



Exploiting causality in gene network reconstruction based on graph embedding

Gianvito Pio¹ · Michelangelo Ceci^{1,2,3} · Francesca Prisciandaro¹ · Donato Malerba^{1,2}

Received: 18 June 2018 / Revised: 19 August 2019 / Accepted: 14 November 2019 /

Published online: 3 December 2019

© The Author(s), under exclusive licence to Springer Science+Business Media LLC, part of Springer Nature 2019

Abstract

Gene network reconstruction is a bioinformatics task that aims at modelling the complex regulatory activities that may occur among genes. This task is typically solved by means of link prediction methods that analyze gene expression data. However, the reconstructed networks often suffer from a high amount of false positive edges, which are actually the result of indirect regulation activities due to the presence of common cause and common effect phenomena or, in other terms, due to the fact that the adopted inductive methods do not take into account possible causality phenomena. This issue is accentuated even more by the inherent presence of a high amount of noise in gene expression data. Existing methods for the identification of a transitive reduction of a network or for the removal of (possibly) redundant edges suffer from limitations in the structure of the network or in the nature/length of the indirect regulation, and often require additional pre-processing steps to handle specific peculiarities of the networks (e.g., cycles). Moreover, they are not able to consider possible community structures and possible similar roles of the genes in the network (e.g. hub nodes), which may change the tendency of nodes to be highly connected (and with which nodes) in the network. In this paper, we propose the method INLOCANDA, which learns an inductive predictive model for gene network reconstruction and overcomes all the mentioned limitations. In particular, INLOCANDA is able to (i) identify and exploit indirect relationships of arbitrary length to remove edges due to common cause and common effect phenomena; (ii) take into account possible community structures and possible similar roles by means of graph embedding. Experiments performed along multiple dimensions of analysis on benchmark, real networks of two organisms (*E. coli* and *S. cerevisiae*) show a higher accuracy with respect to the competitors, as well as a higher robustness to the presence of noise in the data, also when a huge amount of (possibly false positive) interactions is removed. Availability: <http://www.di.uniba.it/~gianvitopio/systems/inlocanda/>

Keywords Causality · Bioinformatics · Network Reconstruction · Link prediction

Editors: Takuya Kida, Takeaki Uno, Tetsuji Kuboyama, Akihiro Yamamoto.

✉ Gianvito Pio
gianvito.pio@uniba.it

Extended author information available on the last page of the article

1 Introduction

Recent studies in biology have been significantly supported by high throughput technologies and computational methods, which have led to an improved understanding of the working mechanisms in several organisms. Such mechanisms can be usually modeled through biological networks, which are able to easily describe the considered biological entities, as well as their relationships and interactions. On the basis of the phenomenon under study, different types of biological networks can be considered. The most prominent example is that of networks modeling the control of transcription into messenger RNAs or proteins (Atias and Sharan 2012; Penfold and Wild 2011). In these networks, called Gene-Regulatory Networks (GRNs), nodes represent molecular entities, such as transcription factors, proteins and metabolites, whereas edges represent interactions (i.e., the up/down regulation of gene expression levels), such as protein-protein and protein-DNA interactions.

The direct observation of the real structure of interaction networks requires expensive in-lab experiments, usually performed through the so-called epistasis analysis. Although in the literature we can find some computational approaches which support such an analysis (Zitnik and Zupan 2015), gene expression data are much easier to obtain, therefore most of the computational approaches proposed in the literature have focused on predicting the existence of interactions from gene expression data, mainly on the basis of link prediction methods. These approaches analyze the expression level of the genes under different conditions (e.g., in the presence of specific diseases or after a treatment with a specific drug) or, alternatively, under a single condition in different time instants. The expression levels observed for each gene are represented as a feature vector. A gene-gene matrix which defines a score of the interaction for each pair of genes is then built by pair-wisely computing a similarity, correlation or information-theory-based measure between the vectors associated to genes (Hempel et al. 2011). Finally, the existence of edges is inferred by imposing a threshold on the obtained score (see Fig. 1). The direction of the interactions can be inferred only if the considered measure is asymmetric.

However, except for those based on clustering (Pio et al. 2015), these methods generally assume independence among the interactions, i.e., they focus on each pair of genes separately, disregarding possible dependencies or indirect influences among them. This assumption leads to predict the false positive interactions, which are usually due to some causality phenomena: (i) common regulator genes (also referred to as *common cause* in the literature (Korb and Nicholson 2010)) or (ii) commonly regulated genes [also referred to as *common effect* in the literature (Korb and Nicholson 2010)]. In the first case (see Fig. 2a), the feature vector associated to a gene C , which exhibits a regulatory activity on two genes A and B , will

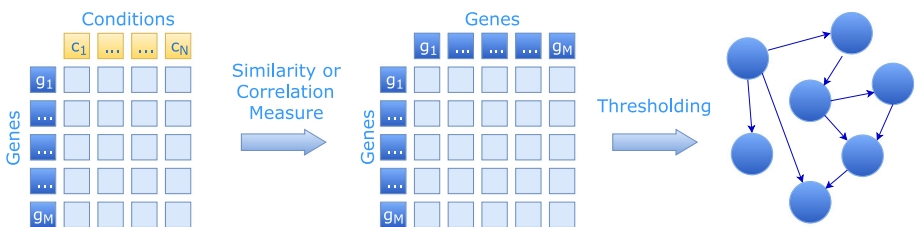


Fig. 1 Network reconstruction from expression data. On the left, a matrix of M genes, each associated to a vector containing the expression level measured under N different conditions. In the middle, the gene-gene matrix obtained by pair-wisely computing a similarity/correlation measure between the vectors. On the right, the reconstructed network obtained by imposing a threshold on the values of the gene-gene matrix

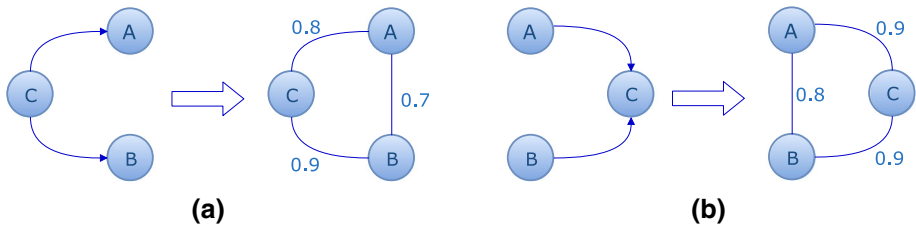


Fig. 2 Issues in the network reconstruction due to common cause (a) or common effect (b) phenomena. The direction of the interactions does not appear in the reconstructed networks if we consider the case of a symmetric similarity/correlation measure

presumably result in a similarity between *A* and *B*. In fact, even if there is no direct interaction between the genes *A* and *B*, since they are both similar to *C*, their feature vectors will also appear similar, therefore an edge between them could possibly be detected. Analogously, in the second case (see Fig. 2b), a gene *C*, which is regulated by two genes *A* and *B*, will presumably result in a similarity between *A* and *B*. Such issues are even more evident when data are affected by noise, since possible measurement errors lead to significantly increasing the number of false positives, due to (noisy) common causes or common effects, compromising the overall quality of the reconstruction.

The identification and removal of false positive interactions would be highly beneficial for domain experts who can concentrate further analyses and resources only on relevant interactions, disregarding those that are due to noise (Thattai and van Oudenaarden 2001) and those due to possible hidden common causes and hidden common effects (Lo et al. 2015). The general approach to solve these issues is based on post-processing the gene-gene matrix. The methods that operate in such a way are usually called *scoring schemes* (Hempel et al. 2011), where the idea is to improve the quality of the reconstructed network by analyzing large sets of genes simultaneously, in order to catch more global interaction activities and possibly reduce false positives, due to the presence of common cause and common effect phenomena.

One of the most popular scoring schemes is ARACNE (Margolin et al. 2006), which evaluates all the possible connected gene triplets and removes the weakest edge. ARACNE is limited to undirected networks and is not able to analyze indirect interactions which involve more than three genes. However, although the idea of removing the weakest edge (i.e., the edge with the lowest score) is very simple, the intuition of considering the score as an indication of the reliability of the interaction is reasonable and has been exploited by other works in the literature (e.g., Bošnački et al. 2012). In order to overcome this limitation, in Pio et al. (2017) we presented the method LOCANDA which, starting from a (possibly noisy) network reconstructed by any supervised/unsupervised reconstruction method, is able to remove false positive interactions on the basis of direct and indirect interactions of arbitrary length (i.e., involving, in a path of the regulatory network, an arbitrary number of genes). LOCANDA has its roots in methods for graph analysis and, in particular, in works for transitive reduction (Aho et al. 1972; Hsu 1975) but, contrary to existing methods, it does not require additional pre-processing steps to handle specific peculiarities of the network, such as weights on the edges, direction of edges and, especially, cycles.

However, although LOCANDA is able to identify common cause and common effect phenomena (see Fig. 2) involving an arbitrary number of genes, it has the important limitation that it does not consider the structural role of the nodes in the network. One prominent example where the structural role of the node is important is that of hub nodes, i.e., nodes influencing

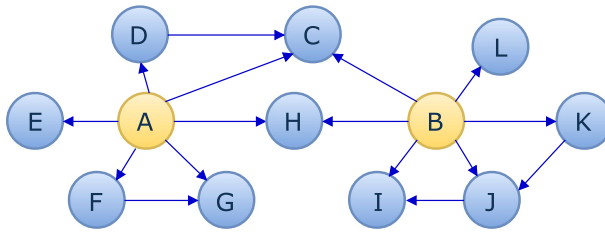


Fig. 3 An example of a network where it is possible to observe two node communities and a similar role (hub) performed by the nodes *A* and *B*

a large number of other nodes. Indeed, in the biological domain, hubs have shown to play an essential role in gene regulation and biological processes. The importance of hub nodes is clear when we consider transcription factors (TFs), i.e., proteins that bind to specific DNA sequences, since, in humans, approximately 10% of genes in the genome code for around 2600 TFs (Babu et al. 2004). The result is that human TFs account for most of the regulation activities in the human genome, especially during the development stage (Yu et al. 2017). Moreover, in the tumor genesis, hub nodes show a determinant role in the genetic networks of tumors (Li and Xie 2015; Selvanathan et al. 2015). This aspect has been recognized in several recent methods for the reconstruction of gene networks, which exploit information about hub genes, either explicitly provided as background knowledge (Yu et al. 2017), or identified by the method itself (Böck et al. 2012) as we do in the present paper. Despite these considerations, LOCANDA does not catch connectivity patterns generally observable in networks, such as node communities and common structural roles (Grover and Leskovec 2016). On the contrary, the analysis of indirect interactions should take into account possible node communities and common structural roles: if two nodes share the same community or have the same role in the network (e.g., they are both hub nodes), they should be treated in a similar way when solving the network reconstruction task (see Fig. 3).

Another limitation of LOCANDA is that it is not able to learn an inductive predictive model, i.e., it is not able to predict whether an unseen edge exists or not. In other words, it cannot make predictions involving unseen nodes or edges, but can only remove (possibly) false positive edges from a given network.

In this paper, we propose INLOCANDA (INductive LONGest-path CAusal Network Discovery Algorithm), which extends LOCANDA towards a more general approach for the reconstruction of Gene Regulatory Networks and aims at solving the limitations of LOCANDA. In particular, the methodological differences and extensions of INLOCANDA with respect to LOCANDA are:

- LOCANDA is only able to catch causality phenomena to remove possible false positive edges, thus it cannot predict the existence of unseen edges in the network; INLOCANDA reduces the network through the same approach followed by LOCANDA, but also exploits the reduced network to build an inductive predictive model able to decide whether an interaction between two genes exists or not, even if it has not been seen during the learning phase.
- INLOCANDA learns and exploits a feature representation from the reduced network that embeds the information on nodes and edges (through graph embedding), catching possible community structures and possible similar roles performed by nodes. This is particularly important in the case of Gene Regulatory Networks, where the role of hub nodes has been recognized in the literature (Böck et al. 2012; Yu et al. 2017).

Besides these methodological differences, this paper extends and revises the paper (Pio et al. 2017) as follows:

- we report a deeper analysis of related work, focusing on methods for the reduction of edges in networks, on approaches for the reconstruction of gene regulatory networks as well as on methods for graph embedding;
- we theoretically show the important property of *reachability equivalence* of the reduced network extracted by INLOCANDA;
- we conduct an analysis of the time complexity of INLOCANDA;
- we perform a broader set of experiments, including both synthetic and real datasets, aiming to evaluate the effectiveness of the proposed approach compared to state-of-the-art methods. Experiments have been conducted along different dimensions of analysis: the strategy adopted to learn the feature representation, the approach adopted to remove possible false positive edges and the supervised learner that builds the final predictive model.

The rest of the paper is organized as follows. In Sect. 2, we briefly discuss existing related work. In Sect. 3, we introduce our method for learning a model for the reconstruction of gene regulatory networks, providing details about the approach adopted for catching causality phenomena, as well as for the construction of the network embedding. In Sect. 4, we discuss the time complexity of our method, while in Sect. 5 we describe the experimental evaluation and comment on the obtained results. Finally, in Sect. 6, we draw some conclusions and outline possible future works.

2 Related work

Our work finds its roots in different research lines, i.e., (i) approaches for the reconstruction of Gene Regulatory Networks, (ii) general-purpose methods for graph reduction that analyze and exploit causal phenomena, (iii) approaches for causal analysis specifically tailored for biological networks, and (iv) methods for learning a compact feature representation (embedding) of nodes and edges. Therefore, in the following subsections, we briefly review previous work related to these areas.

2.1 Gene network reconstruction

In the literature, several experimental approaches based on in vitro experiments have been proposed for the identification of the structure of gene regulatory networks (Berger and Bulyk 2009; Bulyk 2005; Park 2009). However, these techniques are often technically and financially demanding (Penfold and Wild 2011). Therefore, most of the recent developments have focused on computational approaches for inferring the structure of gene regulatory networks from expression data, i.e., measurements of the response of the genes under different perturbation or stress conditions. In Emmert-Streib et al. (2012), Hecker et al. (2009), de Jong (2002), Markowitz and Spang (2007), it is possible to evaluate most of the data-driven approaches to infer the network structure. They are based on different principles, including relevance networks, clustering, probabilistic models, differential equations, Markov chains, Bayesian networks and random walk processes [see Lü and Zhou (2011) for additional details].

A broad evaluation and comparison of existing computational methods for gene network reconstruction has been performed in the context of the DREAM (Dialogue for Reverse

Engineering Assessments and Methods) series of challenges. Marbach et al. (2012) evaluated all the methods proposed in the fifth edition of the challenge and empirically proved that the combination of all the methods, by averaging ranks of predictions (“Wisdom of crowds”), leads to a more accurate reconstruction. Another solution for combining the output of several methods has been proposed in Hase et al. (2013), where predictions of each method are ranked according to their scores and finally combined by taking the k -th highest rank among all the considered methods, where k is an input parameter. Following this stream, Ceci et al. (2015) proposed the system GENERE, which exploits a machine learning solution to combine methods outcomes. In particular, it builds a (possibly stronger) predictive model by taking as input features the scores returned by several “base” algorithms for gene network reconstruction, resorting to a meta-learning solution (Vilalta and Drissi 2002). The authors solved several specific issues raised by the task at hand, through a multi-view learning solution able to work in presence of only positive examples.

However, these methods need a pre-defined threshold in the final stage of the reconstruction (see Fig. 1) to decide whether the edge should be predicted or not. Moreover, they cannot take into account the presence of possible common cause and common effect phenomena, as introduced in Sect. 1.

2.2 Analysis of causality for graph reduction

In the literature we can find several approaches which catch and exploit causality phenomena. Here we concentrate on the problem of graph (transitive) reduction by taking causality into account. A general framework for the identification of causal edges between variables is described in Pearl (2000), which consists in (i) the identification of correlations between variables, which suggest possible (undirected) edges; (ii) the analysis of partial correlations, which can be exploited to remove indirect relationships; (iii) the exploitation of some assumptions on the network structure (e.g., acyclicity), which can suggest the direction of edges. It is noteworthy that such assumptions can be violated in specific domains (e.g., biology), leading to an inaccurate reconstruction.

An example of the application of such a framework can be found in algorithms for learning the structure of Bayesian networks (Korb and Nicholson 2010), which identify causalities between variables by analyzing the *d-separation* among them, which is based on common cause and common effect phenomena (see Sect. 1).

Other approaches exploit the concept of causality to identify a transitive reduction of a graph (Aho et al. 1972; Hsu 1975). These methods analyze a graph and produce a reduced version containing a subset of edges, which guarantees the conveyance of the same information of the original graph, in terms of reachability. This means that, analogously to the method proposed in this paper, these approaches aim at removing edges that can be considered the result of an indirect relationship. For example, the method proposed in Aho et al. (1972) finds a transitive reduction G' of the initial graph G , where G' has a directed path from any vertex u to any vertex v if and only if G has a directed path from vertex u to vertex v and there is no graph with such a property having fewer edges than G' . In other words, the obtained graph G' is the smallest graph (in terms of edges) such that given any pair of nodes $\langle u, v \rangle$, if v is (respectively, is not) reachable from u in the initial graph G , then v is (respectively, is not) reachable in the reduced graph G' . Although it is based on the same principles of INLOCANDA, this approach requires the identification of an equivalent acyclic graph before performing the analysis and is limited to unweighted graphs. Therefore, it cannot exploit information on the reliability (score) commonly associated to each edge in

biological networks. Moreover, this approach aims at achieving the minimality of the resulting graph, which is not a desirable property in the case of gene regulatory networks. Indeed, it is very likely that a redundant graph (in terms of reachability) may naturally represent a gene regulatory network, since a group of genes may regulate each other. We will exploit this concept in Sect. 3.1, where we will propose a constraint to decide whether to remove an edge or not.

Analogously, Hsu (1975) propose the identification of a Minimal Equivalent Graph (MEG), whose definition is the same as the transitive reduction proposed in Aho et al. (1972). The method consists of several steps: (i) the identification of strongly connected components; (ii) the removal of cycles from each component; (iii) the identification of the minimal equivalent graph for each component and (iv) the reintroduction of the edges removed in step (i). Even if it is more sophisticated, this approach suffers from the same limitations described for Aho et al. (1972).

2.3 Analysis of causality in biological networks

Focusing on biological networks, in the literature several approaches have been proposed to consider specific issues raised by such an application domain, by exploiting specific causality phenomena. In particular, causality can be exploited to infer the directionality of the interactions from the time series of gene expression data (Hempel et al. 2011). In this case, the regulator gene (the cause), by definition, should act before the regulated gene (the effect). Therefore, a common strategy consists in computing the similarity between two genes u and v , by performing a progressive shifting forward in time of the time-series associated to the first gene u . If the similarity increases, then it is possible to conclude that u acts before v , therefore, u regulates v . More sophisticated approaches exploit Granger causality (Itani et al. 2008) or hidden (i.e., unobserved, latent) common causes and common effects (Lo et al. 2015). All these methods, however, are applicable only in the analysis of time-series data, while they cannot be applied when each gene is associated with a vector representing its expression values in different (steady) conditions. Margolin et al. (2006) propose the (previously mentioned) method ARACNE, which exploits causality phenomena to identify and remove indirect relationships. ARACNE acts as a post-processing phase of the network reconstruction, aiming at removing interactions considered to be an indirect effect of other interactions. This is performed by assuming that, given a triplet of connected genes, the weakest interaction (i.e., that with the lowest score) is a false positive edge due to common cause or common effect phenomena. As already stated, although it is based on principles similar to those exploited by our method, this approach is limited to indirect interactions involving only three genes, thus it cannot identify indirect interactions of arbitrary length.

In a more recent work (Bošnački et al. 2012), the authors propose a method for the identification of the transitive reduction of biological networks. This method is able to analyze both unweighted networks and possibly cyclic weighted networks. In the latter case, however, following the approach adopted in Pinna et al. (2010), pre-processing of the network is required in order to make it acyclic. In detail, the method (i) identifies and shrinks the strongly connected components into single nodes, (ii) applies the reduction on the resulting acyclic graph, and (iii) re-expands the components. It is noteworthy that this procedure assumes that the genes within each component are fully connected and do not perform any reduction within each component, since the results would strongly depend on the order of the analysis. Moreover, this procedure assumes that the graph resulting from step (i) is acyclic, i.e., there is no cycle among the components.

2.4 Network embedding

In the literature a lot of effort has been made in the design of methods to construct relevant node features from the network structure. Several approaches generate node features on the basis of feature extraction techniques, that usually exploit network properties and a few manually defined features (Gallagher and Eliassi-Rad 2010; Henderson et al. 2011). On the contrary, Grover and Leskovec (2016) consider the feature extraction task to be a representation learning problem that does not require any hand-engineered feature.

Unsupervised methods usually analyze the spectral properties of different matrix representations of networks. According to this perspective, these methods are equivalent to matrix dimensionality reduction techniques. Some examples are (Belkin and Niyogi 2002; Roweis and Saul 2000; Tenenbaum et al. 2000), that, however, suffer from both computational and effectiveness viewpoints. The computational inefficiency is mainly due to the fact that they are generally based on eigendecomposition of a data matrix. On the other hand, the quality of the result is affected by the fact that they do not take into account multiple observable patterns in the network, that is, they are not able to simultaneously consider community structures and common roles of nodes (Grover and Leskovec 2016).

Some recent works exploit approaches initially proposed for natural language processing. In particular, inspired by the Skip-gram model (Mikolov et al. 2013), these approaches represent a network as a document and a possible sequence of nodes as a sequence of words. The aim is to obtain a new representation for words/nodes where features are numeric, such that the new feature space is able to preserve the similarity of the meanings of words appearing in similar contexts, or, in the case of networks, the similarity of roles of nodes showing a similar structure of their neighborhood. In the specific case of networks, the identification of the neighborhood of each node is guided by a search strategy (mainly based on the classical Breadth-First and Depth-First search strategies, or on a combination thereof) and by some approaches aiming at reducing the complexity of the task, such as an upper threshold on the number of neighbours to consider.

Inspired by the work proposed in Grover and Leskovec (2016), we adopt a strategy that is not affected by the specific search strategy [as happens in Perozzi et al. (2014), Tang et al. (2015)], which is also able to catch community structures and common roles performed by nodes in the network. Moreover, we provide a representation for “directed” edges. This is different from what is done in Grover and Leskovec (2016), where the main goal is to learn a representation for nodes, while the representation for edges is obtained by combining pairs of vectors (associated with pairs of nodes) through symmetric binary operators. The main drawback is that it is not possible to provide a representation that takes into account the direction of the edge, which is fundamental in our case.

3 The method INLOCANDA

In this section, we describe in detail the INLOCANDA method. We remember that it takes as input a (possibly noisy) Gene Regulatory Network, which has been reconstructed by any supervised/unsupervised method from gene expression data. The network consists of nodes and directed edges, where the nodes represent genes and directed edges correspond to regulatory activities (i.e., up/down regulation of gene expression levels). Most of the existing reconstruction methods also provide a weight for each edge, i.e., a score which represents the degree of certainty or the confidence provided by the reconstruction method

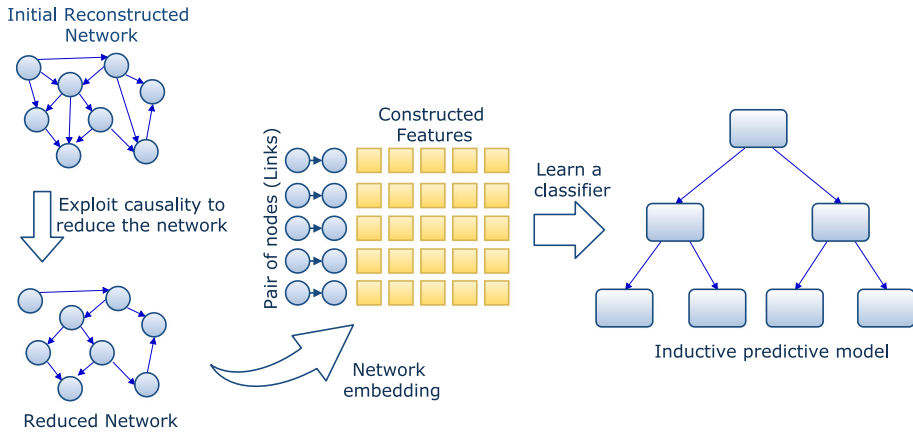


Fig. 4 General workflow of the method INLOCANDA

about the corresponding inferred interaction. We consider the most general case in which the reconstruction method provides a weight in $[0, 1]$ for each edge between any pair of nodes. More formally, let:

- V be the set of genes, i.e., the nodes in the reconstructed network.
- $E \subseteq V \times V \times \mathbb{R}$ be the set of interactions in the reconstructed network, i.e., the weighted edges in the form $\langle \text{source_node}, \text{destination_node}, \text{edge_weight} \rangle$ (henceforth, for every edge $\langle u, v, w \rangle \in E$ involving the nodes u and v we denote its weight w as $w(u, v)$);
- $g: E \rightarrow \{\text{true}, \text{false}\}$ be an ideal function which returns *true* if $e \in E$ is a known existing interaction in the gene regulatory network, *false* otherwise.

The task to be solved can then be formalized as:

Input: the reconstructed network $G = \langle V, E \rangle$

Solution: find an inductive predictive model, in the form of a prediction function $f: V \times V \rightarrow \{\text{true}, \text{false}\}$, which approximates the ideal function g .

INLOCANDA consists of three stages, which are shown in Fig. 4:

1. the reduction of the original (possibly noisy) reconstructed network, i.e., the removal of possible false positive edges, by catching common cause and common effect phenomena in the network;
2. the learning of a feature representation from the reduced network, able to represent the information on single nodes and pairs of nodes (edges), by catching possible community structures and possible similar roles performed by nodes;
3. the learning of an inductive predictive model from edges represented through the identified feature representation, able to predict whether a (possibly unseen) edge between two (possibly previously unseen) nodes exists or not.

Stage 3) can be solved by any learning algorithm able to build a classification model. To solve the specific task of Gene Network Reconstruction we adopt the predictive clustering tree (PCT) framework. The PCT framework views a decision tree as a hierarchy of clusters, where the top-node corresponds to one cluster containing all the data. This cluster is recursively partitioned into smaller clusters, while moving down the tree. The PCT framework is implemented in CLUS (Blockeel et al. 1998), available at sourceforge.net/projects/clus. The motivations behind the adoption of the PCT framework are

the following: (i) the clusters constructed by CLUS are able to group together pairs of nodes, where the first node and the second node of each pair are considered independent, thus considering the direction of the edge; (ii) PCT induction, as in classical induction of decision trees, performs some form of embedded feature selection (i.e. the most important features are selected in the tests at the highest levels of the tree). This is important in our case, since features extracted at stage 2) can be highly redundant; (iii) it is effective in handling highly imbalanced datasets, as it is our case (the amount of positive examples is much lower with respect to the amount of negative examples); (iv) PCT induction is very efficient. This aspect will be clarified in Sect. 4, where we discuss the time complexity of INLOCANDA. Obviously, many other classification systems exhibit the same characteristics and can be used. At this respect, in our experimental evaluation and in the “Appendixes”, we also show the results obtained with three other classifiers, in order to prove the effectiveness of the approach we adopt to reduce the false positive edges, when different classifiers, possibly based on different underlying models, are adopted.

In the following subsections, we describe the first two stages in details.

3.1 Exploiting causality to reduce the reconstructed network

In this section, we describe the approach adopted by INLOCANDA for the identification and removal of false positive edges from a (noisy) reconstructed gene network. As already introduced in Sect. 1, our approach is based on the concepts of common cause and common effect. We remember that INLOCANDA is not limited to the simple cases depicted in Fig. 2, and shows the following distinguishing characteristics: (i) unlike classical methods for the identification of a transitive reduction of networks (Aho et al. 1972; Hsu 1975), it is able to work on weighted networks, which is relevant when dealing with reconstructed biological networks, where edges are associated to a score/reliability; (ii) unlike Margolin et al. (2006), it is able to work on directed networks, which (if available) becomes important in order to correctly consider causality phenomena; (iii) similar to Bošnački et al. (2012) and unlike (Margolin et al. 2006), it is able to catch indirect interactions of arbitrary length, by comparing the reliability of direct edges to that of identified indirect relationships; (iv) contrary to Aho et al. (1972), Bošnački et al. (2012) and Hsu (1975) it is able to directly work on possibly cyclic networks, by guaranteeing the same result independently of the order of analysis.

We catch indirect interactions among genes by identifying and analyzing possible paths connecting them in the network. Indeed, one path in a Gene Regulatory Network can represent a sequence of regulatory activities that can let reconstruction methods identify also nonexistent (i.e., false positive) direct interactions among the genes involved in the path. Therefore, we compare the reliability observed on the paths with that observed on direct edges, in order to decide whether to remove direct edges. Since our method is based on the concepts of “path”, “reachability”, “reliability”, we formally define them:

- A path P between two nodes u (source node) and v (destination node) is a sequence $[v_1, \dots, v_k]$ such that $u = v_1$, $v = v_k$ and $\langle v_i, v_{i+1}, \cdot \rangle \in E$ for every $i(1 \leq i \leq k - 1)$;
- a node u is *reachable* from a node v if there exists a path between u and v .
- given a path $P = [v_1, \dots, v_k]$, its reliability is measured by a function f (as introduced below) such that $f(P)$ is the value in $[0, 1]$ obtained by evaluating the weight $w(v_i, v_{i+1})$ of every edge $\langle v_i, v_{i+1}, w \rangle$ in P .

Based on these definitions, we construct a reduced network $\tilde{G} = \langle V, \tilde{E} \rangle$ by removing each edge $\langle u, v, w \rangle$ in G if it is less reliable than a path P between u to v , i.e., if $w < f(P)$.

Note that, contrary to Aho et al. (1972) and Hsu (1975), we do not require the minimality of the number of edges in the reduced network, since we are not interested in *pure* transitive reduction, but in removing possible false positive edges, identified during the reconstruction of the network. Indeed, we recall that enforcing the minimality of the resulting graph may lead to lose significant interactions, since, in the specific case of gene regulatory networks, it is very likely that a redundant graph (in terms of reachability) may naturally represent a gene regulatory network. Therefore, the fact that the information conveyed by an edge can be represented by a sequence of nodes (a path) is not a sufficient condition to consider the edge as a false positive, due to the presence of common cause or common effect phenomena. For this reason, we introduce an additional constraint to decide whether to remove an edge. In particular, we remove an edge in G only if its reliability appears lower than the reliability of a path $P = [v_1, \dots, v_k]$ in G , measured by $f(P)$, and it does not appear within any path used by INLOCANDA during the analysis of other edges.

In this work, we take into account different measures to estimate the reliability of the path. In particular, if $w(v_i, v_j)$ is the weight associated to the edge between v_i and v_j , we consider the following measures for a path $[v_1, \dots, v_k]$ in G :

- *Minimum (Min)* which is the lowest edge weight in the path, following the principle of the “weakest link in the chain”. Formally:

$$f([v_1, \dots, v_k]) = \min_{i=1,2,\dots,k-1} w(v_i, v_{i+1}).$$

- *Product (Prod)* which is the product of the edge weights involved in the path. This approach is motivated by the common strategy adopted for the combination of probabilities of (naively independent) events. Formally:

$$f([v_1, \dots, v_k]) = \prod_{i=1}^{k-1} w(v_i, v_{i+1}).$$

- *Average (Avg)* which is the average weights of the edges involved in the path. Formally:

$$f([v_1, \dots, v_k]) = \frac{1}{k} \cdot \sum_{i=1}^{k-1} w(v_i, v_{i+1}).$$

- *Weighted Average (WAvg)* which is the average of the edge weights involved in the path, linearly weighted on the basis of their closeness to the source node. This approach can be motivated by the assumption that the influence of the source node on the other nodes in the path fades linearly on the basis of their distance. Formally:

$$f([v_1, \dots, v_k]) = \frac{1}{\sum_{i=1}^{k-1} \frac{1}{i}} \cdot \sum_{i=1}^{k-1} \left[\frac{1}{i} \cdot w(v_i, v_{i+1}) \right].$$

Before formally describing the INLOCANDA algorithm, we briefly describe intuitively the followed approach. In particular, for each gene, we aim at identifying the genes that erroneously appear as regulated by it in the reconstructed network, due to the presence of an indirect regulation activity involving other genes in the network. At this aim, starting from

each node (that we call *source node*), we explore the network, through a depth-first and best-first strategy, in order to identify the reachable nodes and representative paths (i.e., the most reliable ones, according to the edge weights). For each reachable node, if the path from the source node appears to be more reliable than the direct edge, and such a direct edge is not exploited in any other path to reach other nodes, we remove the direct edge. As we already emphasized in the introduction, these choices are in contrast with the minimality property achieved by classical graph reduction techniques (Aho et al. 1972; Hsu 1975), since, in the specific case of gene regulatory networks, it would lead to lose relevant regulatory activities involved in other indirect interactions.

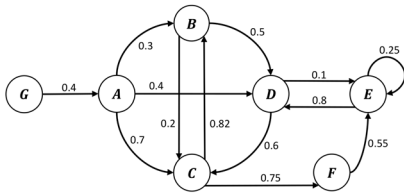
The pseudo-code of the first stage of INLOCANDA, which follows in the footsteps of LOCANDA, is reported in Algorithm 1. We also report a running example in Fig. 5. Before describing it, we recall that the method is able to analyze both undirected and directed networks, weighted according to a score representing the reliability of the interaction (computed by any method for network reconstruction). Here we assume we are working with a weighted directed network (the most general case), since an unweighted network can always be mapped into a directed network by introducing an edge for each direction.

The first step consists in the removal of self-edges (line 2), since some methods for network reconstruction identify them erroneously. Although self-regulation activities are possible in biology, in reconstructed networks such edges are due to errors in the computation of similarity/correlation measures on the vector associated to a single gene. In our example, the self-edge on the node *E* (Fig. 5b) is removed, leading to the network in Fig. 5c. Then the algorithm analyzes each node (that we call *source node*), aiming at identifying all the reachable nodes and a path to reach them. We note that the visit of the network is performed according to a depth-first and best-first search strategy, on the basis of the reliability of the edges. The algorithm works in a greedy fashion, since an exhaustive exploration of all the possible paths would lead to an exponential time complexity. When there are several edges to follow, we consider the path that locally (i.e., by observing the neighborhood) appears the most reliable. This aspect is clarified later.

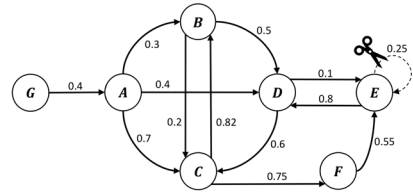
INLOCANDA exploits three data structures: the set of visited nodes (*visited*), the current sequence of nodes (*path*) and a *stack*, according to which the nodes are explored. Moreover, it exploits a structure (*RT*) similar to the routing table used by routing algorithms, which keeps information on the nodes reachable from the source node. In particular, for each reachable node, it stores:

- The **next-hop**, i.e., the node adjacent to the source node that we need to follow to reach it, according to the current path.
- The **path score** associated to the current path. On the basis of the path score, the algorithm chooses the optimal path to keep: INLOCANDA will prefer a new path with respect to a previously identified path if this value is higher.
- The **path reliability**, i.e., the reliability associated to the current path according to $f(\cdot)$, that will be exploited to remove edges.

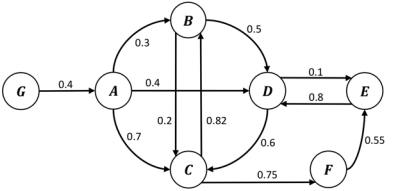
We note that the path score (for the choice of the optimal path) and the path reliability (for the estimation of the reliability of the path) are intentionally kept as two separate measures. This because, generally, they could not be based on the same assumptions. In particular, while, as previously described, the path reliability can be based on several measures, the path score corresponds to the sum of edges in the path, since, combined with the adopted strategy for the choice of the edge to follow (i.e., the highest), it leads to the identification of long and reliable paths.



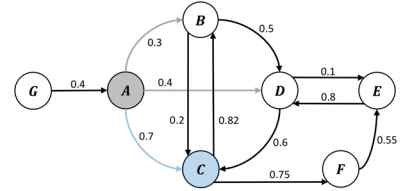
(a) The initial reconstructed network.



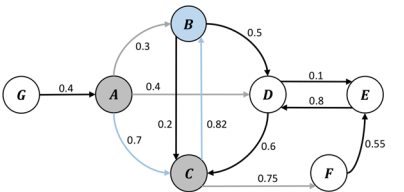
(b) The self edge on node E.



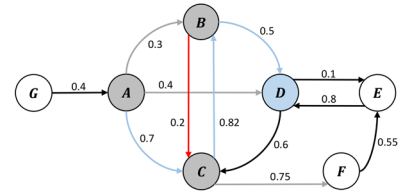
(c) The self edge on node E removed.



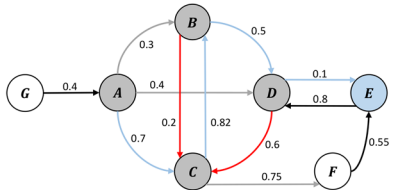
(d) Node A expanded. Node C popped from the stack (since the weight of A→C was the highest).



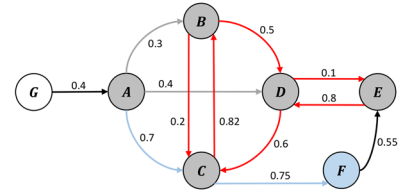
(e) Node C expanded. Node B popped from the stack (since the weight of C→B was the highest).



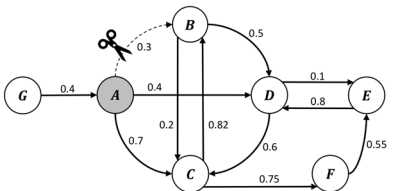
(f) Node B expanded. Node D popped from the stack (it was the only pushed node, since C had been already visited).



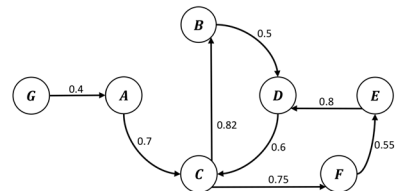
(g) Node D expanded. Node E popped from the stack (it was the only pushed node, since C had been already visited).



(h) Node E expanded. No node pushed into stack, since D had been already visited. It steps back to analyze F from C.



(i) Removal of the edge A→B.



(j) Reduced network, after the analysis of all the nodes ($f(\cdot) = \text{Minimum}$).

Fig. 5 An example of the execution of INLOCANDA and the analysis of the *source node* A. Grey nodes: already expanded; blue node: the current node to analyze, extracted from the stack; black edges: not seen yet; grey edges: already seen, but still not followed; blue edges: belonging to the current path; red edges: will not be followed, since this would lead to already expanded nodes; black-dashed edges: to be removed (Color figure online)

The analysis of a source node is performed as follows. First, the data structures are initialized (line 4), by considering the source node as already expanded and by adding it to the current path. Second, we analyze all its adjacent nodes, i.e., we push them into the stack, put in ascending order according to the edge weight, and initialize the routing table by setting them as their next-hop (lines 5–7). Then, the main part of the algorithm (lines 8–28) iterates until the stack still has some nodes to analyze. In particular, INLOCANDA extracts a node (*current_node*) from the stack (see Fig. 5d), marks it as visited (lines 9–10), and computes the score associated with the current path to reach the *current_node* from the source (lines 11–13). If the current path is the first identified path to reach *current_node* or it has higher score than a previously identified path in the routing table, INLOCANDA updates the routing table (lines 14–16). Then the algorithm expands the current node, by pushing its adjacent nodes into the stack in ascending order with respect to the edge weight, if not already visited (lines 17–19).

If at least one node is pushed (see Fig. 5e–g), the current path is updated by adding *current_node* (lines 20–22), otherwise (see Fig. 5h) the algorithm steps back, until it can find an existing edge between the last node in the path and the next node in the stack (lines 23–28). In both cases, the path and its score are updated incrementally (lines 22 and 26–28).

When there are no more nodes in the stack, INLOCANDA removes all the direct edges, such that the properties previously described are satisfied. In particular, it removes an edge between the source node u and its adjacent v , if v is never used as next-hop to reach other nodes and if the path identified to reach v from u appears more reliable than the direct edge (lines 29–32). The algorithm then proceeds with the next source node. It is noteworthy that the removed edges will never be considered again by the algorithm. This can be done without any risk of losing relevant paths, since those edges would never be considered in any case, even analyzing the nodes of the networks in a different order. For example, the removed edge between A and B in Fig. 5i would not be followed in any case during the analysis of the node G as the source node. Therefore, the order of analysis of source nodes does not affect the resulting reduced network.

A final remark regards the concept of reachability that we introduced at the beginning of this section. Indeed, even if INLOCANDA does not require the minimality of the reduced network, we still guarantee that the reachability of nodes is preserved, i.e., no (indirect) interactions are lost after the reduction of the network. In the following, we state and prove the reachability equivalence between the original network and the network produced by INLOCANDA.

Property 1 Node reachability in the reduced network $\tilde{G} = \langle V, \tilde{E} \rangle$ produced by INLOCANDA is equivalent to that of the initial network $G = \langle V, E \rangle$.

More formally, for each pair $(u, v) \in V \times V$ of nodes, there exists a path $[n_1, n_2, \dots, n_k]$ in (V, E) such that $u = n_1$, $v = n_k$ and $\langle n_i, n_{i+1}, \cdot \rangle \in E$ for every $i (1 \leq i \leq k - 1)$ if and only if there exists a path $[m_1, m_2, \dots, m_r]$ in (V, \tilde{E}) such that $u = m_1$, $v = m_r$ and $\langle m_j, m_{j+1}, \cdot \rangle \in \tilde{E}$ for every $j (1 \leq j \leq r - 1)$.

Proof Since $\tilde{E} \subseteq E$, the if-direction is obvious. Then, we show the only-if direction. Assume that the node v can be reached by the node u in G through just one path $[n_1, n_2, \dots, n_k]$, such that $n_1 = u$ and $n_k = v$, but v cannot be reached from u in \tilde{G} . Since we assume that the path is unique in G , this means that INLOCANDA has removed at least one of the edges in the path $[n_1, n_2, \dots, n_k]$, i.e., there exists at least one edge $\langle n_h, n_{h+1}, w_h \rangle \in E$ s.t. $\langle n_h, n_{h+1}, w_h \rangle \notin \tilde{E}$. However, according to the algorithm followed by INLOCANDA (see

Algorithm 1: Pseudo-code of the first stage of INLOCANDA.

```

Data:
·  $V$ : the set of genes (nodes in the network)
·  $E \in V \times V \times \mathbb{R}$ : the set of interactions (edges in the network), represented as
  (source_node, destination_node, edge_weight)
·  $f(\cdot)$ : the measure for the reliability of a path

Result:
·  $\tilde{E}$ : the updated (reduced) set of interactions

1 begin
2    $\tilde{E} \leftarrow E \setminus E.getSelfEdges()$ ;
3   foreach  $src \in V$  do
4     visited  $\leftarrow \{src\}$ ; path  $\leftarrow [src]$ ; path_score  $\leftarrow 0$ ; stack  $\leftarrow []$ ; RT  $\leftarrow []$ ;
5     /* Initialize the routing table for adjacents of src */
6     foreach ( $src, adj, w$ )  $\in \tilde{E}$  in ascending order w.r.t. w do
7       RT.update(adj, adj, w, f(src,adj));
8       stack.push(adj);
9
10    while stack is not empty do
11      current_node  $\leftarrow$  stack.pop();
12      visited  $\leftarrow$  visited  $\cup$  {current_node};
13      edge_weight  $\leftarrow$   $\tilde{E}.getEdgeWeight(path.getLast(), current_node)$ ;
14      old_path_score  $\leftarrow$  RT.getPathScore(current_node);
15      new_path_score  $\leftarrow$  path_score + edge_weight;
16      /* Update the RT if the route does not exist or if the new path has a
17       higher score than the previous path */
18      if old_path_score = null or old_path_score < new_path_score then
19        next_hop  $\leftarrow$  path.getFirst();
20        RT.update(current_node, next_hop, new_path_score, f(path));
21
22      /* Push non-visited adjacent nodes of the current node into the stack,
23       ordered by the edge weight */
24      foreach (current_node, adj, w)  $\in \tilde{E}$  in ascending order w.r.t. w do
25        if adj  $\notin$  visited then
26          stack.push(adj);
27
28      /* Update the current path */
29      if some nodes were added to stack then
30        path.add(current_node);
31        path_score  $\leftarrow$  new_path_score;
32
33      else if stack is not empty then
34        next  $\leftarrow$  stack.top();
35        while (path.getLast(), next)  $\notin \tilde{E}$  do
36          last  $\leftarrow$  path.getLast();
37          path.removeLast();
38          path_score  $\leftarrow$  path_score -  $\tilde{E}.getEdgeWeight(path.getLast(), last)$ ;
39
40      /* Remove a direct edge if it is not used to reach other nodes and it's
41       less reliable than the indirect edge (path) */
42      all_next_hops  $\leftarrow$  RT.getAllNextHops();
43      foreach ( $src, adj, w$ )  $\in \tilde{E}$  do
44        if adj  $\notin$  all_next_hops and w < RT.getPathReliability(adj) then
45           $\tilde{E} \leftarrow \tilde{E} \setminus \{(src, adj, w)\}$ ;
46
47  return  $\tilde{E}$ ;

```

Algorithm 1, line 31), the edge between n_h and n_{h+1} would have been removed only if there had been an alternative between n_h and n_{h+1} , with higher reliability. Therefore, one of these cases may have occurred:

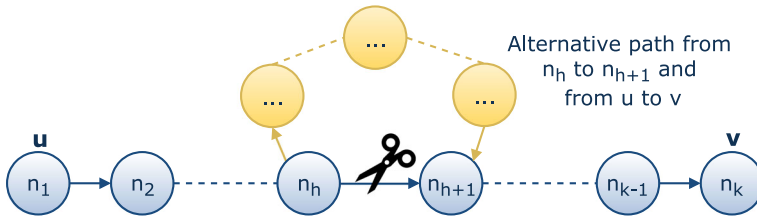


Fig. 6 An example showing a specific case to prove the reachability equivalence. An alternative path between n_h and n_{h+1} implicitly defines an alternative path between u and v

- The path between u and v is not unique in G . In this case, even if INLOCANDA removes the edge between n_h and n_{h+1} , the reachability of v from u is still guaranteed in \tilde{G} by the alternative paths.
- The path between u and v is unique in G , but INLOCANDA does not remove the edge between n_h and n_{h+1} (contradiction reached). Indeed, if there had been an alternative path between n_h and n_{h+1} leading to the removal of the direct edge between n_h and n_{h+1} , it would have also defined an alternative path between u and v (see Fig. 6). \square

3.2 Learning a vector representation for edges

Once we have analyzed the initial reconstructed network and removed the set of possible false positive edges, we aim at learning an inductive predictive model that it is able to decide whether an edge exists or not. Therefore, our learning examples (i.e., the units of analysis) will be the edges, i.e., interactions between pairs of nodes, that need to be represented by a set of features.

In this section, we describe the strategy adopted to learn a feature representation for each edge of the network. As we outlined in Sect. 2.4, in the literature there are several approaches for learning a feature representation for network. In this work, inspired by the work in Grover and Leskovec (2016), we aim at fully exploiting both community structures and possible common roles performed by different nodes in the network.

The basic idea in Grover and Leskovec (2016) is to identify a feature vector for each node in the network. In this paper, given the reduced network $\tilde{G} = \langle V, \tilde{E} \rangle$, this idea corresponds to learning a function that maps each node (and its neighbor) to a feature vector, i.e., $g : V \rightarrow \mathbb{R}^d$, such that nodes with the same neighborhood are “close” in the feature space, while nodes with different neighborhood are “far” in the feature space, according to a search strategy of the neighborhood.¹

However, we need to explicitly represent examples in terms of edges. In Grover and Leskovec (2016), given two nodes u and v , the authors proposed different binary operators that can be applied over the corresponding feature vectors $g(u)$ and $g(v)$, in order to obtain a feature vector for the edge between u and v . In particular, they propose the average, the Hadamard operator, the Weighted-L1 and the Weighted-L2, that lead to feature vectors in the same feature space used to represent single nodes. However, all the proposed operators are symmetric, i.e., they do not take into account the direction of the edge. Therefore, we adopt a different strategy, i.e., we concatenate the feature vectors associated to the nodes u and v . This means that, if nodes are represented in \mathbb{R}^d , then edges are represented in \mathbb{R}^{2d} . Formally,

¹ A formal definition of neighborhood will be introduced later.

the idea is to learn a function g' according to the classical maximum likelihood optimization problem, on the basis of the Skip-gram strategy for networks (Perozzi et al. 2014):

$$g' : V \times V \rightarrow R^{2d}, \quad (1)$$

where $g'(u, v) = g(u) \frown g(v)$ and \frown represents the concatenation operator.

Specifically, let u be a node, $N(u)$ be its neighborhood of fixed size k according to a search strategy, the goal is to find the function $g(u)$ maximizing the log-likelihood of observing $N(u)$ as follows:

$$\max_{g(\cdot)} \sum_{u \in V} \log P(N(u) | g(u)). \quad (2)$$

Multiple search strategies can be adopted to identify the neighborhood of nodes. The classical breadth-first and depth-first strategies lead to focusing either on structural similarities (similar roles performed by nodes) or on communities. In order to catch both the aspects, we adopt the strategy proposed by Grover and Leskovec (2016), where the authors rely on an approach based on random walks. In particular, they propose the adoption of a biased 2nd-order random walk, which takes into account the edge weights, combined with a factor that depends on two parameters: p , that controls the likelihood of immediately revisiting a node in the walk, and q , that controls the likelihood of keeping the search local. More formally, assuming that the random walk traverses the edge $\langle u', u'', w' \rangle$, the transition probability from u'' to another node u''' , via the edge $\langle u'', u''', w'' \rangle$, is computed as $\pi_{u'u'''} = \alpha_{pq}(u', u''') \cdot w''$, where:

$$\alpha_{pq}(u', u''') = \begin{cases} \frac{1}{p} & \text{if } d_{u'u'''} = 0, \text{ that is } u' = u''' \\ 1 & \text{if } d_{u'u'''} = 1 \\ \frac{1}{q} & \text{if } d_{u'u'''} = 2 \end{cases} \quad (3)$$

and $d_{u'u''}$ is the shortest path distance between u' and u''' (see Fig. 7 for a graphical description of the three cases).

According to such probabilities, given a source node u , we perform a random walk of length l . If n_i the i -th node in the walk (where $n_0 = u$), nodes are traversed according to the following distribution:

$$P(n_i = u'' | n_{i-1} = u') = \begin{cases} \pi_{u'u''}/C & \text{if } \langle u', u'', w' \rangle \in \tilde{E} \\ 0 & \text{otherwise,} \end{cases} \quad (4)$$

where C is a normalization constant.

4 Time complexity analysis

In order to show the time complexity of INLOCANDA, we analyze the three stages introduced in Sect. 3, separately. For Stage 1 (i.e., reduction of false positive edges through the exploitation of causality phenomena), we need to consider each node as a source node. For each node, we search for possible paths to reach all the other nodes in the network. However, we recall that each edge can be traversed only once, leading to a total of $|E|$ possible steps for each source node. Therefore, in the worst case, this stage requires $\mathcal{O}(|V| \cdot |E|)$ steps. We emphasize that this is truly the worst case, since all the edges in E are not always traversed for each source node in V and since INLOCANDA immediately removes edges that are considered to be false positive interactions as soon as it finishes the analysis of each source node (see lines 29-32 of Algorithm 1).

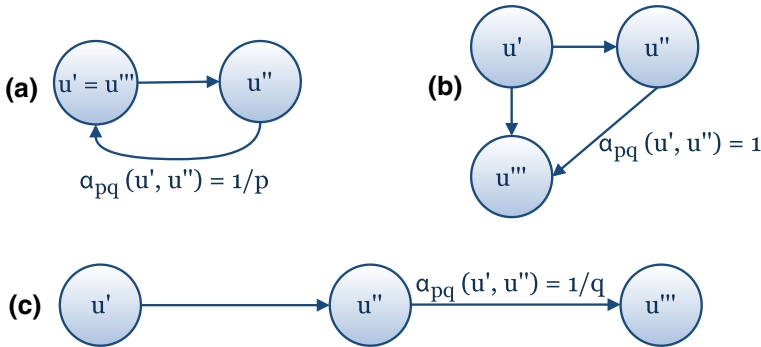


Fig. 7 Examples of 2nd-order random walks, according to the shortest distance from the initial node u' . **a** Node u''' corresponds to the initial node u' : the probability of revisiting u' is governed by the parameter p . **b** Node u''' is another adjacent node of u' (breadth-first visit). The probability of visiting it is only affected by the edge weight (i.e., $\alpha_{pq}(u', u''') = 1$). **c** Node u''' is reachable only after two steps from u' : the probability of performing the visit in depth is governed by the parameter q

Stage 2 (i.e., the learning of a vector representation for edges) exploits random walks that are generally computationally efficient. Moreover, by imposing graph connectivity in the sampling process (Grover and Leskovec 2016), a strong optimization can be obtained by reusing samples across different (linked) source nodes. Indeed, if k is the number of samples to generate per node, and l is the length of the walk, by simulating a random walk of length $l > k$, it is possible to generate k samples for $l - k$ nodes, altogether. This means that the complexity for one node is $O\left(\frac{l}{k \cdot (l - k)}\right)$, leading to the overall complexity $O\left(|V| \cdot \frac{l}{k \cdot (l - k)}\right) = O(|V|)$.

The time complexity of Stage 3 actually depends on the adopted algorithm. Considering CLUS as the classifier, we know that its complexity is $O(2d \cdot (|E| \log |E|))$, that is, linear on the number of descriptive attributes and $n \log n$ on the number of training instances (Blockeel et al. 1998). Hence, we have:

$$O(|V| \cdot |E|) + O(|V|) + O(d \cdot |E| \log |E|) = O(|V| \cdot |E|) + O(d \cdot |E| \log |E|) \tag{5}$$

By assuming that $|V| \geq (d \cdot \log |E|)$, we have the final computational complexity of INLOCANDA as $O(|V| \cdot |E|)$.

5 Experiments

In this section, we describe our experimental evaluation. In particular, in Sect. 5.1 we describe the considered datasets; in Sect. 5.2 we provide details about the experimental setting; in Sect. 5.3 we perform an analysis of the contribution provided by the proposed strategy for the construction of the feature space, in order to show that catching the structural roles of nodes can be beneficial in the specific task of gene network reconstruction; finally, in Sect. 5.4, we evaluate the quality of the reconstructed networks, when different approaches for the reduction of the initial networks, including INLOCANDA (which is based on common cause and common effect phenomena identification) and some competitors, are applied.

5.1 Datasets

The datasets considered in our experiments are those adopted in Ceci et al. (2015), that are divided into SynTReN and DREAM5 datasets.

SynTReN data consist of steady-state expression data (10 conditions), generated by the tool SynTReN (Van den Bulcke et al. 2006), on the basis of the well-known regulatory networks of organisms *E. coli* and *S. cerevisiae* (henceforth Yeast) (Hempel et al. 2011). SynTReN selects connected sub-networks of the input networks and generates gene expression data which best describe the network structure. In the generation of expression data, SynTReN exploits Michaelis-Menten and Hill kinetics so that the generated expression data are very similar to real microarray mRNA measurements (Van den Bulcke et al. 2006). The interactions of the selected sub-networks are considered as gold standard for our evaluation, while the generated expression data are used as input for the initial reconstruction method we adopt. The adoption of synthetically generated data sets, in contrast to real measurements and human-constructed networks (e.g., Geistlinger et al. 2013), allows us to directly observe the quality of the reconstruction, since the topology of the underlying network is perfectly known a priori (Hempel et al. 2011).

We consider sub-networks of 100 and 200 genes, characterized by 121 and 303 edges, with an average degree of 2.42 and 3.03, respectively. In order to evaluate the robustness to noise, as in Ceci et al. (2015), we consider three versions of each dataset, with different levels of (additive, lognormally-distributed) noise, i.e., 0.0 (no noise), 0.1 and 0.5, introduced by SynTReN in the expression data. It is noteworthy that SynTReN does not explicitly introduce noise that can be directly associated to common cause or common effect phenomena.

DREAM5 data include real gene expression data about the *E. coli* organism, proposed in the DREAM5 challenge. It consists of 4,511 genes, for a total of more than 20 million possible interactions. This network has been largely used in the literature for the evaluation of methods for gene network reconstruction (Marbach et al. 2012; Omranian et al. 2016).

Starting from gene expression data, we reconstructed the networks by adopting the system GENERE (Ceci et al. 2015) which, with the best parameter reported in the original paper, obtains state-of-the-art results when compared to other combination strategies, as well as with respect to all the base methods evaluated in Marbach et al. (2012). GENERE produces a weighted (according to its confidence about the existence of each interaction), directed and possibly cyclic network.

5.2 Experimental setting

All the experiments were performed using four different classifiers, namely CLUS (Blockeel et al. 1998), JCHAID [in its most recent version JCHAID* (Ibarguren et al. 2016)], JRIP (Cohen 1995) and KNN (Aha et al. 1991) (keeping the default values for their parameters, and setting $K = 1$ for KNN), which are based on Predictive Clustering Trees (PCTs), Top-Down Induction of Decision Trees, Rule-Based classification and Lazy Instance-based classification, respectively. These methods resulted to be, after a preliminary analysis, the most appropriate ones to solve the considered task, due to their ability to handle highly imbalanced datasets (we remind that, in our case, the amount of positive examples is much lower with respect to the amount of negative examples). Moreover, since they are based on different models, we were also able to evaluate the effectiveness of the proposed approach, dependently on the classification model.

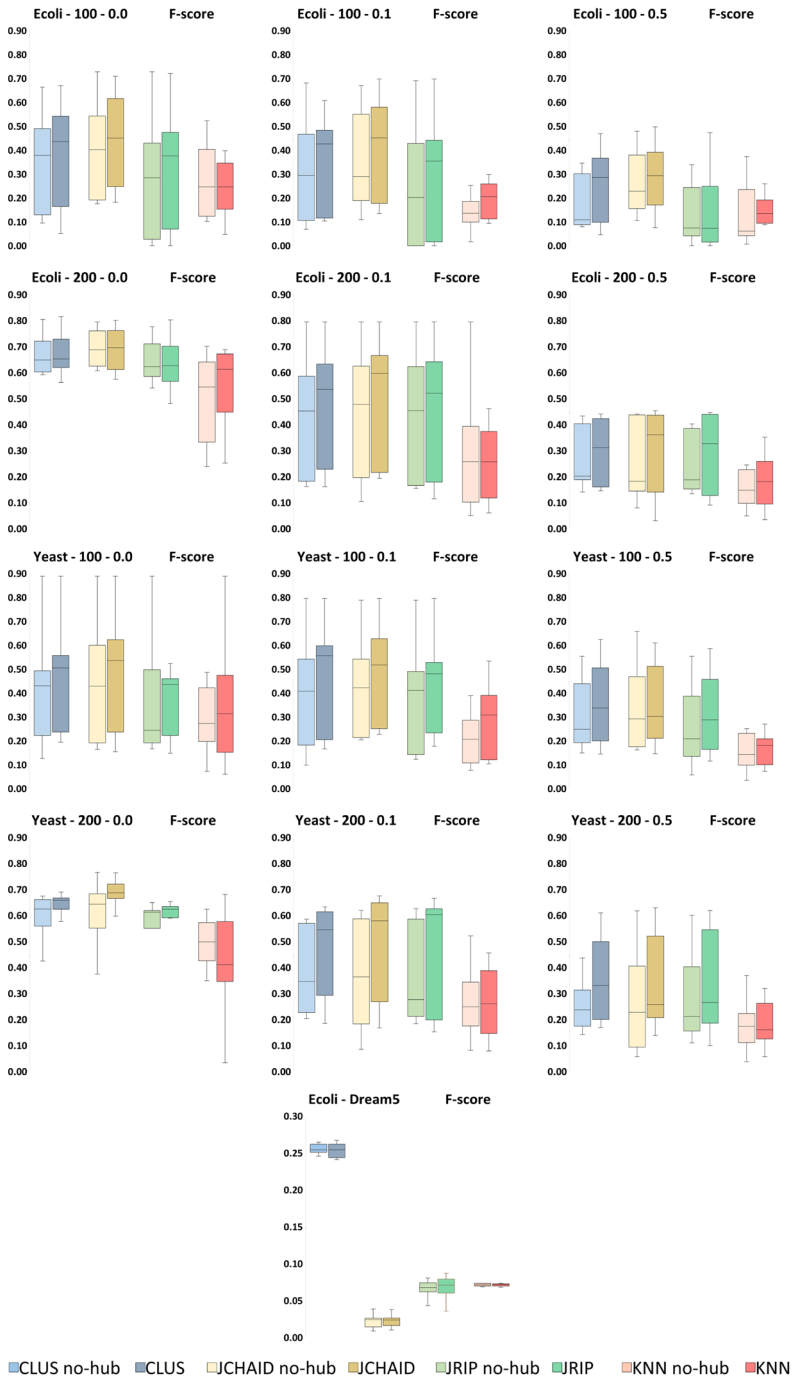


Fig. 8 F-score results obtained with the proposed embedding strategy and with the DFS approach, which is not able to exploit hub nodes (identified as *no-hubs*). The results are obtained with all the learners used in our evaluation, but without any network reduction

Table 1 p values of the Wilcoxon tests (with False Discovery Rate correction) performed between the proposed embedding approach and DFS

	Learning algorithm			
	CLUS	JCHAID	JRIP	KNN
p value	0.0001	0.0001	0.0001	0.0001

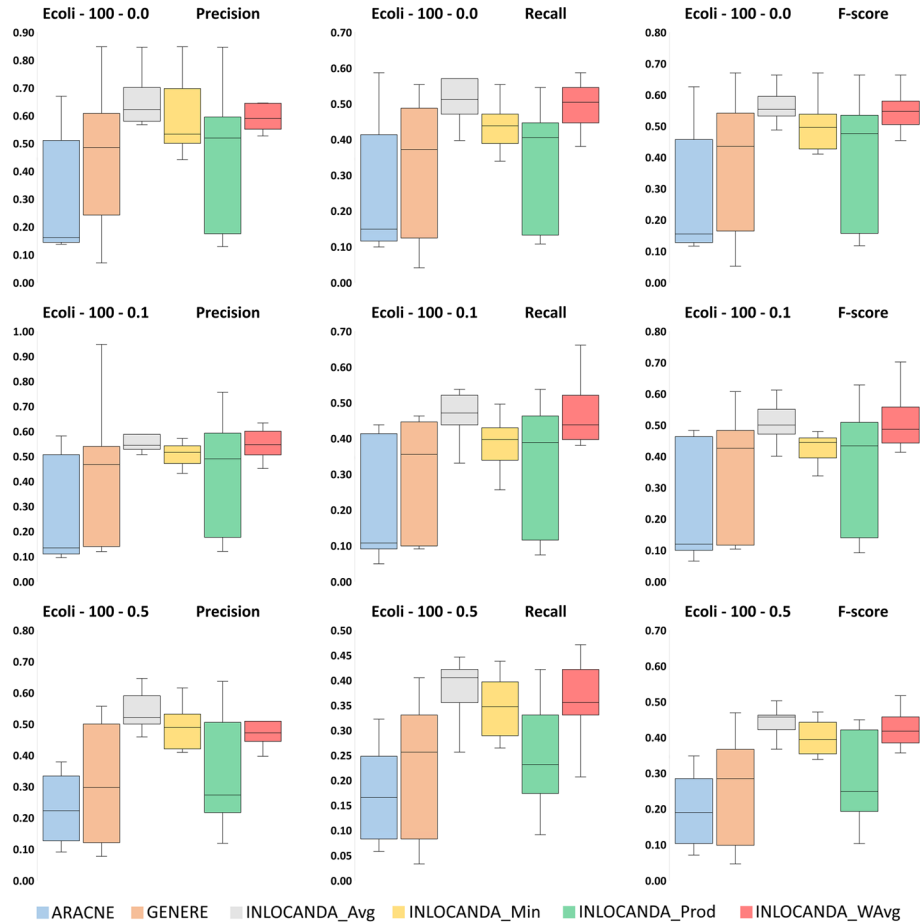


Fig. 9 Box plots depicting the results obtained by CLUS on *Syntren E.coli* datasets with 100 nodes, by varying the threshold on the weight of the original network edges

Our first competitor for the reduction of the network is a baseline approach that removes edges having a weight below a given threshold (in $\{0.0, 0.1, 0.2, \dots, 1.0\}$, leading to 11 networks) from the initial reconstructed network. We call this approach **GENERE**, since it is essentially the original network reconstructed by the system GENERE, with the baseline filtering approach based on the threshold on edges. As a second competitor, we considered the system **ARACNE** (Margolin et al. 2006), that, as we described in Sect. 2.3, analyzes all the possible triplets of connected genes and removes the edge with the lowest weight. In order to make a fair comparison with respect to GENERE, we evaluated the performance obtained by ARACNE starting from all the networks generated by GENERE with the all the considered

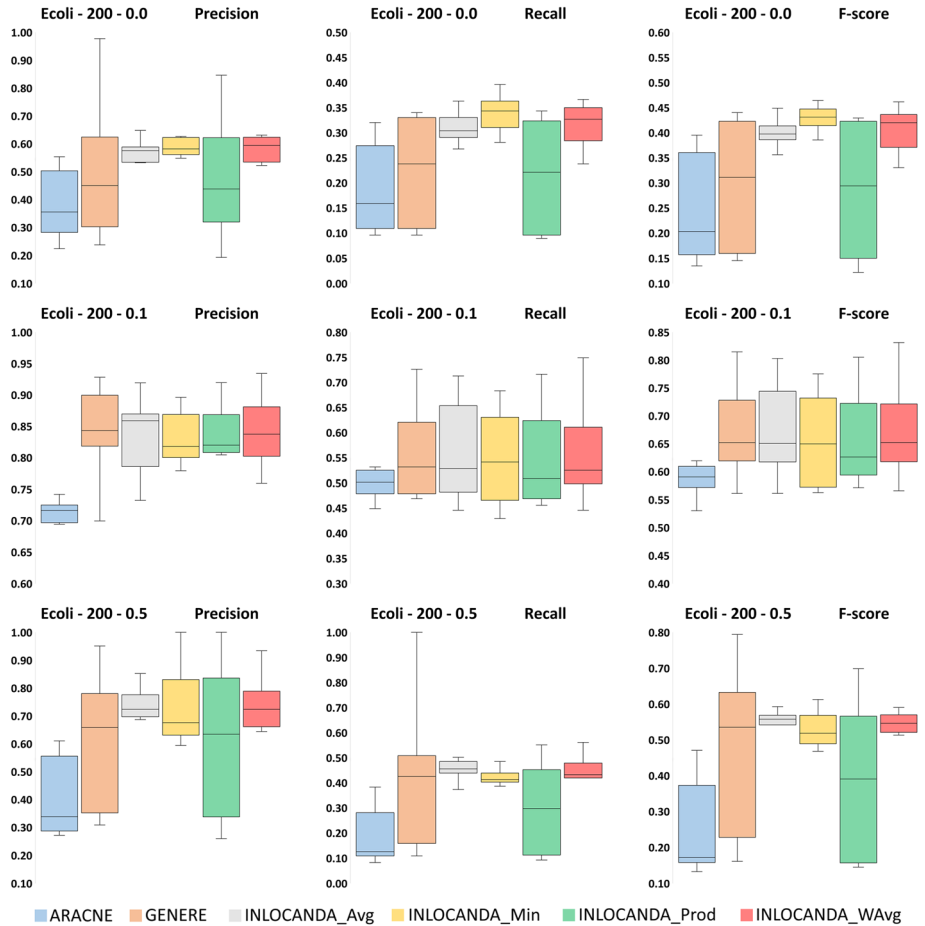


Fig. 10 Box plots depicting the results obtained by CLUS on *Syntren E.coli* datasets with 200 nodes, by varying the threshold on the weight of the original network edges

thresholds. Since ARACNE is not able to process directed networks, we transformed the directed networks produced by GENERE into undirected networks. When, in the network, there were two edges (i.e., for both directions) with a different edge weight between the same genes, we used the highest edge weight for the resulting edge.

For INLOCANDA, we performed the experiments with all the measures for the estimation of the reliability of the path proposed in Sect. 3, that are: minimum (Min), product (Prod), average (Avg) and weighted average (WAvg). As for ARACNE, we evaluated the obtained performance starting from all the networks generated by GENERE with all the considered thresholds.

We adopted a 10-fold cross validation strategy, where folds were built through a stratified random sampling, such that in the training set the percentage of positive/negative edges is the same as for the whole dataset. The stratified sampling was necessary due to the high imbalance between positive and negative instances. In order to guarantee fair comparisons, the folds are the same for all the considered competitors.

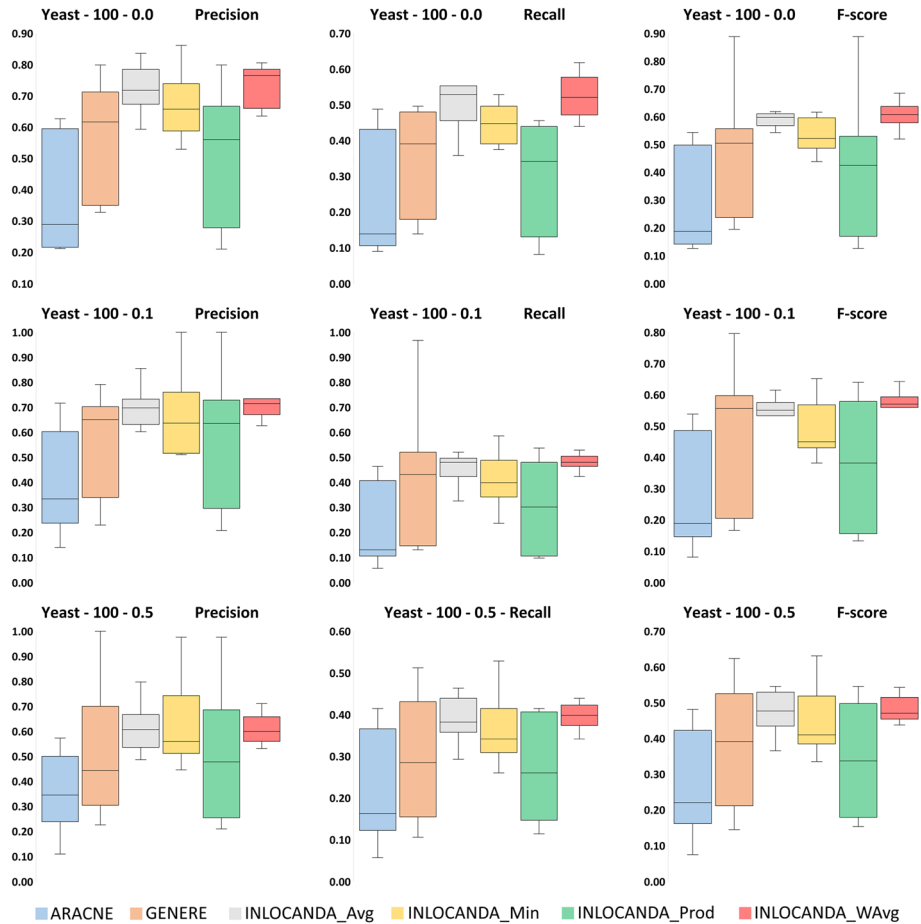


Fig. 11 Box plots depicting the results obtained by CLUS on *Syntren Yeast* datasets with 100 nodes, by varying the threshold on the weight of the original network edges

The results were collected in terms of:

- Average precision, recall and F-score, in order to evaluate the accuracy of the link prediction task.
- Percentage of edge removed by ARACNE and by all the variants of INLOCANDA, with respect to the original network reconstructed by GENERE, in order to evaluate the number of false positive edge detected.
- Area Under the ROC Curve (AUC), by varying the threshold for the removal of edges from the initial reconstructed network, in order to evaluate the overall accuracy of the learned models, independently on the chosen threshold.

Although one of the main goal of the method proposed in this paper is the reduction of false positive interactions, we adopted these evaluation measures, instead of considering only the amount of false positive interactions, since they are able to provide an overall view of the quality of the reconstructed network. Indeed, the minimization of false positive interactions can naively be obtained by a method which always predicts *false* for every possible inter-

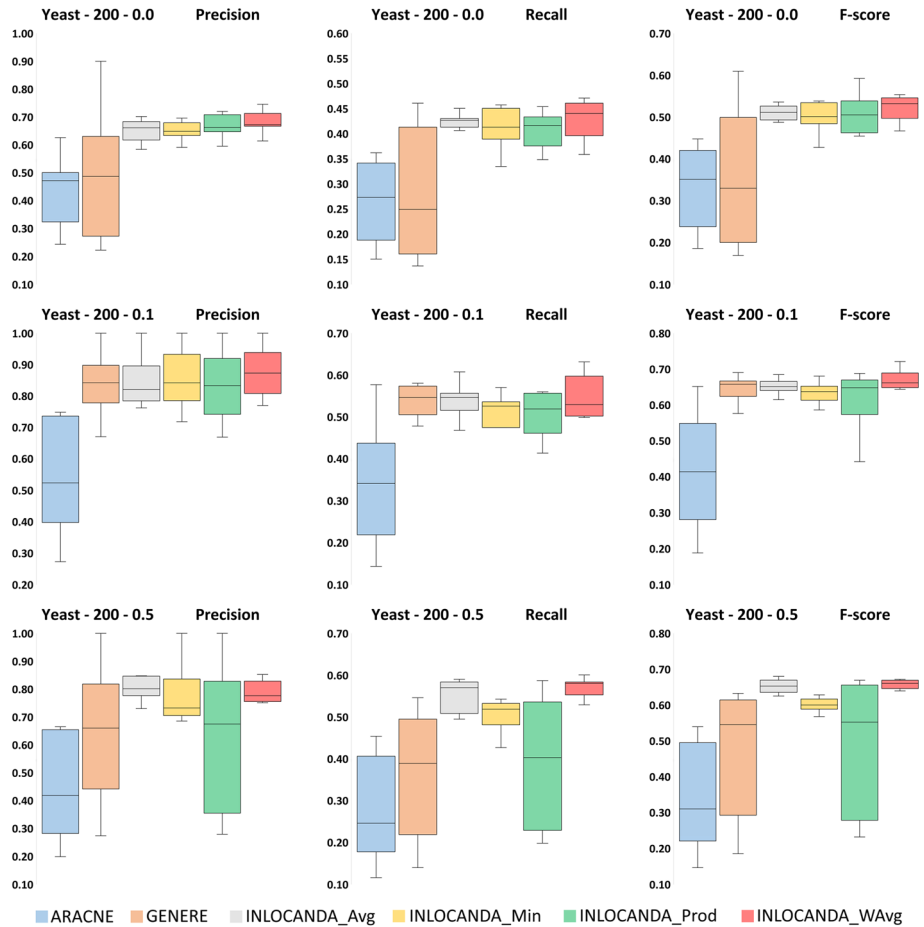


Fig. 12 Box plots depicting the results obtained by CLUS on *Syntren Yeast* datasets with 200 nodes, by varying the threshold on the weight of the original network edges

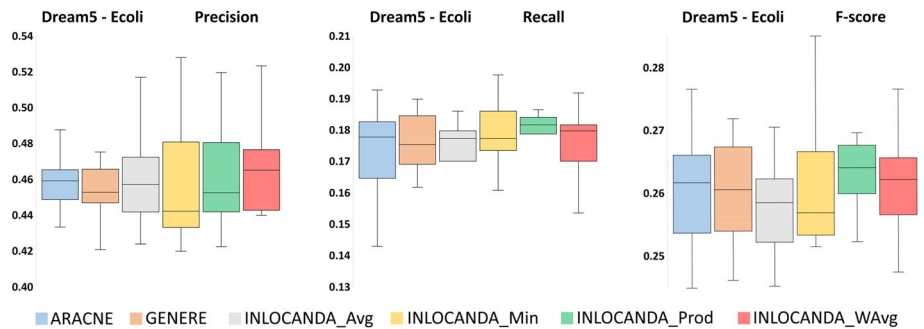


Fig. 13 Box plots depicting the results obtained by CLUS on *DREAM5 E.coli* dataset, by varying the threshold on the weight of the original network edges

Table 2 AUC results obtained on all the considered datasets

	ARACNE	GENERE	INLOCANDA			
			Avg	Min	Prod	WAvg
<i>CLUS</i>						
Ecoli—100—0.0	0.56	0.77	0.77	0.77	0.77	0.77
Ecoli—100—0.1	0.54	0.56	0.76	0.68	0.56	0.83
Ecoli—100—0.5	0.55	0.54	0.70	0.68	0.54	0.69
Ecoli—200—0.0	0.72	0.83	0.77	0.74	0.75	0.76
Ecoli—200—0.1	0.56	0.58	0.74	0.69	0.56	0.74
Ecoli—200—0.5	0.58	0.55	0.75	0.77	0.66	0.76
Yeast—100—0.0	0.58	0.73	0.77	0.71	0.56	0.72
Yeast—100—0.1	0.53	0.71	0.76	0.70	0.65	0.71
Yeast—100—0.5	0.57	0.50	0.73	0.66	0.57	0.71
Yeast—200—0.0	0.57	0.75	0.78	0.77	0.71	0.79
Yeast—200—0.1	0.58	0.59	0.79	0.77	0.60	0.80
Yeast—200—0.5	0.58	0.57	0.71	0.72	0.69	0.72
Dream5—Ecoli	0.59	0.58	0.59	0.59	0.59	0.59
<i>KNN</i>						
Ecoli—100—0.0	0.55	0.68	0.67	0.66	0.55	0.76
Ecoli—100—0.1	0.59	0.82	0.62	0.78	0.76	0.74
Ecoli—100—0.5	0.55	0.55	0.64	0.56	0.63	0.57
Ecoli—200—0.0	0.69	0.85	0.87	0.88	0.87	0.85
Ecoli—200—0.1	0.52	0.77	0.70	0.64	0.70	0.75
Ecoli—200—0.5	0.53	0.52	0.51	0.55	0.53	0.52
Yeast—100—0.0	0.51	0.74	0.69	0.61	0.54	0.69
Yeast—100—0.1	0.53	0.56	0.64	0.63	0.55	0.65
Yeast—100—0.5	0.53	0.50	0.62	0.54	0.56	0.60
Yeast—200—0.0	0.60	0.63	0.68	0.80	0.71	0.68
Yeast—200—0.1	0.54	0.73	0.76	0.70	0.66	0.79
Yeast—200—0.5	0.57	0.53	0.58	0.62	0.62	0.60
Dream5—Ecoli	0.53	0.53	0.53	0.53	0.53	0.53
<i>JCHAID</i>						
Ecoli—100—0.0	0.57	0.79	0.81	0.81	0.80	0.81
Ecoli—100—0.1	0.59	0.58	0.75	0.80	0.57	0.81
Ecoli—100—0.5	0.59	0.58	0.76	0.79	0.64	0.72
Ecoli—200—0.0	0.75	0.74	0.83	0.75	0.76	0.78
Ecoli—200—0.1	0.56	0.58	0.77	0.71	0.56	0.76
Ecoli—200—0.5	0.54	0.55	0.80	0.79	0.73	0.78
Yeast—100—0.0	0.61	0.60	0.77	0.75	0.56	0.78
Yeast—100—0.1	0.61	0.73	0.81	0.70	0.58	0.76
Yeast—100—0.5	0.59	0.50	0.71	0.67	0.60	0.69
Yeast—200—0.0	0.61	0.79	0.79	0.75	0.73	0.79

Table 2 continued

	ARACNE	GENERE	INLOCANDA			
			Avg	Min	Prod	WAvg
Yeast—200—0.1	0.61	0.61	0.79	0.75	0.63	0.79
Yeast—200—0.5	0.59	0.61	0.72	0.72	0.70	0.73
Dream5—Ecoli	0.51	0.51	0.51	0.51	0.51	0.51
<i>JRIP</i>						
Ecoli—100—0.0	0.53	0.80	0.81	0.81	0.81	0.81
Ecoli—100—0.1	0.67	0.75	0.78	0.76	0.76	0.77
Ecoli—100—0.5	0.55	0.63	0.67	0.67	0.63	0.60
Ecoli—200—0.0	0.70	0.78	0.78	0.78	0.79	0.80
Ecoli—200—0.1	0.67	0.73	0.74	0.71	0.74	0.72
Ecoli—200—0.5	0.54	0.67	0.78	0.79	0.80	0.79
Yeast—100—0.0	0.72	0.59	0.74	0.72	0.65	0.69
Yeast—100—0.1	0.68	0.79	0.74	0.68	0.72	0.72
Yeast—100—0.5	0.56	0.50	0.64	0.73	0.76	0.65
Yeast—200—0.0	0.67	0.76	0.74	0.73	0.72	0.75
Yeast—200—0.1	0.60	0.62	0.77	0.77	0.58	0.76
Yeast—200—0.5	0.66	0.60	0.72	0.74	0.71	0.71
Dream5—Ecoli	0.52	0.52	0.52	0.52	0.52	0.52

For each pair of classifier and approach adopted for the network reduction, the best result is highlighted in bold

actions. Obviously, this would lead to have no predicted interactions and, therefore, no true positive edges. The considered evaluation measures, on the contrary, simultaneously evaluate these aspects. For all the evaluation measures we also performed the Friedman test with the Nemenyi post-hoc test, with $\alpha = 0.05$, in order to compare the obtained results from a statistical viewpoint.

5.3 Evaluation of the embedding approach

In this section, we give a specific emphasis to the possible contribution provided by the proposed embedding strategy. It is noteworthy that the reported results are obtained with all the learners used in our evaluation, but without any network reduction. This was necessary in order to avoid any bias in the results that could be possibly introduced by the network reduction algorithm. We performed a specific analysis of the quality of the reconstructed networks, when a different embedding approach, which is not able to catch the possible structural roles of nodes in the network, is used.

In particular, we considered for comparison the DFS algorithm used in Grover and Leskovec (2016). This algorithm is similar to the approach adopted in this paper, but uses a different search strategy to define the neighborhood of each node. More specifically, while the approach used in our work adopts a combination of Breadth-First and Depth-First search strategies, the approach used by DFS is only based on the Depth-First search strategy (DFS is actually the acronym of Depth-First Search). The difference between the two search strategies is that, while DFS has a partial visibility on the reachable nodes, the algorithm used in

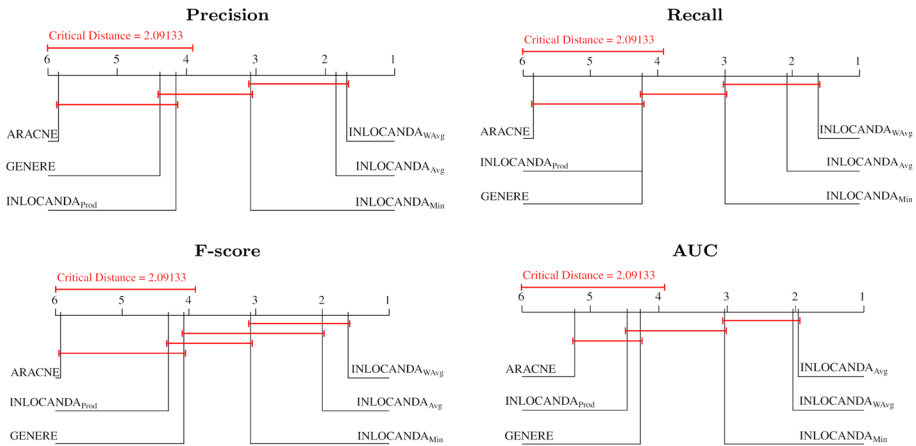


Fig. 14 Results of the Friedman test and Nemenyi post-hoc test on the Precision, Recall, F-score and AUC values obtained on all the datasets by CLUS, with $\alpha = 0.05$

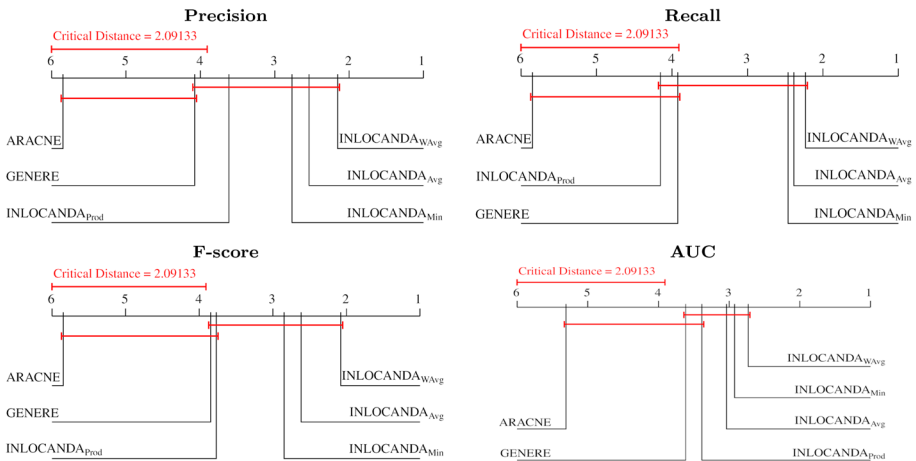


Fig. 15 Results of the Friedman test and Nemenyi post-hoc test on the Precision, Recall, F-score and AUC values obtained on all the datasets by KNN, with $\alpha = 0.05$

our approach has a more complete visibility of close nodes. The effect is that DFS is not able to properly represent the structure of the network in the proximity of a given node. As a consequence, DFS is not able to properly represent hub nodes which, as said before, play a crucial role in gene regulatory networks.

In Fig. 8 we report the F-score results for the link prediction task when both DFS and the solution we adopt in this paper are used (we identify DFS with the label “no-hub”). By observing the results, we can see that the proposed embedding strategy is always able to outperform the DFS approach, independently of the considered dataset and of the adopted learning algorithm. This confirms that, in the specific case of gene regulatory networks, properly catching possible structural roles of genes is fundamental to obtain a high quality reconstructed network. In order to confirm this result from a statistical viewpoint, we performed Wilcoxon tests (one for each classifier) with False Discovery Rate correction. The obtained p values are shown in Table 1, where we can see that the improvement provided by the embedding

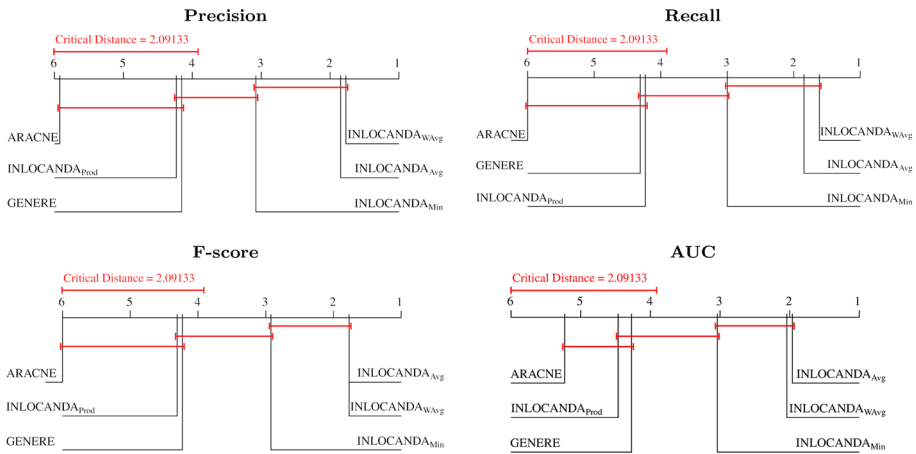


Fig. 16 Results of the Friedman test and Nemenyi post-hoc test on the Precision, Recall, F-score and AUC values obtained on all the datasets by JCHAID, with $\alpha = 0.05$

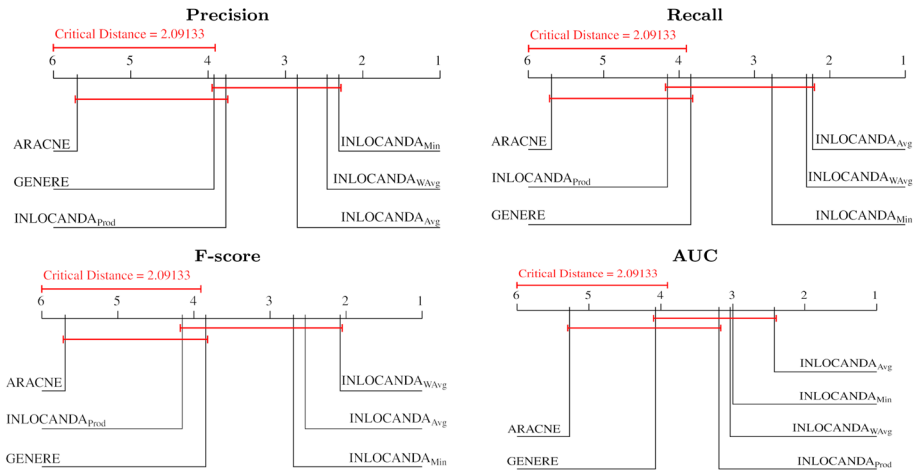


Fig. 17 Results of the Friedman test and Nemenyi post-hoc test on the Precision, Recall, F-score and AUC values obtained on all the datasets by JRIP, with $\alpha = 0.05$

strategy used in this paper is always statistically significant at $\alpha = 0.05$. Therefore, all the following experiments were performed with the embedding approach proposed in Sect. 3.2.

5.4 Evaluation of causality-based network reduction

In this section, we focus on the evaluation of the strategy adopted for the reduction of the input network. The obtained results, considering CLUS as classifier, are plotted in the box plots depicted in Figs. 9, 10, 11, 12 and 13, while the results obtained with the other classifiers are reported in the “Appendices”. The box plots are drawn by considering the different values for the threshold on the edge weight used to generate the input network. This allows us to evaluate the stability of the results with respect to such a parameter for all the systems used.

First, we focus on the SynTREn datasets. On these datasets, we can observe that ARACNE, GENE and INLOCANDA_Prod are more sensitive to the value of the input threshold (wider boxplots), whereas the other variants of INLOCANDA obtain very stable results. Moreover, we can observe that the initial networks reconstructed by GENE appear, in general, quite accurate and often lead to the highest F-Score value (see, for example, the F-score obtained on the datasets Ecoli-200-0.5, Yeast-100-0.0, Yeast-100-0.1 and Yeast-200-0.0). However, such a result can be obtained with a specific value of the input threshold and a wrong decision can lead to very poor results. On the contrary, a non-optimal choice of the value for the input threshold does not affect significantly the results obtained by INLOCANDA_Min, INLOCANDA_Avg and INLOCANDA_WAvg, that lead to stable, high F-score values in almost all the cases. ARACNE generally obtains lower F-score values, that is, it erroneously removes true gene interactions, possibly because it fails to catch causality phenomena. The different stability of the algorithms with respect to the threshold on the input network is also confirmed by the AUC results reported in Table 2, where it is possible to observe that, in general, INLOCANDA_Min, INLOCANDA_Avg and INLOCANDA_WAvg lead to the best, stable, results.

From the results reported in Table 2 we can also notice that the improvement, in terms of AUC, provided by the causality-based network reduction method implemented in INLOCANDA is independent of the specific learning algorithm. This will be more clear later, when we present the results of the statistical tests.

By comparing the results in terms of precision and recall, we can observe that the approaches able to obtain higher precision are the same as those able to achieve higher recall results. This is a very interesting result, since the removal of false positive edges from the network could have led to affect the results in terms of recall. On the contrary, we can observe that INLOCANDA, especially on the variants based on Avg and WAvg, is able to effectively filter out false positive edges, keeping a good result in terms of recall. Looking specifically at the results obtained on the large dataset from DREAM5 (Fig. 13), we can observe that, even if the difference is not so evident, INLOCANDA_WAvg is still able to obtain the most precise result, keeping a good recall.²

Analyzing the influence on the results caused by the presence of noise in the data, we can observe that, without noise or with a low amount of noise, GENE obtains acceptable results (although they are unstable, with respect to the input threshold). However, when the amount of noise increases, it tends to obtain lower, more unstable F-score values. On the contrary, INLOCANDA, especially with the variants based on Avg and WAvg, generally shows good, stable results, even in the case of the datasets with the highest amount of noise. This proves that the proposed method is actually very robust to the possible presence of noise in the data and that the variants based on the averages (Avg and WAvg) are able to make smooth decisions when they have to remove an interaction.

As mentioned in Sect. 5.2, we performed a set of Friedman tests with the Nemenyi post-hoc test, which show the overall superiority of the proposed method. In particular, following the graphical view proposed in Demšar (2006), in Figs. 14, 15, 16 and 17 we plot four graphs for each learning method showing the result of the statistical test in terms of Precision, Recall, F-score and AUC. All the measures show that INLOCANDA_WAvg generally leads to the best results. Moreover, from a statistical viewpoint, the difference with respect to ARACNE is significant with $\alpha = 0.05$ according to all the considered measures. On the contrary, the difference with respect to GENE is statistically significant only when we use CLUS and

² The fact that this dataset is relatively difficult to analyze is confirmed by the maximum AUPRC obtained by the method proposed in Marbach et al. (2012) and by GENE (Ceci et al. 2015), which are 0.09 and 0.12, respectively.

Table 3 Percentage of edges removed by ARACNE and by the different versions of INLOCANDA from the original network reconstructed by GENERE

Dataset	Threshold	ARACNE (%)	INLOCANDA (Stage 1)			
			Avg (%)	Min (%)	Prod (%)	WAvg (%)
Ecoli—100—0.0	0.00	96.97	96.01	69.11	7.22	96.57
	0.30	96.07	95.53	75.87	9.19	95.85
	0.60	89.61	85.85	64.02	20.44	86.78
	1.00	69.18	0.00	0.00	0.00	0.00
Ecoli—100—0.1	0.00	96.24	95.14	78.54	21.66	95.45
	0.30	96.21	95.34	77.21	21.81	95.59
	0.60	88.15	81.63	64.91	48.93	82.45
	1.00	60.50	0.00	0.00	0.00	0.00
Ecoli—100—0.5	0.00	96.36	91.34	76.85	46.99	92.56
	0.30	96.35	91.37	76.96	47.39	92.60
	0.60	91.61	82.09	66.40	52.68	83.49
	1.00	73.27	0.00	0.00	0.00	0.00
Ecoli—200—0.0	0.00	98.29	98.74	85.68	1.85	98.82
	0.30	67.52	59.90	42.77	21.68	61.49
	0.60	48.80	40.92	35.45	33.73	41.78
	1.00	3.86	0.00	0.00	0.00	0.00
Ecoli—200—0.1	0.00	97.54	98.49	83.28	5.32	98.71
	0.30	96.97	98.36	84.35	6.35	98.54
	0.60	81.40	85.02	47.64	28.08	85.73
	1.00	87.50	0.00	0.00	0.00	0.00
Ecoli—200—0.5	0.00	97.08	98.72	85.27	7.91	98.85
	0.30	97.07	98.73	85.41	8.00	98.85
	0.60	88.59	94.78	70.81	8.59	95.84
	1.00	62.50	0.00	0.00	0.00	0.00
Yeast—100—0.0	0.00	96.95	96.07	78.63	9.49	96.86
	0.30	96.93	96.38	81.04	9.64	97.04
	0.60	89.30	86.21	59.65	30.81	88.71
	1.00	0.00	0.00	0.00	0.00	0.00
Yeast—100—0.1	0.00	96.55	95.71	71.60	15.09	96.35
	0.30	96.26	95.94	69.11	15.29	96.22
	0.60	86.23	82.14	53.66	42.11	83.33
	1.00	93.33	0.00	0.00	0.00	0.00
Yeast—100—0.5	0.00	95.68	97.58	84.45	8.81	97.85
	0.30	95.67	97.58	84.60	8.89	97.85
	0.60	77.46	88.30	54.98	16.79	89.00
	1.00	100.00	0.00	0.00	0.00	0.00
Yeast—200—0.0	0.00	98.66	98.53	70.86	4.60	98.80
	0.30	91.79	92.17	54.28	21.76	93.06

Table 3 continued

Dataset	Threshold	ARACNE (%)	INLOCANDA (Stage 1)			
			Avg (%)	Min (%)	Prod (%)	WAvg (%)
	0.60	86.79	85.53	60.09	31.49	85.87
	1.00	31.48	0.00	0.00	0.00	0.00
Yeast—200—0.1	0.00	98.44	93.90	71.90	12.95	94.39
	0.30	98.43	94.12	70.91	13.18	94.48
	0.60	94.26	77.06	43.08	21.67	77.55
	1.00	90.73	0.00	0.00	0.00	0.00
Yeast—200—0.5	0.00	98.05	92.13	77.29	64.98	92.97
	0.30	98.04	92.18	77.47	65.20	92.99
	0.60	97.07	88.85	74.10	60.91	89.63
	1.00	84.77	0.00	0.00	0.00	0.00
Dream5—Ecoli	0.0	97.24	95.09	77.43	6.97	95.45
	0.3	97.24	95.19	80.44	9.39	95.49
	0.6	97.25	95.22	81.10	59.20	95.51
	1.0	98.01	97.10	97.10	97.10	97.10

JCHAID, whereas in the case of JRIP and KNN, INLOCANDA_WAvg still outperforms GENERE, but the difference in the ranks is not statistically significant. This smaller gap between INLOCANDA_WAvg and GENERE for JRIP and KNN can be motivated by lower performances of the classifiers, which make the contribution of the network reduction less evident.

Overall, we can conclude that the effectiveness of the proposed approach is almost independent on the final classifier adopted, provided that it is able to work on highly imbalanced datasets. It is noteworthy that all these considerations are valid for all the measures considered.

In Table 3, we can observe the percentage of edges removed by ARACNE and by the different versions of the Stage 1 of INLOCANDA from the original network reconstructed by GENERE. In particular, we can observe that the variants based on the minimum and (especially) on the product, do not remove many edges, i.e., they are more conservative. This is clearly motivated by the fact that they generally underestimate the reliability of a path, leading to remove less direct edges. On the contrary, the variants based on the average and on the weighted average, as well as ARACNE, lead to remove up to 98% of edges. Figure 18 summarizes these results and show that INLOCANDA_WAvg and ARACNE are the most selective algorithms and there is no statistical difference between them. However, while ARACNE pays the price of such a result with a significantly worse precision, recall, F-score and AUC, INLOCANDA_Avg and INLOCANDA_WAvg are able to obtain a high quality reconstruction, by keeping **only** 2–5% of the original edges.

A final consideration can be drawn from the non-optimal results obtained by INLOCANDA_Prod. This phenomenon can be motivated by the fact that it is based on assumptions that are often violated in biological networks (i.e., the independence of the events). On the contrary, the very good results obtained by the variants Avg and WAvg can be motivated by the fact that their assumptions correctly reflect the real interactions among the genes. In this respect, we can conclude that the variants based on Avg and WAvg are the most

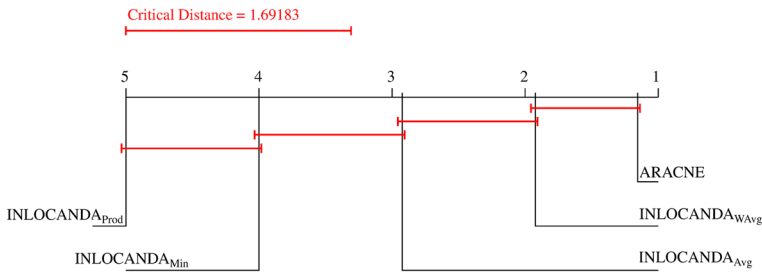


Fig. 18 Results of the Friedman test and Nemenyi post-hoc test on the number of edges removed by ARACNE and by all the variants of INLOCANDA, with $\alpha = 0.05$

appropriate for the reconstruction of gene networks, possibly due to their smoothness in making a decision on cutting an edge. However, this does not mean that such measures are the most appropriate for all kinds of application domains. Indeed, we want to emphasize that INLOCANDA could be applied to other kinds of networks and that its behaviour could be easily adapted to other domains by identifying a proper function $f(\cdot)$, able to catch specific assumptions of the domain in hand.

6 Conclusions and future work

In this work, we proposed INLOCANDA, a machine learning method which is able to learn an inductive predictive model for gene network reconstruction. INLOCANDA has two main characteristics: 1) it is able to analyze and exploit causality phenomena in order to discard, in the reconstructed network, edges which can be considered the result of indirect regulation activities; 2) it is able to take into account possible community structures and possible similar roles by means of a specific graph embedding strategy, which, according to the results, provides a significant boost in terms of the quality of the reconstruction. Moreover, contrary to existing methods for the identification of a transitive reduction of a network or for the identification of redundancies in reconstructed biological networks, INLOCANDA simultaneously offers all the following features: (i) it is able to analyze directed weighted networks, fully exploiting the weights on the edges which represent their reliability; (ii) it does not require any pre-processing step to handle the possible presence of cycles; (iii) it is able to identify indirect interactions of arbitrary length and to exploit them to remove direct edges considered as false positives. The estimation of the reliability of a path is guided by a function, which can be tuned according to specific underlying phenomena and assumptions with respect to the application domain in hand.

The obtained results show that INLOCANDA, especially in its variants based on the averages, is able to obtain (statistically) better, more stable predictive accuracy with respect to the considered competitors, even with highly noisy data and even if the reconstructed networks contain a lower (or comparable) number of edges with respect to direct competitors. Moreover, by means of an extensive experimental evaluation, we proved that such advantages are present with multiple supervised learners, provided that they are able to handle the high imbalance in the dataset. All these aspects allow the expert to concentrate their in-vitro validation activities on few promising gene interactions, possibly representing true causal phenomena.

As future work, we plan to develop a distributed variant of the method within the Apache Spark framework, able to work on large scale networks, and to adopt it for the analysis of gene networks concerning Homo Sapiens. In this respect, we also plan a biological evaluation of the results, guided by biologists. Finally, focusing on the influence of $f(\cdot)$, we will

evaluate the effectiveness of INLOCANDA in the analysis of networks representing data in other domains, where different assumptions on the network structure may hold.

Acknowledgements We would like to acknowledge the support of the European Commission through the Projects MAESTRA - Learning from Massive, Incompletely annotated, and Structured Data (Grant Number ICT-2013-612944) and TOREADOR - Trustworthy Model-aware Analytics Data Platform (Grant Number H2020-688797). We would also like to thank Lynn Rudd for her help in reading and correcting the manuscript.

Appendix 1: Results obtained by KNN classifier

See Figs. 19, 20, 21, 22 and 23.

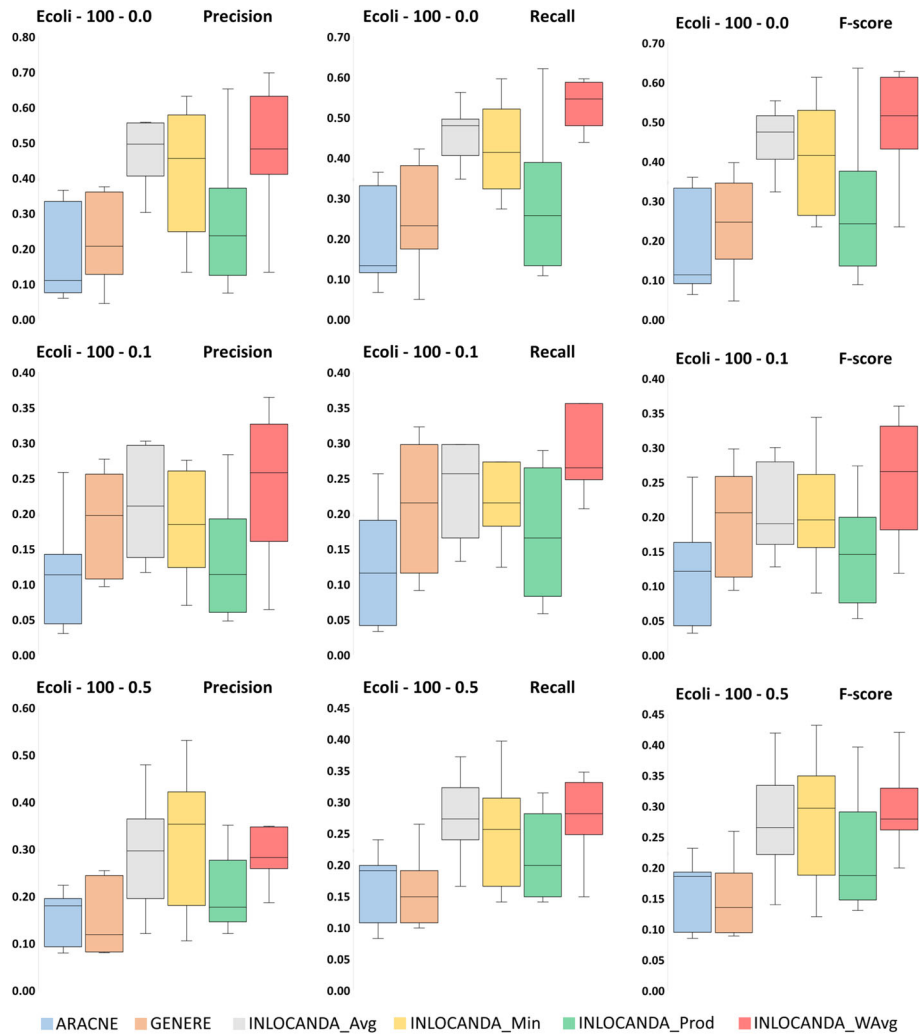


Fig. 19 Box plots depicting the results obtained by KNN on *Syntren E.coli* datasets with 100 nodes, by varying the threshold on the weight of the original network edges

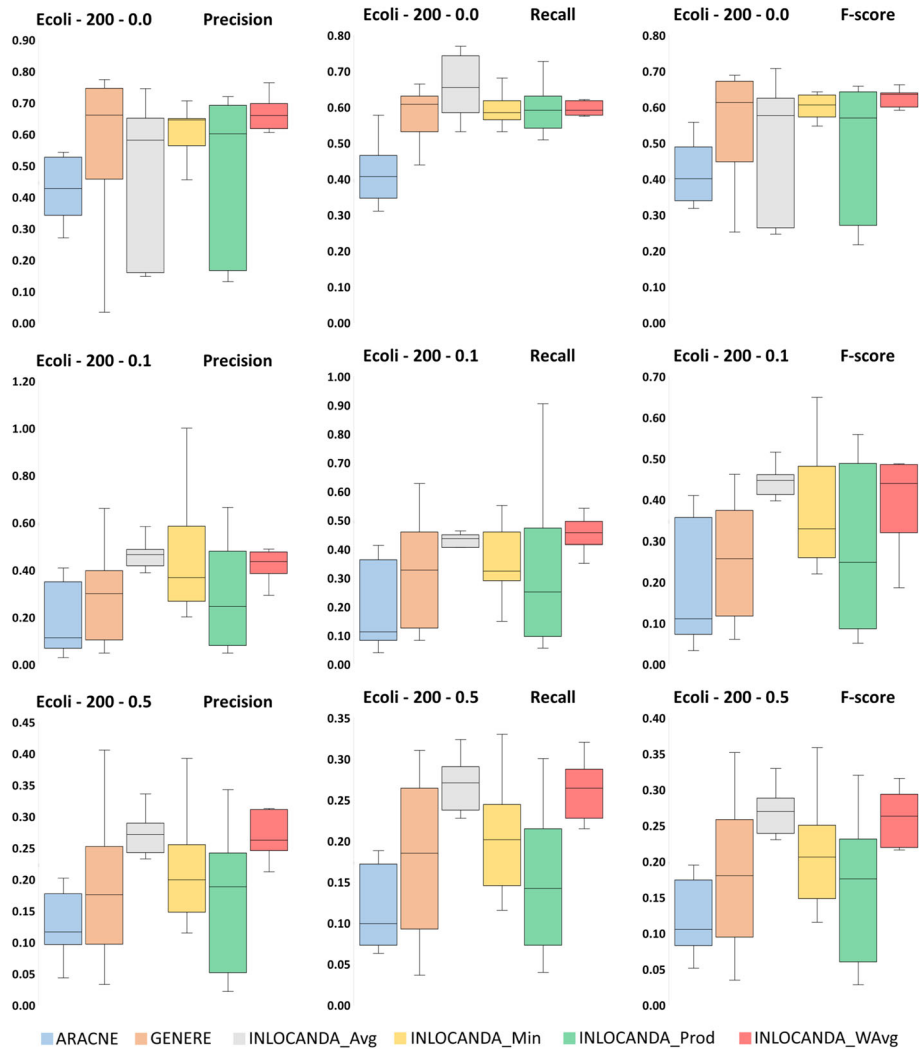


Fig. 20 Box plots depicting the results obtained by KNN on *Synten E.coli* datasets with 200 nodes, by varying the threshold on the weight of the original network edges

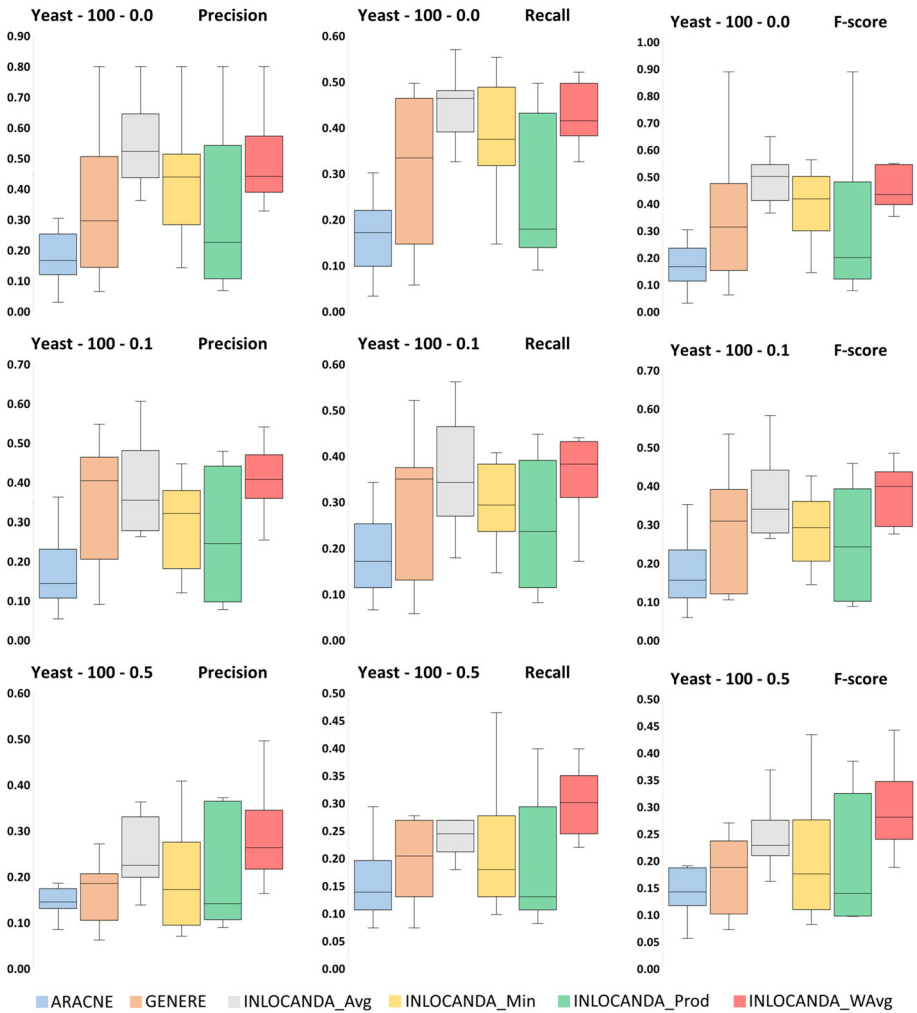


Fig. 21 Box plots depicting the results obtained by KNN on *Syntren Yeast* datasets with 100 nodes, by varying the threshold on the weight of the original network edges

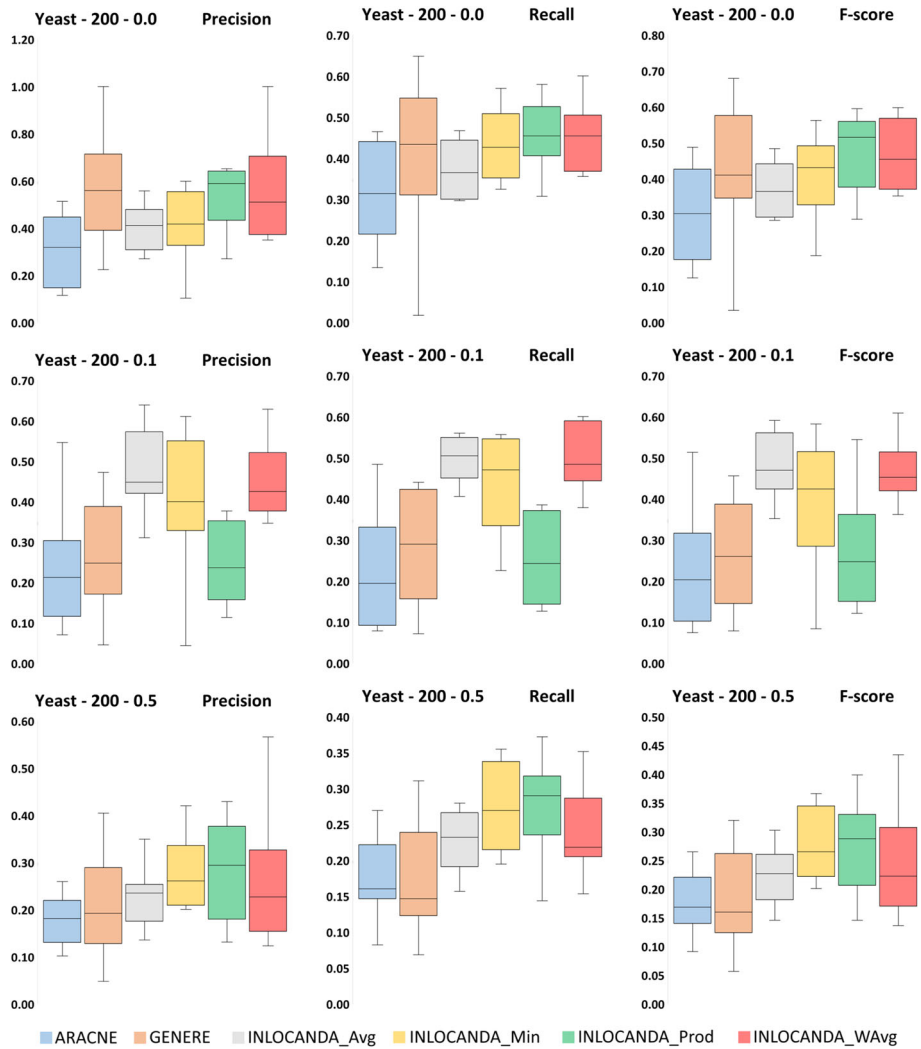


Fig. 22 Box plots depicting the results obtained by KNN on *Syntren Yeast* datasets with 200 nodes, by varying the threshold on the weight of the original network edges

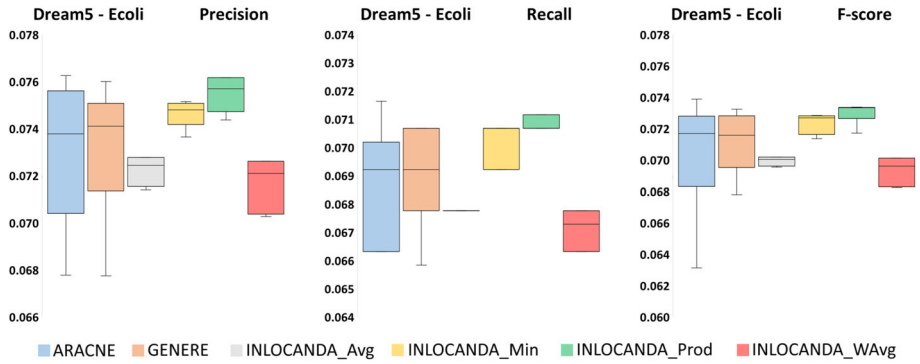


Fig. 23 Box plots depicting the results obtained by KNN on *DREAM5 E.coli* dataset, by varying the threshold on the weight of the original network edges

Appendix 2: Results obtained by JCHAID classifier

See Figs. 24, 25, 26, 27 and 28.

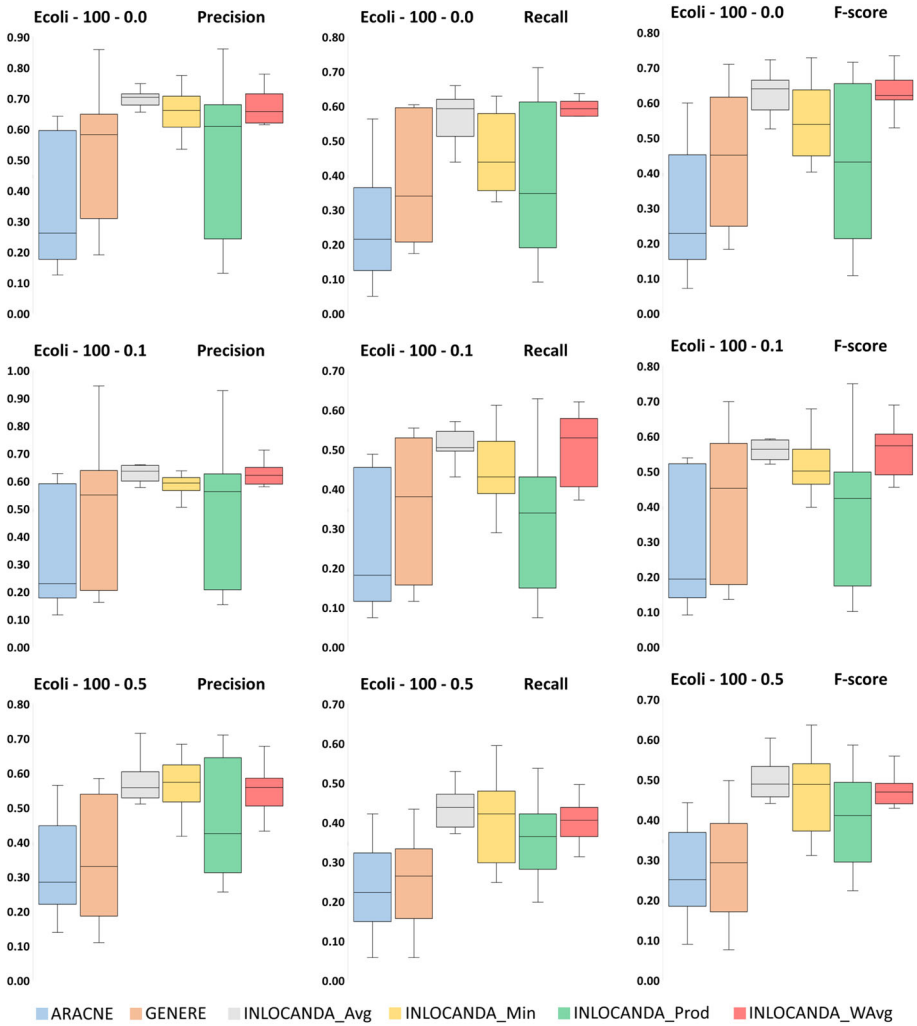


Fig. 24 Box plots depicting the results obtained by JCHAID on *Syntren E.coli* datasets with 100 nodes, by varying the threshold on the weight of the original network edges

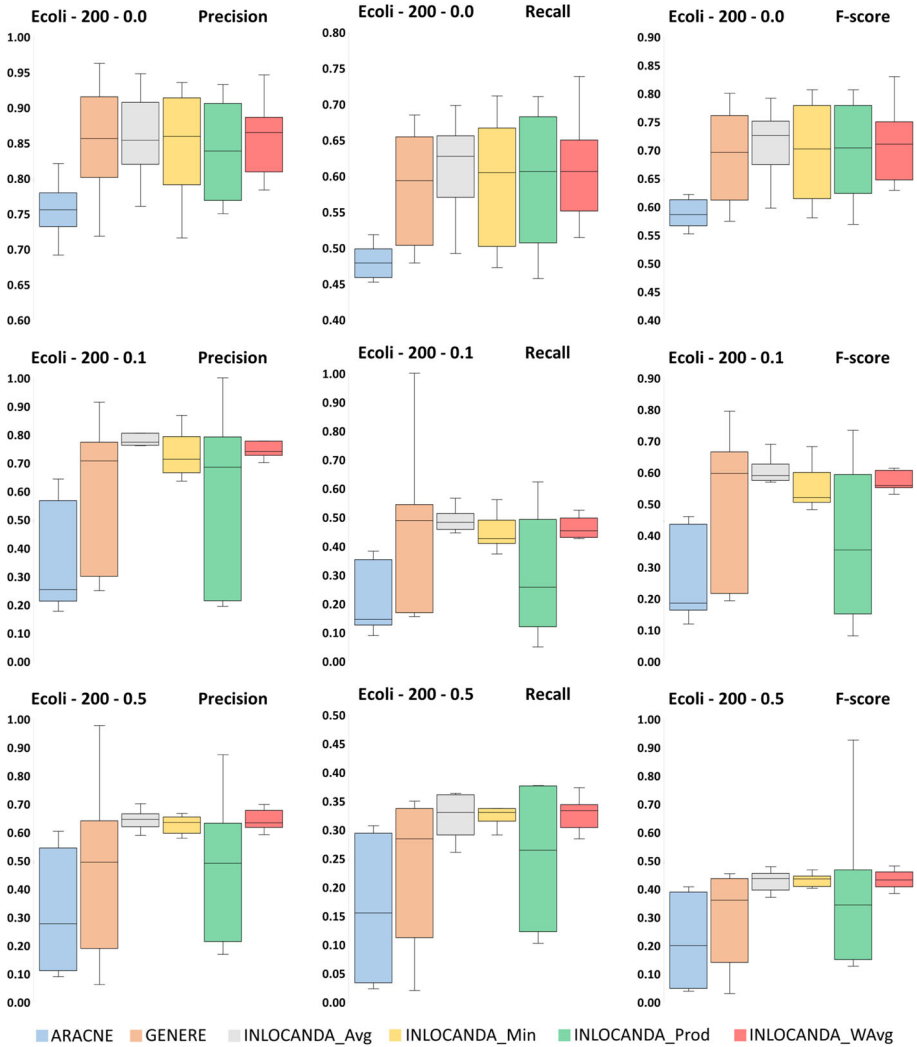


Fig. 25 Box plots depicting the results obtained by JCHAID on *Syntren E.coli* datasets with 200 nodes, by varying the threshold on the weight of the original network edges

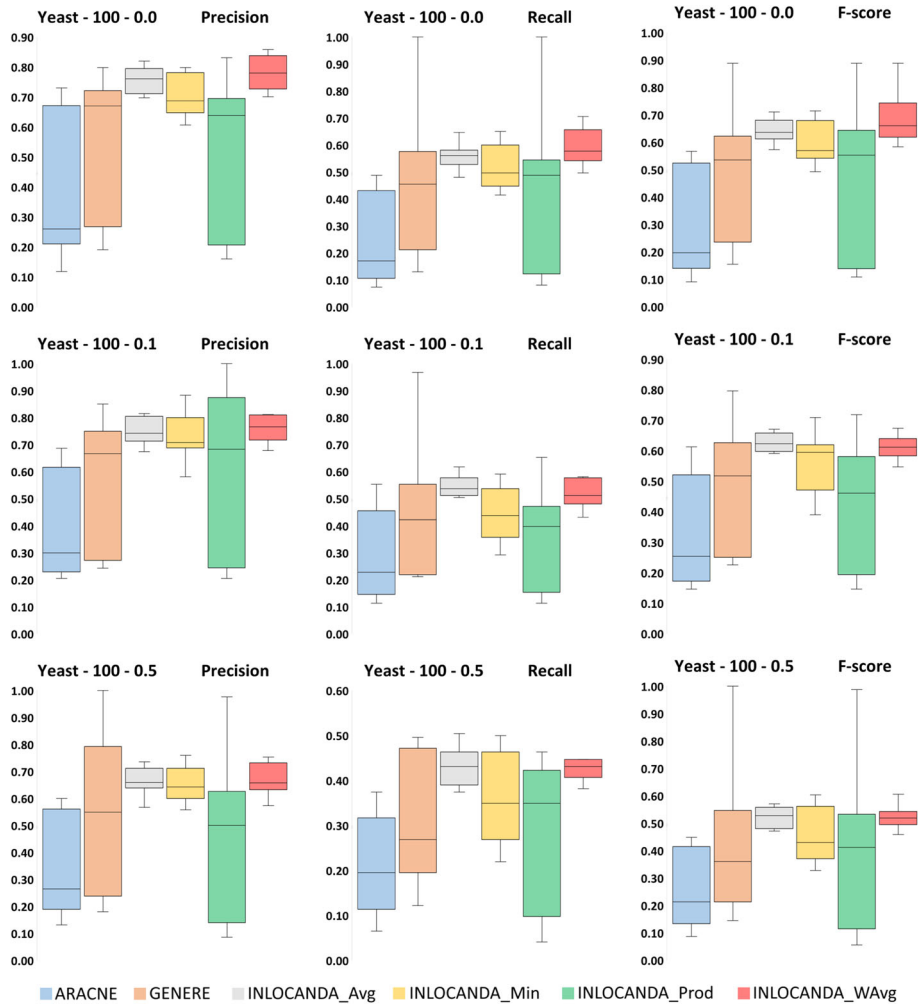


Fig. 26 Box plots depicting the results obtained by JCHAID on *Syntren Yeast* datasets with 100 nodes, by varying the threshold on the weight of the original network edges

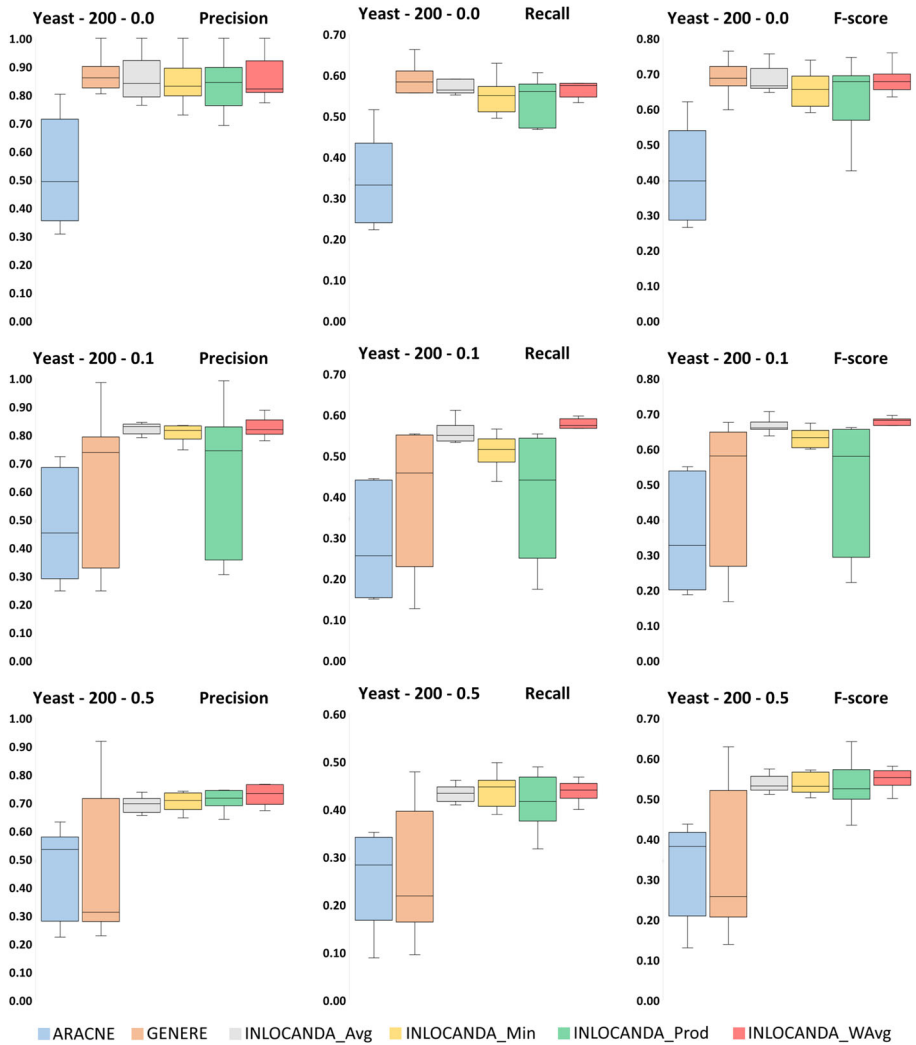


Fig. 27 Box plots depicting the results obtained by JCHAID on *Syntren Yeast* datasets with 200 nodes, by varying the threshold on the weight of the original network edges

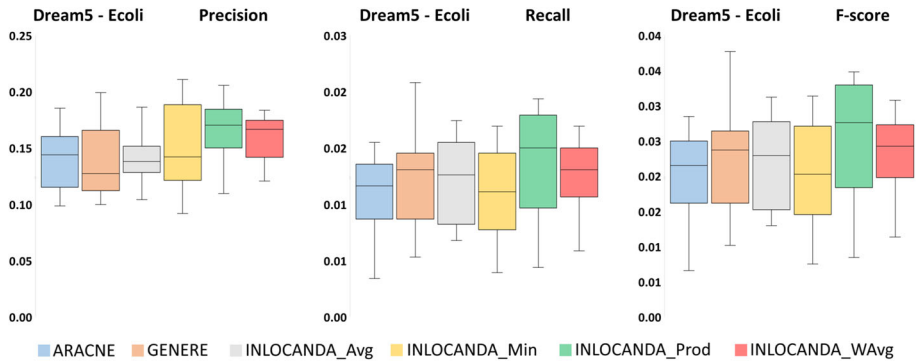


Fig. 28 Box plots depicting the results obtained by JCHAID on *DREAM5 E.coli* dataset, by varying the threshold on the weight of the original network edges

Appendix 3: Results obtained by JRIP classifier

See Figs. 29, 30, 31, 32 and 33.

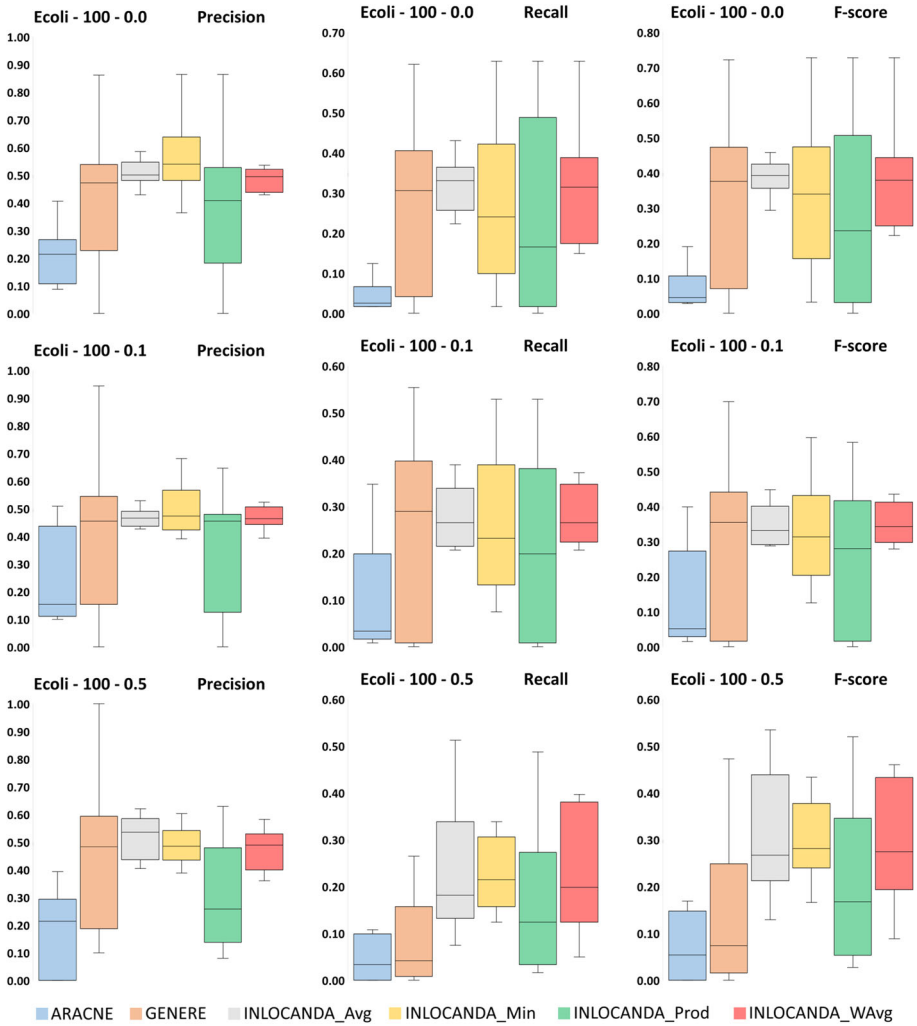


Fig. 29 Box plots depicting the results obtained by JRIP on *Syntren E.coli* datasets with 100 nodes, by varying the threshold on the weight of the original network edges

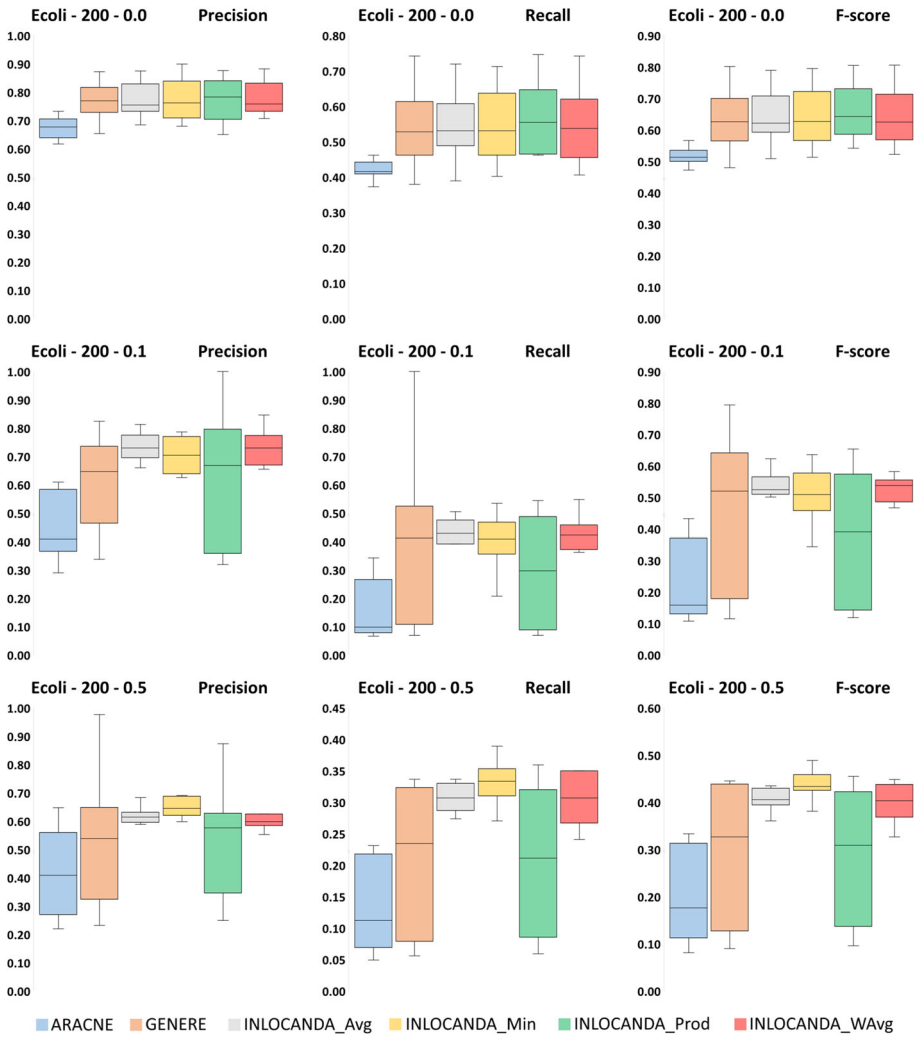


Fig. 30 Box plots depicting the results obtained by JRIP on *Syntren E.coli* datasets with 200 nodes, by varying the threshold on the weight of the original network edges

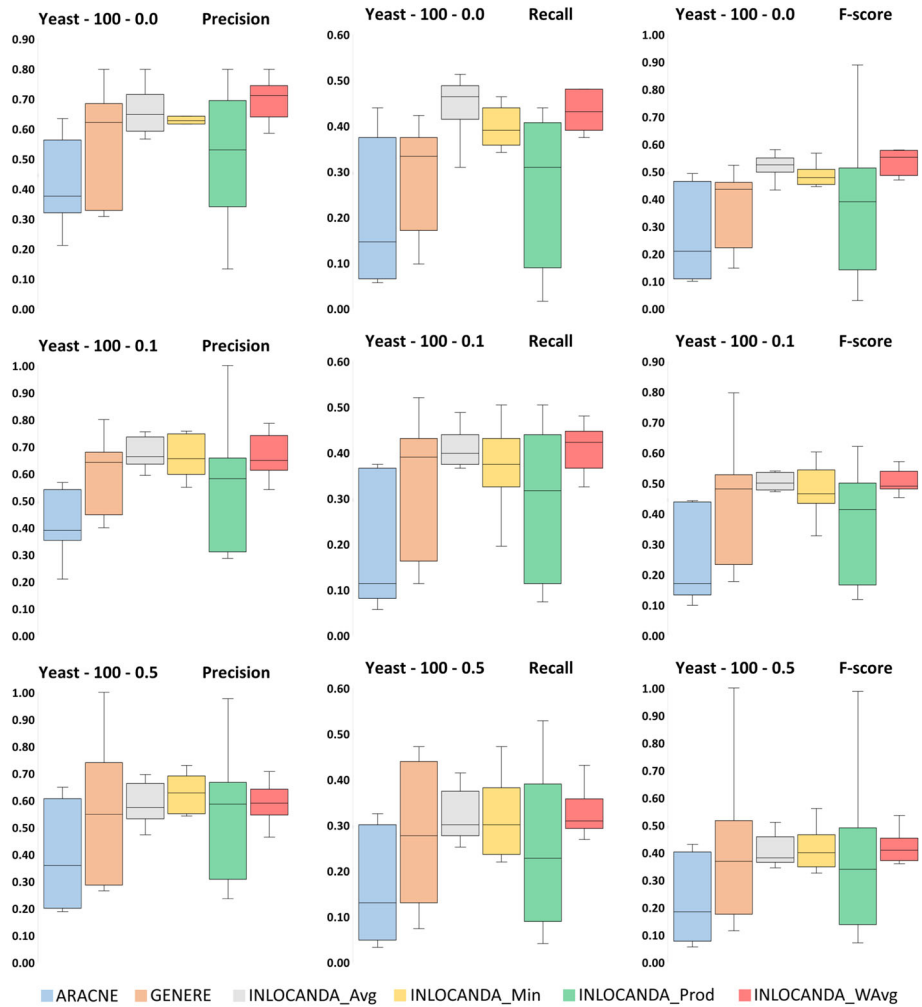


Fig. 31 Box plots depicting the results obtained by JRIP on *Syntren Yeast* datasets with 100 nodes, by varying the threshold on the weight of the original network edges

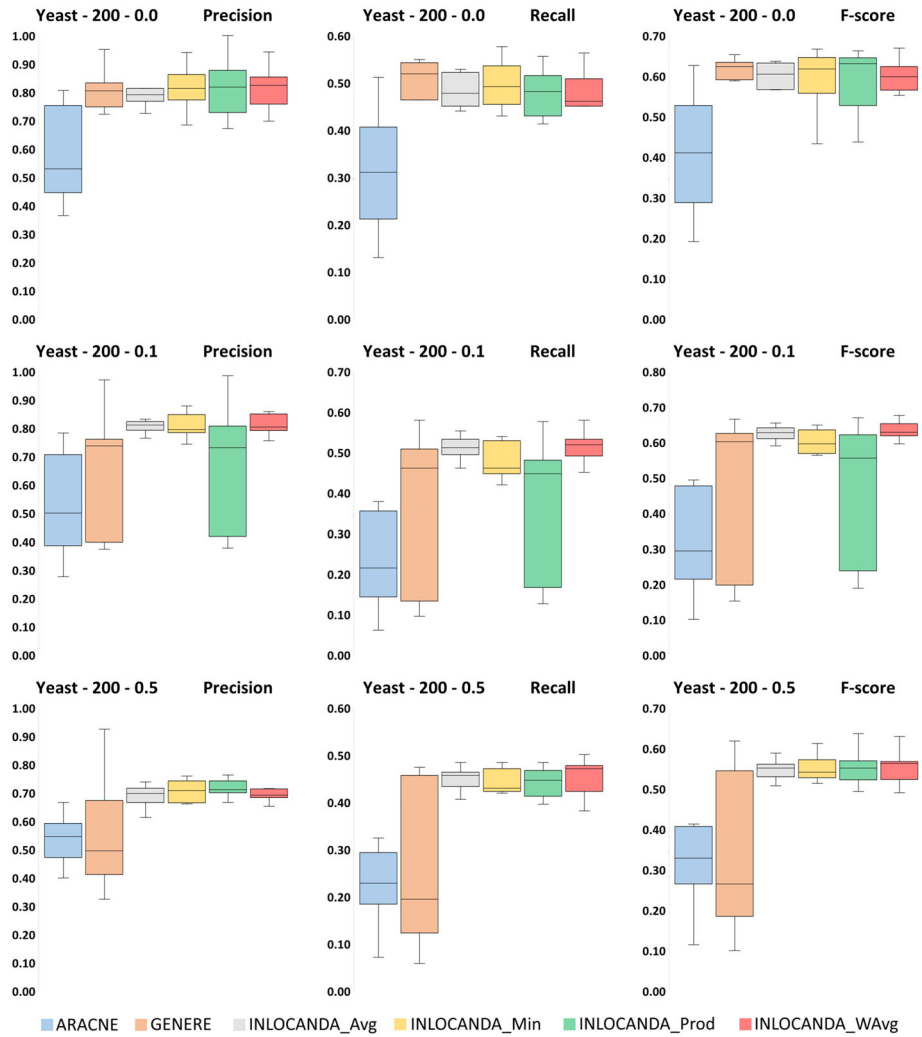


Fig. 32 Box plots depicting the results obtained by JRIP on *Syntren Yeast* datasets with 200 nodes, by varying the threshold on the weight of the original network edges

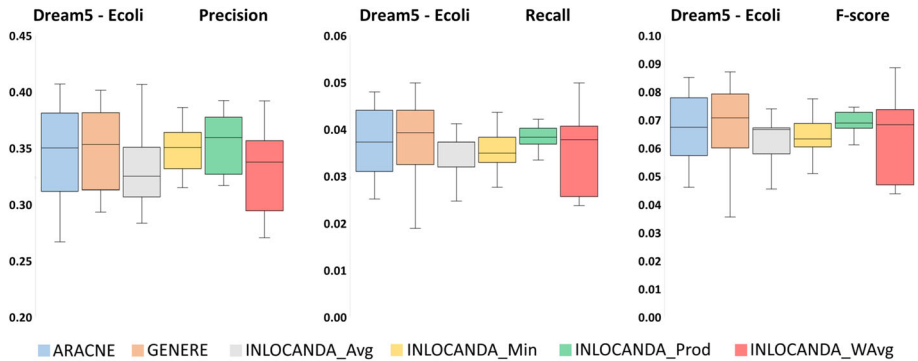


Fig. 33 Box plots depicting the results obtained by JRIP on *DREAM5 E.coli* dataset, by varying the threshold on the weight of the original network edges

References





- Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1), 37–66.
- Aho, A. V., Garey, M. R., & Ullman, J. D. (1972). The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2), 131–137.
- Atias, N., & Sharan, R. (2012). Comparative analysis of protein networks: Hard problems, practical solutions. *Communications of the ACM*, 55(5), 88–97.
- Babu, M. M., Luscombe, N. M., Aravind, L., Gerstein, M., & Teichmann, S. A. (2004). Structure and evolution of transcriptional regulatory networks. *Current Opinion in Structural Biology*, 14(3), 283–291.
- Belkin, M., & Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in neural information processing systems 14* (pp. 585–591). Cambridge: MIT Press.
- Berger, M. F., & Bulyk, M. L. (2009). Universal protein-binding microarrays for the comprehensive characterization of the DNA-binding specificities of transcription factors. *Nature Protocols*, 4(3), 393–411.
- Blockeel, H., Raedt, L. D., & Ramon, J. (1998). Top-down induction of clustering trees. In J. W. Shavlik (Ed.), *ICML 1998* (pp. 55–63). Burlington: Morgan Kaufmann.
- Böck, M., Ogishima, S., Tanaka, H., Kramer, S., & Kaderali, L. (2012). Hub-centered gene network reconstruction using automatic relevance determination. *PLOS ONE*, 7(5), 1–17.
- Bošnački, D., Odenbrett, M. R., Wijs, A., Ligtenberg, W., & Hilbers, P. (2012). Efficient reconstruction of biological networks via transitive reduction on general purpose graphics processors. *BMC Bioinformatics*, 13(1), 281.
- Bulyk, M. L. (2005). Discovering DNA regulatory elements with bacteria. *Nature Biotechnology*, 23(8), 942–944.
- Ceci, M., Pio, G., Kuzmanovski, V., & Džeroski, S. (2015). Semi-supervised multi-view learning for gene network reconstruction. *PLOS ONE*, 10(12), 1–27.
- Cohen, W. W. (1995). Fast effective rule induction. In *Proceedings of the twelfth international conference on international conference on machine learning, ICML'95* (pp. 115–123). San Francisco, CA: Morgan Kaufmann Publishers Inc.
- de Jong, H. (2002). Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1), 67–103.
- Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7, 1–30.
- Emmert-Streib, F., Glazko, G., De Matos Simoes, R., et al. (2012). Statistical inference and reverse engineering of gene regulatory networks from observational expression data. *Bioinformatics and Computational Biology*, 3, 8.
- Gallagher, B., & Eliassi-Rad, T. (2010). Leveraging label-independent features for classification in sparsely labeled networks: An empirical study. In L. Giles, M. Smith, J. Yen, & H. Zhang (Eds.), *Advances in Social Network Mining and Analysis* (pp. 1–19). Berlin: Springer.

- Geistlinger, L., Csaba, G., Dirmeier, S., Küffner, R., & Zimmer, R. (2013). A comprehensive gene regulatory network for the diauxic shift in *Saccharomyces cerevisiae*. *Nucleic Acids Research*, *41*(18), 8452–8463. <https://doi.org/10.1093/nar/gkt631>.
- Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, KDD '16* (pp. 855–864). New York, NY: ACM.
- Hase, T., Ghosh, S., Yamanaka, R., & Kitano, H. (2013). Harnessing diversity towards the reconstructing of large scale gene regulatory networks. *PLoS Computational Biology*, *9*(11), e1003361.
- Hecker, M., Lambeck, S., Toepfer, S., Van Someren, E., & Guthke, R. (2009). Gene regulatory network inference: Data integration in dynamic models—A review. *Biosystems*, *96*(1), 86–103.
- Hempel, S., Koseska, A., Nikoloski, Z., & Kurths, J. (2011). Unraveling gene regulatory networks from time-resolved gene expression data—A measures comparison study. *BMC Bioinformatics*, *12*(1), 292.
- Henderson, K., Gallagher, B., Li, L., Akoglu, L., Eliassi-Rad, T., Tong, H., & Faloutsos, C. (2011). It's who you know: Graph mining using recursive structural features. In *Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '11* (pp. 663–671). New York: ACM.
- Hsu, H. T. (1975). An algorithm for finding a minimal equivalent graph of a digraph. *Journal of ACM*, *22*(1), 11–16.
- Ibarguren, I., Lasarguren, A., Pérez, J. M., Muguerza, J., Gurrutxaga, I., & Arbelaitz, O. (2016). Bfpart: Best-first part. *Information Sciences*, *367–368*, 927–952.
- Itani, S., Ohannessian, M., Sachs, K., Nolan, G.P., & Dahleh, M.A. (2008). Structure learning in causal cyclic networks. In *Proceedings of the international conference on causality: objectives and assessment—Vol. 6, COA'08* (pp. 165–176) JMLR.org.
- Korb, K. B., & Nicholson, A. E. (2010). *Bayesian Artificial Intelligence* (2nd ed.). Boca Raton, FL: CRC Press Inc.
- Li, J., & Xie, D. (2015). Rack1, a versatile hub in cancer. *Oncogene*, *34*(15), 1890–1898.
- Lo, L., Wong, M., Lee, K., & Leung, K. (2015). Time delayed causal gene regulatory network inference with hidden common causes. *PLOS ONE*, *10*(9), 1–47.
- Lü, L., & Zhou, T. (2011). Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, *390*(6), 1150–1170.
- Marbach, D., Costello, J. C., Küffner, R., Vega, N. M., Prill, R. J., Camacho, D. M., et al. (2012). Wisdom of crowds for robust gene network inference. *Nature Methods*, *9*, 796–804.
- Margolin, A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., Favera, R., et al. (2006). Aracne: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, *7*(Suppl 1), S7.
- Markowitz, F., & Spang, R. (2007). Inferring cellular networks—A review. *BMC Bioinformatics*, *8*(Suppl 6), S5.
- Mikolov, T., Chen, K., Corrado, G. S., & Dean, J. (2013). Efficient estimation of word representations in vector space. CoRR arXiv:1301.3781
- Omrani, N., Eloundou-Mbebi, J. M. O., Mueller-Roebber, B., & Nikoloski, Z. (2016). Gene regulatory network inference using fused lasso on multiple data sets. *Scientific Reports*, *6*, 20533.
- Park, P. J. (2009). ChIP-seq: Advantages and challenges of a maturing technology. *Nature Reviews Genetics*, *10*(10), 669–680.
- Pearl, J. (2000). *Causality: Models, reasoning, and inference*. New York, NY: Cambridge University Press.
- Penfold, C. A., & Wild, D. L. (2011). How to infer gene networks from expression profiles, revisited. *Interface Focus*, *1*(6), 857–870.
- Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '14* (pp. 701–710). New York, NY: ACM.
- Pinna, A., Soranzo, N., & de la Fuente, A. (2010). From knockouts to networks: Establishing direct cause-effect relationships through graph analysis. *PLoS ONE*, *10*(5), e12912.
- Pio, G., Ceci, M., Malerba, D., & D'Elia, D. (2015). ComiRNet: A web-based system for the analysis of miRNA-gene regulatory networks. *BMC Bioinformatics*, *16*(9), S7.
- Pio, G., Ceci, M., Prisciandaro, F., & Malerba, D. (2017). LOCANDA: Exploiting causality in the reconstruction of gene regulatory networks. In A. Yamamoto, T. Kida, T. Uno, & T. Kuboyama (Eds.), *Discovery science 2017, Lecture notes in computer science* (Vol. 10558, pp. 283–297). Berlin: Springer.
- Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*(5500), 2323–2326.

- Selvanathan, S. P., Graham, G. T., Erkizan, H. V., Dirksen, U., Natarajan, T. G., Dakic, A., et al. (2015). Oncogenic fusion protein *ews-flt1* is a network hub that regulates alternative splicing. *Proceedings of the National Academy of Sciences*, *112*(11), E1307–E1316.
- Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web, WWW '15* (pp. 1067–1077). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland.
- Tenenbaum, J. B., Silva, V. D., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*(5500), 2319–2323.
- Thattai, M., & van Oudenaarden, A. (2001). Intrinsic noise in gene regulatory networks. *Proceedings of the National Academy of Sciences*, *98*(15), 8614–8619.
- Van den Bulcke, T., Van Leemput, K., Naudts, B., van Remortel, P., Ma, H., Verschoren, A., et al. (2006). SynTReN: A generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC Bioinformatics*, *7*, 43.
- Vilalta, R., & Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial Intelligence Review*, *18*(2), 77–95.
- Yu, D., Lim, J., Wang, X., Liang, F., & Xiao, G. (2017). Enhanced construction of gene regulatory networks using hub gene information. *BMC Bioinformatics*, *18*(1), 186.
- Zitnik, M., & Zupan, B. (2015). Data imputation in epistatic MAPs by network-guided matrix completion. *Journal of Computational Biology*, *22*(6), 595–608.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Affiliations

Gianvito Pio¹  · Michelangelo Ceci^{1,2,3}  · Francesca Prisciandaro¹  ·
Donato Malerba^{1,2} 

Michelangelo Ceci
michelangelo.ceci@uniba.it

Francesca Prisciandaro
f.prisciandaro1@studenti.uniba.it

Donato Malerba
donato.malerba@uniba.it

¹ Department of Computer Science, University of Bari Aldo Moro, Via Orabona, 4, 70125 Bari, Italy

² Big Data Laboratory, National Interuniversity Consortium for Informatics (CINI), Rome, Italy

³ Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia