


RESEARCH

Open Access



# LPbyCD: a new scalable and interpretable approach for Link Prediction via Community Detection in bipartite networks

Maksim Koptelov<sup>1\*</sup> , Albrecht Zimmermann<sup>1</sup>, Bruno Crémilleux<sup>1</sup> and Lina F. Soualmia<sup>2</sup>

\*Correspondence:

maksim.koptelov@unicaen.fr

<sup>1</sup> UNICAEN, ENSICAEN, CNRS  
- UMR GREYC, Normandie  
Univ, 14000 Caen, France

Full list of author information  
is available at the end of the  
article

## Abstract

Many aspects from real life with bi-relational structure can be modeled as bipartite networks. This modeling allows the use of some standard solutions for prediction and/or recommendation of new relations between objects in such networks. In this work, we combine an existing bipartite local models method with approaches for link prediction from communities to address the link prediction problem in this type of networks. The motivation of this work stems from the importance of an application task, drug–target interaction prediction. Searching valid drug candidates for a given biological target is an essential part of modern drug development. We model the problem as link prediction in a bipartite multi-layer network, which helps to aggregate different sources of information into one single structure and as a result improves the quality of link prediction. We adapt existing community measures for link prediction to the case of bipartite multi-layer networks, propose alternative ways for exploiting communities, and show experimentally that our approach is competitive with the state-of-the-art. We also demonstrate the scalability of our approach and assess interpretability. Additional evaluations on data of a different origin than drug–target interactions demonstrate the genericness of the proposed approach.

**Keywords:** Link prediction, Community detection, Bipartite networks

## Introduction

Many real world applications can be modeled as bipartite graphs, vertices of which are divided into two distinct groups: the tasks of user–product recommendation, member–club recommendation, authors–venues recommendation etc. (Sun et al. 2005). Many recommendation systems fall into this category (Li and Chen 2013). The problem setting that motivates our work also falls in the same group—the prediction of links between drug candidates and biological targets, an essential step of computational drug development.

Design of new drugs is a very important and expensive process in modern pharmacology. A molecule cannot be picked by random from the set of chemical elements and optimized until it can be used as a drug. The probability of successful outcome of this event tends to be zero, because the search space of chemical elements and their

combinations is enormous. To develop a valid drug for a certain biological target, many molecules must be tested first to find a subset of active ones towards a given target. Computational methods are employed more and more frequently to automate this search, while the main approach to reliably identifying such molecules still depends on *in vitro* testing, such as high-throughput screening (HTS) (An and Tolliday 2010). Computational methods, in turn, are able to fine-tune the set of candidates *in silico*, cutting down on time and money invested in real-world testing.

A common challenge which the data on drug–target interaction prediction presents is sparsity. It introduces difficulties or makes it impossible to use traditional recommendation systems techniques for making predictions. As a common solution, different sources of information are combined into a heterogeneous structure, i.e. represented by networks the edges of which have different types. Such a representation significantly improves connectivity of the combined entities, but it introduces another challenge: the straightforward use of most existing link prediction methods becomes impossible in this new setting. The state of the art solutions are limited by the number or type of networks, often referred to as *layers*, e.g. three layers, with two assumed to be similarity networks (Chen et al. 2012; Lim et al. 2016; Buza and Peska 2017).

To address these restrictions, we model the problem as a link prediction problem in a bipartite multi-layer graph and take inspiration from existing methods that use community detection to perform link prediction (Guimerà and Sales-Pardo 2009; Yan and Gregory 2012; Soundarajan and Hopcroft 2012; Valverde-Rebaza and de Andrade Lopes 2014). While this decouples the problem into how to find communities in multi-layer graphs, and how to exploit them for prediction, existing link prediction measures (Canistraci et al. 2013; Xie et al. 2014; Ding et al. 2016) are not directly applicable to the bipartite and/or multi-layer setting. To address this restriction, we extend several of those measures to our problem setting. In addition, we propose two ways for exploiting community information, one based on an existing bipartite local model and an alternative, and adapt existing measures to both of these settings.

Our *main contribution is the resulting link-prediction-by-community-detection approach*. The approach consists of three main steps: community detection, community matching and link prediction. Community matching defines how resulting communities will be exploited, then link prediction is performed using matched communities and one of the proposed measures. We emphasize that our approach is agnostic to any particular community detection algorithm; we demonstrate that several such algorithms allow good performance. We perform evaluation with two most common techniques for community detection, spectral partitioning and the Louvain algorithm, both of which can be easily applied to multi-layer graphs. As we show in the experimental evaluation, our approach is able to effectively perform drug-target prediction on bipartite multi-layer data using one of those techniques. Our contribution includes *the proposition of two community matching techniques and the adaptation of existing link-prediction-by-community-detection measures to the bipartite multi-layer setting*. We evaluate them experimentally and select the best measure and community matching combination. In addition, *we propose to set the parameters of the community detection methods via internal cross-validation* and demonstrate the effectiveness of this method. Finally, using publicly available data, *we construct two bipartite multi-layer networks* the origin

of which are different than drug–target interactions and experimentally demonstrate the genericness of our approach on these networks. Taking into account that data in the bipartite multi-layer setting are rare, *we consider the construction of those networks an additional contribution.*

A final contribution consists of *evaluating a number of recently proposed approaches for bipartite link prediction and comparison of those techniques to each other on data other than drug–target interactions* for the first time, to the best of our knowledge.

The rest of the paper is organized as follows. Section “[Related work](#)” discusses related work on link prediction and community detection in multi-layer networks. Section “[Definitions and problem setting](#)” provides basic notations and definitions. Section “[The LPbyCD approach](#)” presents our link-prediction-by-community detection approach and explains how we adapt existing measures for our setting. Section “[Experimental evaluation](#)” describes the data used for evaluation, the experimental setup and presents the results. Finally, Section “[Conclusion](#)” provides the conclusion and future perspectives. In addition, “[Appendix 1](#)” describes the construction of the two generic data sets used for assessing the genericness of our approach.

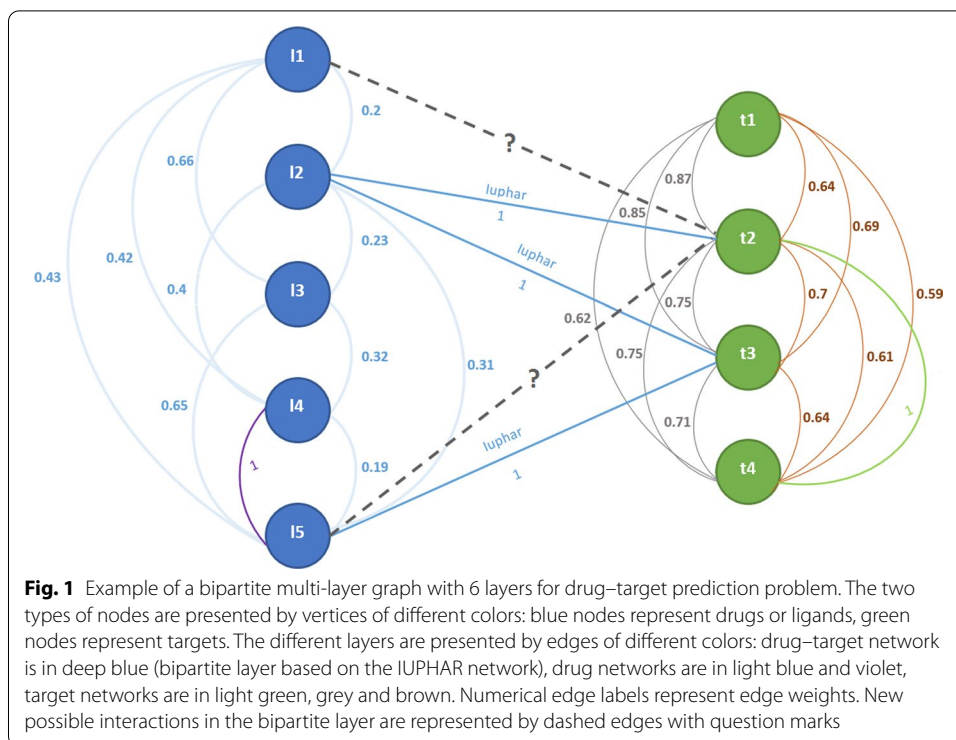
## **Related work**

Existing approaches for link prediction in bipartite multi-layer networks to address the drug–target interaction problem can be grouped into three classes: similarity, random-walker, and latent models based. The first group assumes two out of three possible layers to be similarity networks for drugs and targets respectively, and exploits similarity information to perform link prediction on the third bipartite layer (Ding et al. 2013; Buza and Peska 2017; Buza et al. 2020). When more than two similarity networks are used, an aggregation technique is employed to imitate the standard setting: matrix factorization (Luo et al. 2017), similarity network fusion (Olayan et al. 2018), cross-network embeddings (Chen et al. 2020), or a combination of several functions (Thafar et al. 2020). The use of a such technique increases the overall complexity, and the predictions are performed based on the aggregated data. The second group models the behavior of a random-walker to perform link prediction in multi-layer graph using PageRank adaptations (Chen et al. 2012; Cheng et al. 2012; Wang et al. 2014). Such approaches are dependent on fixing the similarity networks and while the approach was extended to any number of networks in (Koptelov et al. 2018), it pays for this flexibility with high computational cost. The last group of approaches maps drugs, targets and their interactions into a combined feature space, and performs drug–target interaction prediction using regression analysis (Yamanishi et al. 2008), matrix factorization (Zheng et al. 2013; Luo et al. 2017) or denoising models (Tang et al. 2020). The most recent family of approaches in this mold is often referred to as graph embeddings (Goyal and Ferrara 2018; Mohamed et al. 2020; Thafar et al. 2020). The main disadvantage of this group of approaches is a certain lack of interpretability. The common problem of the state of the art on drug–target activity prediction remains the same. Many recent well performing approaches are not suited for de novo drug discovery, when no prior interactions are known for specific drugs and targets at the same time (Olayan et al. 2018; Buza et al. 2020; Mohamed et al. 2020).

The idea to use community information to predict links in graphs is not novel. Clauset et al. (2008) proposed to exploit a learned hierarchical generative community

model to estimate the probabilities of missing links in partially known networks. The authors of Guimerà and Sales-Pardo (2009), Yan and Gregory (2012), Valverde-Rebaza and de Andrade Lopes (2014), Shahriary et al. (2015) combine community detection with existing edge prediction methods to improve prediction accuracy. For instance, in Shahriary et al. (2015) communities are exploited by a PageRank/HITS approach to perform ranking of nodes which are then used for predicting the sign of edges in sign networks. These approaches are based on the hypothesis that vertices in the same community have similar properties, and missing edges are more likely to be found within communities. Missing edges are predicted by node similarity using nearest-neighbor measures (Yan and Gregory 2012), Stochastic Block Models (Guimerà and Sales-Pardo 2009), or in-group/out-group neighbor similarity measures (Valverde-Rebaza and de Andrade Lopes 2014). Edges can be predicted for vertices belonging to the same community even if there is no path between them within the community (Jalili et al. 2017). In Jiang et al. (2020) the authors propose a link prediction strategy which only predicts the edges related to the central nodes of the communities. According to the authors, carrying out link prediction in this way can effectively reduce the probability of predicting edges between communities. More recent studies demonstrate that missing links can be predicted not only for node-pairs within a community, but also for the nodes belonging to different communities (Xu et al. 2020; Ding et al. 2020). The density of links in a particular community or between two communities can be exploited in a naïve Bayes model (Liu et al. 2013). Links can be predicted simultaneously with community detection by decomposing the adjacency matrix to several matrices, one of which is used to derive communities and the others to perform prediction of edges (Shao et al. 2019). The authors of Ahn et al. (2010) reimagine communities as groups of edges rather than vertices, and Soundarajan and Hopcroft (2012) use community detection to modify similarity measures. Finally, there is a set of approaches which extend the concept of shared neighborhoods (Liben-Nowell and Kleinberg 2007) to community neighborhoods (Cannistraci et al. 2013; Xie et al. 2014; Ding et al. 2016). In addition, in Hristova et al. (2016) neighborhood measures have been extended to multiple layers.

Community detection in multi-graphs can also be performed in different ways: directly, by ensemble-based methods, or by graph flattening. The direct methods perform discovery of communities on the multi-layer network directly, e.g. by adapting objective functions for community detection to the multi-layer setting (Kuncheva and Montana 2015; Tagarelli et al. 2017; De Bacco et al. 2017). The main drawback of these methods is the implementation complexity, making them hard to use “out of the box”. Ensemble-based methods perform community detection on each layer separately, and aggregate discovered communities afterwards (Tagarelli et al. 2017). Their main uncertainty is the aggregation mechanism, which requires additional optimization. Flattening methods, finally, summarize multiple edges into single ones and use the resulting single-layer network to discover communities by using one of the common community detection methods such as spectral partitioning (Leskovec et al. 2014) or Louvain algorithm (Blondel et al. 2008). In this work, we test our approach with the last type of method, because it is able to provide the desired result, it is the easiest to use and its computational complexity is potentially the lowest. Please note



that our approach is not fixed to this type of method and can potentially perform prediction with any community detection technique.

### Definitions and problem setting

#### Basic notations

A *graph* is a tuple  $G = (V, E, w)$ , where  $V = \{v_1, v_2, \dots, v_n\}$  denotes a set of vertices or nodes,  $E \subseteq V \times V$  a set of edges defined by distinct vertex pairs  $(u, v) \in V \times V$  with  $u \neq v$  (without self-loops), and  $w$  a labeling function for edges. We also use the notion of a *bipartite graph*, which we define as a graph whose vertices can be divided into two classes  $V_1$  and  $V_2$  such that there is no edge between vertices of the same class:  $G = (V_1 \cup V_2, E, w), E \subset V_1 \times V_2$ .

We address weighted and unweighted graphs in the same manner. We define a *weighted graph* as one with a labeling function for edges  $w : E \mapsto [0, 1]$ , value of which represents a link probability (in case of an interaction network) or vertex similarity (in case of a similarity network). An *unweighted graph* is one where every edge is labeled by 1.

To exploit different sources of information in one single structure, we employ multi-layer networks. We define a *multi-layer network* as a weighted graph where more than one edge  $(u, v)$  can exist for a pair of vertices  $u, v$ . Multi-layer networks can be decomposed into disjunct sets of graphs  $G_i$  that contain at most a single edge for each pair of vertices, called *layers* or just *networks*. As we wrote above, our original setting is a bipartite one. To combine it with the multi-layer framework, we define a *bipartite multi-layer graph* as a multi-layer network whose vertices can be divided into two classes, and where exactly one of the layers is a bipartite graph (see Fig. 1 for an example). Note, following

Kivelä et al. (2014), the multi-layer networks used in this work are **not node-aligned**,<sup>1</sup> **not layer-disjoint**,<sup>2</sup> have *diagonal couplings*<sup>3</sup> which are *categorical*,<sup>4</sup> and the number of non-bipartite layers can be any.

We represent graphs as matrices. The *adjacency matrix*  $A$  has size  $n \times n$ ,  $n = |V|$ , and  $A_{ij}$  represents the weight of the edge  $(v_i, v_j)$  in case of a single layer graph, or the sum of the weights of multiple edges between  $v_i$  and  $v_j$  in case of a multi-layer network. Note that  $A$  has zeros on the main diagonal, because graphs as used in this work have no self-loops. The *degree matrix*  $D$  is the diagonal matrix of same size as  $A$ , where  $deg(v_i)$  represents the degree of vertex  $v_i$ . The *degree* of a vertex is the sum of the weights of the edges adjacent to  $v_i$  (Gallier 2013):

$$D = \begin{bmatrix} deg(v_1) & 0 & \dots \\ 0 & \ddots & 0 \\ 0 & & deg(v_n) \end{bmatrix}, \quad deg(v_i) = \sum_{j=1}^n A_{ij}.$$

The last, and arguably most important, matrix used in this paper is the Laplacian matrix. The *Laplacian matrix*, denoted by  $L$ , is a matrix of the same dimensionality as  $A$  and  $D$ , defined as the difference between the degree matrix and the adjacency matrix:  $L = D - A$ .  $L$  has the same values as  $D$  on the diagonal, and off the diagonal  $L_{ij}$  is equal to  $-A_{ij}$ .

**Problem setting**

We define the problem of link prediction in *bipartite multi-layer graphs* addressed in this paper as follows.

**For** a given bipartite multi-layer graph  $G = G_{V_1V_2} \cup G_{V_1}^{(n)} \cup G_{V_2}^{(m)}$  with  $1 + n + m$  layers, where:

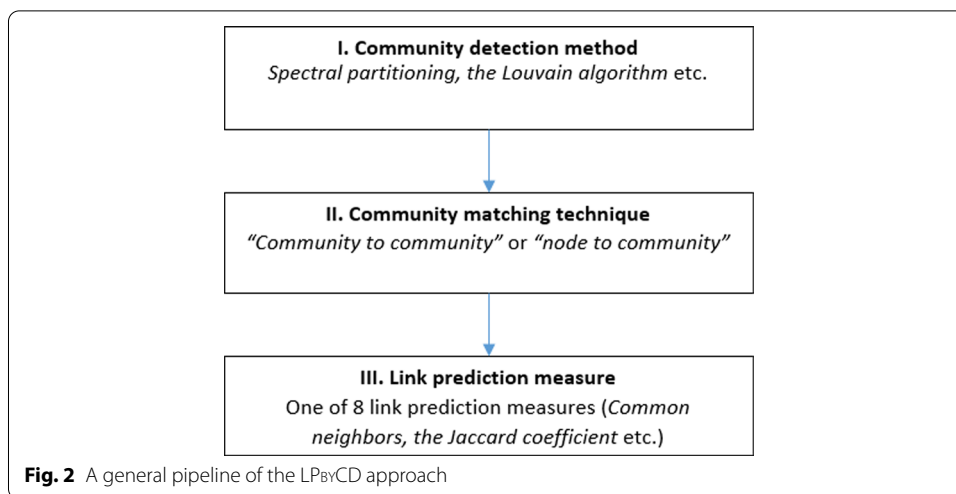
- $G_{V_1V_2} = (V_1 \cup V_2, E_{V_1V_2}, w_{V_1V_2})$  is a *bipartite layer* with  $\forall (u, v) \in E, u \in V_1, v \in V_2$  and labeling function for edges  $w_{V_1V_2} : E_{V_1V_2} \mapsto \{0, 1\}$ ,
- $G_{V_1}^i = (V_1, E_{V_1}^i, w_{V_1}^i)$  with  $w_{V_1}^i : E_{V_1}^i \mapsto [0, 1]$  are layers of type  $V_1$ ,
- $G_{V_2}^j = (V_2, E_{V_2}^j, w_{V_2}^j)$  with  $w_{V_2}^j : E_{V_2}^j \mapsto [0, 1]$  are layers of type  $V_2$ ,

**Predict**, whether for a given  $(u, v) \notin E, u \in V_1, v \in V_2$ , edge labeling function  $w_{V_1V_2}((u, v)) = 1$ .

We limit ourselves to predicting whether there is an activity or not, leaving the prediction of its *strength* as a perspective.

---

<sup>1</sup> In node-aligned networks, all nodes are shared between all layers.  
<sup>2</sup> In layer-disjoint networks, each node is present only in a single layer.  
<sup>3</sup> Inter-layer edges, that cross layers, are only between nodes and their counterparts.  
<sup>4</sup> Diagonal couplings for which all possible inter-layer edges are present.



### The LPByCD approach

In our problem setting, we want to predict links between two distinct types of nodes, e.g. drugs and targets in our experiments (see Fig. 1). To achieve this, we perform community discovery using an existing community detection approach, then exploit the discovered communities to solve the link prediction task. The resulting approach, which we call **Link-Prediction-by-Community-Detection** or LPByCD, can be summarized as a 3 step algorithm (Fig. 2). In the first step of this algorithm, nodes are grouped into communities by a selected community detection method. We perform experiments with spectral partitioning and the Louvain algorithm as such a method, but it can be any community detection algorithm from Section “[Related work](#)”. In the second step, obtained communities are matched using one of the two techniques which we developed. In the last step of the algorithm, the matched communities are exploited with one of the link prediction measures which we adapted to our setting. In this section, we describe how communities are used with our approach (the second step of the algorithm) and explain how we adapt existing measures for link prediction to our setting (the third step).

#### Link prediction by community detection in a bipartite setting

Due to the construction of the networks we use and the community detection methods we evaluate, resulting communities can be *mixed*, i.e. containing both types of nodes, drugs and targets, as well as *pure*, of either type, drugs or targets only. Also, the community detection methods we use produce *non-overlapping* communities only. Mixed communities can be exploited directly with existing measures for link prediction via community detection (see Section “[Related work](#)”), but pure communities cannot, ignoring a large number of drug-target pairs. To treat all communities in a consistent manner, we treat all of them as non-mixed, i.e. each mixed community is treated as two pure ones with links between them. We exploit discovered communities in one of two proposed ways: by matching “community to community” or “node to community”.

---

**Algorithm 1:** The LPBYCD algorithm with common neighbors and the Jaccard coefficient as a link prediction measure

---

```

Input : CD method, parameters, measure, matching, network  $G$ , drug  $d_i$ , target  $t_j$ 
Output:  $\text{score}(d_i, t_j)$ 
 $C_d, C_t \leftarrow \text{CD}(G, \text{method}, \text{parameters})$ 
if  $\text{matching} = \text{'CC'}$  then
   $\text{links\_exist} \leftarrow 0$ 
  for  $d_k$  in  $C_d(d_i)$  do
    for  $t_l$  in  $C_t(t_j)$  do
      if  $(d_k, t_l) \in E$  then
         $\text{links\_exist} \leftarrow \text{links\_exist} + 1$ 
      end
    end
  end
  if  $\text{measure} = \text{'Common neighbors'}$  then
    return  $\text{links\_exist}$ 
  else if  $\text{measure} = \text{'Jaccard coefficient'}$  then
    return  $\text{links\_exist} / |C_d(d_i)| \cdot |C_t(t_j)|$ 
  end
else if  $\text{matching} = \text{'NC'}$  then
   $\text{links\_exist} \leftarrow 0$ 
  for  $t_l$  in  $C_t(t_j)$  do
    if  $(d_i, t_l) \in E$  then
       $\text{links\_exist} \leftarrow \text{links\_exist} + 1$ 
    end
     $\text{links\_possible} \leftarrow \text{links\_possible} + 1$ 
  end
  if  $\text{measure} = \text{'Common neighbors'}$  then
     $\text{score}_1 \leftarrow \text{links\_exist}$ 
  else if  $\text{measure} = \text{'Jaccard coefficient'}$  then
     $\text{score}_1 \leftarrow \text{links\_exist} / |C_t(t_j)|$ 
  end
   $\text{links\_exist} \leftarrow 0$ 
  for  $d_k$  in  $C_d(d_i)$  do
    if  $(d_k, t_j) \in E$  then
       $\text{links\_exist} \leftarrow \text{links\_exist} + 1$ 
    end
  end
  if  $\text{measure} = \text{'Common neighbors'}$  then
     $\text{score}_2 \leftarrow \text{links\_exist}$ 
  else if  $\text{measure} = \text{'Jaccard coefficient'}$  then
     $\text{score}_2 \leftarrow \text{links\_exist} / |C_d(d_i)|$ 
  end
  return  $(\text{score}_1 + \text{score}_2) / 2$ 
end

```

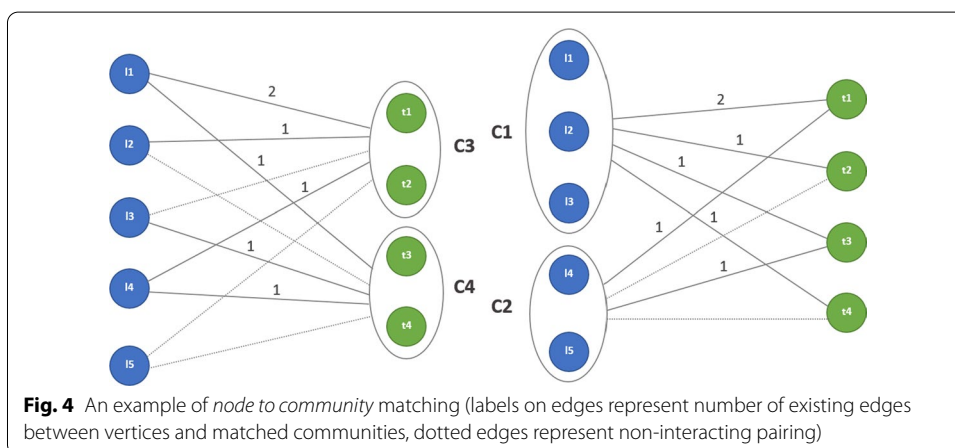
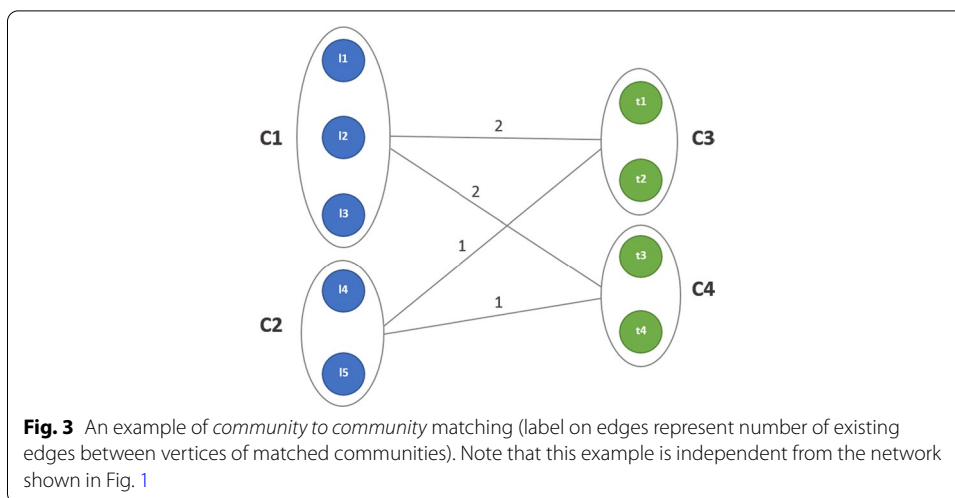
---

### Community to community

In this case, each drug community is paired with each target community, then an adapted measure is used to perform link prediction between paired communities. Each non-interacting drug-target pair between paired communities is assigned the same link probability score. At the end of the matching, each non-interacting drug-target pair from the network will have been assigned a single score, which can be used to rank predictions. We refer to this approach as *community to community* (or *CC*).

In the example shown in Fig. 3, for instance, community C1 is twice connected to community C3 and twice to community C4. C2, on the other hand, is connected once to C3 and once to C4. This matching therefore implicitly assumes that all ligands in C1 have a connection of strength two to all targets in C3 etc. The big advantage of this matching is that even vertices that have no bipartite connection at all can receive a positive score.





An appealing characteristic of our approach is that it can overcome the *cold-start problem*: many recommendation techniques require one item in the predicting pair, e.g. drug (target), to have at least one explicit connection with the other type of entities, e.g. with targets (drugs), that can be exploited. In *CC*, this is not necessary.

**Node to community**

Another way of exploiting communities is to pair each node of one type with communities of the other type. The advantage of that method is that for a selected drug  $d_i$  and target  $t_j$ , the prediction can be made twice: once analyzing connections of a drug with target communities and second analyzing target connections with drug communities, providing a more reliable estimate. This approach is also referred to as Bipartite Local Models (BLM) (Bleakley and Yamanishi 2009).

Figure 4 illustrates the difference of this method with the *CC* matching using the same communities and the same strengths of the connections as in Fig. 3. For example in Fig. 4, there is twice evidence for  $l1$  and  $t1$  to be connected with strength 2, twice evidence for  $l2$  and  $t2$  to be connected with strength 1, and mixed evidence for the connections between pairs  $l1$  and  $t2$ , and  $l2$  and  $t1$  coming from different origin:  $l1$  is strongly connected to the

target community  $C3$  with strength 2 and  $t2$  is weakly connected to the drug community  $C1$  with strength 1, and  $l2$  is weakly connected to the target community  $C3$ , but  $t1$ —strongly to the drug community  $C1$ . In the case of  $CC$  matching, only one evidence is provided with less detailed description: all combinations between  $l1$ ,  $t1$  and  $t1$ ,  $t2$  are strongly connected with strength 2 (Fig. 3).

Vertices belonging to the same community will therefore not necessarily receive the same score but a vertex such as  $l5$  will be strongly punished because it is not connected with a bipartite edge. The link probability score between  $d_i$  and  $t_j$  is computed by aggregating the two results (Buza and Peska 2017). We report results using *mean* as an aggregation function. Our experiments showed that the difference between *max* and *mean* is negligible, and we use *mean* to get a more reliable result. We do not consider *min* as aggregator, because in the case of no evidence for existence of the link in one of the independent predictions the combined probability is also 0. We refer to this approach as *node to community* (or  $NC$ ).

An advantage of our approach with regards to solving the *cold-start problem* while using the  $CC$  matching also holds for  $NC$ . Depending on the link prediction measure that is used,  $NC$  might not be required to have explicit connections with the opposite type of entities (the other type of connections, e.g. similarity networks, might be exploited in such cases).

#### Adapting existing link prediction measures

To be able to use the existing link-prediction-by-community-detection measures, we have to adapt them to the bipartite setting, and to the multi-layer one where it is required. We divide all existing link prediction measures into two broad categories: neighborhood measures and others, which we refer to as community-based. The first group of measures are based on the notion of neighborhood, i.e. the set of vertices directly connected to the examined vertices. The semantic similarity between neighborhood and community, i.e. sets of vertices in both cases, allows us to use neighborhood measures in our setting. The other measures are not based on a notion of neighborhood, but on other metrics, and thus are grouped into a separate group in our work.

#### Adapting neighborhood measures

Many existing link prediction measures exploit the neighborhoods of vertices e.g. in the form of common neighbors (CN), the Jaccard coefficient (JC), preferential attachment (PA), or SimRank (SR). These measures, defined in Liben-Nowell and Kleinberg (2007) as neighborhood measures, take the following form in drug-target formulations:

$$CN(d_i, t_j) = |\{v \mid (d_i, v) \in E\} \cap \{u \mid (t_j, u) \in E\}|, \quad (1)$$

$$JC(d_i, t_j) = \frac{CN(d_i, t_j)}{|\{v \mid (d_i, v) \in E\} \cup \{u \mid (t_j, u) \in E\}|}, \quad (2)$$

$$PA(d_i, t_j) = |\Gamma(d_i)| \cdot |\Gamma(t_j)|, \text{ with } |\Gamma(d_i)| = \sum_{k=1}^{|V|} A_{ik} \text{ and } |\Gamma(t_j)| = \sum_{k=1}^{|V|} A_{jk}, \quad (3)$$

$$SR(d_i, t_j) = \frac{CN(d_i, t_j)}{PA(d_i, t_j)}. \quad (4)$$

$||\Gamma(v)||$  denotes the weight of a neighborhood of a vertex  $v$ , which for individual vertices is equivalent to their degree.

Due to the nature of the communities we obtain, there is little overlap between vertices' neighborhoods, preventing the direct use of neighborhood-based measures. To overcome this, we adapt neighborhood measures for use with our communities, treating them like neighborhoods:

1. The CN measure turns the number of common neighbors of communities  $d_i$  and  $t_j$  into the number of connections. Instead of the measure from Eq. 1, we define the  $CN_{CC}$  version corresponding to  $CC$  matching:

$$CN_{CC}(d_i, t_j) = |\{(d, t) \in E \mid d \in C(d_i), t \in C(t_j)\}|, \quad (5)$$

where  $C(d_i)$ ,  $C(t_j)$  denote sets of nodes that the communities of a drug  $d_i$  and a target  $t_j$  have, respectively. The  $NC$  version of CN is defined as the average of the two independent predictions,  $CN_{NC}(d_i)$  and  $CN_{NC}(t_j)$ , for  $d_i$  and  $t_j$  respectively:

$$CN_{NC}(d_i, t_j) = \frac{1}{2} (CN_{NC}(d_i) + CN_{NC}(t_j)), \quad (6)$$

with  $CN_{NC}(d_i)$  and  $CN_{NC}(t_j)$  defined as:

$$CN_{NC}(d_i) = |\{t \mid (d_i, t) \in E, t \in C(t_j)\}|,$$

$$CN_{NC}(t_j) = |\{d \mid (t_j, d) \in E, d \in C(d_i)\}|.$$

2. JC represents the fraction of all possible connections of  $d_i$  and  $t_j$  that are connected to both. Using the  $CC$  and  $NC$  formulations, our bipartite adaptations of Eq. 2 take the form:

$$JC_{CC}(d_i, t_j) = \frac{CN_{CC}(d_i, t_j)}{|C(d_i)| \cdot |C(t_j)|}, \quad (7)$$

$$JC_{NC}(d_i, t_j) = \frac{1}{2} \left( \frac{CN_{NC}(d_i)}{|C(t_j)|} + \frac{CN_{NC}(t_j)}{|C(d_i)|} \right), \quad (8)$$

where  $|C(d_i)|$ ,  $|C(t_j)|$  represent sizes that the communities of a drug  $d_i$  and a target  $t_j$  have, respectively.

3. PA is defined as the product of degrees of communities of  $d_i$  and  $t_j$ . The  $CC$  formulation of the measure from Eq. 3 takes the form:

$$PA_{CC}(d_i, t_j) = ||\Gamma(C(d_i))|| \cdot ||\Gamma(C(t_j))||, \quad (9)$$

where  $||\Gamma(C(v))||$  denotes the degree of the neighborhood of a community,  $v$  defined as:

$$||\Gamma(C(v))|| = \sum_{v \in C(v), u \notin C(v)} A(v, u), \tag{10}$$

with the neighborhood of a community defined as the set of neighbor vertices the community has:  $\Gamma(C(v)) = \{v \mid v \in C(v), u \notin C(v), (u, v) \in E\}$ . Taking into account that  $PA_{NC}(d_i) = ||\Gamma(d_i)|| \cdot ||\Gamma(C(t_j))||$  and  $PA_{NC}(t_j) = ||\Gamma(t_j)|| \cdot ||\Gamma(C(d_i))||$  we can define the *NC* version for PA (Eq. 3):

$$PA_{NC}(d_i, t_j) = \frac{1}{2}(PA_{NC}(d_i) + PA_{NC}(t_j)). \tag{11}$$

4. Finally, SR is equal to the number of connections of the communities of  $d_i$  and  $t_j$  normalized by the product of their degrees. Instead of the measure from Eq. 4, we define  $SR_{CC}$  and  $SR_{NC}$  versions corresponding to *CC* and *NC* matching respectively:

$$SR_{CC}(d_i, t_j) = \frac{CN_{CC}(d_i, t_j)}{PA_{CC}(d_i, t_j)}, \tag{12}$$

$$SR_{NC}(d_i, t_j) = \frac{1}{2} \left( \frac{CN_{NC}(d_i)}{PA_{NC}(d_i)} + \frac{CN_{NC}(t_j)}{PA_{NC}(t_j)} \right). \tag{13}$$

**Adapting community-based measures**

Other measures proposed in the literature are based on one or several of the following assumptions: all *vertices* have the same semantic, all *edges* have the same semantic, edges are *unweighted*, or vertices whose link is to be predicted find themselves in the *same community*. They exploit community information in some sense, and we thus call them community-based. The different assumptions are violated in our problem setting and we therefore cannot use these measures in a straightforward manner, but we can adapt some to our bipartite setting:

1. Cannistraci et al. (2013) in their *CAR-based* measures propose to exploit the density of communities to reward (or penalize) densely (sparsely) connected neighbors of the vertices whose link is to be predicted. These measures are based on the assumption that common neighbors of the given vertices should belong to the same community, in which case the probability of the connection will be highest. We thus cannot adapt the measures to the *CC* formulation, which assumes that the vertex communities are separated, but can propose the *NC* versions. Our adapted *CAR-based common neighbors* (CCN) will be defined as CN regularized by local community degree, in turn defined as the sum of weights of all edges inside a community. The *NC* formulation of CCN takes the form:

$$CCN_{NC}(d_i, t_j) = \frac{1}{2}(CCN_{NC}(d_i) + CCN_{NC}(t_j)), \tag{14}$$

with  $CCN_{NC}(d_i)$  and  $CCN_{NC}(t_j)$  in turn defined as:

$$CCN_{NC}(d_i) = |\{t \mid (d_i, t) \in E, t \in C(t_j)\}| \cdot \sum_{t_l, t_k \in C(t_j)} A(t_l, t_k) \text{ and}$$

$$CCN_{NC}(t_j) = |\{t \mid (t_j, d) \in E, d \in C(d_i)\}| \cdot \sum_{d_l, d_k \in C(d_i)} A(d_l, d_k).$$

In the same manner, we redefine the *CAR-based Jaccard coefficient* (CJC) as CCN normalized by the size of the community:

$$CJC_{NC}(d_i, t_j) = \frac{1}{2} (CJC_{NC}(d_i) + CJC_{NC}(t_j)), \tag{15}$$

$$\text{with } CJC_{NC}(d_i) = \frac{CCN_{NC}(d_i)}{|C(t_j)|} \text{ and } CJC_{NC}(t_j) = \frac{CCN_{NC}(t_j)}{|C(d_i)|}.$$

- Xie et al. (2014) propose to exploit the connection of vertices to communities, summing over all communities. Their measure, which we refer to as *Neighboring community-based* (NCB), estimates the probability that two given nodes belong to the same community. Our adaptation of the measure is defined as the normalized sum of all CN regularized by the size of the respective community. As in the case of CAR-based measures, we cannot define a CC formulation because of the same reason, and thus propose only an NC formulation:

$$NCB_{NC}(d_i, t_j) = \frac{1}{2} (NCB_{NC}(d_i) + NCB_{NC}(t_j)), \tag{16}$$

with  $NCB_{NC}(d_i)$  and  $NCB_{NC}(t_j)$  in turn:

$$NCB_{NC}(d_i) = \sum_{k=1}^{c_t} \frac{|\{t \mid (d_i, t) \in E, t \in C_k\}|}{|C_k|} \cdot \frac{|\{t \mid t \in C_k\}|}{|T|} \text{ and}$$

$$NCB_{NC}(t_j) = \sum_{k=1}^{c_d} \frac{|\{d \mid (t_j, d) \in E, d \in C_k\}|}{|C_k|} \cdot \frac{|\{d \mid d \in C_k\}|}{|D|},$$

where  $|D|$  and  $|T|$  represent the number of drug and target vertices respectively. Moreover, assuming communities are pure, i.e. consisting only of either drugs or targets, these equations can be simplified to the sum of all CN normalized by the number of vertices of one type:

$$NCB_{NC}(d_i) = \frac{1}{|T|} \sum_{k=1}^{c_t} |\{t \mid (d_i, t) \in E, t \in C_k\}|,$$

$$NCB_{NC}(t_j) = \frac{1}{|D|} \sum_{k=1}^{c_d} |\{d \mid (t_j, d) \in E, d \in C_k\}|.$$

- Ding et al. (2016) in their *Community relevance* measures, finally, propose to extend the notion of node neighborhoods in neighborhood measures to *neighborhoods of communities*. Our CC adaptation of their version of the JC measure is defined as the number of common nodes of examined communities and nodes of the opposite type

connected to those communities normalized by the total number of nodes in this selection. We refer to it as *Community relevance Jaccard coefficient* (CRJC):

$$CRJC_{CC}(d_i, t_j) = \frac{|CRJC_{CC}(d_i) \cap CRJC_{CC}(t_j)|}{|CRJC_{CC}(d_i) \cup CRJC_{CC}(t_j)|}, \quad (17)$$

with  $CRJC_{CC}(d_i)$  and  $CRJC_{CC}(t_j)$  in turn:

$$\begin{aligned} CRJC_{CC}(d_i) &= \{t \mid (d, t) \in E, d \in C(d_i)\} \cup \{d \mid d \in C(d_i)\} \text{ and} \\ CRJC_{CC}(t_j) &= \{d \mid (t, d) \in E, t \in C(t_j)\} \cup \{t \mid t \in C(t_j)\}. \end{aligned}$$

Since the adapted versions of Community relevance are not based on counting specific edges, but on counting nodes instead, the *NC* version of CRJC for a single node would be the same as  $CRJC_{CC}$  for the same node in our interpretation. For the same reason, Community relevance versions of other neighborhood measures in our setting also become meaningless.

The pseudo-code of our approach using two most common link prediction measures is presented in Algorithm 1.

## Experimental evaluation

We test the LPByCD approach with two community detection methods. To evaluate these methods, effects of their parameter settings, and prediction measures defined in the preceding section, we performed experiments on several benchmark data sets for drug-target activity prediction.

### Experimental setup

We begin by evaluating the different measures described in Section “The LPbyCD approach” with two common community detection approaches, keeping most of the parameters fixed, and select the best performing measure. Following this, we show how an internal cross-validation can be used to fix a method’s parameters, and report on their results, which we compare to the state-of-the-art. We also perform experiments on a larger and more challenging data set and give some scalability results.

### Community detection methods

We test our approach with spectral partitioning (Leskovec et al. 2014) and the Louvain algorithm (Blondel et al. 2008) as community detection methods. The first finds the best cut to partition nodes based on eigenvalues of the Laplacian matrix and a threshold method (Spielman and Teng 2007; Fortunato 2010), the second greedily optimizes modularity, a generic measure to determine the quality of any partition produced by a community detection method (Newman and Girvan 2004; Fortunato 2010). We apply spectral partitioning to multi-layer graphs by “flattening” the graph, i.e. summing edge weights to derive the adjacency and degree matrices before performing partitioning. Since Louvain does not employ matrices, we translate the graph into a single-layer graph by summing up the weights of all edges between two vertices. Practically, this is the same operation, but technically it is performed differently.

### Parameters to optimize

Spectral partitioning has two parameters: the value of  $m$  and the thresholding method. The  $m$  parameter represents the number of eigenvectors corresponding to the  $m$  smallest non-zero eigenvalues used to partition the graph into at most  $2^m$  groups. As thresholding several methods can be used: *sign cut* (or *default*), which partitions entries based on whether they are greater or less than zero, *bisection cut* (or *median*), using the median value of entries in an eigenvector as a threshold, producing two components of approximately equal size (Guattery and Miller 1995), and *mean*, which uses the average. In addition to them we propose *sum* that exploits the fact that there are approximately equally as many positive and negative values in eigenvectors of Laplacian matrix, i.e. in practice they sum to a value close to zero, creating groups of vertices approximately equal in size. Moreover, the same thresholding function can be applied to all eigenvectors, or each individual eigenvector can have its own threshold. We call the former approach *global*, and the latter *individualized*. Additionally, the global threshold can be computed by applying the aggregating function (mean, median or sum) to all eigenvectors or only to the  $m$  actually used. We refer to this latter type as *localized*. To sum up, we evaluate 9 different thresholding methods: global, localized, individualized and their combinations with mean, median and sum. The combination *global sum* is a special case since taking the first eigenvector, whose entries all have the same, positive value, into account violates the “close to zero” property sum thresholding exploits. We therefore do not evaluate that thresholding method, but add *default* thresholding to the mix for the experimental evaluation.

The Louvain algorithm has only one parameter—*resolution limit*—which defines a modularity scale. Practically speaking, at different moments of time  $t$ , the difference between optimal partitioning and partitioning produced by the Louvain varies and the resolution parameter represents this change in time (Lambiotte et al. 2008). The resolution limit is not required to be preselected by the user as the  $m$  value is in spectral partitioning. Instead, the default value usually gives satisfying results. However, as in the case of selecting the optimal thresholding method in spectral partitioning, the resolution limit value can be optimized to get a better result.

### Data sets

We perform our experiments on the data sets introduced in Yamanishi et al. (2008): Enzyme, G-protein coupled receptors (GPCR), Ion Channels (IC) and Nuclear Receptors (NR). In addition, we use the Kinase set (Davis et al. 2011). These data sets have been used in prior work on drug–target interaction prediction (Chen et al. 2012; Zheng et al. 2013; Lim et al. 2016; Buza and Peska 2017), and can be considered benchmarks. The data consist of 3 networks: drug similarities, target similarities and drug–target interaction (the bipartite graph). The data sets’ basic properties are presented in Table 1.

In addition, we evaluate our approach on a bigger data set, IUPHAR<sup>5</sup> network, having 6 layers (see Fig. 1 for an example). We have taken that data set from (Koptelov et al.

---

<sup>5</sup> International Union of Basic and Clinical Pharmacology.

**Table 1** Basic properties of benchmark, IUPHAR and generic data sets

Data set	V1	V2	Interactions	Layers	$ V $	$ E $	Sparsity	Connected components
Enzyme	445	664	2926	3	1109	321832	0.524	1
GPCR	223	95	635	3	318	29853	0.592	1
IC	210	204	1476	3	414	44127	0.516	1
NR	54	26	90	3	80	1846	0.584	1
Kinase	68	442	1527	3	510	101266	0.780	1
IUPHAR	8137	2502	12456	6	10639	26706838	0.472	1
MovieLens	943	1682	100000	3	2625	1940394	0.563	1
Unicodelang	254	614	1255	3	868	218996	0.582	1

2018), where it is described in detail. The data set basic properties are also presented in Table 1, which shows that IUPHAR is significantly larger.

Finally, we perform evaluation of our approach on the data sets the origin of which are different than drug–target interactions. To achieve that we construct two new bipartite multi-layer data sets based on publicly available data, which we refer to as generic. The first, the MovieLens network, contains user-movie ratings and the second, the Unicodelang network, concerns countries and languages spoken in them. The construction of the sets is detailed in “Appendix 1”, and the data sets’ basic properties are presented in Table 1.

It is worth to notice that all data sets from Table 1 have one single connected component. The number of connected components of each data set have been reduced to one by aggregating different similarity layers with an original bipartite network, which helps to improve performance of link prediction. This phenomena and the effects of aggregating different layers are discussed in our prior work (Koptelov and Zimmermann 2019).

#### **Evaluation protocol and quality measures**

To perform evaluation of our experiments, we performed a 5×5-fold cross-validation (CV), with each fold containing 20% of all drug–target interactions, acting as test set for link prediction once, while community detection is performed on the other 80%. The process is repeated 5 times, the results are averaged among all runs. We evaluate all the predictions by Area Under ROC Curve (AUC) and Area Under Precision-Recall Curve (AUPR), averaging the results. We also report standard deviation values when it is applicable.

#### **Comparison methods**

*Link prediction measures evaluation* We first test the different link prediction measures and compare the results with the use or spectral partitioning and the Louvain algorithm as a community detection method. To reduce computational complexity, we use default parameters for community detection: *default* as threshold for spectral partitioning and 1.0 as resolution limit for the Louvain algorithm. Note that we optimize  $m$  for spectral partitioning on the test data in this experiment, because there is no a priori number of eigenvalues that will fit all data.



*Parameter selection via internal cross-validation* In link prediction, as in any other predictive task, the main issue is choosing parameter values, in the case of spectral partition  $m$  and the thresholding method, in the case of Louvain the resolution limit. We propose to use *internal cross-validation* as a systematic way to fix community detection approach parameters implemented as follows: splitting off each of 5 *external* test folds (containing 20% of present edges each) to evaluate the model, and using an *internal* cross-validation on the remaining 80% training data as described in Section “[Evaluation protocol and quality measures](#)” to fix the model’s parameters. During the internal cross-validation, we performed grid search over the different parameter settings, varying  $m$  in the interval [1, 25], and testing this value with all nine options for thresholding for spectral partitioning, and varying resolution in the interval [0.1, 1] using steps of 0.1 for Louvain. We report averaged results using mean.

*Comparison with a random walk approach and to the state of the art* Our problem setting was addressed in (Koptelov et al. 2018), using a random walk approach which is basically an extension of PageRank for any number of layers, and thus can be used as a baseline in this work. We perform comparison with the results of the state-of-the-art approaches<sup>6</sup> reported in Buza et al. (2020) for Enzyme, GPCR, IC and NR data sets, and in Buza and Peska (2017) for Kinase data set.

*Performance on a bigger data set* We evaluate our approach on IUPHAR, which is much bigger than any of the benchmark sets, then compare the results with our baseline. As we will show in Section “[Parameter selection via internal cross-validation](#)” that fixing parameter values internally gives effectively the same results as reporting the best result on the testing data, we do the latter for IUPHAR to save time. IUPHAR is too large to search the parameter space for  $m$  starting from  $m = 1$  with step size 1—instead we used step size 10 for the grid search and searched in a more fine-grained manner once we had found a maximum in this way. Note, that negative edges were not removed from IUPHAR as in the structure of the benchmark data sets. Instead, edge labels were ignored for measures computation step, allowing us to compare the results with the baseline taken from Koptelov et al. (2018).

*Running times and scalability* In order to verify scalability of our approach, we compared running times on the benchmark sets to ones with IUPHAR, which is significantly larger than any of the benchmark sets. Note that we show results for complete test folds since optimizing the parameter values via internal cross-validation will require five times this running time before the derived model can be applied to unseen data.

*Performance on generic data sets* In order to evaluate the genericness of our approach, we first repeat parameter selection via internal cross-validation experiment to check if this method also works with generic sets. As in the setting of Section “[Parameter selection via internal cross-validation](#)”, we optimize the thresholding method and the value of  $m$  for spectral partitioning and resolution limit for the Louvain algorithm. The grid search range is same as before:  $m$  is varied in the interval [1, 25] and resolution in [0.1, 1] with steps 0.1. The only exception is that for  $J_{CC}$  on the Unicodelang network, the

---

<sup>6</sup> Using the same evaluation framework with 5×5 cross-validation.

interval was extended by [1, 75], otherwise the local optimum for this setting could not be found.

*State of the art comparison on generic data sets* In addition to evaluating the genericness internally, we optimize our approach on the test data to get an idea of maximum attainable performance and we compare the results with the state of the art approaches: DTIP\_MDHN<sup>7</sup> (Tang et al. 2020), DTiGEMS+<sup>8</sup> (Thafar et al. 2020), ALADIN<sup>9</sup> (Buza and Peska 2017), BLM-NII<sup>10</sup> (Mei et al. 2013), WNN-GIP<sup>11</sup> (Van Laarhoven and Marchiori 2013) and NetLapRLS<sup>12</sup> (Xia et al. 2010). We use these approaches because they can be used directly with our setting and they have source code available. As evaluation protocol we use 5×5 fold cross-validation as described in Section “[Evaluation protocol and quality measures](#)”, which allows us to compare running times with other approaches. We optimize  $m$  in the interval [1,10] for spectral partitioning, other parameters remain unchanged: 9 different thresholds for spectral partitioning and resolution limit from 0.1 to 1 with steps 0.1 for Louvain. In addition to standard quality measures—averaged AUC, AUPR and their standard deviation—we measured both *cpu* time and *exact* time to assess computational complexity of the approaches.<sup>13</sup>

### Implementation

We implemented spectral partitioning with all thresholding methods and all link prediction measures in Python,<sup>14</sup> and we used python-louvain package as implementation of Louvain.

To perform comparison with the state of the art, the source code from Tang et al. (2020) was used as implementation of DTIP\_MDHN, the code from Thafar et al. (2020) as implementation of DTiGEMS+ and the toolbox from Buza and Peska (2017) was used as implementation of ALADIN, BLM-NII, WNN-GIP and NetLapRLS. The evaluation framework was fixed to 5×5 where it was required in the aforementioned implementations, and DTiGEMS+ was used with single similarity network for each type of node.

All experiments in this article were run on servers with 2 processor units of class Intel Xeon E5-2680 with 48 cores in total and 512 Gb of RAM.

## Experimental results

### Link prediction measures evaluation

We first test the different link prediction measures from Section “The LPbyCD approach” on communities produced by either spectral partitioning or the Louvain algorithm. The results are presented in Fig. 5, with *CC* formulations on the left, *NC* ones on the right of each plot. The best-performing measure for each group is indicated by a + sign over

<sup>7</sup> Drug–Target Interaction Prediction method using Marginalized Denoising model on Heterogeneous Network.

<sup>8</sup> A computational method that predicts Drug–Target interactions using Graph Embedding, graph Mining, and Similarity-based techniques.

<sup>9</sup> An Advanced Local Drug–Target Interaction Prediction technique.

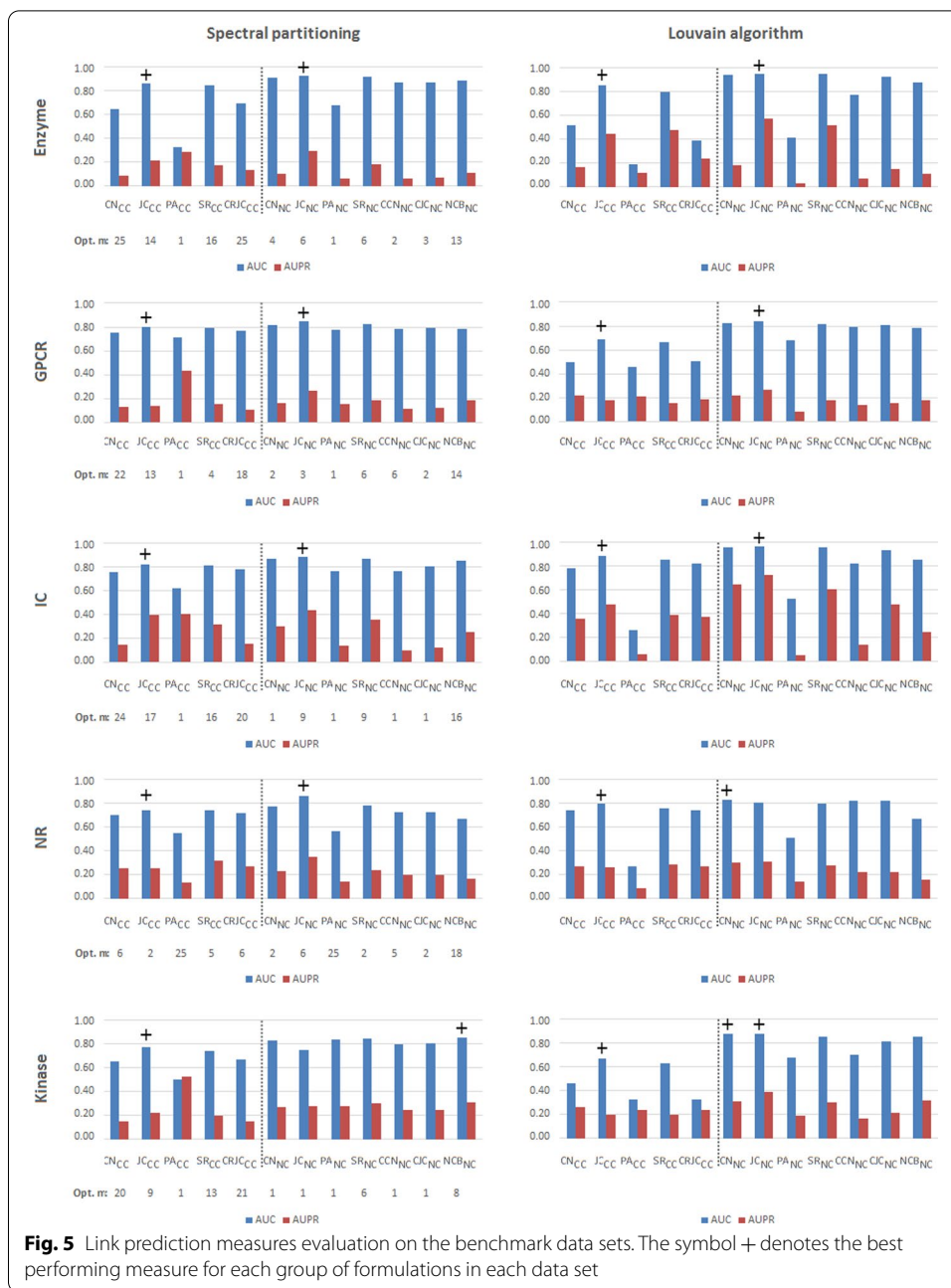
<sup>10</sup> Bipartite local model with neighbor-based interaction-profile inferring.

<sup>11</sup> Predicting Drug–Target Interactions for New Drug Compounds using a Weighted Nearest Neighbor profile.

<sup>12</sup> Laplacian Regularized Least Square method improved for drug–protein interaction Network.

<sup>13</sup> the difference is that *cpu* time gives more accurate estimation when using multi-thread computations.

<sup>14</sup> <https://github.com/koptelovmax/LPbyCD>.



the corresponding bar. We also report the optimal  $m$  value for every measure in spectral partitioning on the figure.

The results show that a number of measures, e.g.  $SR_{CC}$ ,  $CN_{NC}$ ,  $SR_{NC}$ ,  $CCN_{NC}$ ,  $CJC_{NC}$ ,  $NCB_{NC}$ , have acceptable performance in terms of AUC on most of data sets while the Jaccard coefficient performs best for both the  $CC$  and  $NC$  versions in Enzyme, GPCR, IC and NR sets. This might be because the  $JC$  gives less extreme values due to normalization, it also takes community sizes into account, while most of the other measures ( $CN$ ,  $PA$ ,  $SR$ ,  $CCN$  and  $NCB$ ) do not. For Kinase, this is only partially true ( $NCB_{NC}$  is the best in spectral partitioning and  $CN_{NC}$  performs the same as  $JC_{NC}$  for Louvain), which

**Table 2** The LPByCD parameter optimization via internal cross-validation on benchmark data sets and how it compares to the setting when not discovering communities at all

Data set	CD method/measure	Internal validation				External validation			
		AUC	$\sigma$	AUPR	$\sigma$	AUC	$\sigma$	AUPR	$\sigma$
Enzyme	w/o CD <sup>a</sup> /JC <sub>NC</sub>	–	–	–	–	0.88	0.01	0.11	0.01
	Spectral part./JC <sub>CC</sub>	<b>0.85</b>	0.00	0.14	0.04	<b>0.85</b>	0.01	0.19	0.07
	Spectral part./JC <sub>NC</sub>	<b>0.91</b>	0.00	0.19	0.02	<b>0.92</b>	0.01	0.26	0.06
	Louvain/JC <sub>CC</sub>	<b>0.94</b>	0.00	0.52	0.00	<b>0.96</b>	0.00	0.69	0.04
GPCR	Louvain/JC <sub>NC</sub>	<b>0.95</b>	0.00	0.56	0.02	<b>0.96</b>	0.01	0.71	0.02
	w/o CD <sup>a</sup> /JC <sub>NC</sub>	–	–	–	–	0.78	0.02	0.18	0.02
	Spectral part./JC <sub>CC</sub>	<b>0.79</b>	0.01	0.11	0.02	<b>0.80</b>	0.02	0.15	0.02
	Spectral part./JC <sub>NC</sub>	<b>0.84</b>	0.01	0.19	0.01	<b>0.85</b>	0.01	0.25	0.02
IC	Louvain/JC <sub>CC</sub>	<b>0.73</b>	0.01	0.18	0.02	<b>0.80</b>	0.03	0.31	0.05
	Louvain/JC <sub>NC</sub>	<b>0.85</b>	0.01	0.28	0.03	<b>0.89</b>	0.01	0.49	0.04
	w/o CD <sup>a</sup> /JC <sub>NC</sub>	–	–	–	–	0.85	0.01	0.25	0.02
	Spectral part./JC <sub>CC</sub>	<b>0.82</b>	0.01	0.31	0.03	<b>0.83</b>	0.02	0.40	0.04
NR	Spectral part./JC <sub>NC</sub>	<b>0.87</b>	0.00	0.32	0.06	<b>0.88</b>	0.01	0.41	0.05
	Louvain/JC <sub>CC</sub>	<b>0.94</b>	0.00	0.50	0.01	<b>0.95</b>	0.01	0.65	0.04
	Louvain/JC <sub>NC</sub>	<b>0.96</b>	0.00	0.61	0.02	<b>0.97</b>	0.01	0.78	0.03
	w/o CD <sup>a</sup> /JC <sub>NC</sub>	–	–	–	–	0.68	0.08	0.16	0.05
Kinase	Spectral part./JC <sub>CC</sub>	<b>0.73</b>	0.02	0.21	0.03	<b>0.72</b>	0.06	0.24	0.08
	Spectral part./JC <sub>NC</sub>	<b>0.75</b>	0.03	0.19	0.02	<b>0.77</b>	0.06	0.23	0.13
	Louvain/JC <sub>CC</sub>	<b>0.79</b>	0.03	0.27	0.06	<b>0.82</b>	0.06	0.45	0.13
	Louvain/JC <sub>NC</sub>	<b>0.81</b>	0.03	0.30	0.05	<b>0.80</b>	0.09	0.42	0.14
Kinase	w/o CD <sup>a</sup> /JC <sub>NC</sub>	–	–	–	–	0.85	0.01	0.32	0.03
	Spectral part./JC <sub>CC</sub>	<b>0.76</b>	0.01	0.17	0.01	<b>0.77</b>	0.03	0.23	0.02
	Spectral part./JC <sub>NC</sub>	<b>0.85</b>	0.00	0.26	0.02	<b>0.86</b>	0.01	0.35	0.02
	Louvain/JC <sub>CC</sub>	<b>0.72</b>	0.00	0.15	0.01	<b>0.73</b>	0.01	0.19	0.01
Kinase	Louvain/JC <sub>NC</sub>	<b>0.90</b>	0.00	0.38	0.01	<b>0.91</b>	0.01	0.50	0.03

The results for spectral partitioning and the Louvain algorithm used as a community detection (CD) method. The results between internal and external validation align very closely w.r.t. AUC (highlighted in bold)

<sup>a</sup> Assuming  $m = 0$  for spectral partitioning as CD method

can be explained by the fact that Kinase is a quite specific set (has many more targets than drugs, not as sparse as others), and comes from a different source. *We will therefore use JC<sub>CC</sub> and JC<sub>NC</sub> as best performing measures in the following experiments, and test the LPByCD approach further with more diverse data.* Another result is that for these parameter settings spectral partitioning and the Louvain algorithm give approximately the same AUC, but the latter improves on AUPR. *This is not enough to draw a conclusion that Louvain is always better, and we will continue testing both community detection approaches going forward.* Finally, using NC predictions requires a lower  $m$ , i.e. less fine-grained partitions, for spectral partitioning. NC is a more expensive approach than CC in terms of computational time (requires more operations on matching and performs predictions twice), thus this requirement compensates the final cost when searching for the optimal  $m$ .

**Table 3** The LP<sub>BY</sub>CD performance comparison with a baseline approach and the state of the art

Data set	Approach	Performance	
		AUC	AUPR
Enzyme	Baseline	0.84	0.15
	LP <sub>BY</sub> CD(Louvain/JC <sub>NC</sub> )	<b>0.96</b>	0.71
	MOLIERE <sup>a</sup>	<b>0.99</b>	0.90
GPCR	Baseline	0.80	0.22
	LP <sub>BY</sub> CD(Louvain/JC <sub>NC</sub> )	<b>0.89</b>	0.49
	MOLIERE <sup>a</sup>	<b>0.95</b>	0.75
IC	Baseline	0.76	0.27
	LP <sub>BY</sub> CD(Louvain/JC <sub>NC</sub> )	<b>0.97</b>	0.78
	MOLIERE <sup>a</sup>	<b>0.98</b>	0.91
NR	Baseline	0.63	0.26
	LP <sub>BY</sub> CD(Louvain/JC <sub>CC</sub> )	<b>0.82</b>	0.45
	MOLIERE <sup>a</sup>	<b>0.91</b>	0.68
Kinase	Baseline	0.61	0.13
	LP <sub>BY</sub> CD(Louvain/JC <sub>NC</sub> )	<b>0.91</b>	0.50
	ALADIN <sup>b</sup>	<b>0.93</b>	0.60

LP<sub>BY</sub>CD with the Louvain and JC<sub>{NC}</sub> comes close to the performance of the state-of-the-art in terms of AUC (highlighted in bold)

<sup>a</sup> Buza et al. (2020)

<sup>b</sup> Buza and Peska (2017)

**Parameter selection via internal cross-validation**

Table 2 presents the results for spectral partitioning and the Louvain algorithm used as a community detection (CD) method. It shows both the results of internal evaluation, i.e. on the validation set used to fix parameter values, and of the external evaluation, i.e. on the unseen training data. The results between internal and external validation align very closely w.r.t. AUC (highlighted in bold), at the same time, the relatively low standard deviation shows that AUC/AUPR values for different folds do not differ much. Both allow us to draw the conclusion that *the results are rather stable and there is no risk of overfitting when building the model*. Concerning AUPR, interestingly enough, the results on the testing folds are in fact *higher* than for the validation data used in the internal validation. For Enzyme and NR, this might be explainable by slightly higher standard deviation, i.e. for some folds the result remains approximately the same, for others it is an improvement. In addition, *using NC matching always gives better results than CC matching, often by a large margin*.

As in the case of spectral partitioning, the performance of Louvain is rather close in terms of AUC but the differences in AUPR are even more pronounced. GPCR is a bit of an outlier for this experiment in that the differences between internal quality estimation and test fold results are larger than for the other data sets. *This motivates us to also test our approach with more diverse data*. Concerning the superior performance of NC matching, as in the case of spectral partitioning this also holds for Louvain.

**Table 4** Performance ceiling on IUPHAR with optimal parameters

Approach	CD method/measure	Optimal parameters	Performance		Running time, s <sup>a</sup>
			AUC	AUPR	
Baseline	–	eta = 0.2, beta = 0.7	0.57	– <sup>b</sup>	11.9 × 8137 <sup>c</sup>
LPBYCD	Spectral part./J <sub>CC</sub>	m = 400, default <sup>d</sup>	0.74	0.01	3477.8
	Spectral part./J <sub>NC</sub>	m = 0, w/o threshold	0.85	0.01	2484.91
	Louvain/J <sub>CC</sub>	Resolution = 0.1	0.61	0.10	5263.97
	Louvain/J <sub>NC</sub>	Resolution = 0.8	0.82	0.02	2702.62

<sup>a</sup> for 1 fold in average

<sup>b</sup> taken from Koptelov et al. (2018), where AUPR was not reported

<sup>c</sup> must be run for each drug

<sup>d</sup> without thresholding optimization

### Comparison with a random walk approach and to the state of the art

The baseline approach presented in Section “Comparison methods” already had an acceptable performance in terms of AUC (Table 3). However, LPBYCD with J<sub>CC</sub> and J<sub>NC</sub> as a link prediction measure in combination with both spectral partitioning and the Louvain clearly outperforms it (Table 2). Table 3 demonstrates the gap between the results of the baseline and the best performing setting of LPBYCD. Our approach provides a higher margin, because it is based on a more complex heuristic rather than simple random walk. It also demonstrates that the communities it produces are useful.

Concerning comparison with the state-of-the-art, LPBYCD with the Louvain and J<sub>NC</sub> comes close to the performance of the state-of-the-art in terms of AUC (Table 3, highlighted in bold). The reason why our approach does not outperform them could be in the fact that benchmark sets are too small to form good communities, and the regression based approaches, such as ones used in this comparison as the implementations of the state-of-the-art, are better suited for small sets.

### Performance on a bigger data set

The five benchmark data sets on which we have reported to far are relatively small and dense, as shown in Table 1. In this section, we therefore contrast those results with those on the IUPHAR data set.

The results are shown in Table 4. As Table 4 shows, LPBYCD using either spectral partitioning or Louvain clearly improves on the results of the baseline. Contrary to the results on the benchmark data sets, however, the settings with spectral partitioning outperform the ones with Louvain, even though the two are close for node-community matching. The table also shows a very surprising result in that spectral partitioning with node-community matching does best when not creating communities at all! In that case it is the matching technique that does the heavy lifting and effectively treats each entity as a community of size 1 when the time comes to predict new links.

To evaluate whether this is a phenomenon that is specific to IUPHAR or occurs more generally, we compare the results for  $m = 0$  on the benchmark data to the ones achieved by parameter-optimized spectral partitioning in Table 2. As the table shows, optimizing parameters does provide a performance gain, sometimes strongly so, as in the case of NR. Yet at the same time, the results for  $m = 0$  are acceptable. This indicates both that

**Table 5** Relation between network sizes and running times for IUPHAR and benchmark data sets for spectral partitioning and the Louvain algorithm as a community detection method in LP<sub>BYCD</sub>

Data set	Quotient IUPHAR/benchmark set					
	Network size		Running times			
	$ V ^2$	$ E $	Spectral partitioning		Louvain algorithm	
			JC <sub>CC</sub>	JC <sub>NC</sub>	JC <sub>CC</sub>	JC <sub>NC</sub>
Enzyme	92.03	82.98	59.27	34.58	419.11	219.37
GPCR	1119.3	894.61	871.62	528.70	6926.28	3256.17
IC	660.39	605.22	492.60	253.56	5599.97	2525.81
NR	17685.67	14467.41	23185.33	9939.64	131599.25	67565.50
Kinase	435.17	263.73	349.88	204.35	1949.62	1185.36

those *benchmark data sets are not fully representative of the problem setting*, and that for *large data one could do some quick-shot prediction using  $m = 0$  and node-community matching before going to the effort of optimizing parameters.*

#### Running times and scalability

Table 4 shows running times for a single test fold for the different combinations of community detection algorithm and matching method.

According to the results, the settings with spectral partitioning are remarkably faster than ones with Louvain, but for *NC* matching this difference shrinks. Notably, even though *NC* matching is more expensive in itself, it compensates by the fact that this matching technique by its nature requires fewer communities, which are less expensive to discover.

To understand how our approach scales, we compare *relative* network sizes to relative running times. To represent network size we use both the size of the adjacency matrix (i.e. the number of vertices squared) and the number of edges. Dividing these values for IUPHAR by those for the benchmark sets indicate how much bigger, in *relative terms*, the former is than the latter (Table 5, left-hand side).

Relative running times, i.e. running times on IUPHAR divided by running times on the benchmark sets, are shown on the right-hand side of Table 5. With spectral partitioning, we see that size quotients grow faster than running times quotients, with the exception for the NR data set in the *CC* formulation, which is so small that it can be treated in less than a quarter second in this setting. Another exception is the Kinase data set, running times of which grow faster than edge count. This might have to do with the fact that Kinase is rather unbalanced, with far fewer drugs than targets. These are the cases for JC<sub>CC</sub> measure, for JC<sub>NC</sub> quotients of running times are lower than ones of network size for all sets without exceptions. However, JC<sub>NC</sub> setting with  $m = 0$ , i.e. without community discovery at all, might be not representative enough, the JC<sub>CC</sub> does not have this limitation. These results imply that *the approach scales, at least when using spectral partitioning*. Concerning the results with Louvain, the approach does not show similar behavior. Instead, it does not scale at all using neither JC<sub>CC</sub> nor JC<sub>NC</sub>. We believe this is due to the Louvain algorithm implementation that we use, since implementations of

**Table 6** The LP<sub>BY</sub>CD parameter optimization via internal cross-validation on generic data sets

Data set	CD method/measure	Internal validation				External validation			
		AUC	$\sigma$	AUPR	$\sigma$	AUC	$\sigma$	AUPR	$\sigma$
MovieLens	Spectral part./J <sub>C<sub>CC</sub></sub>	0.82	0.00	0.22	0.02	0.82	0.01	0.26	0.02
	Spectral part./J <sub>C<sub>NC</sub></sub>	0.88	0.00	0.30	0.01	<b>0.89</b>	0.00	<b>0.37</b>	0.01
	Louvain/J <sub>C<sub>CC</sub></sub>	0.79	0.01	0.18	0.00	0.77	0.01	0.22	0.01
	Louvain/J <sub>C<sub>NC</sub></sub>	0.88	0.00	0.26	0.00	<b>0.88</b>	0.00	<b>0.34</b>	0.01
Unicodelang	Spectral part./J <sub>C<sub>CC</sub></sub>	0.69	0.01	0.04	0.01	0.68	0.03	0.02	0.01
	Spectral part./J <sub>C<sub>NC</sub></sub>	0.78	0.01	0.14	0.01	<b>0.78</b>	0.01	<b>0.17</b>	0.05
	Louvain/J <sub>C<sub>CC</sub></sub>	0.73	0.00	0.09	0.01	0.72	0.02	0.10	0.03
	Louvain/J <sub>C<sub>NC</sub></sub>	0.80	0.01	0.13	0.01	<b>0.81</b>	0.02	<b>0.16</b>	0.05

The results are summarized in Table 6 with best values for each data set and CD method highlighted in bold

the matching procedures and the evaluation framework remain the same as for spectral partitioning.

#### Performance on generic data sets

So far we performed evaluation of LP<sub>BY</sub>CD approach while solving the drug–target interaction prediction task. This experiment shows that our approach is more general, i.e. it can be used for any kind of bipartite multi-layer data regardless of its origin. The results are summarized in Table 6 with best values for each data set and CD method highlighted in bold.

The results are similar to the ones on benchmark sets (0.89/0.37 as maximum AUC/AUPR among generic sets on MovieLens with spectral partitioning and J<sub>C<sub>NC</sub></sub> measure compared to 0.92/0.26 the best performance on the benchmark data, on IC, for the same measure). *We can therefore conclude that the good results shown above are not due to the specific application domain of biological networks but transfers to other data.* In addition, one can notice that spectral partitioning outperformed Louvain (0.89/0.37 vs. 0.88/0.34 for AUC/AUPR on MovieLens with J<sub>C<sub>NC</sub></sub>), which already happened before when evaluating IUPHAR. This supports the idea that *trying different community detection algorithms can help to improve results.* Finally, we show again that our auto-tuning method works. Our detailed results shown in “Appendix 2” support this observation (and the fact that different folds have different optimal parameters tells us that this optimization is important). Although AUPR values on external validation are slightly better than ones validated internally, the results on external and internal validation still align very closely, thus there is no risk of over-fitting. *We can conclude again that approach parameters can be fixed internally.*

There is another interesting observation, which cannot be seen from the table but appears in our logs. Different thresholding methods behave differently when the number of eigenvalues becomes higher: “local sum” on Unicodelang with CC matching is not working *at all*, “personal sum” starts working *only* at  $m = 46$ , and “default” requires a *higher* number of eigenvalues to reach the optimum compared to others ( $m = 62$  is optimal for “default”, while “personal median” outperforms it on  $m = 49$ ).



**Table 7** The LP<sub>By</sub>CD performance comparison with state of the art approaches on generic data sets

Data set	Approach	Performance				Time, h	
		AUC	$\sigma$	AUPR	$\sigma$	cpu	exact
MovieLens	LP <sub>By</sub> CD(Spectral part. <sup>a</sup> )	0.89	0.00	0.38	0.00	1759	233
	LP <sub>By</sub> CD(Louvain <sup>a</sup> )	0.89	0.00	0.34	0.01	32	32
	DTIP_MDHN	0.96	0.00	0.69	0.00	211	30
	DTiGEMS+	0.94	0.00	0.56	0.00	24	37
	ALADIN <sup>b</sup>	0.93	0.00	0.51	0.00	1076 <sup>b</sup>	4672 <sup>b</sup>
	BLM-NII	0.89	0.01	0.29	0.02	609	20
	WNN-GIP <sup>b</sup>	0.78	0.07	0.21	0.08	3667 <sup>b</sup>	152 <sup>b</sup>
	NetLapRLS <sup>b</sup>	0.92	0.00	0.47	0.00	1267 <sup>b</sup>	17 <sup>b</sup>
Unicodelang	LP <sub>By</sub> CD(Spectral part. <sup>a</sup> )	0.79	0.02	0.17	0.02	91	4
	LP <sub>By</sub> CD(Louvain <sup>a</sup> )	0.80	0.01	0.17	0.03	1	1
	DTIP_MDHN	0.97	0.01	0.71	0.03	5	1
	DTiGEMS+	0.79	0.02	0.23	0.02	2	2
	ALADIN <sup>c</sup>	0.85 <sup>c</sup>	0.01	0.20 <sup>c</sup>	0.03	25	541
	BLM-NII <sup>c</sup>	0.83 <sup>c</sup>	0.01	0.07 <sup>c</sup>	0.00	229	6
	WNN-GIP	0.70	0.01	0.06	0.01	411	12
	NetLapRLS <sup>c</sup>	0.83 <sup>c</sup>	0.01	0.20 <sup>c</sup>	0.02	354	7

<sup>a</sup> with  $J_{C_{NC}}$  measure

<sup>b</sup> quick optimization<sup>16</sup>

<sup>c</sup> slightly modified data

These are all results that we have not observed before. In addition, despite the fact that MovieLens is relatively bigger than Unicodelang in terms of both the number of vertices and edges, it requires a much smaller number of eigenvalues compared to Unicodelang *for both matchings* to reach the optimum (5/11 vs. 10/55 as average  $m$  for MovieLens and Unicodelang with  $NC/CC$  matching respectively). These results let us conclude that *trying different settings have marked impact on results, and our auto-tuning method can help to find an optimal parameter setting by automatically optimizing parameters on the test data.*

#### State of the art comparison on generic data sets

Finally, we measure the performance ceiling on generic data sets by optimizing parameters on test data and compare the results of our approach with state of the art approaches mentioned in Section “[Comparison methods](#)”. We should point out that all these approaches have been proposed for the express purpose of performing drug–target activity prediction, the main bipartite network edge prediction setting in the literature. As a result of this, this is the first time that their performance has been tested on generic data. In addition, this is the first time that very recent approaches for bipartite link prediction such as DTIP\_MDHN, DTiGEMS+ and ALADIN have been compared to each other, to the best of our knowledge.

The results presented in Table 7 demonstrate that DTIP\_MDHN has best AUC and AUPR on both sets, while other approaches behave differently on different sets. On MovieLens, DTiGEMS+, ALADIN and NetLapRLS are the second, the third and the fourth best in terms of both AUC and AUPR respectively. Our approach with Spectral partitioning with  $J_{C_{NC}}$ , also with Louvain with  $J_{C_{NC}}$ , and BLM-NII go next in this

comparison with very similar results: all of the three have same AUC, but first two are better on AUPR. On Unicodelang, ALADIN and NetLapRLS are the second and the third best respectively, and DTiGEMS+ shares approximately similar results with our approach with Spectral partitioning with  $J_{C_{NC}}$ , also with Louvain with  $J_{C_{NC}}$  and BLM-NII: BLM-NII has better AUC than the rest, but the others improve on AUPR. The fact that the last three approaches have approximately similar performance is not surprising. BLM-NII uses a bipartite local model, which is implemented in the  $J_{C_{NC}}$  measure used by our approach to compute the average between predictions, the only difference being that predictions themselves for BLM-NII and our approach are derived in a different manner. The reason why DTiGEMS+ performs differently on different sets could be explained by the fact that the approach is dependent on the presence of multiple similarity measures for each type of nodes but we run our experiments with one type of similarity only. In addition, the Unicodelang network that we constructed shows a certain lack of non-zero similarity information, an issue that we will discuss later in more detail since it also affects other approaches to a certain degree. When it comes to the cpu time comparison, DTiGEMS+, the second best performing approach on MovieLens, has the fastest time on MovieLens, but its exact running time is not the best by far. On Unicodelang, the cpu time of DTiGEMS+ is not the best anymore, but the second best, losing out to LPByCD(Louvain/ $J_{C_{NC}}$ ). The other three best performing approaches, DTIP\_MDHN, ALADIN and NetLapRLS, come in slower than LPByCD(Louvain/ $J_{C_{NC}}$ ), ALADIN and NetLapRLS by an order of magnitude. On the MovieLens data set, NetLapRLS is slower than LPByCD(Spectral part./ $J_{C_{NC}}$ ), but on Unicodelang becomes even slower by an order of magnitude. BLM-NII goes from fourth fastest on MovieLens to sixth fastest on Unicodelang. WNN-GIP has the worst performance and remains the slowest. We use cpu time to illustrate total complexity of each approach implementation when using multi-thread computations. The implementation of LPByCD(Louvain/ $J_{C_{NC}}$ ) does not use this optimization, which is why cpu time and exact time for this approach are equal and its running times are consistently rather low. The only result that is surprising is that the exact time of ALADIN is much higher than its cpu time. We can only conjecture that the cpu time measured by the toolbox with the implementation of this approach is not correct.

One common limitation the ALADIN, NetLapRLS and BLM-NII approaches have, is that their implementation do not support sparse similarity networks, i.e. the non-zero similarity problem we mentioned above. All the state of the art approaches that we tested require to input data in the form of adjacency matrices. Our data often do not have similarity information for entries. When an entry in the data does not have a value for any of the attributes, we cannot compute the similarity for this entry, and the adjacency of such entry to all others is set to zero in the adjacency matrix corresponding to the given similarity layer. This results in many rows in the adjacency matrices representing similarity networks been filled by zeros. As a consequence, ALADIN, NetLapRLS and BLM-NII fail on Unicodelang, similarity networks of which are more sparse than in other sets. To overcome this, we have to perform experiments on slightly modified

**Table 8** Average ranking position for AUC, AUPR and running times for all comparison methods across benchmark and generic data sets

	LPByCD		DTIP_MDHN	DTiGEMS+	ALADIN	BLM-NII	WNN-GIP	NetLapRLS
	1 <sup>a</sup>	2 <sup>b</sup>						
AUC	7.0	5.0	1.0	2.9	2.7	4.3	6.7	4.6
AUPR	6.9	5.6	1.0	2.0	3.0	6.6	6.4	4.1
t <sup>c</sup> , cpu	4.7	1.1	3.4	2.1	4.0	5.9	7.3	7.4
t <sup>c</sup> , exact	3.4	2.3	1.7	3.9	7.3	4.7	6.3	6.4

<sup>a</sup> Spectral partitioning with  $J_{C_{NC}}$  measure

<sup>b</sup> The Louvain algorithm with  $J_{C_{NC}}$  measure

<sup>c</sup> Running time

data.<sup>15</sup> This could also explain the fact that DTiGEMS+, which is very dependent on similarity information, performs worse on Unicodelang. Another issue we faced with ALADIN, NetLapRLS and WNN-GIP on the MovieLens data set, is that we could only run these approaches in quick optimization mode<sup>16</sup> due to incredibly high exact running time in the “full” mode. Taking into account that MovieLens is a bigger set than Unicode by the number of both vertices and edges (Table 1), we suspect that means that those approaches (or at least their implementations) are not scalable enough.

To sum up, DTIP\_MDHN is the clear winner in this comparison. Its complexity and running time are not the best, but can be compensated by the performance it provides. However, DTIP\_MDHN is a latent model based approach, which means that it lacks possibility of interpretability. The latter might not be essential in the user-movie and other recommendation tasks in general domain, but it is very desired functionality in life science applications such as drug–target interaction prediction (the impact from watching a not properly recommended movie is different from one caused by a wrong medication treatment). The other best performing approaches that we tested are data quality dependent (DTiGEMS+, ALADIN, BLM-NII, NetLapRLS) or possess extreme time complexity on a bigger set (ALADIN, WNN-GIP, NetLapRLS). We can conclude that *the LPByCD approach based on the Louvain algorithm with  $J_{C_{NC}}$  measure provides the best trade-off in this experiment*. Being faster than most of the comparison approaches, it has no limitations regarding the input data and its predictive results are not far from the maximum.

#### **State of the art comparison across multiple data sets**

To verify whether the results on generic data sets correspond to the results obtained in previous experiments or it is specific for given data, we computed average ranking position for AUC, AUPR and running times for all comparison methods across benchmark and generic data sets (Table 8). We do not compare the state of the art on

<sup>15</sup> we replace the default similarity value 0.0 for such items without properties by a default similarity value 0.0001.

<sup>16</sup> an undocumented feature available in the toolbox of (Buza and Peska 2017).

IUPHAR since the majority of the comparison methods do not support input with more than 2 similarity layers.

The results are similar to ones from Section “[State of the art comparison on generic data sets](#)”: DTIP\_MDHN performs best and DTiGEMS+ is the second best. On the other hand, LPBYCD with Louvain provides best cpu time and the second best exact time if averaged across all sets. We also performed the significance assessments using the Nemenyi Post-Hoc test (Nemenyi 1963). According to these assessments the difference in ranking between multiple methods in our comparison is not significant since the critical distances which we obtained are 3.97/3.64 at the 0.05/0.1 level respectively. For instance, the difference in performance between LPBYCD with Louvain and DTIP\_MDHN is significant, because the difference in their ranks is higher than any of the two critical distances, but between LPBYCD with Louvain and the two other best performing approaches DTiGEMS+ and ALADIN is not, because their differences are lower than any of the critical distance values.

We can draw a conclusion that the result in its current form just supports the results obtained in the previous sections. To get more reliable ranking the critical distance needs to be reduced. To achieve that one needs to evaluate all comparison methods on a higher number of sets, which is hardly possible taking into account how bipartite multi-layer data are rare in open access. On the other hand, we believe that *the performance of our approach can be improved further if one could find a technique which produces communities of better quality* for our setting.

### **Interpretability**

Our approach offers a *straightforward option for interpretation of a link prediction*. In addition to basic information such as ranking or optimal parameter settings, we can show for each of the two vertices in the network the following:

1. the communities they belong to (with possibility to explore all necessary information about communities: their sizes, nature (were they mixed or pure), identifiers of other drugs and targets in the communities),
2. the weights of intra-community edges,
3. the number and layout of inter-community edges,
4. their numerical translation by the measure and matching technique (which equates to a link probability in the case of a normalized measure, with the JC for instance).

To illustrate how it works, consider the same example network as in Section “[Link prediction by community detection in a bipartite setting](#)”. Suppose that the pair of drug  $l_2$  and target  $t_2$  from that network is predicted to be interacted. The following information can be presented in addition to this result:

1. drug  $l_2$  belongs to community  $C_1$  (size 3, other drugs:  $l_1, l_3$ ) and target  $t_2$  to community  $C_3$  (size 2, other targets:  $t_1$ );
2. in this simplified example, there is no similarity networks, thus the vertices inside communities are not directly connected;
3. there are 2 links between  $C_1$  and  $C_3$  shown as in Fig. 3;

4. in case of  $J_{CC}$  measure the probability of interaction for the predicted pair is  $J_{CC}(l_2, t_2) = 2/6 = 0.33$  (Fig. 3), and in case of  $J_{NC}$  is  $J_{NC}(l_2, t_2) = (1/2 + 1/3)/2 = 0.42$  (Fig. 4).

*This is a possibility that is not available for recent, well-performing techniques based on latent models, such as graph embedding and neural networks.*

This kind of information might be useful for a domain expert (drug researcher in case of drug–target prediction), who will be the end-user of the approach. It should help to get a better understanding and/or more confidence in the prediction results.

## Conclusion

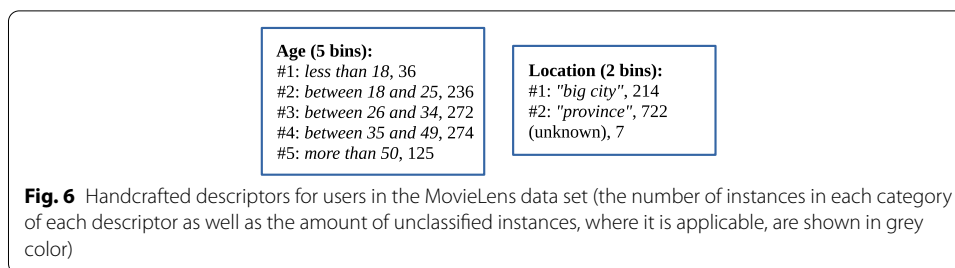
We have presented an approach for link prediction in bipartite multi-layer graphs using graph community structure and link prediction measures adapted from those proposed in the literature. We have found empirically that combining the well-known and relatively straightforward Jaccard coefficient, particularly in a BLM formulation, with the Louvain algorithm for community detection allows us to achieve results that are competitive with the state-of-the-art. We have also demonstrated that it is possible to set the parameter values of the community detection techniques via internal cross-validation and that they transfer well to unseen data.

In addition, we demonstrated scalability of our approach by performing evaluation on the much larger and sparser IUPHAR data set and comparing the time complexity with benchmark data sets, and assessed interpretability.

Finally, we tested our approach on data sets of which the origin is different from drug–target interaction. We demonstrated experimentally that our approach performs well regardless of the domain of the task. Comparison with the state of the art (most of which had not been evaluated on such data before) showed that our approach is more general, does not depend on the structure of data, and can be one of the fastest compared to other approaches, depending on the employed community detection method. The sets which we constructed, in their turn, demonstrated to be challenging enough to be new benchmarks in the field.

Taking into account that the predictive performance of our approach is not the best compared to the state of the art, as well as the advantages of our approach, it can be recommended to be used as a quick-shot prediction tool regardless of the quality of data, particularly with the regards to solving the cold-start problem that many recent approaches have (Olayan et al. 2018; Buza et al. 2020; Mohamed et al. 2020). At the same time, our approach offers the opportunity to interpret its results, which has its own importance in the research on computational drug discovery.

We have limited ourselves to two easy-to-use community detection methods in this work, and propose to evaluate the use of other methods in the future. As we mentioned in Section “Definitions and problem setting,” so far we have only looked at predicting interaction as a binary setting but the more challenging setting would be to predict the *strength* of the interaction. Finally, it would be interesting to add layers derived from other information sources to the networks and use our approach to identify possible redundancies among them.



## Appendix 1

This appendix describes construction of the generic data sets used in section “[Experimental evaluation](#)”.

### MovieLens

MovieLens data sets<sup>17</sup> are well-known benchmarks for recommendation systems (Harper and Konstan 2015). Based on real-life surveys of individuals on existing movies, they are made for solving user-movie recommendation task (Herlocker et al. 1999).

#### *The original set description*

For minimizing evaluation complexity, we base construction of our multi-layer network on the oldest, and thus the smallest, set of the MovieLens group.<sup>18</sup> The version we use includes information about 943 users, 1682 movies and 100,000 movie ratings made by users. In the data, users are presented by age (integer value), gender (binary value), occupation (one of 21 categories) and a post code (in the US or Canadian format for citizens from US and Canada respectively). Movies are described by name, release date, genre (19 multiple categories) and include links to Internet Movie Database (IMDB).<sup>19</sup>

#### *Feature construction*

To build similarity networks, we first create features to describe users and movies. To achieve that, we exploit available properties of both types of entities.

*User descriptors* To describe users by age, we introduced five age groups for users: less than 18, between 18 and 25, between 26 and 34, between 35 and 49, and 50 and more. The first and the last usually have opposing preferences in movies, the middle groups were selected based on principle of balance (Fig. 6). Gender category was used as it is, and occupation as well. Post codes were converted into the exact location names they belong to using publicly available databases.<sup>2021</sup> Then we binarised these locations into two groups: “big city” and “province” based on the top 100 US cities<sup>22</sup> and the top 10

<sup>17</sup> <http://grouplens.org/datasets/movielens>.

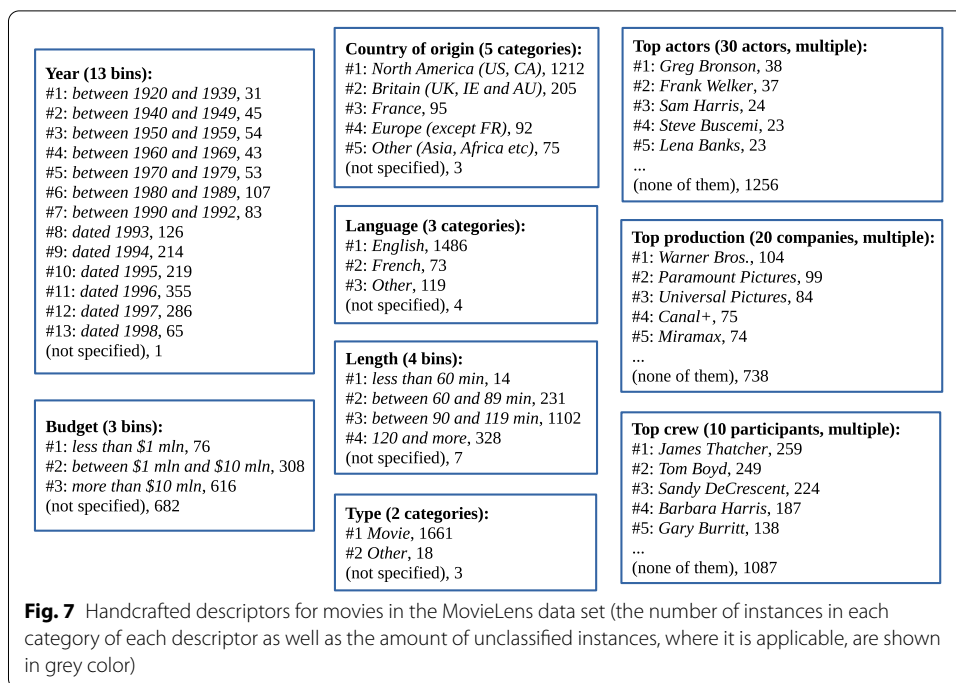
<sup>18</sup> 100K Dataset (containing the user ratings from 1998).

<sup>19</sup> A popular internet database of movies: <http://www.imdb.com>.

<sup>20</sup> Database of ZIP codes of the USA states: <http://www.pier2pier.com/links/files/Countrystate/USA-Zip.xls>

<sup>21</sup> Database of post codes of Canada: <http://www.postalcodesincanada.com>.

<sup>22</sup> According to census of 1990 (published in 1998): <http://www.census.gov/population/www/documentation/twps027/tab22.txt>.



Canadian cities.<sup>23</sup> The resulting partitioning produced groups with roughly 20% of individuals from “big cities” and 80% from “province” (Fig. 6).

To sum up, we encode each user by the vector with 30 binary values:

- 5 bits for age (5 bins),
- 2 bits for gender (2 categories),
- 21 bits for occupation (21 categories),
- 2 bits for location (2 groups).

We use 1 bit per bin because some of the properties can be unknown (they are encoded by 0s in such cases).

*Movie descriptors* The movie descriptors are more complex to build. They require more information to be merged together, however, the vectors constructed with these descriptors are more informative. The basic properties, including release date and genre, do not require much preprocessing. We only binarised the dates into 13 categories based on the principle of balance. For that we kept approximately similar number of items for each group of old movies, and for individual groups corresponding to each year of recent movies, we also used the same principle (Fig. 7).

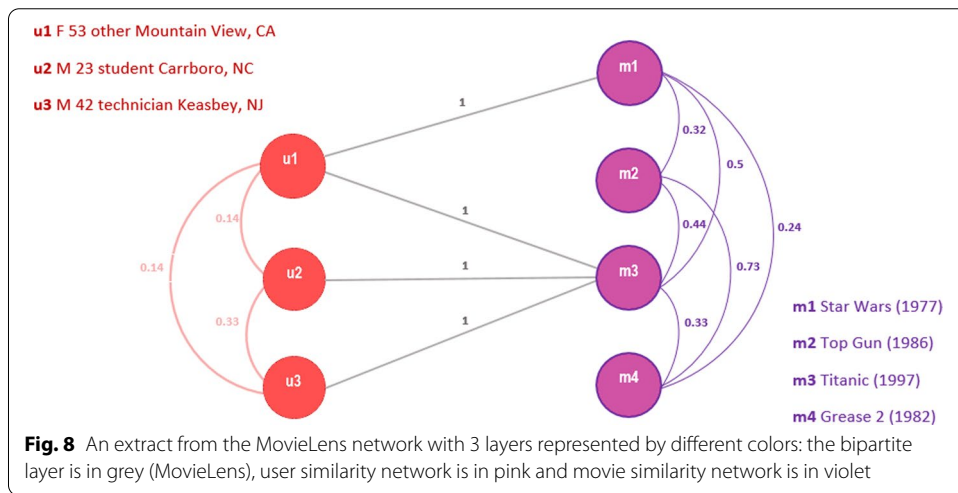
IMDB identifiers provided with the data are not active anymore, and therefore we did not find a better way than to parse new IMDB identifiers from the website. We matched candidate movies by title, year and release date<sup>24</sup> and used edit distance algorithm

<sup>23</sup> According to census of 1996: [http://en.wikipedia.org/wiki/List\\_of\\_largest\\_Canadian\\_cities\\_by\\_census](http://en.wikipedia.org/wiki/List_of_largest_Canadian_cities_by_census).

<sup>24</sup> The imdb.com was accessed on 19/10/2019.

**Table 9** Ranges of Generic data sets' similarity networks

Data set	Network	Similarity values			
		Min	Mean	Median	Max
MovieLens	User similarity	0	0.27	0.33	1
	Movie similarity	0	0.25	0.24	1
Unicodelang	Country similarity	0.03	0.31	0.29	1
	Language similarity	0	0.31	0.27	1



(Damerau 1964) to filter candidates. We then processed the lists of candidates and filled in blanks, manually double checking movie genres. Once IMDB identifiers were collected, the IMDB database was parsed one more time to retrieve additional information about the movies:<sup>25</sup> budget, country of origin, main language, length, type (movie or other), actors, production company and crew members (directors, musicians, operators etc).

We initialized construction of additional descriptors from the budget property, which was converted from multiple currencies into US dollars using historical exchange rates<sup>26</sup> on 1998, 22 of April.<sup>27</sup> Then these values were binarised into 3 categories with meaningful ranges: less than \$1 mln, between \$1 mln and \$10 mln, and more than \$10 mln. The values of the next property, countries of origin, were placed into 5 groups using expert knowledge: USA, Britain and France are major movie production countries of all time,<sup>28</sup> while Europe (except France) has own cinematic trends (Fig. 7). We split languages into three categories: English, French, and other, emphasizing French movies as a separate group since France has own influence on movie production and thus can be placed separately. The lengths of the movies were binarised into four categories with meaningful

<sup>25</sup> The imdb.com was accessed again on 25/10/2019.

<sup>26</sup> <http://www.poundsterlinglive.com/bank-of-england-spot/historical-spot-exchange-rates/usd>.

<sup>27</sup> The last day of the MoveLens survey for 100K Dataset.

<sup>28</sup> <http://www.the-numbers.com/movies/production-countries>.



ranges as shown in Fig. 7. Concerning the last three properties, we have chosen the top  $n$  values for actors, companies and crew members, allowing multiple categories. Value of  $n$  for each property was selected individually using the principle of minimizing redundancy of vector representation.

To sum up, we encode each movie by a vector with 108 binary values: 18 bits for genre that goes with the data, plus 90 bits for the manually created features summarized in Fig. 7.

### **Network construction**

As in the case with IUPHAR, the construction of similarity networks is performed with the use of the Tanimoto coefficient, which is well suited for binary vector comparison. The resulting distribution of values offer us intuition about the quality of constructed networks. As can be seen from Table 9 (first two rows), these values are diverse enough, covering the full range from 0 to 1 with mean and median far from 0 or 1. This allows the conclusion that the resulting similarity networks are informative enough. In other words, it would be possible to find a similar movie for a randomly selected one by maximizing their similarity value (see Fig. 8 for an example: *Grease 2* (1982, comedy, musical, romance) is more similar to *Top Gun* (1986, action, romance), than to *Star Wars* (1977, action, sci-fi)).

As defined in Section “Definitions and problem setting”, we do not predict the strength of the interaction, and therefore user-movie ratings in the bipartite network need to be binarised. We use ranks from 3 to 5 to represent positive interaction and from 1 to 2 as negative. That produced 82,520 positive and 17480 negative edges.

In total we have built three networks:

- the user-movie binary interaction network,
- a user similarity network calculated using the Tanimoto coefficient, and
- a movie similarity network also calculated using the Tanimoto coefficient.

The resulting MovieLens network can be downloaded from the repository.<sup>29</sup> Fig. 8 illustrates an extract from the network.

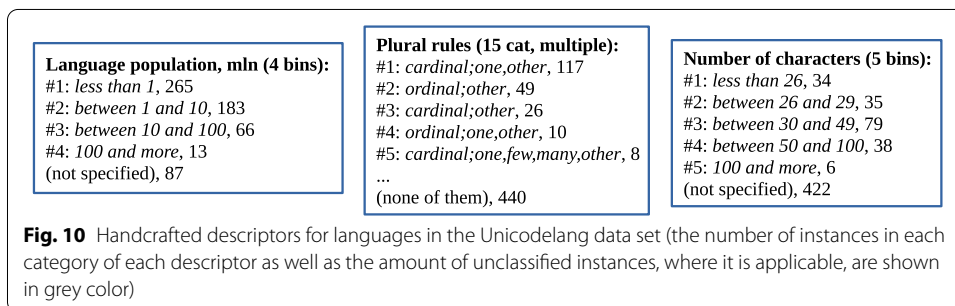
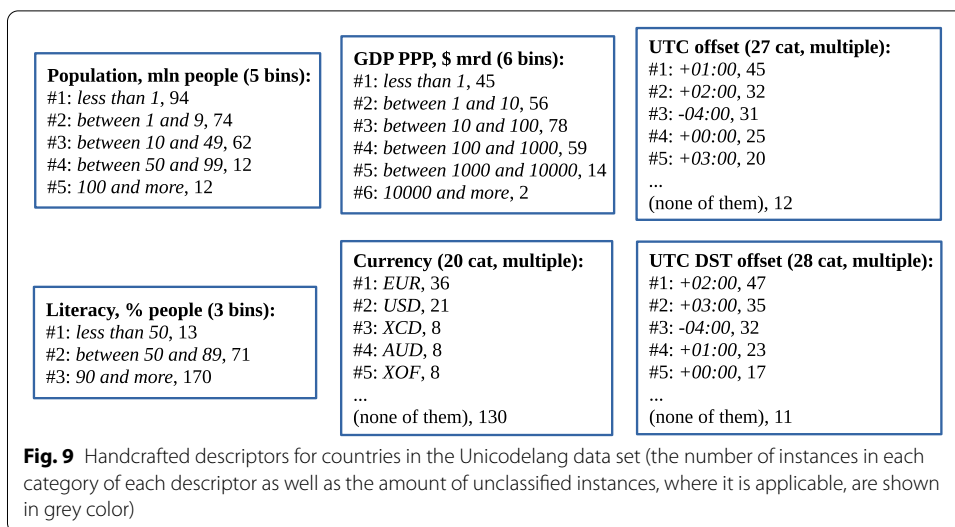
Basic properties of the resulting network are presented in Table 1. As can be seen from this table, the MovieLens network has acceptable sparsity and consists of a single connected component.

### **Unicode languages**

There are not many data sets with bipartite structure which are both open source and provide a rich property set for both types. For instance in KONECT,<sup>30</sup> a popular open-access collection of network data sets, there is only one network satisfying these requirements (Kunegis 2013). Unicode languages or “Unicodelang” is that set describing languages, countries, and their relations.

<sup>29</sup> <https://github.com/koptelovmax/datasets>.

<sup>30</sup> Koblenz Network Collection.



**The original set description**

The Unicodelang network represents 254 countries (territories), 614 languages and 1255 of their “interactions”, denoting the proportion of the population of a given country speaking a given language. As in the case with the previous data set, we perform an extension of the basic network by constructing additional layers.

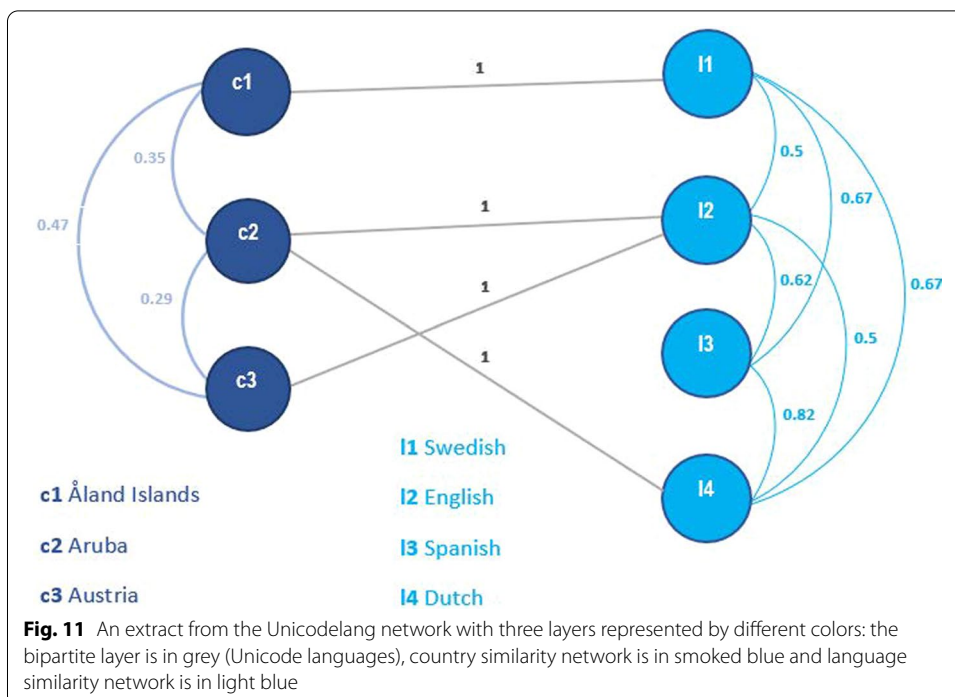
**Feature construction**

As before, to build similarity networks, we first construct features to describe both type of entities. For doing that, we parse country and language properties from the original source of the set used by KONECT. We then exploit collected properties to build descriptors for both of types.

*Country descriptors* The countries are presented by 10 basic properties.<sup>31</sup> We binarise the first three: population size, percentage of literacy and GDP PPP<sup>32</sup>, into 5, 3 and 6 categories respectively using meaningful ranges (Fig. 9).

<sup>31</sup> [http://www.unicode.org/cldr/charts/25/supplemental/territory\\_language\\_information.html](http://www.unicode.org/cldr/charts/25/supplemental/territory_language_information.html), accessed on 7/11/2019.

<sup>32</sup> Gross domestic product based on purchasing power parity.



Another property, currency, was placed into 20 multiple categories, ignoring distinct values and taking into account that some countries can have more than one official currency (Fig. 9). Next six properties we used as they are given: number of days in a week, first day of a weekend, last day of a weekend, measurement system and paper size. Next, we parse supplementary information<sup>33</sup> to get additional properties. That includes continent, subcontinent and two time zone offsets. The first two were used in a straightforward manner, while the time zone names of the latter were converted to the exact values using publicly available chart.<sup>34</sup> Then collected offset values were fitted into 27 and 28 categories for UTC<sup>35</sup> and DST<sup>36</sup> respectively, ignoring distinct values (Fig. 9).

*Language descriptors* The languages are presented by boolean properties whether a language is modern, primary or one/none of them, and a language script type, both of which we use unchanged.<sup>37</sup> Another property, language population, was adopted from the countries' supplementary information<sup>[33]</sup> summing up numbers for several instances (Fig. 10).

The next two, plural rules' types (cardinal, ordinal *etc.*) and plural rules' categories (one, many, other *etc.*), were extracted from the languages' supplementary information.<sup>38</sup> The first attribute, plural rules' types, is used unchanged, while the second, plural rules' categories, was aggregated with the first to create a new feature, "plural rules" (see

<sup>33</sup> [http://www.unicode.org/cldr/charts/25/supplemental/territory\\_containment\\_un\\_m\\_49.html](http://www.unicode.org/cldr/charts/25/supplemental/territory_containment_un_m_49.html), accessed on 7/11/2019.

<sup>34</sup> [http://en.wikipedia.org/wiki/List\\_of\\_tz\\_database\\_time\\_zones](http://en.wikipedia.org/wiki/List_of_tz_database_time_zones), accessed on 8/11/2019.

<sup>35</sup> Coordinated Universal Time.

<sup>36</sup> Daylight Saving Time.

<sup>37</sup> [http://www.unicode.org/cldr/charts/25/supplemental/languages\\_and\\_scripts.html](http://www.unicode.org/cldr/charts/25/supplemental/languages_and_scripts.html), accessed on 7/11/2019.

<sup>38</sup> [http://www.unicode.org/cldr/charts/25/supplemental/language\\_plural\\_rules.html](http://www.unicode.org/cldr/charts/25/supplemental/language_plural_rules.html), accessed on 7/11/2019.

Fig. 10 for an example). The last property values, number of characters in the alphabet, were parsed from the additional supplementary materials for languages<sup>39,40</sup> and placed into five categories with meaningful ranges (Fig. 10).

**Network construction**

As before, we used the Tanimoto coefficient to build similarity layers, and we received acceptable similarity values (last two rows in Table 9). Similarly to MovieLens, we introduced a threshold to binarise bipartite interactions. If the percentage of population speaking given language is greater than or equal to 0.01% then the relation is considered to be positive, negative otherwise, giving us 819 positive and 436 negative edges in the network. As a result, we obtain a three layer graph with network characteristics as in Table 1 (see Fig. 11 for an illustration). As in the case of the MoveLens network, the resulting Unicodelang graph has acceptable sparsity, and is fully connected. The Unicodelang network can also be downloaded from our repository<sup>29</sup>.

**Appendix 2**

This appendix provides additional results for the parameter selection via internal cross-validation experiment performed in Section “Performance on generic data sets” (Tables 10, 11).

**Table 10** Spectral partitioning parameter optimization on generic data sets

Data set	Measure	Fold	Internal CV				External CV	
			Optimal parameters		AUC	AUPR	AUC	AUPR
			thresholding	m				
MovieLens	J <sub>CC</sub>	1	Local median	12	0.82	0.23	0.80	0.28
		2	Individual median	11	0.82	0.18	0.83	0.22
		3	Local median	10	0.82	0.23	0.82	0.25
		4	Local median	11	0.82	0.22	0.83	0.27
		5	Local median	13	0.82	0.24	0.83	0.26
	J <sub>NC</sub>	1	Local median	6	0.88	0.29	0.89	0.39
		2	Local median	5	0.88	0.29	0.88	0.37
		3	Local median	5	0.88	0.30	0.89	0.36
		4	Individual median	4	0.88	0.30	0.89	0.38
		5	Local median	6	0.88	0.30	0.88	0.36
Unicodelang	J <sub>CC</sub>	1	Individual median	54	0.70	0.05	0.72	0.02
		2	Individual median	54	0.68	0.05	0.65	0.01
		3	Individual median	55	0.69	0.03	0.68	0.02
		4	Individual median	54	0.70	0.04	0.66	0.02
		5	Individual median	57	0.68	0.02	0.68	0.01
	J <sub>NC</sub>	1	Individual mean	3	0.79	0.14	0.79	0.13
		2	Local median	9	0.77	0.13	0.77	0.11
		3	Individual sum	16	0.78	0.13	0.78	0.16
		4	Default	16	0.79	0.13	0.78	0.19
		5	Global mean	7	0.78	0.15	0.78	0.25

<sup>39</sup> Using Main Letters/Native attribute

<sup>40</sup> <http://www.unicode.org/cldr/charts/25/summary>, accessed on 7/11/2019.

**Table 11** Louvain algorithm parameter optimization on generic data sets

Data set	Measure	Fold	Internal CV			External CV	
			Optimal resolution	AUC	AUPR	AUC	AUPR
MovieLens	JC <sub>CC</sub>	1	0.3	0.79	0.18	0.79	0.23
		2	0.3	0.79	0.18	0.77	0.22
		3	0.3	0.80	0.19	0.76	0.23
		4	0.3	0.79	0.18	0.77	0.22
		5	0.3	0.79	0.18	0.77	0.22
	JC <sub>NC</sub>	1	0.6	0.88	0.26	0.88	0.34
		2	0.8	0.88	0.27	0.88	0.35
		3	0.6	0.88	0.26	0.89	0.34
		4	0.7	0.88	0.26	0.88	0.33
		5	0.6	0.88	0.26	0.88	0.33
Unicodelang	JC <sub>CC</sub>	1	0.6	0.73	0.10	0.72	0.08
		2	0.6	0.73	0.09	0.73	0.07
		3	0.6	0.73	0.09	0.74	0.10
		4	0.6	0.73	0.09	0.69	0.09
		5	0.6	0.73	0.10	0.73	0.15
	JC <sub>NC</sub>	1	0.7	0.80	0.14	0.80	0.12
		2	0.7	0.79	0.13	0.80	0.13
		3	0.8	0.79	0.14	0.84	0.16
		4	0.7	0.81	0.11	0.79	0.16
		5	0.7	0.80	0.14	0.79	0.25

**Abbreviations**

HTS: High-throughput screening; LPbyCD: The link-prediction-by-community-detection approach; CC: Community to community; NC: Node to community; CN: Common neighbors; JC: The Jaccard coefficient; PA: Preferential attachment; SR: The SimRank measure; CCN: Cannistraci-based common neighbors; CJC: Cannistraci-based Jaccard coefficient; NCB: Neighboring community-based; CRJC: Community relevance Jaccard coefficient; GPCR: G-protein coupled receptors; IC: Ion channels; NR: Nuclear receptors; IUPHAR: International union of basic and clinical pharmacology; AUC: Area under receiver operating characteristic curve; AUPR: Area under precision-recall curve; CV: Cross-validation; DTIP\_ MDHN: Drug–target interaction prediction method using marginalized denoising model on heterogeneous network; DTiGEMS+: A computational method that predicts Drug–Target interactions using graph embedding, graph mining, and similarity-based techniques; ALADIN: An advanced local Drug–Target interaction prediction technique; BLM-NII: Bipartite local model with neighbor-based interaction-profile inferring; WNN-GIP: Predicting drug–target interactions for new drug compounds using a weighted nearest neighbor profile; NetLapRLS: Laplacian regularized least square method improved for drug–protein interaction network; CD: Community detection; IMDB: Internet movie database; KONECT: Koblenz network collection; GDP PPP: Gross domestic product based on purchasing power parity; UTC: Coordinated universal time; DST: Daylight saving time.

**Acknowledgements**

Not applicable.

**Authors' contributions**

AZ, BC and LFS conceptualized the study. MK processed the data, implemented the models and performed the experiments. AZ, BC and LFS analyzed and interpreted the results. MK wrote and edited the draft. All authors reviewed the final manuscript. All authors read and approved the final manuscript.

**Funding**

MK was supported by the Research Grant - Doctoral Program from the region of Normandy. AZ and BC are supported by public funds from the University of Caen Normandy and LS by public fund from the University of Rouen Normandy.

**Availability of data and materials**

All data generated or analysed during this study are included in this published article.

## Declarations

### Competing interests

The authors declare that they have no competing interests.

### Author details

<sup>1</sup>UNICAEN, ENSICAEN, CNRS - UMR GREYC, Normandie Univ, 14000 Caen, France. <sup>2</sup>UNIROUEN, ULH, INSAR - LITIS-TIBS, Normandie Univ, 76800 Rouen, France.

Received: 29 April 2021 Accepted: 5 August 2021

Published online: 27 September 2021

## References

- Ahn Y, Bagrow JP, Lehmann S (2010) Link communities reveal multiscale complexity in networks. *Nature* 466(7307):761
- An WF, Tolliday N (2010) Cell-based assays for high-throughput screening. *Mol Biotechnol* 45(2):180–186
- Bleakley K, Yamanishi Y (2009) Supervised prediction of drug–target interactions using bipartite local models. *Bioinformatics* 25(18):2397–2403
- Blondel VD, Guillaume J, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. *J Stat Mech Theory Exp* 2008(10):P10,008
- Buza K, Peska L (2017) Aladin: a new approach for drug–target interaction prediction. In: *ECML/PKDD*. Springer, pp 322–337
- Buza K, Peška L, Koller J (2020) Modified linear regression predicts drug–target interactions accurately. *PLoS ONE* 15(4):e0230726
- Cannistraci CV, 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* 3:1613
- Chen H, Cheng F, Li J (2020) idrug: integration of drug repositioning and drug-target prediction via cross-network embedding. *PLoS Comput Biol* 16(7):e1008040
- Chen X, Liu MX, Yan GY (2012) Drug–target interaction prediction by random walk on the heterogeneous network. *Mol Biosyst* 8(7):1970–1978
- Cheng F, Zhou Y, Li W, Liu G, Tang Y (2012) Prediction of chemical–protein interactions network with weighted network-based inference method. *PLoS ONE* 7(7):e41064
- Clauset A, Moore C, Newman M (2008) Hierarchical structure and the prediction of missing links in networks. *Nature* 453(7191):98
- Damerau FJ (1964) A technique for computer detection and correction of spelling errors. *Commun ACM* 7(3):171–176
- Davis MI, Hunt JP, Herrgard S, Ciceri P, Wodicka LM, Pallares G, Hocker M, Treiber DK, Zarrinkar PP (2011) Comprehensive analysis of kinase inhibitor selectivity. *Nat Biotechnol* 29(11):1046
- De Bacco C, Power EA, Larremore DB, Moore C (2017) Community detection, link prediction, and layer interdependence in multilayer networks. *Phys Rev E* 95(4):042317
- Ding H, Takigawa I, Mamitsuka H, Zhu S (2013) Similarity-based machine learning methods for predicting drug–target interactions: a brief review. *Brief Bioinform* 15(5):734–747
- Ding J, Jiao L, Wu J, Liu F (2016) Prediction of missing links based on community relevance and ruler inference. *Knowl Based Syst* 98:200–215
- Ding J, Song J, Jiao L, Wu J, Liu F (2020) Multi-resolution prediction model based on community relevance for missing links prediction. *IEEE Access* 8:113981–113993
- Fortunato S (2010) Community detection in graphs. *Phys Rep* 486(3–5):75–174
- Gallier JH (2013) Notes on elementary spectral graph theory. Applications to graph clustering using normalized cuts. CoRR. [arXiv:abs/1311.2492](https://arxiv.org/abs/1311.2492)
- Goyal P, Ferrara E (2018) Graph embedding techniques, applications, and performance: a survey. *Knowl Based Syst* 151:78–94
- Guattery S, Miller GL (1995) On the performance of spectral graph partitioning methods. *SODA* 95:233–242
- Guimerà R, Sales-Pardo M (2009) Missing and spurious interactions and the reconstruction of complex networks. *Proc Natl Acad Sci* 106(52):22073–22078
- Harper FM, Konstan JA (2015) The movielens datasets: history and context. *ACM Trans Interact Intell Syst (tiis)* 5(4):1–19
- Herlocker JL, Konstan JA, Borchers A, Riedl J (1999) An algorithmic framework for performing collaborative filtering. In: *Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval, SIGIR 1999*, Association for Computing Machinery, Inc, pp 230–237
- Hristova D, Noulas A, Brown C, Musolesi M, Mascolo C (2016) A multilayer approach to multiplexity and link prediction in online geo-social networks. *EPJ Data Sci* 5(1):24
- Jalili M, Orouskhani Y, Asgari M, Alipourfard N, Perc M (2017) Link prediction in multiplex online social networks. *R Soc Open Sci* 4(2):160863
- Jiang H, Liu Z, Liu C, Su Y, Zhang X (2020) Community detection in complex networks with an ambiguous structure using central node based link prediction. *Knowl Based Syst* 195(105):626
- Kivelä M, Arenas A, Barthélemy M, Gleeson JP, Moreno Y, Porter MA (2014) Multilayer networks. *J Complex Netw* 2(3):203–271
- Koptelov M, Zimmermann A (2019) What can connectivity characteristics of networks tell us about the quality of link predictions? In: *GEM: graph embedding and mining@ ECML PKDD 2019*
- Koptelov M, Zimmermann A, Crémilleux B (2018) Link prediction in multi-layer networks and its application to drug design. In: *IDA*. Springer, pp 175–187

- Kuncheva Z, Montana G (2015) Community detection in multiplex networks using locally adaptive random walks. In: ASONAM. ACM, pp 1308–1315
- Kunegis J (2013) Konect: the koblenz network collection. In: Proceedings of the 22nd international conference on world wide web, pp 1343–1350
- Lambiotte R, Delvenne JC, Barahona M (2008) Laplacian dynamics and multiscale modular structure in networks. arXiv preprint [arXiv:08121770](https://arxiv.org/abs/08121770)
- Leskovec J, Rajaraman A, Ullman JD (2014) Mining of massive datasets. Cambridge University Press, Cambridge
- Li X, Chen H (2013) Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach. *Decis Support Syst* 54(2):880–890
- Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. *J Am Soc Inf Sci Technol* 58(7):1019–1031
- Lim H, Gray P, Xie L, Poleksic A (2016) Improved genome-scale multi-target virtual screening via a novel collaborative filtering approach to cold-start problem. *Sci Rep* 6(38):860
- Liu Z, He JL, Kapoor K, Srivastava J (2013) Correlations between community structure and link formation in complex networks. *PLoS ONE* 8(9):e72908
- Luo Y, Zhao X, Zhou J, Yang J, Zhang Y, Kuang W, Peng J, Chen L, Zeng J (2017) A network integration approach for drug-target interaction prediction and computational drug repositioning from heterogeneous information. *Nat Commun* 8(1):1–13
- Mei JP, Kwok CK, Yang P, Li XL, Zheng J (2013) Drug-target interaction prediction by learning from local information and neighbors. *Bioinformatics* 29(2):238–245
- Mohamed S, Nováček V, Nounu A (2020) Discovering protein drug targets using knowledge graph embeddings. *Bioinformatics* 36(2):603–610
- Nemenyi PB (1963) Distribution-free multiple comparisons. Princeton University, Princeton
- Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. *Phys Rev E* 69(2):026113
- Olayan RS, Ashoor H, Bajic VB (2018) Ddr: efficient computational method to predict drug-target interactions using graph mining and machine learning approaches. *Bioinformatics* 34(7):1164–1173
- Shahriari S, Shahriari M, MD Noor R (2015) A community-based approach for link prediction in signed social networks. *Sci Program* 2015
- Shao J, Zhang Z, Yu Z, Wang J, Zhao Y, Yang Q (2019) Community detection and link prediction via cluster-driven low-rank matrix completion. In: *IJCAI*, pp 3382–3388
- Soundarajan S, Hopcroft J (2012) Using community information to improve the precision of link prediction methods. In: *WWW*, ACM, pp 607–608
- Spielman DA, Teng SH (2007) Spectral partitioning works: planar graphs and finite element meshes. *Linear Algebra Appl* 421(2–3):284–305
- Sun J, Qu H, Chakrabarti D, Faloutsos C (2005) Neighborhood formation and anomaly detection in bipartite graphs. In: *ICDM*. IEEE, pp 418–425
- Tagarelli A, Amelio A, Gullo F (2017) Ensemble-based community detection in multilayer networks. *DMKD* 31(5):1506–1543
- Tang C, Zhong C, Chen D, Wang J (2020) Drug–target interactions prediction using marginalized denoising model on heterogeneous networks. *BMC Bioinform* 21(1):1–29
- Thafar M, Olayan R, Ashoor H, Albaradei S, Bajic V, Gao X, Gojobori T, Essack M (2020) Dtigems+: drug–target interaction prediction using graph embedding, graph mining, and similarity-based techniques. *J Cheminform* 12(1):1–17
- Valverde-Rebaza JC, de Andrade Lopes A (2014) Link prediction in online social networks using group information. In: *ICCSA*. Springer, pp 31–45
- Van Laarhoven T, Marchiori E (2013) Predicting drug–target interactions for new drug compounds using a weighted nearest neighbor profile. *PLoS ONE* 8(6)
- Wang W, Yang S, Zhang X, Li J (2014) Drug repositioning by integrating target information through a heterogeneous network model. *Bioinformatics* 30(20):2923–2930
- Xia Z, Wu LY, Zhou X, Wong STC (2010) Semi-supervised drug-protein interaction prediction from heterogeneous biological spaces. In: *BMC systems biology*, BioMed Central, vol 4, p S6
- Xie Z, Dong E, Li J, Kong D, Wu N (2014) Potential links by neighbor communities. *Physica A Stat Mech Appl* 406:244–252
- Xu X, Shang K, Xiao J (2020) Quantifying the effect of community structures for link prediction by constructing null models. *IEEE Access* 8:89269–89280
- Yamanishi Y, Araki M, Gutteridge A, Honda W, Kanehisa M (2008) Prediction of drug–target interaction networks from the integration of chemical and genomic spaces. *Bioinfo* 24(13):i232–i240
- Yan B, Gregory S (2012) Finding missing edges in networks based on their community structure. *Phys Rev E* 85(5):056112
- Zheng X, Ding H, Mamitsuka H, Zhu S (2013) Collaborative matrix factorization with multiple similarities for predicting drug–target interactions. In: *KDD*. ACM, pp 1025–1033

## Publisher's Note

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