Process Model Comparison Based on Cophenetic Distance

David Sánchez-Charles[1], Victor Muntés-Mulero[1], Josep Carmona[2]

[1] CA Strategic Research Labs, CA Technologies, Spain
{David.Sanchez,Victor.Muntes}@ca.com
[2] Universitat Politècnica de Catalunya, Spain
jcarmona@cs.upc.edu

Summary. The automated comparison of process models has received increasing attention in the last decade, due to the growing existence of process models and repositories, and the consequent need to assess similarities between the underlying processes. Current techniques for process model comparison are either structural (based on graph edit distances), or behavioural (through activity profiles or the analysis of the execution semantics). Accordingly, there is a gap between the quality of the information provided by these two families, i.e., structural techniques may be fast but inaccurate, whilst behavioural are accurate but complex. In this paper we present a novel technique, that is based on a well-known technique to compare labeled trees through the notion of *Cophenetic distance*. The technique lays between the two families of methods for comparing a process model: it has an structural nature, but can provide accurate information on the differences/similarities of two process models. The experimental evaluation on various benchmarks sets are reported, that position the proposed technique as a valuable tool for process model comparison.

1 Introduction

Nowadays process models are ubiquitous objects in companies and organizations. They are becoming precious for representing unambiguous and detailed descriptions of real processes. On the one hand, *BPMS* platforms, which allow designing, deploying and managing the processes in organizations, are based on process models. On the other hand, evidence-based process models (i.e., process models with a high alignment with respect to the underlying real process) can be used to analyze the process formally, e.g., detecting inconsistencies or performance problems that may hamper the correct and optimal execution of the process. Furthermore, the existence of environments for creating, managing and querying *process model collections* enable the hierarchical and cross-organizational analysis, with process models as atomic objects.

A core technique necessary in many of the aforementioned situations is the automated comparison of process models. Due to its importance, this problem has received significant attention in the BPM field, which can be split into *structural techniques* based on graph-edit distance [8, 9, 10, 18], and *behavioural techniques* that focus on the execution semantics or behavioural relations of the corresponding models [1, 7, 13, 16, 17]. Intuitively, structural techniques are fast but inaccurate (in terms of the differences found), whereas pure behavioural techniques are complex (both in computation time and memory usage) but accurate.

The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-45468-9_9

In this paper we propose a novel method to compare process models[1]. The technique is based on a recent algorithm [5] from the field of *computational phylogenetics*, where the objects to compare are labeled trees showing the inferred evolutionary relationships among various biological species. We adapt the algorithm to the BPM context, thus using *process trees* [4] as notation. Our proposed similarity metric sits halfway between pure structural similarity methods (inheriting their low complexity features), and behavioural similarity metrics (capable of providing similar behavioural information). Moreover, the performance of our approach allow us to consider this metric for large process models.

The paper is organized as follows: next section provides an intuition of the metric over a realistic example. In Section 3 the necessary preliminaries are provided. Then in Section 4 we present the main contribution of the paper: a similarity metric for deterministic process trees. The deterministic restriction is dropped in Section 5, giving rise to a heuristic technique that relies on an approximate matching algorithm. The techniques of the paper are evaluated thoroughly in Section 6. Finally, Section 7 concludes the paper and provides pointers for future investigations.

2 Motivating Example

Let us use a real-life example to motivate the contributions of this paper. A product manager decides to monitor all accesses to an SVN repository. Those accesses are done through HTTP/S, as specified in the WebDAV/DeltaV protocol. It turns out that those read and write requests over HTTP/S can be translated to human-friendly SVN commands such as *svn update* or *svn commit*. Continuing the work done by Li Sun et.al. [15], the product manager plans to model the developer's behaviour based on the SVN commands they execute daily. Our goal is to measure the differences between those models, inducing a behavioural distance between individuals. This way, intruder attacks to the repository can be detected globally by analyzing process behaviour that is clearly separated from the rest.

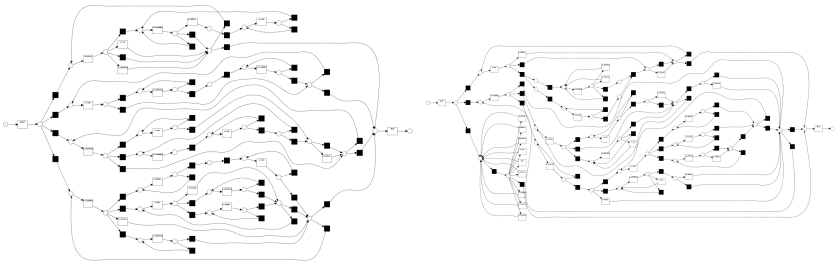


Fig. 1. Two process models describing how two users access an SVN repository.

[1] We assume the problem of dealing with real activity labels, e.g., when the name of an activity in the models does not perfectly match, is resolved prior to the techniques of this paper.

Figure 1 depicts the access behaviour of two users to the same repository, using a block-structured process discovery algorithm[2]. In our preliminary study, the process model of an average user shows lots of concurrency, duplicate activities and iterative behaviour[3]. Existing behavioural comparison techniques struggle when dealing with such models. Either they fall short in describing duplicate activities and loops [16], or the underlying technique does not scale in the presence of concurrent process branches [1].

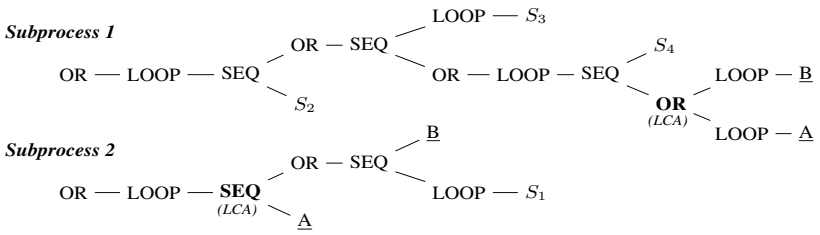


Fig. 2. Extract of the tree representation of the two processes in Figure 1. Only the subtrees related to two common activities A and B are represented, and their least common ancestors are depicted in bold. Activities S_i are unrelated to A and B.

The approach presented in this paper evaluates the difference of the two minimum subtrees containing a selected pair of activities, and extends the comparison to all possible pairs. Analysis over such subtrees is expected to be more simple and efficient, while still capable of comparing both the structure and behaviour of the two processes. See Figure 2 for an example, which focuses on activities A and B in both models. One can check that the difference between the depth of the two activities is an approximation to the graph distance between those two models. For instance, in Figure 2, depths of A are 11 in the first subprocess and 4 in the second subprocess, whilst depths of B are 11 and 6. The difference of their depths sum 12, implying that 8 nodes must be removed and 2 extra edges are needed in order to transform one model into the other. Besides, and more importantly, one can see that the common ancestors of activities A and B in Figure 2 model two different behaviours: On the first subprocess, activities A and B are mutually exclusive; On the second, A is executed after activity B. Notice that the depth of this common ancestor also highlights how long it takes to make the *behavioural decision* of how activities A and B relate to each other. Therefore, by incorporating these notions into the distance function, we would be able to not only measure structural differences but also highlight differences in the behaviour of two process models. For instance, one could obtain the sentence: *Activities A and B in Figure 2 are mutually exclusive in the first subprocess, but activity A always occurs after B in the second subprocess. Besides, the behavioural decision in the first subprocess is done* 6 *steps after the decision is taken in the second subprocess*.

[2] We used *Discover a Process Tree using Inductive Miner (ProM 6.5)* and then converted them to Petri Nets.

[3] The most common sequence of commands in the dataset is *svn -options*, *svn update*, *svn -options* indicating they use an IDE that overwrites the SVN options just to perform an update and then returns to its previous status.

3 Background

3.1 Process Trees: a Tree-like Representation of Business Processes

A **rooted tree** is a directed graph with a distinguished node, called the root, from which every node can be reached with exactly one path. A **weighted rooted tree** is a pair (T, ω) consisting of a rooted tree T and a weight function $\omega : E \to \mathbb{R}_{>0}$ that associates every arc $e \in E$ a non-negative real number $\omega(e) > 0$. A **labeled rooted tree** is a rooted tree T such that there exists a mapping between a subset of the nodes of the tree and a set of labels S.

Let $T = (V, E)$ be a rooted tree. Whenever $(u, v) \in E$, we say that v is a child of u and that u is the parent of v. The nodes without children are the leaves of the tree, and the other nodes are called internal. Whenever there exists a path from a node u to a node v, we say that v is a descendant of u and also that u is an ancestor of v. An internal node is **elementary** if it only has one child. The **depth** of a node u in a tree T, denoted by $\delta_T(u)$, is the sum of the weights of the arcs in the path from the root to u. Weights are usually set to 1, but we will later see that we can encode behavioural information from the process by modifying these weights.

Definition 1 ([4]). *A **process tree** is a labeled rooted tree T in which activities are represented as leaves of the tree and internal nodes describe the control-flow of the process.*

We say that a process tree is **deterministic** if there is a one-to-one mapping between activity labels and leaves of T. For the sake of simplicity, we will label internal labels as *OR*[4], *AND*, *SEQ* and *LOOP* to represent the usual behavioural structures in a process model. We will also denote these internal nodes by **gateways**, following the BPMN nomenclature. We allow silent activities by labeling them as $\emptyset$.

Definition 2. *A process tree is **reducible** if there are elementary nodes, silent transitions hanging over a gateway other than OR, or there exist a pair of internal nodes u and v such that (u, v) is an edge in the graph and both model the same type of gateway.*

Any *reducible* process tree can be converted into an *irreducible* tree by merging all conflicting nodes. We will suppose that all process trees are given in its irreducible form. Figure 3 depicts an example of a reducible process tree and its irreducible counterpart.

Fig. 3. Two process trees modeling exactly the same behaviour. The left model is reducible, and the right model is its irreducible representation. The silent transition $\emptyset$ is removed because it is not part of an OR structure. The OR elementary node does not provide behavioural information.

[4] Following the semantics of block-structured models in [4], only exclusive ORs are modeled.

3.2 Cophenetic vectors

The **least common ancestor** (LCA) of a pair of nodes u and v of a rooted tree T, denoted by $[u, v]_T$, is the unique common ancestor of them that is a descendant of every other common ancestor. The definition of the Cophenetic vector is based on the discrepancies on the depth of the LCA of every pair of activities.

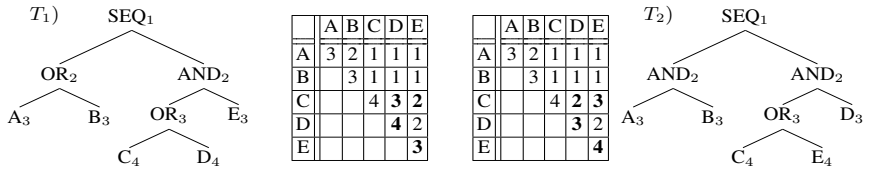
Definition 3 ([14]). *Let S be the set of labels of a weighted labeled rooted tree T. For every pair of different labels i, j, their Cophenetic value is*

$$\varphi_T(i, j) = \delta_T([u, v]_T) \qquad u, v \text{ have labels } i, j$$

To simplify notation, we denote the depth of a node with label i by $\varphi_T(i, i)$, and $\varphi_T(i, j) = 0$ if either i or j are not activities of the process tree T.

Definition 4. *Let T be a weighted rooted tree, and S the set of activity labels of the tree T, its **Cophenetic vector** is*

$$\varphi(T) = (\varphi_T(i, j))_{i,j \in S}$$



	A	B	C	D	E
A	3	2	1	1	1
B		3	1	1	1
C			4	**3**	**2**
D				**4**	2
E					**3**

	A	B	C	D	E
A	3	2	1	1	1
B		3	1	1	1
C			4	**2**	**3**
D				**3**	2
E					**4**

Fig. 4. Example of process trees and their Cophenetic vector (in matrix representation), assuming the depth of the root is 1. For simplicity, we included node's depth as a subscript of the label. For instance, the LCA of activities C and E in T_1 is the *AND* gateway that is one children of the root and, hence, its Cophenetic value is 2.

In an already fifty years old paper [14], Sokal and Rohlf proposed the use of the cophenetic values to compare dendrograms. Authors in [5] show that cophenetic values can also be applied to uniquely project labelled trees into a multidimensional vector space, allowing them to define a distance on labelled trees as Theorem 1 states.

Theorem 1 ([5]). *Two weighted labeled trees without elementary nodes, unlabeled leaves nor repeated labels are equal if, and only if, they share the same Cophenetic vector.*

Cophenetic vectors are not enough for determining process tree similarity: for instance, in Figure 4 if the *OR* and *AND* labels of the left tree are interchanged, the Cophenetic vectors of both trees are equal whilst the behaviour represented is different. Besides, constraints in Theorem 1 do not allow models with multiple silent transitions. Next section shows how to transform process trees in order to overcome this limitation.

4 Distance between Deterministic Process Trees

4.1 Cophenetic Distance definition

As we have seen, Cophenetic values unequivocally represents weighted labeled rooted trees. As it is well known, this allows to induce distance metrics in the set of labeled trees. Let $dist$ be any distance between two points in a vectorial space, we define

$$d(T, T') = dist(\varphi(T), \varphi(T'))$$

as the distance between two trees. For instance, by using the L^1-norm we get

$$d_1(T, T') = \sum_{i,j \in S} |\varphi_T(i, j) - \varphi_{T'}(i, j)|$$

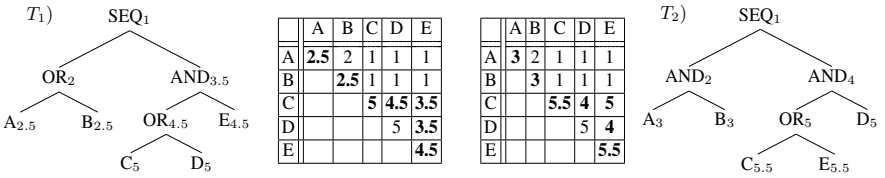
The Cophenetic values were originally conceived to measure structural differences between the leaves of two dendograms, but we can extend its use to deterministic process trees thanks to Theorem 1. This result allow us to modify the depth of each node in order to model the path of gateways we are tracing from the root to activities (the leaves of the tree). In Definition 5 we propose a depth function to overcome the following weaknesses of the original Cophenetic distance over labelled trees: (1) ensures that non-common activities increases the distance between two models; (2) depth of activities in a sequential order increase in the same sequential order, modeling the complexity of the blocks already seen by the process; (3) allows for silent transitions; and (4) differentiates two processes with the same structure but modeling different gateways at the root.

Definition 5. *Let T be a deterministic process tree. We define the depth function δ'_T as follows:*

1. *Root node has depth* 1.
2. *Iterate over all nodes in a pre-order traversal.*
3. *The depth of all nodes is* 1 *plus the depth of its parent, except*
   a) *If the parent is an OR clause, increase* 0.5 *instead of* 1.
   b) *If the activity is silent, increase* 0.25 *the depth of the parent and any other sibling. Afterwards, remove the silent activity.*
   c) *If the parent is the start of a LOOP, increase also by the maximum depth of the underlying tree.*
   d) *If the parent is a SEQ gateway, consider the depth of deepest visited children of the node's siblings instead of the parent.*
4. *Any remaining elementary node will be removed, and its parent and children will be directly connected.*

For the sake of simplicity, $trf(T)$ will denote the combination of the tree T with the aforementioned depth function δ_T.

Figure 5 depicts the transformation of the two processes in Figure 4. With the aforementioned depth function, Cophenetic values now highlight, for example, differences in the two activities A and B due to the behavioural change of their parent node. This transformation allow us to overcome the limitations of Theorem 1, since silent transitions are allowed, but also by ordering children of sequential gateways. As we state in Theorem 2, this transformation uniquely represents deterministic process trees.



	A	B	C	D	E
A	**2.5**	2	1	1	1
B		**2.5**	1	1	1
C			**5**	**4.5**	**3.5**
D				5	**3.5**
E					**4.5**

	A	B	C	D	E
A	**3**	2	1	1	1
B		**3**	1	1	1
C			**5.5**	**4**	**5**
D				5	**4**
E					**5.5**

Fig. 5. Transformation of the process trees in Figure 4. For the sake of simplicity, we included node's depth as a subscript of the label. For instance, depth of the *AND* gateway in T_1 is 3.5 because its parent represents a sequence and the maximum depth of the previous processed branch is 2.5

Theorem 2. *Let T and T' be two deterministic process trees. If $trf(T)$ and $trf(T')$ share the same Cophenetic vector, then T and T' are the same process tree.*

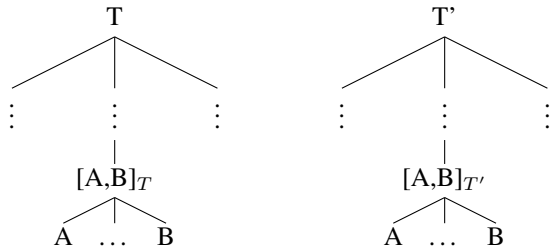
This theorem shows that Theorem 1 is also applicable to the new depth definition, and therefore useful for checking equality of two process trees and measuring differences between the models. The proof of this theorem is based on the observation that the Cophenetic values of any subtree are highly related to the Cophenetic values of the complete tree, as Lemma 1 shows. Details of the proof of this lemma are omitted, but it is a direct consequence of the pre-order traversal approach of Definition 5.

Lemma 1. *Let T be a weighted rooted tree, and S a subtree of T. Then the Cophenetic vector of S satisfies that*

$$\varphi_S(i,j) = \varphi_T(i,j) - \delta'_T(\textit{root of } S) + 1$$

Proof (Theorem 2). Let's proof this by induction.

- For processes with 1 or 2 activities, one can list all possible deterministic process trees and check that no two processes share the same transformed tree.
- For processes with $n > 2$ activities, we will show that every strict subtree[5] of T is equal to another subtree of T'. Let VT be a strict subtree of T. Suppose A and B are two activities such that $[A,B]_T$ is the root of VT. Activities A and B are also included in the deterministic process tree T', and $[A,B]_{T'}$ is the root of a certain subtree VT'.

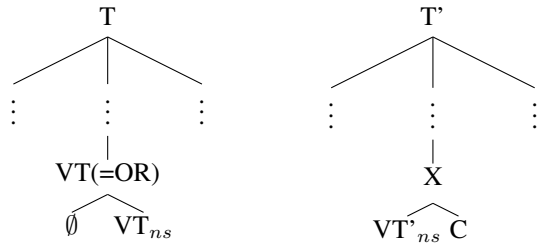


Lemma 1 ensures that

[5] Here a strict subtree of T is any subtree that does not contain the root of T

$$\begin{aligned}\varphi_{VT}(i,j) &= \varphi_T(i,j) - \delta'_T([A,B]_T) + 1 \\ &= \varphi_{T'}(i,j) - \delta'_T([A,B]_{T'}) + 1 = \varphi_{VT'}(i,j)\end{aligned}$$

where the second equality holds since $trf(T) = trf(T')$ and Theorem 1. And therefore, VT and VT' share the same Cophenetic vector and its size is smaller than T and T'. By induction, we can say that both process trees are equal.
There is one case where there are no two activities A and B such that $[A,B]_T$ is the root of VT: The root of VT is an OR-clause, and one children is a silent transition. In this particular case, we can work with the non-silent children VT_{ns}. The combination of two consecutive *OR* conditions is not possible in a valid deterministic process tree, and therefore VT_{ns} falls under the proved assumption. Hence, there is a subtree VT'_{ns} of T' that is equal to VT_{ns}.



VT_{ns} and VT'_{ns} share the same activities, and VT_{ns} is a strict subtree of T. Therefore, VT'_{ns} is also a strict subtree of T'. Let X be its parent node. We will show that X is in fact an *OR* condition, and it only has another silent branch. Let's assume there exists an activity C under X but not included in VT'_{ns}. There are two options:

- X is the root of T'. In that case, we can replace the subtrees VT_{ns} and VT'_{ns} by a mock activity C'. We reduced the problem to the 2 activities case, already solved. In that case, we share the same Cophenetic value but the two process trees are different (T' does not have a silent transition). We arrived to this contradiction by assuming that C exists.
- X is not the root of T'. In that case, the subtree VX induced by the node X is a strict subtree of T' and X is not and *OR* condition. By applying the previous reasoning, there is a subtree W of T that is equal to VX and includes VT_{ns}. Notice that, in that case, the only possibility is that C is a silent transition.

This shows that any subtree of T is equal to a certain subtree of T'. By applying this result to all the direct children of the root of T one can see that T and T' are indeed equal. □

4.2 Behavioural information captured by Cophenetic values

The syntax of process trees allow us to easily check the **direct causality** of two activities in the model: one simply needs to check the behaviour explained by their LCA. Co-occurrence of activities is described by an AND gateway, whilst OR internal nodes induce conflict between their underlying activities. Notice that this causal relation is a property for the minimum subtree containing the pair of activities. For instance, if the two activities are inside a bigger loop structure, we would not be able to retrieve this information due to the loop gateway being some levels above the LCA.

To provide a more global information than the local direct causality, depths given by Definition 5 can be used. They summarize the behavioural situation of the given node. See, for instance, the processes of Figure 5. Depth of activity D could be seen as the sum of the *blocks* found from the root to the node.

$$\begin{aligned}
\delta_{T_1}(D) &= 1(root) + 2.5(Seq) + 1(And) + 0.5(Or) \\
\delta_{T_2}(D) &= 1(root) + 3(Seq) + 1(And) \\
\delta_{T_1}(D) - \delta_{T_2}(D) &= (1-1)(root) + (2.5-3)(Seq) + (1-1)(And) + (0.5)(Or) \\
&= -0.5(Seq) + 0.5(Or) \qquad (1)
\end{aligned}$$

Notice that by considering the difference of the two depths, i.e. the value considered by the Cophenetic distance, we start highlighting where are the differences, and the type of changes committed, of the behaviour up to activity D.

When comparing pairs of activities, the cophenetic distance does not only consider the depth of the two activities but also the LCA. Following the previous example, let's compare activity D with C:

$$\begin{aligned}
\delta_{T_1}(C) - \delta_{T_2}(C) &= (1-1)(root) + (2.5-3)(Seq) \qquad (2) \\
&\quad + (1-1)(And) + (0.5-0.5)(Or) \\
&= -0.5(Seq) \qquad (3) \\
\delta_{T_1}\left([C,D]_{T_1}\right) - \delta_{T_2}\left([C,D]_{T_2}\right) &= (1-1)(root) + (2.5-3)(Seq) + (1-0)(And) \\
&= -0.5(Seq) + 1(And) \qquad (4)
\end{aligned}$$

The Cophenetic value of C stores the differences on the previous block in the sequence, as it did with Activity D. Besides, the Cophenetic value of activities C and D captures again the difference in the sequence and also an AND gateway. Hence, the pair of activities C and D are a step closer to the end in one of the two process models. But more interesting properties could be extracted by measuring the difference of such Cophenetic values: Whilst the cophenetic value $\delta_{T_1}([C,D]) - \delta_{T_2}([C,D])$ gives an idea of the difference of the two processes up to the LCA $[C,D]$, these two new values provides the same differential analysis on the paths from the ancestor to the activities. In this example, $(2) - (3) = 1$ indicates that the position of C with respect to their common ancestor differ in the insertion of an AND gateway; whilst in the case of activity D, $(1) - (2) = 0.5$ recognizes that an OR gateway has been added, or replaced by an *AND*, in one of the models.

This example shows the potential of the LCA, and the Cophenetic values, to generate more understandable and user-friendly comparison tools between process trees. Definition 6 shows two possible sentences we could build thanks to this information.

Definition 6. *A set of human-readable differences can be generated using the Cophenetic values.*

- *Given a pair of activities A and B such that they differ in the behaviour explained by their LCA. We could say that*

  "In the first model, Activities A and B are (in sequential order / co-occurrent / conflict). Whilst they are (in sequential order / co-occurrent / conflict) in the second model. Besides, the position of this behavioural decision differ in $\delta_{T_1}([A,B]) - \delta_{T_2}([A,B])$ units."

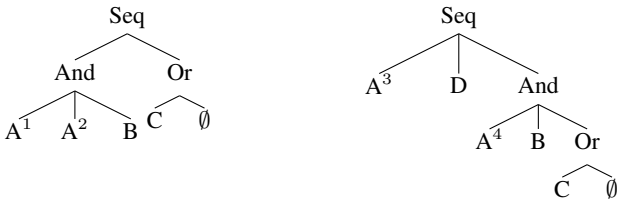


Fig. 6. Example of two indeterministic process trees. Activities A are indexed for the sake of simplicity, but all of them are indistinguishable.

– *Given a pair of activities A and B showing the same causality but $\delta_{T_1}([A,B]) - \delta_{T_2}([A,B]) \neq 0$. We could say that*

"Activities A and B show the same causality, but the position of this behavioural decision differ in $\delta_{T_1}([A,B]) - \delta_{T_2}([A,B])$ units."

In this section a formal guarantee for process tree equality based on Cophenetic distance has been presented, which restricts process trees to be deterministic. Next section lifts this restriction deriving an approximate metric based on the existence of a matching between the two process trees.

5 Distance between Indeterministic Process Trees

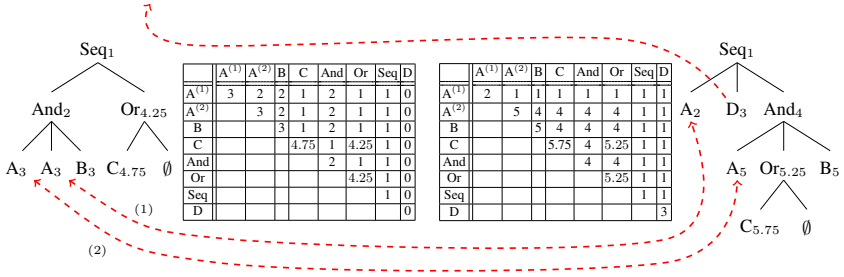
Only a small fraction of the process models generated from the human interaction with the source code repository are deterministic Process Trees. In the general case, each SVN command is executed several times during a developer day of work, and usually in different contexts producing processes with several duplicated activities. Unfortunately, the Cophenetic distance definition does not easily extend to such a kind of process. Figure 6 depicts an example of two indeterministic process trees where one activity, A, is duplicated. The Cophenetic distance cannot be used as it is was previously defined. First, the left model has two options for the depth of activity A. And more importantly, when computing the LCA of A and C, the results depend on which copy of activity A we chose. For instance, the LCA of A^3 and C is the root, but the *AND* gateway w.r.t. A^4. Nevertheless, we can still approximate an upper bound similarity metric between indeterministic process trees. In this section we present a technique that can still be applicable when (some of) the input process trees are indeterministic.

Notice that two process trees T_1 and T_2 are equal if there exists a relabeling of both process trees such that each new label replaces the same label in both models, the resulting process trees are deterministic and their cophenetic distance is zero. Such a relabeling could also be seen as a matching between the activity nodes of both process trees. We could tackle the challenge of extending the Cophenetic distance by making use of such a matching: instead of considering pairs of activities (uniquely represented in a deterministic process tree), this similarity metric compares two pairs of matched nodes. The aforementioned ambiguities among repetitions of an activity are removed by considering these matches.

Definition 7. *Let ω be a matching between the nodes of T_1 and the nodes of T_2, we define their* ***matching Cophenetic distance*** *over ω as*

$$d\omega_\varphi = \sum_{(i_1,i_2)\,\in\,\omega} \sum_{(j_1,j_2)\,\in\,\omega} |\varphi_{T_1}(i_1,j_1) - \varphi_{T_2}(i_2,j_2)|$$

Notice that the nodes i_1, i_2, j_1, j_2 in Definition 7 are not necessarily representing activities in the model. Such a distance considers all nodes as labeled. The quality of such a similarity metric depends on the quality of the matching ω. On top of that, the utility of the measurement decreases if activity labels are not preserved by the matching.



	A(1)	A(2)	B	C	And	Or	Seq	D
A(1)	3	2	2	1	2	1	1	0
A(2)		3	2	1	2	1	1	0
B			3	1	2	1	1	0
C				4.75	1	4.25	1	0
And					2	1	1	0
Or						4.25	1	0
Seq							1	0
D								0

	A(1)	A(2)	B	C	And	Or	Seq	D
A(1)	2	1	1	1	1	1	1	1
A(2)		5	4	4	4	4	1	1
B			5	4	4	4	1	1
C				5.75	4	5.25	1	1
And					4	4	1	1
Or						5.25	1	1
Seq							1	1
D								3

Fig. 7. Example of two indeterministic process trees and a matching Cophenetic distance (represented as a matrix) with respect to a certain node matching. All nodes are matched to their respective nodes with the same label, except activities A (discontinued lines (1) and (2) depict how they are paired) and activity D that does not have a representative node in the first tree. Subscripts depict the depth of the nodes.

Figure 7 depicts an example of such a matching Cophenetic distance of the models of Figure 6. From the set of all the possible matching, we choose to pair activities with the same label and, for activities A and D, we considered the pairs depicted by discontinued lines. Notice that activity D does not have a matched node in the first tree. In the middle of the Figure, one can find the Cophenetic vectors of both process trees. When considering Activity D, we treat this case as is if the activity does not exist in the first model. This example shows how the matching Cophenetic distance is computed for a specific node matching, but we could iterate over all matchings and get the minimum value possible.

Definition 8. *We define the* ***minimum matching Cophenetic distance*** *as*

$$d_{\min\,\varphi}(T_1,T_2) = \min_{\omega}\, d\omega_\varphi(T_1,T_2)$$

where the matching ω preserves activity's labels.

Although the matching Cophenetic distance is a quadratic-time algorithm [5], once we have chosen a particular matching ω, it is still computationally infeasible to compute this distance for each candidate ω in the *minimum matching Cophenetic distance*. In a practical scenario in which the size of the process trees made it impossible to test all

possible matchings, one would be able to bound this ideal distance with an approximate node matching. Although current matching algorithms [2, 11, 12] focus on preserving the structure of the graph, they could be used to approximate this matching due to the structural approach of the Cophenetic distance.

We have chosen the Flexible Tree Matching algorithm (FTM) [11] for estimating the minimum matching Cophenetic distance with indeterministic process trees. The FTM finds the minimum-cost matching that takes into account the cost of relabeling a node, removing or adding a node, and breaking structural relations between nodes (such as direct descendants and siblings). Notice the resemblance to the definition of the Graph Edit Distance (GED): The cost of the matching resulting from FTM is an approximation of the GED, but assessing also the cost of not having the same neighbors. Tuning these costs allows us to focus on mapping nodes with the same activity (we set to 1, the maximum value, the cost of relabeling) and diminishes the relevance of structural differences. The following proposition establishes also a complexity bound:

Proposition 1. *The FTM needs at least $O(M \cdot N^3 \log N^2)$ operations to approximate the matching between two process trees T_1 and T_2. Where M is the number of iterations needed by the algorithm (i.e. the expected quality of the results) and N is the total number of nodes in T_1 and T_2.*

Proof. The Flexible Tree Matching iterates M times over a randomly generated matching, to retrieve the find the best possible matching. In each iteration, the algorithm needs to recompute the N pair of matches. A weighted bipartite N^2 graph is considered, where weights represent the cost of adding such a pair to the matching. To get the best outcomes from this choice, the algorithm sort all the edges and randomly chooses one of the costless edges. Hence, each iteration of the Flexible Tree Marching needs $O(N^3 \log N^2)$ operations, plus the complexity of computing the cost of each pair of nodes (which may involve traversing the whole matching depending on the implementation). □

In summary, extending the technique of this paper to indeterministic process trees requires to first compute a matching and then compute the Cophenetic distance over this matching. This comes with an increase of the complexity due to the need to compute a matching, a step that dominates the complexity of the whole approach. In the next section we evaluate the proposed method on various types of benchmarks.

6 Evaluation

We divided the evaluation of our similarity metric in three experiments: First we consider a small set of synthetic process models to position our metric with respect to already established comparison tools. Secondly we check that the results given by our approach are consistent with two other metrics in a set of real process models. Finally, we stress the Cophenetic distance with large process models to assess its scalability.

Qualitative Comparison. Figure 8 depicts eight models extracted from [3]. These models were used in [3] to evaluate different similarity metrics. All models are deterministic, and share the same activity set except process model V_3. Table 1 depicts the similarity

given by the Cophenetic distance and state-of-the-art process models distances. In order to compute the Cophenetic distance, all models have been represented as process trees. Notice that the inclusive gateway of model V_3 cannot be translated to a deterministic process tree (because only exclusive ORs are accepted), but it was translated to an AND gateway with all internal branches being completely optional. The Cophenetic distance differentiates models V_0 and V_2, but considers V_0 more similar to V_5 than V_4. Discrepances shown in Table 1 highlights the lack of a clear definition of *similarity*. Overall, the Cophenetic distance offers a different view for the comparison with respect to the other metrics.

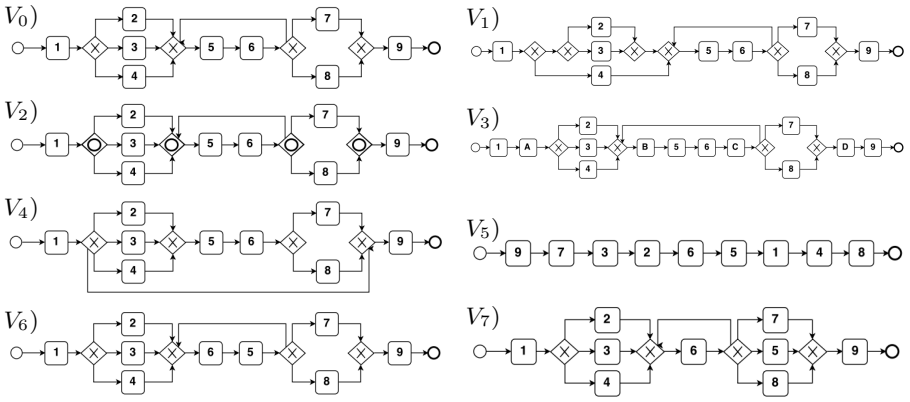


Fig. 8. 8 process models extracted from [3]. Process models $V_1, \ldots, V_7$ are variants from the same process model V_0.

$\mathbf{V_0}$ **compared to**	$\mathbf{V_1}$	V_2	V_3	V_4	V_5	V_6	V_7
Cophenetic Distance							
Percentage of Common Nodes and Edges							
Node- and Link-Based Similarity							
Graph Edit Distance							
Label Similarity and Graph Edit Distance							
Number of High-Level Change Operations							
Comparing PMs Represented as Trees							
Comparing Dependency Graphs							
Causal Behavioural Profiles							
Event Structures							
Longest Common Subsequence of Traces							
Similarity Based on Traces							

Table 1. Similarity of model V_0 to the rest of models from Figure 8 with respect to several similarity metrics. Similar models are depicted by darker cells. Values were extracted from [3], except for the Cophenetic and Event Structures [1].

Correlation with two other metrics. We gathered 700 pairs of deterministic process models from the SAP Reference Book [6] to compare our approach to two other established process model similarity metrics in a real scenario. We have chosen the traditional graph edit distance as a representative of a structural comparison tool; and, for the behavioural part, we have chosen the Event Structures technique [1]. Figure 9 depicts the comparison of the three metrics. The X and Y coordinates of a point depicts the distance given by two comparison tools, and the color represents the density of pairs in such a situation. I.e., the less dark blue a point is, the more pairs of models satisfying this relation. One can check, for instance, that most of the models differ at 10 units by the behavioural technique and the graph edit distance. Histograms show that the measurements given by the three metrics are correlated[6]. It is not clear that the same factual differences are measured by the three metrics, but the scores obtained are aligned with the two other established metrics.

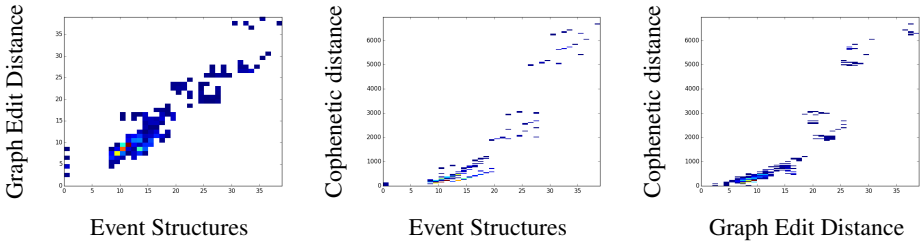


Fig. 9. A set of two-dimensional histograms comparing the results of the three comparison tools in the SAP dataset.

Scalability of the Cophenetic distance. We also study the differences in performance over large process models. We considered 7 additional pairs of process models and run the three comparison tools on each pair. The size of the processes, presence of concurrent blocks and loops varies among the models to test the applicability of the three tools. Table 2 depicts the size of such models, and the time needed to measure the differences. The Graph Edit Distance wins all the tests, the tree structure made this algorithm work way faster than usual. The complexity of the other two tools increases significantly with respect the number of activities, although the growth rate in the Cophenetic distance is considerably smaller. Notice the second pair of models, in which concurrency is present, make the behavioural tool run out of memory. Besides, we decided to stop the behavioural tool after 12 hours in all tested process models with more than 100 activities, even with deterministic process models in which the cophenetic distance showed significantly smaller times. This analysis allow us to recommend the Cophenetic distance over other behavioural approaches to analyze big process models [7].

[6] In all three cases, the Pearson correlation coefficient is above 0.85 with a p-value, for testing non-correlation, below 10^{-12}.

[7] Remember that the scale of metrics d_{φ}, d_{ES} and d_{GED} is different, a fact that explains the differences on the absolute values provided in each one.

[8] We discover these processes by analyzing the accesses of two developers to an internal source code repository. Figure 1 depicts an example of a pair of such type of processes.

Table 2. Time spend in computing the distance between a few selected process models. The table shows the number of activities in each process model, the distance given by the Cophenetic metric and the other two selected comparison tools, and the time used.

Size	Deterministic	Concurrency	Realistic	d_φ	Time	d_{ES}	Time (Event Structures)	d_{GED}	Time (GED)
25	No	No	No	0	1.71 s	0	1.63 s	0	0.03 s
30	Yes	Yes	No	7250	0.005 s	Run out of memory		40	0.009 s
50	No	No	No	3713	54.37 s	9	90.54 s	93	0.12 s
60	No	No	Yes [8]	190	322.48 s.	$>$ 12 hours		167	0.19 s
100	No	No	No	16615	467.23 s	$>$ 12 hours		184	0.42 s
100	Yes	No	No	452299	0.57 s	$>$ 12 hours		189	0.14 s
200	Yes	No	No	2441571	2.28 s	$>$ 12 hours		371	0.53 s

7 Conclusions and Future Work

In this paper we have adapted Cophenetic vectors from computational phylogenetics area to be able to automatically compare process models. Previous techniques were binary classified as structural and behavioural techniques, but we have shown that such a classification is indeed fuzzy. Albeit behavioural techniques are computationally demanding, the structural-but-behavioural intermediate approach we presented will allow BPM experts to efficiently asses behavioural comparison between models.

Next steps would focus on extending behavioural differences from the difference of Cophenetic values. There is also room to improve the utility and efficiency of the comparison of indeterministic process trees. The presented approximated matching is computed without taking into account the Cophenetic distance itself, but there might be a better matching algorithm that exploits the properties of the Cophenetic values.

Acknowledgements. This work is funded by Secretaria de Universitats i Recerca of Generalitat de Catalunya, under the Industrial Doctorate Program 2013DI062, and European Commissions 7th Framework Programme project LeanBigData (Agreement 619606), and the Spanish Ministry for Economy and Competitiveness, the European Union (FEDER funds) under grant COMMAS (ref. TIN2013-46181-C2-1-R).

References

1. Abel Armas-Cervantes, Paolo Baldan, Dumas Marlon, and Luciano García-Bañuelos. Behavioral comparison of process models based on canonically reduced event structures. *Business Process Management: 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings*, pages 267–282, 2014.
2. Vikraman Arvind, Johannes Kobler, Sebastian Kuhnert, and Yadu Vasudev. Approximate graph isomorphism. In *Proceedings of the 37th International Symposium, MFCS 2012*, 2012.
3. Michael Becker and Ralf Laue. A comparative survey of business process similarity measures. *Comput. Ind.*, 63(2):148–167, February 2012.
4. J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst. A genetic algorithm for discovering process trees. In *2012 IEEE Congress on Evolutionary Computation*, 2012.
5. Gabriel Cardona, Arnau Mir, Francesc Rosselló, Lucía Rotger, and David Sánchez. Cophenetic metrics for phylogenetic trees, after sokal and rohlf. *BMC Bioinformatics*, 14(1):1–13, 2013.

6. Thomas Curran, Gerhard Keller, and Andrew Ladd. *SAP R/3 business blueprint: understanding the business process reference model*. 1997.
7. Remco M. Dijkman. Diagnosing differences between business process models. In *BPM 2008, Milan, Italy, September 2-4*, pages 261–277, 2008.
8. Remco M. Dijkman, Marlon Dumas, and Luciano García-Bañuelos. Graph matching algorithms for business process model similarity search. In *BPM 2009, Ulm, Germany, September 8-10*, pages 48–63, 2009.
9. Remco M. Dijkman, Marlon Dumas, Luciano García-Bañuelos, and Reina Käärik. Aligning business process models. In *EDOC 2009, 1-4 September 2009, Auckland, New Zealand*, pages 45–53, 2009.
10. Remco M. Dijkman, Marlon Dumas, Boudewijn F. van Dongen, Reina Käärik, and Jan Mendling. Similarity of business process models: Metrics and evaluation. *Inf. Syst.*, 36(2):498–516, 2011.
11. Ranjitha Kumar, Jerry O. Talton, Salman Ahmad, Tim Roughgarden, and Scott R. Klemmer. Flexible tree matching. In *Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI 2011)*, 2011.
12. Adria Alcala Mena and Francesc Rosselló. Ternary graph isomorphism in polynomial time, after luks. *CoRR*, abs/1209.0871, 2012.
13. Artem Polyvyanyy, Matthias Weidlich, Raffaele Conforti, Marcello La Rosa, and Arthur H. M. ter Hofstede. The 4c spectrum of fundamental behavioral relations for concurrent systems. In *PETRI NETS 2014, Tunis, Tunisia, June 23-27*, pages 210–232, 2014.
14. F. James Rohlf Robert R. Sokal. The comparison of dendrograms by objective methods. *Taxon*, 11(2):33–40, 1962.
15. Li Sun, Serdar Boztas, Kathy Horadam, Asha Rao, and Steven Versteeg. Analysis of user behaviour in accessing a source code repository. Technical report, RMIT University and CA Technologies, 2013.
16. Matthias Weidlich, Jan Mendling, and Mathias Weske. Efficient consistency measurement based on behavioral profiles of process models. *IEEE Tr. Soft. Eng.*, 37(3):410–429, 2011.
17. Matthias Weidlich, Artem Polyvyanyy, Jan Mendling, and Mathias Weske. Causal behavioural profiles - efficient computation, applications, and evaluation. *Fundam. Inform.*, 113(3-4):399–435, 2011.
18. Zhiqiang Yan, Remco M. Dijkman, and Paul W. P. J. Grefen. Fast business process similarity search. *Distributed and Parallel Databases*, 30(2):105–144, 2012.