This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI

|  |  |
| --- | --- |
| 10.1109/ACCESS.2019.2942357, IEEE Access |  |

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

*Digital Object Identifier 10.1109/ACCESS.2017.DOI*

**Preparation of Papers for IEEE ACCESS**

**ZEGUANG LIU1, YAO LI2, AND HUILIN LIU.3**   
1Northeastern University, Computer Science NO. 3-11, Wenhua Road, Heping District, Shenyang, P. R. China (e-mail: 1310382@stu.neu.edu.cn) 2Northeastern University, Computer Science NO. 3-11, Wenhua Road, Heping District, Shenyang, P. R. China (e-mail: 1610544@stu.neu.edu.cn) 3Northeastern University, Computer Science NO. 3-11, Wenhua Road, Heping District, Shenyang, P. R. China (e-mail: liuhuilin@mail.neu.edu.cn)

Corresponding author: HuiLin Liu (e-mail: liuhuilin@mail.neu.edu.cn).

This work was supported by the NSFC (Grant No. 61872072), the State Key Laboratory of Computer Software New Technology Open Project Fund (Grant No. KFKT2018B05), and the National Key R&D Program of China (Grant No. 2016YFC1401900).

|  |  |
| --- | --- |
|  | **ABSTRACT** Link prediction is an important issue in graph data mining. In social networks, link prediction is used to predict missing links in current networks and new links in future networks. This process has a wide range of applications including recommender systems, spam mail classification, and the identification of domain experts in various research areas. In order to predict future node similarity, we propose a new model, Common Influence Set, to calculate node similarities. The proposed link prediction algorithm uses the common influence set of two unconnected nodes to calculate a similarity score between the two nodes. We used the area under the ROC curve (AUC) to evaluate the performance of our algorithm and that of previous link prediction algorithms based on similarity over a range of problems. Our experimental results show that our algorithm outperforms previous algorithms.  **INDEX TERMS** Link Prediction, Common Influence, Similarity Index |

**I. INTRODUCTION**   
**S** constantly changing. With the passage of time, links between number of nodes and links, and the network structure is OCIAL networks are complex, and usually have a large

nodes may disappear or be re-established. These changes are closely related to changes in information. A large number of studies and analyses of link prediction in complex networks show that network structure and information at different times can help predict the existence of links. The information gained by analyzing network information at it changes the next time the link is called link prediction. Link prediction is an important element in social network analysis, it can be applied to many aspects of social network analysis, such as friend recommendations in social networks, prediction of po-tential links in biological protein networks, or the prediction the potential relationships in collaborative networks.

Link prediction generally involves one of two methods: structural methods and feature methods. Structural method-s involve the analysis and summarization of the network structure, including the analysis of nodes, neighbor node analysis, analysis of paths between nodes, link analysis and similarity analysis of relationships between adjacent links. For example, consider two people u and v in a social network. If u and v do not know each other, or have a lot of friends in common, it is likely that u and v will be introduced to each other. The feature method differs from the structural method.

In this case, two scholars who have, for example, published papers relating to link prediction and community clustering, will have a greater probability of cooperating. This study focuses on the analysis of network structure, because general node attribute information is not readily available, and the authenticity of the data obtained cannot be guaranteed.

In many social networks, people connect because of their common interests and hobbies, forming a group. In past stud-ies, researchers have rarely used the probability of propaga-tion between nodes in a network for link prediction. Using the propagation probability to calculate influence between nodes can more reliably reflect the relationships between nodes. Compared to previous algorithms, this measure can more accurately measure the similarity between nodes. In order to achieve link prediction, we first need to find a common group, and calculate the influence of the group on two unconnected nodes. The calculated results are taken as the similarity of two unconnected nodes. Since the precise calculation of the influence between nodes is time-consuming, we use an approximate model to quickly calculate the influence of a node set on a single node. In order to be able to quickly calculate the most possible future top-k links, we developed a set of algorithms to solve this problem.

In this paper, we propose an algorithm based on the propa-gation of influence for calculating similarity. The main idea is that the similarity of two unconnected nodes is the product of

VOLUME 4, 2016 1

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI

|  |  |
| --- | --- |
|  | 10.1109/ACCESS.2019.2942357, IEEE Access  Author *et al.*: Preparation of Papers for IEEE TRANSACTIONS and JOURNALS |

the influence of nodes in the common influence set of these two nodes. We proposes a new similarity index to compute the similarity of two nodes, and improve the efficiency of the similarity algorithm. The main contributions are as follows:

(1) We propose a similarity index based on influence propagation to calculate similarity. The similarity index is calculated by finding the common influence set of two un-connected nodes.

(2) In order to calculate the similarity of two unconnected node pairs efficiently we propose an algorithm to perform off-line indexing of each node in a graph, and we use this off-line index to calculate the upper bound of each two unconnected node pairs.

(3) In order to be able to calculate the top-k similar nodes efficiently, we propose a pruning algorithm which makes use of the upper bound of nodes similarity to improve efficiency.

The rest of the paper is organized as follows. We review some related work in Section 2. In Section 3, we give a statement of the problem. Section 4 gives the details of our algorithm. We present the important experimental results in Section 5 and conclude the paper in Section 6.

**II. RELATED WORK**

Links prediction involves two primary methods: namely, structural and feature-based. Most of the structural-based link prediction methods use network structure to measure node similarities. For example, in a social network, two individuals with many common friends are more likely con-nect in future [16]. Lada [1] proposed a method based on common neighbors to predict relationships between individ-uals. Tsuyoshi [15] proposed a link prediction method which constructed a directed action graph to estimate the similarity of the existence of a link between two nodes in weighted networks. Liu [14] proposed a similarity score based on a common neighbor method mentioned before and LBN(local naive Bayes) which performs better than common neighbors. Paths between nodes may also be used for link prediction, Katz [10] used the number of paths between two nodes and their length, producing reasonable results. L´l´z [11] proposed approach which had high effectiveness and efficiency, a local path index, to estimate the probability of the existence of a link between two nodes. Liu [13] proposed a method that use a local random walk to estimate the probability of the existence of a link between two nodes. Wang [24] proposed a method that uses a clustering-based collaborative filtering approach, including both topological and node attributes. Xu [25] proposed a method that use path entropy as similarity index to measure nodes’ similarity. Shang [22] first proposed a method for using past links to predict the future links. In [21], Shang found that if a pair of nodes are connected, they are more likely to connect to the common nodes in the future networks, and they first use the past links and future links for link prediction. In [20], Shang proposed the metric Precision for the evolving networks. Lee [27] proposed a topology-based similarity measure to predict future friends.

Yu [19] and Ben [23] proposed a machine learning ap-proach that uses nodes’ features to learn and predict missing links. Lars [2] and Jiang [9] combined the network structure features and edge attribute features, and use the Supervised Random Walk(SRW) algorithm for link prediction. L. Berton [3] proposed a supervised learning approach and a semi-supervised learning approach to learn nodes’ features and predict links in the future. Ozcan [17] proposed a novel multi-variate method for link prediction in evolving heterogeneous networks using a neural network, and then [18] proposed a novel method which using multivariate time series to make link prediction in evolving heterogeneous networks. Chen [26] proposed an optimization algorithm that used the AUC function as the optimization goal and translated the link prediction problem into an optimization problem. Bastami [28] proposed a machine learning method which integrated node features, community information, and graph properties. Feature extraction is essential in machine learning-based link prediction methods. In recent years, the rise of the Learning Graph Embedding method based on deep learning can improve the accuracy of link prediction. Pan proposed a deep learning-based multi-task Graph Embedding Learning method base on his previous works [29], [31]. Shang [32] proposed a location recommendation method based on his previous works [33]–[36]. He proposed a metric by corre-lating the trajectory and position in space for measuring the spatiotemporal correlation between trajectories and location-s. Gao [37] proposed a team recommendation algorithm for works with considering the spatial issue.

Link prediction can be applied not only to traditional social networks, but also predict the relationships of objects in videos by the development of object recognition techniques based on images and video [7] [5] [4] [6]. A web-based recommendation system can also predict the user’s connec-tion to an item in the future. In a complex network, users’dynamic interests and topics can be used to recommend products that will be of interest to the user in the future [8]. Similarly in event-based social networks (EBSNs), Li proposed an impact-based collaborative filtering algorithm for recommending events of interest to users [12].

**III. PROBLEM STATEMENT**   
In a social network *G*(*V* ,*E*,*W*), where *V* is the set of nodes, and *E* is the set of edges, *We*(*u*,*v*) represents the probability of propagation of node *u* to node *v*. Link prediction aims to identify unobserved or missing edges in the current network *G*. In this paper we propose a novel algorithm to identify unobserved or missing edges in a network, in descending order the similarity of each pair of such nodes, such that the higher the similarity of two nodes, the greater the possibility is of a link existing.

Because the complexity of this computation is high, the calculation takes a long time. We therefore use a proximate model to calculate the similarity between nodes. In this paper, we use the MIA model to calculate influence. The MIA model uses maximum influence path propagation probability

2 VOLUME 4, 2016

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI

|  |  |
| --- | --- |
| 10.1109/ACCESS.2019.2942357, IEEE Access Author *et al.*: Preparation of Papers for IEEE TRANSACTIONS and JOURNALS |  |

as the influence value between two nodes.

**Definition 1**(Influence Set).In a weighted graph *G*(*V* ,*E*,*W*),the influence set of node u can be denoted as

*Infset*(*u*) = *{v|inf*(*v, u*) *> θ}*  (1)

In Equation 1, *inf*(*u*,*v*) denotes the influence value from node *u* to *v* in network *G*, and *θ* is a threshold value to determine the size of node u’s influence set. We set a thresh-old is to reduce the amount of unnecessary computation, In a high connectivity network, a node can be influenced by many nodes, but most of them contribute little to the final result. So we set a threshold to reduce nodes which have tiny contributions, in order to make a trade-off between time and accuracy.

**Definition 2**(Common Influence Set).In a weighted graph G(V,E,W),the common influence set of node u and node v is a set of nodes which can influence both node u and node v. It can be denoted as

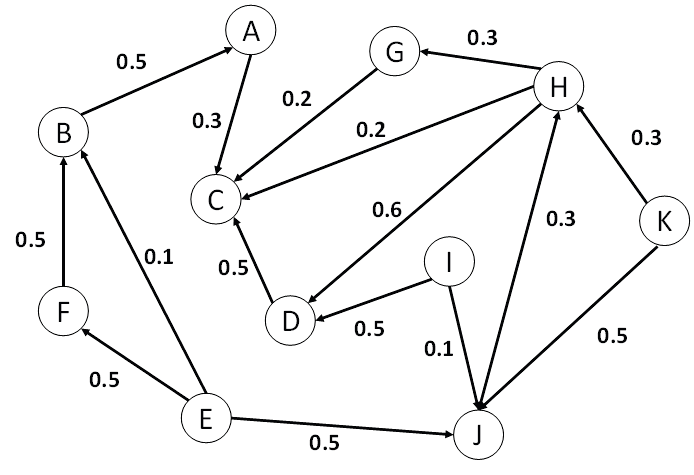
*CIS*(*u, v*) = *{w|inf*(*w, v*) *> θ* & *inf*(*w, u*) *> θ}*  (2) = *Infset*(*u*) *∩Infset*(*v*) In Equation 2, the Common Influence Set is a set of nodes which from both node *u*’s influence set and node *v*’s influence set. In the link prediction process, we need to calculate the similarities of each unconnected pair of nodes, and therefore need to calculate the common influence set. In the process of calculating the common influence set, each pair of unconnected node needs to be calculated. In order to efficiently compute the common influence set of each two unconnected node pairs, we calculate the influence set of each node at the beginning, and store the influence set of each node. In the process of calculating the common influence set, we use the influence set of each node.

*simscore*(*u, v*) = *inf*(*CIS*(*u, v*)*, u*) *× inf*(*CIS*(*u, v*)*, v*) (3) In Equation 3, *inf*(*S*,*u*) denotes the influence value from seed set *S* to node *u*. *CIS*(*u*,*v*) denotes the Common Influ-ence Set of node *u* and node *v*. The formula consists of two parts: the influence value from *CIS*(*u*,*v*) to node *u* and the influence value from *CIS*(*u*,*v*) to node *v*. We use the product of each part as the similarity score.

**IV. THE CIS MODEL**

***A. FIND INFLUENCE SET OF EACH NODE***

The pseudocode for generating the influence set is shown in Algorithm 1. We used the Dijkstra algorithm to find the shortest path in the graph *G*(*V* ,*E*,*W*). In Algorithm 1, we use this approach to find the maximum influence path. In order to archive that we use -log *W*(*u,v*) as edge *e*(*u, v*) ’s weight, be-cause the influence propagation probability of path *P* is cal-culated as*P* When we add a new node *v* to set *S*, then we calculate *eW*(*e*), and*P e−*log *W*(*e*)=*−*log*P eW*(*e*).



**FIGURE 1.** A social network with 11 users.

*inf*(*v,u*), if *inf*(*v,u*) *< θ* then we stop searching *v*’s neigh-bors. In Algorithm 1, in order to find one node’s influence set, we use the Dijkstra algorithm to find the maximum influence path. In line1-line3, we initialize the distance from each node to u +*∞*. In line6-line14, we use -log *We* as edge *e*’s new weight, then set a startup node *u* as seed node, and use the Dijkstra algorithm to find the shortest path in graph *G*. In the procedure of find the shortest path, when *dist*(*u, k*)+*W*(*k, v*) *< dist*(*u, v*), we update the distance index. When the algo-rithm reaches a node v which *d*(*u, v*) *> eθ*, then it stops searching the node’s neighbors. When there are no nodes that can be added to *S* the algorithm stops. In line 15-line17, we convert the distance value to influence propagation probability.

**Algorithm 1** finding influence set algorithm

|  |  |  |
| --- | --- | --- |
| **Require:** | | Graph *G*(*V, W, E*), the node *u* |
| **Ensure:** | | the influence set of node *u* |
| 1: **for** each *v* in *V* **do**  2: 3: **end for** *dist*(*v*) = +*∞*  4: *dist*(*u*) = 0  5: *S* = {*r*}  6: **while** *V* -*S* is non-empty **do** | | |
| 7: 8: 9: 10: | *k* = arg min*v∈V −S dist*(*v*)  **if** *dist*(*k*)*−*log *W*(*k, v*)*<dist*(*v*) **then**  **end if** *dist*(*v*)=*dist*(*k*)*−*log *W*(*k, v*) | |
| 11: | **if** *e−dist*(*v*)*<θ* **then** | |
| 12: | CONTINUE | |
| 13: | **end if** | |
| 14: **end while**  15: **for** each *v* in *V* **do** | | |
| 16: | *dist*(*v*) = *e−dist*(*v*) | |
| 17: **end for**  18: **return** *S* | | |
| Example 1. In Figure 1, we have a social network with 11 users *{a, b, c, d, e, f, g, h, i, j, k}* and the edge value is the influence probability from one node to its neighbors. The threshold *θ* = 0*.*1. This example shows the calculation | | |

VOLUME 4, 2016 3

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI

|  |  |
| --- | --- |
|  | 10.1109/ACCESS.2019.2942357, IEEE Access  Author *et al.*: Preparation of Papers for IEEE TRANSACTIONS and JOURNALS |

steps used to find the influence set of node *c*. The calculation steps are shown in Table 2. In step1, the algorithm puts node *c* into set *S*. In step2, the algorithm finds node *c*’s all in neighbors and find a max influence path to node *c*, due path *< a, c >* is the max influence path, so node *a* should be added to *S*. And so on, for step3,step4,step5,step6. In step7, due to *P*(*k*,*h*,*d*,*c*)*<θ* and *P*(*j*,*h*,*d*,*c*)*<θ*, the algorithm stops searching the neighbor nodes of node *k* and node *j*. step9 is similar to step7, due to *P*(*f*,*b*,*a*,*e*)*<θ* and *P*(*e*,*b*,*a*,*c*)*<θ*, and the algorithm stops searching the neighbor nodes of node *f* and node *e*. When no node can be added to set *S*, the algorithm finishes.

***B. CALCULATION OF THE COMMON INFLUENCE SET*** According to Equation 2, in order to calculate two uncon-nected node pairs’ similarity score, the Common Influence Set of these two nodes should be calculated first. Based on Algorithm 1, we can build influence sets for each node in graph *G*.

Based on Algorithm 1 introduced above, we can build influence sets for each node in graph *G*. Based on Equation 2, to calculate the similarity of two nodes, we first have to calculate the Common Influence Set. Using Algorithm 1, we can get all influence nodes *infsetu* of node *u*, thereby getting the Common Influence Set of two nodes in graph *G*. A naive method of predicting the top-k links in graph *G*(*V* ,*E*,*W*) is shown in Algorithm 2. Using Equation 2, the process of Algorithm 2 is straightforward. In Algorithm 2, line 1 initializes an empty set *S* to store the top-k predicted links. Line2-line7 calculate the similarity score of each pair of unconnected nodes. Line3-line5 is the implementation of Equation 2, and the result is stored in set *S*. In order to get the top-k links, we need to sort set *S* in descending order and return the top-k results(line8-line9).

**Algorithm 2** Naive method for link prediction using CIS similarity index

|  |
| --- |
| **Require:** Graph *G*(*V, W, E*),*infset* , k **Ensure:** top-k missing edges scores *S* 1: S = *∅*2: **for** each edge *e*(*u*,*v*) in *E* **do**  3: *ISu* = *infset*(*u*)  4: *ISv* = *infset*(*v*)  5:  6: *c* = *ISu ∩ISv S*(*u,v*) = *inf*(*c*,*u*)\**inf*(*c*,*v*) 7: **end for**  8: Sort *S* in descending order 9: **return** *S.top*(*k*) |
| For example, for the graph shown in Figure 1, the influence  set of node *C* is *infset*(*C*)=*{A, B, K, D, E}*, the influence set of node I is *infset*(*I*)=*{K, E, F, D}*, so the Common Influence Set of node *C* and node *I* is *CIS*(*C*,*I*)=*{K, D, E}*. Then we calculate the influence of *CIS*(*C*,*I*) to node *C* and  node *I*, and we get the similarity score of node *C* and node *I*,  *simscore*(*C*,*I*) = *inf*(*CIS*(*C*,*I*),*C*) *× inf*(*CIS*(*C*,*I*),*I*). |

For the details of how to calculate common influence value of two unconnected nodes see Algorithm 2. There are many influence paths in the influence tree of a node. when we calculate the similarity score of two unconnected nodes *u* and *v*, we should calculate the influence set of each node. Let *infset*(*u*) and *infset*(*v*) denote the influence set of node *u* and node *v*, and let *cinfset*(*u*,*v*) denote the Common Influence Set of node *u* and node *v*. Then find the common influence nodes of these two influence set, *cinfset*(*u*,*v*)= *infset*(*u*) *∪infset*(*v*), then calculate maximum influ-ence path from these nodes *infpath*(*cinfset*(*u*,*v*),*u*) *∪infpath*(*cinfset*(*u*,*v*),*v*). When the influence paths are in-dependent from each other then the similarity score can be calculate using Equation 2, otherwise we need to consider the situations below.

When a new seed is on the maximum influence path from node *v* to another node *u*, since the seed blocks *v* on the maxi-mum influence path. In this situation the influence from *v* to *u* should be ignored. When we do link prediction, the Common Influence Set would be calculated many times. However, it is quite expensive to compute it frequently. However, we find an upper bound of Equation 3.

Let *infubu* denote the upper bound of *inf*(*CIS*(*u*,*v*),*u*), *infub*(*v*) denote the upper bound of *inf*(*CIS*(*u*,*v*),*v*), then the upper bound of *simscore*(*u*,*v*) is *infub*(*u*) *× infub*(*v*). In this paper we use *simub*(*u*,*v*) denote the upper bound of *simscore*(*u*,*v*). Algorithm 3 show the detail of calculating the influence upper bounds of each node in graph. In next section, we discuss how to use this upper bound to reduce the calculation time. The upper bound formula of *simscore*(*u*,*v*) is shown in Equation 4.

*simub*(*u, v*) = *infub*(*u*) *× infub*(*v*) (4)

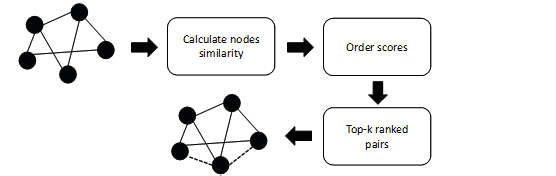
**Algorithm 3** The influence upper bounds of each node in graph

|  |  |
| --- | --- |
| **Require:** Graph *G*(*V, W, E*)  **Ensure:** *UpperBound*  1: **for** each Node *u* in *V* **do** | |
| 2:  3:  4:  5: | let *infsetu* = the influence set of u *UpperBoundu* = 1  **for** each *v* in *infsetu* **do**   **if** (*v, u*) in *E* **then** |
| 6:  7: | **end if** *UpperBoundu* = *UpperBoundu ×* (1- *Wv,u*) |
| 8: | **end for** |
| 9: *UpperBoundu* = 1 - *UpperBoundu*  10: **end for**  11: **return** *UpperBound* | |

In Algorithm 3, for each vertex u in a social network *G*(*V, W, E*), we first get it’s influence set. In line3, we first initialize the upper bound of *u* to 1. Then for all the neighbor nodes *v* which contained *u*’s in influence set, calculate the influence from *v* to node *u*, and update it’s upper bound from

4 VOLUME 4, 2016

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI

|  |  |
| --- | --- |
| 10.1109/ACCESS.2019.2942357, IEEE Access Author *et al.*: Preparation of Papers for IEEE TRANSACTIONS and JOURNALS |  |

**TABLE 1.** Calculation process of example 1.

|  |  |  |
| --- | --- | --- |
| Step | S | Max influence path value |
| 1 | c | a,c=0.3 |
| 2 | c,a |
| 3 | c,a,g | a,c=0.3 g,c=0.2 |
| 4 | c,a,g,h | a,c=0.3 g,c=0.2 h,c=0.2 |
| 5 | c,a,g,h,d | a,c=0.3 g,c=0.2 d,c=0.5 h,d,c=0.3 |
| 6 | c,a,g,h,d,i | a,c=0.3 g,c=0.2 d,c=0.5 h,d,c=0.3 i,d,c=0.25 |
| 7 | c,a,g,h,d,i,k,j | a,c=0.3 g,c=0.2 d,c=0.5 h,d,c=0.3 i,d,c=0.25 k,h,d,c=0.09 j,h,d,c=0.09 |
| 8 | c,a,g,h,d,I,b,k,j | a,c=0.3 g,c=0.2 d,c=0.5 h,d,c=0.3 i,d,c=0.25 b,a,c=0.15 |
| 9 | c,a,g,h,d,I,b,f,e,k,j | a,c=0.3 g,c=0.2 d,c=0.5 h,d,c=0.3 i,d,c=0.25 b,a,c=0.15 f,b,a,e=0.075 e,b,a,c=0.015 |

**Algorithm 4** The advanced link prediction algorithm using   
CIS similarity index

**FIGURE 2.** The process of link prediction.

line4-line7. After we calculated all vertex in *G*, we store the result as an index for pruning unnecessary computations. For example, in figure 3, after we get all nodes’ influence set, the calculation process of the influence upper bound of vertex *a* are as follow: (1). *infseta* = (b, d, c). (2) *W*(*b, a*)=0.4, *W*(*d, a*)=0.3. (3) *infub*(*a*) = 1*−W*(*b, a*) *× W*(*d, a*) = 0.76. The other nodes’ upper bound calculation are shown in Table 2.

***C. CALCULATION OF TOP-K LINKS***

To solve the link prediction problem, it needs to calculate the similarity scores of all pairs of unconnected nodes. We use a generic framework to illustrate the link prediction solution, as shown in Figure 2. Since in a social network, there are many unconnected node pairs. When we use Algorithm 2 to calculate each two unconnected node’s similarity scores, the time efficiency is very hight. In order to calculate the top k similarity score efficiently, we proposed a new algorithm. The main idea is: (1) Calculate two unconnected nodes’similarity score cost a lot of time, but it is more suitable to calculate the similarity score upper bound of unconnected two nodes. (2) since we need only to calculate top-k simi-larity scores, the upper bound can be used to prune necessary calculations.

|  |  |
| --- | --- |
| **Require:** Graph *G*(*V, W, E*), *UpperBound*,*k*  **Ensure:** top k future links L  1: L = *∅*2: Init priority queue *Q* sort by simscore  3: **for** edge *e*(*u, v*) */∈E* **do** 4: *simscore*(*u,v*)=*UpperBoundu*\**UpperBoundv*; | |
| 5: | *simscoretype*(*u,v*)="UpperBound" |
| 6: | *Q*.*push*(*e*(*u*,*v*)) |
| 7: **end for**  8: **while** *Q* is not empty and *size*(*L*) <*k* **do** | |
| 9: 10: | *simscore*(*u,v*) = *Q*.*pop*()  **if** *simscoretype*(*u,v*)="UpperBound" **then** |
| 11: 12: | *simscore*(*u,v*) = *cis*(*u*,*v*)  *simscoretype*(*u,v*)="Value" |
| 13: | *Q*.*push*(*e*(*u*,*v*)) |
| 14: | **else** |
| 15: | *L*.*push*(*e*(*u*,*v*)) |
| 16: | **end if** |
| 17: **end while**  18: **return** *L* | |

For example: in a social graph *G*(*V* ,*E*,*W*) Figure 3, *V* denote the node set of graph *G*, *E* denote the edge set of graph *G*, *W* denote the information propagation probably set. In this example, we set *θ* = 0.21. First we need to calculate the influence set of each node in graph *G*, shown in figure 3b. For node *a* in graph *G*, according to Algorith-m 1, we can calculate the maximum influence path to n-ode *a*, (*e, b, a*),(*c, d, a*),(*b, a*),(*d, a*). Since the influence path *p*(*e*,*b*,*a*) *< θ*, node e does not belong to node *a*’s influence set. Similar we can find the influence set of nodes *b*,*c*,*d* and *e*, as shown in Figure 3.

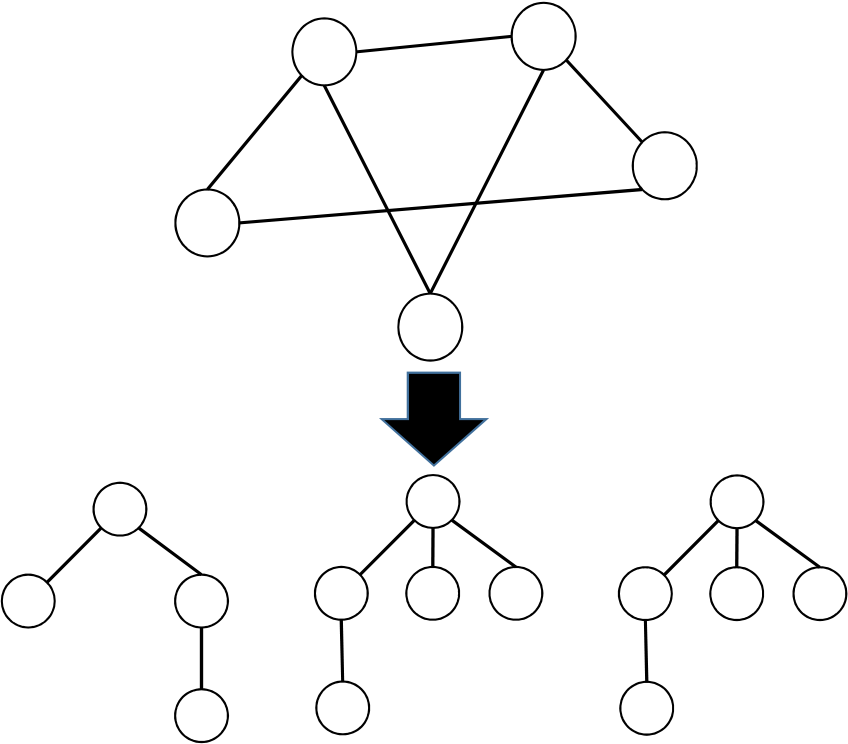
After we calculate the influence set for each node, we can use the upper bounds of each node as calculated using Algorithm 3, and use it to calculate the influence upper bound of each node. The upper bounds can be stored to an index, and this index can quickly return the upper bound of the similarity score of two unconnected nodes, as in Algorithm 3. The e-quation for calculating the upper bounds of two unconnected nodes is presented in Equation 4. for calculations using the sample input, the calculation procedures see Table 2.

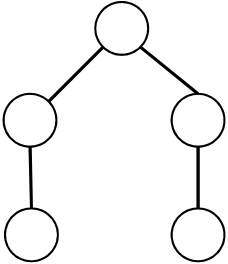
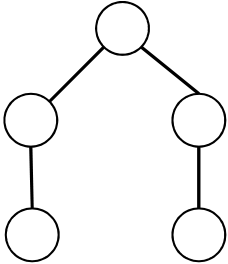
VOLUME 4, 2016 5

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI

|  |  |
| --- | --- |
|  | 10.1109/ACCESS.2019.2942357, IEEE Access  Author *et al.*: Preparation of Papers for IEEE TRANSACTIONS and JOURNALS |

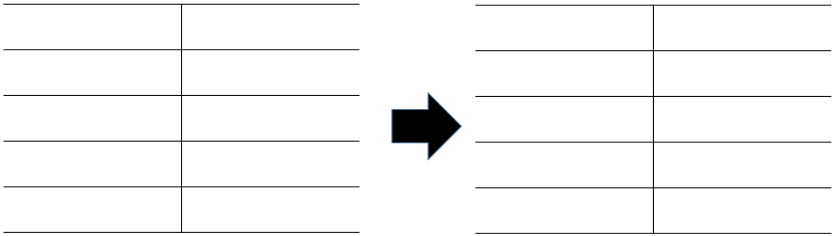


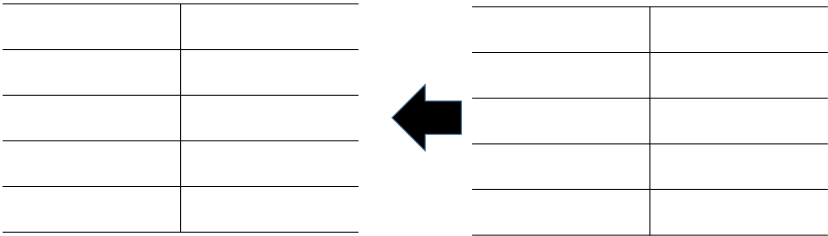
**FIGURE 3.** An example of finding nodes’ influence set.

**TABLE 2.** An example of calculating the upper bound of each node in Figure 3.

|  |  |
| --- | --- |
| node | influence upperbound |
| a  b  c  d  e | 1-(1-0.4)*×*(1-0.6)=0.76  1-(1-0.4)*×*(1-0.5)*×*(1-0.3)=0.79 1-(1-0.5)*×*(1-0.5)*×*(1-0.3)=0.825 1-(1-0.5)*×*(1-0.6)=0.8  1-(1-0.5)*×*(1-0.5)=0.75 |









|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**FIGURE 4.** The process of Algorithm 4.

After retrieving the upper bound of influence of the in-fluence set of each node, we use a pruning algorithm to find the top-k similarity scores of two nodes that are not connected by an edge. According to the description in Al-gorithm 4, we firstly calculate the upper bound of node pairs (*a*,*e*),(*e*,*d*),(*b*,*d*),(*a*,*c*). Based on the Equation 3, we can get *simub*(*a*,*e*)=0.57, *simub*(*e*,*d*)=0.6, *simub*(*b*,*d*)=0.632, *simub*(*a*,*c*)=0.627. As shown in Figure 4a, we save these value to a priority queue, then retrieve the first element from the priority queue. If the value of the element is equal to the upper bound, we calculate the exact value of the element and put the exact value into the priority queue. The process is shown in Figure 4.

**V. EXPERIMENTS**   
In this section, we compare our link prediction method with other methods. We provide details of the datasets used in the experiments. To evaluate the performance of our proposed approach, we compare our algorithm with some mainstream link prediction algorithms which are based on similarity. Experimental results are reported as the area under the ROC curve (AUC).

***A. EVALUATION***   
For link prediction, AUC is an important indicator of the performance of an algorithm. Considering the problem of link prediction, we use *G*0(*V, E*0) to indicate the current state of the network, and *Gt*(*V, Et*) to indicate the state of the network in the future. The AUC is calculated as follows:

similarity score as *s*1, where *Et* represents the edge set in First, find an edge from the set *Et −E*0 and calculate its

*Gt*, *E*0 represents the edge set in *G*0, and *Et −E*0 represents the increased edges of the graph *G* evolving from *G*0 to *Gt*. Second, randomly pick a nonexistence link in *Gt* and calculate its similarity score *s*2. Finally, the AUC calculation method is as reported in [14], and the equation is:

*AUC* =*n′* + 0*.*5*n′′*(5) *n*

In equation 5, *n′*is the number of times that the links in edge set *Et −E*0 got a higher score (*s*1 *> s*2) than unconnected links, while *n′′*is the number of times that they are equal (*s*1 = *s*2), *n* is the number of times that we randomly pick a pair of links from edge set *Et −E*0 and the set of unconnected links, where *n* = *n′*+ *n′′*.

We introduce three mainstream similarity based link pre-diction methods for comparing with our method in our exper-iments.

***B. DATASET***   
Our link prediction method and baseline methods were tested on three real world datasets. These are the High-Energy-Physics literature(HEP-Th) dataset NetScience, Emails net-work and Wikipedia elections.

The HEP-TH [38] (High Energy Physics Theory) citation network is constructed from e-print arXiv and covers all

6 VOLUME 4, 2016

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI

|  |  |
| --- | --- |
| 10.1109/ACCESS.2019.2942357, IEEE Access Author *et al.*: Preparation of Papers for IEEE TRANSACTIONS and JOURNALS |  |

**TABLE 3.** The topological features of datasets.

**TABLE 4.** Comparison of algorithms’ accuracy quantified by AUC.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dataset | N | M | NUMc | K | Dataset | Common Neighbor | Katz | CIS |
| Hep-Th | 8392 | 20387 | 461 | 4.85 | Hepnet | 0.9356 | 0.8716 | **0.9484**  **0.9299**  0.92554 |
| Wiki-Vote | 6260 | 78778 | 34 | 25.17 | Wiki-Vote | 0.9020 | 0.8422 |
| Email | 1133 | 5451 | 1 | 9.62 | Netsience | 0.9621 | **0.9845** 0.7585 |
| NetScience | 1589 | 2742 | 396 | 3.45 | Email | 0.8679 | **0.8980** |

citations in the dataset of 352,807 edges in 27,770 papers. If the paper references paper j, the graphic contains the directed edges from i to j. If the paper references or references a paper outside the data set, the chart does not contain any information about this content. The data covers documents from January 1993 to April 2003 (124 months). It started in the first few months of arXiv, and therefore basically represents the complete history of its HEP-TH part.

The NetScience [40] includes a collaborative network of scientists working on network theory and experimental re-search, written by Mark Newman on May 2006. This network contains all the components of the network, with a total of 1,589 scientists and 2,742 connections. The network is based on two reference books on web-based review articles, M. E. J. Newman, SIAM Review 45, 167-256 (2003) and S. Boccaletti et al. The network is weighted, and the weights are directly assigned according to the number of collaborations between authors, and vice versa according to the number of other authors involved. This weighting is described in M.E.J. Newman, Phys. Rev. E 64,016132 (2001).

The Email network contains 1133 nodes and 5541 edges, if user i send an email to user j, the network contains a directed edge from i to j.

The Wiki-Vote [39] network contains all Wikipedia voting data from Wikipedia until January 2008. The nodes in the network represent Wikipedia users, and the directed edges from node i to node j represent the users I vote on user j. This network set contains 2,794 elections with a total of 103,663 votes and 7,066 (voting or voting). In these 1,235 elections, the promotion was successful, and 1,559 elections did not lead to promotion. Wikipedia is a free wiki database written by volunteers from around the world. A small part of Wikipedia contributors are administrators who are users who have access to other technical features that are helpful for maintenance. In order to make the user an administrator, a management request (RFA) is issued, and the Wikipedi-a community decides to upgrade management by public discussion or voting. Extract all administrator election and voting history data using the latest full dump of Wikipedia page editing history (since January 3, 2008). About half of the votes in the data set come from existing administrators, while the other half comes from regular Wikipedia users. The topological features of these datasets are shown in Table 3. In Table 3, N denotes the total number of nodes, M denotes the total number of edges, NUMc denote the number of the connected components in the network. K denotes the average degree of the network.

***C. RESULTS***

In the experiments, we divided each dataset into two parts, a training set and a test set. Past connections were stored in the training set, and the link prediction methods used the training set to predict future links in the test set. For example, for the *HEP −Th* dataset, it contains connections in the years from 1991 to 2003. We use connections in the years from 1991 to 2000 as the training set, and the connections in the years from 2001 to 2003 as the test set. We used connections in training set to predict connections in the future. For the netscience and email dataset, we randomly removed 20 percent of edges as a test set, since there were no timestamps in the dataset. For the wiki-vote dataset, we used connections which were timestamped between from 1080494700 to 1190000000 as the training set, and the remaining connections as the test set. The *AUC* results are shown in Table 4. In the first column are the datasets considered, in the first row are the methods for link prediction and in the following rows the AUC score of each methods test on each datasets. The result show that CIS performs better than other methods in all datasets except Netscience.

In order to see how the value of *θ* affect the AUC test result, we set different values of *θ* and run AUC tests on all datasets. The results are shown in Figure 5.

Figure 5 shows, as the value of *θ* decreases the AUC value increases. Small values of *θ* can achieve better AUC result s because small values of theta produce bigger influence trees, meaning the influence value of the common influence set to each two unconnected nodes would be more accurate. As the value of *θ* decreases, the time cost of influence calculation increases, and the total running time of the algorithm will increase also.

In this paper, we proposed a new similarity index and provide an effective method to calculate the top-k links for the link prediction problem. In order to compare the running time between the naive method and the advanced method, we randomly generate different sized of networks with average degree equals to 3, and use our naive method and advanced method run the top 100 link prediction. The results are shown in Figure 6. We can observe from Figure 6 that the advanced method ran faster than the naive method on different size of networks, and the advanced method can be applied to large scale networks.

**VI. CONCLUSIONS**   
In this work, we propose a new similarity index for link prediction. Experiments showed that our similarity index

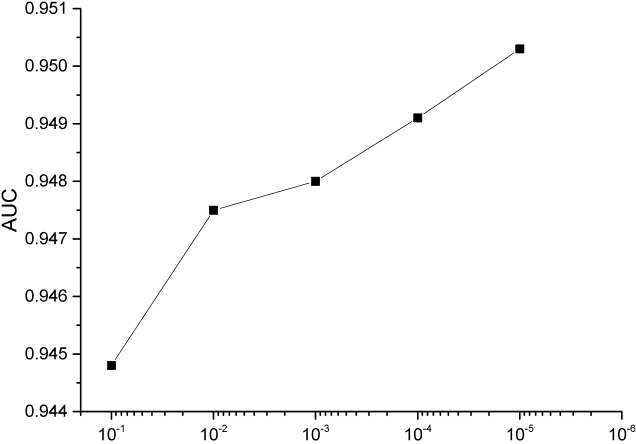
VOLUME 4, 2016 7

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI

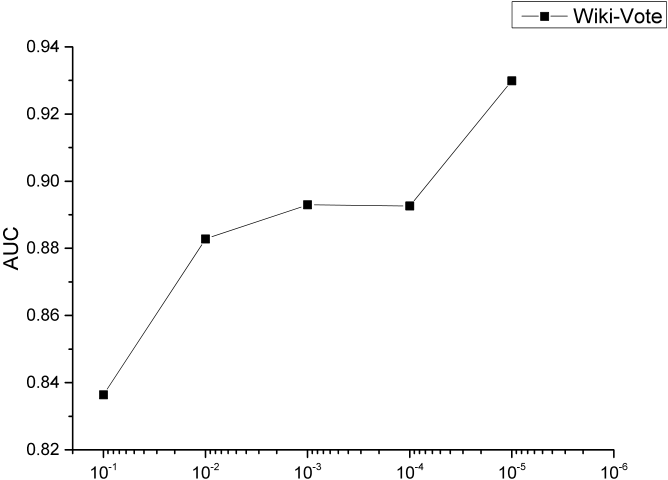
|  |  |
| --- | --- |
|  | 10.1109/ACCESS.2019.2942357, IEEE Access  Author *et al.*: Preparation of Papers for IEEE TRANSACTIONS and JOURNALS |







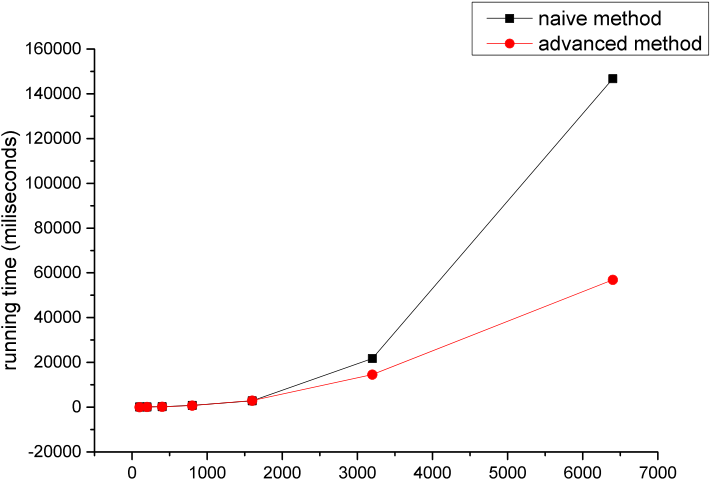
(a) AUC result on Hep-TH





(b) AUC result on Wiki-Vote

**FIGURE 5.** Different values of *θ* affect the result of AUC





**FIGURE 6.** Comparison of naive and advanced method’s time cost on different graph size.

performs better than other main stream similarity indices. Due to the time cost of the similarity score calculation of CIS, we proposed an advanced method to calculate the similarity score efficiently. For future work, we will solve the problem in the dynamic graphs whose structures will change along over time.

**REFERENCES**

[1] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. Social Networks, 25(3):211–230, 2003.

[2] Lars Backstrom and Jure Leskovec. Supervised random walks: predicting and recommending links in social networks. In ACM International Conference on Web Search & Data Mining, pages 635–644, 2010.

[3] L. Berton, J. Valverde-Rebaza, and A. De, Andrade Lopes. Link prediction in graph construction for supervised and semi-supervised learning. In The International Joint Conference on Neural Networks, 2015.

[4] Du Bo and Liangpei Zhang. A discriminative metric learning based anomaly detection method. IEEE Transactions on Geoscience & Remote Sensing, 52(11):6844–6857, 2014.

[5] Du Bo and Liangpei Zhang. Target detection based on a dynamic subspace. Pattern Recognition, 47(1):344–358, 2014.

[6] Bo Du, Shihan Cai, Chen Wu, Liangpei Zhang, and Dacheng Tao. Object tracking in satellite videos based on a multi-frame optical flow tracker. 2018.

[7] Bo Du, Yuxiang Zhang, Liangpei Zhang, and Dacheng Tao. Beyond the sparsity-based target detector: A hybrid sparsity and statistics-based de-tector for hyperspectral images. IEEE Transactions on Image Processing, 25(11):5345–5357, 2016.

[8] Li Gao, Chuan Zhou, Jia Wu, and Yue Hu. Collaborative dynamic sparse topic regression with user profile evolution for item recommendation. In The Thirty-First Conference on Artificial Intelligence (AAAI-17), 2017. [9] Maosheng Jiang, Yonxiang Chen, and Ling Chen. Link prediction in networks with nodes attributes by similarity propagation. Computer Science, 2015.

[10] Leo Katz. A new status index derived from sociometric analysis. Psy-chometrika, 18(1):39–43, 1953.

[11] L´l´z L, C. H. Jin, and T. Zhou. Similarity index based on local paths for link prediction of complex networks. Physical Review E Statistical Nonlinear & Soft Matter Physics, 80(2):593–598, 2009.

[12] Gao Li, Wu Jia, Qiao Zhi, Chuan Zhou, Yang Hong, and Hu Yue.

Collaborative social group influence for event recommendation. In Acm International, 2016.

[13] Weiping Liu and Linyuan Lu. Link prediction based on local random walk. Epl, 89(5):58007–58012(6), 2010.

[14] Zhen Liu, Qian Ming Zhang, Linyuan L´l´z, and Tao Zhou. Link prediction in complex networks: a local naive bayes model. Epl, 96(4):48007, 2011. [15] Tsuyoshi Murata and Sakiko Moriyasu. Link prediction of social net-works based on weighted proximity measures. In Web Intelligence, IEEE/WIC/ACM International Conference on, pages 85–88, 2007.

[16] M. E. Newman. Clustering and preferential attachment in growing networks. Physical Review E Statistical Nonlinear & Soft Matter Physics, 64(2):–, 2001.

[17] Alper Ozcan and Sule Gunduz Oguducu. Link prediction in evolving heterogeneous networks using the narx neural networks. Knowledge & Information Systems, 55(3):1–28, 2017.

[18] Alper Ozcan and Sule Gunduz Oguducu. Multivariate time series link prediction for evolving heterogeneous network. International Journal of Information Technology & Decision Making, 18(2), 2019.

[19] Yu Kai, Chu Wei, Shipeng Yu, Volker Tresp, and Xu Zhao. Stochastic relational models for discriminative link prediction. In International Conference on Neural Information Processing Systems, 2006.

[20] Ke Ke Shang, Michael Small, Xiao Ke Xu, and Wei Sheng Yan. The role of direct links for link prediction in evolving networks. EPL (Europhysics Letters), 117(2):28002, 2017.

[21] Ke Ke Shang, Wei Sheng Yan, and Michael Small. Evolving networks ˛ałus-ing past structure to predict the future. Physica A Statistical Mechanics & Its Applications, 455:120–135, 2016.

[22] Ke Ke Shang, Wei Sheng Yan, and Xiao Ke Xu. Limitation of degree in-formation for analyzing the interaction evolution in online social networks. International Journal of Modern Physics C, 25(10):1450056–, 2014.

[23] Ben Taskar, Ming Fai Wong, Pieter Abbeel, and Daphne Koller. Link prediction in relational data. In Neural Information Processing Systems, pages 659–666, 2004.

[24] Xiangyu Wang, Dayu He, Danyang Chen, and Jinhui Xu. Clustering-based collaborative filtering for link prediction, 2015.

[25] Zhongqi Xu, Cunlai Pu, and Jian Yang. Link prediction based on path entropy. Physica A Statistical Mechanics & Its Applications, 456:294–301, 2016.

[26] Bo Lun Chen, Yong Hua, Yan Yuan, and Ying Jin. Link prediction on directed networks based on auc optimization. IEEE Access, PP(99):1–1, 2018.

[27] Joo Young Lee and Rustam Tukhvatov. Evaluations of similarity measures on vk for link prediction. Data Science and Engineering, 3(3):277–289, 2018.

[28] Esmaeil Bastami, Aminollah Mahabadi, and Elias Taghizadeh. A gravitation-based link prediction approach in social networks. Swarm & Evolutionary Computation, page S2210650217304704, 2018.

8 VOLUME 4, 2016

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI

|  |  |  |
| --- | --- | --- |
| 10.1109/ACCESS.2019.2942357, IEEE Access Author *et al.*: Preparation of Papers for IEEE TRANSACTIONS and JOURNALS | |  |
| [29] S. Pan, J. Wu, X. Zhu, G. Long, and C. Zhang. Task sensitive feature explo-ration and learning for multitask graph classification. IEEE Transactions on Cybernetics, 47(3):1–15, 2017.  [30] Shirui Pan, Ruiqi Hu, Sai Fu Fung, Guodong Long, and Chengqi Zhang. Learning graph embedding with adversarial training methods. 2019.  [31] Shirui Pan, Wu Jia, Xingquan Zhu, Chengqi Zhang, and Philip S. Yu.  Joint structure feature exploration and regularization for multi-task graph | | 9 |
| classification. | IEEE Transactions on Knowledge & Data Engineering, |
| 28(3):715–728, 2016.  [32] Shuo Shang, Lisi Chen, Zheng Kai, Christian S. Jensen, and Panos Kalnis.  Parallel trajectory-to-location join. IEEE Transactions on Knowledge & Data Engineering, PP(99):1–1, 2018.  [33] Shuo Shang, Lisi Chen, Zhewei Wei, Christian S. Jensen, Zheng Kai, and Panos Kalnis. Parallel trajectory similarity joins in spatial networks. Vldb Journal, 27(3):395–420, 2018.  [34] Shuo Shang, Lisi Chen, Zhewei Wei, Christian S. Jensen, Ji Rong Wen, and Panos Kalnis. Collective travel planning in spatial networks. In IEEE International Conference on Data Engineering, 2017.  [35] Shuo Shang, Ruogu Ding, Zheng Kai, Christian S. Jensen, Panos Kalnis, and Xiaofang Zhou. Personalized trajectory matching in spatial networks.  Vldb Journal, 23(3):449–468, 2014.  [36] Shuo Shang, Lisi Chen, Christian S. Jensen, Ji Rong Wen, and Panos Kalnis. Searching trajectories by regions of interest. IEEE Transactions on Knowledge & Data Engineering, 29(7):1549–1562, 2017.  [37] Dawei Gao, Yongxin Tong, Jieying She, Tianshu Song, Lei Chen, and Ke Xu. Top-k team recommendation and its variants in spatial crowd-sourcing. Data Science and Engineering, 2(2):136–150, Jun 2017.  [38] https://kdl.cs.umass.edu/display/public/HEP-Th  [39] http://snap.stanford.edu/data/wiki-Vote.html  [40] http://casos.cs.cmu.edu/computational\_tools/datasets/external/netscience/index11.php  VOLUME 4, 2016 | |

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.