



Fast approach for link prediction in complex networks based on graph decomposition

Abdelhamid Saifi^{1,3} · Farid Nouioua^{2,3,4} · Samir Akhrouf¹

Received: 4 May 2022 / Accepted: 15 February 2023 / Published online: 15 March 2023
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2023

Abstract

Social networks such as Facebook, Twitter, etc. have dramatically increased in recent years. These databases are huge and their use is time consuming. In this work, we present an optimal calculation in graph mining for link prediction to reduce the runtime. For that purpose, we propose a novel approach that operates on the connected components of a network instead of the whole network. We show that thanks to this decomposition, the results of all link prediction algorithms using local and path-based similarity measure scan be achieved with much less amount of computations and hence within much shorter runtime. We show that this gain depends on the distribution of nodes in components and may be captured by the Gini and the variance measures. We propose a parallel architecture of the link prediction process based on the connected components decomposition. To validate this architecture, we have carried out an experimental study on a wide range of well-known datasets. The obtained results clearly confirm the efficiency of exploiting the decomposition of the network into connected components in link prediction.

Keywords Link prediction · Social network · Parallel computing · Graph mining · Local information · Interaction mining · Complex networks

1 Introduction

Interaction mining in complex networks has increasingly attracted the attention of several researchers and has become the subject of many branches of science. Various real-world systems can be modeled as networks in evolving interactions

in the form of graphs (Newman 2003), such as computers, neural networks; chemical as well as online social networks (Zhou et al. 2009). In such networks, entities are represented by nodes or vertices, while edges between pairs of nodes describe interactions, associations, or relationships between nodes (Liao et al. 2015). Several efforts have been made to understand networks evolution (Albert and Barabasi 2002; Dorogovtsev and Mendes 2002), their relationships with both topologies and operation (Newman 2003; Boccaletti et al. 2006), as well as network characteristics (Costa et al. 2007).

Another important area of research in complex networks is interested in links between nodes and consists in studying the persistence of existing links or the prediction of new links when the networks evolves. Persistent links, also called durable links, stable links or permanent links, are links that exist in the past and will continue to exist in the future. On the other side, link prediction is related to detecting links that are missing in the past but that potentially will appear as new links in the future (Liben-Nowell and Kleinberg 2007).

Link prediction can be used in a variety of contexts: it has been used in bioinformatics to predict protein–protein interactions (PPI) (Yu et al. 2021). It is also used in security

✉ Abdelhamid Saifi
abdelhamid_saifi@yahoo.fr

Farid Nouioua
faridnouioua@gmail.com; farid.nouioua@lis-lab.fr

Samir Akhrouf
samir.akhrouf@gmail.com

¹ Department of Computer Science, Mohammed Boudiaf University of M'Sila, M'Sila, Algeria

² Department of Computer Science, Mohammed El bachir El Ibrahimi University of Bordj Bou Arreridj, Bordj Bou Arreridj, Algeria

³ LMSE Laboratory, Mohamed El Bachir El Ibrahimi University of Bordj Bou Arreridj, Bordj Bou Arreridj, Algeria

⁴ LIS Laboratory, UMR-CNRS 7020, Aix-Marseille University, Marseille, France

applications to identify hidden groups of terrorists and criminals (Hasan and Zaki 2011). It has been applied in health-care to predict the spread of epidemic diseases (Kamath et al. 2001) and in the development of strategies to immunize potentially affected people to limit the spread of the epidemic. Link prediction has been used in the development of road networks to improve routes (Liu et al. 2011), in online connected community networks where future associations can be suggested as probable friendships, which can help the system to recommend new friends and thus increase their reliability towards the service (Dorogovtsev and Mendes 2002). Another example of practical use of link prediction is on Amazon, and Alibaba products, where recommended movies on Netflix and advertisements are displayed to users on Google AdWords and Facebook (Ibrahim and Chen 2015). Most link prediction algorithms that have been proposed are similarity-based algorithms, i.e., they are based on a similarity measure which assigns a score to each pair of vertices in order to quantify its probability of existence. Similarity based algorithms are widely used due to their prediction accuracy and their computational efficiency (Yang and Zhang 2016). Similarity-based algorithms can be classified into two types: Algorithms based on local information of nodes (nearest neighbor) and algorithms based on global path information (shortest path). However, since traditional link prediction algorithms try to evaluate all the absent links in a network, they generally cannot effectively handle large-scale complex networks. It is necessary to look for new techniques and approaches able to reduce the amount of computations during link prediction in such networks in order to ensure a reasonable consumption of time and memory space. The aim of the present paper is to address this problem and propose such a new approach based on the decomposition of the initial network into connected components. The idea is that in similarity-based algorithms for link prediction, the links between nodes belonging to different components do not matter. Our approach only focuses on the calculation of local links, i.e., links inside each component.

Moreover, from the observation that the components may be treated separately, in our approach again, we propose a parallel architecture for link prediction which distributes the set of components on several processors.

The main contributions of this paper can be summarized as follows.

- In the first part of our contribution, we show the importance of our approach and the gain we obtain by using the decomposition into connected components instead of working directly on the original graph. We formally analyze this gain in terms of the reduction of the number of calculated links. This gain is also illustrated experimentally in terms of runtime. Moreover, a comparison between the baseline approach, to which belong all pre-

vious works, and our approach has been done using the same databases.

- The second part of our contribution is based on the remark that the decomposition allows us to parallelize the calculations. So, we have derived four variants of our algorithm, according to: (1) the use or not of parallelism and/or (2) the inclusion or not of the decomposition step into the calculation process. The experimental study shows that parallelization allows us to save additional time.

The reminder of the paper is organized as follows. We present in Sect. 2 an overview of various measures based on local or neighborhood data as well as measures based on the shortest paths between nodes. Section 3 gives a formal description of the link prediction problem and highlights the potential difficulties due to the large size of networks for link prediction. Then, it discusses how the decomposition into connected components can be very useful in reducing the amount of computations. In Sect. 4 we present our proposed architecture for a link prediction system based on the decomposition of the network into its connected components. In particular, the proposed architecture exploits the decomposition to ensure a parallel treatment by distributing the components on several processors. In Sect. 5, we present the baseline algorithm which does not use decomposition and four variants of our proposed approach, based on decomposition, depending on whether the decomposition step is included in the scenario or not and whether the treatment is sequential or parallel. Section 6 is devoted to an experimental study for the validation of our proposal. Finally, in Sect. 7, we conclude our work and give some perspectives for future work.

2 Related work

In many link prediction methods, scores are derived from the class of immediate node neighborhoods. In other methods, scores calculation is based on paths between nodes (Wang et al. 2015; Lü and Zhou 2011). Several research works are based on the use of these measures either to compare the performance of new approaches (Li et al. 2020) or to modify them (Yang and Zhang 2016; Ahmad et al. 2020) and propose new extensions (Aziz et al. 2020). Among these measures which are still popularly used, we can quote the following measures categorized in two classes: those based on local information and those used in path methods.

2.1 Similarity indices based on local information

There are numerous local similarity measures. They are basically determined by the number of common neighbors shared by two nodes, say x and y . Let us denote by $I(x)$ the

set of x 's neighbors and by $k(x)=|\Gamma(x)|$ the x 's degree. To each link (x, y) a score is assigned which is denoted by S_{xy} .

- Common neighbors (Liben-Nowell and Kleinberg 2007). This measure captures the common sense idea that two nodes are more likely to have a link if they have many common neighbors; this measure is defined as follows:

$$S_{xy}^{CN} = |\Gamma(x) \cap \Gamma(y)| \tag{1}$$

- Jaccard index (Liben-Nowell and Kleinberg 2007). Jaccard index normalizes the size of common neighbors. This measure is defined by:

$$S_{xy}^{CN} = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|} \tag{2}$$

- Adamic Adar index (Adamic and Adar 2003). This measure is calculated by adding weights to the nodes that are connected to both nodes x and y . It is defined by:

$$S_{xy}^{AA} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log k(z)} \tag{3}$$

- Resource allocation index (Zhou et al. 2009). Considering a pair of nodes x and y that are not directly connected and assuming that the node x needs to give some resources to y with their common neighbors as transmitters, this measure is defined by:

$$S_{xy}^{RA} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{k(z)} \tag{4}$$

- Node and Link Clustering coefficient (NLC) (Wu et al. 2016a). This similarity method is based on the basic topological characteristic of a network called "Clustering Coefficient". Clustering coefficients of nodes and links are incorporated to calculate similarity score. This measure is defined as follows:

$$S_{xy}^{NLC} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{S_{xz}^{CN}}{k_z - 1} * C(z) + \frac{S_{yz}^{CN}}{k_z - 1} * C(z) \tag{5}$$

- Clustering Coefficient for Link Prediction (Wu et al. 2016b). This method uses more information about the local link/triangle structure than the CAR index (Cannistraci et al. 2013), but costs less computation time. The key idea of the method is to exploit the value of links between other neighbors of common neighbors. It is defined as follows:

$$S_{xy}^{CCLP} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{t_z}{k(z)(k(z) - 1)/2} \tag{6}$$

such that: t_z is the number of triangles passing through node z .

- Common neighbor plus preferential attachment (CN + PA) index (Zeng 2016). This method is presented to estimate the probability of existence of a link between two nodes. Based on the combination of the two methods Common neighbor and preferential attachment (Mitzenmacher 2004). It is defined as follows:

$$S_{xy}^{CN+PA} = |\Gamma(x) \cap \Gamma(y)| + \alpha \frac{|\Gamma(x)| * |\Gamma(y)|}{\frac{\sum_{z \in V} |\Gamma(z)|}{|V|}} \tag{7}$$

where α is a parameter (e.g.: $\alpha = 0.01$) and $\frac{\sum_{z \in V} |\Gamma(z)|}{|V|}$ is equal to the mean degree $\langle k \rangle$ of the network.

- Adaptive degree penalization for link prediction (ADP) (Martínez et al. 2016). The measure finds that the degree of penalization which better obtains the results of the link prediction can be estimated by considering the clustering coefficient of the network. The prediction technique automatically adapts to the network. It is defined as follows:

$$S_{xy}^{ADP} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} |\Gamma(z)|^{-c\beta} \tag{8}$$

where c is the average clustering coefficient of the network and β is a constant (e.g. $\beta = 2.5$).

- Fuzzy modeling for link prediction (Tuan et al. 2019). Let $\mu_{\Gamma(x)}$ and $\mu_{\Gamma(y)}$ be the membership degrees of the sets of common neighbors $\Gamma(x)$ and $\Gamma(y)$ of the node x and y respectively, then:

- The fuzzy common neighbor similarity score is defined as:

$$S_{xy}^{FCN} = \sum_{x \in \Gamma(x), y \in \Gamma(y)} (\mu_{\Gamma(x)} \cap \mu_{\Gamma(x)}) \tag{9}$$

- The fuzzy Adamic–Adar Similarity Score is defined as:

$$S_{xy}^{FAA} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log(|\mu_{\Gamma(z)}|)} \tag{10}$$

where the membership function is $\mu_{\Gamma(z)}$.

- The fuzzy Jaccard Coefficient is given by:

$$S_{xy}^{FJC} = \sum_{x \in \Gamma(x), y \in \Gamma(y)} \frac{\mu_{\Gamma(x)} \cap \mu_{\Gamma(x)}}{\mu_{\Gamma(x)} \cup \mu_{\Gamma(x)}} \tag{11}$$

- Triangle Structure Index (TRA) (Bai et al. 2018). This method offers us a new similarity index, by combining the aforementioned triangle structure and the idea of the RA index. It is defined as follows:

$$S_{xy}^{TRA} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1 + \Delta(x, y; z)/2}{k(z)} \tag{12}$$

where Δ is the number of triangles TRA formed by x , y and z , which is: $\Delta(x, y; z) = \Delta(x, z) + \Delta(y, z)$.

- Sam Similarity (Samad et al. 2019). This measure refers to the similarity of x to y as well as the similarity of y to x . It divides the similarity task computation into two parts. First, it determines how much x is similar to y . Second, it computes the similarity between y and x . Finally, the measure is defined by using both outcomes, as follows:

$$S_{xy}^{Sam} = \frac{\frac{|\Gamma(x) \cap \Gamma(y)|}{\Gamma(x)} + \frac{|\Gamma(x) \cap \Gamma(y)|}{\Gamma(y)}}{2} \tag{13}$$

- Common Neighbor to Degree (CN2D) (Mumin et al. 2022). Consider two unconnected nodes a and b , and assuming that they have common neighbors between them. They are able to present and propagate some information through these neighbors in their interaction. The total resources allocated to the target node pair is estimated using the following equation:

$$S_{xy}^{CN2D} = |\Gamma(x) \cap \Gamma(y)| + \beta \left(\frac{1}{\max(k_x, k_y)} \sum_{z \in \Gamma(x) \cap \Gamma(y)} |\Gamma(z)| \right) \tag{14}$$

with a parameter β between 0 and 1, (e.g. $\beta=0.01$).

2.2 Similarity indices based on path methods

Paths between two nodes can also be used to compute similarities between couples of nodes. Such methods are referred to as path-based metrics. The following are some of the most popular methods in this class:

- Common Neighbor and Distance index (Yang and Zhang 2016). This is an extension of common neighbors. It is based on two properties of a complex network, common neighbor and the distance between two nodes x and y .

$$S_{xy}^{CND} = \begin{cases} \frac{CN_{xy} + 1}{2} \Gamma(x) \cap \Gamma(y) \neq \emptyset \\ \frac{1}{d_{xy}} \text{ otherwise} \end{cases} \tag{15}$$

where CN_{xy} is the number of common nodes between node x and y and d_{xy} is the distance between x and y .

- Newton’s Gravitational Law index (Ashraf et al. 2018). This measurement is inspired by Newton’s law of uni-

versal gravitation, which states that the force exerted between two masses is proportional to the product of these masses, and inversely proportional to the square of the distance between their centers. It is defined as follows:

$$S_{xy}^{NGLI} = \frac{C_D(x) * C_D(y)}{SP(x, y)} \tag{16}$$

where C_D denotes the degree of centrality, SP is the shortest path.

- Local major path degree (LMPD) (Yang et al. 2018). Local path between two nodes is equal to the sum of the degrees of the intermediate nodes. The local path designates the paths of two and three stages between two nodes, and this method is defined as follows:

$$S_{xy}^{RP} = \sum_{\omega=1}^{L_2} \frac{1}{(LPD_{\omega})^{\alpha}} + \sum_{m=1}^{L_3} \frac{1}{(LPD_m)^{\alpha}} \tag{17}$$

where $LPD_i = \sum_{j=1}^{n_i} k_j$.

- Relative-path-based algorithm for link prediction (RP) (Li et al. 2020). This method assumes that paths between nodes and neighbors provide basic similarity characteristics. This method uses factorial information about paths between nodes and neighbors, in addition to paths between pairs of nodes, in the similarity calculation for link prediction. It is defined as follows:

$$S_{xy}^{RP} = \alpha \frac{S_{x,y}^{DP_2}}{\frac{1}{2} \left(S_{x,N_x}^{DP_2} + S_{y,N_y}^{DP_2} \right) + 1} + \beta \frac{S_{x,y}^{DP_3}}{\frac{1}{2} \left(S_{x,N_x}^{DP_3} + S_{y,N_y}^{DP_3} \right) + 1} \tag{18}$$

where $S_{x,y}^{DP_2} = \sum_{z_1, z_2 \in P_2(x,y)} \frac{1}{k_{z_1} k_{z_2}}$ et $S_{x,y}^{DP_3} = \sum_{z_1, z_2 \in P_3(x,y)} \frac{1}{k_{z_1} k_{z_2}}$, $P_2(x,y)$ et $P_3(x,y)$ represent all intermediate nodes on paths of length 2 and 3 respectively between nodes x and y .

- Common neighbor centrality index (Ahmad et al. 2020). This measure is based on two vital properties of nodes, namely the number of common neighbors and their centrality. Common neighbor refers to the common nodes between two nodes. Centrality refers to the prestige that a node enjoys in a network, it is defined by:

$$S_{xy}^{CNC} = \alpha (|\Gamma(x) \cap \Gamma(y)|) + (1 - \alpha) \cdot \frac{n}{d_{xy}} \tag{19}$$

where α is a parameter that varies in the interval $[0, 1]$ and d_{xy} denotes the shortest distance between x and y .

3 Using decomposition to improve link prediction performance

The aim of this section is to formally define the problem of link prediction and study the impact of using as input the connected components of a network instead of the whole network on the amount of necessary computation and consequently on the runtime.

Definition 1 (Complex Network). A complex network is defined by a non-oriented graph $G(V,E)$ where V is a set of nodes and $E \subseteq V \times V$ is a set of links between pairs of nodes.

We put $n=|V|$ (number of nodes) and $e=|E|$ (number of links) and we denote by E' the set of absent links in the graph: $E' = \{(x,y) \in V^2 | (x,y) \notin E\}$.

Notice that if G is a complete graph (where there is a link between every pair of nodes), then the number of links is $n.(n-1)/2$.

In general, the size of E' which represents the set of non-existing links in the network is $n.(n-1)/2 - e$.

Existing link prediction algorithms based on local and path measure compute the scores for all the non-existing links belonging to E' since they consider that any current non existing link may become an actual link in the future. Accordingly, for very large datasets, the number of potential links to evaluate may be very important. For example, in twitter-dataset (Reza and Huan 2009), we have 11.316.811 nodes and 85.331.846 links and then, the number of scores that should be calculated for non-existing links is more than 64 trillions!!

To overcome this problem, the solution we propose is based on the decomposition of our graph into its connected components. Indeed, as it will be shown later, for all approaches based on local and path measures, we only need to compute scores between nodes that both belong to the same component. This allows one to save runtime by reducing the number of required calculation operations. Moreover, the fact that connected components can be treated independently, allows one to perform parallel computation by affecting to each available processor a subset of components.

Definition 2 (Connected components). Let $G=(V,E)$ be a graph representing a complex network.

A set $C = \{G_1=(V_1,E_1), G_2=(V_2,E_2), \dots, G_c=(V_c,E_c)\}$ such that each $G_i=(V_i,E_i)$ is a sub-graph of G defines a decomposition of G into its connected components if:

- $\bigcup_{i=1}^c V_i = V$ and $\bigcup_{i=1}^c E_i = E$
- $\forall i \in \{1, \dots, c\}, \forall x, y \in V_i$ there is a path relating x and y

- $\forall i, j \in \{1, \dots, c\}$, such that $i \neq j, V_i \cap V_j = \emptyset$ and $E_i \cap E_j = \emptyset$

Suppose that $\Gamma(x)$ denotes the set of neighbors of the node x , i.e., nodes having direct links with x , $paths_{x,y}$ denotes the set of all paths of links relating x and y , and $d_{x,y}$ denotes the distance between nodes x and y , i.e., the length of the shortest path between x and y . The following proposition states that nodes belonging to distinct components do not share any neighbors and are not related by any paths.

Proposition 1 Let $C = \{G_1=(V_1,E_1), G_2=(V_2,E_2), \dots, G_c=(V_c,E_c)\}$ be the set of connected components of a complex network $G=(V,E)$. We have:

$$\forall i, j \in \{1, \dots, c\} s.t. i \neq j, \forall x \in V_i, \forall y \in V_j \left\{ \begin{array}{l} \Gamma(x) \cap \Gamma(y) = \emptyset \text{ (hence } |\Gamma(x) \cap \Gamma(y)| = 0) \\ \text{and} \\ paths_{x,y} = \emptyset \text{ (hence } d_{xy} = \infty) \end{array} \right. \quad (20)$$

Proof For any x, y such that $x \in V_i, y \in V_j$ and $i \neq j$ we have: $\Gamma(x) \subseteq V_i$ and $\Gamma(y) \subseteq V_j$ (all the neighbors of x (resp. y) are inside V_i (resp. V_j)). Then, since $V_i \cap V_j = \emptyset$. It follows that $\Gamma(x) \cap \Gamma(y) = \emptyset$. This immediately leads to the fact that $|\Gamma(x) \cap \Gamma(y)| = 0$.

Now, if we suppose that there is a path relating x and y then there must be a path between some x' and y' such that $x' \in V_i$ and $y' \in V_j$ which is impossible since $V_i \cap V_j = \emptyset$ and $E_i \cap E_j = \emptyset$. Hence we obtain $paths_{x,y} = \emptyset$. Since there is no path between x and y , then it follows immediately that $d_{xy} = \infty$.

Corollary 1 Given a local or a path-based similarity measure f we have:

$$\forall i, j \in \{1, \dots, c\} s.t. i \neq j, \forall x \in V_i, \forall y \in V_j : f(x, y) = 0 \quad (21)$$

Proof Follows immediately from Proposition 1 and the definitions of local and path-based similarity measures.

This corollary is important since it states that it suffices to only compute the similarity measure between couples of nodes belonging to the same component.

Now let $Gain(G)$ be the gain obtained by using the decomposition of G into its connected components. $Gain(G)$ is the number of couples of nodes (x,y) where x and y belong to two different components.

Proposition 2 Let $C = \{G_1=(V_1,E_1), G_2=(V_2,E_2), \dots, G_c=(V_c,E_c)\}$ be the set of connected components of a social network $G=(V,E)$. We put $|V|=n, |V_i|=n_i$ for $i \in \{1, \dots, c\}$ and $|E|=e$. We have:

$$Gain(G) = \sum_{i=1}^{c-1} \sum_{j=i+1}^c (n_i * n_j) \tag{22}$$

$$scores(G) = \bigcup_{i=1}^c \left\{ (x, y, s_{xy}^{method}) / x, y \in V_i \text{ and } (x, y) \notin E_i \right\} \tag{25}$$

Proof Let $NEval(G)$ (resp. $NEval'(G)$) be the number of evaluated potential links by using decomposition (resp. without using decomposition). Then, $Gain(G) = NEval'(G) - NEval(G)$.

$$NEval(G) = \sum_{i=1}^c \left(\frac{n_i(n_i - 1)}{2} - e_i \right)$$

$$= \frac{1}{2} \sum_{i=1}^c n_i(n_i - 1) - \sum_{i=1}^c e_i$$

$$= \frac{1}{2} \sum_{i=1}^c (n_i^2 - n_i) - e$$

The number of evaluated links is:

$$|E'| - Gain(G) = \sum_{i=1}^c |E'_i| \tag{26}$$

For all the other non-existing links, the similarity measure is set to 0:

$$\forall i, j, s.t. i \neq j, \forall i \in V_i, \forall j \in V_j, s_{xy}^{method} = 0 \tag{27}$$

Let us consider some special cases:

- **Case of one connected component** ($c = 1$). In this case, the decomposition does not matter and $Gain(G) = 0$.

$$NEval'(G) = \frac{n(n-1)}{2} - e = \frac{1}{2}(n^2 - n) - e = \frac{1}{2}((n_1 + \dots + n_c)^2 - (n_1 + \dots + n_c)) - e$$

$$= \frac{1}{2}((n_1^2 + n_2^2 + \dots + n_c^2) + 2(n_1n_2 + \dots + n_1n_c + n_2n_3 + \dots + n_{c-1}n_c) - (n_1 + \dots + n_c)) - e$$

$$= \frac{1}{2} \left(\sum_{i=1}^c n_i^2 + 2 \sum_{i=1}^{c-1} \sum_{j=i+1}^c n_i n_j - \sum_{i=1}^c n_i \right) - e$$

$$= \frac{1}{2} \sum_{i=1}^c (n_i^2 - n_i) - e + \sum_{i=1}^{c-1} \sum_{j=i+1}^c n_i n_j = NEval(G) + \sum_{i=1}^{c-1} \sum_{j=i+1}^c n_i n_j$$

It follows that:

$$Gain(G) = \sum_{i=1}^{c-1} \sum_{j=i+1}^c (n_i * n_j)$$

Without using decomposition, all similarity indices based on local information and path methods calculate a score list (denoted $scores'(G)$) of all non-existent links in the graph:

$$scores'(G) = \left\{ (x, y, s_{xy}^{method}) / x, y \in V \text{ and } (x, y) \notin E \right\} \tag{23}$$

The number of evaluated links is:

$$|E'| = Gain(G) + \sum_{i=1}^c |E'_i| = \sum_{i=1}^{c-1} \sum_{j=i+1}^c (n_i * n_j) + \sum_{i=1}^c |E'_i| \tag{24}$$

After decomposition, this score list (denoted $scores(G)$) becomes:

- **Many components** ($c > 1$). If there is more than one component, the gain is strictly positive. We can distinguish the two following particular cases:

- *Case where $n_1 \approx n_2 \approx \dots \approx n_c \approx n_c$* . In this case the gain is maximal and it is given by:

$$Gain^{max}(G) = \sum_{i=1}^{c-1} \sum_{j=i+1}^c \left(\frac{n}{c} \right)^2$$

$$= \frac{c(c-1)}{2} \left(\frac{n}{c} \right)^2$$

$$= \frac{n^2(c-1)}{2c}$$

- *Case where $n_1 = (n - c + 1), n_2 = n_3 = \dots = n_c = 1$* . This case corresponds to the situation where each $c - 1$ components contains one node, and one component contains all the remaining $n - c + 1$ nodes. In this case the gain is minimal, and is given by:

$$\begin{aligned}
 \text{Gain}^{\min}(G) &= \sum_{i=1}^{c-1} \sum_{j=i+1}^c (n_i n_j) \\
 &= \sum_{j=2}^c n_1 * 1 + \sum_{i=2}^{c-1} \sum_{j=i+1}^c (1 * 1) \\
 &= \sum_{j=2}^c (n - c + 1) + \sum_{i=2}^{c-1} \sum_{j=i+1}^c (1) \\
 &= (c - 1)(n - c + 1) + \frac{(c - 1)(c - 2)}{2} \\
 &= \frac{(c - 1)(2n - c)}{2}
 \end{aligned}$$

Example 1 Let us consider the graph $G(V,E)$ depicted in Fig. 1a with $|V|=n=9$ nodes and $|E|=e=8$ edges. G contains $c=3$ connected components. Without using decomposition, to predict future links, we need to calculate the score of all nonexistent links (28 links) as shown in Fig. 1b.

By using the decomposition of G into its connected components, instead of calculating the scores of the links that do not exist in the complete graph, we only calculate the scores of the links that do not exist inside the three components $(c_1), (c_2)$ and (c_3) (see Fig. 1c). We have 0 links in (c_1) , 0 links in (c_2) and 2 links in (c_3) . The obtained gain is 26.

Example 2 We assume that $G(V, E)$ is a training graph with $|V|=n=9$ nodes. The number of connected components is $c=3$. Table 1 illustrates the different possible cases of the nodes distribution on the components and the gain associated with each of these distributions.

We can notice that the gain clearly depends on the distribution of the nodes in the connected components: the more the distribution is homogeneous, i.e., where the nodes are more uniformly distributed over the components, the greater is the gain. By contrast, heterogeneous distributions lead to less important gains.

Now, we show that this idea may be captured by two further measures:

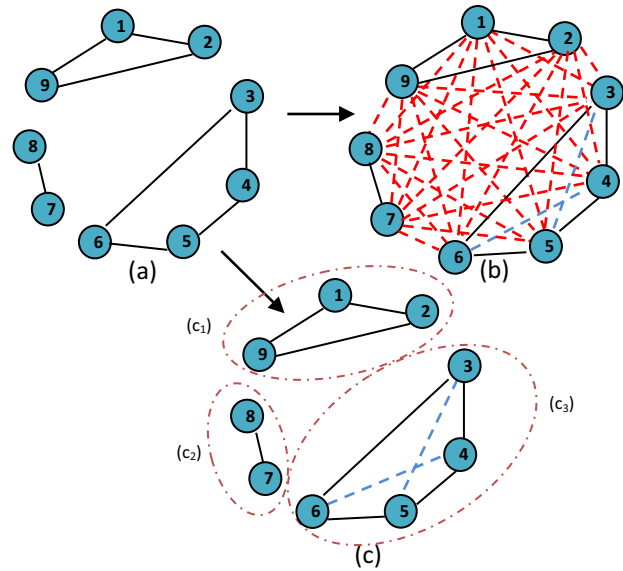


Fig. 1 The gain in terms of the number of calculated links by using decomposition illustrated on a simple undirected network

- **The Gini coefficient** is a measure that captures the inequality or the heterogeneity of a distribution. It used in different domains, namely in supervised learning for the construction of decision trees (Breiman et al. 1984; Daniya et al. 2020). For a decomposition $C = \{G_1 = (V_1, E_1), G_2 = (V_2, E_2), \dots, G_c = (V_c, E_c)\}$ of a network $G = (V, E)$, the Gini coefficient is defined by the following formula:

$$\text{Gini}(G) = 1 - \sum_{i=1}^c p^2(n_i) \tag{28}$$

where $P(n_i)$ is the proportion of nodes present in the component $G_i(V_i, E_i)$ and it is determined as follows:

$$p(n_i) = \frac{|V_i|}{|V|} = \frac{n_i}{n} \tag{29}$$

- **The variance** is a statistical measure of the dispersion of the values of a distribution. It is calculated as follows:

Table 1 Various possible distributions of nodes on the components and their calculation results of *Gain*, *Gini* and *Variance*

n°	n_1	n_2	n_3	Var	Var ^{Norm}	Gini	Gini ^{Norm}	Gain	Gain ^{Norm}
1	7	1	1	8.00	1.00	0.37	0.00	15	0.00
2	6	2	1	4.67	0.58	0.49	0.42	20	0.42
3	5	1	3	2.67	0.33	0.57	0.67	23	0.67
4	5	2	2	2.00	0.25	0.59	0.75	24	0.75
5	4	4	1	2.00	0.25	0.59	0.75	24	0.75
6	4	3	2	0.67	0.08	0.64	0.92	26	0.92
7	3	3	3	0.00	0.00	0.67	1.00	27	1.00

$$Var(G) = \frac{\sum_{i=1}^c |n_i - m|^2}{c} \tag{30}$$

where m is the mean value of the distribution, i.e., $m = n/c$.

The three measures (Gain, Gini and Variance) can be normalized to take their values in $[0, 1]$. For that purpose, we use the following formula:

$$X^{norm} = \frac{X - X^{min}}{X^{max} - X^{min}} \in [0, 1] \tag{31}$$

where $X \in \{Gain, Gini, Var\}$, X^{norm} (resp. X^{max} , X^{min}) is the normalized (resp. maximal, minimal) value of X . Namely, we can check that:

$$Gini^{max}(G) = 1 - \frac{1}{c} \text{ and } Gini^{min}(G) = \frac{(c-1)(2n-c)}{n^2}$$

$$Var^{max}(G) = \frac{(c-1)(n-c)^2}{c^2} \text{ and } Var^{min}(G) = 0$$

$$Gain^{max}(G) = \frac{n^2(c-1)}{2c} \text{ and } Gain^{min}(G) = \frac{(c-1)(2n-c)}{2}$$

Table 1 gives the directed and normalized values of the three measures (Gain, Gini and Variance) for the different possible distributions in our example. Figure 2 graphically depicts the obtained results.

Interestingly enough, we can notice that the normalized gain and the normalized Gini coefficient coincide ($Gain^{norm}(G) = Gini^{norm}(G)$). In contrast, the normalized variance is inversely proportional to the normalized Gini and Gain, and more precisely: $Var^{norm}(G) = 1 - Gain^{norm}(G)$.

4 The proposed architecture

In this paper, we propose an architecture for the link prediction problem which is based on six main steps instead of four steps (Ying et al. 2014), namely: Data gathering, preprocessing, decomposition, parallelization, prediction and evaluation. Our architecture is depicted in Fig. 3.

– **Data gathering.** This step consists of preparing the datasets corresponding to a network, into training and test subsets, the complete dataset is divided according to one of the two following cases:

- The first principle relies on the evolution time of the dataset (Liben-Nowell and Kleinberg 2007). Since each interaction takes place in a particular instance, we define a training interval $[t_0, t'_0]$ which in turn defines the subgraph $G^{Tr} = (V^{Tr}, E^{Tr})$ containing all the interactions performed between the instants t_0 and t'_0 . Similarly, another test interval $[t_1, t'_1]$ defines a test sub-graph $G^{Ts} = (V^{Ts}, E^{Ts})$ containing all the interactions performed between the instants t_1 and t'_1 . This

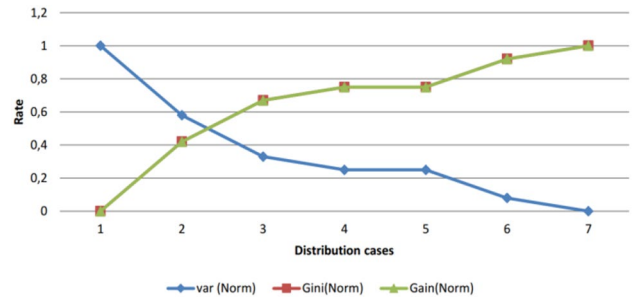


Fig. 2 Gain, Gini and Variance rates according to the distribution of the nodes on the components

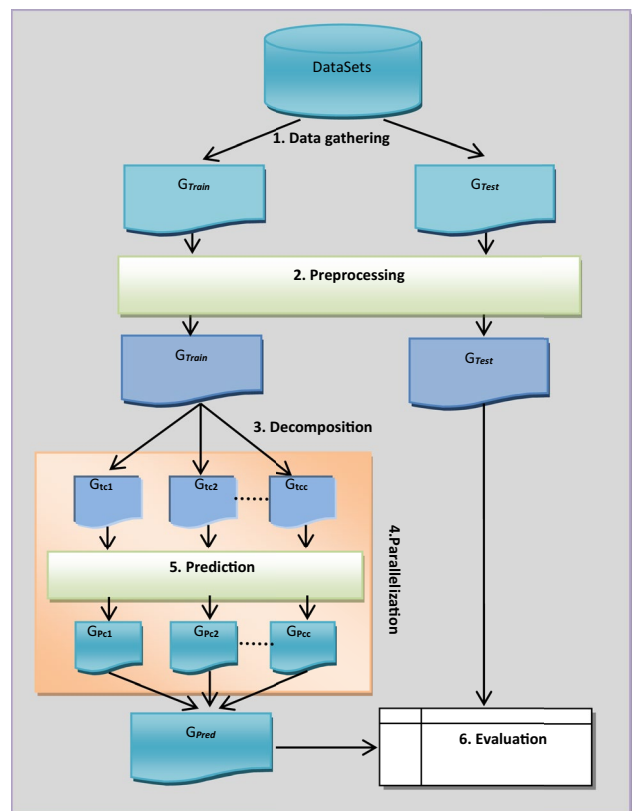


Fig. 3 Parallel architecture for link prediction based on connected components decomposition

- principle is used to predict the likelihood of future links (in dynamic networks (Kumar et al. 2020)).
- Following the second principle, we randomly divide the set of links of the dataset into two subsets, namely, 90% (Lü and Zhou 2011; Xiao and Zhang 2015; Wang et al. 2017; Daminelli et al. 2015) [80% (Ahmad et al. 2020; Wu 2018), 70% (Sui et al.

2013)] of links in a training sub-graph $G^{Tr} = (V^{Tr}, E^{Tr})$ and 10% (20%, 30%) of links in a test subset $G^{Ts} = (V^{Ts}, E^{Ts})$ where $E = E^{Tr} \cup E^{Ts}$ and $E^{Tr} \cap E^{Ts} = \emptyset$. This principle is used to find the missing links [in static networks (Kumar et al. 2020)].

- **Preprocessing.** This step consists of filtering some nodes of our network by:
 - Eliminating the isolated nodes (Lu et al. 2009) since it is difficult to predict anything about isolated nodes because one has no network-based information.
 - Discarding sometimes the low-degree nodes (Liben-Nowell and Kleinberg 2007; Lu et al. 2009; Scripps et al. 2009) because one has too little information about them.
 - Converting a directed network into an undirected one and subsequently, applying link prediction (Shibata et al. 2012).
 - Restricting the analysis to nodes that are common to the train and test networks (Lu et al. 2009) and eliminating the other nodes.
- **Decomposition.** In this step we decompose the training graph $G = (V, E)$ into its connected components:

$$C = \{G_1 = (V_1, E_1), G_2 = (V_2, E_2), \dots, G_c = (V_c, E_c)\}.$$
- **Parallelization.** The connected components are distributed through several processors. Each processor receives one or more components. We use the term local nodes to denote the nodes that each processor receives.

- **Prediction.** Each component is locally evaluated with its local nodes, and a partial score list is established. At this stage, only one similarity measure is used. In most cases, we want to predict only new links, i.e., links that are not present in the training component. At the end of this operation, all the local list scores are merged into a global list score
- **Evaluation.** Finally, it becomes possible to evaluate the performance of the prediction results by comparing them with the test subsets. We use different thresholds, parameters and performance measures.

5 Algorithms

In this section we present and compare, in terms of runtime, different scenarios that can be obtained by considering different choice criteria: whether we exploit or not the decomposition of the graph into connected component; whether the algorithm integrates the decomposition step, or directly receives as input the set of connected components, and whether the execution is sequential or parallel. This leads to five scenarios (Algo1–Algo5) explained in the following:

5.1 Without decomposition (Baseline method)

Algo 1. This algorithm corresponds to the baseline scenario, which corresponds to existing works, where the input is the whole graph, no decomposition is done, and of course the execution is sequential.

Algorithm 1 Without Decomposition (Baseline method)

Require: $G = (V, E)$: a network.

Ensure: $List_of_scores(G)$.

1: compute the scores of all absent links in G :

$$List_of_Scores(G) = \{(x, y, s_{xy}^{method}) / x, y \in V \text{ and } (x, y) \notin E\}$$

2: return $List_of_Scores(G)$;

5.2 With decomposition (our approach)

In this part we propose four new variants of algorithms which exploit the decomposition (see Table 2): Algo2 is the basic algorithm of our approach which will be compared with the baseline algorithm (Algo1). The other algorithms

(Algo3–Algo5) are improvements of our basic algorithm (Algo2). The proposed improvements are the following:

- Instead of considering the decomposition step inside the algorithm of link prediction, it suffices to perform it once and start the algorithm from the connected components.

Table 2 The four variants of algorithms derived from our decomposition-based approach

Type of execution	With decomposition step	
	Included	Not included
Sequential execution	Algo2	Algo3
Parallel execution	Algo4	Algo5

This avoids the useless repetitive execution of the same decomposition step in several executions.

- Since each component may be handled separately from the others, one can parallelize the algorithm instead of using a sequential treatment.

Algo3 (resp. Algo4, Algo5) results from Algo2 by integrating the first improvement (resp. second improvement, both improvements).

Algo2. This algorithm includes the decomposition of the input graph into connected components. Then, the prediction task is sequentially executed on the set of connected components.

Algorithm 2 Decomposition & Sequential Execution 1

Require: $G = (V, E)$: a network .

Ensure: $List_of_scores(G)$.

- 1: Decompose the graph $G = (V, E)$ into c connected components :

$$C \leftarrow \{G_1 = (V_1, E_1), G_2 = (V_2, E_2), \dots, G_c = (V_c, E_c)\}$$

- 2: For every component $G_i = (V_i, E_i)$, compute the scores of absent links:

$$Scores(G_i) = \{(x, y, s_{xy}^{method}) / x, y \in V_i \text{ and } (x, y) \notin E_i\}$$

- 3: Merge the partial scores to form the global list of scores:

$$List_of_Scores(G) = \bigcup_{i=1}^c Scores(G_i) = \bigcup_{i=1}^c \{(x, y, s_{xy}^{method}) / x, y \in V_i \text{ and } (x, y) \notin E_i\}$$

- 4: Return $List_of_Scores(G)$;
-

Algo3. This algorithm is similar to **Algo2**. but it does not include the decomposition step, but directly receives as input the set of connected components. The prediction task is then

sequentially executed on the set of connected components too. **Algo3**. is obtained from **Algo2**. by modifying the input to be the set of connected components and by eliminating line 1.

Algorithm 3 Decomposition & Sequential Execution 2

Require: $C = \{G_1 = (V_1, E_1), G_2 = (V_2, E_2), \dots, G_c = (V_c, E_c)\}$: a set of connected components of a network $G = (V, E)$.

Ensure: $List_of_scores(G)$.

- 1: For every component $G_i = (V_i, E_i)$, compute the scores of absent links:

$$Scores(G_i) = \{(x, y, s_{xy}^{method}) / x, y \in V_i \text{ and } (x, y) \notin E_i\}$$

- 2: Merge the partial scores to form the global list of scores:

$$List_of_Scores(G) = \bigcup_{i=1}^c Scores(G_i) = \bigcup_{i=1}^c \{(x, y, s_{xy}^{method}) / x, y \in V_i \text{ and } (x, y) \notin E_i\}$$

- 3: Return $List_of_Scores(G)$;
-

Table 3 Illustration of properties of 22 real world networks

Name	lvl	E	$\langle k \rangle$	C	ASPL	D	H	R
ADV	5155	39,285	15.24	0.25	3.22	9	5.4060	- 0.0951
BUP	105	441	8.40	0.49	3.08	7	1.4207	- 0.1279
CDM	16,264	47,594	5.85	0.64	5.82	18	2.2087	0.1846
CEG	297	2148	14.46	0.29	2.46	5	1.8008	- 0.1632
CGS	6158	11,898	3.86	0.49	3.62	14	3.9467	0.2426
EML	1133	5451	9.62	0.22	3.61	8	1.9421	0.0782
ERD	6927	11,850	3.42	0.12	3.78	4	12.6708	- 0.1156
FBK	4024	87,887	43.68	0.59	3.98	13	2.4320	0.0707
GRQ	5241	14,484	5.53	0.53	5.05	17	3.0523	0.6593
HMT	2426	16,630	13.71	0.54	3.15	10	3.1011	0.0474
HPD	8756	32,331	7.38	0.11	4.19	14	4.5133	- 0.0510
HTC	7610	15,751	4.14	0.49	5.68	19	2.0986	0.0293
INF	410	2765	13.49	0.46	3.63	9	1.3876	0.2258
KHN	3772	12,718	6.74	0.25	3.63	12	9.4220	- 0.1205
LDG	8324	41,532	9.98	0.31	4.37	16	6.1880	- 0.0997
NSC	1461	2742	3.75	0.69	2.59	17	1.8486	0.4616
PGP	10,680	24,316	4.55	0.27	7.49	24	4.1465	0.2382
SMG	1024	4916	9.6	0.31	2.98	6	3.9475	- 0.1925
UAL	332	2126	12.81	0.63	2.74	6	3.4639	- 0.2079
UPG	4941	6564	2.67	0.08	18.99	46	1.4504	0.0035
YST	2284	6646	5.82	0.13	4.29	11	2.8479	- 0.0991
ZWL	6651	54,182	16.29	0.32	3.85	10	2.5851	0.0006

$|V|$ denotes the number of nodes, $|E|$ is the number of edges, $\langle k \rangle$ denotes the average degree, C represents the clustering coefficient, $ASPL$ is the average shortest path length, D is the graph density, H is the heterogeneity indice and R s the assortativity degree

Algo4. Now, this algorithm includes the decomposition of the input graph into connected components. Then, the

prediction task is performed by a parallel execution on the set of connected components.

Algorithm 4 Decomposition & Parallel Execution 1

Require: $G = (V, E)$: a network.

Ensure: $List_of_scores(G)$.

1: Decompose the graph $G = (V, E)$ into c connected components :

$$C \leftarrow \{G_1 = (V_1, E_1), G_2 = (V_2, E_2), \dots, G_c = (V_c, E_c)\}$$

2: Find an equitable distribution of the set of components on the set of available m processors: $Proc_1, \dots, Proc_m$.

3. Parallel computation of the scores of absent links in each component:

$$Scores(G_i) = \{(x, y, s_{xy}^{method}) / x, y \in V_i \text{ and } (x, y) \notin E_i\}$$

$Scores(G_i)$ is computed in processor $proc_j$ to which the component G_i is affected.

4: Merge the partial scores to form the global list of scores:

$$List_of_Scores(G) = \bigcup_{i=1}^c Scores(G_i) = \bigcup_{i=1}^c \{(x, y, s_{xy}^{method}) / x, y \in V_i \text{ and } (x, y) \notin E_i\}$$

5: Return $List_of_Scores(G)$;

Table 4 Various possible distributions of nodes on the components and their calculation results of gain, Gini and variance

Datasets train			Without Decomposition	With Decomposition		Gain ₁		Rate (Norm)		
G	V _{Tr}	E _{Tr}	E' _{Tr}	C	∑ E' _{c,Tr}	Nb/link	Rate _{gain1} %	Gain ₂	Var	Gini
BUP	105	352	5108	1	5108	0	–	0.00	0.00	0.00
CEG	297	1718	42,238	1	42,238	0	–	0.00	0.00	0.00
FBK	4024	70,310	8,023,966	27	7,867,859	156,107	1.95	0.01	0.99	0.01
INF	410	2212	81,633	5	80,003	1630	2.00	0.00	1.00	0.00
ZWL	6651	43,346	22,071,229	114	21,241,223	830,006	3.76	0.00	1.00	0.00
SMG	1024	3932	519,844	35	483,648	36,196	6.96	0.00	1.00	0.00
EML	1133	4360	636,918	48	583,711	53,207	8.35	0.00	1.00	0.00
LDG	8324	33,225	34,607,101	337	31,089,311	3,517,790	10.16	0.02	0.98	0.02
UAL	332	1701	53,245	14	47,764	5481	10.29	0.03	0.97	0.03
KHN	3772	10,175	7,101,931	218	6,123,668	978,263	13.77	0.03	0.97	0.03
ADV	5155	31,428	13,253,007	330	11,333,166	1,919,841	14.49	0.02	0.98	0.02
HPD	8756	25,865	38,303,525	669	31,436,609	6,866,916	17.93	0.04	0.96	0.04
YST	2284	5316	2,601,870	202	2,068,432	533,438	20.50	0.04	0.96	0.04
ERD	6927	9480	23,978,721	1045	17,245,416	6,733,305	28.08	0.00	1.00	0.00
CDM	16,264	38,075	132,212,641	1288	87,661,470	44,551,171	33.70	0.22	0.78	0.22
HMT	2426	13,304	2,928,221	201	1,907,080	1,021,141	34.87	0.23	0.77	0.23
PGP	10,680	19,453	57,006,407	1230	35,945,368	21,061,039	36.95	0.19	0.81	0.19
UPG	4941	5275	12,198,995	458	7,494,909	4,704,086	38.56	0.25	0.75	0.25
GRQ	5241	11,588	13,719,832	652	7,387,831	6,332,001	46.15	0.30	0.70	0.30
HTC	7610	12,601	28,939,644	1022	14,331,576	14,608,068	50.48	0.34	0.66	0.34
CGS	6158	9519	18,947,884	1305	5,437,329	13,510,555	71.30	0.54	0.46	0.54
NSC	1461	2193	1,064,337	329	61,890	1,002,447	94.19	0.90	0.10	0.90

Algo5. This algorithm is similar to **Algo4**. but it does not include the decomposition step but directly receives as input the set of connected components. The prediction task

is then executed in parallel on the m processors. **Algo5**. is obtained from **Algo4**. by modifying the input to be the set of connected components and by eliminating line 1.

Algorithm 5 Decomposition&ParallelExecution 2

Require: $C = \{G_1 = (V_1, E_1), G_2 = (V_2, E_2), \dots, G_c = (V_c, E_c)\}$: a set of connected components of a network $G = (V, E)$.

Ensure: $List_of_scores(G)$.

1: Find an equitable distribution of the set of components on the set of available m processors: $Proc_1, \dots, Proc_m$.

2. Parallel computation of the scores of absent links in each component:

$$Scores(G_i) = \{(x, y, s_{xy}^{method}) / x, y \in V_i \text{ and } (x, y) \notin E_i\}$$

$Scores(G_i)$ is computed in processor $proc_j$ to which the component G_i is affected.

3: Merge the partial scores to form the global list of scores:

$$List_of_Scores(G) = \bigcup_{i=1}^c Scores(G_i) = \bigcup_{i=1}^c \{(x, y, s_{xy}^{method}) / x, y \in V_i \text{ and } (x, y) \notin E_i\}$$

4: Return $List_of_Scores(G)$;

6 Experimental results

6.1 Datasets

We have performed our experiments on several datasets representing different real-world complex networks from different sources and application domains. This collection covers a wide range of properties, sizes, average degrees, clustering coefficients and heterogeneity indices (see Table 3). This collection is freely downloadable from the address: <http://noesis.ikor.org/datasets/linkprediction>

- UPG, (Watts and Strogatz 1998) is a power distribution network.
- HPD, (Peri et al. 2003), YST, (Bu et al. 2003) and CEG, (Watts and Strogatz 1998) are biological networks.
- ERD, KNH, LDG, SMG, ZWL, CGS, (Batagelj and Mrvar 2006), HTC, CDM, (Newman 2001), NSC, (Newman 2006) and GRQ, (Leskovec et al. 2007) are co-authorship networks for different fields of study.
- HMT, (Kunegis 2013), FBK, (McAuley and Leskovec 2012), and ADV, (Massa et al. 2009) are social networks.
- UAL, (Massa et al. 2009) is an airport traffic network.
- EML, (Guimera et al. 2003) is a network of individuals who shared emails.
- PGP, (Boguña et al. 2004) is an interaction network of users of the Pretty Good Privacy algorithm.
- BUP, (Kunegis 2013) is a network of political blogs.
- Finally, INF, (Isella et al. 2011) is a network of face-to-face contacts in an exhibition.

6.2 Results and discussion

To demonstrate the effectiveness of our idea on real-world datasets, we have randomly divided the existing links of each network into two sets: The training set and the test set representing 80% and 20% of the total links, respectively. We have used 22 different datasets in the experiments to evaluate the performance of the different methods proposed in this study.

All the calculations have been performed on a machine of type HP Z620 Workstation with processor of model: Intel (R) Xeon (R) CPU E5-2620 v2 12cores of frequency 2.10 GHz and a memory size of 128 GB.

6.2.1 Impact of decomposition on the number of calculated links in different datasets

The first part of our experiments is devoted to evaluate the gain obtained for the different used datasets. Table 4

summarizes the obtained results. We notice that $Gain_1$ denotes the difference between the total number of links calculated without decomposition and those calculated by using decomposition. The gain is expressed in Table 4 both in terms of the number of treated links and in terms of their rate with respect to the total number of links in the train input graph. The corresponding value are put in bold font:

$$Gain_1 = |E'_{Tr}| - \sum_{i=1}^c |E'_{i,Tr}| \quad (32)$$

and

$$rate_{Gain1} = \frac{|E'_{Tr}| - \sum_{i=1}^c |E'_{i,Tr}|}{|E'_{Tr}|} = \frac{Gain_1}{|E'_{Tr}|} \quad (33)$$

$Gain_2$ is the normalized gain contained in the interval $[0,1]$:

$$Gain_2 = \frac{Gain_1 - Gain_{min}}{Gain_{max} - Gain_{min}} \quad (34)$$

Notice that **Var** and **Gini** in Table 4 denote the normalized values.

From Table 4, we can extract two graphs:

- In the first graph (see Fig. 4) we can observe that the value of the gain is zero for the first two datasets because they contain only one component and therefore the number of links calculated before and after the decomposition. Thereafter we have a gain progression which attains up to 94.19% of the number of links calculated without decomposition but ignored by using decomposition.
- The second graph (Fig. 5) illustrates the influence of the heterogeneity degree of nodes distribution through connected components on the obtained gain, captured equivalently by the Gini, the Variance or the direct gain normalized measures. We can notice that for the cases where there is only one component, the distribution has no meaning since all the measures (Gain, Gini and Variance) equal zero. When the number of components is greater than 1, The Figure confirms that the more (resp. less) the distribution is uniform the more (resp. less) important is the gain, i.e., the number of ignored links. This graph also confirms, on several real datasets, the relationships between normalized gain, Gini and variance measures already evoked in Sect. 3 on a small example.

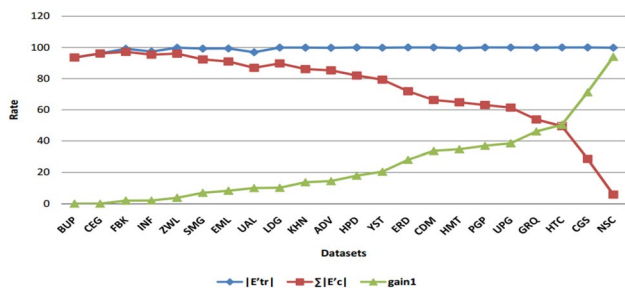


Fig. 4 Comparison between the calculated number of links with and without decomposition

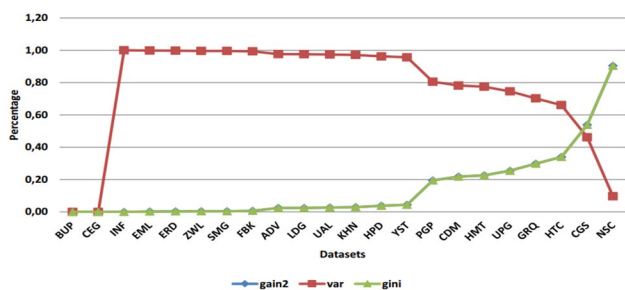


Fig. 5 Comparison between the Gain, Variance and Gini values on several datasets

To conclude this section, we can notice that the decomposition brings us for sure a gain as soon as the social network contains at least two connected components. This obtained gain is more important in the situations where the distribution of nodes through the components is more uniform which allows us to considerably save runtime, especially for large datasets. The following section focuses on the study of the impact of decomposition on runtime.

6.2.2 Impact of decomposition on runtime

The aim of this section is to show how the decomposition of the training graph helps one to substantially shorten runtime. Saving runtime is ensured thanks to decomposition by two means: First as discussed above, the decomposition allows one to reduce the very amount of necessary computations to perform (first experiment). In addition, the decomposition allows one to save additional runtime by paralleling the link prediction process on several processors (second experiment). Moreover, to have a global view, we give an overall comparison of all algorithms (the baseline algorithm and the different variants of our algorithms) by using four similarity measures and four datasets (third experiment).

- Experiment 1. Comparison between our basic algorithm and the baseline algorithm

In this first experiment, we use Jaccard index as a score measure. We execute all algorithms for 100 times on different synthetic datasets and we take the average runtime obtained for each synthetic dataset. We start from the BUP dataset (see Table 3) which contains one component and we generate new synthetic datasets by successively duplicating the initial one to obtain networks with 2 components, 3 components and so on until 10 components. Notice that the obtained datasets are perfectly homogeneous since all the components are identical and accordingly, the obtained gain is optimal (see Table 5 and 6). In Table 5, the number of links treated by Algo2 (after decomposition) as well as its run time are put in bold font to highlight the fact that these results obtained by Algo2 are always better than those obtained by Algo1. In Table 6, the best obtained results in terms of runtime appear in bold font.

We compare our basic proposed algorithm which uses decomposition and performs in sequential way (Algo2) with the baseline algorithm without decomposition which corresponds to the existing approaches (Algo1). The aim of this comparison is to show the impact of exploiting decomposition into connected components on shortening runtime in link prediction.

Table 5 illustrates the results obtained from the first experiment. From Table 5, we can extract the graph depicted in Fig. 6.

From the graph depicted in Fig. 6 we can notice that our algorithm using decomposition (Algo2) clearly outperforms the basic algorithm which does not use decomposition (Algo1) with a very important gap, especially in the case of large graphs with an important number of components, so it is better to decompose the graph before the calculations.

- Experiment 2. Comparison between the variants of our proposed approach

We have demonstrated in Sect. 6.2.1 and Experiment 1 above the necessity of decomposition to minimize the number of links and the runtime. In this second experiment, we limit the comparison to the four variants of our proposed approach (Algo2–Algo5). Recall that the four variants are based on decomposition and they vary according to: (1) the use or not of parallelism and/or (2) the inclusion or not of the decomposition step into the calculation process. Notice that in this experiment we use the same settings and follow the same steps as in Experiment 1.

Table 6 illustrates the results obtained from the second experiment. From Table 6, we can extract the graph depicted in Fig. 7.

On the one hand, we can observe that the necessary time to decompose the graph into connected components is almost negligible. Indeed we can see that the curve for the sequential (resp. parallel) algorithm Algo2 (resp. Algo4)

Table 5 Results obtained by Algo1 (basic algorithm) and our algorithm Algo2 (with decomposition) on different synthetic datasets with Jaccard index

	Datasets train			Without decomposition		With decomposition		Gain			
	C	V _{Tr}	E _{Tr}	E _u '	Algo1 (s)	∑ E _c '	Algo2 (s)	Link		Time	
								Diff	Rate %	Diff	Rate %
1	105	353	5107	0.23	5107	0.23	0	0.00	0.00	0.00	
2	210	706	21,239	0.95	10,214	0.46	11,025	51.91	0.49	51.58	
3	315	1059	48,396	2.16	15,321	0.69	33,075	68.34	1.47	68.06	
4	420	1412	86,578	3.84	20,428	0.91	66,150	76.41	2.93	76.30	
5	525	1765	135,785	6.00	25,535	1.15	110,250	81.19	4.85	80.83	
6	630	2118	196,017	8.68	30,642	1.38	165,375	84.37	7.30	84.10	
7	735	2471	267,274	11.80	35,749	1.60	231,525	86.62	10.20	86.44	
8	840	2824	349,556	15.46	40,856	1.82	308,700	88.31	13.64	88.23	
9	945	3177	442,863	19.59	45,963	2.06	396,900	89.62	17.53	89.48	
10	1050	3530	547,195	24.15	51,070	2.28	496,125	90.67	21.87	90.56	

Table 6 Results obtained by our four algorithms with decomposition on connected components on different synthetic datasets with Jaccard index

	Datasets train			With decomposition				
	C	V _{Tr}	E _{Tr}	∑ E _c '	Sequential execution		Parallel execution	
					Algo2 (s)	Algo3 (s)	Algo4 (s)	Algo5 (s)
1	105	353	5107	0.23	0.23	0.23	0.23	
2	210	706	10,214	0.46	0.45	0.27	0.26	
3	315	1059	15,321	0.69	0.67	0.28	0.27	
4	420	1412	20,428	0.91	0.89	0.31	0.28	
5	525	1765	25,535	1.15	1.12	0.33	0.30	
6	630	2118	30,642	1.38	1.34	0.36	0.30	
7	735	2471	35,749	1.60	1.57	0.38	0.35	
8	840	2824	40,856	1.82	1.79	0.43	0.40	
9	945	3177	45,963	2.06	2.01	0.45	0.40	
10	1050	3530	51,070	2.28	2.23	0.47	0.42	

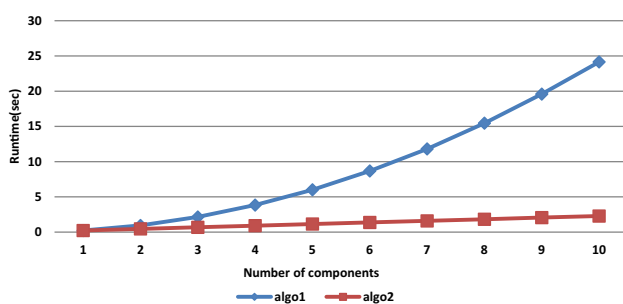


Fig. 6 Comparison between the basic algorithm (Algo2) and our proposed algorithm (Algo1) in terms of runtime on synthetic datasets.

including the decomposition step is very close to that of the sequential (resp. parallel) algorithm Algo3 (resp. Algo5) which directly receives as input the decomposed graph. On the other hand, we can observe that the parallel algorithms

Algo4 and Algo5 are much faster than the sequential algorithms Algo2 and Algo3.

In summary, we can confirm the efficiency of using decomposition prior to link prediction. Indeed, this considerably reduces the amount of required computations and consequently leads to a remarkable decrease in runtime.

Moreover, the parallelization of the link prediction process, made possible thanks to the use of connected components instead of a whole network as input, provides us with an additional gain in terms of runtime. The more the network is large, i.e. has a larger number of components and the distribution of nodes through the components is more homogeneous, the more important is the effect of parallelization in reducing the runtime.

- Experiment 3. Overall Comparison between all the algorithms

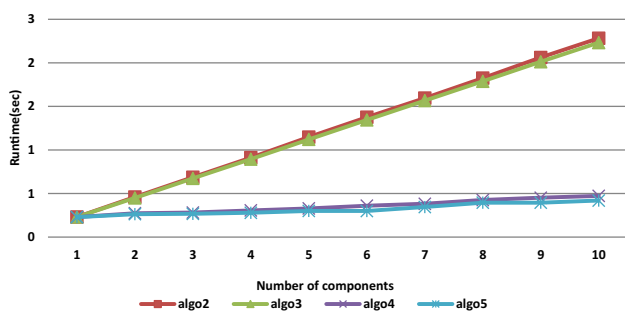


Figure 7 Comparison between the four ours algorithms (Algo2–Algo5) using decomposition in terms of runtime on synthetic datasets

In the third experiment we have applied the five algorithms (Algo1–Algo5) on four real datasets: NSC, CGS, HTC, and GRQ. In addition, we have used for each dataset four score measures: Jaccard index, Common Neighbor (CN), Adamic Adar index (AA) and Resource allocation index (RA). The implementation of these measures can be found in the networkx library.^{1]}

The results are shown in the graphs of Fig. 8. According to these graphs:

- We can notice that the four variants of our proposed approach using decomposition (Algo2–Algo4) clearly outperform the baseline algorithm which does not use decomposition (Algo1) with significant gap, especially in the case of large graphs with an important number of components.
- It is also worth noting that the gain rate in terms of calculation time between the baseline algorithm Algo1 and the other proposed algorithms varies according to the change in the variance (resp. Gini) rate: the lower (resp. higher) is the variance (resp. Gini) rate, the higher is the gain rate and vice versa.

7 Conclusion

In this paper, we presented an efficient approach for link prediction in large social networks which starts from the set of connected components of a network instead of the whole network itself. The key idea of our proposal is that in all algorithms based on local and path-based measures, it suffices to evaluate links relating nodes of the same component. We have shown that this enables us to considerably save runtime by reducing the number of necessary calculations.

¹ https://networkx.org/documentation/stable/reference/algorithms/link_prediction.html

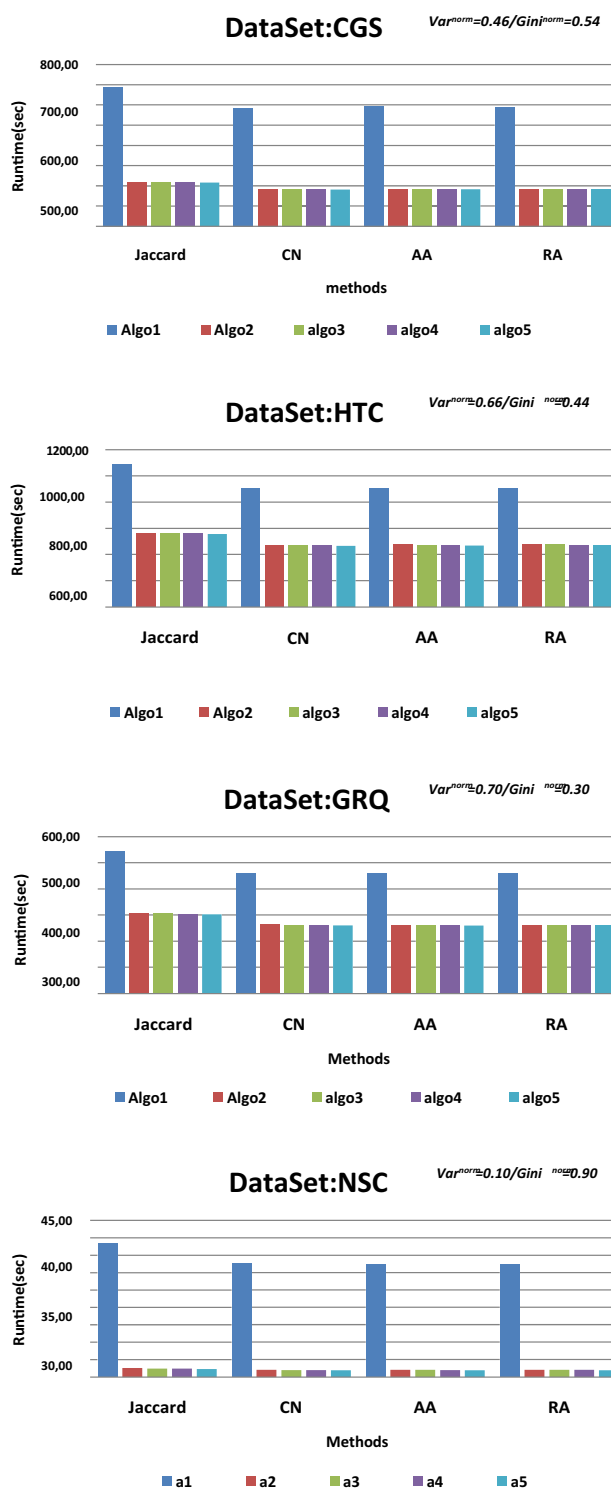


Fig. 8 Comparison between all five algorithms on four datasets and using four similarity measures

We have formally studied the gain obtained thanks to decomposition and shown both theoretically and experimentally that this gain is more important when the distribution

of nodes on the components of the considered network is homogeneous.

Moreover, we have also shown that using decomposition makes possible the parallelization of the link prediction process by distributing the components on several processors. This may provide us with additional gain in terms of runtime, especially for large networks, having a large number of components and where the distribution of nodes on components is homogeneous.

As future work, we plan to develop a fast link prediction library based on our idea of network decomposition and including all local information-based and path-based similarity indices. We plan also to develop a method for combining local and global information for the evaluation of inter-components links that are completely neglected in local and path-based methods, but may make sense in some practical contexts.

Data availability This collection is freely downloadable from the address: <http://noesis.ikor.org/datasets/linkprediction>.

References

- Adamic LA, Adar E (2003) Friends and neighbours on web. *J Soc Netw* 25:211–230
- Ahmad I, Akhtar MU, Noor S, Shahnaz A (2020) Missing link prediction using common neighbor and centrality based parameterized algorithm. *Sci Rep* 10:1–9
- Albert R, Barabasi AL (2002) Statistical mechanics of complex networks. *Rev ModPhys* 74:47
- Ashraf A, Budka M, Musial K (2018) Newton’s gravitational law for link prediction in social networks. Springer, Berlin, pp 93–104. https://doi.org/10.1007/978-3-319-72150-7_8
- Aziz F, Gul H, Uddin I, Gkoutos GV (2020) Path-based extensions of local link prediction methods for complex networks. *Sci Rep* 10:1–11
- Bai S, Li L, Cheng J, Xu S, Chen X (2018) Predicting missing links based on a new triangle structure. *Complexity* 2018:e7312603. <https://doi.org/10.1155/2018/7312603>
- Batagelj V, Mrvar A (2006) Pajek datasets. <https://vlado.fmf.uni-lj.si/pub/networks/data/>
- Boccaletti S, Latora V, Moreno Y, Chavez M, Huang D-U (2006) *Bla bla*. *Phys Rep* 424:175
- Boguña M, Pastor-Satorras R, Diaz-Guilera A, Arenas A (2004) Models of social networks based on social distance attachment. *Phys Rev E* 70:056122
- Breiman L, Friedman J, Olshen RA, Stone CJ (1984) Classification and regression trees (the wadsworth statistics and probability series). CHAPMAN & HALL/CRC. <https://doi.org/10.1201/9781315139470>
- Bu D, L.C., Y. Zhao, et al (2003) Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucleic Acids Res* 31:2443–2450
- Cannistraci C, Alanis-Lobato G, Ravasi T (2013) From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks. *Sci Rep* 1:1. <https://doi.org/10.1038/srep01613>
- Costa LDF, Rodrigues FA, Travieso G, Boas PRU (2007) Characterization of complex networks: a survey of measurements. *Adv Phys* 56:167–242
- Daminelli S, Thomas JM, Durán C, Cannistraci CV (2015) Common neighbours and the local-community-paradigm for topological link prediction in bipartite networks. *New J Phys* 17:113037
- Daniya T, Geetha M, Dr SKK (2020) Classification and regression trees with gini index. *Adv Math Sci J* 2020:1857–8438
- Dorogovtsev SN, Mendes JF (2002) Evolution of networks. *Adv Phys* 51:1079–1187
- Guimera R, Danon L, Diaz-Guilera A, Giralt F, Arenas A (2003) Self-similar community structure in a network of human interactions. *Phys Rev E* 68:065103(R)
- Hasan MAA, Zaki MJ (2011) A survey of link prediction in social networks. *Soc Netw Data Anal* 2011:243–275
- Ibrahim NMA, Chen L (2015) Link prediction in dynamic social networks by integrating different types of information. *Appl Intell* 42:738–750
- Isella L, Stehlé J, Barrat A, Cattuto C, Pinton JF, Broeck WVD (2011) What’s in a crowd? Analysis of face-to-face behavioral networks. *J Theor Biol* 271:166–180
- Kamath PS, Wiesner RH, Malinchoc M, Kremers W, Therneau TM, Kosberg CL, D’Amico G, Dickson ER, Kim WR (2001) A model to predict survival in patients with end-stage liver disease. *Hepatology* 33:464–470
- Kumar A, Singh SS, Singh K, Biswas B (2020) Link prediction techniques, applications, and performance: a survey. *Phys Stat Mech Its Appl* 553:124289
- Kunegis J (2013) KONECT—The Koblenz network collection. In: Proc. Int. Conf. on World Wide Web Companion, pp 1343–1350
- Leskovec J, Kleinberg J, Faloutsos C (2007) Graph evolution: densification and shrinking diameters. *ACM Trans Knowl Discov Data* TKDD 2007:1–2
- Li S, Huang J, Liu J, Huang T, Chen H (2020) Relative-path-based algorithm for link prediction on complex networks using a basic similarity factor. *Chaos Interdiscip J Nonlinear Sci* 30:013104
- Liao H, Zeng A, Zhang YC (2015) Predicting missing links via correlation between nodes. *Phys Stat Mech Its Appl* 436:216–223
- Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. *J Am Soc Inf Sci Technol* 58:1019–1031
- Liu Z, Zhang Q-M, Lü L, Zhou T (2011) Link prediction in complex networks: a local naïve Bayes model. *EPL Europhys Lett* 96:48007. <https://doi.org/10.1209/0295-5075/96/48007>
- Lu L, Jin CH, Zhou T (2009) Similarity index based on local paths for link prediction of complex networks. *Phys Rev E* 80:046122
- Lü L, Zhou T (2011) Link prediction in complex networks: a survey. *Phys Stat Mech Its Appl* 390:1150–1170
- Martínez V, Berzal F, Cubero J-C (2016) Adaptive degree penalization for link prediction. *J Comput Sci* 13:1–9
- Massa P, Salvetti M, Tomasoni D (2009) Bowling alone and trust decline in social network sites. In: Eighth IEEE international conference on dependable, autonomic and secure computing (DASC’09), pp 658–663
- McAuley J, Leskovec J (2012) Learning to discover social circles in ego networks. *Adv Neural Inf Process Syst* 2012:548–556
- Mitzenmacher M (2004) A brief history of generative models for power law and lognormal distributions. *Internet Math* 2004:1. <https://doi.org/10.1080/15427951.2004.10129088>
- Mumin D, Shi L-L, Liu L (2022) An efficient algorithm for link prediction based on local information: considering the effect of node degree. *Concurr Comput Pract Exp* 34:e6289. <https://doi.org/10.1002/cpe.6289>
- Newman ME (2001) The structure of scientific collaboration networks. *Proc Natl Acad Sci* 98:404–409
- Newman ME (2003) The structure and function of complex networks. *SIAM Rev* 45:167–256

- Newman ME (2006) Finding community structure in networks using the eigenvectors of matrices. *Phys Rev E* 74:036104
- Peri S, Amanchy R, Navarro JD et al (2003) Development of human protein reference database as an initial platform for approaching systems biology in humans. *Genome Res* 13:2363–2371
- Reza Z, Huan L (2009) Social computing data repository at ASU, Arizona State University, School of Computing, Informatics and Decision Systems Engineering. <http://socialcomputing.asu.edu>
- Samad A, Qadir M, Nawaz I (2019) SAM: a similarity measure for link prediction in social network. In: 13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS'2019), pp 1–9
- Scrapps J, Tan PN, Esfahanian AH (2009) Measuring the effects of preprocessing decisions and network forces in dynamic network analysis. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 747–756
- Shibata N, Kajikawa Y, Sakata I (2012) Link prediction in citation networks. *J Am Soc Inf Sci Technol* 63:78–85
- Sui X, Lee T, Whang JJ, Savas B, Jain S, Pingali K, Dhillon I (2013) Parallel clustered low-rank approximation of graphs and its application to link prediction. In: international workshop on languages and compilers for parallel computing, pp 76–95
- Tuan TM, Chuan PM, Ali M, Ngan TT, Mittal M, Son LH (2019) Fuzzy and neutrosophic modeling for link prediction in social networks. *Evol Syst* 10:629–634. <https://doi.org/10.1007/s12530-018-9251-y>
- Wang P, Xu B, Wu Y, Zhou X (2015) Link prediction in social networks: the state-of-the-art. *Sci China Inf Sci* 58:1–38
- Wang T, He XS, Zhou MY, Fu ZQ (2017) Link prediction in evolving networks based on popularity of nodes. *Sci Rep* 7:1–10
- Watts DJ, Strogatz SH (1998) Collective dynamics of small-world networks. *Nature* 393:440–442
- Wu J (2018) A generalized tree augmented naive Bayes link prediction model. *J Comput Sci* 27:206–217
- Wu Z, Lin Y, Wan H, Jamil W (2016a) Predicting top-L missing links with node and link clustering information in large-scale networks. *J Stat Mech Theory Exp* 2016:083202
- Wu Z, Lin Y, Wang J, Gregory S (2016b) Link prediction with node clustering coefficient. *Phys Stat Mech Its Appl* 452:1–8
- Xiao X, Zhang Z (2015) Web-age information management. Springer, Berlin
- Yang J, Zhang XD (2016) Predicting missing links in complex networks based on common neighbors and distance. *Sci Rep* 6:1–10
- Yang X-H, Yang X, Ling F, Zhang H-F, Zhang D, Xiao J (2018) Link prediction based on local major path degree. *Mod Phys Lett B* 32:1850348
- Ying D, Ronald R, Dietmar W (2014) Measuring scholarly impact: methods and practice. Springer, Berlin
- Yu B, Chen C, Wang X, Yu Z, Ma A, Liu B (2021) Prediction of protein–protein interactions based on elastic net and deep forest. *Expert Syst Appl* 176:114876. <https://doi.org/10.1016/j.eswa.2021.114876>
- Zeng S (2016) Link prediction based on local information considering preferential attachment. *Phys Stat Mech Its Appl* 443:537–542. <https://doi.org/10.1016/j.physa.2015.10.016>
- Zhou T, Lü L, Zhang YC (2009) Predicting missing links via local information. *Eur Phys J B* 71:623–630

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

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.