applied sciences

MDPI

Article

A Novel Algorithm for Local Network Alignment Based on Network Embedding

Pietro Hiram Guzzi [1,*,†], Giuseppe Tradigo [2,†] and Pierangelo Veltri [1]

[1] Department of Surgical and Medical Science, University Magna Graecia of Catanzaro, 88100 Catanzaro, Italy; veltri@unicz.it
[2] University eCampus Novedrate (CO), 22060 Novedrate, Italy; giuseppe.tradigo@uniecampus.it
* Correspondence: hguzzi@unicz.it; Tel.: +39-096-1369-4149
† These authors contributed equally to this work.

Abstract: Networks are widely used in bioinformatics and biomedicine to represent associations across a large class of biological entities. Network alignment refers to the set of approaches that aim to reveal similarities among networks. Local Network Alignment (LNA) algorithms find (relatively small) local regions of similarity between two or more networks. Such algorithms are in general based on a set of seed nodes that are used to build the alignment incrementally. A large fraction of LNA algorithms uses a set of vertices based on context information as seed nodes, even if this may cause a bias or a data-circularity problem. Moreover, using topology information to choose seed nodes improves overall alignment. Finally, similarities among nodes can be identified by network embedding methods (or representation learning). Given there are two networks, we propose to use network embedding to capture structural similarity among nodes, which can also be used to improve LNA effectiveness. We present an algorithm and experimental tests on real and syntactic graph data to find LNAs.

Keywords: LNA; Local Network Alignment; topology; biological entities

check for updates

Citation: Guzzi, P.H.; Tradigo, G.; Veltri, P. A Novel Algorithm for Local Network Alignment Based on Network Embedding. *Appl. Sci.* **2022**, *12*, 5403. https://doi.org/10.3390/app12115403

Academic Editor: Giacomo Fiumara

Received: 20 April 2022
Accepted: 25 May 2022
Published: 26 May 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Nowadays, networks are used to model and study several applications in many real scenarios. In molecular biology, networks are used to model associations among genes or proteins in a unified formalism known as Protein Interaction Networks (PINs) [1]. Networks are also used to model association among drugs and diseases, to model protein structures [2–9], or to study multilayer applications [10]. In social sciences, network-based models are used to represent relations among users of social networks [11,12]. It has been demonstrated that valuable information can be obtained by analyzing single networks as well as by comparing two or more networks. Comparison between networks is also associated with Network Alignment (NA) algorithms [1,13]. NA algorithms fall into two major classes: local (LNA, for Local Network Alignment) and global (GNA, for Global Network Alignment). LNA algorithms usually find (relatively small) regions of similarity between two or more networks and are also used to compare small regions extracted from two or more input networks. GNA algorithms aim to find a global mapping among nodes, discarding local regional similarity. It has been demonstrated that GNA algorithms are the best choice to transfer knowledge from a well-studied network to other networks [14]. LNA algorithms usually perform well in finding corresponding substructures representing, for instance, protein complexes [2,13,15–18]. In brain sciences, NA has been used to reconstruct connectomes from images without the use of any atlas [19,20]. In this paper, we focus on LNA algorithms to study biological networks [1]. We also focus on embedding techniques to improve LNA. LNA uses as input two (or more) networks and a set of initial mapped nodes (known as *seed nodes*) to build the alignment. Such information is usually derived from biological considerations, which, in principle, may cause a circularity problem or bias [16,19]. Consequently, many different works have tried to improve existing LNA

algorithms by using topological information as input [21]. The rationale behind these approaches is that topological information can quantify the structural similarity among nodes. Similarly, network embedding methods aim to encode structural information about networks. The latter methods are usually referred to as *graph embedding* or *graph representation learning* [22]. The goal of each algorithm is to map each node of a given undirected and unweighted graph $G = (V, E)$ into points of a low-dimensional vector space, i.e., into a vector $z \in \mathbb{R}^d$ where $d < |V|$, with $d < n$, and n being the dimension of the graph adjacency matrix [22].

We are interested in node-based embeddings; thus, all these methods calculate a mapping among nodes and points of the embedding space so that geometric relationships among embedded objects reflect the structure of the original graph. For instance, Figure 1 represents an example of LNA and embedding.

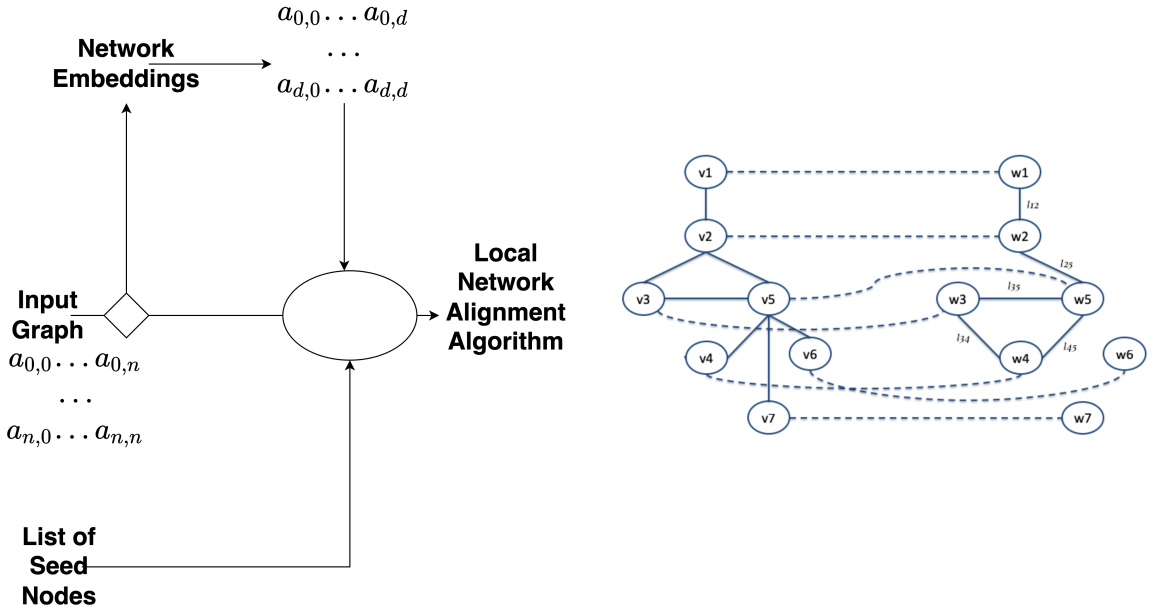


Figure 1. Figure depicts the proposed approach. The input graphs are converted into a low-dimensional representation through network embeddings, which are then used to determine the similarity among the nodes of the input graphs and used as potential seed nodes. Finally, these nodes are integrated with the initial list of seed nodes to build the LNA.

Let $G(V, E)$ be a graph with $\|E\|$ edges and $\|V\|$ nodes. The desired behavior of an embedding algorithm is that, given two nodes $v_i \in V$ and $v_j \in V$ the similarity between the embeddings of v_i and v_j should reflect the similarity between the original nodes, i.e., minimizing loss l, a measure to quantify the noise introduced by mapping nodes into a new space (e.g., two adjacent nodes are not necessarily mapped into close points in the new space, called latent (or embedding) space). More formally, given two nodes v_i and v_j in V, let emb_{v_i} and emb_{v_j} be their embeddings and let l be the loss value between the similarity among nodes and their embeddings. Then, the embedding algorithm should be geared in such a way that its parameters will minimize the overall loss function $\mathcal{L} = \sum_{v_i, v_j \in V} l(v_i, v_j), (emb_{v_i}, emb_{v_j})$. To reach such an optimal embedding, methods have to capture the structural similarity by analyzing nodes in the latent space of the embedding. We focus on network embeddings methods to compare nodes belonging to different networks. It has been demonstrated that using such methods can be more efficient than adopting other classical ones [23,24].

From these considerations, we conceived the idea of using network embeddings to build an initial set of seed nodes starting from the similarity of such nodes in the embedding space. We then developed a novel algorithm for network alignment based on the embedding of the input networks to find an initial set of pairs of similar nodes. This

set is then merged with the set of similar nodes obtained by contextual or domain-specific information (e.g., biologically similar nodes) and the resulting set is then used to build the alignment. Figure 2 depicts the workflow of the proposed algorithm. The described approach can be extended to any LNA algorithm receiving a set of seed nodes as input.

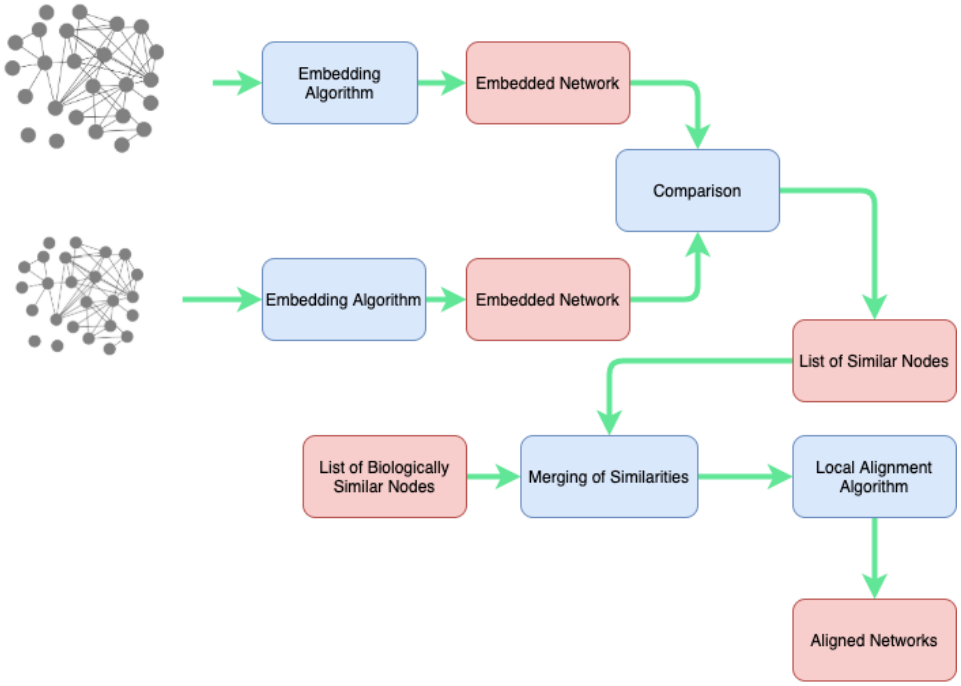


Figure 2. Workflow of the proposed approach.

The rest of the paper is structured as follows: Section 2 discusses the main related works. Section 3 describes the proposed approach. The results are reported in Section 4, where we discuss case studies. Finally, Section 5 concludes the paper and outlines future works.

2. Related Work

The study, definition and development of new methods for encoding structural information in order to improve graph analysis is a novel topic recently attracting researchers. Currently, many algorithms and many classification attempts have been described in surveys [11,22,25–30]. These methods are usually referred to as *graph representation learning* or *graph embedding* [22]. Here, we focus on node-based embeddings, i.e., on methods that build mappings between graph nodes and points in the embedding space so that geometric relationships among embedded objects reflect the structure of the original graph. As for the relevant literature, we follow the classification proposed by Hamilton et al. [22], which presents an overview and a discussion of some related methods, in particular the preprocessing methods being used by the approach presented here to improve network embeddings and alignments. All of the embedding algorithms may be represented by using two functions: an encoder ($Enc: V \rightarrow R^d$), which maps each node to the embedding, and a decoder Dec, which decodes structural information about the graph from the embeddings. It is possible to define a similarity among nodes of the G graph, and a loss function that measures the difference (i.e., the loss) of similarity of two original nodes and their embeddings $l(v_i, v_j), (emb_{v_i}, emb_{v_j})$. The simplest similarity function is the adjacency matrix, according to which similarity equals 1 for adjacent nodes and 0 otherwise. A common similarity function among node embeddings is the dot product (i.e., the cosine angle) between the embedding vectors. Embedding algorithms are usually based on the minimization of the loss function l in the embedding space (see Section 1). Embedding approaches can also be also based on *shallow embedding*, which consists of encoding each node of the graph into a

single vector through the use of a simple encoding function. The main objective is always to map nodes into embedding space. Methods differ in the use of different loss functions and similarity measures. Some representative methods are GraRep [31] and Hope [32]. One main drawback of these methods is that the similarity measure considered only takes into account the presence of edges of nodes, since they only consider nodes in the first-order neighborhood. Conversely, two nodes can be structurally similar even when they are far from each other in the network. Moreover, the encoding must be entirely recalculated when new nodes or edges are deleted or inserted. Consequently, a set of methods to investigate higher-order neighborhoods has been introduced.

The higher-order neighborhood of each node is usually explored through Random Walks, which are short random paths that consist of a succession of random steps over the network. The key idea is to derive the similarity between two nodes on the basis of the co-occurrence of nodes in the respective random walks, by observing that two similar nodes have a greater chance to have similar random walks. As reported in [22], the literature reports many strategies that differ on the strategy used to calculate similarity, e.g., the way to simulate random walks. Thus, each method depends on the adopted random walk strategy. e.g., **DeepWalk** employs an unbiased random walk of fixed-length starting from each node [33]. **Node2Vec**, instead, uses random walks of different lengths, defined as biased random walks [34]. **struct2vec** employs biased random walks generated and propagated in a modified version of the network, in which nodes are structurally similar to the original network and close to the starting node. **LINE** [35] combines first- and second-order proximity to take into account both the local and the global network structures. First-order proximity is evaluated by considering the pairwise proximity between pair of nodes, while second-order proximity is evaluated on the basis of the neighbors of the nodes.

A common characteristic of all the previous methods is that they learn a unique embedding vector for each node. The simplicity of this approach, however, has two main drawbacks: (i) the impossibility to consider any other metadata associated with the nodes; (ii) the need to learn the whole representation again in the event of any modification in the graph (i.e., deletion or insertion of edges or nodes). This is highly problematic for data-rich graphs, such as social or biological networks, and for evolving graphs. Recently, to address these issues, a number of approaches have been proposed which use more complex encoders able to gather information from nodes and to generate representations of nodes without the use of the whole graph each time. Some use Deep Neural Networks as encoders. Deep Neural Graph Representations (DNGR) [36] and Structural Deep Network Embeddings (SDNE) [37] are two examples of such methods. Finally, other methods, such as GraphSAGE [22], use Graph Convolutional Networks to learn the mappings.

In our proposal, we use DeepWalk, node2vec and LINE to build the initial list of seed nodes and to improve network alignment, and we implement the strategy in an available tool, tested to prove its effectiveness in building alignments by using embeddings.

3. Workflow of the Proposed Algorithm

The proposed method can be described by using the workflow reported in Figure 2. We start from two input networks which should be aligned. First, we map both networks into the embedding space using the same algorithm to embed both networks. Then, we compare the representations of all the nodes using cosine similarity in the embedding space; in this way, we can compare pairs of nodes in the embedding space, searching for those having comparable similarity values. Then, the ranked list of pairs is merged with the list of biologically similar nodes. The merging is performed by means of linear combination of the weights associated with each couple of identical pairs. The resulting list is given as input to the local aligner.

After evaluating the similarity among nodes, we use as input a list of seed nodes derived from evaluating embeddings, and the list of seed nodes derived from biological (or contextual) information.The weights of node pairs contained in M_{emb} are merged with those contained in M_w following a linear combination schema, which produces a final list of mapped nodes, namely M_{final}.

We then build the local alignment using AlignMCL, an algorithm reported in [21]. As input, AlignMCL receives two networks and an initial weighted list of seed nodes, and it develops a local alignment by merging all of the input data into a single graph, called alignment graph. The Markov Clustering Algorithm (MCL) [38] is used to extract communities. During the building of the alignment graph, AlignMCL scores the links between node pairs in the alignment graph.

4. Embedding Scenarios

In this section, we report a number of observations regarding the application of the proposed methodology to real cases. The methods should consider that the embeddings may introduce noise and are not guaranteed to preserve network properties. In the first case study, reported as *case study 1* below, we use synthetic networks, and point out that embeddings preserve the similarity of nodes. We demonstrate this empirically in our experiments, since in general, embeddings do not preserve inter-network similarities. In the second case study, reported as *case study 2*, we use embeddings on biological (i.e., real) networks.

4.1. Case Study 1: Evaluation of the Similarity of Embeddings

In this case study, we show that the embeddings do not modify the similarity among nodes. Using publicly available network generators in the NetworkX package [39], we build synthetic networks with scale free topology. We build two initial networks, $Rand_1$ and $Rand_2$, having 6000 edges, respectively. Then, we obtain a noisy counterpart for each network by rewiring 5%, 10%, 20%, 25% and 50% of the edges, respectively. Next, we build five rewired network instances at each noise level obtaining a total of ten networks.

We consider the first network and its randomized versions 256 (R1-0, R1-5, R1-10, R1-15, R1-20, R1-25, R1-50) and the second network and its randomized 257 versions (R2-0, R2-5, R2-10, R2-15, R2-20, R2-25, R2-50). For each network, we determine 258 embeddings using the DeepWalk, node2Vec and LINE algorithms. Let GDV [14] be the signature of each node of a network, calculated on the basis of the orbits of the graphlet. We measure the similarity across all of the node pairs v_{1i}, w_{2j} of the two networks (e.g., network 1 and network 2) by using the cosine distance among the Graphlet Degree Vectors (GDVs):

$$\langle GDV(v_{1i}), GDV(w_{2j}) \rangle \quad v_{1i} \in V_1, w_{2j} \in V_2 \tag{1}$$

where the dot product between $GDV(v_{1i})$ and $GDV(w_{2j})$ is proportional to the $cos(\theta)$, with θ being the angle between the two vectors.

Then, we use the cosine similarity among all the embeddings $cos(f(v_{1i}), f(v_{2j})$ as a similarity measure, in line with previous works (e.g., see [23]). We obtain two vectors representing the ranked similarities among node pairs measured in the different spaces. In order to compare these rankings, the Rank-Biased Overlap (RBO) [40] is used. The RBO measure, developed in the information retrieval to calculate a score for comparing indefinite rankings, is used to handle tied ranks and rankings of different lengths. It provides a monotonic score in the interval $[0, 1]$. We calculate the RBO score for each pair of randomized networks by using Algorithm 1 reported below. Given two networks, G_1 and G_2, and two embeddings, $F(G_1)$ and $F(G_2)$, the RBO Score calculates the overall agreements among the similarity of pairs of nodes and the corresponding pairs of embeddings. RBO takes values in the range (0, 1), where 0 means disjoint and 1 means identical ranked lists. Thus, in our case, a higher value means that there is agreement between rankings; hence, the embedding preserves inter-network similarities.

Algorithm 1 Comparing Node Similarities as Rankings

Require: $G_1 = (V_1, E_1)$, and $G_2 = (V_2, E_2)$
Require: $F(V_1), F(V_2)$.
Require: GDV {Any other similarity measure providing a monotonic score of similarity}
Ensure: $Score_{RBO}$ {Score of the Ranked Biased Overlap}
  for all $v, w \in V_1, V_2$ **do**
    $Sim(v, w) = GDV(v, w)$
  end for
  for all $f(v), f(w) \in F(V_1), F(V_2)$ **do**
    $SimEmb(f(v), f(w)) = cosine(f(v), f(w))$
  end for
  $Rank1 = (Sim)$
  $Rank2 = (SimEmb)$ {Let sort both the results on the basis of similarity score and make a ranking}
  $Score_{RBO} = RBO(Rank1, Rank2)$ {Calculate the RBO between rankings}

Tables 1–3 summarize the results. We consider the first network and its randomized versions (R1-0, R1-5, R1-10, R1-15, R1-20, R1-25, R1-50) and the second network and its randomized versions (R2-0, R2-5, R2-10, R2-15, R2-20, R2-25, R2-50). For each network, we determine the embeddings using the algorithms (DeepWalk, node2Vec and LINE). We then measure the similarity among all node pairs and all embedding pairs. Finally, we measure the agreement of rankings using RBO values.

Table 1. Comparison of RBO Scores considering Synthetic Randomized Networks and the DeepWalk algorithm.

Networks	R2-0	R2-5	R2-10	R2-15	R2-20	R2-25	R2-50
R1-0	0.98	0.96	0.93	0.94	0.97	0.95	0.97
R1-5	0.98	0.96	0.96	0.96	0.95	0.96	0.98
R1-10	0.98	0.96	0.96	0.93	0.96	0.95	0.96
R1-15	0.96	0.92	0.93	0.92	0.96	0.93	0.93
R1-20	0.95	0.96	0.93	0.96	0.96	0.96	0.96
R1-25	0.91	0.93	0.93	0.93	0.93	0.93	0.91
R1-50	0.92	0.91	0.91	0.91	0.91	0.91	0.91

Table 2. Comparison of RBO Scores considering Synthetic Randomized Networks and the Node2Vec algorithm.

	R2-0	R2-5	R2-10	R2-15	R2-20	R2-25	R2-50
R1-0	0.97	0.93	0.91	0.88	0.82	0.93	0.80
R1-5	0.98	0.93	0.93	0.95	0.92	0.76	0.91
R1-10	0.97	0.96	0.79	0.80	0.84	0.77	0.92
R1-15	0.95	0.81	0.89	0.83	0.74	0.90	0.89
R1-20	0.95	0.79	0.93	0.82	0.86	0.81	0.87
R1-25	0.90	0.91	0.83	0.73	0.92	0.90	0.73
R1-50	0.92	0.75	0.77	0.73	0.85	0.80	0.72

Table 3. Comparison of RBO Scores considering Synthetic Randomized Networks and the Line algorithm.

	R2-0	**R2-5**	**R2-10**	**R2-15**	**R2-20**	**R2-25**	**R2-50**
R1-0	0.97	0.94	0.80	0.91	0.86	0.80	0.88
R1-5	0.97	0.88	0.89	0.89	0.85	0.85	0.81
R1-10	0.98	0.92	0.83	0.81	0.84	0.89	0.79
R1-15	0.95	0.85	0.86	0.84	0.87	0.81	0.77
R1-20	0.95	0.93	0.89	0.78	0.79	0.82	0.80
R1-25	0.91	0.91	0.85	0.86	0.80	0.85	0.82
R1-50	0.92	0.82	0.83	0.88	0.81	0.72	0.71

4.2. Case Study 2: Alignment of the Protein Interaction Network

As a second scenario, we consider real biological networks. We downloaded the Protein Network available at [41] provided by Zitnik et al. [42]. The protein–protein association network includes both protein–protein interactions, and functional associations. Nodes in the network represent proteins, while edges represent their associations. The network has 19,081 nodes and 715,612 edges. We preprocess the network considering only nodes and edges in the largest strongest connected component, obtaining a subnetwork of 19,065 nodes and 715,602 edges.

We randomize this network (data and code are available at the web site of the project (https://github.com/hguzzi/EMB-Align (accessed on 20 April 2022)), obtaining five networks with 5%, 10%, 15%, 25% and 50% of noise, respectively. For each network, we apply LINE and DeepWalk embeddings. Then, we calculate the alignments using Align-MCL.

We align the original network with respect to the noised versions. Each alignment returns a list of aligned subgraph. We measure the quality of the obtained alignment in terms of the semantic similarity within each aligned subgraphs. We compare the overall average semantic similarity over all the obtained subgraphs with respect to the classical Align-MCL and calculate the improvement. Table 4 summarizes these results.

Table 4. Comparison of the average semantic similarity obtained by EMB-Align w.r.t. Align-MCL considering the Decagon-PPI Dataset (DEC) vs. its randomized versions.

	DEC vs. DEC-5	**DEC vs. DEC-10**	**DEC vs. DEC-15**	**DEC vs. DEC-20**	**DEC vs. DEC-25**	**DEC vs. DEC-50**
EMB-Align	0.97	0.90	0.80	0.83	0.75	0.66
Align-MCL	0.92	0.88	0.75	0.71	0.64	0.54

5. Conclusions

It has been demonstrated that the use of network embedding methods (or representation learning) may capture the structural similarities among nodes better than other methods. Therefore, we proposed to use network embeddings to learn structural similarities among nodes and used such similarities to improve LNA extendings our previous algorithms. Finally, we defined a framework for LNA. Our approach is applicable to all network models. Future works will focus on two main topics: (i) the use of Graph Neural Networks as embeddings to gather information associated with nodes; (ii) extending the approach to multiple network alignment.

Author Contributions: Conceptualization, P.H.G., G.T. and P.V.; writing—original draft preparation, P.H.G., G.T. and P.V.; project administration, P.H.G.; funding acquisition, P.V. All authors have read and agreed to the published version of the manuscript.

Funding: The APC was funded by PON-VQA Mise.

Data Availability Statement: Data are available at https://github.com/hguzzi/EMB-Align (accessed on 20 April 2022).

Acknowledgments: Authors thank Ugo Lo Moio for performing some experiments.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

PIN	Protein Interaction Network
LNA	Local Network Alignment
GNA	Global Network Alignment
RBO	Rank-Biased Overlap Score
GDV	Graphlet Degree Vector
MCL	Markov Clustering Algorithm

References

1. Guzzi, P.H.; Milenković, T. Survey of local and global biological network alignment: The need to reconcile the two sides of the same coin. *Briefings Bioinform.* **2018**, *19*, 472–481. [CrossRef] [PubMed]
2. Agapito, G.; Cannataro, M.; Guzzi, P.H.; Marozzo, F.; Talia, D.; Trunfio, P. Cloud4SNP: Distributed analysis of SNP microarray data on the cloud. In Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics, Washington, DC, USA, 22–25 September 2013; pp. 468–475.
3. Bargmann, C.I.; Marder, E. From the connectome to brain function. *Nat. Methods* **2013**, *10*, 483–490. [CrossRef] [PubMed]
4. Ortuso, F.; Mercatelli, D.; Guzzi, P.H.; Giorgi, F.M. Structural genetics of circulating variants affecting the SARS-CoV-2 spike/human ACE2 complex. *J. Biomol. Struct. Dyn.* **2021**, 1–11. [CrossRef]
5. Guzzi, P.H.; Tradigo, G.; Veltri, P. Using dual-network-analyser for communities detecting in dual networks. *BMC Bioinform.* **2021**, *22*, 1–16. [CrossRef]
6. Cristiano, F.; Veltri, P. Methods and techniques for miRNA data analysis. *Methods Mol. Biol.* **2016**, *1375*, 11–23.
7. Cannataro, M.; Guzzi, P.H.; Mazza, T.; Tradigo, G.; Veltri, P. Using ontologies for preprocessing and mining spectra data on the Grid. *Future Gener. Comput. Syst.* **2007**, *23*, 55–60. [CrossRef]
8. Guzzi, P.H.; Di Martino, M.T.; Tradigo, G.; Veltri, P.; Tassone, P.; Tagliaferri, P.; Cannataro, M. Automatic summarisation and annotation of microarray data. *Soft Comput.* **2011**, *15*, 1505–1512. [CrossRef]
9. Tradigo, G.; De Rosa, S.; Vizza, P.; Fragomeni, G.; Guzzi, P.H.; Indolfi, C.; Veltri, P. Calculation of Intracoronary Pressure-Based Indexes with JLabChart. *Appl. Sci.* **2022**, *12*, 3448. [CrossRef]
10. Ren, Y.; Sarkar, A.; Veltri, P.; Ay, A.; Dobra, A.; Kahveci, T. Pattern discovery in multilayer networks. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2022**, *19*, 741–752. [CrossRef]
11. Cannataro, M.; Guzzi, P.H.; Sarica, A. *Data Mining and Life Sciences Applications on the Grid. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*; Wiley: Hoboken, NJ, USA, 2013; Volume 3, pp. 216–238.
12. Tradigo, G.; Vizza, P.; Fragomeni, G.; Veltri, P. On the reliability of measurements for a stent positioning simulation system. *Int. J. Med. Inform.* **2019**, *123*, 23–28. [CrossRef]
13. Cho, Y.R.; Mina, M.; Lu, Y.; Kwon, N.; Guzzi, P.H. M-finder: Uncovering functionally associated proteins from interactome data integrated with go annotations. *Proteome Sci.* **2013**, *11*, 1–12. [CrossRef] [PubMed]
14. Milenković, T.; Leong, W.; Pržulj, N. Optimal network Alignment with Graphlet Degree Vectors. *Cancer Inf.* **2010**, *9*, 121–137. [CrossRef] [PubMed]
15. Nassa, G.; Tarallo, R.; Guzzi, P.H.; Ferraro, L.; Cirillo, F.; Ravo, M.; Nola, E.; Baumann, M.; Nyman, T.A.; Cannataro, M.; et al. Comparative analysis of nuclear estrogen receptor alpha and beta interactomes in breast cancer cells. *Mol. Biosyst.* **2011**, *7*, 667–676. [CrossRef] [PubMed]
16. Mina, M.; Guzzi, P.H. Improving the robustness of local network alignment: Design and extensive assessment of a Markov Clustering-based approach. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2014**, *11*, 561–572. [CrossRef] [PubMed]
17. Grillone, K.; Riillo, C.; Scionti, F.; Rocca, R.; Tradigo, G.; Guzzi, P.H.; Alcaro, S.; Di Martino, M.T.; Tagliaferri, P.; Tassone, P. Non-coding RNAs in cancer: Platforms and strategies for investigating the genomic "dark matter". *J. Exp. Clin. Cancer Res.* **2020**, *39*, 1–19. [CrossRef] [PubMed]
18. Guzzi, P.H.; Agapito, G.; Cannataro, M. coresnp: Parallel processing of microarray data. *IEEE Trans. Comput.* **2013**, *63*, 2961–2974. [CrossRef]
19. Milano, M.; Milenković, T.; Cannataro, M.; Guzzi, P.H. L-HetNetAligner: A novel algorithm for local alignment of heterogeneous biological networks. *Sci. Rep.* **2020**, *10*, 3901. [CrossRef]
20. Milano, M.; Guzzi, P.H.; Tymofieva, O.; Xu, D.; Hess, C.; Veltri, P.; Cannataro, M. An extensive assessment of network alignment algorithms for comparison of brain connectomes. *Bmc Bioinform.* **2017**, *18*, 31–45. [CrossRef]
21. Milano, M.; Guzzi, P.H.; Cannataro, M. Glalign: A novel algorithm for local network alignment. *IEEE/Acm Trans. Comput. Biol. Bioinform.* **2018**, *16*, 1958–1969. [CrossRef]
22. Hamilton, W.L.; Ying, R.; Leskovec, J. Representation learning on graphs: Methods and applications. *arXiv* **2017**, arXiv:1709.05584.

23. Gu, S.; Jiang, M.; Guzzi, P.H.; Milenković, T. Modeling multi-scale data via a network of networks. *Bioinformatics* **2022**, *38*, 2544–2553. [CrossRef] [PubMed]
24. Kukic, P.; Mirabello, C.; Tradigo, G.; Walsh, I.; Veltri, P.; Pollastri, G. Toward an accurate prediction of inter-residue distances in proteins using 2D recursive neural networks. *BMC Bioinform.* **2014**, *15*, 6. [CrossRef]
25. Cui, P.; Wang, X.; Pei, J.; Zhu, W. A survey on network embedding. *IEEE Trans. Knowl. Data Eng.* **2018**, *31*, 833–852. [CrossRef]
26. Su, C.; Tong, J.; Zhu, Y.; Cui, P.; Wang, F. Network embedding in biomedical data science. *Briefings Bioinform.* **2020**, *21*, 182–197. [CrossRef] [PubMed]
27. Nelson, W.; Zitnik, M.; Wang, B.; Leskovec, J.; Goldenberg, A.; Sharan, R. To embed or not: Network embedding as a paradigm in computational biology. *Front. Genet.* **2019**, *10*, 381. [CrossRef]
28. Goyal, P.; Ferrara, E. Graph embedding techniques, applications, and performance: A survey. *Knowl. Based Syst.* **2018**, *151*, 78–94. [CrossRef]
29. Guzzi, P.H.; Cannataro, M. μ-CS: An extension of the TM4 platform to manage Affymetrix binary data. *BMC Bioinform.* **2010**, *11*, 315. [CrossRef]
30. Mirarchi, D.; Petrolo, C.; Canino, G.; Vizza, P.; Cuomo, S.; Chiarella, G.; Veltri, P. Applying mining techniques to analyze vestibular data. *Procedia Comput. Sci.* **2016**, *98*, 467–472. [CrossRef]
31. Cao, S.; Lu, W.; Xu, Q. Grarep: Learning graph representations with global structural information. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, Melbourne, VIC, Australia, 19–23 October 2015; pp. 891–900.
32. Ou, M.; Cui, P.; Pei, J.; Zhang, Z.; Zhu, W. Asymmetric transitivity preserving graph embedding. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 1105–1114.
33. Perozzi, B.; Al-Rfou, R.; Skiena, S. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 701–710.
34. Grover, A.; Leskovec, J. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864.
35. Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; Mei, Q. Line: Large-scale information network embedding. In Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015; pp. 1067–1077.
36. Cao, S.; Lu, W.; Xu, Q. Deep neural networks for learning graph representations. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
37. Wang, D.; Cui, P.; Zhu, W. Structural deep network embedding. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 1225–1234.
38. van Dongen, S. Graph Clustering by Flow Simulation. Ph.D. Thesis, University of Utrecht, Utrecht, The Netherlands, 2000.
39. NetworkX.org. NetworkX Libary for Network Analysis in Python. 2020. Available online: https://networkx.org/ (accessed on 20 April 2022).
40. Webber, W.; Moffat, A.; Zobel, J. A similarity measure for indefinite rankings. *Acm Trans. Inf. Syst.* **2010**, *28*, 1–38. [CrossRef]
41. Leskovec, J.; Krevl, A. SNAP Datasets: Stanford Large Network Dataset Collection. 2014. Available online: http://snap.stanford.edu/data (accessed on 20 April 2022).
42. Zitnik, M.; Agrawal, M.; Leskovec, J. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics* **2018**, *34*, i457–i466. [CrossRef]