Improved Algorithm for the Network Alignment Problem with Application to Binary Diffing

Elie Mengin, Fabrice Rossi

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 192 (2021) 961–970

Procedia Computer Science

www.elsevier.com/locate/procedia

25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Improved Algorithm for the Network Alignment Problem with Application to Binary Diffing

Elie Mengin[a,1,*], Fabrice Rossi[c]

[a] *SAMM, EA 4543, Université Paris 1 Panthéon-Sorbonne, 75013 Paris*
[b] *Quarkslab SA, 13 rue Saint-Ambroise, 75011 Paris*
[c] *CEREMADE, CNRS, UMR 7534, Université Paris-Dauphine, PSL University, 75016 Paris*

Abstract

In this paper, we present a novel algorithm to address the *Network Alignment problem*. It is inspired from a previous message passing framework of Bayati et al. [2] and includes several modifications designed to significantly speed up the message updates as well as to enforce their convergence. Experiments show that our proposed model outperforms other state-of-the-art solvers. Finally, we propose an application of our method in order to address the *Binary Diffing problem*. We show that our solution provides better assignment than the reference differs in almost all submitted instances and outline the importance of leveraging the graphical structure of binary programs.

Keywords: Network Alignment ; Graph Matching ; Belief Propagation ; Binary Diffing ; Binary Code Analysis

1. Introduction

The problem of finding a relevant one-to-one correspondence between the nodes of two graphs may be known as the *Network Alignment problem* (NAP). It has a wide variety of applications such as image recognition [27, 14, 30], ontology alignment [2], social network analysis [28, 16] or protein interaction analysis [15, 10]. Multiple formal definitions have been proposed. For instance, one may only consider the node content and aim at finding the mapping with maximum overall similarity score. Such formulation reduces to the *Maximum Weight Matching problem* (MWM) [4]. On the contrary, one may only focus on the graph topologies and search for the alignment with maximum induced overlapping edges. In this case, the problem is known as the *Maximum Common Edge Subgraph problem* (MCS) [1]. In this paper, we propose a mixed formulation: given any two directed attributed graphs A and B, and two measures of similarity on both nodes and edges, find the one-to-one mapping that maximizes the sum of similarity of both matched

* Corresponding author.
E-mail address: elie.mengin@gmail.com

nodes and induced edges. In other words, we are seeking the assignment that maximizes a linear combination of MWM and MCS.

Several methods have been proposed to address this problem. Among them, NetAlign [2] introduces a complete message passing framework based on *max-product belief propagation*. In the present paper, we propose several modifications to this algorithm in order to significantly speed up the computation and to control the messages convergence. We show that these modifications provide better assignments than the original model as well as other state-of-the-art solvers in most problem instances. Finally, we show that our method could be used to efficiently retrieve the differences between two binary executables.

The rest of this paper is organized as follows. Section 2 introduces in more details the network alignment problem and reviews some existing solutions. The original model of NetAlign as well as our proposed optimizations are described in Section 3. Finally, Section 4 is dedicated to the experimental evaluation of our method.

2. Problem formulation

Let us consider any two directed attributed graphs $A = (V_A, E_A)$ and $B = (V_B, E_B)$, where $V_A = \{1, \dots, n\}$ are the vertices of A (resp. $V_B = \{1', \dots, m'\}$ for B) and $E_A = \{(i, j) | i, j \in V_A^2, i \neq j\}$ are the edges of A (resp. $E_B = \{(i', j') | i', j' \in V_B^2, i' \neq j'\}$ for B). Without loss of generality, we assume that none of the graph includes self-loops (they can be considered as node attributes if necessary).

We assume given two arbitrary non-negative measures of similarity on both nodes and edges, $\sigma_V : i, i' \in V_A \times V_B \rightarrow \mathbb{R}_+$ and $\sigma_E : (i, j), (i', j') \in E_A \times E_B \rightarrow \mathbb{R}_+$. Using these measures, we may encode all pairwise node similarity scores into a flatten vector $\mathbf{p} \in \mathbb{R}_+^{|V_A| \times |V_B|}$ such that $p_{ii'} = \sigma_V(i, i')$, as well as the similarity of all potential induced edges into matrix $Q \in \mathbb{R}_+^{|V_A|^2 \times |V_B|^2}$ such that $Q_{ii'jj'} = \sigma_E((i, j), (i', j'))$ if and only if $(i, j) \in E_A$ and $(i', j') \in E_B$, and 0 otherwise. Finally, we describe any one-to-one node mapping through a binary vector $\mathbf{x} \in \{0, 1\}^{|V_A| \times |V_B|}$ for which $x_{ii'} = 1$ if and only if node i in A is matched with node i' in B.

Given these definitions, the network alignment problem consists in solving the following constrained quadratic program:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \ \alpha \mathbf{x}^T \mathbf{p} + (1 - \alpha) \mathbf{x}^T Q \mathbf{x} \quad \text{subject to} \quad \forall i \in V_A, \sum_{j' \in V_B} x_{ij'} \leq 1 \text{ and } \forall i' \in V_B, \sum_{j \in V_A} x_{ji'} \leq 1, \tag{NAP}$$

where $\alpha \in [0, 1]$ is an arbitrary constant that determines the trade-off between node and edge similarity.

This problem is a generalization of the *Quadratic Assignment problem* and, as such, is known to be NP-complete and even APX-hard [23]. Though exact algorithms exist, they rapidly become intractable as the number of vertices rises [7]. In practice, the computation of the NAP for graphs of more than a hundred nodes must be approximated. In the rest of this section, we review some of the existing approaches proposed to address the problem.

Related work

Amongst the first approaches to approximate the NAP are the spectral methods that can be distinguished into two main categories. On one hand, spectral matching approaches are based on the idea that similar graphs share a similar spectrum [25]. Thus, they aim at best aligning the (leading) eigenvectors of the two affinity matrices (or Laplacians) [14, 21]. On the other hand, PageRank methods approximates the NAP through an *eigenvalue problem* over the matrix Q [24]. The idea consists in computing the principal eigenvectors of Q, and to use it as a similarity score of every possible correspondences. The resulting assignment can then be computed using conventional MWM solvers. Over the years, several improvements have been proposed to enhance the procedure [16, 20, 12, 28].

Other common approaches propose to directly address the quadratic program by means of relaxations. The most common convex relaxation consists in extending the solution set to the set of doubly stochastic matrices. The relaxed problem can then be exactly solved using convex-optimization solvers, and is finally projected into the set of permutation matrices to provide an integral assignment. However, when the solution of the convex program is far from the optimal

permutation matrix, the final projection may result in an incorrect mapping [18]. Other approaches make use of a concave [27] or indefinite [26] relaxations. The induced programs are generally much harder to solve but yield better results when properly initialized. Note that most methods use a combination of both relaxations [30, 29].

Several other methods are based on a linearization of the NAP objective function. The idea is to reformulate the quadratic program into an equivalent linear program, and to solve it using conventional (mixed-integer) linear programming solvers. However, this reformulation usually requires the introduction of many new variables and constraints, and computing the exact solutions of the linear program may become prohibitively expensive. In most cases, relaxations must also be introduced. A successful method, based on Adams and Johnson linearization and using a Lagrangian dual relaxation have been proposed by Klau [15] and later improved by El-Kebir et al. [10].

Finally, a message passing framework has shown promising results [2]. It is directly derived from a previous model that provided important results on solving the MWM using *max-product belief propagation* [4]. In our work, we chose to apply this model to our use case.

Note that an important limitation to the NAP is the size of the matrix Q that grows quartically with the size of the graphs. This memory requirement may become prohibitive for relatively large graphs encountered in many real-world problems. It is mainly due to the intrinsic nature of the problem which requires that every potential correspondence is evaluated with regards to other candidates, in order to take into account its topological consistency. In practice, most methods are designed to efficiently exploit the potential sparseness of the matrix Q. Therefore, they generally apply to sparse graphs only. Moreover, several approaches propose to also restrain the number of potential candidates [10, 2] and thus the problem complexity. This pre-selection may rely on prior knowledge or on arbitrary decision rules. It mostly aims at preventing the algorithm to compute the assignment score of highly improbable correspondences. The framework we introduce in the next section make use of this interesting feature.

3. Network alignment via Max-Product Belief Propagation

In this section, we first recall the original model of Bayati et al. [2] and then introduce our proposed improvements. More details about the complete framework and practical implementation of NetAlign can be found in [3].

3.1. Original model

The most common reformulation of a constrained optimization program into probabilistic graphical models usually requires the introduction of a factor-graph which assigns maximum probability to the solution of the program. For more details about factor-graph graphical models, we refer the reader to Kschischang et al. [17]. In order to design such graphical model, we must encode both the objective function and the constraints of NAP into an equivalent probability distribution. This encoding is done through the factorization of several functions (function nodes) over the different variables of the program (variable nodes). In this form, the mode of the distribution, can be efficiently computed (at least approximated) using the *max-product algorithm*. In the following, we introduce the factor-graph designed to address NAP.

The factor-graph

We first define the set of variable nodes $X = \{X_{ii'} \in \{0,1\}, ii' \in V_A \times V_B\}$. These variables record the correspondences belonging to the current mapping. We then introduce the different function nodes of our graphical model. We distinguish factors providing the energy to the objective function from factors encoding the program constraints.

On one hand, the objective function is encoded via two sets of function nodes $c_{ii'} : \{0,1\} \to \mathbb{R}^+$ and $c_{ii'jj'} : \{0,1\}^2 \to \mathbb{R}^+$, such that $\forall ii' \in V_A \times V_B, c_{ii'}(x_{ii'}) = e^{\alpha x_{ii'} p_{ii'}}$, and $\forall ii', jj' \in (V_A \times V_B)^2, c_{ii'jj'}(x_{ii'}, x_{jj'}) = e^{(1-\alpha) x_{ii'} Q_{ii'jj'} x_{jj'}}$

On the other hand, the hard-constraints of NAP are encoded using $\{0,1\}$ Dirac measures $f_i : \{0,1\}^{|\partial f_i|} \to \{0,1\}$, and similarly for $g_{i'}$, such that:

$$\forall i \in V_A, f_i(x_{\partial f_i}) = \begin{cases} 1, & \text{if } \sum_{j' \in V_B} x_{ij'} \leq 1. \\ 0, & \text{otherwise.} \end{cases} \qquad \forall i' \in V_B, g_{i'}(x_{\partial g_{i'}}) = \begin{cases} 1, & \text{if } \sum_{j \in V_A} x_{ji'} \leq 1. \\ 0, & \text{otherwise.} \end{cases}$$

where we denote $x_{\partial f_i} = \{x_{ij'} \in \mathbf{x}, j' \in V_B\}$, and similarly, $x_{\partial g_{i'}} = \{x_{ji'} \in \mathbf{x}, j \in V_A\}$.

By factorizing all the function nodes, we obtain the probability distribution of our factor-graph:

$$p_X(\mathbf{x}) = \frac{1}{Z}\left[\prod_{i=1}^{n} f_i(x_{\partial f_i}) \prod_{j=1}^{m} g_j(x_{\partial g_j})\right] e^{\alpha \mathbf{x}^T \mathbf{p} + (1-\alpha)\mathbf{x}^T Q \mathbf{x}}, \tag{1}$$

where the normalization constant Z denotes the partition function of the model. It is clear that the support of our model distribution (1) is equivalent to the set of feasible solutions in NAP. Furthermore, the vector $\mathbf{x}$ with maximum probability corresponds to the optimal solution of the NAP.

The message passing framework

The main interest of the factor-graph (1) is the ability to efficiently compute an approximation of its mode using the *max-product algorithm*. Following the message passing framework proposed by Pearl [22], and denoting by $m^{(t)}$ the value of the message m after t iterations, we may apply the following updates [2] :

$$\mu^{(t+1)}_{X_{ii'} \to f_i}(x_{ii'}) = \lambda^{(t)}_{c_{ii'} \to X_{ii'}}(x_{ii'}) \lambda^{(t)}_{g_{i'} \to X_{ii'}}(x_{ii'}) \prod_{jj'} \lambda^{(t)}_{c_{ii'jj'} \to X_{ii'}}(x_{ii'})$$

$$\mu^{(t+1)}_{X_{ii'} \to g_{i'}}(x_{ii'}) = \lambda^{(t)}_{c_{ii'} \to X_{ii'}}(x_{ii'}) \lambda^{(t)}_{f_{i'} \to X_{ii'}}(x_{ii'}) \prod_{jj'} \lambda^{(t)}_{c_{ii'jj'} \to X_{ii'}}(x_{ii'})$$

$$\mu^{(t+1)}_{X_{ii'} \to c_{ii'jj'}}(x_{ii'}, x_{jj'}) = \lambda^{(t)}_{c_{ii'} \to X_{ii'}}(x_{ii'}) \lambda^{(t)}_{f_{i'} \to X_{ii'}}(x_{ii'}) \lambda^{(t)}_{g_{i'} \to X_{ii'}}(x_{ii'}) \prod_{kk' \neq jj'} \lambda^{(t)}_{c_{ii'kk'} \to X_{ii'}}(x_{ii'})$$

$$\lambda^{(t)}_{f_i \to X_{ii'}}(x_{ii'}) = \max_{\mathbf{x}_{\partial f_i \setminus \{ii'\}}} f_i(\mathbf{x}_{\partial f_i}) \prod_{j' \neq i'} \mu^{(t)}_{X_{ij'} \to f_i}(x_{ij'}), \qquad \lambda^{(t)}_{c_{ii'} \to X_{ii'}}(x_{ii'}) = c_{ii'}(x_{ii'})$$

$$\lambda^{(t)}_{g_{i'} \to X_{ii'}}(x_{ii'}) = \max_{\mathbf{x}_{\partial g_{i'} \setminus \{ii'\}}} g_{i'}(\mathbf{x}_{\partial g_{i'}}) \prod_{j \neq i} \mu^{(t)}_{X_{ji'} \to g_{i'}}(x_{ji'}) \quad \lambda^{(t)}_{c_{ii'jj'} \to X_{ii'}}(x_{ii'}) = \max_{x_{jj'}} c_{ii'jj'}(x_{ii'}, x_{jj'}) \mu^{(t)}_{X_{jj'} \to c_{ii'jj'}}(x_{jj'})$$

Since $x_{ii'}$ are binary valued, computing any message $\lambda_{a \to b}(x_{ii'})$ for $x_{ii'} = 0$ and $x_{ii'} = 1$ is redundant. Therefore, we may halve the computation cost, by only considering its log-ratio $m_{a \to b} = \log \frac{\lambda_{a \to b}(1)}{\lambda_{a \to b}(0)}$. Following this notation, it can be shown [2] that the messages from the variable nodes to the function nodes introduced above simplify to:

$$m^{(t)}_{f_i \to X_{ii'}} = -\left(\max_{k' \neq i'} m^{(t)}_{X_{ik'} \to f_i}\right)_+, \qquad m^{(t)}_{c_{ii'} \to X_{ii'}} = \alpha p_{ii'}$$

$$m^{(t)}_{g_{i'} \to X_{ii'}} = -\left(\max_{k \neq i} m^{(t)}_{X_{ki'} \to g_{i'}}\right)_+, \qquad m^{(t)}_{c_{ii'jj'} \to X_{ii'}} = \left((1-\alpha) Q_{ii'jj'} + m^{(t)}_{X_{jj'} \to c_{ii'jj'}}\right)_+ - \left(m^{(t)}_{X_{jj'} \to c_{ii'jj'}}\right)_+$$

where we use the notations: $x_+ = \max(0, x)$. Consequently, the message-passing framework reduces to the following updates:

$$m^{(t+1)}_{X_{ii'} \to f_i} = m^{(t)}_{c_{ii'} \to X_{ii'}} + m^{(t)}_{g_{i'} \to X_{ii'}} + \sum_{jj'} m^{(t)}_{c_{ii'jj'} \to X_{ii'}} \tag{2}$$

$$m^{(t+1)}_{X_{ii'} \to g_{i'}} = m^{(t)}_{c_{ii'} \to X_{ii'}} + m^{(t)}_{f_i \to X_{ii'}} + \sum_{jj'} m^{(t)}_{c_{ii'jj'} \to X_{ii'}} \tag{3}$$

$$m^{(t+1)}_{X_{ii'} \to c_{ii'jj'}} = m^{(t)}_{c_{ii'} \to X_{ii'}} + m^{(t)}_{f_i \to X_{ii'}} + m^{(t)}_{g_{i'} \to X_{ii'}} + \sum_{kk' \neq jj'} m^{(t)}_{c_{ii'kk'} \to X_{ii'}} \tag{4}$$

3.2. Proposed modifications

Solution assignment

After each message-passing iteration, the algorithm must compute the current best solution based on the updated messages. In their work, Bayati et al. [2] proposed several mechanisms, called "rounding strategies". Unfortunately, all of them require to address an instance of the MWM problem. Even worse, according to the authors, the best rounding strategy requires to solve exactly two MWM problems. Though this can be done in reasonable time, proceeding to this computational step after each iteration seriously slows down the algorithm.

To overcome this issue, we propose another simple assignment procedure based on the current estimated "max-marginals". In fact, following the notation introduced in Section 3.1, and referring to the computation rules of Pearl [22], we may estimate the max-marginal distribution of each variable node such that:

$$\hat{p}^{(t)}_{X_{ii'}}(x_{ii'}) \propto \lambda^{(t)}_{c_{ii'}\to X_{ii'}}(x_{ii'})\lambda^{(t)}_{f_i\to X_{ii'}}(x_{ii'})\lambda^{(t)}_{g_{i'}\to X_{ii'}}(x_{ii'})\prod_{jj'}\lambda^{(t)}_{c_{ii'jj'}\to X_{ii'}}(x_{ii'})$$

After t iterations, we may deduce the current best assignment from the sign of each max-marginal log-ratios $\log\frac{\hat{p}^{(t)}_{X_{ii'}}(1)}{\hat{p}^{(t)}_{X_{ii'}}(0)}$:

$$\hat{X}_{ii'} = \underset{x\in\{0,1\}}{\operatorname{argmax}}\,\hat{p}^{(t)}_{X_{ii'}}(x_{ii'}) = \operatorname{sign}\left(m^{(t)}_{c_{ii'}\to X_{ii'}} + m^{(t)}_{f_i\to X_{ii'}} + m^{(t)}_{g_{i'}\to X_{ii'}} + \sum_{jj'} m^{(t)}_{c_{ii'jj'}\to X_{ii'}}\right)$$

Note that this mechanism may result in a partial mapping. Therefore, after the last iteration, i.e. when the updates converge or reach the maximum number of iterations, we propose to enhance the resulting assignment with less confident matches by solving a MWM problem on the estimated max-marginal log-ratio of each unmatched node $\hat{X}_{ii'}$.

Auction based ϵ-complementary slackness

A well known problem of the *Max-Product algorithm* when running on loopy graphical models is that it is not guaranteed to converge. In fact, it may fall into infinite loops and oscillate between few states [19]. Therefore, most implementations include a mechanism that enforces convergence [6, 13]. In their work, Bayati et al. [2] propose a damping factor to mitigate the updates over iterations. Once this damping is sufficiently low, the message updates become insignificant, and the algorithm converges.

In our work, we propose another mechanism based on the concept of ϵ-complementary slackness [5]. This relaxation has been originally proposed for the *Auction algorithm* to address MWM instances that admit multiple optimal solutions. The idea is to prevent the saturation of the complementary slackness with a small constant ϵ margin. This scheme not only breaks ties and ensures the convergence but also provably finds to the optimal solution for an ϵ small enough [5]. Furthermore, for larger ϵ values, it shows to generally provide near-optimal assignments in much less computation time. Though very similar in its MWM version ($\alpha = 1$), our model is quite different from an *Auction algorithm* in the general case. In order to adapt the idea of ϵ-complementary slackness to our message-passing scheme, we propose the following modifications of updates (2) and (3):

$$m^{(t)}_{f_i\to X_{ii'}} = -\left(\max_{k'\neq i'} m^{(t)}_{X_{ik'}\to f_i}\right)_+ - 1_{m^{(t)}_{X_{ii'}\to f_i}\neq\max_{k'} m^{(t)}_{X_{ik'}\to f_i}}\epsilon \qquad m^{(t)}_{g_{i'}\to X_{ii'}} = -\left(\max_{k\neq i} m^{(t)}_{X_{ki'}\to g_{i'}}\right)_+ - 1_{m^{(t)}_{X_{ii'}\to g_{i'}}\neq\max_{k} m^{(t)}_{X_{ki'}\to g_{i'}}}\epsilon$$

In our experiments, this mechanism shows to strongly favor the messages convergence and thus reduce the number of required running iterations. More importantly, this scheme tends to improve the overall final assignment score in many cases.

Table 1: Description of our benchmark dataset.

Source	A	B	$\lvert V_A\rvert$	$\lvert V_B\rvert$	$\lvert E_A\rvert$	$\lvert E_B\rvert$
Zaslavskiy et al. [27]	1EWK	1U19	57	59	3192	2974
El-Kebir et al. [11]	dmela	scere	9459	5696	25635	31261
Zhang and Tong [28]	flickr	lastfm	15436	12974	32638	32298
Bayati et al. [2]	lcsh	wiki	1919	2000	3130	7808
Zhang and Tong [28]	offline	online	1118	3906	3022	16328

However, this relaxation suffers from an important drawback: the value of the introduced ϵ must be chosen carefully. If set too small, the mechanism cannot fully play its part and the algorithm may reach a maximum number of iterations before converging. On the contrary, if ϵ is too high, the algorithm tends to converge too quickly to a poor local optimum. In their work, Bertsekas [5] propose an iterative method, called ϵ-scaling, to properly setup the relaxation. The idea consists in repeatedly decreasing ϵ after the messages converged, until it reaches a small enough value, known to provide an optimal solution. In our work, we suggest the opposite scheme. The model starts with a rather small ϵ that helps to softly break local ties. Then, as the algorithm iterates, we propose to rise the relaxation value each time the messages have not improved the current objective function for few iterations. As ϵ rises, the messages are more and more likely to escape their local optimum and to fall into another better one. As soon as the current assignment improves, ϵ is set back to its original value, such that the new local solutions can be carefully explored.

4. Evaluation

The proposed evaluation of our method, named QBinDiff, is twofold. We first analyze its performances as a NAP solver. To do so, we submit the exact same problems to several state of the art solvers, and compare the computed alignment scores, without any consideration about the underlying purpose of the problem instance. Then, we evaluate the relevance of our solution in order to address the binary diffing problem. Therefore, we compare the resulting mappings to *ground truth* assignments and evaluate each matching method with respect to accuracy metrics. In all our experiments, we ran our method with default parameters: $\epsilon = 0.5$, and within a maximum number of 1000 iterations.

4.1. Benchmark experiments

We compare our method to four state-of-the-art NAP solvers and their associated benchmarks (see Table 1): PATH [27], NetAlign [2], Natalie 2.0 [11], and Final [28]. All these solvers have been configured with their default parameters. In order to analyze the ability of each solver to provide proper solutions both in terms of node similarity and edge overlaps, we tested different values of the trade-off parameter α. Notice that some problems include a similarity score matrix with several zero entries. In some models (ours, NetAlign, Natalie), those entries are considered as unfeasable matches whereas they are legal correspondences in others. These models would thus optimize the problem on a subset of all possible one-to-one mappings.

Our results show that our approach outperforms or nearly ties the other existing methods on every problems (see Table 2). It appears to provide better results on sparse graphs, while it may compute slightly suboptimal assignments on the densest one (1EWK-1U19). It also seems to be the best fitted to perform diffing at different arbitrary setting of the trade-off parameter α, even in the degenerated MCS case ($\alpha = 0$), unlike most other evaluated methods.

In terms of computing time, as expected, QBinDiff takes much less time to approximate the NAP than NetAlign. Regarding other solvers, both Natalie and Final run within comparable time while Path tends to be very expensive, and may become prohibitive for larger problem instances. Note that it was not able to provide a solution to the Flickr-Lastm problem when $\alpha = 0.25$ within 8 days, and was considered timed-out. Of course these timings depends on the quality of the implementation and should only be considered with respect to their order of magnitude. We used the implementations provided by the authors of other methods.

4.2. Binary Diffing experiments

In a second set of experiments, we propose to evaluate the relevance of our model in order to address the *Binary Diffing problem*. Given two binary executables A and B, this problem consists in retrieving the correspondence between the functions of A and those of B that best describes their semantic differences. This problem can be reduced to a network alignment problem over the call graphs of A and B.

The results of the alignment should be compared to some ground truth. We computed it as follows. We first downloaded the official repository of a program, then compiled the different available versions using GCC v7.5 for x86-64 target architecture and with -O3 optimization level. Once extracted, each binary was stripped to remove all symbols, then disassembled using IDA Pro v7.2[1], and finally exported into a readable file with the help of BinExport[2]. During the problem statement, only plain text functions determined during the disassembly process are considered. For each program, assuming that this extraction protocol provided us with n different versions, we propose to evaluate our method in diffing all the $\frac{n(n-1)}{2}$ possible pairs of different binaries.

For all these diffing instances, we finally had to extract the ground truth assignments. We proceed in two steps. We first manually determine what we think to be the function mapping that best describes the modifications between two successive program versions. This is done considering the binary symbols, as well as the explicit commit descriptions. Then, we deduce all the remaining pairwise diffing assignments by extrapolating the mappings from version to versions. Formally, if we encode the mapping between A_1 and A_2 into a boolean matrix $M_{A_1 \to A_2}$ such that $M_{A_1 \to A_2 ii'} = 1$ if and only if function i in A_1 is paired with function i' in A_2, then, our extrapolating scheme simply consists in computing the diffing correspondence between A_k and A_n as follows: $M_{A_k \to A_n} = \prod_{i=k}^{n-1} M_{A_i \to A_{i+1}}$. We applied this extraction protocol to three well known open source programs, namely Zlib [3], Libsodium [4] and OpenSSL [5] from which we collected respectively 18, 33 and 17 different binary versions and therefore 153, 528 and 136 diffing instances. Statistics describing our evaluation dataset are given in Table 3.

As a baseline, we chose to compare to the two most common diffing tools: BinDiff and Diaphora. BinDiff uses a matching algorithm close to VF2 [8] originally introduced to approximate the MCS whereas Diaphora proposes a different greedy assignment strategy that first matches the most similar functions and then searches for potential correspondences in the remaining ones. This mapping mechanism is known to provide $\frac{1}{2}$-approximate solutions to the MWM problem [9]. Note that, in order to compare with NAP solvers, and because, in general, binary diffing favors recall over precision, QBinDiff is designed to produce a complete assignment and does not include a mechanism to limit the mapping of very unlikely correspondences during computation.

Our experiments show that QBinDiff generally outperforms other matching approaches in both alignment score and recall (see Table 4). In fact, our method appears to perform clearly better at diffing more different programs, whereas it provides comparable solutions on similar binaries. This highlights that the local greedy matching strategies of both BinDiff and Diaphora are able to provide good solutions on simple cases but generalize poorly on more difficult problem instances. This results should be view as promising in the perspective of diffing much more different binaries.

We reproduced our experiments with different trade-off parameters α, in order to estimate which setup should be used such that the optimal assignment meets the ground truth. Our computations suggest that a trade-off around 0.75 is a fair choice though a slightly higher value could provide satisfying assignments as well, especially while diffing OpenSSL programs.

4.3. Limitations

Though our method improves the state-of-the-art, some difficulties remain.

A first limitation of our approach concerns the design of the network alignment itself. Indeed, the determination of the trade-off parameter α is subject to arbitrary considerations and may have an important impact on the resulting

[1] https://www.hex-rays.com/products/ida
[2] https://github.com/google/binexport
[3] https://github.com/madler/zlib
[4] https://github.com/jedisct1/libsodium
[5] https://github.com/openssl/openssl

Table 2: Resulting objective scores of each solver on different benchmark problems. The last column records the average computing time in seconds.

Problem	Matcher	$\alpha = 0.0$	$\alpha = 0.25$	$\alpha = 0.5$	$\alpha = 0.75$	$\alpha = 0.9$	Time
1EWK-1U19	QBinDiff	2890.000	2183.366	1473.608	764.758	**339.628**	22.532
	Final	2874.000	2169.750	1465.500	761.250	338.700	17.944
	Natalie	**2896.000**	2172.496	1448.992	725.488	291.385	852.921
	NetAlign	**2896.000**	**2185.750**	**1474.016**	**765.023**	338.700	1620.431
	Path	**2896.000**	2169.750	1465.500	761.250	338.700	0.700
dmela-scere	QBinDiff	**255.000**	**347.937**	**441.554**	**543.832**	**616.475**	32.784
	Final	112.000	250.953	389.906	528.859	612.230	45.898
	Natalie	174.000	163.109	142.558	126.837	119.156	8.677
	NetAlign	224.000	332.401	431.732	543.442	615.557	115.397
	Path	47.000	230.809	376.118	522.177	609.812	17951.033
flickr-lastfm	QBinDiff	**6144.000**	**6955.018**	**7775.695**	**8594.405**	**9118.512**	412.266
	Final	1980.000	3890.405	5800.810	7711.215	8857.458	156.849
	Natalie	6012.000	5164.405	4282.425	3417.638	2896.056	94.092
	NetAlign	5830.000	6742.407	7567.495	8427.405	9025.683	434531.487
	Path	10.000	NA	4316.110	7403.845	8845.276	387114.102
lcsh-wiki	QBinDiff	**632.000**	**614.867**	**612.690**	605.062	614.086	36.344
	Final	574.000	585.217	596.435	**607.652**	**614.383**	20.302
	Natalie	584.000	496.465	419.760	337.640	287.927	2.932
	NetAlign	506.000	547.398	552.335	584.123	610.774	52.317
	Path	238.000	385.085	462.935	539.480	594.963	4332.587
offline-online	QBinDiff	**2608.000**	**2138.206**	**1678.450**	**1103.162**	**812.036**	95.696
	Final	40.000	222.352	404.704	587.055	696.466	22.685
	Natalie	198.000	315.160	327.387	392.081	446.390	1553.770
	NetAlign	1772.000	1507.908	1352.685	1002.279	756.600	27971.596
	Path	244.000	789.614	725.422	690.011	742.723	13881.264

Table 3: Description of our binary diffing dataset. The last five columns respectively record the number of different binary versions, the number of resulting diffing instances, the average number of functions and function calls and the average ratio of conserved functions in our manually extracted ground truth.

Program	Versions	Problems	$\overline{\lvert V\rvert}$	$\overline{\lvert E\rvert}$	GT ratio
Zlib	18	153	153	235	0.96
Libsodium	33	528	589	701	0.79
OpenSSL	17	136	3473	18563	0.72

solution. Moreover, this trade-off certainly depends on the density of the graphs since densest graphs mechanically include more potential edge overlaps. Finally, the proposed model is designed to compute complete assignments and therefore does not include any mechanism to optimize the precision. This later could be done in introducing a penalty term ζ to the node similarity scores such $p_{ii'} = \sigma_V(i, i') - \zeta$. Correspondences with negative similarity scores would thus belong to the final mapping if they induce enough topological similarity.

Table 4: Average normalized resulting scores for each matching method. Computing time in seconds.

Project	Matcher	Similarity	Squares	Objective	Precision	Recall	Time
Zlib	QBinDiff	0.929	0.807	**0.888**	0.955	**0.995**	0.245
	BinDiff	0.921	0.765	0.869	0.943	0.975	1.239
	Diaphora	0.863	0.655	0.792	**0.978**	0.940	10.494
	Final	0.929	0.784	0.881	0.948	0.986	18.469
	Natalie	0.587	0.812	0.663	0.901	0.603	0.354
	NetAlign	0.929	0.807	**0.888**	0.955	**0.995**	21.935
	Path	0.930	0.789	0.883	0.953	0.992	0.589
Libsodium	QBinDiff	0.706	0.604	**0.677**	0.722	**0.880**	6.616
	BinDiff	0.662	0.544	0.628	0.752	0.869	1.278
	Diaphora	0.605	0.470	0.567	**0.783**	0.831	35.327
	Final	0.710	0.455	0.636	0.686	0.829	30.264
	Natalie	0.424	0.565	0.462	0.596	0.438	36.984
	NetAlign	0.703	0.597	0.673	0.722	0.879	283.130
	Path	0.700	0.519	0.648	0.696	0.836	61.784
OpenSSL	QBinDiff	0.720	0.595	**0.643**	0.605	**0.783**	213.252
	BinDiff	0.643	0.501	0.553	0.572	0.681	3.944
	Diaphora	0.543	0.332	0.408	0.577	0.565	228.666
	Final	0.719	0.355	0.487	0.476	0.627	264.397
	Natalie	0.620	0.589	0.601	0.599	0.668	733.802
	NetAlign	0.682	0.587	0.623	**0.625**	0.755	15193.922
	Path	0.722	0.521	0.596	0.556	0.715	7667.840

Another difficulty of our approach is the determination of the relaxation parameter ϵ. As previously mentioned, its setting controls on a trade-off between the quality of the solution and the speed at which the messages converges. While the proposed solution gives very satisfactory results, other rising schemes could be used to adapt the parameter to the current solution during computation. We leave this investigations to future work.

Finally, our formulation is limited by its memory consumption associated to the use of a quartic memory matrix Q. Though the proposed model enables to significantly reduce the problem size by limiting the solution set to the most probable correspondences, this relaxation inevitably induces information loss, especially for large graphs where the relaxation must rise consequently. In practice, graphs of several thousands of nodes can be handled efficiently. For larger instances, a solution could consists in first partitioning the graphs into smaller consistent subgraphs, and then proceed the matching among them. However such partition is not trivial and might result in important alignment errors.

5. Conclusion

In this paper, we introduced a new algorithm to address the network alignment problem. It leverages a previous model and includes new mechanism to enforce message convergence as well as speed-up the computation. Moreover, it proved to provide better assignments than the original version in almost all alignment instances.

Our evaluation showed that our approach outperforms state of the art solvers. It also appears to be very well fitted to compute proper solutions for different trade-offs between node similarity and edge overlaps. We finally proposed an application of our method to address the binary diffing problem. Our experiments showed that our algorithm provides

better assignments than other existing approaches for most diffing instances. This result suggests that the formulation of the binary diffing problem as a network alignment problem is the correct approach.

References

[1] Bahiense, L., Manić, G., Piva, B., De Souza, C.C., 2012. The maximum common edge subgraph problem: A polyhedral investigation. Discrete Applied Mathematics 160, 2523–2541.

[2] Bayati, M., Gerritsen, M., Gleich, D.F., Saberi, A., Wang, Y., 2009. Algorithms for Large, Sparse Network Alignment Problems, in: Proceedings of the 2009 Ninth IEEE International Conference on Data Mining, IEEE Computer Society, USA. pp. 705–710.

[3] Bayati, M., Gleich, D.F., Saberi, A., Wang, Y., 2013. Message-Passing Algorithms for Sparse Network Alignment. ACM Transactions on Knowledge Discovery from Data (TKDD) 7, 3:1–3:31.

[4] Bayati, M., Shah, D., Sharma, M., 2005. Maximum weight matching via max-product belief propagation, in: Proceedings. International Symposium on Information Theory, 2005. ISIT 2005., pp. 1763–1767.

[5] Bertsekas, D.P., 1992. Auction algorithms for network flow problems: A tutorial introduction. Computational Optimization and Applications 1, 7–66.

[6] Braunstein, A., Zecchina, R., 2006. Learning by Message Passing in Networks of Discrete Synapses. Physical Review Letters 96, 030201.

[7] Burkard, R.E., Çela, E., Pardalos, P.M., Pitsoulis, L.S., 1998. The Quadratic Assignment Problem, in: Du, D.Z., Pardalos, P.M. (Eds.), Handbook of Combinatorial Optimization: Volume1–3. Springer US, Boston, MA, pp. 1713–1809.

[8] Cordella, L.P., Foggia, P., Sansone, C., Vento, M., 1998. Subgraph Transformations for the Inexact Matching of Attributed Relational Graphs, in: Jolion, J.M., Kropatsch, W.G. (Eds.), Graph Based Representations in Pattern Recognition, Springer, Vienna. pp. 43–52.

[9] Duan, R., Pettie, S., 2014. Linear-Time Approximation for Maximum Weight Matching. Journal of the ACM 61, 1:1–1:23.

[10] El-Kebir, M., Heringa, J., Klau, G.W., 2011. Lagrangian relaxation applied to sparse global network alignment, in: Proceedings of the 6th IAPR international conference on Pattern recognition in bioinformatics, Springer-Verlag, Delft, The Netherlands. pp. 225–236.

[11] El-Kebir, M., Heringa, J., Klau, G.W., 2015. Natalie 2.0: Sparse Global Network Alignment as a Special Case of Quadratic Assignment. Algorithms 8, 1035–1051.

[12] Feizi, S., Quon, G., Recamonde-Mendoza, M., Médard, M., Kellis, M., Jadbabaie, A., 2020. Spectral Alignment of Graphs. IEEE Transactions on Network Science and Engineering 7, 1182–1197.

[13] Frey, B.J., Dueck, D., 2007. Clustering by Passing Messages Between Data Points. Science 315, 972–976.

[14] Horaud, R., Forbes, F., Yguel, M., Dewaele, G., Zhang, J., 2011. Rigid and Articulated Point Registration with Expectation Conditional Maximization. IEEE Transactions on Pattern Analysis and Machine Intelligence 33, 587–602.

[15] Klau, G.W., 2009. A new graph-based method for pairwise global network alignment. BMC Bioinformatics 10, S59.

[16] Kollias, G., Mohammadi, S., Grama, A., 2012. Network Similarity Decomposition (NSD): A Fast and Scalable Approach to Network Alignment. IEEE Transactions on Knowledge and Data Engineering 24, 2232–2243.

[17] Kschischang, F.R., Frey, B.J., Loeliger, H.A., 2006. Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory 47, 498–519.

[18] Lyzinski, V., Fishkind, D.E., Fiori, M., Vogelstein, J.T., Priebe, C.E., Sapiro, G., 2016. Graph Matching: Relax at Your Own Risk. IEEE Transactions on Pattern Analysis and Machine Intelligence 38, 60–73.

[19] Murphy, K.P., Weiss, Y., Jordan, M.I., 1999. Loopy belief propagation for approximate inference: an empirical study, in: Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., Stockholm, Sweden. pp. 467–475.

[20] Nassar, H., Veldt, N., Mohammadi, S., Grama, A., Gleich, D.F., 2018. Low Rank Spectral Network Alignment, in: Proceedings of the 2018 World Wide Web Conference, International World Wide Web Conferences Steering Committee, Lyon, France. pp. 619–628.

[21] Patro, R., Kingsford, C., 2012. Global network alignment using multiscale spectral signatures. Bioinformatics 28, 3105–3114.

[22] Pearl, J., 1982. Reverend bayes on inference engines: a distributed hierarchical approach, in: Proceedings of the Second AAAI Conference on Artificial Intelligence, AAAI Press, Pittsburgh, Pennsylvania. pp. 133–136.

[23] Sahni, S., Gonzalez, T., 1976. P-Complete Approximation Problems. Journal of the ACM (JACM) 23, 555–565.

[24] Singh, R., Xu, J., Berger, B., 2008. Global alignment of multiple protein interaction networks with application to functional orthology detection. Proceedings of the National Academy of Sciences 105, 12763–12768.

[25] Umeyama, S., 1988. An Eigendecomposition Approach to Weighted Graph Matching Problems. IEEE Transactions on Pattern Analysis and Machine Intelligence 10, 695–703.

[26] Vogelstein, J.T., Conroy, J.M., Lyzinski, V., Podrazik, L.J., Kratzer, S.G., Harley, E.T., Fishkind, D.E., Vogelstein, R.J., Priebe, C.E., 2014. Fast Approximate Quadratic Programming for Large (Brain) Graph Matching. arXiv:1112.5507 [cs, math, q-bio] .

[27] Zaslavskiy, M., Bach, F., Vert, J.P., 2009. A Path Following Algorithm for the Graph Matching Problem. IEEE Transactions on Pattern Analysis and Machine Intelligence 31, 2227–2242.

[28] Zhang, S., Tong, H., 2016. FINAL: Fast Attributed Network Alignment, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, NY, USA. pp. 1345–1354.

[29] Zhang, Z., Xiang, Y., Wu, L., Xue, B., Nehorai, A., 2019. KerGM: Kernelized Graph Matching, in: Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F.d., Fox, E., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 32. Curran Associates, Inc., pp. 3335–3346.

[30] Zhou, F., 2012. Factorized graph matching, in: Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, USA. pp. 127–134.