IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 21, 2019, accepted April 8, 2019, date of publication April 12, 2019, date of current version September 25, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2910804

Cloud Service Community Detection for Real-World Service Networks Based on Parallel Graph Computing

YU LEI[1], (Member, IEEE), AND PHILIP S. YU[2], (Fellow, IEEE)
[1]Department of Computer Science, Inner Mongolia University, Hohhot 010021, China
[2]Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607, USA

Corresponding author: Yu Lei (yuleiimu@sohu.com)

This work was supported by grants from NSFC under Grant (No. 61962040).

ABSTRACT Heterogeneous information networks (e.g. cloud service relation networks and social networks), where multiple-typed objects are interconnected, can be structured by big graphs. A major challenge for clustering in such big graphs is the complex structures that can generate different results, carrying many diverse semantic meanings. In order to generate desired clustering, we propose a parallel clustering method for the heterogeneous information net-works on an efficient graph computation system (Spark). We use a multi-relation and path-based method to create similarity matrices, and implement our method based on graph computation model. It is inefficient to directly use existing data-parallel tools (e.g. Hadoop) for graph computation tasks, and some graph-parallel tools (e.g. Pregel) do not effectively address the challenges of graph construction and transformation. Therefore, we implemented our parallel method on the Spark system. The experiment results of clustering show our method is more accuracy.

INDEX TERMS Heterogeneous information networks, cluster, parallel computing, service mashup, community detection.

I. INTRODUCTION

Most of the world's densely populated areas have entered the era of cloud computing and the domestic communication industry is about to enter the 5G era. People need to access the network anytime, anywhere, using various convenient cloud services. We established a cloud service community to provide rich in-site cloud services, thereby attracting service providers to release more services and finally promoting their services to fixed and mobile users. The number of service developers and users are increasing dramatically. Therefore we built a web site to help them find each other. Furthermore, we study the web site data and test our various methods on it. Other researchers are also welcome to test their methods in the web site. Our web site is indexed and analyzed by a search engine Baidu. Total page views so far are 16632, and user locations are shown in Fig.1 according to Baidu's statistics.

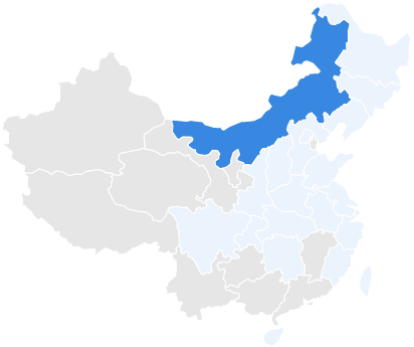

FIGURE 1. User locations.

According to these statistics, we want to mine interesting service information. Link-based clustering methods are able to discover hidden knowledge in big data. Contrast to other algorithms [1], link-based clustering methods use links instead of object attributes to classify objects when the attributes are missing. Most link-based clustering methods are used in homogeneous networks where only one type of links exists. However, some networks are heterogeneous, which have multiple types of objects and multiple types of links. Any two objects can be linked via different types

The associate editor coordinating the review of this manuscript and approving it for publication was Shuiguang Deng.

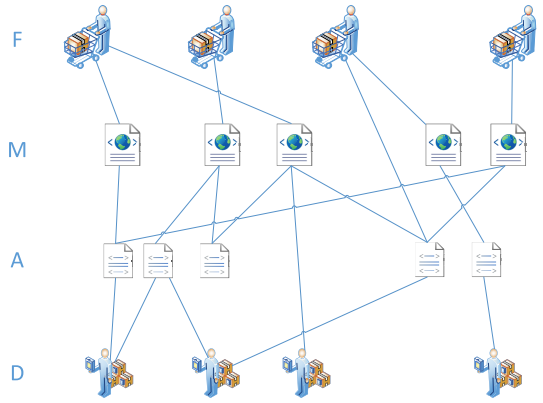

FIGURE 2. A heterogeneous network for cloud services.

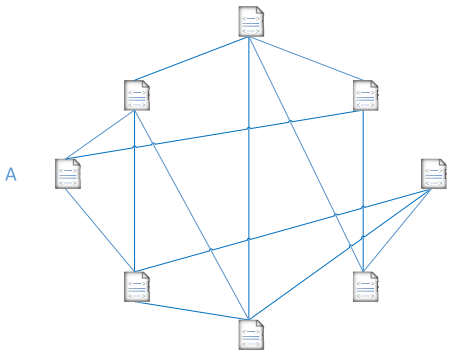

FIGURE 3. API similarity graph under different paths.

of paths. When clustering on diverse paths, different results will be generated.

A heterogeneous information network is shown in Fig. 2, which contains four types of objects: developers (D), Followers (F), APIs (A) and Mashups (M), and four types of links. Links exist between developers and APIs by the relations of ''develop'' and ''developed by'', between APIs and Mashups by ''integrate'' and ''integrated by'', between developers and Mashups by ''develop'' and ''developed by'', between followers and Mashups by ''invoke'' and ''invoked by'', and between followers and APIs by ''invoke'' and ''invoked by''.

In Fig.2, APIs (third layer) are indirectly connected via different paths. A − D − A denotes a relation between APIs via developers (fourth layer), whereas A − F − A shows a relation between APIs via followers (first layer) and A − M – A shows a relation of Mashups (second layer). Different paths generate different connection graphs. For example, APIs can be connected via developers and form two clusters; APIs can be connected via followers and form two different clusters; APIs can be connected via a mashup developed cooperatively by different developers, and construct two different clusters. Whereas in Fig.3, a connection graph combining the three paths may generate 3 clusters.

The three clusters carry diverse semantics, and thus users should choose their desired path. It is easier for domain experts to specify one path or multiple weighted paths. Mashup Web sites (e.g., ProgrammableWeb [2]) can help users classify APIs into different categories.

In a heterogeneous information network, we use varying path and domain knowledge to cluster varying types of objects, where the domain knowledge is used to assign weights of varying paths.

To cluster APIs (nodes) into 2 clusters in Fig.3, weights 0.5, 0.3, and 0.2 are assigned to the three paths: A − D − A, A − F − A and A − D − M − D − A. The combination of the paths is shown in Fig.3. For a clustering task, we assign the weight of each path, which should be consistent with the clustering results implied by the domain knowledge, and then output the clustering result.

We propose a clustering method for varying types of objects. An efficient distributed graph-parallel algorithm is developed. We summarize contributions as follows.

- A novel multiple label propagation model is proposed to cluster overlapping community in a heterogeneous real-world service network we built.
- We propose a parallel computing algorithm based on the multiple label propagation model. The parallel computing algorithm (GMPLA, Graph-based Multiple Label Propagation Algorithm) is implemented by graph computing (Spark), where the graph node relations are constructed by different paths.

II. RELATED WORK

Different types of relationships have different semantic meanings in determining the similarity between target objects. Clustering on the homogenous network have been studied for long. A heuristic density-based approach for community detection is proposed [1], but it highly depends on the distance function. Arab and Afsharchi [3] proposed a bottom up community detection method which starts with fine-grained communities (the condition is too strict in common cases). Merging preliminary small communities is done in a hybrid way to maximize two quality functions: modularity and NMI. Papadakis *et al.* [4] proposed an unsupervised distributed algorithm that finds the entire community structure of a network, on the basis of local interactions between neighboring nodes. The proposed approach is based on the use of Vivaldi synthetic network coordinates. Wang *et al.* [5] developed an algorithm to extract a hierarchy of overlapping communities, which can zoom into a network at multiple different resolutions and determine which communities reflect a targeted behavior. However, time complexity is too high for big data. Qi *et al.* [6] proposed an algorithm for community selection. Based on the Max-Flow Min-Cut theorem, the algorithm can output an optimal set of local communities automatically. However, the algorithm is hardly implemented in parallel style. Dinh and Thai [7] proposed polynomial-time approximation algorithms for the modularity maximization problem together with their theoretical justifications in the context of scale-free networks. This work did not discuss heterogenous Web service data.

Clustering on the heterogenous network also have been studied. We introduced several clustering methods on the heterogenous networks. Exploring the heterogenous digital footprints of LBSN (location-based social networks) users in the cyber-physical space, Wang *et al.* [8] proposed an edge-centric co-clustering framework to discover overlapping communities. Zhou and Liu [9] presented a social influence based clustering framework for analyzing heterogeneous information networks with three unique features. Kuo *et al.* [10] predicted the opinion holder in a heterogeneous social network without any labeled data. This question can be generalized to a link prediction with aggregative statistics problem. They devised an unsupervised framework to solve this problem, and evaluated the method using four datasets: preference (Foursquare), repost (Twitter), response (Plurk), and citation (DBLP). Tang *et al.* [11] showed that representative community detection methods for single-dimensional networks can be presented in a unified view. Comar *et al.* [12] considered the problem of multi-task learning on heterogeneous network data. They presented a framework that enables one to perform classification on one network and community detection in another related network. Angelova *et al.* [13] modeled the mutual influence of nodes as a random walk in which the random surfer aims at distributing class labels to nodes while walking through the graph. The methods of papers solving the problem of heterogenous network are not suitable for the large-scale parallel environment.

Recently researchers proposed several extended label propagation algorithms. Lin *et al.* [14] proposed a community detection method based on the label propagation algorithm with community kernel. They assigned a corresponding weight to each node according to node importance in the whole network and update node labels in sequence based on weight. Liu and Murata [15] analyzed a modularity-specialized label propagation algorithm (LPAm) for detecting network communities. To escape local maxima, they used a multistep greedy agglomerative algorithm (MSG) that can merge multiple pairs of communities at a time. Combining LPAm and MSG, they proposed a modularity-specialized label propagation algorithm (LPAm+). The extended label propagation algorithms have good prospects for solving large-scale data, but the convergence condition is hard to determine.

Clustering based on parallel algorithms have been prosed recently. Gregori *et al.* [16] presented a parallel k-clique community detection method The method has an unbounded, user-configurable, and input-independent maximum degree of parallelism, and hence is able to make full use of computational resources. Bu *et al.* [17] proposed a fast parallel modularity optimization algorithm. To deal with large graphs with millions of vertices and edges, Prat *et al.* [18] proposed a disjoint community detection algorithm called Scalable Community Detection (SCD). By combining different strategies, SCD partitions the graph by maximizing the Weighted Community Clustering, which is based on triangle analysis. Lu *et al.* [19] presented parallelization heuristics for fast community detection using the Louvain method as the serial template. The Louvain method is an iterative heuristic for modularity optimization. Compared to the serial Louvain implementation, the parallel implementation is able to produce community outputs with a higher modularity for most of the inputs tested. Shi *et al.* [20] presented a community-detection solution for massive-scale social networks using MapReduce. They proposed a set of degree-based preprocessing and postprocessing techniques that improve both the community-detection accuracy and performance. Most parallel algorithms are complex, inefficient, and hard to implement, while parallel LP algorithm is a good choice.

Cluster ensemble [21] is a method that composes clustering results of different methods. Given different partitions of objects, cluster ensemble methods can find a mean partition. Nevertheless, the clusters representing different purposes of clustering tasks may conflict with each other, and undesired by users.

Instead composing clustering results at output, our method compose related objects during the formation of clusters. Our work differs from traditional semi-supervised feature selection, which focus on vector space features. Our method provides a source of features (i.e. path), instead of a concrete feature. In addition, we show that good solution cannot be obtained by simple combinations of features from different sources.

III. PARALLEL MULTIPLE LABEL PROPAGATION ALGORITHM BASED ON GRAPH COMPUTING

A heterogeneous information network is a network where multiple types of objects link each other with multiple types of links. Objects can link various types of objects by different paths in a heterogeneous information network. Each path may lead to different quality of clustering. We must determine the target type of objects, before we cluster the objects. Then, we must determine the type of connection. Our parallel label propagation method will cluster objects once paths are selected.

A. LABEL PROPAGATION

A label, such as an integer, is associated with each vertex in the basic label propagation algorithm.

(1) A unique label is initially assigned to each vertex v.
(2) Vertex v iteratively replaces its label by the neighbors' label which has the maximum quantity.
(3) Finally, the vertices with same label tend to a community.

Step 2 is called propagation phase, which may not terminates where all vertices will not change their labels. Asynchronous updating can be used to ensure the propagation phase converge to a stable state. Vertex v's new label is updated in successive iterations according to the labels of its neighbors in the previous iteration. When each vertex has a label that

is used by a maximum number of neighbors, the algorithm terminates.

We analyzed the basic label propagation algorithm. Its time complexity is linear to the network size. Step 1 takes time O(n), step 2 takes time O(n) for each iteration and step 3 takes time O(n) for processing disconnected communities. The number of iterations required depends on the network size. We compared asynchronous updating with synchronous updating. The results show that synchronous updating is much more stable, but it requires more iterations.

There are also some methods than are able to constrain the propagation of labels for limiting the size of communities, and able to detect hierarchical communities. Nevertheless, the basic label propagation algorithm is not able to detect overlapping communities.

B. PARALLEL OVERLAPPING COMMUNITIES CLUSTERING

1) LABEL PROPAGATION FOR OVERLAPPING COMMUNITIES

Each vertex belongs to a single community identified by a label in the basic label propagation algorithm. However, each vertex may belong to more than one community when communities overlap. Therefore, more than one community identifiers are needed for overlapping communities.

As a naive method, we allow a vertex to obtain all labels that it can obtain from neighbors, and label each vertex by a unique community identifier in the beginning. In the successive iterations, the vertex will obtain all of neighbors' community identifiers into its label set. Each vertex end up gathered all community identifiers, thus this method does not work well.

In another way, each vertex v can be labeled by a range of community identifiers c. We let $b(c)$ of vertex v indicate the degree of membership of community c, and set sum $b(c)$ of v is 1. In each label propagation step, the value of $b(c)$ will change since the vertex constantly obtain neighbors' labels. Therefore, we let $b_t(c)$ indicate the membership in a certain iteration t, which is based on iteration $t-1$. In Equ.1, $N(v)$ denotes all neighbors of v.

$$b_t(c) = \frac{\sum_{u \in N(v)} b^u_{t-1}(c)}{|N(v)|} \tag{1}$$

We just keep more than one community identifier in each vertex, and then delete the labels whose $b_t(c)$ is less than a threshold during each propagation step. The threshold is $1/cmax$. The parameter $cmax$ shows the maximum number of communities that a vertex belong, and $cmax$ can be input by users.

There is a situation that vertex $v's$ all degree of membership are less than the threshold. Then, the greatest degree of membership will be kept, and all others will be deleted. When it has multiple maximum $b_t(c)$ that below the threshold, we randomly select one of them. After deleting these labels, renormalize the degree of membership is needed so that they are summed to 1.

2) PROPAGATION THRESHOLD

The design of propagation threshold is critical. Several alternative thresholds are considered when we design it in the propagation phase.

A low-degree vertex may belong to less communities than a high-degree vertex. Thus, the maximum number of communities of a vertex rely on its degree. To avoid this problem, an alternative threshold is given: max ($1/cmax$, $c/d(v)$). We keep community labels sent by at least c neighbors of vertex v, instead of a *fraction* of neighbors. If $d(v)$ is the degree of v, the threshold becomes $c/d(v)$ instead of $1/cmax$. The result is that vertex v can belong to $d(v)/c$ communities at most. However, this threshold is not good enough partly because low-degree vertices are common and it requires two parameters.

To specifically deal with the problem that all degree of membership of a vertex are less than the threshold. Instead of keeping *one* of them, we can leave the vertex unlabeled, and expect that it will be labeled in the future iterations.

To prevent forming excessively large communities, we count the number of times that a dominant label appears on a vertex. If a label often appears on vertex v, we set the label to v only and delete all other labels on vertex v.

3) ALGORITHM

Parallel program pattern is used to implement our above algorithm (it is necessary to read paper [22]).

IV. EXPERIMENTS

Multiple label propagate algorithm can be implemented in several ways. However, using data-parallel tools (Hadoop) to graph computation tasks is inefficient. Thus, new graph-parallel tools (Pregel, etc.) designed to efficiently execute graph algorithms are developed. Unfortunately, some graph-parallel tools do not efficiently address the graph construction and transformation. Spark GraphX combines the advantages of both graph-parallel and data-parallel systems, thus we use GraphX as our implementation tool. Comparing *GMLPA* with several methods in this section, we show the efficiency of our algorithm.

A. LABEL PROPAGATION

We use DBLP network [23], ProgrammableWeb network, and a real-world service network [24] for performance study. Several clustering tasks are provided for each network followed by evaluation:

Three clustering tasks are designed for DBLP Network:

(1) Fig.4: Major conferences, including all related authors, papers and terms in Data Mining, Information Retrieval and Machine Learning, are clustered. The selected paths are $V-P-T-P-V$ and $V-P-A-P-V$. We denote that Term (T), Author (A), Paper (P), and Venue (V).
(2) Fig.5: Authors are clustered according to the number of publications and research areas. The selected paths are

Algorithm 1 Parallel GMLPA

Randomly assign a label to each node in the beginning
// gather from neighbors
gather(D_u, D_{u-v}, D_v) {
    obtain b_t (c) *of* D_v
    return w_{u-v} $b_t(c)$
    //where w_{u-v} is the weight of the edge $\{u,v\}$
}
sum(a, b) {return a ∪ b}
apply(D_u, acc) {
//Calculate b_t (c) *of* D_u *from* acc, acc contains the result of the gather.
Let the maximum $b_t(c)$ *be* b_{max}
    i = 0
    While (i < type numbers of obtained labels)
    {
        $p_w(i) = b_t(i)\ /b_{max}$
        // note that if the value of b_{max} is small, then the other $b_t(i)$ is small
        if($p_w(i)$ > threshold)
        {B ∪ $b_t(i)$}
        //B is the set of labels which u contains
    }
    Rank $b_t(c)$ *in B*
    Only retain top-v $b_t(c)$
    if(all of $p_w(i)$ < threshold)
        {B ∪ b_{max}}
    Normalize $b_t(c)$ *of* D_u
    Calculate r_t
}
    // send to neighbors
scatter(D_u, D_{u-v}, D_v) {
    Activate node v if node u changes sufficiently
    if(r_t != r_{t-1})
        Activate(v)
}

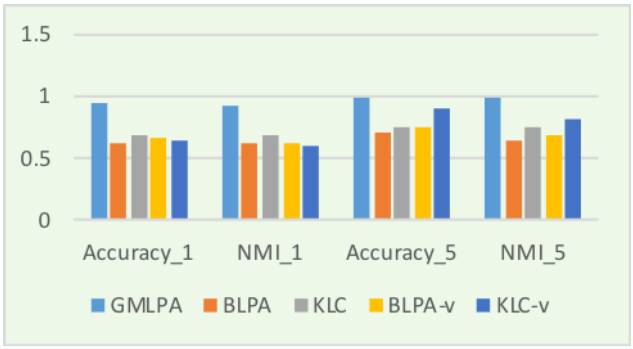

FIGURE 4. Clustering accuracy for DBLP.

$A-P-T-P-A, A-P-V-P-A, A-P-A-P-A$, and $A-P-A$.

(3) Fig.6: Cluster authors studying in machine learning. The selected paths are the same with previous settings.

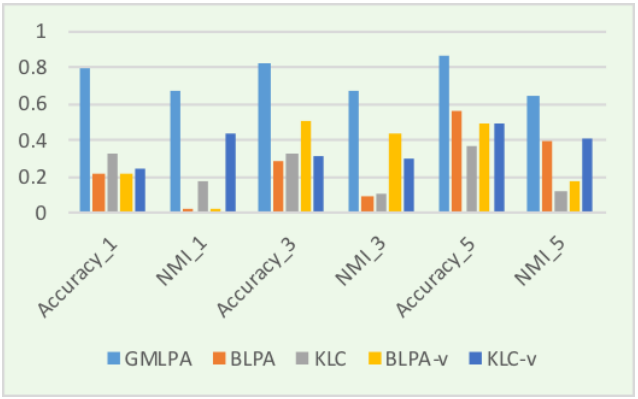

FIGURE 5. Clustering accuracy for DBLP.

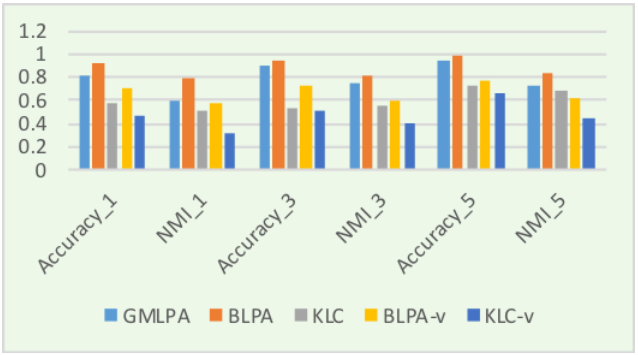

FIGURE 6. Clustering accuracy for DBLP.

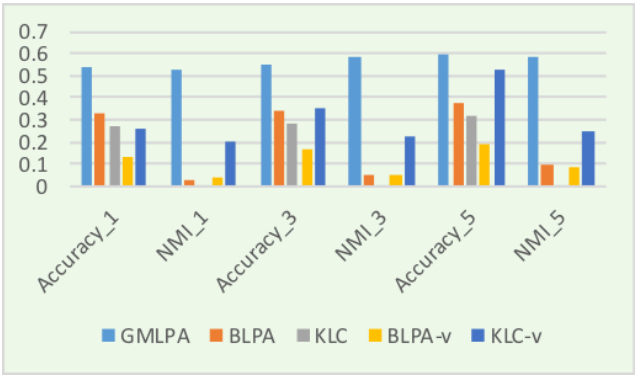

FIGURE 7. Clustering accuracy for ProgrammableWeb.

Three clustering tasks are designed for ProgrammableWeb Network. ProgrammableWeb contains 6160 mashups and 12769 APIs. For the ProgrammableWeb network, we use its subnetwork including 8 categories. They are Financial (1371), Food (331), Mapping (4343), Music (1060), Gambling (47), Sports (585), Travel (1044), and Weather (342).

(1) Fig.7: APIs (A) of the eight popular categories (Financial, Food, Mapping, Music, Gambling, Sports, Travel, and Weather) are clustered based on three paths: $A-D-M-D-A$, $A-F-A$ and $A-D-A$.

(2) Fig.8: APIs in the subcategories (booking, transportation, time) under the upper category "travel" are clustered based on the same paths.

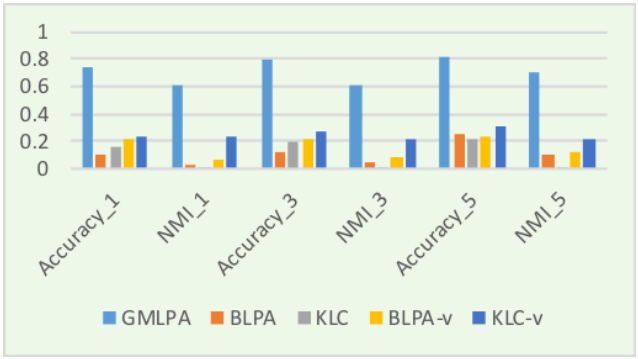

FIGURE 8. Clustering accuracy for ProgrammableWeb.

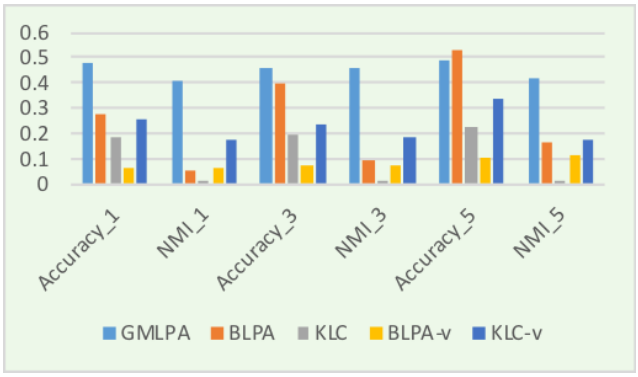

FIGURE 9. Clustering accuracy for ProgrammableWeb.

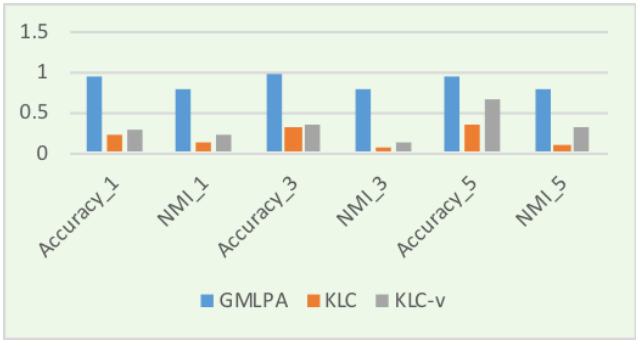

FIGURE 10. Clustering accuracy for service network.

(3) Fig.9: APIs in the subcategories (audio, streaming, lyrics) under the upper category ''music'' are clustered based on the same paths.

A real world network is constructed as the test set. Two clustering tasks are designed for this network.

(1) Fig.10: Three relation matrices for 200 objects are generated. All relation matrices have different clustering structure, and they are added some noise to test whether assigning low weights to low quality relation matrices by *GMLPA* can improve the clustering results.

(2) Fig.11: Three relation matrices for 200 objects are generated with different paths. For improving the clustering accuracy, we test whether *GMLPA* can assign low weights to relation matrices that are irrelevant to the domain knowledge.

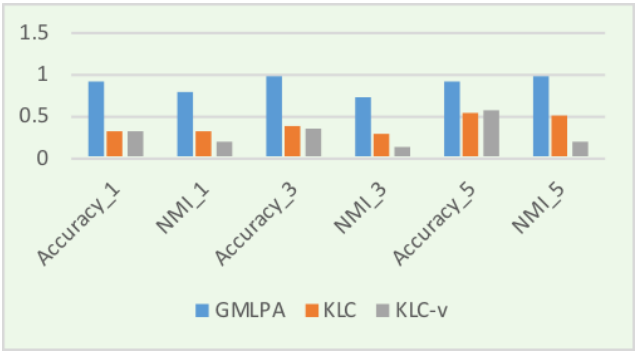

FIGURE 11. Clustering accuracy for service network.

B. ACCURACY EVALUATION

Comparing our algorithm with several methods, we study the effectiveness of *GMLPA* for different tasks.

Three methods are used, which did not consider the path selection problem. All paths are inputs of these algorithms. The first algorithm is a domain-knowledge and adaptive k-means algorithm. It is an information theoretic-based k-means clustering (KLC) by replacing Euclidean distance to KL-divergence, which is proposed in [24]. We compose all relation matrices into one single relation matrix for the input, and objects are high dimensional feature vectors.

The second algorithm is the basic label propagation algorithm (BLPA), which uses link relation to send labels to the rest of network. We compose all relation matrices into one single relation matrix for the input. We restrain our paths that have same start and end type, because BLPA is used for homogeneous networks. BLPA is a soft clustering method, i.e. it can discover overlapping communities.

The third algorithm is the ensemble method (like majority voting), which first uses different types of objects for clustering and then takes the label that is the majority on different paths. Either KLC or BLPA can be used as the ensemble method for clustering algorithms, thus we have the ensemble methods: KLC-v and BLPA-v.

We use two evaluation metrics to test the clustering results, and transform the soft clustering labels into hard cluster labels.

The first metric is *accuracy*. It is the percentage of target objects classified to the correct cluster.

The second metric is *normalized mutual information (NMI)*, which quantifies the difference between two random variables. The normalized mutual information of two discrete variables X and Y, which are vectors containing cluster labels for all the target objects is computed as:

$$NMI(X, Y) = \frac{I(X; Y)}{\sqrt{H(X)H(Y)}} \tag{2}$$

$$I(X; Y) = \sum\sum p(x, y) \log\left(\frac{p(x, y)}{p(x)p(y)}\right) \tag{3}$$

where I(X;Y) is the mutual information of two variables X and Y. The marginal entropies are H(X) and H(Y).

The marginal probability distribution functions of X and Y are p(x) and p(y) respectively. The joint probability distribution function of X and Y is p(x,y).

A higher value of both metrics shows a better clustering result, and the two metrics are in the range of 0 to 1.

Accuracy under different lengths of paths are tested, and the lengths of paths are denoted by Accuracy_L and NMI_L in above figures. The results are obtained through 20 runs.

We summarize the accuracy for all the aforementioned tasks in Fig.4 - Fig.11. The results show that *GMLPA* outperforms the rest of methods in most tasks. Other methods obtain relatively good clustering results for certain task, while *GMLPA* still gives good clustering results. We can conclude from these results that *GMLPA* is able to consistently obtain satisfied results from various tasks in the different networks.

V. APPLICATION OF OUR ALGORITHM

Our service mashup platform [25] is a standalone platform that connects API service providers and service users (including secondary developers and service users) [26], [27]. The platform is dedicated to providing users with the most comprehensive and convenient services, as well as helping service providers register services and increase their API invocation times. The platform has brought together more than 80 services required for application development [28]–[31].

VI. CONCLUSION

Clustering in heterogeneous information networks, especially in service mashup network, is an important task. Objects in heterogeneous networks connect each other through various relations. The relations are delegated by meaningful paths in our model. We construct paths with the domain-knowledge in service mashup networks. A parallel algorithm GMLPA is proposed, which is able to cluster varying types of communities in heterogeneous information networks. Graph-based computation enable to organize data naturally and to process data efficiently in parallel, which is the advantage of our algorithm. The experiments demonstrate that our algorithm produce stable and accurate results compared with other clustering methods. In addition, negative paths and weights automatic assigning by machine learning are future topics. There are two future directions for this work. On one hand, we will propose service recommendation mechanisms based on the novel algorithm, which is an indeed efficient and effective recommendation approach leveraging deep learning for trust-aware social recommendations in service community; on the other hand, we will try to propose methods and mechanisms to provide services in complex environment. For this direction, Deng et al. proposed an optimized service cache policy for IoT applications by taking advantage of the composability of services to improve the performance of service provision systems; and also proposed optimal and efficient service selection and composition algorithms in mobile environment, which are proved to be real useful and practical in real-world applications. These work really gave us good reference for our future direction of the cloud service community.

REFERENCES

[1] M. Gong, J. Liu, L. Ma, Q. Cai, and L. Jiao, "Novel heuristic density-based method for community detection in networks," *Phys. A, Stat. Mech. Appl.*, vol. 403, pp. 71–84, Jun. 2014.

[2] *Mashup Website*. Accessed: Sep. 16, 2019. [Online]. Available: http://www.programmableweb.com/

[3] M. Arab and M. Afsharchi, "Community detection in social networks using hybrid merging of sub-communities," *J. Netw. Comput. Appl.*, vol. 40, pp. 73–84, Apr. 2014.

[4] H. Papadakis, C. Panagiotakis, and P. Fragopoulou, "Distributed detection of communities in complex networks using synthetic coordinates," *J. Stat. Mech., Theory Exp.*, vol. 2014, no. 3, 2014, Art. no. P03013.

[5] X. Wang, L. Tang, H. Liu, and L. Wang, "Learning with multi-resolution overlapping communities," *Knowl. Inf. Syst.*, vol. 36, no. 2, pp. 517–535, 2013.

[6] X. Qi, W. Tang, Y. Wu, G. Guo, E. Fuller, and C.-Q. Zhang, "Optimal local community detection in social networks based on density drop of subgraphs," *Pattern Recognit. Lett.*, vol. 36, pp. 46–53, Jan. 2014.

[7] T. N. Dinh and M. T. Thai, "Community detection in scale-free networks: Approximation algorithms for maximizing modularity," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 6, pp. 997–1006, Jun. 2013.

[8] Z. Wang, X. Zhou, D. Zhang, D. Yang, and Z. Yu, "Cross-domain community detection in heterogeneous social networks," *Pers. Ubiquitous Comput.*, vol. 18, no. 2, pp. 369–383, 2014.

[9] Y. Zhou and L. Liu, "Social influence based clustering of heterogeneous information networks," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Chicago, IL, USA, 2013, pp. 338–346.

[10] T.-T. Kuo, R. Yan, Y.-Y. Huang, P.-H. Kung, and S.-D. Lin, "Unsupervised link prediction using aggregative statistics on heterogeneous social networks," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Chicago, IL, USA, 2013, pp. 775–783.

[11] L. Tang, X. Wang, and H. Liu, "Community detection via heterogeneous interaction analysis," *Data Mining Knowl. Discovery*, vol. 25, no. 1, pp. 1–33, 2012.

[12] P. M. Comar, P.-N. Tan, and A. K. Jain, "Simultaneous classification and community detection on heterogeneous network data," *Data Mining Knowl. Discovery*, vol. 25, no. 3, pp. 420–449, 2012.

[13] R. Angelova, G. Kasneci, and G. Weikum, "Graffiti: Graph-based classification in heterogeneous networks," *World Wide Web*, vol. 15, no. 2, pp. 139–170, 2012.

[14] Z. Lin, X. Zheng, N. Xin, and D. Chen, "CK-LPA: Efficient community detection algorithm based on label propagation with community kernel," *Phys. A, Stat. Mech. Appl.*, vol. 416, pp. 386–399, Dec. 2014.

[15] X. Liu and T. Murata, "Advanced modularity-specialized label propagation algorithm for detecting communities in networks," *Phys. A, Stat. Mech. Appl.*, vol. 389, no. 7, pp. 1493–1500, 2010.

[16] E. Gregori, L. Lenzini, and S. Mainardi, "Parallel K-clique community detection on large-scale networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 8, pp. 1651–1660, Aug. 2013.

[17] Z. Bu, C. Zhang, Z. Xia, and J. Wang, "A fast parallel modularity optimization algorithm (FPMQA) for community detection in online social network," *Knowl.-Based Syst.*, vol. 50, pp. 246–259, Sep. 2013.

[18] A. Prat-Pérez, D. Dominguez-Sal, and J.-L. Larriba-Pey, "High quality, scalable and parallel community detection for large real graphs," in *Proc. 23rd Int. Conf. World Wide Web*, Seoul, South Korea, 2014, pp. 225–236.

[19] H. Lu, M. Halappanavar, and A. Kalyanaraman, "Parallel heuristics for scalable community detection," in *Proc. IEEE 28th Int. Parallel Distrib. Process. Symp. Workshops*, Phoenix, AZ, USA, May 2014, pp. 1374–1385.

[20] J. Shi, W. Xue, W. Wang, Y. Zhang, B. Yang, and J. Li, "Scalable community detection in massive social networks using MapReduce," *IBM J. Res. Develop.*, vol. 57, no. 3, pp. 12:1–12:14, 2013.

[21] K. Punera and J. Ghosh, "Consensus-based ensembles of soft clusterings," *Appl. Artif. Intell.*, vol. 22, pp. 780–810, Sep. 2008.

[22] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica, "GraphX: Graph processing in a distributed dataflow framework," in *Proc. USENIX Conf. Oper. Syst. Design Implement. (OSDI)*, 2014, pp. 599–613.

[23] *Research Website*. Accessed: Sep. 16, 2019. [Online]. Available: http://dblp.uni-trier.de/
[24] S. Basu, A. Banerjee, and R. Mooney, "Semi-supervised clustering by seeding," in *Proc. 19th Int. Conf. Mach. Learn. (ICML)*, 2002, pp. 1–8.
[25] *Mashup Website We Built*. Accessed: Sep. 16, 2019. [Online]. Available: http://www.servicebigdata.cn
[26] Y. Duan, Z. Lu, Z. Zhou, X. Sun, and J. Wu, "Data privacy protection for edge computing of smart city in a DIKW architecture," *Eng. Appl. Artif. Intell.*, vol. 81, pp. 323–335, May 2019.
[27] Z. Song, Y. Duan, S. Wan, X. Sun, Q. Zou, H. Gao, and D. Zhu, "Processing optimization of typed resources with synchronized storage and computation adaptation in fog computing," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 3794175:1-3794175:13, May 2018.
[28] S. Deng, L. Huang, G. Xu, X. Wu, and Z. Wu, "On deep learning for trust-aware recommendations in social networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 5, pp. 1164–1177, May 2017.
[29] S. Deng, Z. Xiang, J. Yin, J. Taheri, and A. Y. Zomaya, "Composition-driven IoT service provisioning in distributed edges," *IEEE Access*, vol. 6, pp. 54258–54269, 2018.
[30] S. Deng, L. Huang, J. Taheri, J. Yin, M. Zhou, and A. Y. Zomaya, "Mobility-aware service composition in mobile communities," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 555–568, Mar. 2017.
[31] S. Deng, L. Huang, D. Hu, J. L. Zhao, and Z. Wu, "Mobility-enabled service selection for composite services," *IEEE Trans. Services Comput.*, vol. 9, no. 3, pp. 394–407, May/Jun. 2016.

YU LEI received the master's and Ph.D. degrees in computer science and technology from the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, in 2009 and 2014, respectively, under the guidance of M. Luoming, who is a Distinguished Chang Jiang Scholar. He is currently with the College of Computer Science, Inner Mongolia University, China, where he is also with the Inner Mongolia Engineering Lab of Cloud Computing and Service Software. His research interests include service computing, cloud technologies, and service-oriented applications.

PHILIP S. YU is a Fellow of the ACM and the IEEE. He received several IBM honors, including the two IBM Outstanding Innovation Awards, the Outstanding Technical Achievement Award, the two Research Division Awards, and the 94th Plateau of Invention Achievement Awards. He was an IBM Master Inventor. He received the Research Contributions Award from the IEEE International Conference on Data Mining, in 2003, and the IEEE Region 1 Award for "promoting and perpetuating numerous new electrical engineering concepts," in 1999. He is on the Steering Committee of the IEEE Conference on Data Mining and the ACM Conference on Information and Knowledge Management and was a member of the IEEE Data Engineering Steering Committee. In addition to serving as a Program Committee Member on various conferences, he was the Program Chair or Co-Chair of the 2009 IEEE International Conference on Service-Oriented Computing and Applications, the IEEE Workshop of Scalable Stream Processing Systems (SSPS'07), the IEEE Workshop on Mining Evolving and Streaming Data, in 2006, the 2006 joint conferences of the 8th IEEE Conference on E-Commerce Technology (CEC' 06) and the 3rd IEEE Conference on Enterprise Computing, E-Commerce and E-Services (EEE' 06), the 11th IEEE International Conference on Data Engineering, the 6th Pacific Area Conference on Knowledge Discovery and Data Mining, the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, the 2nd IEEE International Workshop on Research Issues on Data Engineering: Transaction and Query Processing, the PAKDD Workshop on Knowledge Discovery from Advanced Databases, and the 2nd IEEE International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems. He served as the General Chair or Co-Chair of the 2009 IEEE International Conference on Data Mining, the 2009 IEEE International Conference on Data Engineering, the 2006 ACM Conference on Information and Knowledge Management, the 1998 IEEE International Conference on Data Engineering, and the 2nd IEEE International Conference on Data Mining. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, from 2001 to 2004. He also served as an Associate Editor for the *ACM Transactions on the Internet Technology*, from 2000 to 2010, and *Knowledge and Information Systems*, from 1998 to 2004. He is the Editor-in-Chief of the *ACM Transactions on Knowledge Discovery from Data*.