ISSN 2278-3091
Volume 9, No.4, July – August 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse254942020.pdf
https://doi.org/10.30534/ijatcse/2020/254942020

Distributed Computation Approach for Link Prediction in Graph Stream using DCS Features and Gradient Boosted Trees

Tu-Anh Nguyen-Hoang[1], Khanh-Duy Le-Trinh[2], Anh-Thu Nguyen-Thi[3]
[1] University of Information Technology, Vietnam National University HCMC, Vietnam, anhnht@uit.edu.vn
[2] University of Information Technology, Vietnam National University HCMC, Vietnam, 15520159@gm.uit.edu.vn
[3] University of Information Technology, Vietnam National University HCMC, Vietnam, thunta@uit.edu.vn

ABSTRACT
Nowadays, link prediction in the networks is one of the fields of greatest attraction in data mining. In link prediction, graph stream has become an essential model to represent interactive elements in the massive networks. It is a promising key to performing many real-world applications and can be applied in many fields, such as social networks, Ecommerce networks, and telecommunication networks. However, the most exciting link prediction methods just center on studying predicted the existence of links in snapshot graphs, while most recent applications in the form of the graph streams. When applying these methods to graph streams, we have two challenges: the large amount of links and the rapid evolvement of graph streams. This paper introduces an efficient method to predict real-time existence link in graph streams. We propose an efficient Graph Stream Distributed Computation framework (GSDC) which is directly amendable to parallelization, facilitating a scalable distributed execution on the Apache Spark platform. With our framework, we present the Distributed Computation Score feature (DCS) and Gradient Boosted Trees (GBT) which are designed to distribute computation approach. Our experimental results on three realistic social networks prove our method can reduce the feature extraction time by 3x times and the errors always below 9%. Additionally, our method can increase the prediction precision by 21% over the baseline methods.

Key words: Link prediction, distributed method, data mining, social network, graph stream.

1. INTRODUCTION

The network is a method of expressing connections and interactions of objects in a collection. In the network, objects (nodes) are connected by pairwise interactions (links). In recent years, the Internet, social networks, transport networks, and biological networks... have been built as network systems. The links are expressed in many different ways, such as friendships, collaborations, and marriages as in the social network and airlines and railways as the transportation network. This simple idea can be used to describe most applications whose systems can range from simple to extremely complex. The basis of recommended systems in social networks, Ecommerce services, and online marketing is developed by link prediction. Link prediction can support to decrease the cost and time of researching protein-protein interactions, the interaction of biological cells, and genetic inheritance. In several situations, link prediction proposes an effective transport network and detects the structure of the criminal network. However, the applications will expand over time, which makes the networks grow more in size and speed. This most applications are in the form of the graph streams, bringing new challenges and opportunities in data mining field.

Link prediction studies the existence of new links (unaware interactions or new relationships) between pairs of nodes based on their features and the present considered links. The real-world network is a graph stream with the large amount of noisy and dynamic links. Essentially, the link prediction problem discovers knowledge and remodels topology on this dataset. Additionally, link prediction discovers the structure of the real-world network in the future. However, the datasets of the real-world networks are commonly dynamic, changing with the new links appear over time and become unobservability. These datasets are in the graph streams, not in snapshot graphs.

The most previous researches of exciting link prediction just concentrate studying link prediction in snapshot graphs, while most recent applications in the form of the graph streams. When predicting the existence of link in graph streams, we have two challenges: a) a large number of n nodes, which produce n^2 possible links, resulting in significant complexity and b) the rapid evolvement of graph stream. To increase the applicability, we research the link existence prediction subject in a more realistic context. Given a graph stream, this network has the large amount of links and has the rapid evolvement. To solve this problem, we present GSDC framework – an efficient framework can easily perform parallel computation and scale up the distributed system when necessary. GSDC framework can be used to compute the distributed similarity scores between nodes by the distribution of the information about edges and distributed machine learning. The outputs of GSDC framework are a DCS feature and a GBT method. DCS feature is a general feature that outperforms most features in

criteria such as precision, generalization, and speed. GBT method is a distributed method to deal with the massive datasets available today.With DCS feature and GBT method, we can use them to predict real-time existence links in graph stream. We experiment on three real-world networks. Our results on these datasets prove that our approach can be applied to various applications which require real-time computation.

The remaining of this paper is setting as follows. Sections 2 discusses related work. We present our proposed method called GSDC framework with DCS feature and GBT method in section 3. We show the results of experiment and discussion in section 4. The ending of this paper is the conclude in section 5.

2. RELATED WORK

Over the last decade, the scientific research community has had a particular interest in the link prediction problem [1]-[2]. It can be applied in any applications which can be modelled into a network. There are various research studies aimed to resolve the problems of link prediction [3]-[4]. Most of these studies have focused of a static graph [5]-[6].

The datasets of the real-world networks are commonly dynamic, changing with the new links appear over time and become unobservability. This real-world datasets are in the form of the graph streams. However, the number of research studies on link prediction in graph streams [7]-[8] is smaller than that of the static graph. The approaches to link prediction in graph streams include approximate matching, connectivity properties, frequencies of subgraphs, and graph distances. There are a few recently published studies related to the approximate matching approach, such as the cost-effective method proposed by Zhao et al [7]. This method is based on the neighborhood-based measures to calculate the accurate estimation similarity scores between any two nodes in graph streams. This is a great online approximate estimation method to trade accuracy for time and space cost.

Distributed Computing is a popular scientific field of computer science major with the target of data explosion (like Big Data), which has inspired researchers and developers to develop Distributed Systems in recent years [9]-[10]. Distributed Systems have solved many issues of data explosion or data mining, such as the massive volume and the high rate of evolving data [11]-[12].

Thus, we apply Distributed Systems to solve similar issues in link prediction in the graph stream problem. We present GSDC framework – an efficient framework can easily perform parallel computation and scale up the distributed system when necessary. Figure 1 shows the brief overview of the framework that we propose. The outputs of GSDC framework are a DCS feature and a GBT method.

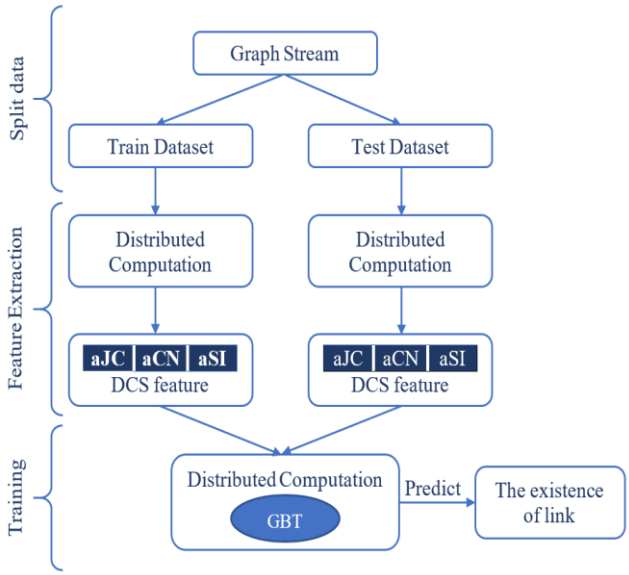


Figure 1: Graph Stream Distributed Computation framework

GSDC framework can be used to compute the distributed similarity scores between nodes by the distribution of the information about edges and distributed machine learning. With GSDC framework, we can use them to predict real-time existence links in graph stream.

3. THE PROPOSED METHOD

In this section, we describe a proposed method for link prediction in the graph stream problem. For the link prediction in the graph stream problem, we introduce the formal definitions firstly. Secondly, we introduce our features for the link prediction in the graph stream problem. Finally, we propose GBT method is a distributed method to deal with the massive datasets available today.

3.1 Problem Formulation

We assume a graph stream $G = (V, E)$ which contains the sequence of vertices and edges. We let V indicates the set of vertices, E indicates the set of edges and $G(t)$ is a graph stream at time t with a sequence of edges $E = (e_0, e_1, e_2, \ldots, e_n)$, where $e_i = (u, v, t)$ is the edge created by the interaction of two vertices u and v at the time t with $i \geq 0$.

In this paper, we study the link prediction in terms of predicting the existence of unobserved links or future links between pairs of nodes in a network. The overview of the whole link existence prediction in graph stream is given in the Figure 2. Link existence prediction in graph stream is defined with the following input and output:

- Input: The set of links in a graph stream at time T_i with $1 < i < N$.
- Output: The set of links will exist in the future T_N.

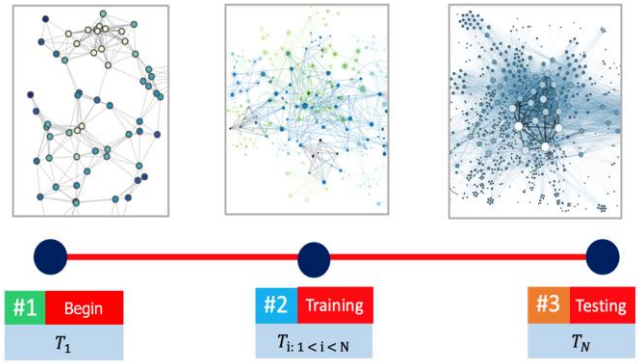


Figure 2: Link prediction in graph stream

The training phase uses the information of a graph stream at time T_i with $1 < i < N$. The testing phase and evaluation are at time T_N.

3.2 Proposed Features

The most previous researches of exciting link prediction just focused research on link existence prediction problems in snapshot graphs. In snapshot graph, with ***the exact matching method***, the similarity scores of two vertices computed by the formulas below:

$$CN(x,y) = |\Gamma(x) \cap \Gamma(y)| \quad (1)$$

$$SI(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x)| + |\Gamma(y)|} \quad (2)$$

$$JC(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|} \quad (3)$$

However, in a graph stream, the massive sizes of the adjacent vertex $\Gamma(x)$ and $\Gamma(y)$ result in inefficient computation cost for all pairwise vertices. So the main concept of our Distributed Computing splits the Graph Stream into multiple partitions, each of which is used to execute separate tasks in the system. Each task we present is used to compute the following similarity scores: approximation Common Neighbor (aCN), approximation Sorensen Index (aSI), and approximation Jaccard (aJC). We propose a Distributed Computation Score (DCS) feature which is constructed by combining these three indexes (aCN, aSI, and aJC). Our workflow using the Distributed Computing approach is shown in Figure 3 where the output is a DCS feature of all pairwise vertices in the graph stream.

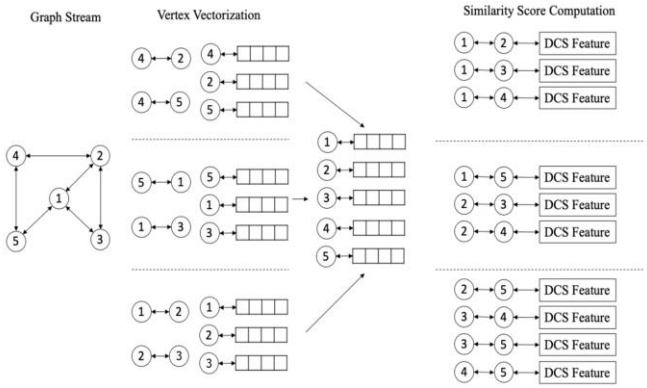


Figure 3: Illustration of the Distributed Computing Approach for DCS feature

To compute DCS feature, the paper describes **two computation steps** which are ***vertex vectorization*** and ***similarity score computation***.

A. Vertex vectorization

In the vertex vectorization step, we split a big graph stream into the smaller partitions $G(V,E) = (P_0, P_1, P_2, \ldots, P_m)$ and each partition is computed by an executor. In a partition, we map a sequence of edges into a sequence of vectors. Each vector is called the vectorial vertex and managed by a determined vertex. The dimension of these vectorial vertices is much smaller than that of adjacent vertices of raw edges. These vectorial vertices are the inputs for calculating the target similarity scores (DCS Feature in the next step). DCS Feature has aCN, aSI, and aJC score. We build vectorization methods suitable to compute each of these similarity scores. These methods are ***Sampling*** and ***MinHash method***.

The Sampling method for link prediction in graph stream has been published in [7]. This idea is combined with our distributed computation idea to extend link prediction in the graph stream problem. In aCN and aSI score, we use the method of sampling the adjacent vertex to reduce the space of the adjacent vertex. In graph stream, most vertices are high-degree, and hence, getting the sample in the traditional ways (like a reservoir sampling) is inefficient. Those reservoir sampling methods help us choose the same vertices in a sequence of adjacent vertices without knowing exactly their size. However, this traditional method can't choose the identical vertices from different adjacent vertices of each vertex so we propose to use a variant of reservoir sampling. This variant sampling is a biased-sampling, where each vectorial vertex is forced to be dependant on a hash function. We choose a vectorial vertex for each vertex with the lowest values generated by the hash function $\mathcal{H}(u): u \in V \rightarrow (0,1)$ where u is an identified vertex in the adjacent vertex. We set the number K as the budget for choosing the vertices of the vectorial vertex. The size of the vectorial vertex will be equal to or less than K. Then we also record the size of the actual adjacent vertices $\Gamma(x)$ as $d(x)$ for a computation approximation similarity score in the next step.

MinHash method [13] is an efficient technique that estimates the similarity (aJC) between two adjacent vertices. We choose H as the number of hash functions as well as the dimension of the vectorial vertex. Each element in the vector is a minimum adjacent hash value created by mapping adjacent identified vertices into adjacent hash values. For example, vectorial vertex x is a collection of elements $h_i(x)$. With each edge in a partition, the hash functions map the identified vertices x and y into two vectors managed by vertices x and y. The computed result of each partition is a collection of the distinct H- dimensional vectorial vertices.

At the end of the process mentioned above, we have a collection of distinct vectorial vertices in each partition. The dimension of the vectorial vertice is shown in Figure 4. After that, we use aggregation operators to collect all vectorial

vertices.

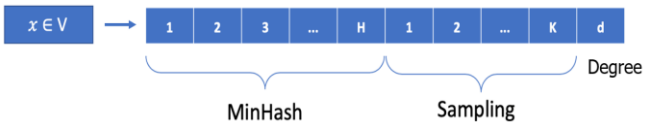


Figure 4: Illustration of the dimension of the vectorial vertices

B. Similarity score computation

After the vertex vectorization step, we will go through the similarity score computation step. We use the first H elements of vectorial vertex computed after vertex vectorization step to estimate the approximation Jaccard similarity score between two vertices with the following formula:

$$aJC(x,y) = \frac{\sum_i^K [h_i(x) = h_i(y)]}{H} \quad (4)$$

In vectorial vertex, we have the remaining elements with the budget K elements. We use these elements to compute the aCN and aSI score. We also propose to add a fraction $n(x)$ for preparing the next similarity score computation:

$$n(x) = min\left\{1, \frac{K}{d(x)}\right\} \quad (5)$$

We compute the remaining similarity scores using a distributed computation method with the following formulas. After that, we combine these scores with the aJC score into DCS feature.

$$aCN(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{max\{n(x), n(y)\}} \quad (6)$$

$$aSI(x,y) = aCN(x,y) * \frac{1}{d(x) + d(y)} \quad (7)$$

3.3 Distributed machine learning

In graph stream, we present Gradient Boosted Trees (GBT) method which can use to predict real-time existence links. GBT is a distributed machine learning method. We borrow the Gradient boosting from J. H. Friedman (2001) [14] to design the distributed computation approach (GBT method).

Gradient descent and Boosting are two techniques and are incorporated into gradient boosting. Gradient boosting is also called as steepest descent method. To understand gradient boosting clearly, we introduce ***Gradient-descent optimization*** [14] and ***Boosting*** [14] first. Then, we explain on ***Gradient boosting*** for classification in this section.

A. Gradient-descent optimization

Deterministic and stochastic approaches can be separately used in the optimization problems. In these problems, deterministic methods are confirmed that it normally quicker than stochastic ones. However, deterministic methods are at higher risk of being stranded in a local minima than stochastic ones. Some stochastic methods use to turn hyperparameterand threshold. The gradient-descent is a deterministic nonparametric iterative approach. This approach has normally been used to decrease the experienced risk by numerical function optimization [15].

With a function $f: R^n \rightarrow R$, we have:

$$\nabla f(x) = grad\ f(x) = \left(\frac{\partial f(x)}{x_1}, \dots, \frac{\partial f(x)}{x_n}\right)^T \quad (8)$$

The gradient is an only vector field. It is a result of applying the *nabla* operator ∇ on function f. We chose a starting point $x^{(0)}$ as an initial evaluate. This step is a first step for decreasing function $f(x)$. In iteration m, the starting point will be improved with:

$$x^{(m)} = x^{(m-1)} - v\nabla f\left(x^{(m-1)}\right) \text{ for } m = 1, \dots, M \quad (9)$$

Where v can change and manage the step size towards the steepest descent after each iteration. This helps minimize the objective function:

$$v^{(m)} = argmin_{v>0} f\left(x^{(m-1)} - v\nabla f\left(x^{(m)}\right)\right) \quad (10)$$

In this line search function, the algorithm reachesan $x^{(m)} \in R^n$ with $\nabla f\left(x^{(m)}\right) = 0 \in R^n$ after the iterations. Then, a local minima is reached.

B. Boosting

Valiant [16] and Kearns [17] presented the Boosting in their PAC-learning framework. Then, Robert E. Schapire et al. [18] developed it and demonstrated that we can train additional ones to appreciate the performance of a weak learner. In predictive performance problem, the random chance has lower performance than weak learners with such required property. In the modeling phase, this concept is the basis of Boosting. The goal is to minimize the empirical risk:

$$\mathcal{R} = \sum_{i=1}^{n} L\left(y_i, f(x_i)\right) = \sum_{i=1}^{n} L\left(y_i, \sum_{m=1}^{M} \beta_m h(x_i, \theta_m)\right) \quad (11)$$

The parameters β_m, parameters θ_m and the learners $h(x, \theta_m)$ affect $\mathcal{R}$. Thus, the R needs to be scaled down in terms of parameters $(\beta, \theta) = ((\beta_0, \theta_0), \dots, (\beta_M, \theta_M))$ by using the loss function L. Following J. H. Friedman (2001) [14], the iterative modeling approach can more optimization.

The front models are not readjusted and become invariable when supplement each component phase. In the machine learning context, it is called the Boosting and improve the predictive performance strongly.

C. Gradient boosting for classification

Gradient descent and Boosting are two techniques and are incorporated into the gradient boosting. This algorithm use gradient descent to minimize the empirical risk $\mathcal{R}$. The target variable does not accommodate consecutive. This variable has only two other class levels, e.g. $y \in \{0, 1\}$. This is the difference with regression. The function $\mathbb{I}(f(x) > 0)$ will set discrete predictable results if the model's output gives real values. The positive value is determined as class 1 and the negative value is determined as class 0.

The K tree models predict the pseudo residuals $r_{ik,m}$ in each iteration m. In districts $\{R_{1k,m} \dots, R_{Jk,m}\}$, each tree has J terminal nodes to corresponding update

$$\gamma_{jk,m} = argmin_{\gamma_{jk}} \sum_{i=1}^{n} \sum_{k=1}^{K} \emptyset \left(\gamma_{jk}, f_{k,m-1}(x_i) + \sum_{j=1}^{J} \gamma_{jk} \mathbb{I}(x \in R_{j,m}) \right) \quad (12)$$

With $\phi(y_k, f_k(x)) = -y_k \ln(\pi_k(x))$

This formula is approximated by a single Newton-Raphson step because there is no closed solution. This formula is split into individual calculations for each terminal node, following J. Friedman et al. [14].

$$\gamma_{jk,m} = \frac{K-1}{K} \frac{\sum_{x_i \in R_{jk,m}} r_{ik,m}}{\sum_{x_i \in R_{jk,m}} |r_{ik,m}|(1-|r_{ik,m}|)} \quad (13)$$

which serves for the update

$$f_{k,m}(x) = f_{k,m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jk,m} \mathbb{I}(x \in R_{jk,m}) \quad (14)$$

Finally, after M steps, $f_{k,M}(x)$ is returned as final model. A formal description of ***K-class Classification Gradient Tree Boosting Algorithm*** [14] is given in Table 1.

Table 1: K-class Classification Gradient Tree Boosting Algorithm

Algorithm 1: K-class Classification Gradient Tree Boosting Algorithm.

Initialize: $f_{k,0}(x) = 0, k = 1, \dots, K$

for $m = 1 \to M$ **do**

  Set $\pi_k(x) = \frac{exp(f_k(x))}{\sum_{j=1}^{J} exp(f_j(x))}$ **for** $k = 1 \to K$ **do**

    Calculate $r_{ik,m} = y_{ik} - \pi_{k,m-1}(x_i), i = 1, \dots, n.$

    Fit regression tree to the pseudo-residuals $r_{ik,m}$ *given terminal regions* $R_{jk,m}, j = 1, \dots, J_m$

    for $j = 1 \to J_m$ **do**

      $\gamma_{jk,m} = \frac{K-1}{K} \frac{\sum_{x_i \in R_{jk,m}} r_{ik,m}}{\sum_{x_i \in R_{jk,m}} |r_{ik,m}|(1-|r_{ik,m}|)}$

    end

    Update

    $f_{k,m}(x) = f_{k,m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jk,m} \mathbb{I}(x \in R_{jk,m})$

  end

end

Output: $\hat{f}_k(x) = f_{k,M}(x)$

This algorithm is designed and executed on the distributed computation system. This method is called ***Gradient Boosted Trees*** (GBT) method, facilitating a scalable distributed execution on the Apache Spark platform. The overview of the whole our predictive execution system is given in the Figure 5.

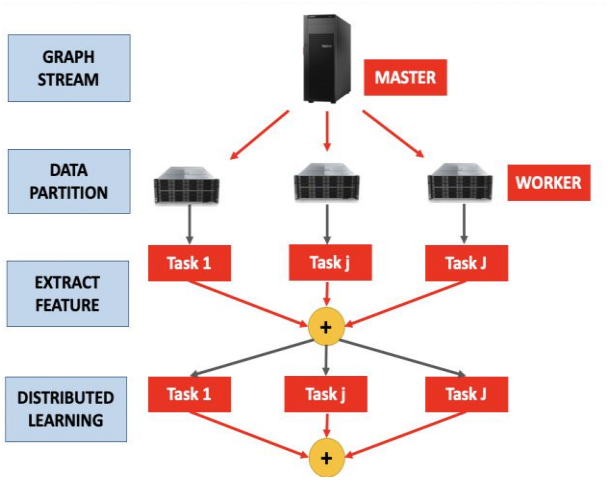


Figure 5: Simulate predictive execution system

The entire proposed model is based on distributed computing. Each Worker machine will assume a role in computing the distributed operators. Worker workstations do their own tasks, then aggregate and send them to the Master server.

4. EXPERIMENTS

This section evaluates our link prediction approach by the experimental studies. We introduce three real-world graph streams datasets. Then, we briefly present the evaluation methods and experimental settings. After conducting experiments, we show experimental results and discussions.

4.1 Datasets

We choose three real-world graph streams datasets:

1) ***DBLP***[19]: This dataset is a bibliographic information dataset on major computer science publications. We extracted all conference papers from 1940 to 2015. This graph includes 1,411,376 vertices and 10,597,380 edges.

2) ***Wikipedia*** [20]: This graph stream represents the development of the Wikipedia knowledge base. This graph stream contains 1,870,709 vertices and 39,953,145 edges. In this graph, vertices are articles in Wikipedia, and edges define the relationship between articles.

3) ***Amazon*** [21]: This graph stream gets information about various products (Books, music CDs, DVDs and VHS video tapes). It includes these product metadata. There is a total of 410,271 vertices and 11,179,587 edges in the graph streams.

Table 2 present the properties of the three networks.

Table 2: The properties of the three networks

Network	Vertices	Edges
DBLP	1,411,376	10,597,380
Wikipedia	1,870,709	39,953,145
Amazon	410,271	11,179,587

4.2 Evaluation Methods

Our method is evaluated by three criteria:

A. Differences between values

We use the root-mean-square error (RMSE). This measure is used to assess the difference between sample and population values. As our evaluation, RMSE shows the difference between approximate similar score and exact similar score. It is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \quad (15)$$

B. Precision

The precision evaluates the classify model by the ratio between tp is the amount of true positives and fp the amount of false positives. The formula for this measurement is calculated as follows:

$$Precision = \frac{TP}{TP + FP} \quad (16)$$

C. Computation cost

Our distributed computation framework aim to predict realtime existence link graph stream, so the computation cost is an important element for efficient evaluation. We measure the execution time of the methods.

4.3 Experimental Settings

We install nine methods on two stages for evaluating experiment.

A. The feature extraction:

- *The exact matching method*: Using CN, SI and JC to measure the similarity scores of two vertices.
- *DCS feature (**our feature**)*: Using aCN, aSI and aJC to measure the similarity scores of two vertices.

B. Predicting the existence of links:

- *CN+GBT*: Using CN score with GBT method.
- *SI+GBT*: Using SI score with GBT method.
- *JC+GBT*: Using JC score with GBT method.
- *aCN+GBT*: Using aCN score with GBT method.
- *aSI+GBT*: Using aSI score with GBT method.
- *aJC+GBT*: Using aJC score with GBT method.
- *DCS+GBT (**our approach**)*: Using DCS feature with GBT method.

To prove the efficiency of our approach, we execute all experiments, differences between values and precision, and the scalability when the data evolves. Our experiment setup is implemented in the distributed mode on the cluster. We use Apache Spark [22] – a unified analytics engine to perform our experiment studies on this cluster.

This cluster includes 4 cloud servers on Open Stack as 1 master and 3 workers. Each worker runs on the Ubuntu Server operating system and comes with the following specifications: Intel Xeon E3-12xx v2 3.0GHz, 8 virtual processors, 32 GB RAM. We set 100 to the value of both H (the number of hash functions) and K (the reservoir budget).

4.4 Experiments Results and Discussions

A. The feature extraction

We evaluate the computation cost of our approach through each step. We present the execution time of the whole process of similarity feature extraction which is a *DCS feature* (our feature) to compare to the feature of *the exact matching method*. Figure 6shows the runtime cost of similarity feature extraction of two methods.

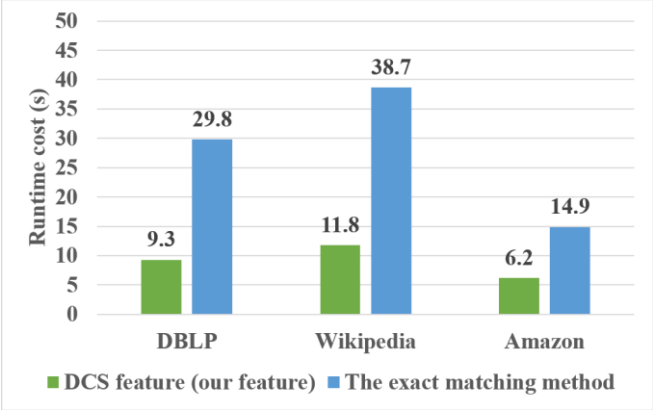


Figure 6: Runtime cost (s) of similarity feature extraction between DCS feature and the feature of the exact matching method

In Table 3, we show the error of similarity score computation.

Table 3: The RMSE of similarity score feature computation

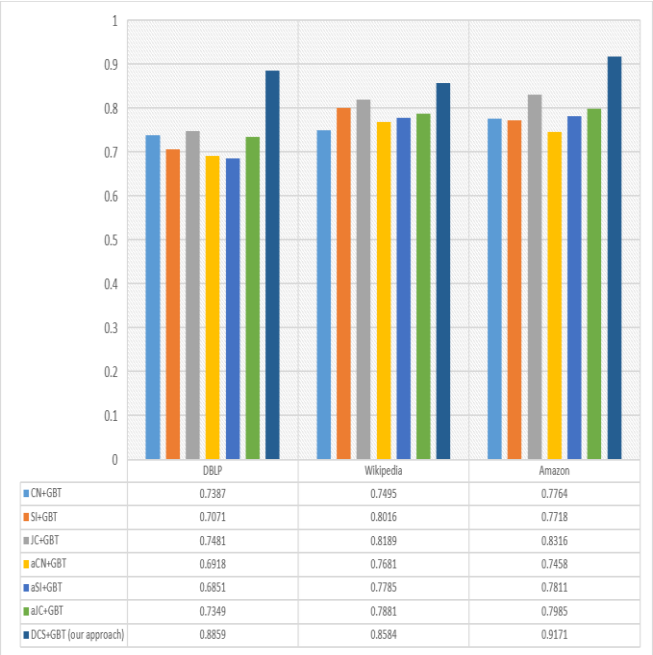
Score \ Dataset	DBLP	Wikipedia	Amazon
aCN	5.89%	3.13%	4.24%
aSI	4.56%	2.68%	4.71%
aJC	8.42%	8.81%	5.25%

DCS feature is the output of GSDC framework which is directly amendable to parallelization, facilitating a scalable distributed execution on the Apache Spark platform. Thus, this distributed systems have solved two challenges in link prediction in the graph stream problem.

In comparison with the exact matching method, Figure 6and Table 3 show that our feature can reduce the execution time by 3x times and the errors always below 9%. So GSDC framework can be used to figure out the distributed similarity scores between nodes by the distribution of the information about edges in real-time.

B. Predicting the existence of links

The Precision score is used to evaluate our approach for link prediction problem. In Figure 7, we present the Precision of *CN+GBT*, *SI+GBT*, *JC+GBT*, *aCN+GBT*, *aSI+GBT*, *aJC+GBT* and *DCS+GBT* on three real-world graph streams datasets.



	DBLP	Wikipedia	Amazon
CN+GBT	0.7387	0.7495	0.7764
SI+GBT	0.7071	0.8016	0.7718
JC+GBT	0.7481	0.8189	0.8316
aCN+GBT	0.6918	0.7681	0.7458
aSI+GBT	0.6851	0.7785	0.7811
aJC+GBT	0.7349	0.7881	0.7985
DCS+GBT (our approach)	0.8859	0.8584	0.9171

Figure 7: Precision of existing link prediction

The DCS feature that is the output of GSDC framework has the errors always below 9%. The GBT method is a distributed method to deal with the massive datasets available today. Therefore, ***DCS+GBT (our approach)*** uses DCS feature with GBT method have high precision.

Figure 7shows that our method is 21% more precision than the baseline method. With DCS feature and GBT method, we can use them to predict real-time existence links in graph stream.

5. CONCLUSION

We researched link prediction problem in graph stream using the distributed computation approach in this paper. We focus how to effectively real-time existence link in graph streams. We propose an efficient Graph Stream Distributed Computation framework (GSDC) which is directly amendable to parallelization, facilitating a scalable distributed execution on the Apache Spark platform. With our framework, we present the Distributed Computation Score feature (DCS) and Gradient Boosted Trees (GBT) which are designed to distribute computation approach. Our method significantly improves on advance methods in two criteria precision and speed. The results of experiments on three networks DBLP, Wikipedia and Amazon prove it. Our approach has solved two challenges in link prediction in the graph stream problem.

ACKNOWLEDGEMENT

This research is funded by Vietnam National University Ho Chi Minh City (VNU-HCM) under grant number C2018-26-10.

REFERENCES

1. Z. Huang, X. Li and H. Chen,**Link prediction approach to collaborative filtering**, In *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, 141142, New York, USA, 2005.
2. W. Q. Wang, Q. M. Zhang and T. Zhou,**Evaluating network models: A likelihood analysis**, *Europhys. Lett. 98*, 28004, 2012.
3. D. Liben-Nowell and J. Kleinberg,**The link-prediction problem for social networks**, *J. Am. Soc. Inf. Sci. Tec. 58*, 10191031, 2007.
4. A. Clauset, C. Moore and M. E. Newman,**Hierarchical structure and the prediction of missing links in networks**, *Nature 453*, 98101, 2008. https://doi.org/10.1038/nature06830
5. P. Wang, B. Xu, Y. Wu and X. Zhou,**Link prediction in social networks: the state-of-the-art**, *Science China Information Sciences 2015; 58(1):138.*
6. L. L and T. Zhou,**Link prediction in complex networks: A survey**, *Physica A 390*, 11501170, 2011.
7. P. Zhao, C. Aggarwal and G. He,**Link prediction in graph streams**, *2016 IEEE 32nd International Conference on Data Engineering (ICDE). IEEE*, 2016, pp. 553–564.
8. S. Chang, Y. Zhang, J. Tang, D. Yin, Y. Chang, Hasegawa-Johnson and M.A. Huang,**Positive-unlabeled learning in streaming networks**, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 755–764, 2016.
9. Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin and J. M. Hellerstein,**Distributed Graphlab: A framework for machine learning and data mining in the cloud**, In *PVLDB*, 2012. https://doi.org/10.14778/2212351.2212354
10. J. Ahn, S. Hong, S. Yoo, O. Mutlu and K. Choi,**A scalable processing-in-memory accelerator for parallel graph processing**, *Proc. ISCA*, 2015.
11. Syed.Karimunnisa, Dr.Vijaya Sri Kompalli, **Cloud Computing: Review on Recent Research Progress and Issues**, In *IJATCSE*, Volume 8, No.2, pp. 216-223, 2019. https://doi.org/10.30534/ijatcse/2019/18822019
12. Saida EL MENDILI, Younès EL BOUZEKRI EL IDRISSI, Nabil HMINA, **Big Data Processing Platform on Intelligent Transportation Systems**, In *IJATCSE*, Volume 8, No.4, pp. 1099-1109, 2019. https://doi.org/10.30534/ijatcse/2019/16842019
13. Broder and Z. Andrei,**On the resemblance and containment of documents**, *Compression and Complexity of Sequences*, Italy, June 11-13, 1997.
14. J. H. Friedman,**Greedy function approximation: A gradient boosting machine**, *Ann. Statist. 29.5*, pp. 1189–1232, 2001.
15. G. E. Hinton, J. L. McClelland & D. E. Rumelhart,**Parallel distributed processing: Explorations in the microstructure of cognition**, *Vol. I: Foundations, ed. D. E. Rumelhart, J. L. McClelland & the PDP. Research Group*, pp. 77–109, 1986.

16. L. G. Valiant,**A Theory of the Learnable**, *Commun. ACM 27.11*, pp. 1134–1142, 1984.
17. J. K. Michael and V. Umesh,**An Introduction to Computational Learning Theory**, *Cambridge*, MA, USA: MIT Press, 1994.
18. E. S. Robert, F. Yoav, B. Peter and S. L. Wee,**Boosting the margin: a new explanation for the effectiveness of voting methods**, *Ann. Statist. 26.5*, pp. 1651–1686, 1998.
19. M. Ley,**Dblp: some lessons learned**, *Proc. VLDB Endow.*, 2(2):1493–1500, Aug. 2009.
20. A. Mislove,**Online social networks: Measurement, analysis, and applications to distributed information systems**, *Ph.D. dissertation, Department of Computer Science, Rice University*, 2009.
21. J. Leskovec, L. Adamic and B. Adamic,**The Dynamics of Viral Marketing**, *ACM Transactions on the Web (ACM TWEB)*, 1(1), 2007. https://doi.org/10.1145/1232722.1232727
22. M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker and I. Stoica,**Spark: Cluster computing with working sets**, *in Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, vol. 10 , p. 10, Boston, MA, USA, 2010.