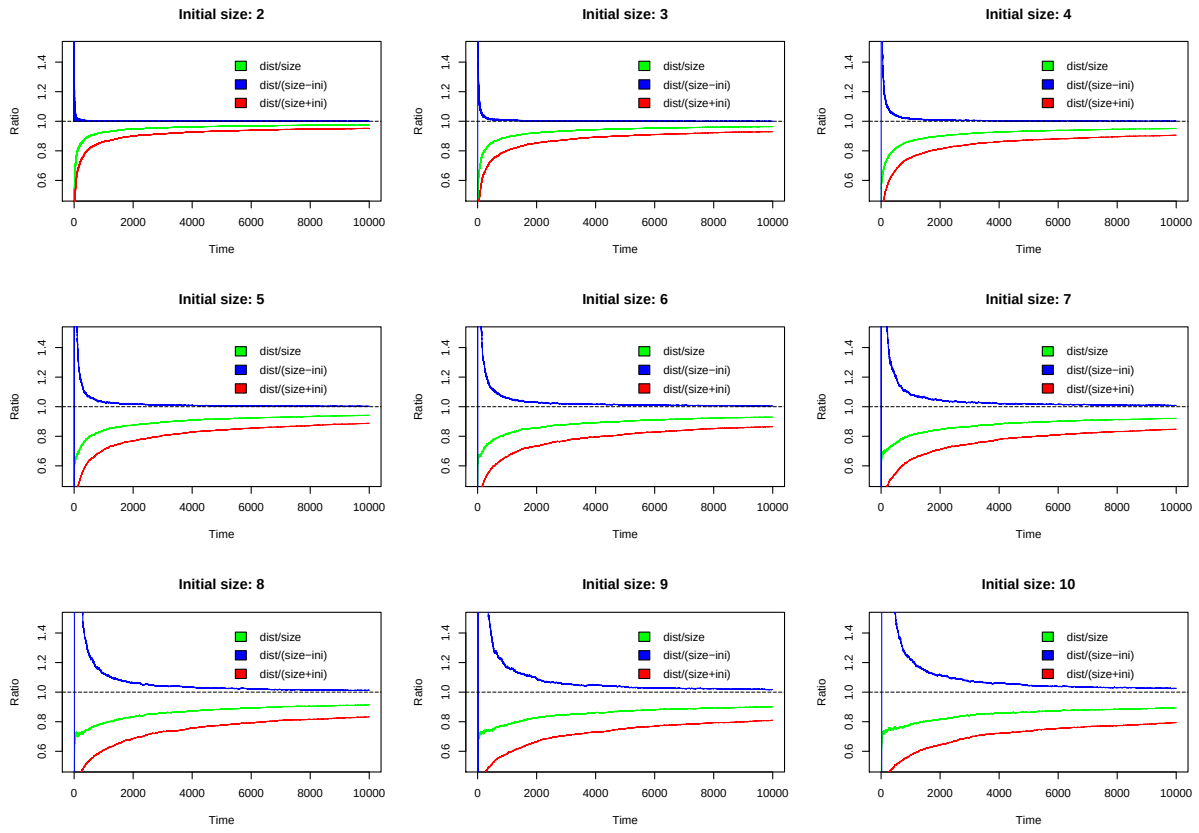


Figure 11: Ratio of average end-to-end distance process $(\delta(T_0, T_h))_{h\geq 0}$ over (i) average size process $(\#T_h)_{h\geq 0}$ (in green), (ii) average size process minus initial size $(\#T_h - \#T_0)_{h\geq 0}$ (in blue), and (iii) average size process plus initial size $(\#T_h + \#T_0)_{h\geq 0}$ (in red).

[Edmonds and Karp, 1972] Edmonds, J. and Karp, R. M. (1972). Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264.

[Edmonds and Karp, 2003] Edmonds, J. and Karp, R. M. (2003). Theoretical improvements in algorithmic efficiency for network flow problems. In *Combinatorial Optimization—Eureka, You Shrink!*, pages 31–33. Springer.

[Göbel and Jagers, 1974] Göbel, F. and Jagers, A. (1974). Random walks on graphs. *Stochastic Processes and their Applications*, 2(4):311 – 336.

[Goldberg and Tarjan, 1990] Goldberg, A. V. and Tarjan, R. E. (1990). Finding minimum-cost circulations by successive approximation. *Mathematics of Operations Research*, 15(3):430–466.

[Gouëzel, 2017] Gouëzel, S. (2017). Analyticity of the entropy and the escape rate of random walks in hyperbolic groups. *Discrete Analysis*, 2017(7).

[Katriel, 2011] Katriel, G. (2011). Asymptotic behavior of random walks on a half-line with a jump at the origin. *arXiv:1108.5621*.

[Király and Kovács, 2012] Király, Z. and Kovács, P. (2012). Efficient implementations of minimum-cost flow algorithms. *arXiv preprint arXiv:1207.6381*.

[Lu, 1979] Lu, S. (1979). A tree-to-tree distance and its application to cluster analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):219–224.

[Mazur, 1965] Mazur, J. (1965). Distribution function of the end-to-end distances of linear polymers with excluded volume effects. *Journal of research of the National Bureau of Standards. Section A, Physics and chemistry*, 69A:355–363.

[Tarjan, 1983] Tarjan, R. E. (1983). *Data structures and network algorithms*. Society for industrial and Applied Mathematics.

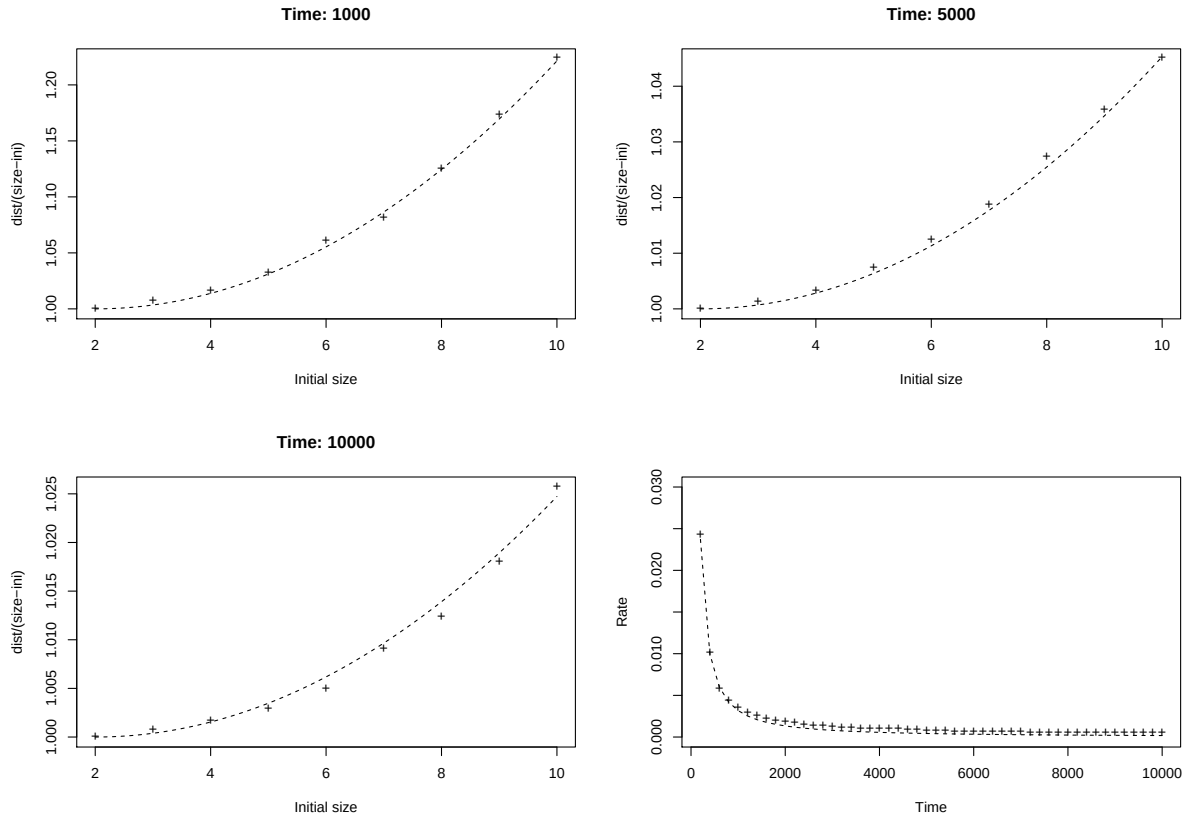


Figure 12: Ratio of average end-to-end distance process $(\delta(T_0, T_h))_{h\geq 0}$ over average size process minus initial size $(\#T_h - \#T_0)_{h\geq 0}$ as a function of the initial tree size (crosses) and their quadratic fitting of the form $1 + \alpha_h \#T_0^2$ (dashed lines), at 3 time steps (top left: $h = 1\,000$, top right: $h = 5\,000$, and bottom left: $h = 10\,000$). Fitting coefficients α_h as a function of time h (crosses) and fitting of the form $\beta/h^{5/4}$ (dashed line) with $\beta = 17.83953$ (bottom right).

[Woess, 2000] Woess, W. (2000). *Random Walks on Infinite Graphs and Groups*. Cambridge Tracts in Mathematics. Cambridge University Press.

[Zhang, 1996] Zhang, K. (1996). A constrained edit distance between unordered labeled trees. *Algorithmica*, 15(3):205–222.

[Zhang et al., 1992] Zhang, K., Statman, R., and Shasha, D. (1992). On the editing distance between unordered labeled trees. *Inf. Process. Lett.*, 42(3):133–139.

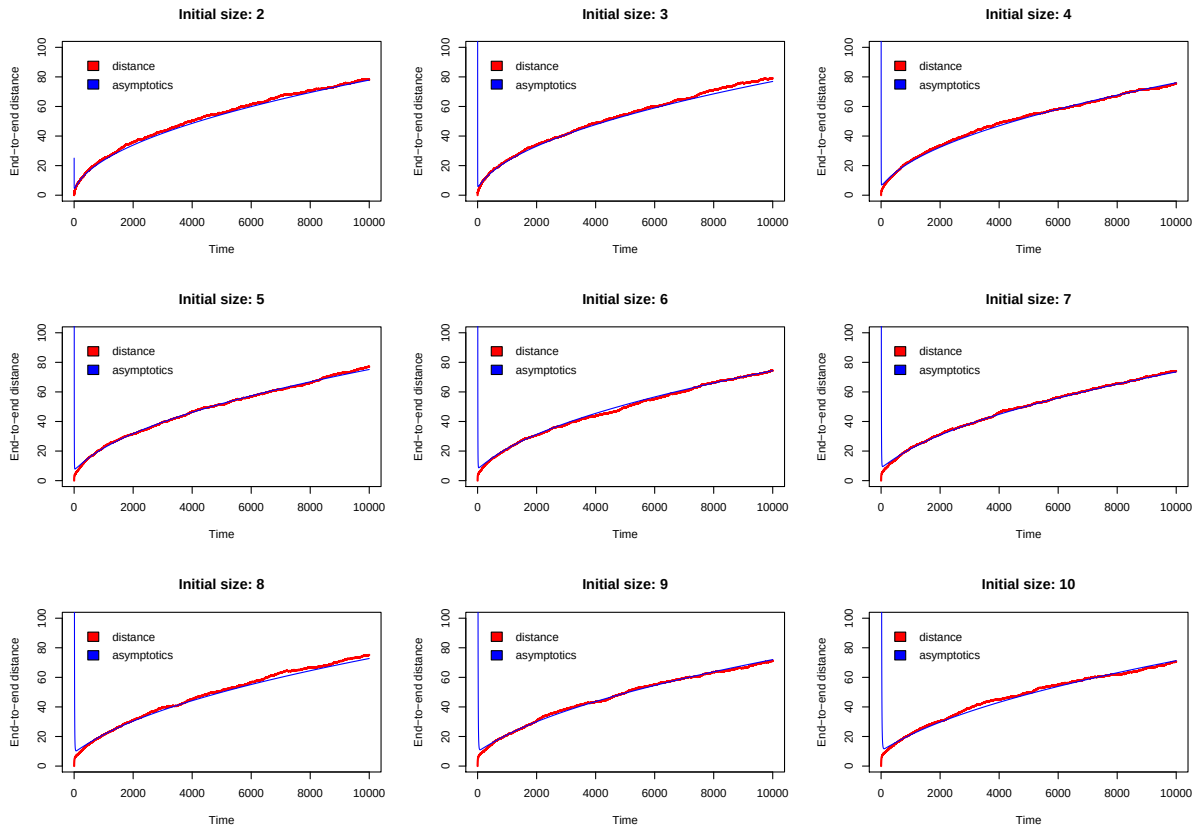


Figure 13: Average end-to-end distance process $(\delta(T_0, T_h))_{h\geq 0}$ (in red) starting from trees of size between 2 and 10, and estimated asymptotics (in blue) given in (8).

A Pseudo-code of the incremental edit distance algorithm

Algorithm 1: The incremental computation of edit distance between two trees T_0 and T_{h+1}.

Function incremental edit distance

Input :
– Two unordered rooted trees T_0 and T_{h+1}
– The distance information computed for $\delta(T_0, T_h)$
– The edited vertex k in T_{h+1}

Output:
– The new distance information computed for $\delta(T_0, T_{h+1})$.

for $v \in T_0$ **do**
  Compute the distance information at (v, k) according to the type of edit-operation on k (see Subsection 3.2)
  Update the distance information at $(v, \mathcal{P}(k))$:
    solve problem B (to restore a valid forest mapping between $F_{h+1}[v]$ and $F_{h+1}[\mathcal{P}(k)]$)
    solve problem A (to restore an optimal forest mapping between $F_{h+1}[v]$ and $F_{h+1}[\mathcal{P}(k)]$)
  for $w \in \mathcal{P}^+(\mathcal{P}(k))$ **do**
    Update the distance information at (v, w):
      solve problem A to restore an optimal forest mapping between $F_{h+1}[v]$ and $F_{h+1}[w]$)

return The distance information of $\delta(T_0, T_{h+1})$