Integrating Structural Affinity via Graph Product for Relational Learning

Xuwen Lang, Pin Zhang, and Dehong Qiu

Abstract—**Identifying the existence of relationship between the entities across multiple domains is fundamental to scientific discovery. The entities in different domains are usually described through different structured or semi-structured data. It is interesting to integrate structural affinity of entities in individual domains in relational learning because the structured data collected from various domains are often complementary to each other. However, this is a difficult task and has not yet been solved well. In this paper, we propose a novel approach to deal with this challenge that hampers relational learning across multiple domains. The key idea of our approach is to integrate multiple structural affinities through graph product operations that can map the structural affinities in individual domains onto a single integrated structural affinity. Furthermore, the proposed approach allows for emphasizing the function of the known relationships during integration. The problem of relationship discovery between entities is subsequently cast as a problem of node classification on the product graph. Since the integrated structural affinity captures higher order relationships of the entities from different domains, it is no surprise that we obtain more reliable results of relational learning. We validate the new approach on a real-world dataset. Experimental studies show that the proposed approach outperforms its competitors on the benchmark dataset.**

Index Terms—**Graph product, information fusion, relational learning, structured data.**

I. Introduction

Determining whether entities in one domain are related to entities in another domain is a fundamental problem in science and engineering. The traditional approach for determining whether two entities are related to, or statistically dependent on each other is the correlation analysis of the properties of the entities. However, as data collection techniques develop quickly, we can obtain not only the property data of each entity, but also the structured data of all entities. We now desire approaches that can correctly detect the relationship or dependence between entities according to their properties and structures together. That is, given an entity v_i belonging to the set V_1 of the graph $G_1(V_1, E_1)$ in the domain $\mathfrak{D}_1$ and an entity v_j belonging to the set V_2 of another graph $G_2(V_2, E_2)$ in another domain $\mathfrak{D}_2$, the question is to determine the relationship $r(v_i, v_j)$ between v_i and v_j exists $r(v_i, v_j) = 1$ or not $r(v_i, v_j) = 0$ according to the properties of the entities and the structures $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ together. Such question arises in many applications such as high-throughput screening [1], developing imaging biomarkers for diseases [2], causal analyses [3], and machine learning tasks [4], where structured data from diverse sources are available and need to be combined for relational learning.

Structured data, however, present challenges for learning relationship between the entities across multiple domains. The first challenge is how to combine the internal structures in individual domains for joint learning the relationship between the entities across different domains. Otherwise, we cannot take advantages of the complementary information provided by each graph. Second, the integration of multiple graphs usually leads to a substantially high computation cost. We desire methods that can efficiently tackle the increasing complexity of graph combination. Additionally, the supervised information, i.e., the known relationships between entities in different domains, is typically extremely sparse in relational learning, which requires us effectively leverage the massively available unlabeled relationships in addition to the labeled ones. Therefore it is becoming more desirable to develop efficient and accurate relational learning approaches, so as to better handle a large variety of disparate structured data across different domains.

A number of fusion methods have been proposed to integrate structured data from multiple sources. One popular way is kernel-based techniques that have been widely studied for fusing multi-source graphs, because multiple kernels can be combined through algebraic operations, which results in a desired kernel matrix that combines multi-source graphs. However, kernel techniques are not appropriate to combine graphs in different domains, because each kernel matrix corresponds to a typical similarity description among a single type of objects [5], [6]. Tensor-based methods can combine graphs among multiple types of objects [7], [8]. However, because the internal graph structure for each type of entities is usually ignored, most tensor models lack convexity, which leads to ill-posed optimization problems particularly in high-order scenarios.

Besides kernel-based and tensor-based techniques, graph product can map multiple graphs onto a single graph. Graph product-based fusion techniques have received more and more attention. This kind of fusion methods enjoys the flexible choices in the formulation of graph products. In this paper, we propose a novel approach to integrating multiple graphs based on graph product. The graph product operation map individual graphs of different kinds of entities onto a single integrated graph. A key step in our approach is to define the weighed function of the edges of the product graph, which not only introduces structural constraints but also leverages the effectiveness of the known relationships.

Manuscript received June 9, 2020; revised April 22, 2021. This work was supported in part by the National Natural Science Foundation of China 61873100.

All authors are with the School of Software Engineering, Huazhong University of Science and Technology, Wuhan, Hubei, China (e-mail: m201776139@hust.edu.com, m201776184@hust.edu.com, qiudehong@163.com).

Additionally, another critical step is to integrate the higher order relations between entities in individual graphs into the product graph. The higher order relations can capture more information than the pairwise relations. Subsequently, we cast the problem of relational learning across multiple graphs as a problem of label classification of the integrated nodes of the product graph. We demonstrate the utility of the proposed approach on real-world data.

The rest of this paper is organized as follows: In Section II, we review the existing approaches that combine structured data using graph products. In Section III, we describe our approach in detail. We present experimental studies on a benchmark dataset in Section IV. Finally, we conclude our findings in Section V.

II. Related Work

A graph product is a binary operation on graphs, which produces a graph that is called product graph [9]. Graph products have been widely investigated in mathematics and have many significant applications. Since graph products can map multiple graphs onto a single graph, they have been used to integrate structured data. The fusion methods based on graph product have already been used in different applications, including biological and biomedical networks [10], [11], signal and image processing [12], [13], computational sciences and data mining [14], collaborative filtering [15], citation network analysis [16], product recommendation [17], and so on. A desirable property of graph product is that it provides a natural reduction from the original problem over multiple graphs to a problem over a single graph. Furthermore, the fusion methods based on graph products enjoy the flexible choices in the formulation of graph products. Therefore, graph product-based fusion techniques are attracting more and more researchers' attention.

Although the fusion methods based on graph products have produced exciting results, there are still some important limitations. One major weakness of these fusion methods based on graph products is the computation cost will increase quickly with the size and the number of factor graphs. One strategy to tackle this problem is to apply low-rank approximation on each factor graph [17]. Another one is to consider local but not whole structure of individual graphs [18], [19]. In this paper, we emphasize the function of the dominant local structure, that is, the substructure around the known relationships, during integration. Another problem is that the product graph usually becomes larger and only limited training data are available, which would influence negatively the performance of relational learning. One typical strategy to tackle this problem is the transductive learning, which takes advantage of both labeled and unlabeled relationships [16]. Here, we make use of the higher order structural affinity between entities in product graph, which leads to a better performance in experiments.

III. Our Approach

In this section, we present the new approach to integrating structural affinity based on graph product. Given graphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$, $V(*)$ and $E(*)$ is the node set and the edge set of the graph $*$ ($* = G$ or H) respectively. $v(*)_i \in V(*)$ corresponds to the i^{th} entity from the domain $*$. $W(*)$ is the weight matrix of the graph $*$, whose entry in row i and column j stores the weight $w(*)_{i,j}$ of the edge $\left(v(*)_i, v(*)_j\right) \in E(*)$. Suppose L be the set of the known relationships between $V(G)$ and $V(H)$. The task is to determine the unknown relationship $l\left(v(G)_i, v(H)_j\right) \notin L$ between the entity $v(G)_i \in V(G)$ and the entity $v(H)_j \in V(H)$ exists or not. This relational learning problem is illustrated in Fig. 1.

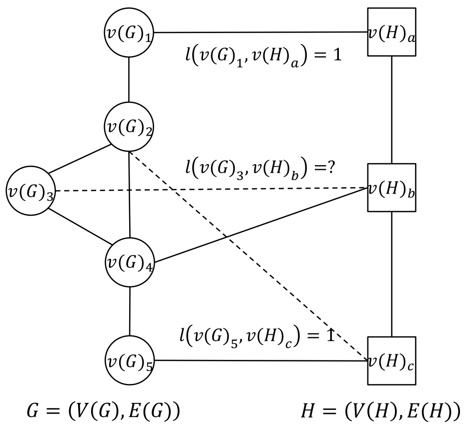


Fig. 1. Illustration of the relational learning problem.

A. Graph Product

Graph product is a fundamental tool with rich applications in both graph theory and theoretical computer science. For a general overview we refer the interested readers to [9]. According to [9], a graph product is the result of a multiplication operation defined for graphs, which is defined as below.

Definition 1 (Graph Product). Given two factor graphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$ and the graph product operator “ □ ”, the graph product of $G = (V(G), E(G))$ and $H = (V(H), E(H))$ is denoted by $P(V(P), E(P)) = G\square H$, which is a graph with the node set $V(P) = V(G) \times V(H)$ and the edge set $E(P) = E(G) \square E(H)$.

Different realizations of the graph product operator “□” will lead to the different graph product $P(V(P), E(P))$, which provides flexibility for researchers when they solve different problems. In this paper, we are mainly concerned with Cartesian graph product. Other graph products have a vertex set that is the Cartesian product of the factors’ vertex sets and differ only in their edge sets. Cartesian graph product is defined as below.

Definition 2 (Cartesian Graph Product). Given two factor graphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$, the Cartesian product of these two graphs denoted by $G\otimes H$ is defined to be a graph whose vertex set is $V(G) \times V(H)$, where $V(G) \times V(H)$ is the Cartesian product of the sets $V(G)$ and $V(H)$ and any two distinct vertices $\left(v(G)_i \diamond v(H)_j\right)$ and $(v(G)_m \diamond v(H)_n)$ of $G\otimes H$ are adjacent if

(i) $v(G)_i = v(G)_m$ and $\left(v(H)_j, v(H)_n\right) \in E(H)$, or

(ii) $(v(G)_i, v(G)_m) \in E(G)$ and $v(H)_j = v(H)_n$.

Fig. 2 shows the results of the Cartesian product of the two graphs shown in Fig. 1.

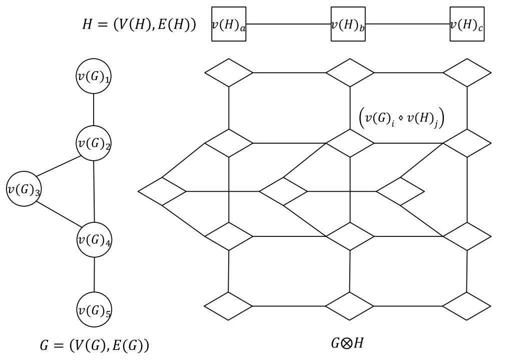


Fig. 2. An example of Cartesian graph product.

B. Enhancing Structural Affinity of Individual Graph

The factor graph, for example, $G = (V(G), E(G))$, expresses the structure of the entities of the same type in a certain domain. The corresponding weight matrix $W(G)$ gives the affinity between entities that are directly linked. However, in most applications, the weight matrix is usually sparse because it is expensive to collect the data completely or it is impossible to extract the all structure between entities. In order to make full use of the known links, we enhance the structural affinities between entities on single graph by taking into account not only the direct links but also the *k*-step chains. We modify the weight matrix W like below,

$$W' = \beta W + \beta^2 W^2 + \cdots + \beta^k W^k$$

The column sums of W give the affinity of the direct links to the entity corresponding to each column. Also, the column sums of W^2 give the affinity of two-step entities to the entity corresponding to each column. The column sums of W^k, affinity of *k*-step entities, etc. The parameter β is used to allow for the effectiveness of longer chains, which has the force of a probability of the effectiveness of a single link. Usually, the parameter β is selected smaller than the largest eigenvalue of W, depending on the context of the particular investigation. A *k*-step chain, then, has probability β^k of being effective.

C. Integrating Structural Affinities of Multiple Graphs

As shown in Fig. 2, the Cartesian graph product maps the graph G and the graph H onto a single graph $G \otimes H$. The definition 2 only gives the condition that two vertices of the product graph $G \otimes H$ that are connected should satisfy. However, the definition 2 does not define the weight function of the product graph $G \otimes H$. In this paper, we take into considerations of the known relationships between the graph G and the graph H when we calculate the weight of the edges of the product graph $G \otimes H$.

Fig. 3 shows the three typical cases happened during weighting the edges of the product graph $G \otimes H$. Fig. 3(a) shows the case that $v(G)_i$ and $v(G)_j$ are adjacent and there is a known relationship between the node $v(G)_j$ and the node $v(H)_a$. We define the weight of the edge that connects node $(v(G)_i \diamond v(H)_a)$ and $(v(G)_j \diamond v(H)_a)$ as below.

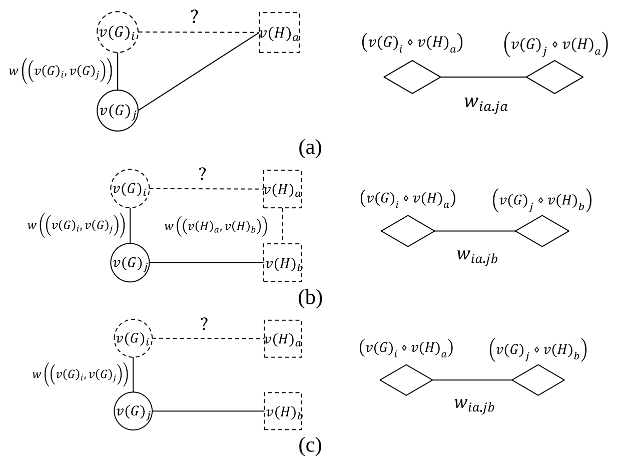
$$w_{ia.ja} = w\left((v(G)_i \diamond v(H)_a), \left(v(G)_j \diamond v(H)_a\right)\right)$$
$$= w\left(\left(v(G)_i, v(G)_j\right)\right)$$

This definition guarantees that the more closely related $v(G)_i$ and $v(G)_j$ are, the more possibly the relationship between $v(G)_i$ and $v(H)_a$ exists.

Similarly, the weight of the edge connecting node $(v(G)_i \diamond v(H)_a)$ and $\left(v(G)_j \diamond v(H)_b\right)$ shown in Fig. 3(b) is defined as below.

$$w_{ia.jb} = w\left((v(G)_i \diamond v(H)_a), \left(v(G)_j \diamond v(H)_b\right)\right)$$
$$= w\left(\left(v(G)_i, v(G)_j\right)\right) \times w\left((v(H)_a, v(H)_b)\right)$$

This definition presumes that nodes $v(G)_i$ and $v(H)_a$ are related when they are respectively related to their neighbors $v(G)_j$ and $v(H)_b$ that are already related.



?
$v(G)_i$
$v(H)_a$
$w\left(\left(v(G)_i, v(G)_j\right)\right)$
$v(G)_j$
$(v(G)_i \diamond v(H)_a)$
$(v(G)_j \diamond v(H)_a)$
$w_{ia.ja}$
(a)
?
$v(G)_i$
$v(H)_a$
$w\left(\left(v(G)_i, v(G)_j\right)\right)$
$w\left(\left(v(H)_a, v(H)_b\right)\right)$
$v(G)_j$
$v(H)_b$
$(v(G)_i \diamond v(H)_a)$
$(v(G)_j \diamond v(H)_b)$
$w_{ia.jb}$
(b)
?
$v(G)_i$
$v(H)_a$
$w\left(\left(v(G)_i, v(G)_j\right)\right)$
$v(G)_j$
$v(H)_b$
$(v(G)_i \diamond v(H)_a)$
$(v(G)_j \diamond v(H)_b)$
$w_{ia.jb}$
(c)

Fig. 3. Definition of weight for different cases.

The case shown in Fig. 3(c) indicates a situation where $v(H)_a$ and $v(H)_b$ are not directly connected. The weight of the edge connecting node $(v(G)_i \diamond v(H)_a)$ and $\left(v(G)_j \diamond v(H)_b\right)$ is defined as below.

$$w_{ia.jb} = w\left((v(G)_i \diamond v(H)_a), \left(v(G)_j \diamond v(H)_b\right)\right)$$
$$= \alpha \times w\left(\left(v(G)_i \diamond v(G)_j\right)\right)$$

where, α is a constant that is selected to be much smaller than the minimum value of $W(H)$.

D. Vertex Label Propagation over Product Graph

The graph product operation maps individual graphs of different kinds of entities onto a single integrated graph. The problem of determining the relationship $l\left(v(G)_i, v(H)_j\right)$ between the entity $v(G)_i \in V(G)$ and the entity $v(H)_j \in V(H)$ exists or not is cast as a problem of determining the label of the node $\left(v(G)_i, v(H)_j\right)$ of the product graph. We use the vertex label propagation algorithm to determine the label of the nodes of the product graph.

Given the weighted product graph $G_P = (V_P, E_P, W_P)$, consisting of a set of vertices $V_P = V(G) \times V(H)$, a set of edges E_P, and a nonnegative weight function $W_P: E_P \to [0,1]$, the label propagation algorithm computes the node scores for each label to determine the label of unlabeled nodes. The labeling scores are defined as the optimal solution that minimizes a cost function.

Let the $|V_P| \times |C|$ size matrix F correspond to a classification on the node set V_P by labeling each node $v \in V_P$ a label $c \in C$, which holds the labeling scores of all nodes for each label. C is the set of class labels. $Y \in R^{|V_P|\times|C|}$ is a matrix where $y_{ij} = 1$ if node point $v_i \in V_P$ is initially labeled as $y(v_i) = l_j$ and $y(v_i) = 0$ otherwise. Let F_i and Y_i be the i-th row vector of F and Y, respectively, i.e., $F = \left[F_1, F_2, \cdots, F_{|V_P|}\right]^T$ and $Y = \left[Y_1, Y_2, \cdots, Y_{|V_P|}\right]^T$, the cost function $C(F)$ associated with classification matrix F is defined as follows:

$$C(F) = \frac{1}{2}\sum_{i,j=1}^{n} W_P^{ij} \left\| \frac{F_i}{\sqrt{D_{ii}}} - \frac{F_j}{\sqrt{D_{jj}}} \right\|^2 + \left(\frac{1}{\gamma} - 1\right)\sum_{i=1}^{n} \|F_i - Y_i\|^2$$

where, D is a diagonal matrix, whose i-th entry is given by $D_{ii} = \sum_{j=1}^{n} W_P^{ij}$; γ is a constant parameter, which is selected from the range $[0, 1]$.

The first term in the cost function $C(F)$ corresponds to the smoothness constraint, which means that a good classifying function should not change too much between similar nodes. The second term of $C(F)$ corresponds to the fitting constraint, which means good classification should not change too much from the initial label assignment. The cost function $C(F)$ is designed to enhance the accuracy of label prediction. Minimizing the cost function yields the optimal F in the following closed form:

$$F = \left(I - \gamma D^{-1/2} W_P D^{-1/2}\right)^{-1} Y$$

where I is an identity matrix of size $n \times n$.

IV. Experiments

In this section, we empirically evaluate the proposed approach on a practical dataset. We first describe the benchmark dataset and the data preprocessing briefly. Subsequently, we introduce the evaluation metrics. Finally, we provide the experimental results in detail.

A. Dataset

We utilized the Enzyme benchmark dataset to evaluate our approach. Table I shows the statistical information of the dataset. The Enzyme dataset was originally proposed by Yamanishi and coworkers [20], which includes 445 drugs and 664 targets with 2926 known interactions between them. The Enzyme dataset consists of three matrices. One is the drug similarity matrix $W_d \in R^{445\times445}$, which denotes the chemical similarities between drugs computed by using the SIMCOMP tool [21]. We construct the graph $G_d = (V_d, E_d, W_d)$ to represent the pairwise relationships between the drugs, where the set of nodes V_d represents the drugs and the set of edges E_d, which is weighted by W_d, represents the relationships between drugs. Another matrix of the Enzyme dataset is the target similarity matrix $W_t \in R^{664\times664}$, which denotes the sequence similarities between protein targets calculated by using normalized Smith–Waterman scores [22]. Similarly, we construct the graph $G_t = (V_t, E_t, W_t)$ to represent the pairwise relationships between the targets. The last one is the drug-target interaction matrix $L_{dt} \in R^{445\times664}$, whose value l_{ij} is 1 if the drug i interacts with target j and 0 otherwise. The 0 value in the drug-target interaction matrix does not necessarily mean that the corresponding target is irrelevant to the drug. We construct a bipartite graph to represent the interaction between drugs and targets, where there are only 2926 crossing links that represent the known drug-target interactions. We use cross-validation with balanced folds to find the best model. Specifically, we split the train data into 10 subsets of roughly equal size. One subset is randomly selected as the test set.

TABLE I: Statistics of the Benchmark Dataset

Dataset	Drugs	Targets	Links	Sparsity
Enzyme	445	664	2926	0.01

B. Evaluation Metrics

The performance of a supervised binary classifier is usually evaluated by the metrics including True Positive Fraction (TPF), True Negative Fraction (TNF), False Positive Fraction (FPF) and False Negative Fraction (FNF). They are not independent, satisfying TPF+FNF=1 and FPF+TNF=1. A good 2-class classifier should maximize TNF with very low FNF, or minimize FPF with very high TPF. Usually, we only use TPF and FPF to evaluate the performance. To obtain the best tradeoff between TPF and FPF, we could plot the Receiver Operating Characteristic (ROC), which is a plot of TPF versus FPF contained within the unit square. The Area Under a ROC (AUC) is a comprehensive metric. AUC lies between 0 and 1 and the greater the AUC, the better the binary classifier.

However, because the negative label does not necessarily mean that the corresponding target is irrelevant to the drug, the metric AUC may not be able to reflect the truth precisely. Recall is more important in a situation like this, which is defined as below.

$$Recall = \frac{TP}{TP + FN}$$

In this paper, we use AUC and Recall to evaluate the performance of the proposed approach.

C. Experimental Results

In order to assess the performance of our approach, we compare it with other three popular algorithms, that is, DBSI [23], NetCBP [24] and PUDTI [25], on the Enzyme dataset. DBSI is a network-based inference method that only uses drug-target bipartite network topology similarity to infer new targets for known drugs. NetCBP is a semi-supervised learning method that uses labeled and unlabeled interaction information to predict drug-protein interactions. PUDTI was designed to deal with the lack of negative samples in the Enzyme dataset. It first screened strong negative drug-protein interaction examples based on positive-unlabeled learning, and then used SVM-based optimization model to classify the remaining ambiguous samples.

Table II summarizes the comparison results between the proposed approach and the previous prediction algorithms. Most of the scores reported here for the previous prediction approaches agreed with those reported in the original papers or review paper. It can be found that our approach outperformed DBSI and NetCBP, being statistically significant, but did not work as well as PUDTI in terms of AUC. Probably this was because of the fact that PUDTI screened many negative drug-protein interaction examples during prediction, which would be benefit to the tradeoff between TPF and FPF.

However, our approach outperformed the three competing approaches in terms of recall. We believe that this improvement is due to that the influences of the known drug-protein interactions were utilized appropriately during the integration of structural affinities via graph product. These comparison results indicate that our approach can achieve better or approximate predictive performance, which proves that the graph product can effectively integrate multiple structural affinities into a single graph.

TABLE II: COMPARISON RESULTS OF CROSS VALIDATION ON ENZYME DATASET BETWEEN OUR APPROACH AND ITS COMPETITORS

Approach	AUC	Recall
DBSI[23]	0.806	0.808
NetCBP[24]	0.825	0.816
PUDTI[25]	0.898	0.845
Our Approach	0.856	0.868

Furthermore, we investigated the influences of the number of training drug-protein interactions on recall. Fig. 4 shows the change of recall of our approach with the proportion of drug-protein interactions used in train. It is clear that the recall increases with the increase of the number of training drug-protein interactions. This suggests that the prediction of drug-protein interactions depends upon the known interactions mainly. The more the validated training interactions, the more successfully we predicted the test interactions. Assuming the unvalidated drug-target interactions as negative samples is a double-edged sword. It may be able to raise the value of AUC, but may worsen the value of recall.

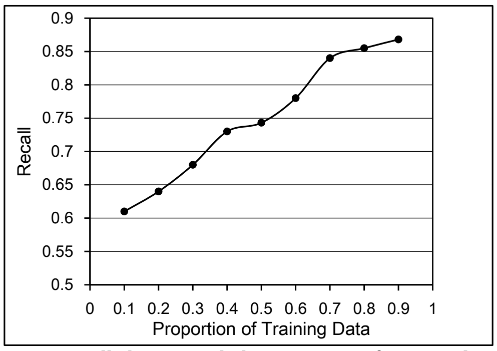


Fig. 4. Recall changes with the proportion of training data.

V. CONCLUSIONS

We demonstrated that integrating structural affinities of entities could be effective for learning relationships between the entities in different domains. Our approach is able to integrate structural affinities from multiple graphs efficiently. To achieve this goal, we first enhanced the structural affinities of entities in each individual graphs by taking into account not only the direct links but also the *k*-step chains between entities. Then, we integrated the structural affinities of multiple graphs through the operation of graph product. Subsequently, we defined the weight function of the product graph, which leveraged the function of the known relationships between the entities. As a result, the problem of relational learning across multiple graphs was cast as a problem of label identification of the combined nodes of the product graph. The effectiveness of our method was validated on a practical benchmark dataset. Experimental results showed that the proposed approach outperformed several popular approaches in the real-world task of drug-protein interaction prediction. It is expected that our approach could be further improved and widely used for relational learning in other application domains.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Xuwen Lang, Pin Zhang, and Dehong Qiu concuted the research; Xuwen Lang, Pin Zhang and Dehong Qiu analyzed the data; Xuwen Lang wrote the paper. All authors had approved the final version.

ACKNOWLEDGMENT

The authors would like to thank their colleagues and the anonymous reviewers for their helpful comments and suggestions.

REFERENCES

[1] J. H. Zhang, T. D. Chung, and K. R. Oldenburg, “A simple statistical parameter for use in evaluation and validation of high throughput screening assays,” *J. Biomol. Screen.*, vol. 4, pp. 67-73, Mar. 1999.

[2] J. W. Prescott, “Quantitative imaging biomarkers: The application of advanced image processing and analysis to clinical and preclinical decision making,” *J. Digit. Imaging*., vol. 26, pp. 97-108, Feb. 2013.

[3] J. Pearl, *Causality: Models, Reasoning, and Inference*, Cambridge University Press, New York, NY, United States, 2009.

[4] T. Hastie, R. Tibshirani, and J. H. Friedman, *Elements of Statistical Learning*, Springer, New York, NY, United States, 2001.

[5] H. Song, J. J. Thiagarajan, P. Sattigeri, K. N. Ramamurthy, and A. Spanias, “A deep learning approach to multiple kernel fusion,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 2292-2296.

[6] S. Kung, *Kernel Methods and Machine Learning*, Cambridge: Cambridge University Press, 2014.

[7] S. Rabanser, O. Shchur, and S. Günnemann, “Introduction to tensor decompositions and their applications in machine learning,” *arXiv e-prints*, arXiv: 1711.10781, 2017.

[8] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, “Tensors for data mining and data fusion: Models, applications, and scalable algorithms,” *ACM T. Intel. Syst. Tec.*, vol. 8, p. 16, Aug. 2016.

[9] W. Imrich and S. Klavzar, *Product Graphs*. Wiley-Interscience, New-York, 2000.

[10] M. Xie, T. Hwang, and R. Kuang, “Prioritizing disease genes by bi-random walk,” in. *Lecture Notes in Computer Science*, vol. 7302, P. N. Tan, S. Chawla, C. K. Ho, and J. Bailey, Eds., Springer, Berlin, Heidelberg, 2012, pp. 292-303.

[11] Z. L. Li, R. Petegrosso, S. Smith, D. Sterling, G. Karypis, and R. Kuang, “Scalable label propagation for multi-relational learning on tensor product graph,” arXiv: 1802.07379, 2018.

[12] X. Yang and L. J. Latecki, “Affinity learning on a tensor product graph with applications to shape and image retrieval,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 2369-2376.

[13] D. E. Dudgeon and R. M. Mersereau, *Multidimensional Digital Signal Processing*, Prentice Hall, 1983.

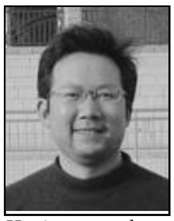
[14] E. Acar, R. J. Harrison *et al.*, “Future directions in tensor-based computation and modeling,” Workshop Report, Arlington, VA, 2009.

[15] R. Raymond and H. Kashima, “Fast and scalable algorithms for semi-supervised link prediction on static and dynamic graphs,” in *Lecture Notes in Computer Science*, vol. 6323, J. L. Balcázar, F. Bonchi, A. Gionis, and M. Sebag, Eds. Springer, Berlin, Heidelberg, 2010, pp. 131-147.

[16] H. X. Liu and Y. M. Yang, "Cross-graph learning of multi-relational associations," in *Proc. the 33rd International Conference on International Conference on Machine Learning*, 2016, pp. 2235-2243.
[17] H. X. Liu and Y. M. Yang, "Bipartite edge prediction via transductive learning over product graphs," in *Proc. the 32nd International Conference on Machine Learning*, 2015, pp. 1880-1888.
[18] S. Jendoubi, A. Martin, L. Lietard, and B. B. Yaghlane, "Classification of message spreading in a heterogeneous social network," *IPMU*, pp. 66–75, 2014.
[19] M. Pavan and M. Pelillo, "Dominant sets and pairwise clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 167-172, Jan. 2007.
[20] Y. Yamanishi, M. Araki, A. Gutteridge, W. Honda, and M. Kanehisa, "Prediction of drug-target interaction networks from the integration of chemical and genomic spaces," *Bioinformatics*, vol. 24, i232-i240, July 2008.
[21] M. Hattori, Y. Okuno, S. Goto, and M. Kanehisa, "Development of a chemical structure comparison method for integrated analysis of chemical and genomic information in the metabolic pathways," *J. Am. Chem. Soc.*, vol. 125, pp. 11853-11865, Oct. 2003.
[22] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. Mol. Biol*., vol. 147, pp. 195-197, Mar. 1981.
[23] F. Cheng, C. Liu, J. Jiang, W. Lu, W. Li, G. Liu, W. Zhou, J. Huang, and Y. Tang, "Prediction of drug-target interactions and drug repositioning via network-based inference," *PLOS Comput. Biol.,* vol. 8, pp. 357-372, May 2012.
[24] H. Chen and Z. Zhang, "A semi-supervised method for drug-target interaction prediction with consistency in networks," *PLOS ONE*, vol. 8, e62975, May 2013.
[25] L. Peng, W. Zhu, B. Liao, Y. Duan, M. Chen, Y. Chen, and J. Yang, "Screening drug-target interactions with positive-unlabeled learning," *Sci. Rep.*, vol. 7, p. 8087, Aug. 2017.

Xuwen Lang received the bachelor's degree in electrical engineering and automation from the University of Jinan, Jinan, China. He is currently pursuing the M.Sc. degree in Huazhong University of Science and Technology, Wuhan, China. His research interests include information security and machine learning.

Pin Zhang received the bachelor's degree in software engineering from Wuhan Institute of Technology, Wuhan, China. He is currently pursuing the M.Sc. degree in Huazhong University of Science and Technology, Wuhan, China. His research interests include software engineering, network security, and machine learning.



Dehong Qiu received the Ph.D. degree in electrical engineering from Hongkong Polytechnic University, Hongkong, China, in 2001. He was a post-doctoral researcher with the School of Computer Science, Huazhong University of Science and Technology, Wuhan, China, from 2001 to 2002.

From 2003 to 2004, he was an associate researcher with the Department of Electrical Engineering, Hongkong Polytechnic University, Hongkong, China. He is currently a professor with the School of Software Engineering, Huazhong University of Science and Technology, Wuhan, China. His research interests include machine learning, data mining, software engineering, and big data engineering.