Focused SANA: Speeding Up Network Alignment

Ilia Leybovich, Rami Puzís, Roni Stern, Maor Reuben

Ben Gurion University of the Negev, Be’er Sheva, Israel
ilialey@post.bgu.ac.il, puzis@bgu.ac.il, sternron@bgumail.bgu.ac.il, maorreu@post.bgu.ac.il

Abstract

Network Alignment (NA) is a generalization of the graph isomorphism problem for non-isomorphic graphs, where the goal is to find a node mapping as close as possible to isomorphism. Recent successful NA algorithms follow a search-based approach, such as simulated annealing. We propose to speed up search-based NA algorithms by pruning the search-space based on heuristic rules derived from the topological features of the aligned nodes. We define several desirable properties of such pruning rules, analyze them theoretically, and propose a pruning rule based on nodes’ degrees. Experimental results show that using the proposed rule yields significant speedup and higher alignment quality compared to the state of the art. In addition, we redefine common NA objective functions in terms of established statistical analysis metrics, opening a wide range of possible objective functions.

Introduction

Network Alignment (NA) is a generalization of the graph isomorphism problem for non-isomorphic graphs, where the goal is to find a node mapping, as close to isomorphism as possible. NA is both a problem domain and a component of many algorithms in the Artificial Intelligence (AI) field, see a survey by (Frank, et al., 2016). Finding NA is useful in many AI problem domains including pattern recognition (Conte, et al., 2004), image recognition (Hsieh & Hsu, 2004), NLP (Bayati, et al., 2013), ontology mapping (Li, et al., 2009), and bio-informatics (Elmsallati, et al., 2016). State of the art methods for finding NA employ network embedding (Liu, et al., 2016) heuristic search techniques, e.g., simulated annealing (Mamano & Hayes, 2017), genetic algorithms (Clark & Kalita, 2015; Saraph & Milenković, 2014). SANA is a state-of-the-art NA method based on simulated annealing that was shown to be superior to many alternative methods in almost all parameters (Mamano & Hayes, 2017; Kanne & Hayes, 2017). Unfortunately, the NA search space grows exponentially with the number of nodes in the aligned networks, which limits the scalability of SANA. Thus, SANA and similar local search algorithms may not be applicable to large networks with hundreds of thousands or millions of nodes.

The primary contribution of this paper is a theory for effective pruning of the NA search space. This pruning focuses the search on alignments that are more likely to be valuable and ignoring alignments where dissimilar nodes are mapped to each other. We present a sufficient condition for such pruning rules that maintains a connected search space. Then, we propose a heuristic pruning rule that satisfies this condition and show empirically that it improves the performance of SANA in both speed and accuracy. For example, on a benchmark of biological networks, the proposed algorithm found a better alignment than SANA in a third of the time.

Beyond improving the search process, we also provide a deeper understanding of the NA objective functions. Finding an appropriate objective function is a hot topic in NA research and a range of objective functions have been proposed in recent years (Kanne & Hayes, 2017). Many of them are ad-hoc and their correctness and effectiveness are only analyzed experimentally. We reframe the NA problem in terms of a prediction problem. This allows importing well-established prediction quality metrics to the field of NA, along with their theoretical guarantees and background.

Problem Definition and Background

There are several definitions of the Network Alignment (NA) problem. We focus on what is known as a pairwise global alignment, defined as follows. Let $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ be two undirected and unweighted graphs (networks), where $V_i$ and $E_i$ denote the nodes and edges respectively and $|V_1| \leq |V_2|$. A pairwise global alignment between $G_1$ and $G_2$ is an injective function $\alpha: V_1 \rightarrow V_2$. Let $\alpha^*$ denote the ideal, unknown, alignment function. The NA problem we address in this paper is to find an alignment that is as close as possible to $\alpha^*$. A common measure for evaluating an alignment $\alpha$ is node correctness, which is the percentage of correctly aligned nodes. That is...
Two alignments are considered neighbors if we can obtain one by a single alignment manipulation action on the other. The temperature schedule used by SANA is an exponential decay function (see Mamano & Hayes, 2017) for details). SANA was shown to be very effective in practice across a range of objective functions. In fact, Mamano and Hayes (2017) consider investigation of effective objective functions for search-based NA as the primary direction for further research.

Limitations of SANA
Although SANA has many advantages, it also has a few shortcomings. First, SANA does not scale well to very large networks. This is because the size of its search space is a factorial of the number of nodes in the aligned networks ($|V_2|!(|V_2|−|V_1|)!$, and the branching factor is $(|V_2| + 0.5|V_1|(|V_1| − 1)$), grows quadratically with the size of $G_1$. Second, SANA is blind to node properties that are not encompassed by the objective function. Thus, if additional knowledge about the correct alignment is available, SANA cannot consider it without merging it into a single objective function, which is not always possible. In addition, the creators of SANA observed that there are many different alignments with close to perfect $S^3$ score but far from perfect NC score (Mamano & Hayes, 2017). The NA algorithm we propose addresses these shortcomings.

Focused SANA
Next, we present Focused SANA (F-SANA), an improvement over SANA. The key difference between SANA and F-SANA is two-fold. First, F-SANA creates an intelligent initial alignment. Second, it imposes additional restrictions over the state transitions that can be applied at every iteration of SANA. Informally, these restrictions verify that nodes mapped to each other in any considered alignment are similar, where nodes similarity can be computed based on their properties or based on network topology. Using these restrictions drastically reduces the size of the search space, and consequently improves the algorithm's running time. Also, we show that these restrictions result in finding better alignments.

To formally describe the F-SANA algorithm, we introduce the following definitions.

**Definition 1 [Ranking Function, Rank]:** For a network $G(V, E)$, $\text{Rank}: V \to \mathbb{N}$ is called the Ranking Function, and $\text{Rank}(v)$ is called the Rank of the node $v$.

The ranking induced by a ranking function, may be application dependent (e.g. for social networks it might be address, age etc.), or application independent (e.g. node's degree, betweenness etc.). A perfect ranking function for a given NA problem gives the same rank only to nodes that are aligned to each other in $\alpha^*$, i.e. for all $v \in V_1$ and $u \in V_2$ it holds that $(\text{Rank}(v) = \text{Rank}(u)) \iff (\alpha^*(v) = u)$. Finding a perfect ranking function is hard. For an imperfect ranking function,
the ranks of \( u \) and \( v \), such that \( a^*(v) = u \), may differ. To this end, we define a candidate function (CF) that defines for every rank \( r \) a set of ranks (\( CF(r) \subseteq \mathbb{N} \)). We expect nodes with rank \( r \) to be aligned by \( a^* \) to nodes with ranks in \( CF(r) \).

**Definition 2 [Candidate Function]:** For two networks \( G_1(V_1, E_1) \) and \( G_2(V_2, E_2) \), a Candidate Function \( CF: \text{Rank}(V_1) \rightarrow \text{powerset}(\text{Rank}(V_2)) \) is a function that maps a rank to a set of ranks.

For a given node \( v \in V_1 \), we call the set of nodes in \( V_2 \) with a rank in \( CF(\text{Rank}(v)) \) the set of candidates of \( v \). We say that NA is legal if each node in the source network is aligned to one of its candidates in the target network. Formally:

**Definition 3 [Legal NA]:** Given NA \( a: V_1 \rightarrow V_2 \) is Legal if \( \forall v \in V_1: \text{Rank}(a(v)) \in CF(\text{Rank}(v)) \).

The NA algorithm we propose, called F-SANA, uses a given candidate function to limit SANA to consider only alignment manipulation operations that result in a legal alignment. This is done by modifying the random state transition choice as follows.

Let \( a \) be the current alignment. For a change operation, we choose a random node \( v \) from \( V_1 \). Then, we choose a random node from the "unoccupied" candidates of \( v \), and the new alignment maps \( v \) to this candidate. For a swap operation, after choosing a random node \( v \) from \( V_1 \), we choose a random node \( u \) from the candidates of \( v \) that has an origin in \( V_1 \) in the current alignment, i.e., \( \exists v' \in V_1 \) such that \( a(v') = u \). The swap operation is only allowed if \( a(v) \) is a candidate of \( v' \). This can be implemented efficiently with appropriate data structures. Note that, for the trivial ranking function that maps all nodes in both networks to 1, F-SANA is exactly SANA. Thus, F-SANA is, in fact, a generalization of SANA. As we discuss below and show experimentally, it is possible to define a non-trivial candidate function that causes F-SANA to perform significantly better than SANA.

**Effective candidate functions**

The effectiveness of F-SANA depends on the chosen candidate function. We suggest three desirable properties for a candidate function:

1. **Completeness preserving (CP).** A candidate function is completeness preserving if every legal alignment can be reached from every other legal alignment by using only the state transition functions allowed by F-SANA. That is, the search space of F-SANA is connected when using this candidate function.

2. **Efficient.** A candidate function is efficient if its computation time is lower than the time saved during the search process due to the search space pruning.

3. **Optimality preserving (OP).** A candidate function is optimality preserving if \( a^* \) is legal w.r.t. its ranking function.

In addition, the running time of F-SANA depends on the number of candidates for every node. Thus, it is desirable for an effective candidate function to map as few candidates as possible for each node. However, there is a tradeoff, since we need to be careful to stay as close as possible to being OP.

**Conditions for a CP candidate function**

Some candidate functions are not CP (an example was omitted due to space limitations).

Next, we propose a sufficient condition for identifying that a given candidate function is CP. Checking whether it is also efficient requires a runtime complexity analysis. We leave providing sufficient conditions for OP to future research.

**Theorem 1:** Let \( G_1(V_1, E_1) \), \( G_2(V_2, E_2) \), be networks such that \( |V_1| \leq |V_2| \), with a ranking function \( \text{Rank} \) and a candidate function \( CF \). If \( \forall X \in \text{Imgs}(CF): X \) is continuous (i.e. \( \forall i, j \in X: i \leq h \leq j \rightarrow h \in X \)), then every 2 legal alignments are connected in the space of legal NAs.

Proof is omitted due to space constraints.

For example, consider two social networks where each node represents a person, and is associated with that person’s age and gender. Now assume we want to align these networks so that nodes that represent the same person are aligned to each other. Assuming that the edge and gender are correctly reported in both social networks, we can build a perfect ranking function by marking nodes with the same age and gender by the same number with respect to lexicographical order of age and gender. Obviously this candidate function is CP by Theorem 1. It is efficient since \( O(|V|) \) is much smaller than the complexity of the search part of the algorithm. It is also OP (assuming that the data is correct and the networks consist of the same people).

**NA Objective Functions**

F-SANA improves on SANA by focusing its search using the candidate functions. A key factor in the success of search algorithms is the objective functions they aim to optimize. In fact, (Mamano & Hayes, 2017) claimed that the research for better objective functions should be the main focus of NA research. Indeed, many objective functions have been proposed for NA. To choose in a principle manner the appropriate objective function, we map in this section common NA objective functions to classical measures from the Machine Learning (ML) literature for evaluating classifiers.

A binary ML classifier assigns either Positive (P) or Negative (N) label to any given instance (i.e. an object being classified). When the classifier is evaluated the predicted values are compared to the ground truth. Instances that are labelled by the classifier as Positive and are indeed positive according to the ground truth are called True Positive (TP) instances. Instances that are incorrectly classified as Positive by the classifier are called False Positive (FP) instances. True Negative (TN) and False Negative (FN) instances are defined symmetrically.
For the sake of the following discussion consider a classifier
\( cl: V_1 \times V_1 \rightarrow \{ P, N \} \) which assigns a Positive (P) label to a
pair of nodes if they are adjacent in \( G_1 \) and a Negative (N) label otherwise:

\[
cl((v_1, v_2)) = \begin{cases} P, & (v_1, v_2) \in E_1 \\ N, & Otherwise \end{cases}
\]

We say that an alignment \( a \) maps an edge \((v_1, v_2) \in E_1\) to an
edge in \( G_2 \) iff \( (a(v_1), a(v_2)) \in E_2 \). We expect all edges
in \( G_1 \) to be mapped to edges in \( G_2 \) and all non-adjacent pairs
of nodes in \( G_1 \) to be mapped to non-adjacent pairs of nodes
in \( G_2 \). Thus the classifier \( cl \) assigns the correct label for
\((v_1, v_2) \in E_1 \) iff \( (v_1, v_2) \in E_1 \iff (a(v_1), a(v_2)) \in E_2 \).

Next we define the confusion matrix (TP, FP, TN, FN) in
terms of NA. The TP is the set of edges in \( E_1 \subseteq V_1 \times V_1 \)
hence positive) that were mapped by \( a \) to edges in \( E_2 \) (hence true): \( TP = \{(v, v') \in E_1 : (a(v), a(v')) \in E_2 \} \).

FP is the set of edges in \( E_1 \) that were mapped to non-edges
in \( G_2 \): \( FP = \{(v, v') \in E_1 : (a(v), a(v')) \in E_2 \} \).

FN are the pairs of nodes which are not edges in \( G_1 \) (hence negative) that were mapped to edges of \( G_2 \) (hence false):
\( FN = \{(v, v') \in E_1 : (a(v), a(v')) \in E_2 \} \).

Finally, TN is the set of non-edges that were mapped to non-edges.

These observations allow representing many NA objective
functions in the same terms in which classification performance
measures are defined (e.g. precision, accuracy, recall). Moreover, we show (omitted due to space restrictions)
that some objective functions suggested in literature are "re-
invented" measures, e.g. \( EC(a) \) by (Kuchaiev, et al., 2010)
& \( IC\delta (a) \) by (Patro & Kingsford, 2012) are actually precision & recall respectively. Also the score \( S^3(a) = \frac{|TP|}{(|TP| + |FP| + |FN|)} \) is very similar to the well
known \( F_1 \)-score = \( \frac{2|TP|}{2|TP| + |FP| + |FN|} \). Beyond the elegance of mapping NA specific measures to
common measures from the ML literature, this mapping opens the door to importing a wide range of more sophisti-
cated objective functions from the ML literature back to NA,
leveraging the years of ML research behind them.

**Experiments**

To show the advantages of F-SANA in practice, we com-
pared it experimentally with SANA. In this paper we present
only one set of experiments due to space constraints.

As a ranking function, we used the nodes’ degrees. We ex-
perimented with several candidate functions, and ended up
using the following. First, we sorted the node of each network
by their degrees. Let \( v_1, \ldots , v_n \) and \( u_1, \ldots , u_n \) be
the nodes, where \( \deg(v_i) \leq \deg(v_{i+1}) \) and \( \deg(u_i) \leq \deg(u_{i+1}) \) for every \( i \). Then, we create an initial alignment, denoted \( a_0 \), by mapping nodes according to their order, i.e., \( a_0(v_i) = u_i \). For every rank \( r \) of nodes in \( V_1 \) we denote by
\( R(\nu) \) the set of nodes in \( V_2 \) that were mapped to nodes in \( V_1 \)
with rank \( r \):

\[
R(\nu) = \{ u \in V_2 \mid \exists v \in V_1 : a_0(v) = u \land \text{Rank}(v) = r \}
\]

Let \( Ranks(V_2) \) be the set of all ranks in \( V_2 \). We defined the
candidate function \( CF \) as follows

\[
CF(\nu) = \{ r' \mid r' \in Ranks(V_2) \land r' \in \{ m - \lceil \sqrt{\nu} \cdot M + \lceil \sqrt{\nu} \cdot M \} \}
\]

Intuitively, \( CF \) extends the initial alignment so that nodes of
the same rank will have the same set of candidates and this
set is continuous, hence \( CF \) is CP (Theorem 1). Also, we
extend the rank bounds by \( \lceil \sqrt{\nu} \cdot M \) and \( \lceil \sqrt{\nu} \cdot M \).

The dataset we experimented on is a standard NA benchmark
from (Collins, et al., 2007), that was used to evaluate
SANA and many other algorithms. It consists of 6 networks
denoted Y0, Y5, Y10, Y15, Y20, and Y25. Each network
represents a protein-protein interaction (PPI) network of
yeast. All the networks have 1004 nodes. The difference
between the six networks is the number of edges where YX
has \% more edges than Y0, which has 8323 edges.

To assess the quality of the alignments generated by SANA
and F-SANA, we used the NC (since we know \( a^* \)). In
addition, we report the \( S^3 \)-score and the Largest Common
Connected Subgraph (LCCS), which indicates whether the
alignment found a large similar subgraph (Kuchaiev, et al.,
2010; Saraph & Milenković, 2014).

We ran SANA and F-SANA to align the base network Y0
to each of its noisy variants Y05,..,Y25. Figure 1 shows the
NC, \( S^3 \), and LCCS results as a function of the number of
edges added to the base network (x-axis). Different data
lines correspond to different algorithms and time budgets (3,
5, and 10 minutes). The same general trend can be observed
in these results also: F-SANA is able to find higher quality
alignments, according to all evaluated measures (NC, \( S^3 \),
and LCCS) for every given time budget and network pair,
regardless of how many edges were added. Even F-SANA
with 3 minutes time budget (yellow solid line with diamond
markers) achieves better results than SANA with 10
minutes, in every measure and every network pair. These
results show the robustness of F-SANA’s advantage over
SANA. In conclusion, F-SANA exhibits higher quality net-
work alignments in shorter times compared to SANA over a
range of datasets, time budgets, and network variants.

**Conclusions**

We proposed two contributions to the NA search problem.

Primarily, we propose an effective way to perform search
space pruning for NA. We provide some theoretical foundation for this pruning and implement this approach into an improved algorithm based on SANA. The proposed pruning approach is applicable to other search based NA methods as well. Additionally, we reframe the NA in terms of classification/prediction in order to significantly expand the set of objective functions (an important research focus in NA) with well-established performance metrics used in statistical analysis. We suggest that research in those two directions will provide great improvement in the ability to align networks, and make NA even more widely applicable in even more domains than it already is.

Acknowledgement

This research was funded by the Israeli Ministry of Defense as a part of the Social Network Digestion project and by the Israeli Science Foundation (ISF) grant no. 210/17 to Roni Stern.

References


Guzzi, P. H. & Milenkovic, T., 2017. Survey of local and global biological network alignment: the need to reconcile the two sides of the same coin. *Briefings in bioinformatics*.


Ibragimov, R., Malek, M., Guo, J. & Baumbach, J., 2013. GEDEVO: An Evolutionary Graph Edit Distance Algorithm for Biological Network Alignment. s.l., German Conference on Bioinformatics.


Mamano, N. & Hayes, W. B., 2017. SANA: simulated annealing far outperforms many other search algorithms for biological network alignment. *Bioinformatics*.


