Large-Scale Image Phylogeny: Tracing Back Image Ancestry Relationships
Zanoni Dias, Siome Goldenstein, Senior Member, IEEE, Anderson Rocha, Member, IEEE

Abstract—Similar to organisms that evolve in biology, a document can change slightly overtime while each new version is also able to generate other versions. Multimedia Phylogeny investigates the history and evolutionary process of digital objects which includes finding the causal and ancestry document relationships, source of modifications and the order and transformations that originally created the set of near duplicates. Multimedia Phylogeny has direct applications in security, forensics, and information retrieval. In this paper, we explore the phylogeny problem for near-duplicate images in large-scale scenarios, and present solutions that have straightforward extension to other media such as videos. Experiments with about two million test cases (with synthetic and real data) show that our methods automatically build image phylogeny trees from partial information about the near-duplicates, improving the efficiency and effectiveness of the whole process, and represent a step-forward determining causal relationships of digital images overtime.

Index Terms—Image Phylogeny; Image Dependencies; Ancestry Relationships; Near-Duplicate Detection and Recognition.

I. INTRODUCTION

It is estimated that the year of 2009 generated more data by individuals than the entire history of mankind through 2008 [1], [2]. A January 2012 report by the World Economic Forum in Davos, Switzerland, declared data “a new class of economic asset, like currency or gold” [3]. However, how can we make sense of such a massive amount of data, transform it into information and, ultimately, into knowledge? Surely sophisticated algorithms are paramount for gleaning knowledge and insights from big data.

Parallel to its recognized importance, big data has also numerous problems as most of it is not typically ready for traditional databases or data mining. It is common, for instance, that several documents are duplicates or near-duplicates of one another. While it is straightforward to find exact duplicates among available media that is not true for many types of media objects when they undergo slight modifications that change them without interfering on their semantical meaning leading to the so called near-duplicate media objects.

The detection and recognition of near-duplicate media objects have received attention from researchers in the last years, specially for digital images and videos [4]–[7]. The literature has named this family of problems as near-duplicate detection and recognition (NDDR) of a document. While detection aims at verifying whether or not two documents are near copies of each other, recognition seeks for all member documents that are near copies of a given query in a large collection of documents. While NDDR techniques are targeted usually at improving search efficiency and improving storage, there are other applications such as tracking the legal distribution and spread of a document on the Internet.

Neglected until recently, a more challenging task arises when we have the task of identifying which document is the original within a set of near-duplicates or the structure of generation of each near-duplicate pointing out the ancestry relationships for all documents in a set. Only recently there were the first attempts to go beyond NDDR and to identify the structure of relationships within a set of near-duplicates, a new problem called Multimedia Phylogeny [8]. Multimedia Phylogeny explores the history and evolutionary process of digital objects, looking for the causal and ancestry document relationships, the source of modifications, and the order and transformations that originally created the set of near duplicates. Some applications are:

- **Security**: the transformation history of a set of documents can describe the flow of content distribution;
- **Forensics**: we can identify the original document out of a near-duplicate set to perform document forensic analysis;
- **Copyright enforcement**: we can devise traitor tracing methods without watermarking;
- **News tracking services**: near-duplicate relationships can explain key elements about the opinion forming process across time and space;
- **Content-based Retrieval**: we can design advanced information retrieval systems.

There are mainly two perspectives to consider when finding the structure of relationships within a set of near-duplicate images or videos [8]–[12]: first, we need to find an informative dissimilarity function to compare the near-duplicates; second, we need to devise a proper algorithm capable of creating a tree of relationships from the dissimilarity measurements. Normally, such a dissimilarity matrix correlating two near_duplicates is expensive to compute and, these early investigations were constrained to the case of image and video near_duplicates in small scenarios, with no more than 50 or 60 documents for analysis each time.

Here, we go beyond prior work and aim at finding the structure of relationships in a large-scale context. Our focus in this paper is on digital images, but the ideas also apply to videos and text. We devise methods for dealing with
partial constructed dissimilarity matrices, which can actively request new entries on-the-fly. This means that we can start the procedure calculating only a subset of the entries of the dissimilarity matrix and the methods we propose will take care of finding additional entries while optimizing the search in order to use as few entries as possible. We perform experiments with more than one million test cases and show that our solutions represent a step forward to find ancestry relationships in an efficient and effective way.

II. RELATED WORK

A near-duplicate is normally seen as a transformed version of a document that preserves its semantics. Joly et al. [7] formally proposes a definition of what a duplicate is, based on the notion of tolerated transformations. According to the authors, a document \( D_1 \) is a near-duplicate of a document \( D \), if \( D_1 = T(D) \), \( T \in \mathcal{T} \), where \( \mathcal{T} \) is a set of tolerated transformations. \( D \) is called the original document or the root of the document evolution tree. A family of transformations \( \mathcal{T} \) can contain several combinations of transformations such as \( D_3 = T_3 \circ T_2 \circ T_1(D) \), \( T_{\beta=1,2,3} \in \mathcal{T} \).

An image duplicate links the original document to any of its variations through a transformation (e.g., warping and cropping, compression, color and intensity correction and adjustments). The formal definitions of Joly et al. [7] are not totally followed in literature for image and video near-duplicates. Typical NDDR methods normally tag as “duplicates” any pair of images depicting similar content. This might be a problem in a forensic scenario, for instance, when we need to probe whether a suspect indeed manipulated a protected photography.

In spite of that, in the past decade, we have seen some exceptional progress on the development of efficient and effective systems to identify document near-duplicates in the wild for text, videos and images [4]–[7], [13]. However, only recently there were the first attempts to go beyond the detection of near duplicates, with attempts to identify the structure of relationships within a set of near-duplicates.

Kennedy et al. [9] first addressed the problem of parent-child relationships between pairs of images and termed it “document archaeology”. De Rosa et al. [11] proposed to detect the image dependencies by considering images’ mutual information calculated using content similarities as well as image noise acquisition telltales.

Dias et al. [10] defined the problem of Image Phylogeny Tree reconstruction, and later extended it [8] for dealing with more than one tree (user interaction needed) and for videos [12].

Kender et al. [14] studied content-based relationships among YouTube video clips related to the same event, and illustrated the construction of a content-dependency graph representing how videos evolve regarding mutations, crossover and other operators.

None of these approaches, however, are suitable for large-scale analyses. Here, our objective is to determine the structure of relationships within a set of near-duplicate images in a large-scale scenario using partial matrices.

III. IMAGE PHYLOGENY FORMALIZATION

Finding the structure of relationships in a set of near-duplicate images normally requires two steps: a dissimilarity function \( d \) responsible to calculate how different is each pair of images and how likely they are father and son on the tree and a tree reconstruction algorithm that operates on this matrix. An image phylogeny final product is an image phylogeny tree (IPT) which connects images according to their ancestry/descendant relations [8].

Formally, let \( T_\beta \) be an image transformation from a family \( \mathcal{T} \). We can devise a dissimilarity function between two images \( I_A \) and \( I_B \) as the minimum

\[
d_{x_A,x_B} = \left| I_B - T_\beta(I_A) \right|_{\text{point wise comparison method } \ell}
\]

for all possible values of \( \beta \) that parameterizes \( \mathcal{T} \). Equation 1 measures the amount of residual between the best transformation of \( I_A \) to \( I_B \), according to the family of operations \( \mathcal{T} \), and \( I_B \) itself. We can perform such comparison using any point wise method \( \ell \) such as pixel wise minimum square error.

With a set of \( n \) near-duplicate images, the first task for creating an image phylogeny tree is to calculate the dissimilarity between every pair of such images. For that we need a reasonable set of possible image transformations, \( \mathcal{T} \), from which one image can generate an offspring [8], [11].

IV. BUILDING AN IMAGE PHYLOGENY TREE

Dias et al. [8], proposed an approach to calculate image dissimilarities and also for finding IPTs. For the dissimilarity calculations, the authors:

1) Find interest points in each pair of images (i.e., using SURF [15]) to estimate warping and cropping parameters robustly using RANSAC [16].

2) Calculate pixel color normalization parameters by mapping the color channels of one image onto the color channels of the other image.

3) Compress one of the images with the same compression parameters of the other.

4) Uncompress both images and compare them pixel wise according to the minimum square error metric.

Figure 1 depicts the process. The last step consists of finding the actual phylogeny tree associated with the calculated dissimilarity matrix. The Oriented Kruskal algorithm [8], for example, finds the root of the tree and builds the oriented tree representing image parenthood relationships with complexity \( O(n^2 \log n) \), for \( n \) available image near-duplicates.

Given a dissimilarity matrix \( M \) of \( n \) near-duplicate images, the Oriented Kruskal algorithm starts with each image near-duplicate as the root of a tree. Next, the algorithm sorts all positions \((i,j)\) of \( M \) and then analyzes each position \((i,j)\) according to the sorted order, joining different trees and checking whether or not the endpoints of the analyzed position is a root.

Figure 2 depicts a step-by-step toy example execution of such algorithm for \( n = 6 \) near-duplicate images. The algorithm starts with a forest with \( n = 6 \) roots and sorts all the positions \((i,j)\) in \( M \) according to their dissimilarity value.
Figure 1. To calculate image dissimilarities between a pair of images $I_1$ and $I_2$, we find robust points of interest in both images and for those which are good matches (yellow stars) we calculate an homography matrix representing the necessary parameters to transform one image to another’s domain. Once we perform the mapping, we can compare both images pixel wise within the region of interest they overlap.

The most expensive task in image phylogeny is the dissimilarity matrix calculation. For $n = 1,000$ images, we need to calculate $n^2 - n$ entries (the matrix is not symmetric and the diagonal is not calculated). Each entry in this matrix requires the mapping of one image onto another image’s domain which requires estimating the necessary transformation parameters.

According to the algorithm in [8], we need about one second to calculate each entry of a matrix using a conventional 2.1GHz two-core machine with 4GB of memory. For a $n = 1,000$ matrix, this scales up to 11.5 processing days. For comparison, the reconstruction process once the dissimilarity matrix is calculated takes less than one minute. Although parallelization techniques could be used to speed up each dissimilarity calculus, even with such improvements, methods for finding the phylogeny tree from only a fraction of the matrix is paramount since they will drastically reduce the number of computations required.

V. PROPOSED METHODOLOGY

The most expensive task to reconstruct an image phylogeny tree is the dissimilarity matrix calculation. Often we need to deal with hundreds or thousands of near-duplicate images...
or videos each time and 11.5 computing days to calculate the dissimilarity matrix before using an IPT reconstruction algorithm for \( n = 1,000 \) images is prohibitive.

Here, we expand upon the best known image phylogeny solution, Oriented Kruskal [8], for dealing with large-scale image phylogeny problems. We introduce three methods able to reconstruct an image phylogeny tree with a partially complete dissimilarity matrix.

We start the procedure calculating only a subset (randomly chosen) of elements of the matrix and, if necessary, these methods automatically demand the calculation of missing entries. The whole problem in this paper is that it is prohibitive to calculate the entire dissimilarity matrix in order to seek for the phylogeny tree. Instead, we propose solutions to deal with partial matrices (sparse), which demand new calculations as necessary. Only a small percentage of new entries are calculated, lowering the final number of dissimilarity calculations.

### A. Grandpa Heuristic

We call the first method we propose to deal with a partially complete dissimilarity matrix \( M \), **Grandpa Heuristic:**

1. **Calculate** the initial IPT using the partially complete dissimilarity matrix and the algorithm discussed in Section IV.

2. **For each node** \( x \) in the reconstructed IPT, check if its dissimilarity to its grand-parent \( y \) is known. If not, calculate the dissimilarity \( d(x, y) \) and \( d(y, x) \).

3. **Reconstruct the IPT** from the updated matrix including the new calculated entries.

4. **Repeat Steps 2 and 3** until convergence or until the amount of new calculation solicitations exceeds \( t\% \) of the total entries in the matrix (here \( t = 10\% \)).

The word “convergence” means that any of the proposed heuristics will stop if the calculated tree at a given iteration \( t \) is the same as the one calculated in the step \( t - 1 \) (the tree does not change with more entries in the matrix).

This algorithm checks \( O(n) \) entries at each iteration in the worst case (e.g., a degenerate tree in the form of a linked list). The rationale is that knowing the dissimilarity between each triplet (grand-son, father, grand-father) for the current IPT gives the algorithm a broader overview of the reconstruction process allowing it to better decide if each triplet is consistent or not and also to decide new possible triplets.

Figure 3(a) depicts one algorithm’s round for \( n = 16 \) near-duplicates. The tree depicted is the result of a reconstruction from a sparse dissimilarity matrix. After the first IPT construction, the algorithm checks, for each node, if it knows the dissimilarity of that node to its grand-parent. We need to perform 13 grand-parent tests in this iteration. The grand-parent of Node 13 is Node 5 while Node 2 does not have a grand-parent in this tree.

### B. Direct Ancestry Heuristic

We call the second method we propose to deal with a partially complete dissimilarity matrix \( M \), **Direct Heuristic:**

1. **Calculate** the initial IPT using the partial dissimilarity matrix and the algorithm discussed in Section IV.

2. **For each node** \( x \) in the reconstructed IPT, check if its dissimilarity to each of its direct ancestors \( y_i \) is known. If not, calculate the dissimilarity \( d(x, y_i) \) and \( d(y_i, x) \).

3. **Reconstruct the IPT** from the updated matrix including the new calculated entries.

4. **Repeat Steps 2 and 3** until convergence or until the amount of new calculation solicitations exceeds \( t\% \) of the total entries in the matrix (here \( t = 10\% \)).

This algorithm checks \( O(n^2) \) entries at each iteration in the worst case (e.g., a degenerate tree in the form of a linked list). It is a generalization of the Grandpa Heuristic. The rationale is that knowing the entire ancestry line of a node \( x \) for the current IPT gives the algorithm the ability to check the consistency of
all possible triplets (grand-son, father, grand-father), quadruplets (grand-grand-son, father, grand-father, grand-father), and X-plets (more than four elements) until its very first ancestor, the root of the tree. This knowledge gives a broader overview about the tree and allows the algorithm to better decide about node dependencies.

Figure 3(b) depicts one algorithm’s round for \( n = 16 \) near-duplicates. After the first IPT construction, the algorithm checks, for each node \( x \), if it knows the dissimilarity of that node to each of its direct ancestors. We need to perform 26 direct ancestry tests in this iteration. For instance, Node 13 has three direct ancestors besides its parent (Nodes 5, 2, and 1) while Node 6 has one and Node 12 has two.

C. (In)Direct Ancestry Heuristic

We call the third method we propose to deal with a partially complete dissimilarity matrix \( M \), (In)Direct Heuristic:

1. **Calculate** the initial IPT using the partially complete dissimilarity matrix and the algorithm discussed in Section IV.

2a. For each node \( x \) in the reconstructed IPT, check if its dissimilarity to each of its direct ancestors \( y_i \) is known. If not, calculate the dissimilarity \( d(x, y_i) \) and \( d(y_i, x) \).

2b. If no dissimilarity was calculated in Step 2a, then
   - For each node \( x \) in the reconstructed IPT, check if its dissimilarity to each of the direct descendants \( y_i \) of its direct ancestors \( z_i \) (i.e., \( x \)'s brothers, uncles, grand-uncles, etc.) is known. If not, calculate the dissimilarity \( d(x, y_i) \) and \( d(y_i, x) \).

3. **Reconstruct** the IPT from the updated matrix including the new calculated entries.

4. **Repeat** Steps 2 and 3 until convergence or until the amount of new calculation solicitations exceeds \( t\% \) of the total entries in the matrix (here \( t = 10\% \)).

This algorithm checks \( O(n^2) \) entries at each iteration in the worst case (e.g., for a degenerate tree in the form of a linked list or for a star-shaped tree in which one node is the parent of all other nodes). We can see it as a generalization of the Direct Heuristic Method which is good to find roots and improves upon it to better find ancestors and node connections (edges). The rationale is that knowing all direct relatives of a node \( x \) for the current IPT gives the algorithm freedom to change any necessary weak relationship in the tree.

Figure 3(c) depicts one algorithm’s round for \( n = 16 \) near-duplicates. After the first IPT construction, the algorithm checks, for each node \( x \), if it knows the dissimilarity of that node to all descendants of a direct ancestor of \( x \). We need to perform 87 (in)direct ancestry tests in this iteration. For instance, Node 13 has Node 14 as a direct descendant of Node 8. Nodes 8, 9, and 10 are direct descendants of the 13’s direct ancestor, Node 5. Also, Nodes 4 and 5 are direct descendants of the 13’s direct ancestor, Node 2. Finally, Nodes 2 and 3 are direct descendants of the 13’s direct ancestor, Node 1. Therefore, Node 13 has nine (in)direct ancestors.

VI. EXPERIMENTS AND VALIDATION

Now we turn to the methodology we use to validate the techniques we discuss in this paper. We follow the protocol proposed in [8] which uses a controlled dataset of near-duplicates. We also provide experiments with web images showing the methods’ effectiveness.

A. Evaluation Metrics

We use four quantitative measures of success [8] to analyze results: Root, Edges, Leaves, and Ancestry for scenarios for which we have Ground Truth. We calculate the four metrics according to

\[
\text{metric}(\text{IPT}_1, \text{IPT}_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|},
\]

where \( \text{metric} \) is the evaluation metric of interest (e.g., ancestry). \( \text{IPT}_1 \) are the calculated tree and the one used as reference (e.g., the tree ground-truth), \( S_1 \) is the set of elements in the first tree corresponding to the metric (e.g., set of ancestors for all nodes of the first tree) and \( S_2 \) is the equivalent for the reference tree. These metrics evaluate complementary properties of the results providing a big picture of the reconstruction algorithm’s behavior.

B. Setup for the Controlled Experiments

We consider image phylogeny trees with \( n = 1,000 \) image near duplicates. When generating a near-duplicate, an image can undergo several possible transformations. Here, we select typical transformations an image can undergo such as the ones Table I shows. We use the Uncompressed Color Image Database (UCID) [17] dataset which comprises 512 × 384-pixel resolution images without compression artifacts.

<table>
<thead>
<tr>
<th>Table I</th>
<th>Transformations and their operational ranges for creating the dataset following the protocol proposed in [8].</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transformation</strong></td>
<td><strong>Geometry</strong></td>
</tr>
<tr>
<td>(1) Resampling (Up/Down)</td>
<td>- Global scaling</td>
</tr>
<tr>
<td>(2) Rigid Transformation</td>
<td>- Global scaling</td>
</tr>
<tr>
<td></td>
<td>- Rotation</td>
</tr>
<tr>
<td>(3) Generic Affine Transformation</td>
<td>- Scaling by axis</td>
</tr>
<tr>
<td></td>
<td>- Rotation</td>
</tr>
<tr>
<td></td>
<td>- Off-diagonal correction</td>
</tr>
<tr>
<td>(4) Cropping</td>
<td></td>
</tr>
<tr>
<td>(5) Brightness Adjustment</td>
<td></td>
</tr>
<tr>
<td>(6) Contrast Adjustment</td>
<td></td>
</tr>
<tr>
<td>(7) Gamma Correction</td>
<td></td>
</tr>
<tr>
<td>(8) Re-compression</td>
<td></td>
</tr>
</tbody>
</table>

We validate six different scenarios, each one with 190,000 test cases totaling 1,140,000 test cases. Each test case represents a known percentage of entries of the dissimilarity matrix for reconstructing a phylogeny tree, a tree topology, a set of transformation parameters, and a different original
image. In all cases, we consider trees with 1,000 nodes (an original image with its 999 near-duplicates) from different tree topologies (forms of the tree), and combination parameters from Table I.

C. Baseline with Different Sparse Dissimilarity Matrices

We use the algorithm proposed in [8] as baseline. We designed three experiments (cases) to evaluate this algorithm for reconstructing IPTs with different percentages of known dissimilarity matrix entries. Case #1 evaluates the algorithm when knowing \( p \%) of the entries but with no symmetry warranty (knowing \( d(x, y) \) but not necessarily \( d(y, x) \)) while the Case #2 imposes this constraint. Finally, Case #3 evaluates the algorithm when knowing \( p \%) of the entries but for each entry we have its corresponding row and column known.

Figure 4 presents the best result out of this three experiments: Case #2, knowing \( d(x, y) \) and \( d(y, x) \) for \( p \%) of the dissimilarity matrix. Each point in the plot represents 100 tests with different tree topologies/parameters and images totaling 190,000 test cases. We do not show the results for the other two experiments (380,000 test cases) because their results are worse than the one Figure 4 depicts.

For a complete matrix, the baseline algorithm finds the root in nearly all cases and finds 54\% of nodes’ correct ancestors. On the other hand, we see that the algorithm performs poorly when it knows only a percentage of the full matrix. For instance, for \( p = 30\% \), it correctly finds the root in only 34\% of the cases and roughly 12\% of correct nodes’ ancestors.

D. Grandpa, Direct, and (In)Direct Ancestry Heuristics

The results discussed in Section VI-C showed that the state-of-the-art Oriented Kruskal algorithm [8] is not appropriate to deal with partially complete dissimilarity matrices and show huge performance loss when comparing to a full matrix.

This section presents the results for the three methods we designed to demand additional calculations to a partially complete matrix in order to improve the results. Figure 5(a-b) present the results for Grandpa and Direct heuristics while Figure 5(c-d) presents results for the (In)Direct heuristic. Each point in the plot is associated with the grid mark just before it, i.e., point \( p = 30\% \) for Direct heuristic means the algorithm started with 30\% of known entries, solicited a few more to be calculated, and converged with 35\%. Section VI-E discusses the the overhead for new calculations.

With \( p \approx 10\% \) known entries, the Direct heuristic (Figure 5(b)) successfully finds the root of the tree in nearly 100\% of the cases which represents a huge implication for forensics or copyright infringement lawsuits, for instance, when we need to trace back to the original source of a digital document and study how it was spread out. The improvement when finding the nodes ancestors is also remarkable. For \( p \approx 30\% \) of the entries known the heuristics successfully finds 30\% of the ancestors against 10\% of the baseline as discussed in Section VI-C. The overhead for Grandpa Heuristic is lower than for Direct Ancestry but in both cases the heuristics, on average, did not request more than 5-7\% new entry calculations.

Figure 5(c) shows the results for the (In)Direct Ancestry Heuristic method with respect to the ground-truth while Figure 5(d) compares it with the baseline [8]. We can see it is better than the both Direct and Grandpa heuristics. Also, comparing with the baseline, we see that while the baseline with \( p = 5\% \) of known matrix entries successfully finds the root of the trees in 13\% of the cases while the (In)Direct method starts with the same 5\%, automatically solicits a few more calculations and, with \( p \approx 15\% \) it successfully finds the root in about 100\% of the cases saving 85\% calculations and thus improving the efficiency in more than six times. The difference for the other metrics is also remarkable with the proposed method soliciting, at most, 10\% new entries while improving the effectiveness by several percent points.

E. Number of Iterations

Figure 5(e) depicts the number of iterations for the three methods. Each iteration refers to the number of times the method needs to check its ancestors and request for new entry calculations. The three heuristics stop either when they converge or when the amount of new entry calculations extrapolates 10\% of the matrix.

Figure 5(e) shows that more known entries in the matrix lead to fewer iterations. Grandpa is the heuristic which requires fewer iterations, and thus fewer new dissimilarity calculations. (In)Direct Heuristic requires more iterations early on, as it has to calculate the (in)direct ancestors of the initial set of known entries, but it stabilizes overtime.

Regardless the number of iterations, note that none of the three methods actually calculates more than 10\% new entries.

F. Real Near-Duplicate Sets from the Web

We also perform experiments with near-duplicate images collected from the web in two different scenarios: considering the same near-duplicate images used in [8] and using a famous online photography broadcast by the White House in 2011.

1) Scenario #1: For this experiment, we evaluated 760,000 test cases using the 10 images highlighted in [8]. For the baseline and the three proposed heuristics, we performed 190,000 random experiments (1,000 rounds for each of 10
internet near-duplicate sets, and 19 bins on the percentage scale).

We do not know the image parenthood relationships within each set of near-duplicate we download from the web. Therefore, we created five modified images (near-duplicates) for each original image, leading to sets with 96 to 120 images. Although we do not know the correct ancestry and relationships of the original images, we know the parents of the modified generated images (and that they are all leaves). Figure 6(a) depicts examples of the analyzed near-duplicate sets.

For the experiments with the proposed heuristics, we limited the number of new entries in 10% of the matrix. Also, as leaves are more than 85% of the tree, this metric shall be analyzed with caution, as it creates a bias for that particular topological structure. Nonetheless, it is an important metric to assess. For instance, leaves might be of particular interest since normally they are the most modified images in a phylogeny tree – they play an important role in some applications, such as sentiment analysis and news tracking.

As Figure 7 depicts, the heuristics improve the raw reconstruction. For example, with $\frac{1}{4}$ of the matrix, (In)Direct Ancestry heuristic finds the root twice as often as Oriented Kruskal, with similar results for edges and ancestry metrics.
Comparing with the synthetic validation, (In)Direct Ancestry Heuristic now has only a slight advantage over Direct Ancestry Heuristic.

2) Scenario #2: Here we want to exemplify the actual evolutionary tree of a famous 2011 photograph named *The Situation Room*. White House photographer Pete Souza took this photograph on May 1st, 2011. It depicts the US President, along with his national security team, receiving live updates of the Operation Neptune Spear, which led to the Osama bin Laden’s killing. After its online publishing, this image was reproduced by different communication channels, social networks, activists etc.

In this experiment, we have collected the original image released by the White House along with 98 near-duplicate images through Google Images. A manual analysis of the set shows, at least, nine different patterns of modifications of the images cohabiting in the web. Along with the original image which we termed ID 0000, we have regular near-duplicate images (ID 0*), cases of inserting Italian soccer player Mario Balotelli (ID a*), in the center of the image, text overlay (ID b*), watermarking (ID c*), face swap (ID d*), insertion of elements such as joystick (ID e*), other people (ID f*), hats (ID g*), etc.

Figure 6(b) depicts the resulting tree using the best proposed algorithm, (In)Direct heuristic, using only 25% of the dissimilarity matrix (four times faster than the algorithm proposed in [8]). As expected, the algorithm correctly finds the root of the tree and puts simple near-duplicate images close to it (ID 0*, red ellipses). It automatically groups most of the cases we discussed. For instance, there are explicitly two subtrees containing the Balotelli case (blue ellipses). Also, as watermarking normally is inserted by different people, it is reasonable to expect the images not forming a single dense sub-tree and that is exactly what happens here (ID c*, purple ellipses).

Although real-world cases do not have ground truth, the
VII. Conclusions and Remarks

Investigating the past history of objects with just a few clues is a challenging research topic in many science fields. For instance, in paleontology, we can analyze fossil fragments and their changes overtime to determine their age and origins. In life sciences, phenotypic and genomic studies enable us to trace back the evolution of living species. Similarly, in Multimedia Phylogeny we aim at collecting image telltales to study the evolutionary processes to which multimedia objects are subject overtime.

Most of the changes related to near-duplicate multimedia objects are natural and not necessarily harmful. However, the distribution might cause copyright infringement or represent a criminal action [8], [18], and the spreading pattern itself can help companies understand the demographics and the effectiveness of an ad campaign or a product.

Determining the structure of modifications of a set of near-duplicates is now an important problem. To determine the modification trees is computationally expensive and the current solutions are not designed to deal with large-scale analyses. Here, we have extended upon our prior work to handle large-scale scenarios and presented three methods to select extra entries of an incomplete dissimilarity matrix on-the-fly that will lead to improvements in the reconstruction results. Our best method successfully recovers the root of a tree with 1,000 image near-duplicates with only 15% necessary direct dissimilarity computations out of the $\approx (1,000^2 - 1,000)$ necessary for the full matrix (a remarkable improvement over other solutions). Since the proposed methods mainly deal with transformations on the dissimilarity matrix, its extension to deal with videos is straightforward. Results with real internet cases also show the effectiveness of the proposed methods.

The main advantages of the proposed heuristics are that they can, on-the-fly, request the computation of new (dis)similarities saving crucial time since this is the most expensive task for phylogeny analysis. The heuristics do so while preserving the accuracy of tree root finding and ancestry relationships estimation. The three proposed heuristics only require few additional calculations (e.g., at most 10%) of extra fast-to-compute calculations.

We are now exploring faster ways to compute the dissimilarities, and studying extensions to deal with other types of media such as videos and text.
ACKNOWLEDGMENT

This work was partially supported by São Paulo Research Foundation – FAPESP (grant 2010/05647-4), National Counsel of Technological and Scientific Development – CNPq (grants 307018/2010-5, 304352/2012-8, 306730/2012-0, and 477692/2012-5), Microsoft, and the European Union through the Rewind project. The Rewind project acknowledges the financial support of the Future and Emerging Technologies (FET) program within the Seventh Framework Program for Research of the European Commission (under FETOpen grant 268478).

REFERENCES