Query From Sketch: A Common Subgraph Correspondence Mining Framework

Lingheng Zhu[1], Wu Liu[1]✉, Lingyang Chu[2], Peiye Liu[1], Xiaoyan Gu[3]✉

1. Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, China
2. School of Computing Science, Simon Fraser University, Vancouver V5A1S6, Canada
3. Information of Information Engineering, Chinese Academic of Science, Beijing 100093, China

{Zhulingheng0712, liuwu, liupeiye}@bupt.edu.cn, lca117@sfu.ca, guxiaoyan@iie.ac.cn

***Abstract*—We investigate a subgraph mining framework, that can connect similar entities according to their structure and attribute similarities. We take one mapping between two related points chosen from the query and target graph as one vertex in the correspondence graph and decide the weight of the edge based on the similarity score. In this way, we transform the problem to a dense subgraph discovery problem. To adapt this method to large scale, we choose the candidate group by some effective pruning methods. We also add some techniques to make our method more flexible to fit uncertain user sketched input. We investigate how changes to certain parameters in the algorithm can influence the results. By integrating all these adjustments into the framework, we can provide a method that exhibits both accuracy and flexibility in many situations with a degree of generality. Experiments on both certain and uncertain query graphs can give satisfactory and informative results.**

***Keywords*-data mining; graph search; pattern recognition; similarity search**

I. INTRODUCTION

Most current map searching services are based on precise keyword search methods, like Google Maps or Baidu Map. Although those keyword search methods can pinpoint targets according to their ID or address, they still retrieve several redundant results because of the duplicate keywords belonging to different targets. This flaw causes traditional keyword search services to require manual intervention for correct search results. Nevertheless, most of the time, users can provide extra information about the targets, which can be used to assist in searching.

Considering the plentiful extra information available, we construct a graph model regarding the target, and then convert the data mining question to a graph pattern match problem. In this way, we can utilize some computer vision approaches to solve the problems, providing better accuracy. Nevertheless, the extra uncertainty in this information also introduces several challenges: 1) due to the influence of uncertainty noise and other unpredictable factors, the target graph is hard to define; 2) query uncertainty occurs because a user may not know precisely what he wants and thus not care about some of the attributes and relations; and 3) the classic algorithm [3] is very time-consuming if we put every possible element into it. The first two challenges are also mentioned in some work on schemaless graph querying [2].

The graph query pattern match problem has been studied by many researchers for decades, and several algorithms have been proposed for obtaining better accuracy [1] and universality [2]. NeMa [1] converted the neighborhood of each node into a multi-dimensional vector and obtain a similarity value that includes both structure and label (we call it attributes) similarities. It is very accurate but cannot handle large scale data due to the lack of a good pruning method. Most graph pattern match methods divide the query and target, where as we use methods from the computer vision field [3] to define a correspondence graph based on both and convert the problem to a dense subgraph problem. This method constructs the adjacency matrix from the large correspondence graph and finds a dense subgraph by computing the *replicator equation* iteratively. We made some improvements based on [3]. There are also some works that try to address the user input uncertainty. A good learning method is proposed in [2] that can automatically adapt to the preferences of different users. Nevertheless，these works still lack the ability to handle large scale data in a relatively short time. In addition, the combination of keyword search and graph search will weaken the accuracy of the graph search algorithm.

In this paper, we transform this search problem to a common subgraph search problem, and proposed a robustness algorithm that can search for a real target based on real world entities (e.g., a street block on the map, a community in the social network) or abstract graph sketches [21], which contain various valuable information.

For the concrete procedure, we first define the vertexes and edges of the correspondence graph. A vertex represents the correspondence of one entity of the query graph and one entity of the target graph and an edge is the affinity score between two correspondences. After this definition, we choose candidates from all the possible correspondences (up to $m \times n$) and divide them into groups, which prunes the data to be processed and makes our method faster without reducing its effectiveness. Then, we calculate the affinity score of every pair of correspondences and generate the affinity matrix. Finally, we apply a common pattern searching algorithm to give the top k matches and arrange them using a confidence score. This method allows violations of strict isomorphism which can easily be affected by noise, and it supports a scaling factor which can be quite useful.

For a user sketched graph search, we first let the user set the attributes and relations he is interested in. Then, we use a consistent method for the ones the user does not set: we

978-1-5090-6549-3/17 $31.00 © 2017 IEEE
DOI 10.1109/BigMM.2017.80

automatically set them to the mean value of the others or a proportion of the maximum of the others chosen to obtain the most satisfactory results. To avoid same-source correspondences, which means that they contain the same entity in the query, we add the same-source penalty. Otherwise, we may not find that the main part of the results graph is similar to the query. We add these improvements to the original algorithm [3] to adapt it to user sketched graphs.

By redefining the structure and attribute similarity based on the real situation and continuing to use the properties that we find useful, we can develop a useful framework to adapt to other applications.

Our contributions can be summarized as follows:

1) We convert the traditional data mining problem to a graph pattern match problem, which offers more accuracy.
2) We propose several valuable properties for handling user uncertainty. Furthermore, we evaluate the influence of several parameters in the algorithm on the results.
3) Our framework is robust to different queries, including real world entities and user sketches. Moreover, the extra parameters can be changed for different search requirements.

II. Related Work

In the early years, graph searching was studied through isomorphic matching [14] and approximate matching [13, 15, 18. These methods are based on fixed functions and metrics. Then users based methods [2, 9, 20] was proposed that uses a machine learning method to generate results according to the habits of different users. In this stage, user uncertainty was mentioned but still not well studied. With increasing data volume, many algorithms have tried to solve this problem for large scale graphs. Thus, some heuristic methods [1, 2] have been proposed. There are other works [3, 16, 17] that translate the point-matching problem into a maximum clique problem. This method is not typically used in data mining but in computer vision.

The isomorphic matching method is well used in graph without noise, such as graphs constructed based on the molecular structures of proteins. In our problem, noise is generally present, which means we usually cannot find two isomorphic matching graphs.

The approximate method and heuristic method are similar except that the heuristic method tries to solve the problem in polynomial time [7, 8]. Because the NP-hard property and APX-hard property were proved in [1], many seek a heuristic method. Loopy belief propagation with a back tracking strategy [2] and NeMaInfer [1] are both good examples. However, the long time required and low tolerance for noise remain problems when analyzing big data. Furthermore, NeMaInfer lacks sufficient consideration of the connection between nodes resulting in low structure similarity.

The machine learning method in [2] addresses user diversity fairly well but dose not mention how to handle the information that the user does not care. Furthermore, large-scale data can add too much of a burden to the learning process.

One of our advantage compared to these method involves transforming the problem into the maximum clique problem. However, the maximum clique problem is NP-hard, and some methods [16, 17, 19] cannot solve it in polynomial time. Moreover, user-sketched graph search is not supported in these methods [3, 16, 17].

In general, the ability to handle large-scale data, noise tolerance and user sketch support are now important and under-researched qualities, and our framework has all three of these qualities.

III. Framework

A. Problem Formulation

Consider a given query graph $Q=(V_q,E_q,L_q)$ (containing the *query entity q* and neighbors) and a target graph $G=(V_g,E_g,L_g)$, where V_q and V_g are the vertex sets, E_q and E_g are the edge sets, L_q and L_g are the attribute sets. As shown in Fig. 1(a), given a vertex mapping $f:V_q\rightarrow V_g$ and a pair of vertexes (V_q,V_g) in query graph q, we have the *attribute similarity score* $sim_{L_i}(l_{q_i},l_{g_i})$, $sim_{L_j}(l_{q_j},l_{g_j})$ and the *structure similarity score* $sim_{S_{ij}}(s_{q_{ij}},ks_{g_{ij}})$, where s is the distance between two vertexes, and k is the scale factor. The *combined similarity score* for two maps is:

$$sim_f(i,j)=sim_{L_i}sim_{L_j}sim_{S_{ij}} \tag{1}$$

To find the matches of graph q in graph g, we need to find a mapping f such that:

$$\max_f \frac{1}{|V_q|(|V_q-1)|}\sum_{v_i\in V_q}\sum_{v_j\in V_q} sim_f(v_i,v_j) \tag{2}$$

$$s.t.\ \ v_i\neq v_j$$

We will explain this point after we define the *correspondence graph.*

B. Find Candidate Group

We assume that one of the attributes of the vertex is category, because in the specific problem we are addressing users that have a specific goal of finding a place in a certain category, such as a hospital, school, or estate.

First, in the target graph (we will call it G for short), we find all vertices with the same categories, forming the *same category group*. If the user-chosen query has multiple categories, then we will treat each equally. If the vertices in G have multiple categories, then it will be counted if any one of those categories matches the query. Then, we prune the group with the *attribute similarity score.* Next, we find the 50 nearest neighbors of each vertex we choose (the neighbors of the query are in the query graph). Then, we compare the neighbors of the set of vertices chosen from G with the query's neighbors and remove the vertex from the neighbor set if we cannot find at least one neighbor of the query with a matching category. This step is shown in Algorithm 1.

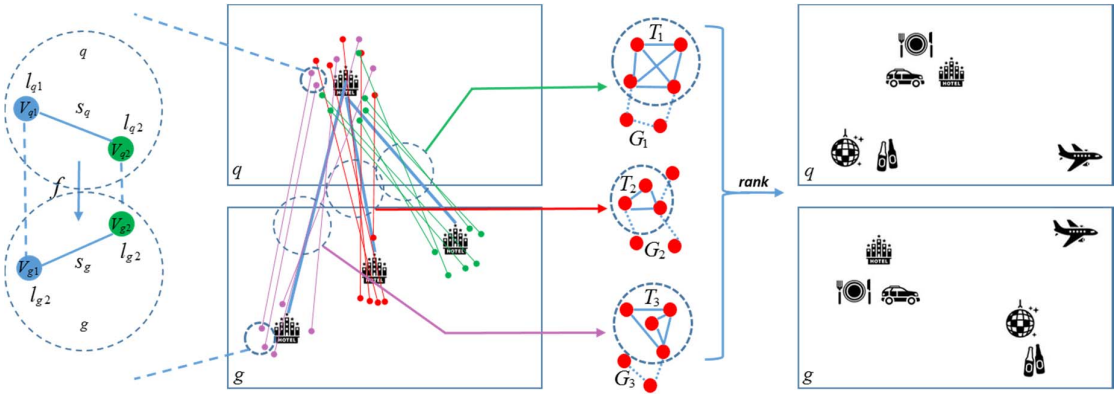


Fig. 1. Illustration of the main idea in our method. Find all candidate group shown in (b) by category and attribute values. Then with the definition in (a) we form the correspondence graph G shown in (c). The common pattern corresponds to the dense subgraph T of G. Then we rank all the dense subgraph by the average value of edge. Finally, we obtain the result shown in (d).

Algorithm 1: Find all candidate and form adjacency matrix A

Input: Real world query graph Q ;

1. **for** *each vertices of graph* Q **do**
   (a) Bind the *same category group* in G , and the first group is for *query entity q;*
   (b) Trench the group using *attribute similarity score*;

   end
2. **for** *each vertex* g *of first same category group* ***do***
   (a) Find the 50 nearest neighbors of the vertex;
   (b) Check if the neighbors are in the *same category group* of neighbors in Q , otherwise remove it;
   (c) Map each neighbor with the source of the *same category group* that it is in and gather all the maps with the first map of *q* and *g* to form one *candidate group;*
   (d) Calculate weighted adjacency matrix;

   end

Output: Weighted adjacency matrix set A.

Algorithm 2: Find the common pattern

Input: Weighted adjacency matrix set A

1. **for** *every group in C*
   (a) Build set $T = q$ and initialize $x(1)$ in T as, $x_i(1) = 1/|T|$, if $i \in T$, otherwise set $x_i(1) = 0$;
   (b) Use replicator equation (5) to find the Common pattern of the group;
   (c) Remove it if query entity q is not in the pattern;
2. Decide the density score by the final $f(x)$;
3. Sort all the common pattern by the density score;

Output: The result common pattern.

C. Common Pattern Discovery

We define every mapping of f as a vertex of *correspondence graph* D and the weight of the edge is defined as $w_{ij} = sim_f(i, j)$. The adjacency matrix of the *candidate graph* is A defined as follows:

$$A_{ij} = \begin{cases} 0, & i = j \\ w_{ij}, & i \neq j \end{cases} \tag{3}$$

This procedure is similar to the work in [3]. While T is a subgraph of D, according to (2), we now need to find T from D such that the *average affinity score* $sim_{av} = \frac{1}{n^2}\sum_{i\in T, j\in T} A(i, j)$ can be maximized. We define y as $y(i) = 1$ if $i \in T$ and zero otherwise. Thus, the *average affinity score* can be represented as a quadratic form $sim_{av} = \frac{1}{n^2} y^T A y$, and then we define $x = \frac{y}{n}$, so $sim_{av} = x^T A x$. Moreover, because $\sum_i y(i) = n$, $\sum_i x(i) = 1$. Equation (2) then is equal to:

$$\begin{aligned} &\text{maximize} \quad f(x) = x^T A x \\ &\text{subject to} \quad x \in \Delta \end{aligned} \tag{4}$$

where $\Delta = \{x \in R^m : x \geq 0 \text{ and } |x|_1 = 1\}$. This can be explained using the Motzkin-Straus theorem [5], which establishes a connection between the maximum of $f(x)$ and the maximum clique of the *correspondence graph,* while the maximum clique can give us a common visual pattern. Although this theorem is for un-weighted graphs, Pavan and Pelillo [6] generalize it to weighted graphs.

To solve (4), we can use the *replicator equation,* which is derived from evolutionary game theory [4]. The discrete-time version of the first-order replicator equation has the following form:

$$x_i(t+1) = x_i(t)\frac{(Ax(t))_i}{x(t)^T Ax(t)}, \; i = 1, \ldots, n \tag{5}$$

This equation has two properties according to [4] that suit our problem. First, it has closure for the set Δ , so x can remain in Δ . Second, when A is symmetric and nonnegative, $f(x) = x^T Ax$ is strictly increasing along any trajectory. Thus, the asymptotically stable point is also the local solution of (4).

We always use the entity we are querying to initialize

Algorithm 3: Find all candidate and form adjacency matrix A(user edition)

Input: Real world query graph Q ;

1. **for** *each vertices of graph* Q **do**
   (a) Bind the *same category group* in G , and the first group is for *query entity q;*
   (b) Trench the group using *attribute similarity score*;
   end
2. Find the maximum distance s of query graph;
3. **for** *each vertex* g *of first same category group* ***do***
   (a) Find all the neighbors that is nearer than $2\times s$.
   (b) Check if the neighbors are in the *same category group* of neighbors in Q , otherwise remove it;
   (c) Map each neighbor with the source of the *same category group* that it is in and gather all the maps with the first map of q and g to form one *candidate group;*
   (d) Calculate the mean of user set attributes similarity and set user don't-care attribute to the mean. Calculate the maximum of structure similarity and set user don't-care distance to $0.7\times\max$;
   (e) Calculate weighted adjacency matrix;
   end

Output: Weighted adjacency matrix set A.

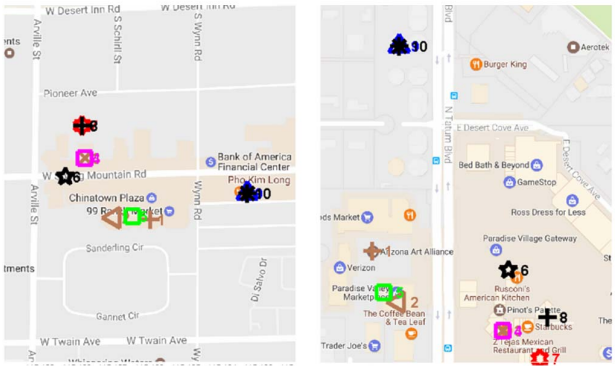
the algorithm, which we call the *initial query property* (similar to the Local property in [3]), because this approach can give the greatest possibility that our query entity occurs in the final results.

This step is shown in Algorithm 2.

D. Adaption to User Sketch

For application to user-sketched query graphs, some crucial changes are made to these algorithms. First, because users will give a concrete distance between two entities, it will be necessary to choose target entities with a distance closer than double the maximum of what the users set in the query. Considering that the user usually does not know precisely what he wants to query, he may leave some of the information blank, such as the attribute or the distance. Accordingly, we give this algorithm two additional properties:

1) *Fixed Part Property.* The blank information means that the user does not care much about its result. Therefore, we set a fixed value for this information, to prevent it from influencing the results.
2) *Same-Source Penalty Property.* According to our experiments, if we set the distance between two identical entities in the query graph to zero, because the target graph size is much bigger than the query, the algorithm may choose several same-source candidates as the result, which is not desirable. Therefore, we set a large enough value for the same-source distance to add a penalty.



		Name	(query)review	star	Name	(target)review	star
1	✚	Beaut Point Hair Studio	21	4.5	Amazing Lash Studio Tatum & Shea	40	4
2	◁	Sesame House	5	4	Pick Up Stix	5	4
3	✖	Jackpot Foot Massage	19	3.5	Elements Massage	36	3.5
4	□	Kapuso Kapamilya	40	3	Paradise Bakery	66	3
5	□	The Little Kitchen Cafe	15	3.5	Soma Cafe	13	3.5
6	★	Pretty Girls Maid Cafe	9	3	Panda Express	9	3
7	✿	Szechuan Express	40	4	My Fit Foods	41	4
8	✚	Szechuan Express	40	4	Dcberrito Mexican restaurant	42	4
9	△	Patrick Whittier-1st Priority Realty	15	5	Orion Mortgage Corporation	24	5
10	✱	Shane Nguyen-Top Agent-1st Priority Realty	23	5	Orion Mortgage Corporation	24	5

Fig. 2. Result of real-world entity query. Same marker and number refer to business of same category, thus a correspondence. For better viewing, please see the original color pdf file.

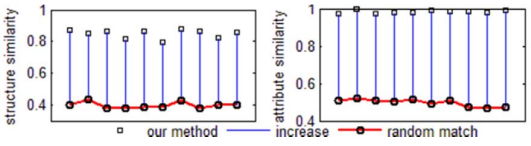


Fig. 3. Similarity score of 10 groups of results for 10 random chose real-world inputs.

For a user-sketched query graph, we change Algorithm 1 to Algorithm 3.

E. Time Complexity

Assuming that we already know the distance between every two entities, the two main procedures in terms of time consumed are 1) calculating the similarity and 2) iteration of the replicator equation. The time complexity for the former is $O(m^2)$ for one candidate group, where m is the size of the group. Because x is a sparse matrix and we do not need to calculate zeros in x, the time complexity for the latter is $O(kmd)$, where k is the iteration of equation (5), and d is the nonzero entry number.

IV. Experiments

We select the Yelp dataset to evaluate our algorithm because of its large-scale data, which include types of business locations in cities. Moreover, the data in Yelp is well categorized and attributed, which provides plentiful extra information for our method.

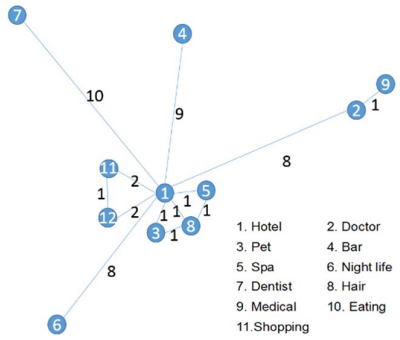


Fig. 4. User sketched query graph. The unit of distance is km.

	Name	(target)review	star	review	star(query)
1	Hampton Inn	40	4	50	4
2	Phoenix Neurological Associates	9	3	N/A	3
4	Breakroom Bar & Grill	55	4	N/A	N/A
5	Central Barber Shop	34	4	35	4
6	'4 Kings North	4	4	N/A	N/A
7	Desert Dentistry	11	5	N/A	N/A
8	Ide Mania	40	4.5	40	4
9	Mountain View Family Physicians	30	4	35	4
10	Whataburger	36	3	45	3
11	SAS Fabrics	44	4	40	4

Fig. 5. Top Match result of the query. The sequence number matches the same sequence number in Fig. 4.

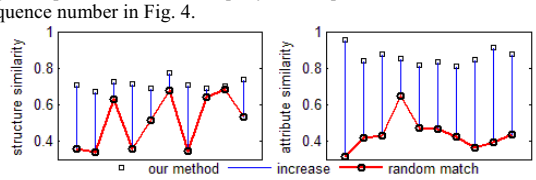
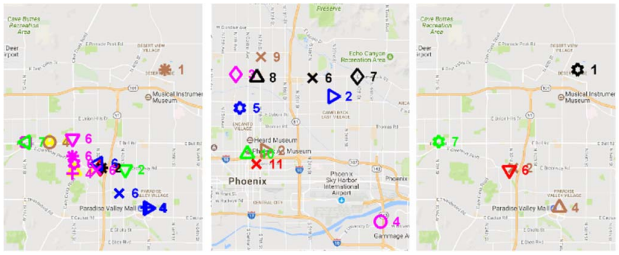


Fig. 6. Similarity score of 10 groups of results for 10 random sketches.



(a) (b) (c)

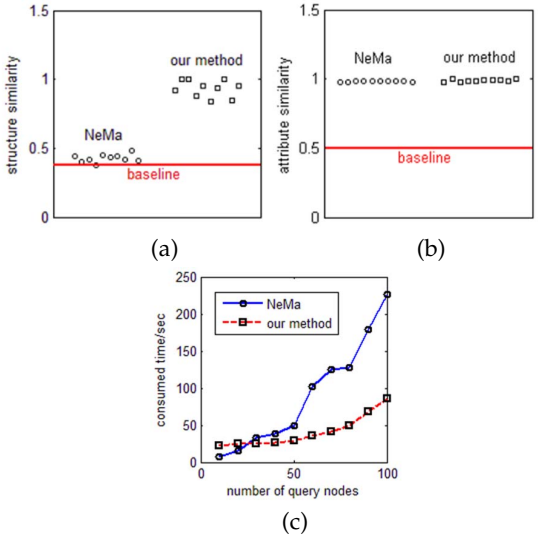
Fig. 7. Typical unsatisfying results causing by improper parameters.

A. *Real-World Query*

For a real-world query problem, we randomly select 10 businesses in Phoenix and then attempt to match similar businesses in Las Vegas with a similar geographical distribution.

Fig. 2 is one groups of results selected from among many. We can see considerable structure similarity of the common pattern. Businesses of the same category are at similar locations in both graphs. We can see this similarity by judging the distance between any two entities.

The value sim_s is the structure similarity for two correspondences between the query and target graph and changes inversely with the difference in distance. Thus, the



(a) (b) (c)

Fig. 8. (a) and (b) compare the similarity score of the results for 10 random chose real-world inputs generated from two methods. Baseline is got by computing the average score of 1000 random matches. (c) shows the time consuming property of two methods.

structure similarity of the pattern can be represented as:

$$\frac{\sum_{v_i, v_j \in Q} sim_{s_{ij}}(s_{q_{ij}}, s_{g_{ij}})}{n^2} \tag{6}$$

We can compare all groups of our results with totally random matches as shown in Fig. 3(a).

In the Yelp dataset, the most important attribute is stars which represent the quality of the business, and review count, which represents the degree of popularity. By computing the average of attribute similarity of every two nodes which is defined previously as $sim_L(l_q, l_g)$, we can get a similarity score for the two graphs. Similarly, we compare all groups of results with totally random matches in Fig. 3(b) to see the effect.

B. *User-Sketch Query*

We show an example of a user-sketched graph in Fig. 4. It can to some degree represent what some users want. For example, the user may be healthy and not need to see doctors frequently, so he may set a relatively long distance to doctors; and he may think that he will need to buy medicine after seeing a doctor, so he may set a pharmacy near a doctor.

We also search this graph in Phoenix. The top ranked result is shown in Fig. 5. We can see that we nearly satisfied the user's needs, except for finding a pet business. The reason we are missing one point is that the algorithm can exclude points to obtain a higher score, but this effect can be minimized by properly set the fixed value of structure similarity. We can still use (6) to determine the structure similarity but we must exclude those distance users did not set and we can get attribute similarity in similar way. We input 10 different graph (randomly generated), and the similarity is shown in Fig 6.

If we do not apply the *Same-Source Penalty Property,* the

result will be similar to Fig. 7(a). We can see that the algorithm may give us a result with many of the same entities.

If the fixed value of structure similarity is too high, the result will be similar to Fig. 7(b). The query entity is missing. In this situation, we give entities with few links to other entities and have many distances unset by the user too much priority, causing the query entity with the most links to others to be downgraded by the algorithm. This effect is displayed for all 10 groups of results.

If the fixed value of structure similarity is too low, then the average similarity score can be low, so entities can be difficult to include. Thus, entities relatively far from the query entity may overshadow others when the algorithm can choose from a large range of choices. The results for other sketch queries also show this effect.

C. Comparison with Other Algorithms

NeMa [1] is a fast graph search algorithm proposed in 2013, and it outperformed the classic graph search algorithms such as NESS [10], gStore [11], and Blinks [12].

As shown in Fig. 8(a), for structure similarity, this algorithm performs better compared to wholly random matches, but performs not as good as our method. This result occurs because NeMa has serious disadvantages in our map-search problem. It finds an optimal match of every node in the query graph based on the label similarity and neighborhood similarity without enough consideration of the connections between nodes. Although the memory technique can compensate for this problem to some extent, the results for our problem can still be unsatisfactory. As for label similarity, NeMa and our method can both generate satisfactory results as illustrated in Fig. 8(b).

NeMa is more time-consuming than our method as shown in Fig. 8(c). We can see that with the increase in query nodes, the time used by NeMa increases rapidly and exceeds the time consumed by our method at most sample points.

V. Conclusion

We have presented a general method to mine subgraph correspondence between two graphs. Instead of using all the data, we develop a way to find candidate groups. We have also studied how to manage user input uncertainty and noise in a target graph to generate better results by implementing certain useful properties. Our experiments show that our method can generate satisfactory results and is robust to outliers.

VI. Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (No. 61602049), and the CCF-Tencent Open Research Fund (AGR20160113).

References

[1] A. Khan, Y Wu, C.C. Aggarwal, and X Yan, “NeMa: fast graph search with label similarity,” *Proceedings of the Vldb Endowment,* vol. 6, no. 3, 2013, pp. 181-192.

[2] S Yang, Y Wu, H Sun, and X Yan, “Schemaless and Structureless Graph Querying,” *Proceedings of the Vldb Endowment,* vol. 7, no. 7, 2014, pp. 565-576.

[3] H. Liu, and S. Yan, “Common Visual Pattern Discovery via Spatially Coherent Correspondences,” *IEEE Conf.* CVPR 2010, 2010, pp. 1609-1616.

[4] J. Weibull, *Evolutionary Game Theory,* MIT press, 1997.

[5] Motzkin, T.S., and E.G. Straus, “Maxima for Graphs and a New Proof of a Theorem of Tur’an,” *Canndian Journal of Mathematics,* vol. 17, no. 17, 1965, pp. 533-540.

[6] M. Pavan, and M. Pelillo, “Dominant Sets and Pairwise Clustering,” *IEEE Transections on Pattern Analysis & Machine Intelligence,* vol. 29, no. 1, 2007, pp. 167-172.

[7] L.Y. Chu, S.Q. Jiang, S.H. Wang, Y.Y. Zhang, Q.M. Huang, “Robust Spatial Consistency Graph Model for Partial Duplicate Image Retrieval,” IEEE Trans. Multimedia, vol. 15, no. 8, 2013, pp. 1982-1996.

[8] L.Y. Chu, Y.Y. Zhang, G.R. Li, S.H. Wang, W.G. Zhang, Q.M. Huang, “Effective Multimodality Fusion Framework for Cross-Media Topic Detection,” IEEE Trans. Circuits Syst. Video Techn., vol. 26, no. 3, 2016, pp. 556-569.

[9] W. Liu, T. Mei, Y.D. Zhang, C. Che, J.B. Luo, "Multi-Task Deep Visual-Semantic Embedding for Video Thumbnail Selection", IEEE CVPR, 2015, pp. 3707-371

[10] A. Khan,X N. Li, X. Yan, Z. Guan, S. Chakraborty, and S. Tao, “Neighborhood Based Fast Graph Search in Large Networks,” *SIGMOD,* 2011, pp. 901-912.

[11] L. Zou, J. Mo, L. Chen, M.T. Zsu, and D. Zhao, “gStore: Answering SPARQL Queries via Subgraph Matching,” *Proceedings of the Vldb Endowment,* vol. 4, no. 8, 2011, pp. 482-493.

[12] H. He, H. Wang, J. Yang, and P.S. Yu, “BLINKS: Ranked Keyword Searches on Graphs,” *ACM SIGMOD International Conference on Management of Data*, 2007, pp.305-316.

[13] L.Y. Chu, S.Q. Jiang, Q.M. Huang, “Fast common visual pattern detection via radiate geometric model,” ICIP 2011, pp. 2465-2468.

[14] J. Perez, M. Arenas, and C. Gutierrez, “Semantics and Complexity of SPARQL,” *ACM Transection on Database Systems,* vol. 34, no. 3, 2006, pp. 16.

[15] J. Cheng, J.X. Yu, B. Ding, P.S. Yu, and H. Wang, “Fast Graph Pattern Matching,” *ICDE*, 2008, pp. 913-922

[16] R. Horaud, T. Skordas, “Stereo Correspondence Through Feature Grouping and Maximal Cliques,” *IEEE Transections on Pattern Analysis & Machine Intelligence,* vol. 11, no. 11, 1989, pp. 1168-1180.

[17] M. Pelillo, K. Siddiqi, and S.W. Zucker, “Matching Hierarchical Structures Using Association Graphs,” *IEEE Transections on Pattern Analysis & Machine Intelligence,* vol. 21, no. 11, 1999, pp. 1105-1120.

[18] W. Liu, T. Mei, Y.D. Zhang. "Instant Mobile Video Search with Layered Audio-Video Indexing and Progressive Transmission," IEEE Trans. on Multimedia, vol.16, no.8, pp.2242-2255, 2014.

[19] W. Liu, Y.D. Zhang, S. Tang, J.H. Tang, R.C. Hong and J.T. Li, "Accurate Estimation of Human Body Orientation From RGB-D Sensors," IEEE Trans. on Cybernetics, vol.43, no.5, pp.1442-1452.

[20] W. Liu, J.Y. Liu, X.Y. Gu, K. Liu, X.W. Dai, H.D. Ma, “Deep Learning Based Intelligent Basketball Arena with Energy Image,” MMM 2017, pp. 601-613.

[21] L.Y. Chu, Z.F. Wang, J. Pei, J.N. Wang, Z.J. Zhao, E.H. Chen: Finding Gangs in War from Signed Networks. KDD 2016: 1505-1514.