Applied Soft Computing 132 (2023) 109832

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

AntNetAlign: Ant Colony Optimization for Network Alignment

Guillem Rodríguez Corominas [a], Maria J. Blesa [b], Christian Blum [a,*]

[a] Artificial Intelligence Research Institute (IIIA-CSIC), Bellaterra, Spain
[b] Universitat Politècnica de Catalunya (UPC - BarcelonaTech), Barcelona, Catalonia

ARTICLE INFO

Article history:
Received 20 December 2021
Received in revised form 13 October 2022
Accepted 10 November 2022
Available online 15 November 2022

Keywords:
Network alignment
Ant colony optimization
Combinatorial optimization
Graph theory

ABSTRACT

Network Alignment (NA) is a hard optimization problem with important applications such as, for example, the identification of orthologous relationships between different proteins and of phylogenetic relationships between species. Given two (or more) networks, the goal is to find an alignment between them, that is, a mapping between their respective nodes such that the topological and functional structure is well preserved. Although the problem has received great interest in recent years, there is still a need to unify the different trends that have emerged from diverse research areas. In this paper, we introduce AntNetAlign, an Ant Colony Optimization (ACO) approach for solving the problem. The proposed approach makes use of similarity information extracted from the input networks to guide the construction process. Combined with an improvement measure that depends on the current construction state, it is able to optimize any of the three main topological quality measures. We provide an extensive experimental evaluation using real-world instances that range from Protein–Protein Interaction (PPI) networks to Social Networks. Results show that our method outperforms other state-of-the-art approaches in two out of three of the tested scores within a reasonable amount of time, specially in the important S^3 score. Moreover, it is able to obtain near-optimal results when aligning networks with themselves. Furthermore, in larger instances, our algorithm was still able to compete with the best performing method in this regard.

1. Introduction

The Network Alignment problem requires to find a mapping between the nodes of two (or more) given networks such that a certain, application-dependent, quality measure is optimized. Ultimately, the goal of aligning two networks is to transfer knowledge from one network to the other, and one of the main areas of application is Biology. In biological contexts, for example, this allows to transfer knowledge about well-studied organisms to lesser known ones. Moreover, aligning two networks can be used to identify orthologous relationships between different proteins or phylogenetic relationships between species [1].

Network Alignment also plays a crucial role in various fields and applications concerned with social networks [2]. For example, in the context of users that have accounts in different social networks, it is natural to assume that a user behaves similarly in all networks. One example is what is commonly referred to as *link prediction* [3,4]: if two users are friends in a social network, it is probable that they are also friends in other social networks. Another example is *cross-network recommendation* [5], where holistic user profiles are generated based on their behavior in a source network and then used to suggest interests to the users in the target network.

Due to the different contexts of application, there exist many algorithms to tackle the Network Alignment problem. As a consequence, many of these algorithms are specifically designed for a certain application area. Not surprisingly, their performance might decrease when applied to networks from other application areas. In this work, we introduce an ACO algorithm that aims to unify the different algorithmic trends.

In the remainder of this section, we introduce some basic concepts that will help to define the tackled problem. In Section 2, we give a general view of the state of the art. In Section 3, we present our approach. In Section 4, the experimental results are presented. Finally, in Section 5 we conclude our work and present some ideas for future work.

1.1. Terms and concepts

An undirected graph (or network) G is denoted as $G = (V, E)$, where V is the non-empty set of *vertices* (or *nodes*) and E is the

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail address: christian.blum@csic.es (C. Blum).

https://doi.org/10.1016/j.asoc.2022.109832

set of undirected *edges*. An edge $e = (u, v) \in E$ connects two vertices $u, v \in V$. It is said that $e = (u, v)$ is *incident* to u and v, and that these two vertices are the *endpoints* of e. Moreover, two edges are said to be *adjacent* when are incident to a common vertex. Similarly, two vertices are called *adjacent* if there is an edge connecting them. We denote the set of vertices of a graph G as $V(G)$, and the set of edges as $E(G)$. The *order* of a graph G refers to the number of vertices, while the *size* of a graph refers to the total number of edges (i.e., $|V(G)|$ and $|E(G)|$, respectively). Additionally, let $G[V']$ be the induced subgraph of G with node set $V' \subseteq V$, i.e., the graph $G[V'] = (V', E')$ where $E' = \{(u, v) \mid u, v \in V', (u, v) \in E\}$.

Networks are assumed not to have loops (i.e., there are no edges of the form (u, u)) and then we can distinguish different neighborhoods. The *open neighborhood* of a vertex v refers to the set of vertices adjacent to v i.e. $N(v) := \{u \in V \mid (v, u) \in E\}$. Similarly, we refer to the *closed neighborhood* as $N[v] := N(v) \cup \{v\}$. Hence, given two vertices $u, v \in V$, u is called a neighbor of v if and only if $u \in N(v)$. Furthermore, the *degree* of a vertex refers to the number of edges incident to it, i.e., $deg(v) = |N(v)|$. Additionally, we define the *D-restricted neighborhood* of a vertex $v \in V$ as all the vertices in set D that are in the open neighborhood of v, i.e., $N_D(v) := N(v) \cap D$.

One way to measure the importance of a vertex of a given network is to calculate its *centrality*. Given that the meaning of the importance of a vertex might vary from context to context, there exist different definitions of centrality. In this paper, when we mention the centrality of a vertex, we refer to the *betweenness centrality*, i.e., the number of shortest paths between pairs of vertices that pass through the indicated vertex.

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where $|V_1| \leq |V_2|$, an *alignment* is an injective function $f : V_1 \mapsto V_2$, i.e., a mapping between their respective nodes. G_1 and G_2 are usually referred to as the *source* and *target* networks, respectively. Note that there is no loss of generality by assuming that the source network has smaller order than the target network. Two nodes $v \in V_1$ and $v' \in V_2$ are said to be *aligned* if and only if $f(v) = v'$.

Given an alignment f between two networks G_1 and G_2, an edge in the target network is said to be *induced* by the alignment if, and only if, there exist two (not necessarily connected) nodes from the source network that are mapped to the endpoints of the induced edge. More formally, an edge $e' = (u', v') \in E_2$ is said to be induced if and only if $\exists u, v \in V_1$ such that $f(u) = u'$ and $f(v) = v'$. The aforementioned definition can be simplified by saying that an edge is induced if and only if both of its endpoints are aligned with a node from the other network. By an abuse of notation, we can express the set of induced edges from G_2 given a mapping over the subset of nodes $V' \subseteq V_1$ as $E(G_2[f(V')])$. Induced edges are usually only considered on the target network, as, given a complete mapping, all the edges on the source network are induced. However, as our algorithm will deal with partially-defined mappings, we extend the previous definitions to the source network. Lastly, an edge $(u, v) \in E_1$ is said to be *conserved* if and only if $(f(u), f(v)) \in E_2$.

1.2. Problem formulation

The *Network Alignment* (NA) problem asks to find an alignment between two networks such that the topological and functional structure is well preserved. Given the wide range of areas and contexts of applications, there exist many different measures to quantify the quality of an alignment. Some of them are limited to the functional aspect, while others focus solely on the topological or structural facet. Functional scoring methods are usually more context-dependent, taking into account the application of the alignment. For instance, one can consider the significance of the biological function when aligning Protein–Protein Interaction (PPI) networks. However, in other areas, functional information may refer to the role of the aligned nodes inside the networks. On the other hand, structural quality measures are usually based on the inherent topological similarities of the aligned networks.

Given two networks $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, a given alignment f between G_1 and G_2 may be evaluated by the following structural quality measures:

- *Conserved Interactions Under Alignment* (CIUA): Number of edges (interactions) conserved under the given alignment, i.e.,

  $$\mathsf{CIUA} = |f(E_1) \cap E_2|$$

  where $f(E_1) = \{(f(u), f(v)) : (u, v) \in E_1\}$.
- *Edge Correctness* (EC)[1] [1]. Measures the percentage of edges from the first network that are mapped to edges from the second network:

  $$\mathsf{EC} = \frac{|f(E_1) \cap E_2|}{|E_1|} \times 100$$

  As EC is defined only with respect to the source network, it fails to penalize alignments that map sparse regions of the source network to denser ones.
- *Induced Conserved Structure* (ICS) [7]: Ratio of edges conserved by the alignment with respect to the number of edges in the induced subnetwork of G_2 formed by the mapped nodes from G_1:

  $$\mathsf{ICS} = \frac{|f(E_1) \cap E_2|}{|E(G_2[f(V_1)])|} \times 100$$

  Since ICS is defined only with respect to the target network, conversely to EC, this measure unsuccessfully penalizes alignments mapping denser network regions to sparser ones.
- *Symmetric Substructure Score* (S^3) [8]: In this case, we consider that the source graph (G_1) and the induced subnetwork of the target graph under the alignment ($G_2[f(V_1)]$) are overlaid into a composite graph, so the score measures the proportion of conserved edges with respect to the number of unique edges in this composite graph. More formally:

  $$\mathsf{S}^3 = \frac{|f(E_1) \cap E_2|}{|E_1| + |E(G_2[f(V_1)])| - |f(E_1) \cap E_2|} \times 100$$

  As the S^3 score takes into account the composition of the source and target graphs, it penalizes both alignments that map denser regions to sparser ones, and vice versa.
- *Largest Connected Component* (LCC) [1]: Number of edges in the largest connected conserved subgraph, i.e., the subgraph obtained when considering only the edges conserved by the alignment. Note that the maximum size of this connected subgraph is equal to the smallest of the largest connected components of both input networks.
- *Node Correctness* (NC): Fraction of nodes that are correctly aligned. This measure can only be used in cases when the correct mapping is known, e.g., when aligning synthetic networks or a network with itself.

Then, the objective of the NA problem is to find an optimal alignment, i.e., a maximal one, with respect to a given quality measure. It is easy to see that the NA problem is *NP*-hard [9–11]

[1] The name *Edge Correctness* is widely used in the literature. However, as pointed out in [6], this name may be misleading, since in most alignments there is no meaningful definition of correctness. Therefore, it would be more suitable to use the terms *Edge Conservation* or *Edge Coverage*.

by reducing the well-known *NP*-complete *subgraph isomorphism* problem to the decision version of the NA problem.

The subgraph isomorphism problem consists in finding whether a graph G contains a subgraph isomorphic to another graph H. Recall that two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there exists a bijection $f : V_1 \mapsto V_2$ such that $(u, v) \in E_1$ implies $(f(u), f(v)) \in E_2$. Note that, according to the provided formulation of the NA problem, an alignment between two isomorphic networks has EC, ICS and S^3 scores equal to 1. Therefore, we can reformulate the subgraph isomorphism problem as finding an alignment between H and G such that $S^3 = 1$.

2. State of the art

Given the wide range of applications of the NA problem, different algorithms for tackling it can be found in the literature. Nonetheless, many of those algorithms are very specific to the respective areas and problem applications [2,12,13].

The most commonly used methods are heuristic approaches, which provide a good equilibrium between execution time and quality of the solutions. Roughly, heuristic methods consist in constructing a solution in a greedy manner by using (if provided) some type of node similarity score gathered from the input networks. One of the most popular branch of heuristic methods concerns seed-and-extend approaches. As their name indicates, these algorithms first align an initial pair of nodes (seed) and then iteratively try to align their neighbors (extend). Examples of such heuristic approaches include HubAlign [14], Spinal [11], Ghost [7] and Netal [15].

One of the most popular family of algorithms is the Graal family. All the algorithms from this family are based on the idea of *graphlets* [16,17]. Graphlets are small connected non-isomorphic induced subgraphs of a large network (see the Supplementary Material for more information). More specifically, they usually use the graphlet signature similarity in order to build an alignment. They differ, however, in the construction phase. Some of the algorithms from this family, in order of their publication, are Graal [1], h-Graal [18], mi-Graal [19] and l-Graal [20].

Later on, the use of metaheuristics was becoming popular. Metaheuristics are upper-level frameworks that use heuristics or other methods to efficiently explore the search space in order to find an optimal or near-optimal solution. These kind of methods usually try to directly maximize a given score, without taking into consideration the underlying topology of the input networks during the construction phase. This group of methods includes Genetic Algorithms (GedEvo [21], Magna [8] and Magna++ [22]), Ant Colony Optimization (ACOgna [23], ACOgna2 [24]), Memetic Algorithms (OptNetAlign [25]) and Simulated Annealing (NetCoffee [26], Sana [6]).

Representation learning methods try to learn an embedding function that maps each node to a low-dimensional embedding [27]. A mapping function is then learnt in order to associate both embedding spaces. The most popular methods of this kind are Regal [28], Pale [29] or Dana [30].

Another large class of techniques is the one of mine-and-merge methods. In contrast to the previous methods, they usually produce local-many-to-many alignments by finding small dense conserved sub-networks and then subsequently compare the obtained modules. Some of the most well-known mine-and-merge methods are PathBLAST [31], Phunkee [32], Beams [33], SSAlign [34] and Pinalog [35].

Finally, some works have considered the use of exact techniques, often by means of Integer Linear Programming (ILP) formulations. The different ILP models differ in the number of variables and in the type of constraints. The most well-known ILP methods are Natalie [9] and Natalie 2.0 [36].

3. Ant colony optimization

Most of the existing methods in the literature focus on either obtaining a good similarity score between the respective nodes and then aligning them in a greedy fashion, or on optimizing a given quality measure in a straightforward way without taking into consideration the underlying topological and structural information of the input networks. As a reply, we propose AntNetAlign, a new Ant Colony Optimization (ACO) algorithm to solve the Network Alignment problem that tackles this issue and brings together the different trends. ACO algorithms are based on the following general idea. At each iteration, solutions to the tackled optimization problem are probabilistically generated based both on the values of a so-called pheromone model and on information obtained from greedy functions. Some of the best solutions that were generated in the current or in earlier iterations are then used to modify the values of the pheromone model. In this way, the algorithm shifts the probability distribution for the generation of solutions in such a way that better and better solutions are generated over time. In our opinion, ACO algorithms are good candidates for tackling network alignment problems, because they allow for an easy incorporation of one or more greedy functions, for example, in the form of similarity scores. In fact, ACO algorithms are strongly dependent on the choice of well-working greedy functions for the problem to be solved. This is in contrast to metaheuristics based on local search – such as simulated annealing and tabu search – that require a neighborhood function as their main ingredient, and in which the exploitation of greedy functions is generally more difficult.

More specifically, we propose a $\mathcal{MAX} - \mathcal{MIN}$ Ant System (MMAS) in the hybercube framework [37] for tackling the NA problem. Together with the Ant Colony System (ACS) [38] variant, MMAS algorithms are among the most popular ACO variants nowadays. The implementation of MMAS in the hypercube framework is characterized by an automatic scaling of the pheromone values such that they can only assume values from [0, 1]. This allows for a structured way of initializing the pheromone values and for computing measures such as the convergence factor, which is an indication for the current state of the search process.

3.1. AntNetAlign

A solution to the NA problem for two given networks $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is represented in AntNetAlign by a vector π of size $|V_1|$, where $\pi_i = j$ if and only if node $v_i \in V_1$ is mapped to node $\overline{v}_j \in V_2$. Recall that a valid solution has to comply with the following restrictions from the problem formulation:

1. Each node from the source network has to be aligned to *exactly* one node from the target network.
2. Each node from the target network can be aligned to *at most* one node from the source network.

Note that, with this solution representation, we avoid mapping a node from the source network to different target nodes (restriction 1). Additionally, to satisfy restriction 2, we enforce that all the elements in the solution vector must be different.

The pheromone model $\mathcal{T}$ is composed of pheromone values τ_{ij} for each possible alignment of a source node $v_i \in V_1$ with a target node $\overline{v}_j \in V_2$. Although the hypercube framework naturally restricts the pheromone values to be between 0 and 1, further limits $0 < \tau_{min} \leq \tau_{ij} \leq \tau_{max} < 1$ are usually introduced in order to prevent the complete convergence of the algorithm. Typically τ_{min} is set to 0.001 and τ_{max} to 0.999. Initially, all pheromone values are set to 0.5.

Additionally, an *evaluation function* $g : \pi \mapsto \mathbb{R}$ is defined, which measures the quality of a given solution. In our case,

Algorithm 1 AntNetAlign

1: **input:** a problem instance (i.e., two networks $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$)
2: $\pi^{bsf} :=$ NULL #*Best-so-far solution*
3: $\pi^{rb} :=$ NULL #*Restart-best solution*
4: $cf := 0$ #*Convergence factor*
5: $gc :=$ FALSE #*Global convergence?*
6: `initializePheromones`($\mathcal{T}$)
7: **while** stopping criteria do not met **do**
8: $\pi^{ib} :=$ NULL #*Iteration-best solution*
9: **for** $it = 1, \ldots, n_{ants}$ **do**
10: $\pi :=$ `generateSolution`($G_1, G_2, \mathcal{T}$) #*see Section 3.2*
11: **if** $g(\pi) > g(\pi^{ib})$ **then** $\pi^{ib} := \pi$ **end if**
12: **if** $g(\pi) > g(\pi^{rb})$ **then** $\pi^{rb} := \pi$ **end if**
13: **if** $g(\pi) > g(\pi^{bsf})$ **then** $\pi^{bsf} := \pi$ **end if**
14: **end for**
15: `updatePheromones`($\mathcal{T}, cf, gc, \pi^{ib}, \pi^{rb}, \pi^{bsf}$)
16: $cf :=$ `computeConvergenceFactor`($\mathcal{T}$)
17: **if** $cf > 0.999$ **then**
18: **if** $gc =$ TRUE **then**
19: `initializePheromones`($\mathcal{T}$) #*Restart*
20: $\pi^{rb} :=$ NULL
21: $gc :=$ FALSE
22: $cf := 0$
23: **else**
24: $gc :=$ TRUE
25: **end if**
26: **end if**
27: **end while**
28: **output:** π^{bsf}, the best solution found

g can be any of the structural quality measures introduced in Section 1.2. When the solution is empty (i.e., $\pi =$ NULL), then $g(\pi) := 0$. If the solution is incomplete (i.e., not all nodes from the source network are aligned), then the evaluation function g only considers the subset of nodes aligned so far.

The pseudo-code of AntNetAlign is provided in Algorithm 1. Basically, at each iteration of the algorithm, n_{ants} solutions are constructed in a probabilistic manner. Subsequently, the pheromones values are updated based on the constructed solutions. This process is repeated until the algorithm converges. Then, the algorithm decides whether to restart the process, in which case the pheromone values are re-initialized to 0.5. A more detailed description is presented hereafter.

One of the characteristics of the MMAS algorithm is that it maintains the following three solutions during its execution:

- π^{bsf}: The best solution found since the beginning of the execution (i.e., the best-so-far solution).
- π^{rb}: The best solution generated since the last restart of the algorithm (i.e., the *restart-best* solution).
- π^{ib}: The best solution constructed at the current iteration (i.e., the *iteration-best* solution).

Furthermore, two control variables are used to keep track of the current state of the convergence process: gc is a boolean variable that indicates whether the algorithm has globally converged, and $cf \in \{0, 1\}$ is the so-called convergence factor. They are used to decide whether the algorithm needs to be restarted.

At the beginning of the execution (see lines 2 to 6 of Algorithm 1), all solutions are initialized to NULL, global convergence (gc) is set to FALSE and $cf = 0$. Moreover, the pheromone values in $\mathcal{T}$ are all initialized to 0.5 in function `initializePheromones`($\mathcal{T}$).

At each algorithm iteration, n_{ants} solutions are probabilistically constructed in function `generateSolution`($G_1, G_2, \mathcal{T}$), both based on greedy information (which uses pairwise node similarity) and on the pheromone values. Note that the solution construction process will be outlined in detail in Section 3.2. Each generated solution is then evaluated using the evaluation function g. Then, solutions π^{ib}, π^{rb} and π^{bsf} are properly updated (see lines 11 to 13).

Table 1
Values for weights w_{ib}, w_{rb}, and w_{bsf} with respect to the convergence factor cf and control variable gc.
Source: Extracted from [39].

gc	FALSE				TRUE
cf	<0.4	[0.4, 0.6)	[0.6, 0.8)	≥ 0.8	
w_{ib}	1	2/3	1/3	0	0
w_{rb}	0	1/3	2/3	1	0
w_{bsf}	0	0	0	0	1

Next, the pheromone values are updated as in any other MMAS algorithm implemented in the hypercube framework. This is done in function `updatePheromones`($\mathcal{T}, cf, gc, \pi^{ib}, \pi^{rb}, \pi^{bsf}$). Here, each of the three solutions π^{ib}, π^{rb} and π^{bsf} is given a weight w_{ib}, w_{rb} and w_{bsf}, respectively. Note that each weight represents the contribution of the corresponding solution to the pheromone update. These weights vary depending on the changing values of cf and gc (see Table 1). In fact, when the convergence factor value is low – that is, $cf < 0.4$ – the *iteration-best* solution has a more significant impact. As the convergence factor increases, the influence of the *iteration-best* solution π^{ib} decreases in favor of the *restart-best* solution π^{rb}. More specifically, when $0.4 \leq cf < 0.6$, the weight of the iteration-best solution decreases to 2/3 and the weight of the restart-best solution is set to 1/3. Moreover, when $0.6 \leq cf < 0.8$, the weight of the iteration-best solution decreases further to 1/3, while the weight of the restart-best solution increases to 2/3. When the convergence factor surpasses 0.999 for the first time after a pheromone (re-)initialization – this happens when $cf > 0.999$ and $gc =$ FALSE (see lines 17 and 23–25 of Algorithm 1) – the value of gc is set to TRUE in order to indicate that the algorithm has reached the state of global convergence. In this case, the weights of the iteration-best solution and the restart-best solution are set to 0, and the *best-so-far* solution π^{bsf} is given the complete weight – that is, weight 1 – (see Table 1). In case π^{bsf} is different to π^{rb}, this will cause the value of the convergence factor to decrease for some iterations, before increasing again until surpassing again the value 0.999 (see lines 17 and 18 of Algorithm 1). If this happens, the algorithm is re-started (lines 19–22 of Algorithm 1). Note that $w_{ib} + w_{rb} + w_{bsf} = 1$ at all times. Then, the pheromone update is conducted as follows:

$$\tau_{ij} := \tau_{ij} + \rho \cdot (\xi_{ij} - \tau_{ij}) \quad \forall\, \tau_{ij} \in \mathcal{T}, \tag{1}$$

where

$$\xi_{ij} := w_{ib} \cdot \chi(\pi_i^{ib}, j) + w_{rb} \cdot \chi(\pi_i^{rb}, j) + w_{bsf} \cdot \chi(\pi_i^{bsf}, j) \quad . \tag{2}$$

Parameter $\rho \in [0, 1]$ is the so-called learning rate, which determines the strength of the pheromone update. Function $\chi(\pi_i, j)$ evaluates to 1 if and only if node $v_i \in V_1$ is aligned to node $\overline{v}_j \in V_2$ in solution π (i.e., $\pi_i = j$). Otherwise, it evaluates to 0. To assure that the pheromone values τ_{ij} are still between the allowed bounds after the pheromone update, we set

$$\tau_{ij} = \max(\tau_{min}, \min(\tau_{ij}, \tau_{max})).$$

Afterwards, the convergence factor cf is computed according to the updated pheromone values in function `computeConvergenceFactor`($\mathcal{T}$). The degree of convergence of the algorithm increases as the pheromone values get closer to the boundaries. Therefore, $cf = 0$ only in the extreme case in which all the

Algorithm 2 Function `generateSolution`(G_1, G_2, $\mathcal{T}$) of Algorithm 1

input: two networks $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, pheromone model $\mathcal{T}$
$U := V_1$ *#Set of unaligned nodes*
$C := V_2$ *#Set of candidate nodes*
$\pi :=$ NULL
while $U \neq \emptyset$ **do** *#While not all vertices have been aligned*
  $v_i :=$ `selectNextNode`(U)
  $\overline{v}_j :=$ `selectCandidate`(C, i)
  $\pi_i := j$
  $U := U \setminus \{v_i\}$
  $C := C \setminus \{\overline{v}_j\}$
end while
output: solution π

pheromone values are set to 0.5, and $cf = 1$ when all pheromones are either at τ_{min} or at τ_{max}. In general, the convergence factor has a value ranging between 0 and 1. It is calculated in the following way:

$$cf := 2\left(\left(\frac{\sum_{\tau_{ij}\in\mathcal{T}} max\{\tau_{max}-\tau_{ij}, \tau_{ij}-\tau_{min}\}}{|\mathcal{T}|\cdot(\tau_{max}-\tau_{min})}\right)-0.5\right) \quad (3)$$

Lastly, the global convergence of the algorithm is determined by the value of the convergence factor. If $cf > 0.999$, we say that the algorithm has converged. Hence, binary variable gc is set to TRUE. Then, when the algorithm converges again (this time to the global-best solution), it is restarted: gc is again set to FALSE, the *restart-best* solution π^{rb} is initialized to NULL and the pheromone values are set to 0.5. Consequently, cf is then calculated as 0.

When the stopping criteria are met, the algorithm terminates. In our case, we used a maximum number of constructed solutions as the termination criterion. Although other criteria have been considered, this one balances well the different aspects of the algorithm. For instance, setting a maximum computation time requires some prior knowledge of the algorithm's behavior for each considered problem instance. Setting a tight time limit will prevent the algorithm from converging or finding a reasonably solution, while allowing the algorithm a large computation time will result in wasted computation time, as the algorithm may not be capable to increase the quality of the current solution at later stages of the execution.

3.2. Solution construction

The pseudo code of the solution construction mechanism employed by function `generateSolution`(G_1, G_2, $\mathcal{T}$) of Algorithm 1 is presented in Algorithm 2. The solution construction mechanism receives as input the two networks $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ and the pheromone model $\mathcal{T}$. During a solution construction, we maintain the set of unaligned nodes from the source network $U \subseteq V_1$ and the set of candidate nodes from the target network $C \subseteq V_2$, i.e., C is the set of nodes from the target network to which no node from the source network has yet been aligned. Initially, it holds that $U = V_1$ and $C = V_2$. The objective is then to map each node in U to a node in C. Moreover, $A = V_1 \setminus U$ denotes the set of already aligned nodes from the source network.

The solution construction procedure starts with an empty solution $\pi =$ NULL. At each step, exactly one node $v_i \in U$ is selected by applying function `selectNextNode`(U). The chosen node v_i is the next node of the source network that is to be aligned. As the greedy information used by our algorithm is based on the principle that two nodes are similar if and only if their respective neighborhoods resemble, the choice of node $v_i \in U$ from the source network is based on the assumption that nodes from the source network with a higher number of already-aligned neighbors may be better candidates for finding suitable target candidate nodes. Let $N_A(v_i) = N(v_i) \cap A$ be the set of neighbors of node v_i that are already aligned. Moreover, let *centrality*(i) be node v_i's centrality score, which refers in this context to the so-called *betweenness centrality*. Then, a random number $r \in [0, 1]$ is uniformly drawn. If $r \leq d_{rate}^{select}$, node v_i with the highest number of aligned neighbors is chosen (ties are randomly solved), i.e.,

$$v_i = \underset{v\in U}{\operatorname{argmax}}|N_A(v)|.$$

Otherwise, node v_i is selected by roulette-wheel-selection, whereby the probability of a node v_i to be chosen is proportional to $|N_A(v_i)| + centrality(i)$.[2] In the specific case where $U = V_1$ (i.e., no node has been aligned yet) or $N_A(v_i) = \emptyset$ for all $v_i \in U$ (i.e., no node has an aligned neighbor), the algorithm returns a node $v_i \in U$ by roulette-wheel selection based solely on the respective centrality scores.

Next, a target node $\overline{v}_j \in C$ is chosen among all the possible candidates in C as follows. First, a probability p_{ij} is computed for each $\overline{v}_j \in C$, where

$$p_{ij} = \frac{\eta_{ij}\cdot\tau_{ij}}{\sum_{\overline{v}_k\in C}\eta_{ik}\cdot\tau_{ik}} \quad . \quad (4)$$

Hereby,

$$\eta_{ij} = s_{ij}\cdot\varphi_{ij} \quad (5)$$

is the greedy information, which is obtained by a combination of two aspects:

1. s_{ij}, the pairwise similarity score concerning nodes v_i and $\overline{v}_j$;
2. φ_{ij}, an *improvement score* that estimates how much the current partial solution would improve if nodes v_i and $\overline{v}_j$ were aligned.

Joining these two scores allows us to incorporate different sources of knowledge into the solution construction process. Note also that s_{ij} provides global information, while φ_{ij} depends on the state of the current partial solution. The calculation of both scores is outlined in detail below.

After calculating probabilities p_{ij}, in `selectCandidate`(C, i) the chosen node v_i from the source network is mapped to a candidate node $\overline{v}_j$ selected from C. Consequently, π_i is set to j. For the purpose of this choice, a random number $r \in [0, 1]$ is drawn from a uniform distribution. If $r \leq d_{rate}^{align}$ (the so-called *determinism rate*), the node $\overline{v}_j$ with the highest probability is chosen. Otherwise, $\overline{v}_j$ is chosen by roulette-wheel selection based on the respective probabilities.

3.2.1. Calculation of the pairwise similarity scores

For calculating the similarity between pairs of nodes of both networks, we used the topological similarity introduced for NETAL [15]. NETAL uses a $|V_1| \times |V_2|$ topological score matrix S, where each position $s_{ij} \in S$ indicates the topological similarity between nodes $v_i \in V_1$ and $\overline{v}_j \in V_2$. This similarity is based on the assumption that two nodes are topologically similar if and only if their respective neighbors are similar. Note that this score matrix directly represents inter-network similarity between the nodes. Each element $s_{ij} \in S$ is initialized to 1 and iteratively updated based on the values of the neighbors of v_i and $\overline{v}_j$ at the previous iteration. Let s_{ij}^{l} be the similarity value between the two nodes at iteration l. Then, s_{ij}^{l+1} is calculated as follows:

[2] In roulette-wheel-selection, given a set of candidates with an assigned score each, a candidate is selected with a probability proportional to its score. Thus, candidates with higher scores are more likely to be selected.

1. A complete weighted bipartite graph $G_b = (V_b, E_b)$ is generated, where V_b is formed by two disjoint sets of nodes $N(v_i)$ and $N(\overline{v}_j)$ (i.e., the sets of neighbors of v_i in G_1 and of $\overline{v}_j$ in G_2), and $E = \{(v_{i'}, \overline{v}_{j'}) \mid v_{i'} \in N(v_i), \overline{v}_{j'} \in N(\overline{v}_j)\}$. The weight of each edge $(v_{i'}, \overline{v}_{j'}) \in E_b$ is set to the similarity of its endpoints at the last iteration (i.e., $w(v_{i'}, \overline{v}_{j'}) = s^l_{i'j'}$).
2. A *matching*[3] M is assembled by a greedy algorithm as follows. First, an edge $e = (v_{i^*}, \overline{v}_{j^*})$ is chosen so that $w(v_{i^*}, \overline{v}_{j^*}) \geq w(v_{i'}, \overline{v}_{j'})$ for every $(v_{i'}, \overline{v}_{j'}) \in E_b$. Among all the edges satisfying this condition, one is chosen uniformly at random. Then, edge e is added to set M, and both $v_{i^*}, \overline{v}_{j^*}$ and their respective incident edges are removed from G_b. This process is repeated until all edges are removed from the bipartite graph G_b.
3. Once the previous greedy procedure has ended, s^{l+1}_{ij} is calculated as follows:

$$s^{l+1}_{ij} = \frac{\sum_{(i^*,j^*)\in M} s^l_{i^*j^*}}{\max\{deg(v_i), deg(\overline{v}_j)\}} \quad , \tag{6}$$

where the numerator is the sum of the similarities concerning the matched neighbors. The denominator not only helps to normalize the values, but also assigns higher similarity scores to nodes with comparable degrees.

The procedure is repeated for a fixed number of iterations.

Fig. 1 shows an example of how the topological similarity matrix S is initialized and updated at each iteration. In Fig. 1(a), we can see two example networks, G_1 and G_2, and the corresponding $|V_1| \times |V_2|$ matrix S in which all positions are initialized to 1.

The values of S after the first iteration are displayed in Fig. 1(b). Moreover, we can see an example of how this topological similarity is calculated between node $b \in G_1$ and all the nodes of network G_2. More specifically, it shows the complete weighted bipartite graph that is constructed for each of the pairs, and how the subsequent similarity score is calculated for each of them. In fact, it becomes clear that the topological similarity score at the first iteration for each pair of nodes $v_i \in V_1$ and $\overline{v}_j \in V_2$, is

$$s^1_{ij} = \frac{\min\{deg(v_i), deg(\overline{v}_j)\}}{\max\{deg(v_i), deg(\overline{v}_j)\}}.$$

The same information can be found for the second iteration in Fig. 1(c). For instance, for calculating the updated topological similarity score for this iteration for nodes $b \in G_1$ and $\overline{a} \in G_2$, i.e. $s^2_{b\overline{a}}$, we create a bipartite weighted graph with all the neighbors of b in G_1 ($\{a, c, d\}$) on one side, and all the neighbors of $\overline{a}$ in G_2 ($\{\overline{d}, \overline{b}\}$) on the other. The weights of the edges are the respective topological similarities from the previous iteration, e.g., $w(c, \overline{d}) = s^1_{c\overline{d}} = 0.33$. Then, in the second step, we assemble the matching using the previously specified greedy approach. First, we can select, for example, edge $(c, \overline{b})$, as it has the highest possible weight (0.5). Consequently, all the edges adjacent to either node $c \in G_1$ or node $\overline{b} \in G_2$ are removed. Then, we can select edge $(d, \overline{d})$ with weight 0.33. With this, all the remaining edges are removed and the greedy matching approach is finished. Finally, the value of the topological similarity score between nodes $b \in G_1$ and $\overline{a} \in G_2$ for the next iteration is calculated as $s^2_{b\overline{a}} = {}^{0.5+0.33}/_{max(3,2)} \simeq 0.27$.

3.2.2. Calculation of the improvement scores

The *improvement score* φ_{ij} is calculated as follows. Let $ind_{G_1}(i,j)$ and $ind_{G_2}(i,j)$ be the number of edges that would be induced in the source and target networks, respectively, if node v_i was mapped to $\overline{v}_j$. More formally,

$$ind_{G_1}(i,j) = |\{v_k \in N(v_i) \mid v_k \in A\}| \quad ,$$

$$ind_{G_2}(i,j) = |\{\overline{v}_k \in N(\overline{v}_j) \mid \overline{v}_k \in f(A)\}| \quad .$$

Similarly, we define

$$con(i,j) = |\{\overline{v}_k \in N(\overline{v}_j) \mid \overline{v}_k \in f(N(v_i) \cap A)\}|$$

as the number of edges that would be conserved. Note that it happens that $con(i,j) \leq ind_G(i,j)$. Then,

$$\varphi_{ij} = \frac{con(i,j) + \epsilon}{ind_{G_1}(i,j) + ind_{G_2}(i,j) - con(i,j) + \epsilon} \quad ,$$

where ϵ is a very small constant to avoid obtaining a value of zero in the numerator, respectively the denominator. Note that this *improvement score* favors the alignment of nodes within areas of similar density while constructing the solution. Although some optimizing measures, such as EC, do not take into account the induced edges in the target network – respectively, the source network in the case of ICS– we believe it to be beneficial to keep the same improvement formula no matter which objective function is used. For instance, if the improvement score only considered conserved edges when maximizing EC, the algorithm might start greedily aligning nodes within sparse areas to denser ones, as it is easier to conserve edges in this way. However, this could be counterproductive in the long run, as the remaining unaligned regions of the respective networks could be of remarkably different density, making it harder to conserve edges at later stages of the construction process.

4. Experimental evaluation

In order to test the behavior and the applicability of our approach in different scenarios, in this section, we present a profound experimental evaluation. A variety of real-world instances were selected from different application areas, which are described hereafter. In the following subsections, we first describe the chosen competitors for our algorithm. Subsequently, the experiments and results for different sets of problem instances are detailed.

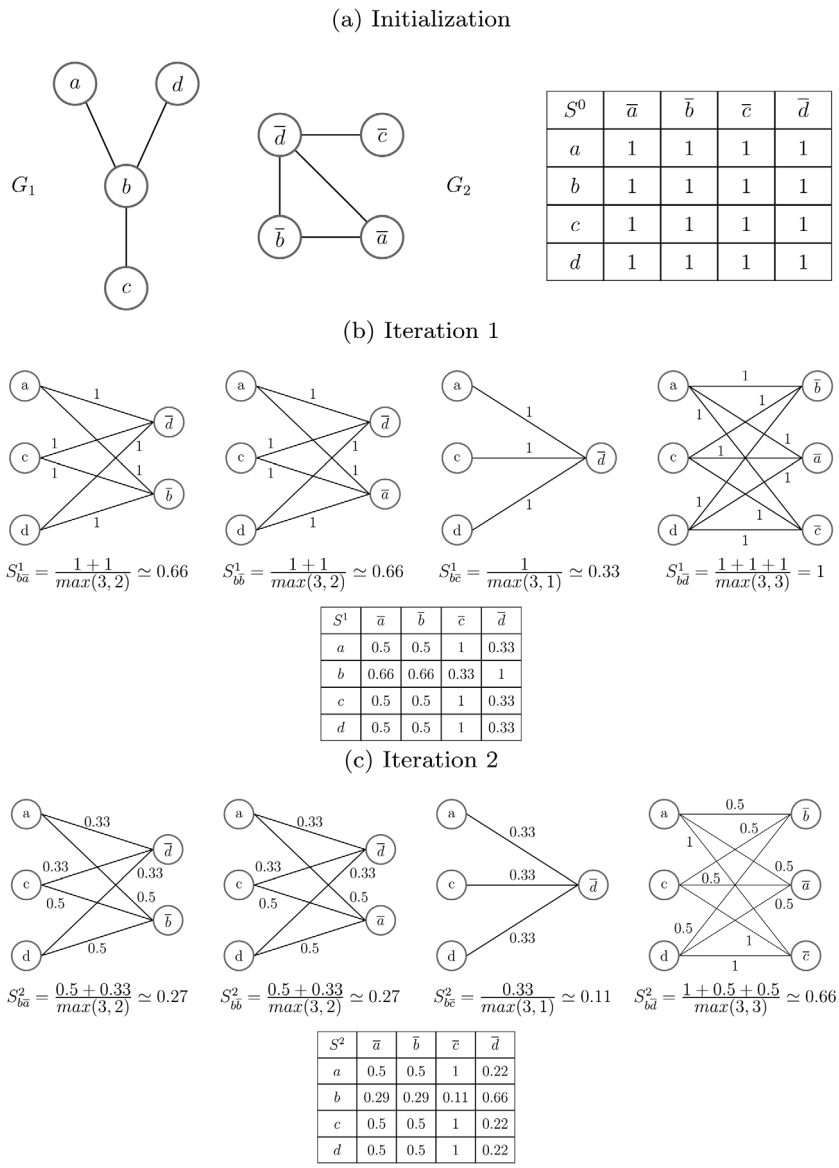
The benchmark instances considered in this work contain neither multi-edges nor isolated nodes (self-loops), as they were previously removed from the networks. This is done because many methods for the NA problem from the literature do not allow these kind of interactions. Moreover, isolated vertices can be aligned separately based solely on their own information, as they do not induce or conserve any edge.

4.1. Considered approaches

As the NA problem is relevant in various different application areas, the proposed algorithms are usually tested on instances from the targeted application. This makes it impossible to determine which methods are the current state of the art in general. Therefore, we compare AntNetAlign with approaches that have been proven to work very well in their respective area of application, while also having some kind of algorithmic relationship with our approach. Furthermore, we also consider algorithms using different types of similarity metrics. The following algorithms were selected:

- HubAlign [14]. We decided to use this method as (1st) it is a popular seed-and-extend method which has been proven to be state-of-the-art for the alignment of PPI networks; and (2nd) it uses a novel centrality measure to determine the importance of a node. HubAlign has a parameter α that controls the contribution of sequence similarity (in their case, the normalized BLAST bitscore for two proteins) relative to topological similarity. As we do not make use of sequence similarity, parameter α is set to 1 (only topological information is used). Other parameters were set to their default values.

[3] A matching is a set of edges without common vertices.



S^0	$\bar{a}$	$\bar{b}$	$\bar{c}$	$\bar{d}$
a	1	1	1	1
b	1	1	1	1
c	1	1	1	1
d	1	1	1	1

S^1	$\bar{a}$	$\bar{b}$	$\bar{c}$	$\bar{d}$
a	0.5	0.5	1	0.33
b	0.66	0.66	0.33	1
c	0.5	0.5	1	0.33
d	0.5	0.5	1	0.33

S^2	$\bar{a}$	$\bar{b}$	$\bar{c}$	$\bar{d}$
a	0.5	0.5	1	0.22
b	0.29	0.29	0.11	0.66
c	0.5	0.5	1	0.22
d	0.5	0.5	1	0.22

Fig. 1. Example of how the topological similarity matrix S is calculated. Reproduced (and extended) from [15].

- NETAL [15]. Similarly to HUBALIGN, NETAL is a greedy approach which has obtained good results for aligning PPI networks. Moreover, ANTNETALIGN makes use of the topological score matrix proposed in the context of NETAL. Again, we leave the parameters at their default values and only consider topological similarities.
- L-GRAAL [20]. This is the most recent member of the popular GRAAL family, which has been widely used in many different contexts. This method makes use of the graphlet signature similarity (see Section 2) to construct a sub-instance of the problem, for which it will optimize seed alignments. These seed alignments are then heuristically improved by considering all possible node mappings. All the parameters are set to default. Thus, the algorithm is given a maximum number of 1000 iterations and a time limit of 1 h (as default), which we consider fair given the limit of 1000 solution constructions of our proposals. Again, we do not make use of sequence information. The graphlet signatures are computed with the provided software, which considers graphlets up to a size of four.
- MAGNA++ [8]. This is a popular Memetic Algorithm (MA) which maximizes an edge-based similarity measure. We set the fitness measure to S^3 to be consistent with the optimization measure of our approach. Moreover, we selected a maximum of 1000 generations in order for MAGNA++ to have all the resources for begin able to work at its best. The population size and the fraction of elite individuals were obtained by parameter tuning (see Section 4.2.2). Although MAGNA++ accepts existing alignments as input – in addition to starting with a random initial population – we do not make use of this feature. This is because we do not want to bias the results obtained by the algorithm by combining it with other approaches.

Our algorithm was implemented in ANSI C++, using GCC 7.5.0 for compiling the code. The experimental evaluation was performed on a cluster of computers with "Intel® Xeon® CPU 5670" CPUs of 12 nuclei of 2933 MHz and (in total) 32 Gigabytes of RAM [40]. Note that, for the experimental evaluation, we used the implementations provided by the respective authors.

Table 2
Summary of benchmark instances: high-quality and low-quality PPI networks in ascending order of the networks.

Scientific name	Label	Organism	V	High-quality		Low-quality		
				E	⟨k⟩	E	⟨k⟩	Addl. edges (%)
B.taurus	bt	Cattle	119	114	1.91	136	2.28	19.30
O.sativa	os	Rice	181	192	2.12	196	2.16	2.08
R.norvegicus	rn	Rat	289	236	1.63	474	3.28	100.85
B.subtilis	bs	Bacterium	334v	775	4.64	883	5.28	13.94
E.coli	ec	Bacterium	709	692	1.95	1,200	3.38	73.41
S.pombe	sp	Yeast	1,335	2,176	3.26	2,683	4.02	23.30
M.musculus	mm	Mouse	1,966	2,704	2.75	7,237	7.36	167.64
C.elegans	ce	Worm	4,762	11,889	4.99	12,646	5.31	6.37
S.cerevisiae	sc	Yeast	5,260	22,064	8.39	47,924	16.86	117.20
A.thaliana	at	Plant	5,686	25,306	8.90	32,123	11.30	26.94
D.melanogaster	dm	Fly	8,382	31,754	7.58	41,610	9.93	31.04
H.sapiens	hs	Human	12,466	60,248	9.67	161,277	25.87	167.69

4.2. Experiments on Protein–Protein Interaction networks

Protein–Protein Interaction (PPI) networks represent pairwise biophysical protein interactions of an organism. Most of the proteins perform their functions via interactions. Hence, finding good alignments between these networks provide crucial information for decoding functional and biological roles and improving our understanding of the respective organisms.

4.2.1. Benchmark instances

The PPI networks were taken from HINT [41], a database of high-quality protein–protein interactomes. HINT compiles information extracted from several databases and resources. Low-quality or erroneous interactions are filtered both systematically and manually and removed from the database. Moreover, the database is regularly updated with new interactions.

We chose HINT over other databases from the literature because it is of utmost importance to ensure that these networks are of high quality, as results derived from erroneous hypotheses could lead to invalid conclusions. HINT assures this with its well-curated data. Although we do not aim to provide a complex biological analysis of the results, the use of rigorous real-world instances can validate the correctness of our approach.

Table 2 shows a summary of the PPI networks used in this work. We obtained PPI networks for a total of 12 different organisms. The scientific name of each organism is displayed alongside a label (which we will henceforth use to identify the networks) and the generic name of the organism to which they belong. Moreover, the order and size of each graph is given (number of proteins and interactions, respectively), alongside the mean degree ⟨k⟩, i.e., $\langle k \rangle = {}^{2|E(G)|}/_{|V(G)|}$. Note that there is a high variance in sizes and densities between the different networks.

Furthermore, we generate noisy versions of all the chosen PPI networks. For this reason, we use the low-quality PPI networks from the same database. As these low-quality networks may contain a larger number of proteins than their high-quality counterparts, we only consider those interactions between proteins that are included in the latter. The information of the resulting networks obtained after filtering the interactions of the original low-quality networks can also be found in Table 2. We additionally include the percentage of extra edges with respect to the original high-quality networks.

As it can be observed, the increase in the number of interactions considerably depends on the individual networks. Therefore, we consider the noise proportional to the difference between the number of interactions of the high-quality networks and the respective low-quality networks. More specifically, we start with the high-quality network as the initial state (0% noise) and increase the noise step-by-step by 10%, adding (at each step) 10% of those edges that are not present in the initial network, until we obtain the low-quality network as the final state (100% noise). Note that in the case of 100% noise, all low-quality interactions are included.

Table 3
Summary of the parameters of AntNetAlign with their respective domains and the tuning results. The parameter n_{ants} is the number of solution constructions per iteration (i.e., number of ants); ρ is the learning rate; d_{rate}^{select} is the determinism rate for selecting the next node to align; and d_{rate}^{align} is the determinism rate for candidate selection.

Parameters	Considered domain	Tuning results
n_{ants}	{3, 5, 10, 20}	10
ρ	{0.1, 0.2, 0.3, 0.4, 0.5}	0.3
d_{rate}^{select}	{0.0, 0.1, . . . , 0.9.1.0}	0.8
d_{rate}^{align}	{0.0, 0.1, . . . , 0.8, 0.9}	0.9

4.2.2. Algorithm tuning

In order to conduct a scientifically sound experimental evaluation, it is mandatory to find well-working values for the algorithm parameters. In our case, we use the `irace` tool [42] in order to tune the parameters of our algorithm (AntNetAlign) and the ones of Magna++. The main purpose of `irace` is to automatically configure optimization algorithms by finding the most appropriate parameter setting given a test set of problem instances. Note that AntNetAlign and Magna++ are the two only metaheuristic approaches considered in the comparison, while the other algorithms are heuristics.

As tuning instances, we selected two pairs of medium-sized networks. In particular, *E. coli* is aligned with *S.pombe* (`ec-sp`), and *S.pombe* with *M.musculus* (`sp-mm`).

***Tuning of* AntNetAlign**

Some of the parameter values of AntNetAlgin are fixed by experimental observation. For instance, we set the maximum number of solution constructions per run to 1000, as the algorithm is capable to converge (usually more than once) before this limit is reached. Moreover, the algorithm generally requires a reasonable amount of time for executing this number of iterations with respect to the size of the input networks. Furthermore, the number of iterations for calculating the topological similarities (s_{ij}) is set to 4, which is a small value (as recommended), but allows for a certain convergence of the values.

Table 3 shows a summary of the parameters chosen for tuning, together with a short description and the considered domains. The tuning budget, i.e., the number of maximum algorithm runs that the `irace` tool is allowed to use when optimizing the parameters, was set to 750.

Table 3 also shows, for each parameter, the best value obtained by the tuning process. As it can be observed, the tuning tool detected that 10 solution constructions per iteration and a learning rate of 0.3 work best. Moreover, the degree of determinism when selecting the next node to align should be high, but still low enough to ensure a certain degree of randomness. This helps to generate sufficiently different solutions, especially at the

Table 4
Summary of the parameters of MAGNA++ with their respective domains and the tuning results. The parameter p_{size} is the size of the population and *elite* is the fraction of elite individuals.

Parameters	Considered domain	Tuning results
p_{size}	{1, 2, . . . , 1999, 2000}	1824
elite	{0.30, 0.31, . . . , 0.69, 0.70}	0.31

beginning of the search process. Finally, a rather deterministic selection of the candidate node is also required.

***Tuning of* MAGNA++**

As previously stated, the number of iterations is fixed to 1000. Table 4 shows a summary of the tuned parameters and the considered domains. In this case, given that the number of parameters involved in the tuning process is lower than in the case of ANTNETALIGN, a maximum of 500 algorithm runs was given to the `irace` tool.

Table 4 also shows the best parameter values obtained. Notice that the tuning tool decided that a large population size and a small fraction of elite individuals was preferred, which is a rather expected result. A large population size allows the algorithm to explore more solutions, while a rather small ratio of elite individuals allows it to escape from local optima by generating deviant mutant individuals. Finally, note that – with these settings – MAGNA++ makes use of many more computational resources than ANTNETALIGN. Thus, the experimental setting is not to the disadvantage of MAGNA++.

4.2.3. Results

In order to test the performance of ANTNETALIGN, we compared it with the aforementioned methods in different scenarios. As our approach (and some of the selected ones) are of stochastic nature, we applied them 10 times to each problem instance. Moreover, note that all algorithms were given a maximum computation time of one hour. Note that ANTNETALIGN and MAGNA++, for example, have their natural stopping criteria of 1000 solution constructions (in the case of ANTNETALIGN) and 1000 iterations (in the case of MAGNA++). However, if they do not stop due to their natural stopping criterion, they are terminated after one hour of computation time. Moreover, for each produced alignment, we compute the following scores: the Edge Correctness (EC), the Induced Conserved Structure (ICS) and the Symmetric Substructure Score (S^3).

Pairwise alignment of PPI Networks

The first set of experiments consists of aligning all different pairs of the High-Quality PPI networks. Thus, as there are a total of 12 PPI networks representing different organisms, we have a total of $(12 \times 11)/2 = 66$ problem instances. Note that, when aligning two networks, the source network has to be the one with smaller order. This is why the number of possible combinations is divided by two. By considering all the different combinations we dispose of a diverse data set of problem instances with which we can test the behavior of the considered approaches in different scenarios.

Figs. 2–4 show the obtained results regarding the S^3, EC and ICS scores, respectively. Row labels specify the problem instances. Note that they are ordered in increasing order of the source and target networks, respectively.[4] Different background colors are used to partition the group of instances with respect to the source networks. For each instance and for each algorithmic method, the obtained mean (point) and standard deviation (vertical line) is given. Note that the lines joining the different results are only for visual clarity, as there exist no real relation between the results. The exact numerical results are included in the supplementary material to this paper.

Moreover, the results were statistically evaluated by means of applying the `scmamp` tool, which is a package for the comparison of optimization algorithms in R. First, this tool compares all the approaches simultaneously by applying the Friedman test. As a result we can reject the hypothesis that all the algorithms perform equally. Furthermore, the tool performs all pairwise comparisons using the Nemenyi post-hoc test [43]. Fig. 5 shows the resulting critical difference (CD) plots [44] for each of the three scores. These plots show the average algorithm ranks (on the horizontal axis) with respect to the considered set of instances. This is done for each of the three scores. A bold horizontal bar joining the markers indicates that the performances of the respective algorithms are considered statistically equivalent, i.e., they are below a critical difference threshold (with a significance level of 0.05).

After examining the obtained results regarding the S^3 score, the following analysis can be made:

- In general, ANTNETALIGN is the best-performing algorithm. Moreover, ANTNETALIGN outperforms the competing approaches with statistical significance (see the CD plot in Fig. 5(a)). Furthermore, L-GRAAL outperforms the remaining methods (MAGNA++, NETAL and HUBALIGN). Among the three worst-performing algorithms, MAGNA++ obtains better results than NETAL and HUBALIGN, with no statistical difference between these two.
- In the context of stochastic algorithms a certain variance of the results is expected. Nonetheless, the standard deviation of the results of our approach is generally very low. In particular, the standard deviation of the results of ANTNETALIGN is around 1% for smaller instances. The standard deviation seems even to decrease as the size of the instances increases. On the other side, this trend can be expected, as one miss-alignment in the context of smaller networks has, generally, a much bigger effect on the solution quality when compared with one miss-alignment in the context of larger networks.
- L-GRAAL obtains rather consistent results, regularly being in second place behind our algorithm. However, the performance of L-GRAAL drops dramatically when aligning certain networks (e.g. `bt-bs` or `rn-ce`). This can especially be observed in bigger instances. One reason behind this decrease in performance could be that – in the aforementioned cases – the structures of the two networks to align are so different that the first filtering by means of the similarity threshold is not able to generate a good sub-instance. Thus, the obtained seed alignments are not a good starting point. Recall that, occasionally, L-GRAAL is not able to fully converge in the given time limit, respectively in the maximum number of iterations. Another aspect to be mentioned is that L-GRAAL is characterized by a standard deviation of zero in the results, which is due to the deterministic nature of this technique.
- MAGNA++ obtains rather good results for the smaller instances, but they considerably deteriorate as instances become larger. The variation of the results is inversely proportional to the instance size, following a pattern similar to the one of ANTNETALIGN. However, a greater variability of the results is observed (e.g. up to a standard deviation of 4.94 for smaller instances).

[4] NOTE: The L-GRAAL algorithm requires the source network to have a smaller size (number of edges) than the target network, instead of the usual approach to consider the source network as the one with smaller order (number of nodes). Thus, when aligning the `bs` and `ec` networks, we had to swap their positions with respect to the other methods. Therefore, in this case, the presented scores are not consistent with the other results. We mark this case with an asterisk (∗).

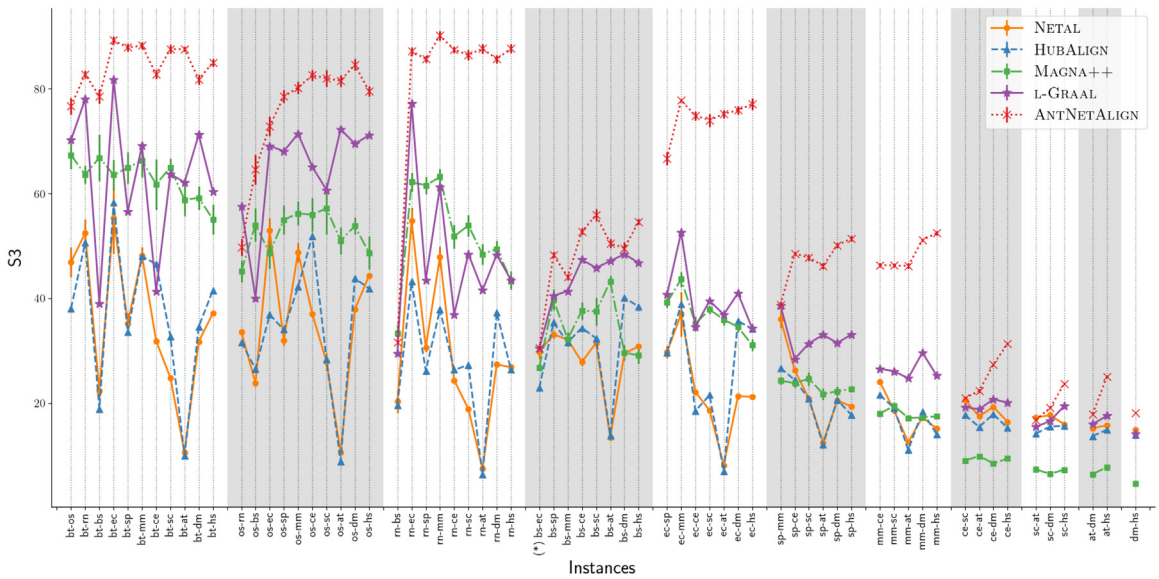


Fig. 2. Results concerning the S^3 score for all pairs of high-quality PPI networks.

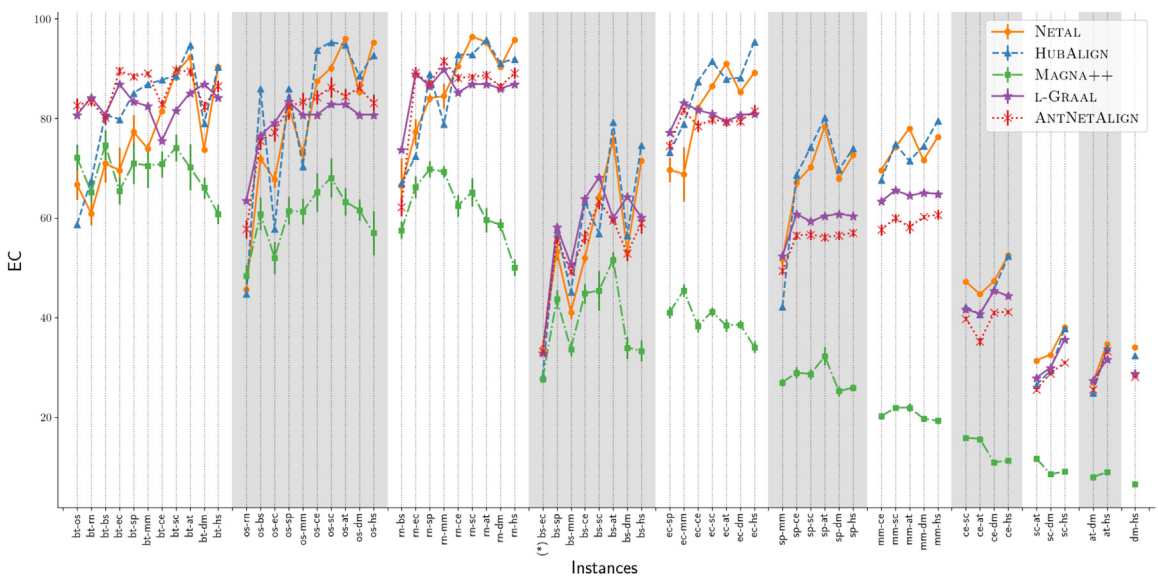


Fig. 3. Results concerning the EC score for all pairs of high-quality PPI networks.

- HUBALIGN and NETAL exhibit a fairly similar performance, which makes sense because they are both greedy methods that iteratively try to align – directly or indirectly – neighboring nodes. However, both methods are not able to obtain high-quality alignments with respect to the S^3 score, specially for small instances. On the one hand, NETAL seems to suffer (specially) from random decisions, as the standard deviation is rather large (up to 6.9) for smaller instances. Although the general trend is that the standard deviation decreases with an increasing instance size, some inconsistencies can be observed in this regard. In particular, NETAL sometimes obtains a standard deviation of 0 for smaller instances such as `bt-hs` and `os-ce`). This may be caused by tie-breaking decisions during the alignment process. On the other hand, although HUBALIGN also has to deal with tie breaking, its implementation appears to deal with them in a deterministic way.

Concerning the results for the EC score, the following observations can be made:

- HUBALIGN is the best-performing method, followed by NETAL, L-GRAAL and ANTNETALIGN. However, no statistical difference was detected among HUBALIGN NETAL, and L-GRAAL, and between L-GRAAL and ANTNETALIGN; see Fig. 5(b). Note that MAGNA++ is clearly the worst-performing algorithm concerning the EC measure.
- ANTNETALIGN obtains the best results for smaller instances, along with L-GRAAL. For medium-size instances, HUBALIGN seems to perform best, closely followed (and sometimes outperformed) by L-GRAAL and/or NETAL. The latter often performs best in the context of larger instances. As it can be observed, NETAL and HUBALIGN are both capable of consistently obtaining high-quality EC scores.
- The performance of MAGNA++ follows a similar trend to what was already observed concerning the S^3 score. That is, the quality of the solutions decreases as the size of the instances increases. Moreover, the difference with respect to the best-performing methods is more notable for the larger

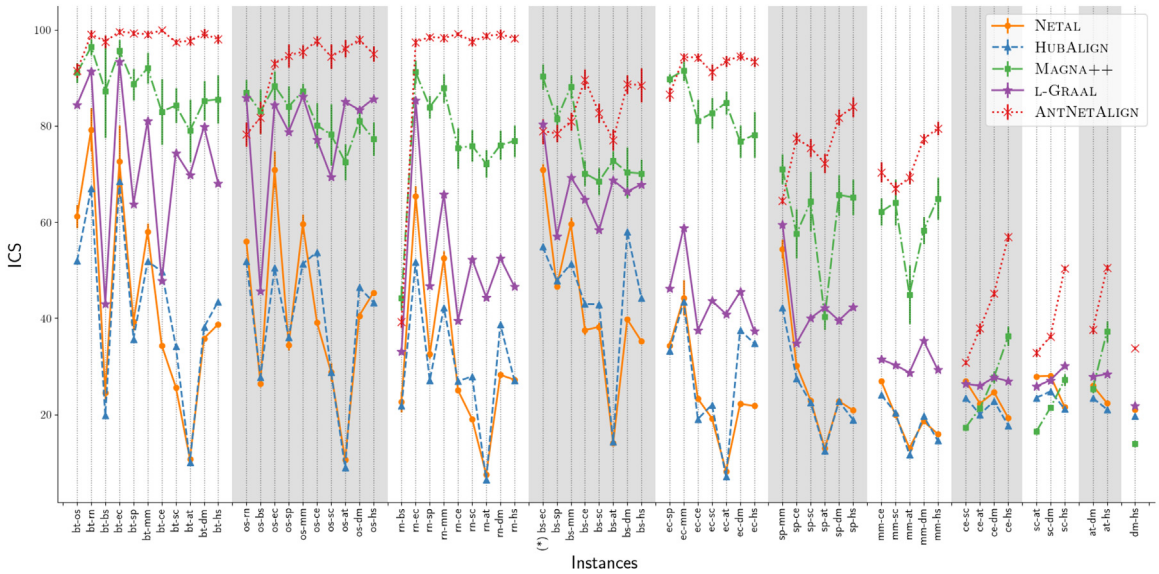


Fig. 4. Results concerning the ICS score for all pairs of high-quality PPI networks.

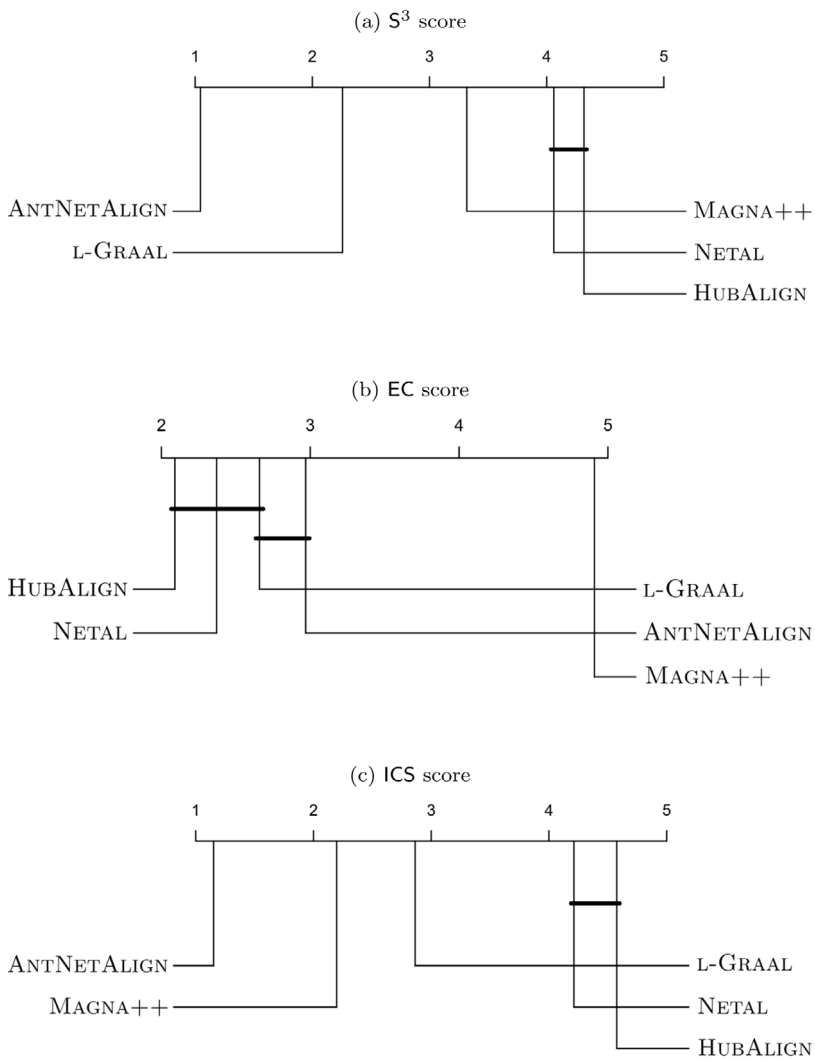


Fig. 5. Critical difference plots for the alignment of high-quality PPI networks.

instances, where MAGNA++ only accomplishes to conserve at most 10% of the edges of the source networks.
- With respect to the robustness (as seen by means of the standard deviations) of the approaches, the observations are very similar to the ones made in the context of the S^3 score.

Finally, the following observations can be made regarding the results for the ICS score:

- ANTNETALIGN is, as in the case of the S^3 score, the method that shows the best performance (see CD plot in Fig. 5(c)). In second and in third place are MAGNA++, respectively L-GRAAL. NETAL and HUBALIGN are the worst-performing techniques. The difference between NETAL and HUBALIGN is not statistically significant.
- MAGNA++ and L-GRAAL behave similarly to what was observed previously. The only exception is in the context of larger instances, where we can observe that MAGNA++ performs better as the size difference of the networks increases.
- Once more, NETAL and HUBALIGN act alike. However, they under-perform remarkably concerning this score. One special case to note is that their performance decays even more when aligning any network with network `at` (e.g. `bt-at` or `os-at`). This holds specially for those cases in which there is a large difference in the order of the networks. A poor ICS score indicates that the algorithm aligns sparse regions of the source network to denser ones of the target network. Note that the `at` network is the second most dense network, with an average node degree of approx. 8.9, only topped by the `hs` network with a value of 9.67. This possibly shows that both methods incorrectly align small sparse source networks to denser regions of the `at` network. The same does not seem to happen in the case of the `hs` network, which may indicate that, although having a similar density, both networks have different inherent structures.

In summary, the following conclusions can be drawn:

- ANTNETALIGN outperforms the other approaches with respect to the S^3 score, which is the most relevant measure, as it ensures that areas of similar density are aligned, as opposed to both EC and ICS that unsuccessfully penalize aligning sparse regions to denser ones and vice-versa. By minimizing the S^3 score, we obtain a balance between both EC and ICS, which can be observed in the CD plots (Fig. 5). In addition, our approaches is characterized by robust results, as the variance in the results is not as large as the one of the other approaches that employ random elements.
- MAGNA++, on the other hand, does not manage to obtain satisfactory results in many cases. Although it is the second best method concerning the ICS score, it is tied in the last place in the case of the other two measures, with no statistical difference. This demonstrates that making use of similarity metrics and dynamically-estimated improvements while constructing solutions, as it is done by ANTNETALIGN, can help to avoid poor regions of the search space, as opposed to just minimizing the objective function. This can be specially noted in the context of larger instances.
- NETAL and HUBALIGN exhibit a similar behavior. Although they stand out regarding the EC score, they fail at aligning sparse regions to areas of similar density in the target networks, thus obtaining a low ICS score. This is more accentuated as the difference in size between both networks increase. This may be because these greedy methods are not aware of the surrounding structure of the vertices when aligning them, besides using the information incorporated in the similarity matrix.
- L-GRAAL obtains satisfactory results regarding the different scores. However, it struggles when both networks are so different that the graphlet signature similarity is not good enough to filter the alignments and construct a well-delimited sub-instance, from which it can generate good-quality seed alignments to be heuristically improved.

Finally, we also compare the running times of the five competitors. The initialization of the algorithms is not taken into account when calculating the total execution time. Here, we consider the running time of an algorithm as the time at which the best solution generated by this algorithm in a run is found. Fig. 6 shows the running time of each algorithm for all instances ordered according to their increasing size. More specifically, we consider the size of an instance to be equal to the product of the number of vertices of both input networks ($|V(G_1)| \times |V(G_2)|$). We believe this is appropriate taking into account the nature of the problem. The running time is given in seconds. Note that the y-axis is in logarithmic scale.

As we can observe, the running time of all the methods is linear with respect to the size of the instances, which is quadratic. The only exception is L-GRAAL, which already reaches the maximum running time allowed for each algorithm (1 h) for medium size instances. This is expected, as the complexity of solving the relaxed problem is cubic. Generally, NETAL is faster than the other methods. ANTNETALIGN and HUBALIGN show a similar running time, with HUBALIGN being slightly faster for smaller instances. MAGNA++ follows a similar trend to the previous two methods, but it is slower in all of the cases. This is more notorious in the context of small and medium instances, where the running time difference can be of up to one order of magnitude.

Self-alignment of PPI Networks

As the tested PPI networks have diverse sizes and structures, the experiments described above allowed us to test the performance of our algorithm in very different scenarios. However, in order to gain a better understanding of the algorithms' behavior, we also evaluate the performance of all competitors when the source and target networks are equal.

This particular problem version is similar to the well-known graph *automorphism* problem, i.e., an *isomorphism* from a graph to itself. In this case, though, the trivial automorphism is allowed. In contrast to the NA problem, which was proved to be *NP*-hard, it is not known whether the graph isomorphism problem belongs to *P* or to *NP* [45], assuming that $P \neq NP$. However, if the problem of graph isomorphism is *NP*-complete, then the polynomial-time hierarchy collapses to Σ_2^p [46].

Note that, when aligning networks with the same number of nodes, all edges in both networks will be induced. Moreover, as both networks share the same number of edges, the EC and ICS scores will be exactly the same. It makes therefore no sense to make a distinction between them. Table 5 shows the EC (also ICS) scores obtained by aligning each of the 12 PPI networks to itself, following the experimental setup as explained above. As the S^3 score would not provide any additional information, the results of this score are not shown. In fact, we are only interested in the number (or fraction) of conserved edges.

Although the ground truth is known in the context of aligning networks to themselves, the Node Correctness (NC) score is not provided, as the tested algorithms have no way of discriminating between symmetric parts of the graphs. Thus, an alignment which maps a node to an equivalent one rather than to itself, is not considered incorrect. In the case of PPI networks, one could make use of sequence information to avoid this issue, such as the *normalized bit score* (e.g. obtained with BLAST [47]), which is independent of the database size. However, this is not considered in this work. Moreover, we do not provide CD plots since, in our opinion, the numerical information in Table 5 is sufficiently clear.

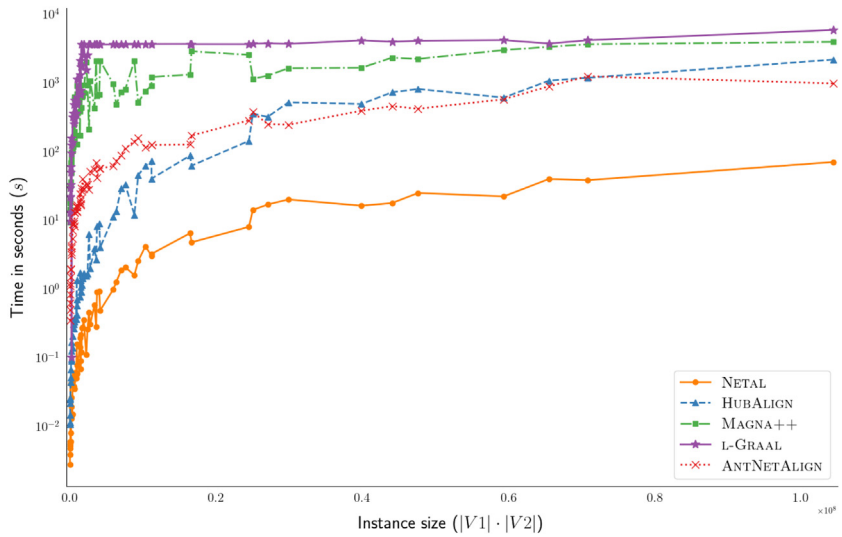


Fig. 6. Running times (in logarithmic scale) for all PPI instances ordered from small (left) to large (right). The definition of the running time is provided in the text.

Table 5
EC (also ICS) scores for aligning PPI networks to themselves.

Networks	NETAL	HUBALIGN	MAGNA++	L-GRAAL	ANTNETALIGN
bt-bt	45.96 ± 12.78	100.00 ± 0.00	80.88 ± 3.55	100.00 ± 0.00	100.00 ± 0.00
os-os	42.76 ± 1.08	100.00 ± 0.00	64.11 ± 3.91	100.00 ± 0.00	100.00 ± 0.00
rn-rn	71.44 ± 6.37	100.00 ± 0.00	63.98 ± 2.42	100.00 ± 0.00	100.00 ± 0.00
bs-bs	65.45 ± 18.80	100.00 ± 0.00	73.43 ± 5.23	100.00 ± 0.00	99.99 ± 0.04
ec-ec	72.20 ± 4.72	100.00 ± 0.00	55.66 ± 1.71	100.00 ± 0.00	100.00 ± 0.00
sp-sp	86.53 ± 7.73	100.00 ± 0.00	35.53 ± 1.36	100.00 ± 0.00	99.96 ± 0.05
mm-mm	75.97 ± 4.09	100.00 ± 0.00	27.93 ± 0.45	100.00 ± 0.00	99.93 ± 0.13
ce-ce	92.68 ± 3.11	100.00 ± 0.00	12.79 ± 0.46	100.00 ± 0.00	99.71 ± 0.07
sc-sc	99.76 ± 0.32	100.00 ± 0.00	10.18 ± 0.27	100.00 ± 0.00	99.59 ± 0.07
at-at	99.09 ± 0.12	100.00 ± 0.00	18.18 ± 0.45	100.00 ± 0.00	97.26 ± 0.38
dm-dm	97.55 ± 0.36	100.00 ± 0.00	5.71 ± 0.08	99.99 ± 0.00	99.75 ± 0.03
hs-hs	100.00 ± 0.00	100.00 ± 0.00	5.03 ± 0.08	100.00 ± 0.00	99.08 ± 0.12

After analyzing the obtained results, the following observations can be made:

- HUBALIGN and L-GRAAL are capable of always identifying the automorphism (except in one particular case when the latter does not). In the case of L-GRAAL, this result is expected, as the true alignments (from a node to itself) will always be considered in the sub-instance (as their similarity will be 1, which is greater than any possible threshold). In this way, the solver is able to find high-quality seed alignments. Thus, L-GRAAL is particularly good when both input networks resemble each other. In the case of HUBALIGN, the utilized importance measure is surely very useful for deciding which nodes to align. However, we believe that the reason behind the perfect alignments is the following. As we have previously mentioned, although the algorithm is expected to yield non-deterministic results given the randomness in its method for the similarity measure generation, its implementation deals with it in a deterministic way. In addition, the algorithms are provided with both networks in an edge-list format. As the same network file is passed twice for the self-alignment, the edges in this list will be ordered equivalently for both the source and the target network. Thus, HUBALIGN will calculate the similarity measure in exactly the same way in both cases, meaning that equal nodes will have the same importance score. If, on the other side, the edges would be slightly rearranged in the two lists (concerning source and target networks) we would expect the algorithm to compute the scores differently, making it harder for the algorithm to determine the automorphism.
- Surprisingly, the performance of NETAL improves with an increasing network size. In fact, NETAL is able to find the automorphism for the largest network (`hs`). A reason for this could be that the topological similarity obtained by the method works better for larger instances. However, it is worth noting that miss-aligning one node in the context of the smaller instances (and consequently its neighbors) can deteriorate the quality of the solution to a large extent. This can also be observed in the variance of the results, as the standard deviation tends to be higher for smaller instances (e.g. 12.78 for `bt-bt` or 18.80 for `bs-bs`) and vice versa.
- On the other side, MAGNA++ starts with high-quality solutions for rather small networks, but its performance deteriorates with an increasing network size. At the same time the variability of the results decreases. It is worth mentioning that MAGNA++ is only able to conserve around 5−6% of the edges in the two larger instances, while the rest of the methods are able (or nearly) to detect the automorphism.
- ANTNETALIGN performs almost optimally, specially for smaller instances, where it is able to find the automorphisms. For larger instances it consistently obtains an EC (also ICS) score of +99%. In addition, it obtains robust results as indicated by a low standard deviation. This shows that adding both a global similarity measure and dynamic information to the construction process of the solutions is beneficial, specially when both networks are rather similar.

Alignment of noisy PPI Networks

So far, we have conducted two types of experiments. First, the alignment of pairs of networks in which the source and the

target network are different to each other. And second, the self-alignment of networks. Remember that some of the considered methods behave differently in the two cases: while they obtain optimal or near-optimal solutions in the case of self-aligning networks, they under-perform when aligning different networks with each other.

In a third experiment, we now aim to test the robustness of the methods in the context of noisy instances. For this purpose we generated noisy instances for the problem by aligning the original high-quality PPI networks with noisy variants of these networks. More specifically, for each of 10 high-quality PPI networks we generated a series of noisy instances in which the level of noise is gradually increased. This will make it possible to study the evolution of algorithm performance as the networks to be aligned become less similar. The *D.melanogaster* and *H.sapiens* networks are not considered in this experiment, as their noisy variants surpass the network size limit accepted by some of the tested algorithms.

Fig. 7 shows the evolution of the S^3 score for each of the 10 considered networks as the noise increases. Networks are ordered in increasing number of nodes from the top-left to the bottom-right corner. The *x*-axis indicates the level of noise of the target network (from 0 to 100%). In summary, the following observations can be made:

- Both L-Graal and AntNetAlign seem more resistant to noise than other methods. Although L-Graal outperforms AntNetAlign for some of the medium-size instances, the opposite is the case in the context of the two larger instances.
- HubAlign accomplishes optimal results when both networks are equal (0% noise). However, its performance rapidly decreases with an increasing level of noise. Although Netal does not perform as well as HubAlign at low levels of noise, both methods tend to show a similar performance as the noise increases. This is in line with observations for previous experiments, where both methods worked similarly when aligning different PPI networks, but HubAlign outperformed Netal when aligning a network with itself.
- The performance of Magna++ seems quite invariant to noise. Nonetheless, while it obtains medium quality results in the context of smaller instances, it usually is the worst-performing method otherwise, specially for larger instances. This difference is much more eminent in some cases, like when aligning the *S.pombe*, *C.elegans* and *A.thaliana* networks to their noisy variants.

Thus, we can conclude that AntNetAlign is able to obtain high-quality alignments for real-world instances, partially due to its high resistance to noise.

Component analysis

In a last set of experiments we aim to show the usefulness of all components of the greedy information used in AntNetAlign, that is, the global similarity measure and the dynamic information. Recall that $\eta_{ij} = s_{ij} \cdot \varphi_{ij}$ is the greedy information, where s_{ij} is the pairwise similarity score of nodes v_i and $\overline{v}_j$, and φ_{ij} is a measure (an estimation) of how much the current partial solution would improve if nodes v_i and $\overline{v}_j$ were aligned (see also Eq. (5) in Section 3.2). Our experiments will test the importance of each of the two components of η_{ij}. For this purpose, the high-quality PPI networks are used again.

Fig. 8 shows the obtained results graphically by means of a so-called heat map. Column labels indicate which of the two greedy function components is maintained: the *similarity* ($\eta_{ij} = s_{ij}$), the *improvement* ($\eta_{ij} = \varphi_{ij}$), or *none* of them ($\eta_{ij} = 1$). Row labels indicate the respective instances. The value of each cell is calculated by dividing the score obtained by the respective reduced version of AntNetAlign with the score obtained by the complete method. In fact, removing any component always yields worse results. Therefore, this ratio oscillates between 0 and 1, which translates into the gradual lightness of the colors, that is, the darker the color of a cell, the worse is the respective reduced algorithm variant with respect to the complete algorithm.

As a general trend, the results obtained when removing components from the greedy function are always worse than those obtained with the complete function ($\eta_{ij} = s_{ij} \cdot \varphi_{ij}$). This difference is more noticeable as the size of the instances increases. Furthermore, the *improvement* seems to have the highest influence on the quality of the algorithm, as sometimes the respective algorithm variant is able to obtain nearly the same results as the complete algorithm. The *similarity*, in comparison, does not seem as influential as the *improvement*. One important thing to note is that the results obtained by only keeping the *improvement* are worse when aligning one network with the next one in size, but improve as the size difference of both networks increases. The same happens with the *similarity*, but inversely. Thus, we can deduce that both greedy function components complement each other. Additionally, given the observed results for larger instances, which are considerably worse than the ones of the complete method, we can conclude that both greedy function components are necessary.

Table 6
Summary of the benchmark instances (mental health disorders) in ascending order of the networks.

Network	V	E	⟨k⟩
DSM-IV (Core)	208	1949	18.74
DSM-IV	439	2626	11.96
ICD10	588	6169	20.98

4.3. Classification of mental health disorders

Nowadays, there exist two major nosological systems for the diagnostic classification of mental health disorders: the *International Classification of Diseases* (ICD-10) [48] and the *Diagnostic and Statistical Manual of Mental Disorders* (DSM-IV) [49]. Although both manuals categorize similar mental disorders, research has shown that there still exist some major differences between the current versions with respect to specific operationalization of many diagnoses [50]. Thus, NA can also be used to find concordance and prevalence of disorders between the two different classifications of mental disorders [51].

4.3.1. Benchmark instances

In [52], the authors used a network approach to study the overlapping structure between the diagnostic networks of the ICD-10 and DSM-IV systems. To do so, a network was constructed for each one by representing symptoms as nodes and connecting them by an edge if both symptoms co-occurred as a diagnostic criterion for at least one disorder. In this work, we make use of the same instances. Moreover, we also consider the giant connected component of the DSM-IV network (DSM-IV Core) used in [53] for community detection. Table 6 shows a summary of the resulting networks.

4.3.2. Results

As in the experiments previously described, the stochastic competitors AntNetAlign and Magna++ are applied 10 times to each considered case. Moreover, the same stopping criteria and the same parameter settings are used as outlined before. We first align the core of the DSM-IV network to the full DSM-IV network, to see if the algorithms are capable of fitting the subgraph

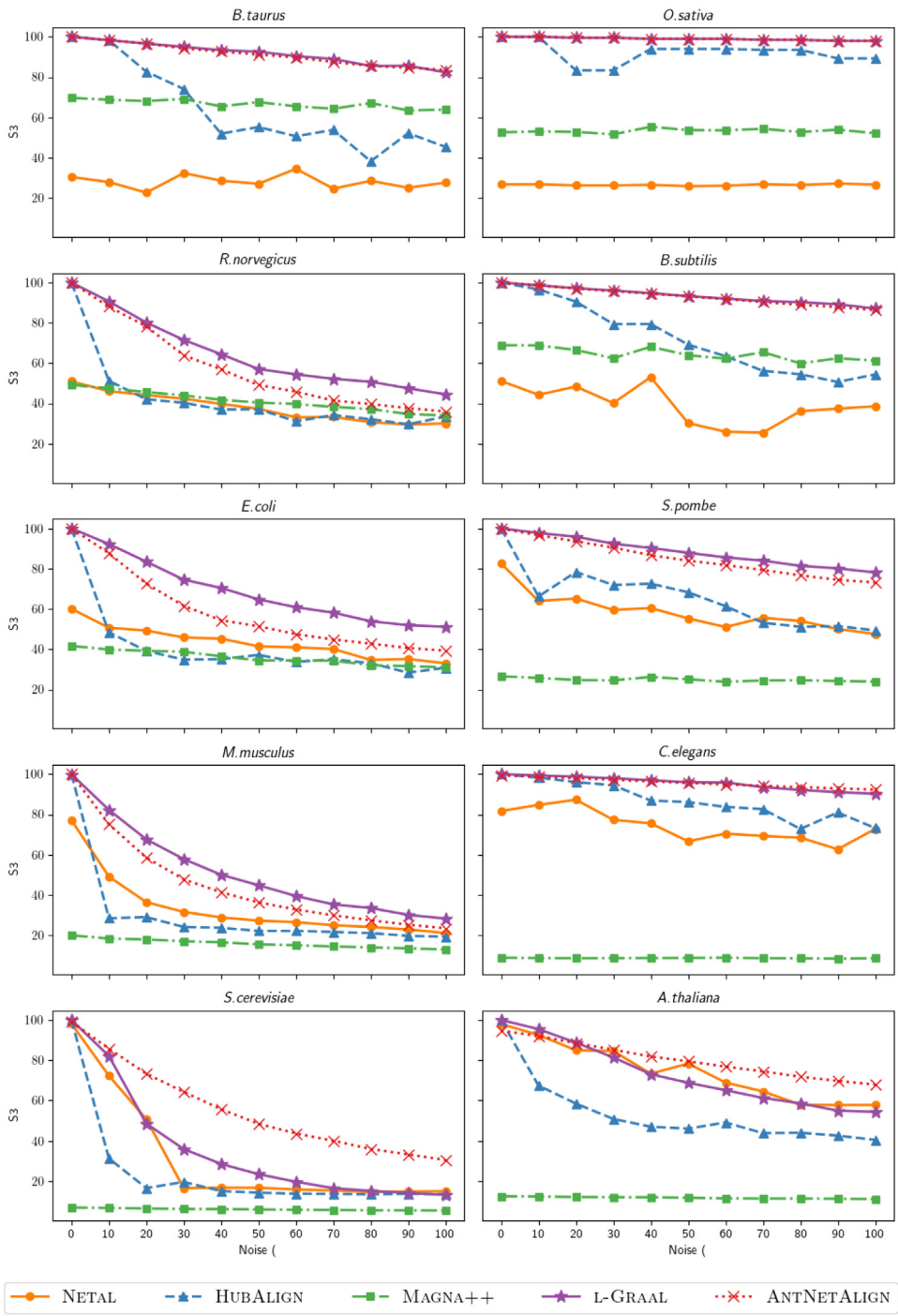


Fig. 7. S^3 scores obtained by the five competitors for series of noisy PPI networks.

into the larger network. The obtained results can be found in Table 7. We can observe that these results are almost identical to the ones from the Self-Alignment of PPI Networks (Table 5), where HubAlign and L-Graal are able to identify the subgraph isomorphism, while AntNetAlign performs near to optimality.

On the other hand, we try to align the DSM-IV network (both its core and the full network) with the ICD-10 network to see if the algorithms are capable of identifying similar regions between them. The S^3 score of the obtained results can also be found in Table 7. In this case, we can see that Magna++ is the best-performing method, closely followed by AntNetAlign. The L-Graal algorithm is only able to compete while aligning the DSM-IV Core to the ICD-10 network, but its performance decreases in the last case. Netal and HubAlign both obtain much worse results. Although the good performance of Magna++ may be surprising, this is certainly not in discordance to previous results, because Magna++ was able to find good quality alignments in small and medium instances regarding the S^3 score (see Fig. 2). Moreover, in this case, as the target network is denser than both source networks, it is easier to conserve edges in the source network (and thus, to obtain good EC scores). As Magna++ usually achieves good ICS scores, both aspects together may cause the

Table 7
S^3 scores for aligning mental health disorders networks.

Networks	NETAL	HUBALIGN	MAGNA++	L-GRAAL	ANTNETALIGN	ANTNETALIGN-1M
DSM-IV-Core-DSM-IV	57.95 ± 26.51	100.00 ± 0.00	75.17 ± 2.07	100.00 ± 0.00	99.93 ± 0.13	100.00 ± 0.00
DSM-IV-Core-ICD10	28.24 ± 4.38	35.84 ± 0.00	68.37 ± 2.70	59.40 ± 0.00	60.16 ± 1.61	76.65 ± 1.81
DSM-IV-ICD10	31.60 ± 0.34	27.46 ± 0.00	62.36 ± 3.67	40.61 ± 0.00	57.28 ± 2.87	67.49 ± 1.33

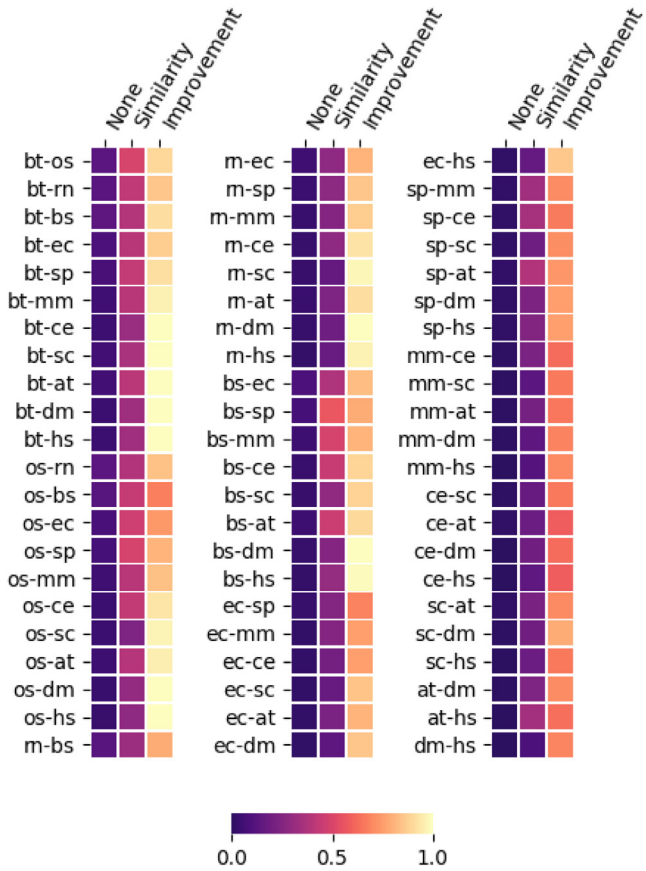


Fig. 8. Comparative results obtained by removing components of the greedy function.

generation of solutions in which both scores are balanced, thus obtaining good S^3 scores.

However, it is important to note that MAGNA++ is allowed to generate many more solutions than our approach during its execution. More specifically, at each generation MAGNA++ generates around $1824 \times 0.69 \approx 1258$ solutions, resulting in a total of about 1,258,000 solutions. Compared to the 1000 solutions generated by ANTNETALIGN, this is a huge number. Thus, we repeated the experiments concerning ANTNETALIGN with a new stopping criterion of one million solutions, which is much closer to the number of solutions generated by MAGNA++. The corresponding results can be found in the last column of Table 7 (AntNetAlign-1M). It can be observed that our algorithm is then capable of outperforming MAGNA++ and of increasing its performance by around 20% to 25%.

4.4. Social networks

As previously mentioned, one of the most important application areas NA is the area of social networks. For instance, one might want to align two social networks in order to discover similar behaviors between users. Following other papers on this area, we test the considered approaches therefore on real-world social networks.

Table 8
Summary of the benchmark instances from the application on social networks.

Network	V	E	⟨k⟩
Douban (Offline)	1118	1511	2.70
Douban (Online)	3906	8164	4.18
Flickr	12 974	16 149	2.49
Last.fm	15 436	16 319	2.11
Flickr	6714	7333	2.18
Myspace	10 693	10 686	1.99

4.4.1. Benchmark instances

For the experimental evaluation we make use of three different pairs of social networks, following the same procedure as in [2]. Table 8 provides a summary of the benchmark instances, grouped into pairs. They can be described as follows:

- Douban (Online) vs. Douban (Offline). Douban is an online social network that provides user reviews and recommendations ranging from music to books and films. Douban (Online) is a network which shows the relationship between users [54], while Douban (Offline) is a network where users are connected by an edge if they attend the same in-person social event(s) [55].
- Flickr vs. Last.fm. Flickr is an image and video hosting service, while Last.fm is a music recommendation website. Following the same procedures as in [2], both networks are crawled from Flickr and Last.fm [56] and processed as in [55].
- Flickr vs. MySpace. Finally, MySpace is a popular social networking service. Note that, here, Flickr corresponds to a different part of the Flickr network. Both these final networks are obtained following the same procedure as described before.

4.4.2. Results

In this last set of experiments, all the competitors were again applied with the experimental setup as already described before. In particular, ANTNETALIGN and MAGNA++ were applied 10 times to each of the three network pairs, with the same parameter settings and stopping criteria as utilized before. The obtained results can be observed in a way which summarizes over all three network pairs in Fig. 9. In particular, results are shown – regarding the S^3, EC and ICS scores – by means of critical difference plots. Interestingly, we observe a similar trend as in previous results: L-GRAAL and ANTNETALIGN obtain the best results regarding the S^3 score, with a difference between them which is not statistically significant. Moreover, NETAL is also able to achieve very good results. On the other side, MAGNA++ is clearly worst-performing algorithm. This can be explained by the fact that two of the three considered pairs of networks include large networks. In fact, these results are in line with the ones obtained when aligning PPI Networks, where MAGNA++ was not able to achieve good quality results for larger instances. Regarding the EC score, we can see that HUBALIGN performs now weaker when compared to previous experiments, and NETAL is able to obtain the best results. Concerning the results for the ICS score, L-GRAAL, ANTNETALIGN and MAGNA++ are able to outperform the other two methods, with no statistical difference among them.

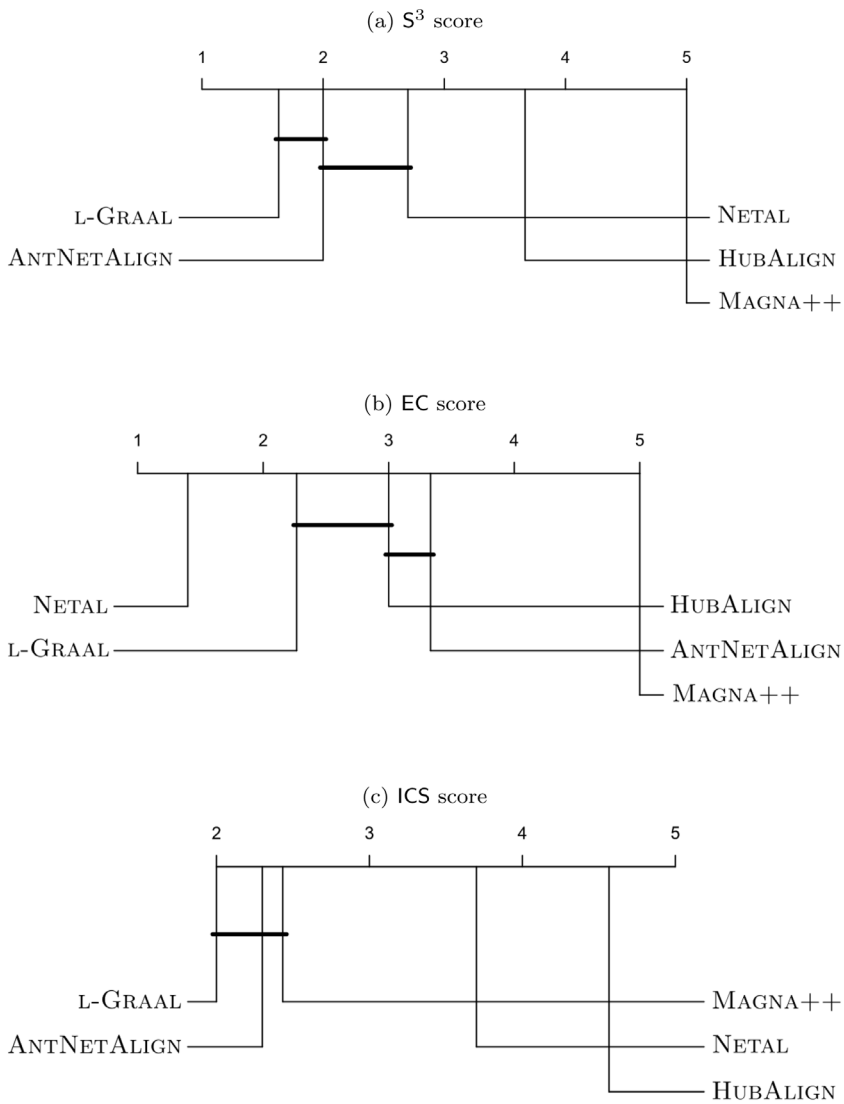


Fig. 9. Results for social networks shown in a summarized way by means of critical difference plots.

5. Conclusions and future work

In this work, we have introduced AntNetAlign, a new Ant Colony Optimization algorithm for solving the Network Alignment problem. The key novelties of this approach are the following. First, the algorithm is able to make use of any pairwise node similarity information to guide the solution construction process. As such a similarity measure is not restricted to any specific kind, the algorithm allows for a high versatility for being applied in different contexts. Second, it combines this similarity metric with an improvement measure that takes into account the current state of the solution construction. Thus, the algorithm is provided with both a global and a local view of the undergoing construction process.

In general, our approach is able to outperform the considered competitors in the context of two out of three of the tested quality scores. More specifically, it obtains significantly better results when using the superior S^3 score in a reasonable amount of time. Additionally, our method obtains near-optimal results when aligning networks with themselves. One of the reasons behind the good performance of our approach might be its high resistance to noise. Moreover, in the context of some instances from the field of mental health disorders for which Magna++ was able to outperform our approach, AntNetAlign was still able to perform better than Magna++ when given a similar number of solution constructions as Magna++. For larger instances, our method was always able to compete with the best-performing approaches.

Given the generality of our approach, one possible line of future research is to apply the ACO algorithm to networks from other application areas. For example, one might align linguistic networks to discover syntactic and semantic relationships between words of different languages. A second possibility concerns the transfer of ageing-related knowledge from well studied species to poorly annotated species [57].

Another task that is of great interest is link prediction. When studying alignments obtained by any algorithm, it is common to notice edge mismatches, i.e., connected nodes in the source network that are mapped to disconnected nodes in the target network or vice-versa. Sometimes, these mismatches may indicate that an edge may be missing, specially when the topological or functional contexts of the nodes are similar.

Lastly, we plan to extend/improve AntNetAlign by adding *negative learning*, in addition to the usual ACO feature of learning from positive examples. Recently, a new model for negative learning was introduced in [39], which makes use of additional optimization methods for identifying components of solutions

that should receive negative feedback. We believe that such an algorithm extension could be very useful in the context of network alignment.

CRediT authorship contribution statement

Guillem Rodríguez Corominas: Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Writing – original draft, Visualization. **Maria J. Blesa:** Investigation, Writing – review & editing, Supervision, Funding acquisition. **Christian Blum:** Conceptualization, Methodology, Investigation, Resources, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

Christian Blum and Guillem Rodríguez Corominas, Spain were supported by grants PID2019-104156GB-I00 and TED2021-129319B-I00 funded by MCIN/AEI/10.13039/501100011033. Maria J. Blesa acknowledges support from AEI, Spain under grant PID-2020-112581GB-C21 (MOTION) and the Catalan Agency for Management of University and Research Grants (AGAUR), Spain under grant 2017-SGR-786 (ALBCOM).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.asoc.2022.109832.

References

[1] O. Kuchaiev, T. Milenković, V. Memišević, W. Hayes, N. Pržulj, Topological network alignment uncovers biological function and phylogeny, J. R. Soc. Interface 7 (50) (2010) 1341–1354.
[2] H.T. Trung, N.T. Toan, T.V. Vinh, H.T. Dat, D.C. Thang, N.Q.V. Hung, A. Sattar, A comparative study on network alignment techniques, Expert Syst. Appl. 140 (2020) 112883.
[3] M.A. Ahmad, Z. Borbora, J. Srivastava, N. Contractor, Link prediction across multiple social networks, in: 2010 IEEE International Conference on Data Mining Workshops, IEEE, 2010, pp. 911–918.
[4] Y. Dong, J. Tang, S. Wu, J. Tian, N.V. Chawla, J. Rao, H. Cao, Link prediction and recommendation across heterogeneous social networks, in: 2012 IEEE 12th International Conference on Data Mining, IEEE, 2012, pp. 181–190.
[5] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, C. Zhu, Personalized recommendation via cross-domain triadic factorization, in: Proceedings of the 22nd International Conference on World Wide Web, ACM Press, 2013, pp. 595—606.
[6] N. Mamano, W.B. Hayes, SANA: simulated annealing far outperforms many other search algorithms for biological network alignment, Bioinformatics 33 (14) (2017) 2156–2164.
[7] R. Patro, C. Kingsford, Global network alignment using multiscale spectral signatures, Bioinformatics 28 (23) (2012) 3105–3114.
[8] V. Saraph, T. Milenković, MAGNA: Maximizing accuracy in global network alignment, Bioinformatics 30 (20) (2014) 2931–2940.
[9] G.W. Klau, A new graph-based method for pairwise global network alignment, BMC Bioinform. 10 (S59) (2009).
[10] W.B. Hayes, An introductory guide to aligning networks using SANA, the simulated annealing network aligner, in: Methods in Molecular Biology, Vol. 2074, 2019, pp. 263–284.
[11] A.E. Aladağ, C. Erten, SPINAL: scalable protein interaction network alignment, Bioinformatics 29 (7) (2013) 917–924.
[12] C.-Y. Ma, C.-S. Liao, A review of protein–protein interaction network alignment: From pathway comparison to global alignment, Comput. Struct. Biotechnol. J. 18 (2020) 2647–2656.
[13] P.H. Guzzi, T. Milenković, Survey of local and global biological network alignment: the need to reconcile the two sides of the same coin, Brief. Bioinform. 19 (3) (2017) 472–481.
[14] S. Hashemifar, J. Xu, HubAlign: an accurate and efficient method for global alignment of protein–protein interaction networks, Bioinformatics 30 (17) (2014) 438–444.
[15] B. Neyshabur, A. Khadem, S. Hashemifar, S.S. Arab, NETAL: a new graph-based method for global alignment of protein–protein interaction networks, Bioinformatics 29 (13) (2013) 1654–1662.
[16] N. Pržulj, D.G. Corneil, I. Jurisica, Modeling interactome: scale-free or geometric? Bioinformatics 20 (18) (2004) 3508–3515.
[17] T. Milenković, N. Pržulj, Uncovering biological network function via graphlet degree signatures, Cancer Inform. 6 (2008) 257–273.
[18] T. Milenković, W.L. Ng, W. Hayes, N. Pržulj, Optimal network alignment with graphlet degree vectors, Cancer Inform. 9 (2010) 121–137.
[19] O. Kuchaiev, N. Pržulj, Integrative network alignment reveals large regions of global network similarity in yeast and human, Bioinformatics 27 (10) (2011) 1390–1396.
[20] N. Malod-Dognin, N. Pržulj, L-GRAAL: Lagrangian graphlet-based network aligner, Bioinformatics 31 (13) (2015) 2182–2189.
[21] R. Ibragimov, M. Malek, J. Guo, J. Baumbach, GEDEVO: An evolutionary graph edit distance algorithm for biological network alignment, OpenAccess Ser. Inform. 34 (2013) 68–79.
[22] V. Vijayan, V. Saraph, T. Milenković, MAGNA++: Maximizing accuracy in global network alignment via both node and edge conservation, Bioinformatics 31 (14) (2015) 2409–2411.
[23] H.T. Ngoc, H.H. Xuan, ACOGNA: An efficient method for protein-protein interaction network alignment, in: 2016 Eighth International Conference on Knowledge and Systems Engineering, IEEE, 2016, pp. 7–12.
[24] N.A.V. Thi, N.H. Tran, D.D. Do, P. Thai, ACOGNA2: A novel algorithm for maximizing accuracy in global network alignment, in: 2019 6th NAFOSTED Conference on Information and Computer Science, IEEE, 2019, pp. 44–48.
[25] C. Clark, J. Kalita, A multiobjective memetic algorithm for PPI network alignment, Bioinformatics 31 (12) (2015) 1988–1998.
[26] J. Hu, B. Kehr, K. Reinert, NetCoffee: a fast and accurate global alignment approach to identify functionally conserved proteins in multiple networks, Bioinformatics 30 (4) (2013) 540–548.
[27] A. Grover, J. Leskovec, Node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 855—864.
[28] M. Heimann, H. Shen, T. Safavi, D. Koutra, REGAL: Representation learning-based graph alignment, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, ACM, 2018, pp. 117—126.
[29] T. Man, H. Shen, S. Liu, X. Jin, X. Cheng, Predict anchor links across social networks via an embedding approach, in: Proceedings of the 25th International Joint Conference on Artificial Intelligence, AAAI Press, 2016, pp. 1823—1829.
[30] T. Derr, H. Karimi, X. Liu, J. Xu, J. Tang, Deep adversarial network alignment, in: Proceedings of the 30th ACM International Conference on Information and Knowledge Management, ACM, 2021, pp. 352—361.
[31] B.P. Kelley, B. Yuan, F. Lewitter, R. Sharan, B.R. Stockwell, T. Ideker, PathBLAST: a tool for alignment of protein interaction networks, Nucleic Acids Res. 32 (Web Server issue) (2004) 83–88.
[32] A.P. Cootes, S.H. Muggleton, M.J. Sternberg, The identification of similarities between biological networks: Application to the metabolome and interactome, J. Mol. Biol. 369 (4) (2007) 1126–1139.
[33] F. Alkan, C. Erten, BEAMS: backbone extraction and merge strategy for the global many-to-many alignment of multiple PPI networks, Bioinformatics 30 (4) (2013) 531–539.
[34] A. Elmsallati, S. Roy, J.K. Kalita, Exploring symmetric substructures in protein interaction networks for pairwise alignment, in: Bioinformatics and Biomedical Engineering, Springer International Publishing, 2017, pp. 173–184.
[35] H.T.T. Phan, M.J.E. Sternberg, PINALOG: a novel approach to align protein interaction networks—implications for complex detection and function prediction, Bioinformatics 28 (9) (2012) 1239–1245.
[36] M. El-Kebir, J. Heringa, G. Klau, Natalie 2.0: Sparse global network alignment as a special case of quadratic assignment, Algorithms 8 (4) (2015) 1035–1051.
[37] C. Blum, M. Dorigo, The hyper-cube framework for ant colony optimization, IEEE Trans. Syst. Man Cybern. B (Cybern.) 34 (2) (2004) 1161–1172.
[38] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1) (1997) 53–66.

[39] T. Nurcahyadi, C. Blum, Adding negative learning to ant colony optimization: A comprehensive study, Mathematics 9 (4) (2021) 361.
[40] Research & development lab (/**rd**lab). Universitat politècnica de catalunya - BarcelonaTech, 2021, URL https://rdlab.cs.upc.edu.
[41] J. Das, H. Yu, HINT: High-quality protein interactomes and their applications in understanding human disease, BMC Syst. Biol. 6 (92) (2012).
[42] M. López-Ibáñez, J. Dubois-Lacoste, L.P. Cáceres, M. Birattari, T. Stützle, The irace package: Iterated racing for automatic algorithm configuration, Oper. Res. Perspect. 3 (2016) 43–58.
[43] S. García, F. Herrera, An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, J. Mach. Learn. Res. 9 (89) (2008) 2677–2694.
[44] B. Calvo, G. Santafé, Scmamp: Statistical comparison of multiple algorithms in multiple problems, R J. 8 (1) (2016) 248–256.
[45] M.R. Garey, D.S. Johnson, Computers and Intractability : A Guide To the Theory of NP-Completeness, W.H. Freeman, San Francisco, 1979.
[46] U. Schöning, Graph isomorphism is in the low hierarchy, J. Comput. System Sci. 37 (3) (1988) 312–323.
[47] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, D.J. Lipman, Basic local alignment search tool, J. Mol. Biol. 215 (3) (1990) 403–410.
[48] WHO, The ICD-10 classification of mental and behavioural disorders : diagnostic criteria for research, 1993, URL https://www.who.int/publications/i/item/9241544228.
[49] American Psychiatric Association, et al., Diagnostic and statistical manual of mental disorders, text revision (DSM-IV-TR®), 2010.
[50] C. Adornetto, A. Suppiger, T. In-Albon, M. Neuschwander, S. Schneider, Concordances and discrepancies between ICD-10 and DSM-IV criteria for anxiety disorders in childhood and adolescence, Child Adolesc. Psychiatry Mental Health 6 (1) (2012).
[51] M.N. Haque, L. Mathieson, P. Moscato, A memetic algorithm approach to network alignment: Mapping the classification of mental disorders of DSM-IV with ICD-10, in: Proceedings of the Genetic and Evolutionary Computation Conference, ACM, 2019, pp. 258–265.
[52] P. Tio, S. Epskamp, A. Noordhof, D. Borsboom, Mapping the manuals of madness: Comparing the ICD-10 and DSM-IV-TR using a network approach, Int. J. Methods Psychiatr. Res. 25 (4) (2016) 267–276.
[53] M.N. Haque, P. Moscato, The cohesion-based communities of symptoms of the largest component of the DSM-IV network, J. Interconnect. Netw. 19 (2019) 1940002:1–1940002:20.
[54] E. Zhong, W. Fan, J. Wang, L. Xiao, Y. Li, ComSoc: adaptive transfer of user behaviors over composite social network, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, 2012, pp. 696—704.
[55] S. Zhang, H. Tong, FINAL: Fast attributed network alignment, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 1345—1354.
[56] Y. Zhang, J. Tang, Z. Yang, J. Pei, P.S. Yu, COSNET: Connecting heterogeneous social networks with local and global consistency, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 1485—1494.
[57] F.E. Faisal, H. Zhao, T. Milenković, Global network alignment in the context of aging, IEEE/ACM Trans. Comput. Biol. Bioinform. 12 (1) (2015) 40–52.