World Wide Web (2019) 22:751–770
https://doi.org/10.1007/s11280-017-0513-6

Context-aware graph pattern based top-k designated nodes finding in social graphs

Guanfeng Liu[1,2] · Qun Shi[1] · Kai Zheng[3] · Zhixu Li[1] · An Liu[1] · Jiajie Xu[1]

Received: 15 August 2017 / Revised: 19 October 2017 / Accepted: 14 November 2017 /
Published online: 24 March 2018

Abstract Graph Pattern Matching (GPM) plays a significant role in many real applications, where many applications often need to find Top-K matches of a specific node, (named as the designated node v_d) based on a pattern graph, rather than the entire set of matching. However, the existing GPM methods for matching the designated node v_d in social graphs do not consider the social contexts like the social relationships, the social trust and the social positions which commonly exist in real applications, like the experts recommendation in social graphs, leading to deliver low quality designated nodes. In this paper, we first propose the conText-Aware Graph pattern based Top-K designed nodes finding problem

This article belongs to the Topical Collection: *Special Issue on Deep vs. Shallow: Learning for Emerging Web-scale Data Computing and Applications*
Guest Editors: Jingkuan Song, Shuqiang Jiang, Elisa Ricci, and Zi Huangs

✉ Guanfeng Liu
gfliu@suda.edu.cn

Kai Zheng
zhengkai@uestc.edu.cn

Zhixu Li
zhixuli@suda.edu.cn

An Liu
anliu@suda.edu.cn

Jiajie Xu
xujj@suda.edu.cn

[1] School of Computer Science and Technology, Soochow University, 333 Ganjiang Road, Suzhou, Jiangsu, China

[2] Jiangsu Key Laboratory of Image and Video Understanding for Social Safety, Nanjing University of Science and Technology, Nanjing, 210094, People's Republic of China

[3] School of Computer Science and Engineering and Big Data Research Center, University of Electronic Science and Technology of China, 4 Jianshe North Rd 2nd Section, Chengdu, Sichuan, China

(TAG-K), which involves the NP-Complete Multiple Constrained GPM problem, and thus it is NP-Complete. To address the efficiency and effectiveness issues of TAG-K in large-scale social graphs, we propose two indices, MA-Tree and SSC-Index, which can help efficiently find the Top-K matching. Furthermore, we propose an approximation algorithm, A-TAG-K. Using real social network datasets, we experimentally verify that A-TAG-K outperforms the existing methods in both efficiency and effectiveness for solving the TAG-K problem.

Keywords Graph pattern matching · Social graph

1 Introduction

1.1 Background

Graph pattern matching (GPM) identifies matching subgraphs $M(G_P, G_D)$ in a data graph G_D for a given pattern graph G_P. It has become increasingly used in computer vision [4], chemical structure [23] and social networks [20]. GPM is typically defined in terms of subgraph isomorphism [30]. That is, a graph G_D is a match of G_P such that there exists a bijective function f from the nodes of G_P to the nodes of G_D, (a) for each node v in G_P, v and $f(v)$ have the same label, and (b) there exists an edge from v to v' in G_P if and only if $(f(v), f(v'))$ is an edge in G_D. This makes graph pattern matching NP-complete [15], which is often too restrictive to actual applications. Therefore, *Graph Simulation* [22] has been proposed with fewer restrictions. A data graph G_D is a match of G_P such that there exists a binary relation $R \subseteq V_q \times V$, (1) for each $(u, v) \in R$, u and v have the same label; and (2) for each edge (u, u') in G_P, there exists an edge (v, v') in G_D such that $(u', v') \in R$. In contrast to subgraph isomorphism, graph Simulation [22] has less restrictions but more capacity to extract more useful subgraphs with better efficiency.

Example 1 Consider G_D and G_P in Figure 1, where each node denotes a person, e.g., project manager (PM), database developer (DB) and programmer (PRG). Moreover, each edge indicates a supervision relationship, e.g., edge (PM_1, PRG_1) indicates that PM_1 supervises PRG_1. When graph pattern matching is defined in terms of subgraph

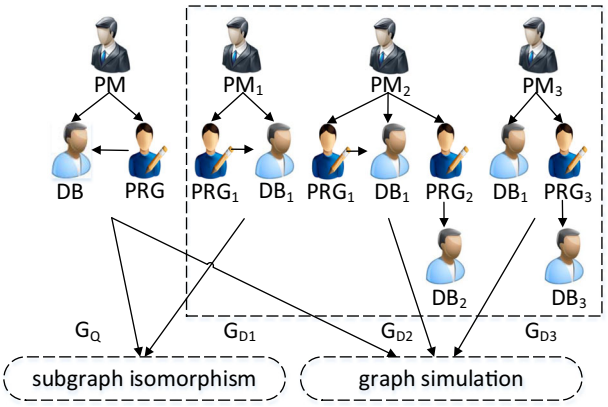

Figure 1 Data graph G_D and Pattern graph G_P

isomorphism, G_D1 is a matching of G_P. However, when graph pattern matching is defined in terms of graph simulation, G_D1, G_D2 and G_D3 are the matchings of G_P.

Graph simulation aims to extract all matching subgraphs $M(G_P, G_D)$ from G_D. However, many applications often need to find Top-K matches of a designated v_d in terms of a given G_P, rather than the entire set matching. Such as expert recommendation [21, 24] and egocentric search [5].

Example 2 Recall G_D and G_P in Figure 1. Suppose a company issues a graph search query to find the matches of PM in G_D based on the pattern G_P, where PM needs to supervise both a DB and a PRG, and moreover, the DB needs to work under the supervision of a PRG. The requirement for the matches of PM is expressed as a graph pattern G_P. When a graph pattern matching is defined in terms of graph simulation in G_D, the corresponding designated node can be identified, like PM_1, PM_2, and PM_3 in G_D1, G_D2 and G_D3.

In such a query introduced in the above example, It is unnecessary and too costly to compute the entire matching set $M(G_P, G_D)$. Thus Top-K GPM based on the graph simulation (TopKP) is proposed [10], where given a pattern graph G_P, a data Graph G_D and a designed node v_d in G_P, it is to find Top-K matching nodes of v_d ranked by a quality function in G_D.

1.2 Problem and challenges

As shown in [17], many applications, for example, crowd-sourcing travel [19] and social network based e-commerce [15], people are willing to incorporate social contexts like the social relationships, the social trust and the social positions in v_d finding, which have significant influence on people's collaborations and decision making [14]. For example, the experts recommendation in social graphs, people prefer to find an expert who has intimate relationships with members of the team led by the expert.

However, the existing Top-K designated nodes finding method, like TopKP [10] only takes the number of nodes included in a matching into consideration, where the more the nodes in a matching, the better the quality of v_d. This strategy in designated nodes finding neglects the influence of social contexts, and thus can hardly find the high quality v_d in social graphs.

Example 3 Consider G_D and G_P in Figure 2, suppose PM_3 is the father of DB_1, and PM_2 is the classmate of DB_1. TopKP [10] will regard PM_2 as the best matching of the designated node, PM, in G_P, as the corresponding matching g_2 has 4 nodes except PM_2. However, the relationship between PM_3 and DB_1 is more intimate than that between PM_2 and DB_1. Thus, PM_2 may not be the best one in designated node finding.

This example inspires a new type of conText-Aware Graph pattern based Top-K designed nodes finding (TAG-K) problem, where we need to consider social contexts, i.e., social trust, social relationships and the impact of social position. Based on the theories in *Social Science* [1], these social contexts have significant influence on peoples decision making. In real applications, people are willing to incorporate the requirements of these social contexts in graph pattern matching, forming the Multi-Constrained Simulation (MCS) [17]. GPM in terms of MCS is NP-Complete as it subsumes the classical NP-Complete multi-constrained path selection problem [15]. Thus, TAG-K aims to find the Top-K designated nodes in terms of MCS, which brings the challenges of the efficiency and effectiveness issues.

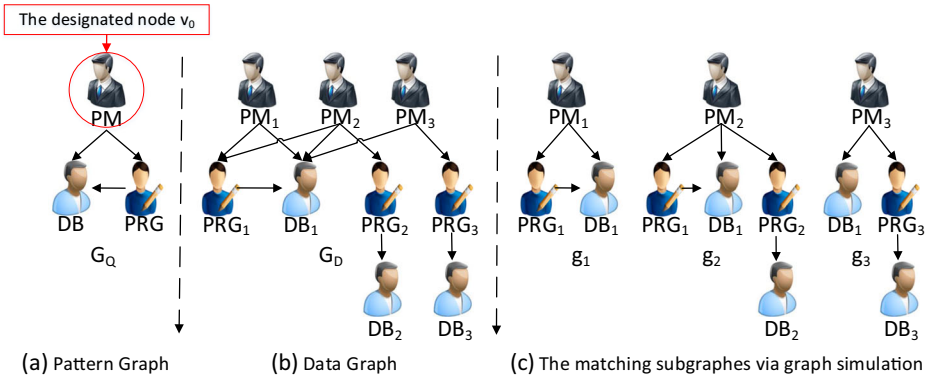

Figure 2 Designated node finding in terms of graph simulation

In this paper, we propose a TAG-K problem, where given a pattern graph G_P, which has multiple constraints on social contexts, a data graph G_D, which contains social contexts, and a designated node v_d in G_P, it is to find Top-K matches of v_d included in the graph pattern matching results, M(G_P, G_D) in terms of MCS. TAG-K covers the multi-constrained GPM problem, which is NP-Complete [17]. So, the main challenge of our work is to develop an efficient and effective approximation method to support the TAG-K query. Our contributions are summarized as below.

- We first propose a TAG-K problem, where, in the graph pattern based designated node finding, TAG-K considers the constraints of social contexts, like social trust, social relationships and the impact of social position.
- We then propose a Multi-Attribute Tree (MA-Tree) index to record the labels, outdegree and indegree of nodes in G_D, which can get candidates of v_d efficiently. Moreover, we propose an SSC-Index, which records more details of the decedents of a node in a *Strong Social Component*, where the nodes have strong social relation and highly impact of social position, pruning unpromising nodes effectively.
- We develop an efficient and effective algorithm, called A-TAG-K, which incorporates MA-Tree and SSC-Index. A-TAG-K can deliver a set containing Top-K designated nodes with $O(E_P N_D^2 + E_P N_D)$ time complex, where N_D is the number of nodes in G_D and E_P is the number of edges in G_P.
- We conduct experiments onto five real social network datasets, and the experimental results demonstrate our A-TAG-K greatly outperforms the existing methods in both efficiency and effectiveness.

The rest of this paper is structured as follows. In Section 2, we present related work about Top-K GPM problem. Section 3 is the preliminary of our work. Section 4 presents two indices, MA-Tree and SSC-Index. Section 5 presents the A-TAG-K algorithm. Experimental results are described in Section 6. Finally, we conclude the paper in Section 7.

2 Related work

Top-K GPM problem has already been widely studied in the literature, which can be classified into (1) the isomorphism-based Top-K GPM. (2) the simulation-based Top-K GPM. Below we analyze them in detail.

Isomorphism-based Top-K GPM This type of Top-K GPM is based on the subgraph isomorphism [30]. Tian et al. [29] propose the concept of approximate subgraph matching, which allows node mismatches and node/edge insertions and deletions. For coping with approximate subgraph matching problem, an index-based method is presented in [29], called TALE. In addition, Ding et al. [7] define the matching similarity between a data graph and a query graph to order results. Built on NH-index in [29], Ding et al. [7] employ the index to prune unpromising candidate nodes for each query node. Furthermore, Zhu et al. [35] consider the entire structure matching rather than substructure matching, and propose an algorithm to respond to the Top-K graph similarity query using two distance lower bounds with different computational costs. By using some typical hashing methods [25, 27, 31, 34], we can improve the efficiency of GPM. But sometimes it suffers the low effectiveness due to missing some important features of original data graphs.

Simulation-based Top-K GPM Existing isomorphism-based Top-K GPM methods still too strict to be used in some applications, e.g., finding social experts [11] and project organization [10]. Based on graph simulation [22], Fan et al. [10] propose a novel Top-K graph pattern matching method supporting a designated pattern node v_d, which can find the Top-K matches of v_d without computing the entire graph matching results. In addition, Chang et al. [3] study the problem of Top-K tree pattern matching, where the edges in the tree are mapped to the shortest paths in G connecting the corresponding nodes. A novel and optimal enumeration paradigm [3] has been presented, which is based on the principle of Lawler's procedure. Furthermore, Gao et. al. [26] propose a graph learning method based on the graph simulations, where multiple features of a graph are considered.

The existing Top-K GPM methods do not consider the social contexts in GPM in social graphs. As indicated in [17], such queries are common in social network based applications, like crowd-sourcing travel [19] and social network based e-commerce [15], which motives us to develop a new type of context-aware graph pattern based designated nodes finding method.

3 Preliminaries

In this section, we first introduce the Multi-Constrained Simulation (MCS) [15], and then we define the TAG-K problem, and propose the ranking function in Top-K nodes finding.

3.1 Multi-constrained simulation (MCS)

Given a data graph G_D, and a pattern graph G_P, MCS [15] provides conditions, which subgraph must satisfy, if subgraph matches the pattern graph.

Data graph A data graph is a Contextual Social Graph (CSG) [15], which is a labeled directed graph $G = (V, E, LV, LE)$, where

- V is a set of vertices;
- E is a set of edges, and $(u_i, u_j) \in E$ denotes a directed edge from vertex u_i to vertex u_j;
- LV is a function defined on V such that for each vertex u in V, $LV(u)$ is a set of labels for v. Intuitively, the vertex labels may for example represent social roles in a specific domain;
- LE is a function defined on E such that for each edge (u_i, u_j) in E, $LE(u_i, u_j)$ is a set of labels for (u_i, u_j), like social relationships and social trust in a specific domain.

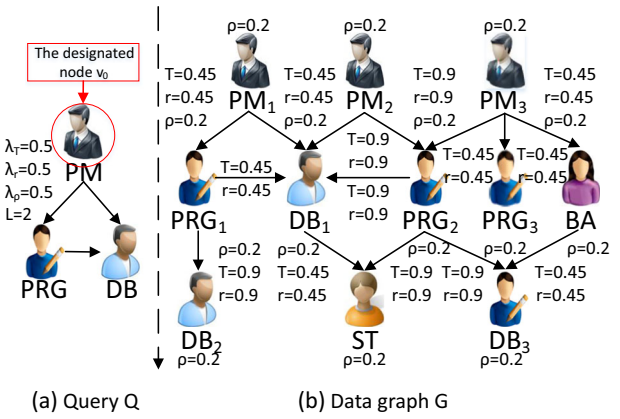

Figure 3 Data graph G_D and pattern graph G_P

Example 4 As shown in the data graph in Figure 3b, where each vertex is associated with a *role impact factor*, denoted as $\rho \in [0, 1]$, to illustrate the impact of a participant in a specific domain, which is determined by the expertise of the participant. $\rho = 1$ indicates that the people is a domain expert while $\rho = 0$ indicates that the people has no knowledge in that domain. Moreover, each edge is associated with *social trust*, denoted as $T \in [0, 1]$, and *social intimacy degree*, denoted as $r \in [0, 1]$, to illustrate trust and intimacy social relationships between participants. T, r and ρ are called *social impact factors*, whose values can be extracted by using the data mining techniques [13, 15, 16, 18, 33]. For example, in academic social networks formed by large databases of Computer Science literature (e.g., DBLP[1] or ACM Digital Library[2]), the social relationships between two scholars (e.g., co-authors, a supervisor and his/her students) and the role of scholars (e.g., a professor in the field of data mining) can be mined from publications or their homepages. The social intimacy degree and role impact factor values can be calculated as an example by applying the PageRank model [28].

Based on theories in *Social Psychology* [1], we adopt the multiplication method to aggregate T and r values of a path, and adopt the average method to aggregate the ρ values of the vertices in a path. The details of the aggregation method have been discussed in [15]. The aggregated values of a path p is denoted as $A_S(p) =< A_T(p), A_r(p), A_\rho(p) >$. If each of the aggregated social impact factor value of p is greater than the corresponding one of path p', then p *dominates* p', which is denoted as $p \propto p'$.

Pattern graph A pattern graph is defined as $G_P = (V_p, E_p, f_v, f_e, s_e)$. (1) V_p and E_p are the set of vertices and the set of directed edges, respectively; $f_v(v)$ is the node label of v; (2) $f_e(v_i, v_j)$ is the bounded length of (v_i, v_j), represented by L; (3) $s_e(v_i, v_j)$ is the multiple constraints of the aggregated social impact factor values of (v_i, v_j) represented by λ_T, λ_r and λ_ρ, which are in the scope of [0, 1];

[1] http://www.informatik.uni-trier.de/ley/db/

[2] http://portal.acm.org/

Example 5 As showed in the pattern graph in Figure 3a, where the multiple constraints, i.e., $\lambda_T, \lambda_r, \lambda_\rho$ and L are given to edge (PM, PRG) which must be satisfied in the graph pattern matching.

GPM based on multi-constrained simulation (MCS) G_D matches G_P via MCS, if there exists a binary relation $S \in V_q \times V$ such that (1) for all $v \in V_q$, there exists a node $v \in V$ such that $(v, u) \in S$; (2) for each edge (v, v') in E_q, there exist nonempty paths from u to u' in G, and $length(u, u') \leq L$, if $f_e(v, v') = L$; (2) $T(u, u') > \lambda_T, r(u, u') > \lambda_r$ and $\rho(u, u') > \lambda_\rho$, if $s_e(v, v') = \{\lambda_T, \lambda_r, \lambda_\rho\}$.

If G_D matches G_P, then there exists a unique maximum relation $M(G_P, G_D)$. If G_D does not match G_P, $M(G_P, G_D)$ is the empty set. This maximum relation $M(G_P, G_D)$ is referred to as the set of matches of G_P in G_D. The relation $M(G_P, G_D)$ can be depicted as the set of matches of G_P in G_D.

Example 6 Figure 4 displays two matching subgraphs in terms of MCS based on G_P and G_D in Figure 3, where the multiple constraints, i.e., $\lambda_T, \lambda_r, \lambda_\rho$ and L are satisfied in the two graph pattern matchings.

3.2 The matches of the designated node v_d

Given a pattern graph G_P, a data graph G_D, and a designated node v_d, If the nodes in G_D can match the designated node v_d, matching subgraphs in G_D containing these nodes must match G_P based on MCS. We denote the matches of v_d in G_D based on G_P as $M_u(v_d, G_P, G_D)$, and denote the matching of u_i to v_d as $u_i \cong v_d$.

Example 7 As shown in Figure 4c, the two matching subgraphs can match G_P in Figure 3 in terms of MCS. Thus PM_2 and PM_3 are two matching nodes of PM. Namely, $M_{PM_2}(PM, G_P, G_D) = \{DB_1, PRG_2\}$, $M_{PM_3}(PM, G_P, G_D) = \{PRG_2, BA, DB_3\}$ and $PM_2 \cong PM$ and $PM_3 \cong PM$.

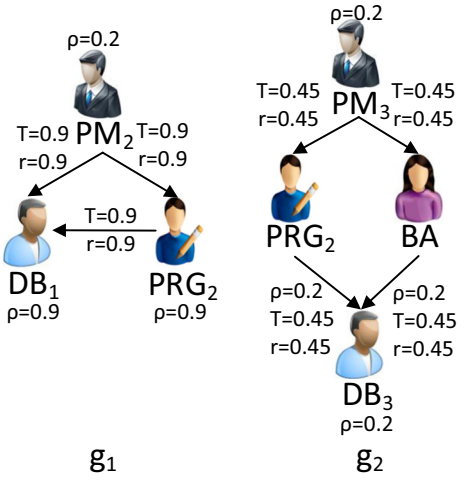

Figure 4 The matching subgraphs of G_P in terms of MCS

3.3 Ranking of matching

As the number of nodes included in the matching result, and the social impact factor values have significant influence on the quality of the pattern matching [10, 15], in order to rank the delivered pattern matching results to identify Top-K designed nodes, we propose the below ranking functions.

The relevant set for each descendant v' of v_d in G_P, the relevant set, $R_{(u,v_d)}$, includes all matches u' of v', such that if v_d reaches v' via a path $(v_d, v_1, ..., v_n, v')$, then u reaches u' via $(u, u_1, ..., u_n, u')$, where $(u_i, u_j) \in M(G_P, G_D)$.

That is, $R_{(u,v_d)}$ includes all nodes of the matching in $M_u(v_d, G_P, G_D)$. The larger the $R_{(u,v_d)}$, the better the matching [12] .

Example 8 Base on Figure 3, $R_{(PM_1,PM)} = 3$, $R_{(PM_2,PM)} = R_{(PM_3,PM)} = 2$.

Based on the utility function in (1) [15], we propose the new utility function as (2) that considers the average impact of social contexts of each matching path from u_i to u_j in the pattern matching, $M_u(v_d, G_P, G_D)$.

Utility function

$$U(p) = w_T * A_T(p) + w_r * A_r(p) + w_\rho * A_\rho(p) \tag{1}$$

where w_T, w_r and w_ρ are the weights of T_p, r_p and ρ_p respectively; $0 < w_T, w_r, w_\rho < 1$ and $w_T + w_r + w_\rho = 1$.

The value of these weights can be specified by users to illustrate their different requirements in different applications. For example, in crowdsourcing travel, a user could give a high value to ω_t to illustrate the concern about the social relationship between two people, while in employment, a user could give a high value to ω_ρ to illustrate the concern about the social impact of a people.

$$\delta(u, v_d) = \frac{\sum_{u_i \in R_{(u,v_d)}, u_j \in R_{(u,v_d)}} U(p(u_i, u_j))}{N} \tag{2}$$

where N is the number of matching path in $M_u(v_d, G_P, G_D)$.

Ranking function Based on the *relevant set* and the *utility function*, we propose the ranking function $RF(u, v_d)$, on a match u of v_d as a bi-criteria objective function in (3), to capture the influences of both the number of matching nodes and the social contexts.

The range of value of relevance function is in the scope of $[0, N_D]$ (N_D is the number of nodes in G_D), the range of value of the utility function is in the scope of [0, 1]. Thus, we need to normalize $RF(u, v_d)$ to the scope of [0, 1]. In the literature, there are some methods for normalization, such as the log function [6], the min-max normalization [6] and $arctan()$ function [6]. As the range of value of the revelent function is in the scope of $[0, N_D]$, the log function [6] and the min-max normalization [6] cannot be applied to the normalization. For $x \in [0, N_D]$, the value of $arctan(x)$ is in the scope of $[0, \pi/2]$. Thus, we use $arctan()$ function to normalize $RF(u, v_d)$, into the scope of [0, 1]. Here, α is used to adjust the weight between the relevance function and the utility function.

$$RF(u, v_d) = \alpha * arctan(R(u, v_d) * 2/\pi) + (1 - \alpha) * \delta(u, v_d) \tag{3}$$

3.4 TAG-K problem

Based on the ranking function $RF(u, v_d)$, we propose the context-Aware Graph pattern based Top-K designated nodes finding problem (TAG-K for short). Given a contextual social graph as the data graph G_D, a pattern graph G_P, a designated node v_d in G_P, a positive integer K, TAG-K is to identify Top-K matches M_u^K ($M^K \subseteq M_u(v_d, G_P, G_D)$), such that

$$\underset{M_u^K \subseteq M_u(v_d, G_P, G_D)}{argmax} \sum_{u_i \in M_u^K} RF(u_i, v_d) \tag{4}$$

That is TAG-K is to identify a set of K matches of v_d that maximizes the ranking function $RF(M_u^K)$. Namely, $\forall M^* \subseteq M(v_d, G_P, G_D)$, if $|M^*| = K$, then $RF(M_u^K) \geqslant RF(M^*)$.

Example 9 In Figure 3, suppose there is a requirement of finding the best match of the designated node PM, and the revelent set and the utility function have the same weight in the ranking function.Thus, based on the social impact factor values in G_D and (3), we can get $RF(PM_2, PM) = 0.69$, $RF(PM_3, PM) = 0.57$, $RF(PM_1, PM) = 0.55$. Thus, PM_2 is the best matching of PM via G_P in Figure 3.

4 Indexing structure

To improve the efficiency of TAG-K finding, we propose two index structures, i.e., *Multi-Attributes Tree* index, (MA-Tree), and *Strong Social Component* index, (SSC-Index), to record the labels, indegree, outdegree, the shortest path and the aggregated social impact factor values, which can help efficiently find the candidates of v_d in G_D.

4.1 MA-Tree

4.1.1 The purpose of the index

Based on the GPM in term of the MCS, if $u \cong v_d$, (1) the label of u must be the same as v_d, (2) the outdegree/indegree of u must be greater than 0, if the outdegree and/or the indegree of v_d is/are greater than 0. Thus, in order to investigate the potential candidates of v_d in G_D, based on B^+ tree, we propose a Multi-Attributes Tree (MA-Tree) index to record the multiple attributes, including label, indegree and outdegree of each node in G_D.

4.1.2 Structure

In MA-Tree, (1) each leaf node contains a pointer, which points to an array that saves the nodes with the same label, indegree and outdegree, and (2) for each non-leaf node, MA-Tree records a tuple, including (category number, indegree, outdegree), where each category is coded by using a digit as the category number, and the tree structure is established based the values of the category number, the indegree and the outdegree of each node respectively. Similar to B^+ tree, MA-Tree index can effectively reduce the search space when finding the candidate of v_d, and thus can improve the efficiency of TAG-K designated node finding.

Example 10 Figure 5 is a MA-Tree of G_D in Figure 3, where the digits, 0, 1, 2, 3 and 4 are given to represent the five categories, PM, DB, PRG, BA and ST, respectively. As the indegree of PM_1 is 0, and the outdegree of PM_1 is 2. Then, the tuple (0,0,2) is inserted

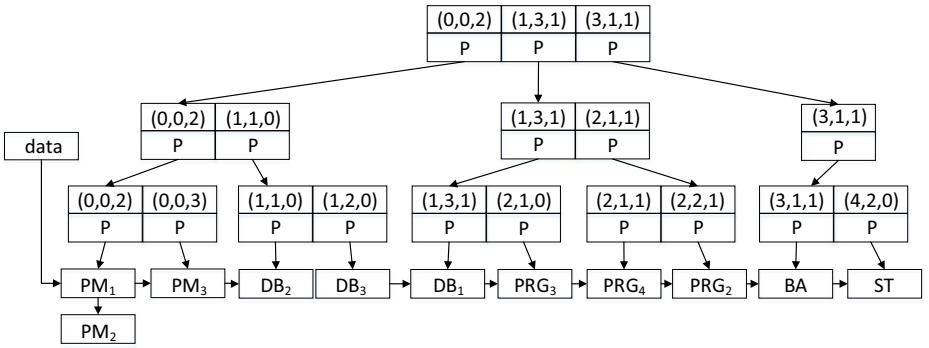

Figure 5 The MA-Tree of G_D in Figure 3

into MA-Tree. Based on the comparison of the category numbers, indegree and outdegree, we can establish the corresponding MA-Tree of G_D shown in Figure 5.

4.1.3 The searching process

Based on MA-Tree structure, we can fast investigate if a node is a candidate of v_d by searching MA-Tree from root nodes to leaf nodes. The time complexity of searching MA-Tree is the same as B^+ tree, i.e., $O(log_b N_D)$, where N_D is the number of of nodes in G_d and b is the number of children nodes at each level in MA-Tree.

4.2 Strong social component index (SSC-Index)

In addition to the information indexed by MA-Tree, we need to investigate if the decedents of the candidate can be a match of G_P. Thus, we first propose a concept of the strong social component where the nodes and edges have large social impact factor values, and thus have high probability to satisfy the social context constraints in G_P, then we build up the SSC-Index to record the labels, the shortest path length, and the aggregated social impact factor values of the decedents of each node in an SSC, which can be utilized to prune unpromising nodes.

In graph theory [2], a graph G is said to be strongly connected if every vertex is reachable from every other vertex, and a strongly connected component of a directed graph G is a subgraph that is strongly connected. Based on the definition of the strong connection, we give the definition of a *Strong Social Component* as follows.

Definition 2 Strong Social Component Given a CSG $< V, E, LV, LE >$, and two parameters λ_V and λ_E with $0 \leqslant \lambda_v \leqslant 1$ and $0 \leqslant \lambda_E \leqslant 1$, the subgraph induced by a subset of node set $V' \in V$ and edge set $E' \in E$ is an SSC if, and only if the following two conditions hold:

- $\forall v \in V', LV(v) \geqslant \lambda_V$
- $\forall e \in E', LE(e) \geqslant \lambda_E$

where $E' = E \bigcap (V' \times V')$.

In a CSG, a subgraph is said to be *socially strongly connected* if each vertex associated with a high role impact factor value in a specific domain is connected with the edges associated with intimate social relationships and strong social trust relationships. A *Strong Social Component* (SSC) is a subgraph that is socially strongly connected.

Based on the theories in *Social Psychology* [1], in an SSC, the social structure and the social contexts, including the social trust and social relationships on edges, and the social roles associated with vertices usually stay stable in a very long period of time. This property makes it realistic to index and compress the graph in an SSC with a low update cost. Identifying all the SSC in a specific domain subsumes the classical NP-Complete maximum clique problem [2], which is very time consuming. Alternatively, we can identify up to K SSCs by randomly selecting K vertices that are associated with high role impact factor values as the seeds. Then from each of the seeds, our algorithm adopts Breadth-First Search (BFS) method to find the vertices associated with high role impact factor values connected by the edges associated with high social intimacy degrees and social trust values. In the worst case, our method needs to visit all the vertices and edges in a data graph. The time complexity of the SSC identification is $O(N_D E_D)$, where N_D and E_D are the number of nodes and edges in G_D.

4.2.1 The purpose of the index

Based on MCS, if $u \cong v_d$, (1) For each descendant of v' of v_d, u has a descendant, u', which has the same label as v'. (2) For each edge, (v_d, v'), there exists a path $p(u, u')$, satisfying the constraints of social contexts associated with (v_d, v'). Thus, we build up the SSC-Index to record the labels, the shortest path length, and the aggregated social impact factor values of the decedents of each node in an SSC.

4.2.2 Index structure

Reachability index This index records a list of vertices that one can research another in a graph, where the index of each vertex contains the ancestors and predecessors of the vertex. As the size of SSC is usually much less than the whole data graph [1], building the reachability index is not computationally expensive [32].

Example 11 Figure 6 is an example of our index for the SSC of the graph depicted in Figure 6. From the figure, we can see the indices of each vertex include three parts: they are the reachability index, graph pattern index and social context index. We take vertex E as an example, as it has both ancestors and descendants. The reachability index of E records its ancestor C (i.e., Anc.: C), and its descendant H (i.e., Des.: H). Similarly, we construct the reachability index for each of the other vertices of the graph.

Given a reachability query in G_P, if the candidate nodes are included in the SSC, we can investigate the reachability immediately, greatly saving query processing time.

Graph pattern index After indexing the reachability information, we further index the graph pattern information to improve the efficiency of graph pattern matching in designated node finding. This index records the shortest path length between any two nodes in the graph of an SSC.

Example 12 Consider the graph pattern index shown in Figure 6. For vertex E, in addition to indexing the reachability information, the graph pattern index records the shortest path length from its ancestor C to E (i.e., $Slen = 1$), and from E to its descendant H (i.e., $Slen = 1$). Similarly, we construct the graph pattern index for each of the other nodes.

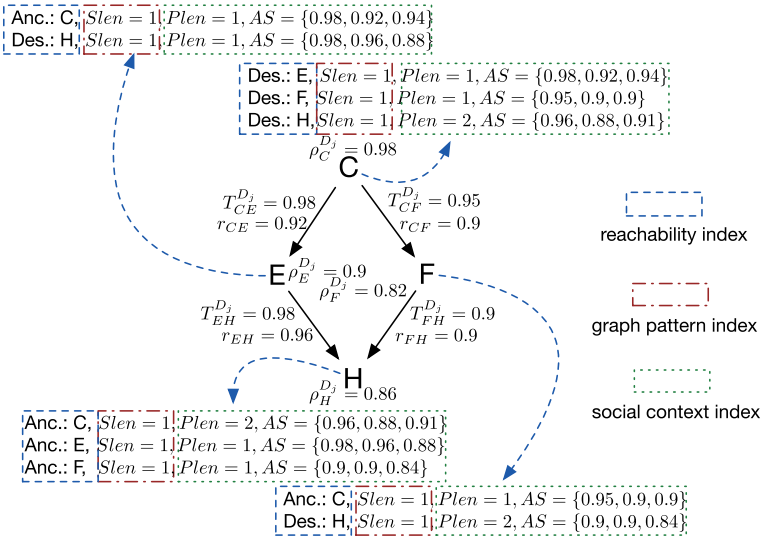

Figure 6 The index of an SSC

Given a query of the graph pattern matching with the bounded length, based on the graph pattern index, we can investigate if the indexed path length is greater than the bounded length, and thus can efficiently find a pattern matching result.

Social context index In order to improve the efficiency of TAG-K finding, we construct the social context index to record the maximal aggregated social impact factor values of the mapped paths in a data graph. Below are the details of the index.

- If $p \propto p'$, we index that path length and the corresponding aggregated social impact factor values of p instead of p'.
- Otherwise, we index up to three paths that have the maximal aggregated T, r and ρ values respectively.

Example 13 Consider the social context index shown in Figure 6. Here we take the vertex C as an example, where there are two paths from C to its descendant H, e.g., path $p1_{(C,E,H)}$ and $p2_{(C,F,H)}$. As $p1 \propto p2$, we index $A_S(p1_{(C,E,H)}) = \{0.96, 0.88, 0.91\}$ and its path length $Plen(p1_{(C,E,H)}) = 2$ at C. Similarly, we construct the social context index for each of the other vertices. Given a graph pattern query with multiple constraints, based on the social context index, we can quickly investigate if there exists a pattern match in the data graph, and thus saving query processing time.

4.3 Summary

In TAG-K designated node finding, if the two nodes with a connection in G_P can be mapped into a path in SSCs, this indexed information can be used to quickly investigate if there is an edge pattern match, and thus greatly saving graph pattern matching time. In addition, in the worst case, we need to perform the Dijkstra's algorithm four times, and thus the time complexity of the index construction is $O(N_D log N_D + E_D)$, where N_D and E_D are the nodes and edges in G_D. Furthermore, as mentioned in Section 4.2, the structure and the social contexts

of the graph in an SSC usually stay stable in a very long period of time [1]. Therefore, usually it is not necessary to update the indices frequently, which reduces the cost of index maintenance. When there are some changes of the social contexts and/or graph structure in an SSC, we can adopt the existing method [9] to first establish the matrices of the shortest path length, the ancestors and descendants, and the aggregated social impact factor values between vertices, and then iteratively investigate the updated SSC graph, finding the affected vertices and edges to update the corresponding matrices. The index maintenance in dynamic graphs is another challenging research topic and thus it is not discussed in this paper.

5 A-TAG-K: an approximation algorithm for TAG-K

In order to solve the NP-Complete TAG-K designated node finding problem, we propose an approximation algorithm, A-TAG-K, by adopting the Top-K shortest path algorithm [8] to investigate if there is a path in G_D can match an edge in G_P. Our A-TAG-K can deliver a set, $M_u^s(v_d, G_P, G_D)$, where $M_u^K(v_d, G_P, G_D) \subseteq M_u^s(v_d, G_P, G_D)$ without accessing all the nodes in G_D.

5.1 Algorithm overview

A-TAG-K first computes the *upper bound* a ranking function of a matching based on all the candidate nodes in the matching, and then A-TAG-K adopts the Top-K shortest path algorithm [8] to investigate the path matching between these candidate nodes, and update the *lower bound* of the ranking function based on the investigation. When finding K elements where the minimal lower bound is larger than the maximal upper bound of other matching in G_D, we deliver the K elements as the TAG-K designated node finding results.

5.2 The bound of TAG-K matching

In order to improve the efficiency of A-TAG-K, we first fast to compute an estimated approximate values of $RF(u, v_d)$ as the *lower bound* and the *upper bound* of $RF(u, v_d)$ respectively, denoted as $RF^L(u, v_d)$ and $RF^U(u, v_d)$, where $RF^L(u, v_d) \leqslant RF(u, v_d) \leqslant RF^U(u, v_d)$. If $RF^L(u_i, v_d) > RF^U(u_j, v_d)$, we can know $RF(u_i, v_d) > RF(u_j, v_d)$. Namely u_i is a better designated node matching than u_j. Thus, after efficiently comparing the lower bound and upper bound of each candidate in G_D, A-TAG-K can stop as soon as the Top-K matches are identified, without computing the entire $M(G_P, G_D)$. Below is the details of the computation of $RF^L(u, v_d)$ and $RF^U(u, v_d)$.

Upper bound Given a candidate u of v_d, we use $D(u, v_d)$ to denote the set of all descendants of u, where each of the descendant can match the corresponding label of the node v' in G_P. In $D(u, v_d)$, we compute $RF(D(u, v_d), v_d)$ as $RF^U(u, v_d)$. As some of the paths between nodes in $D(u, v_d)$ may not be a match of (v, v') in G_P. Namely, the aggregated social impact factor values of these paths cannot satisfy the corresponding constraints in G_P Thus, $RF(u, v_d) \leqslant RF^U(u, v_d)$.

Lower bound Initially, set $RF^L(u, v_d) = 0$, A-TAG-K investigates each pair of the descendants in $D(u, v_d)$ to find the Top-K shortest path between the nodes based on the algorithm in [8] to compute the utility of the path by (2). After each investigation, $RF^L(u, v_d)$ increases when investigated the path between two nodes is a matching. During

the iterations of investigation, $RF^L(u, v_d) < RF(u, v_d)$, and after all the iterations, we can get $RF^L(u, v_d) = RF(u, v_d)$. Based on the *lower bound* and *upper bound*, we have the below *Lemma 1*.

Lemma 1 *A K-element set* $M_u^K(v_d, G_P, G_D) \subseteq D(u, v_d)$ *is a set of Top-K matches of* v_d *if (1) each* u_i *in* $M_u^K(v_d, G_P, G_D)$ *is a match of* v_d*, and (2)* $RF_{min}^L(u_i, v_d) \geqslant RF_{max}^U(u_j, v_d), u_j \in D(u, v_d)$.

Based on the above Lemma 1, we can perform the early termination after finding the K elements meeting the corresponding requirements, without computing the entire $M_u(v_d, G_P, G_D)$*, greatly saving the processing time of TAG-K finding.*

5.2.1 Search procedure

The algorithm A-TAG-K mainly has two stages. The first stage is to compute the upper bound of each candidate node, and the second stage is to compute the lower bound of each candidate node. Below are the details of A-TAG-K.

(1) **UpperBound (Algorithm 1)** A-TAG-K selects a node u from $V_{candidate}$ to updates the $RF^U(u, v_d)$. During this process, u firstly is put into V_{temp}. Then, for each adjacent edge (v_d, v') of the corresponding v_d of u, A-TAG-K finds the matching via the BFS method to find the set V' of all descendants of u that have the same labels as the children of v_d. For each node in V', based on the Top-K shortest path selection algorithm [8], A-TAG-K computes $RF^U(u, v_d)$, and updates the upper bound All descendants in V' will be put into V_{temp}, and $RF^U(u, v_d)$ takes a node u' from V_{temp}, and updates the corresponding upper bound. If V_{temp} is empty, based on (3), the upper bound of u can be delivered.

Algorithm 1 Upperbound

Input: A DAG pattern $G_P = (V_p, E_p, f_v, f_e, s_e, v_0), G$.
Output: $V_{leafnode}$, max-heap Max.

```
while each u ∈ V_candidate do
    put u into V_temp;
    while V_temp ≠ ∅ do
        pop up u' from V_temp;
        while ∃(v', v'') ∈ AdjEdges of v' do
            V' = BFS(u', v'');
            while ∃u ∈ V' do
                D(u, v_d) = D(u, v_d) ∪ u;
                δ(u, v_d) = max{U(u'_i, u'_j) | u'_i ∈ D(u, v_d), u'_j ∈ D(u, v_d)};
                N_u ++;
        V_temp = V' ∪ V_temp;
    if AdjEdges of v' = ∅ then
        V_leafnode = V_leafnode ∪ u';
    RF^U(u, v_d) = α * arctan(D(u, v_d)) * 2/π + (1 − α) * δ(u, v_d);
    push u onto Max;
```

Algorithm 2 LowerBound

Input: $V_{leafnode}$, max-heap Max.
Output: M, the top-K matches of v_d.

```
1  min-heap Min;
2  while V_leafNode ≠ ∅ do
3      pop up u' from V_leafnode;
4      while ∃u ∈ Parent(u') do
5          if u' is a match of v_0 then
6              V_leafnode = V_leafnode ∪ u;
7              R_(u,v_d) = R_(u',v_d) ∪ u';
8              T(u, v_d) += T(u', v_d) + U(P(u, u'));
9          if f_v(v_0) = f(u) then
10             δ(u, v_d) = T(u, v_d)/N_u;
11             RF^L(u, v_d) = α * arctan(R_(u,v_d)) * 2/π + (1 − α) * δ(u, v_d);
12             update Max and Min;
13         if RF^L(Root(Min), v_d) > RF^U(Root(Max), v_d) then
14             break;
15 return Min;
```

(2) **LowerBound (Algorithm 2)** A-TAG-K selects a node u' from $V_{leafnode}$ to check if its parent node u can be a matching, and updates the lower bound of u. After updating the lower bound of u, A-TAG-K will detect the early termination, and A-TAG-K will stop if the early termination can be satisfied. Because u' is a leaf node, we can update the corresponding ranking function. Then, if u is the candidate of v_d, A-TAG-K will calculate the lower bound of u.

Summary Our proposed A-TAG-K algorithm is an efficient and effective method for the TAG-K problem in large-scale contextual social graphs. Our method achieves $O(E_P N_D^2 + E_P N_D)$ computation cost, where N_D is the number of nodes in G_D, and E_P is the number of edges in G_P.

6 Experiments

We conduct experiments on five large-scale real-world social graphs to evaluate (1) the effectiveness our algorithm in finding TAG-K designated nodes; and (2) the efficiency of our A-TAG-K algorithm.

6.1 Experiment setting

Datasets We use five real social graphs available at *snap. stanford.edu*, which have been widely used in the literature for graph pattern matching and social network analysis. The details of these datasets are shown in Table 1.

Table 1 The Social Datasets

Name	Nodes	Edges	Description
citHepTh	27,769	352,808	citation network
Slashdot	77,361	905469	social network
DBLP	317,081	1,049,868	collaboration network
Twitter	741,165	2,400,000	social network
YouTube	1,728,561	3,843,883	social network

Pattern graph and parameter setting

- We use a popular social network generation tool, SocNetV (socnetv.org, with version 2.2) to generate five query graphs, and the details of these graphs are shown in Table 2, where we random select a node from each of the pattern graph as the designated node v_d.
- A set of relative low constraints are specified as $\lambda_T = 0.05$, $\lambda_r = 0.05$ and $\lambda_\rho = 0.2$, to ensure the high possibility of returning TAG-K designated nodes in a data graph [17]. Otherwise, no or only few answers might be returned by all the algorithms, making it difficult to investigate their performance.
- As we discussed in Section 3, the social context impact factor values (i.e., T, r and ρ) can be mined from the existing social networks, which is another very challenging problem, but out of the scope of this work. Moreover, in the real cases, the values of these impact factors can vary from low to high without any fixed patterns. Without loss of generality, we randomly set the values of these impact factors by using the function rand() in SQL. In addition, in each of the datasets, the SSC number is set to 20, 40, 60, 80 and 100, respectively; α is set to 0.6, 0.7, 0.8, 0.9 and 1; K is set to 5, 10, 15, 20 and 25; and the maximal bounded path length of the pattern matching is set as 5, 10, 15, 20 and 25.

6.2 Implementation

We compare our method with the most promising algorithm in Top-K designated nodes finding, TopKP [10]. All A-TAG-K and TopKP algorithms are implemented using Java running on a PC with Intel Core i5-3470 3.20GHz CPU, 16GB RAM, Windows 10 operating system. All the experimental results are averaged based on five independent runs.

6.3 Experimental results and analysis

Exp-1: Effectiveness This experiment is to investigate the effectiveness of our A-TAG-K by comparing the average ranking function values of the Top-K designated nodes based on different setting of parameters.

Table 2 The pattern graphs

Name	Nodes	Edges
Q_5	5	7
Q_{10}	10	21
Q_{15}	15	34
Q_{20}	20	50
Q_{25}	25	63

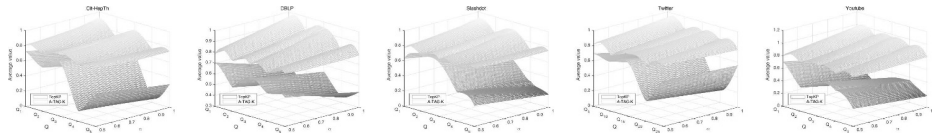

Figure 7 The average ranking function value based on different α

Results Figures 7, 8, 9, and 10 depict the average ranking function values of the delivered pattern matching with different setting of paramaters, by each of A-TAG-K and TopKP. From these figures, we can see that the average ranking function values of the Top-K designated nodes returned by TopKP are always less than that of A-TAG-K. Statistically, on average, A-TAG-K can return answers with the average ranking function value which is *72.54%* less than that of TopKP.

Analysis The experimental results illustrate that (1) TopKP considers the number of nodes only in designated node finding, but does not take the social contexts into consideration; and (2) our A-TAG-K can deliver the Top-K pattern matching results with considering both the number of nodes and the social contexts, which can effectively improve the quality of the query results.

Exp-2: efficiency This experiment is to investigate the efficiency of our A-TAG-K by comparing the average query processing time of A-TAG-K and TopKP based on different setting of parameters.

Results Figures 11, 12, 13, and 14 depict the average query processing time of A-TAG-K and TopKP in returning different numbers of designed nodes with different setting of parameters. From these figures, we can see that A-TAG-K has better efficiency than TopKP for the Top-K designated nodes finding in all the cases in the five datasets. Statistically, on average, the query processing time of A-TAG-K is *44.25%* less than that of TopKP.

Analysis The experimental results illustrate that A-TAG-K can efficiently find the candidate node of v_d in terms of MA-Tree, and can efficiently find the pattern matching in terms of the SSC-Index, which avoids to visit all the nodes and edges in a data graph. Thus, A-TAG-K can greatly save the query processing time.

6.4 Summary

The above experimental results demonstrate that the proposed A-TAG-K algorithm provides an effective means to find context-aware graph based Top-K designated nodes. In addition, with our proposed index structures, A-TAG-K can efficiently find the candidates, which greatly saves query processing time. Therefore A-TAG-K significantly outperforms the existing most promising algorithm for Top-K designated node finding, TopKP, in both

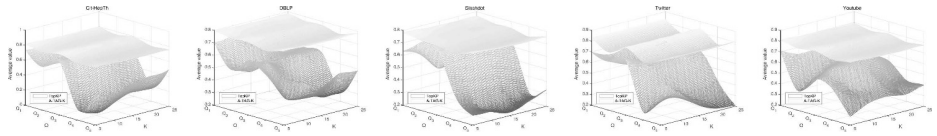

Figure 8 The average ranking function values based on different K

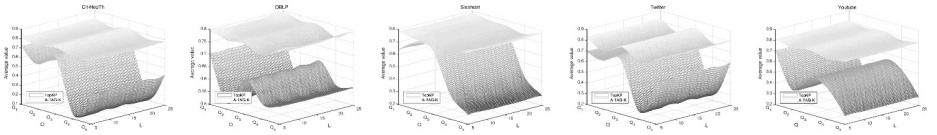

Figure 9 The average ranking function value based on different L

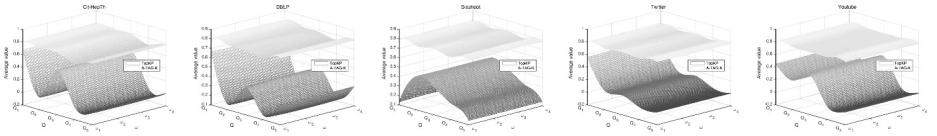

Figure 10 The average ranking function value based on different ω

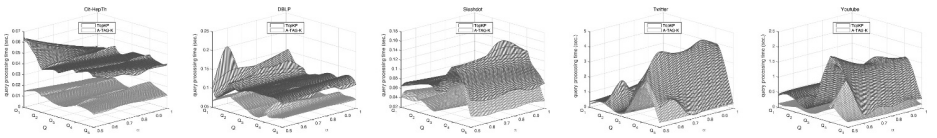

Figure 11 The query time based on different α

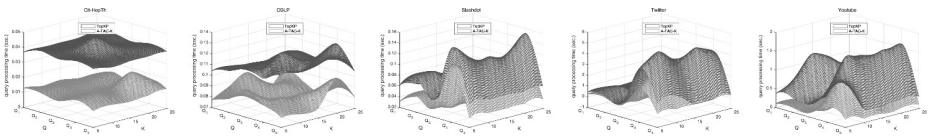

Figure 12 The query time based on different K

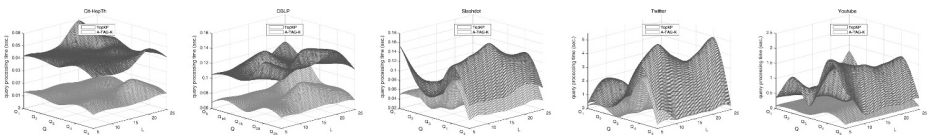

Figure 13 The query time based on different L

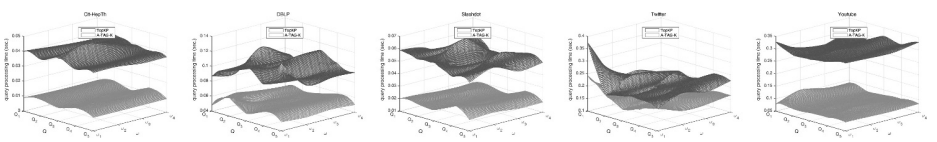

Figure 14 The query time based on different ω

effectiveness and efficiency. Therefore, A-TAG-K is a very competitive algorithm for the new TAG-K designated node finding problem in social network based applications.

7 Conclusion

In this paper, we have proposed an approximate algorithm A-TAG-K to support a new type context-aware graph pattern based Top-K designated node finding problem that is a corner stone for many social network based applications. A-TAG-K achieves $O(E_P N_D^2 + E_P N_D)$ in time cost, where N_D is the number of nodes in G_D, and E_P is the number of edges in G_P, and the experiments conducted on five real-world large-scale social graphs have demonstrated the superiority of our proposed approaches in terms of effectiveness and efficiency.

Acknowledgments This work was partially supported by Natural Science Foundation of China (Grant Nos. 61303019, 61572336, 61532018, 61402313, 61502324), Doctoral Fund of Ministry of Education of China (20133201120012), Postdoctoral Science Foundation of China (2015M571805, 2016T90492), Collaborative Innovation Center of Novel Software Technology and Industrialization, Jiangsu, China, and project supported by the Jiangsu Key Laboratory of Image and Video Understanding for Social Safety (Nanjing University of Science and Technology), Grant No. 30916014107.

References

1. Berger, P., Luckmann, T.: The Social Construction of Reality: A Treatise in the Sociology of Knowledge. Anchor Books, New York (1966)
2. Biggs, N., Lloyd, E., Wilson, R.: Graph Theory. Oxford University Press, Oxford (1986)
3. Chang, L., Lin, X., Zhang, W., Yu, J.X., Zhang, Y., Qin, L.: Optimal enumeration: efficient top-k tree matching. VLDB **8**(5), 533–544 (2015)
4. Chevalier, F., Domenger, J.P., Pineau, J.B., Delest, M.: Retrieval of objects in video by similarity based on graph matching. Pattern Recogn. Lett. **28**(8), 939–949 (2007)
5. Cohen, S., Kimelfeld, B., Koutrika, G., Jan, V.: On principles of egocentric person search in social networks. In: Workshop on VLDS, pp. 3–6 (2011)
6. Demuth, H.B., Beale, M.H., Jess, O.D., Hagan, T.M.: Neural Network Design. Martin Hagan Press (2014)
7. Ding, X., Jia, J., Li, J., Liu, J., Jini, H.: Top-k similarity matching in large graphs with attributes. In: DASFAA, pp. 156–170 (2014)
8. Eppstein, D.: Finding the k shortest paths. SIAM J. Comput. **28**(2), 652–673 (1998)
9. Fan, W., Li, J., Wang, X., Wu, Y.: Query preserving graph compression. In: SIGMOD'12, pp. 157–168 (2012)
10. Fan, W., Wang, X., Wu, Y.: Diversified top-k graph pattern matching. VLDB **6**(13), 1510–1521 (2013)
11. Fani, W., Li, J., Ma, S., Tang, N., Wu, Y., Wu, Y.: Graph pattern matching: from intractable to polynomial time. VLDB **3**(1-2), 264–275 (2010)
12. Lappas, T., Liu, K., Terzi, E.: A survey of algorithms and systems for expert location in social networks. In: Social Network Data Analytics, pp. 215–241 (2011)
13. Liu, G., Wang, Y., Orgun, M.A.: Trust transitivity in complex social networks. In: AAAI, pp. 1222–1229 (2011)
14. Liu, G., Wang, Y., Orgun, M.A.: Social context-aware trust network discovery in complex contextual social networks. In: AAAI, vol. 12, pp. 101–107 (2012)
15. Liu, G., Wang, Y., Orgun, M.A., et al.: Optimal social trust path selection in complex social networks. In: AAAI, pp. 1397–1398 (2010)
16. Liu, G., Wang, Y., Orgun, M.A., Lim, E.P.: Finding the optimal social trust path for the selection of trustworthy service providers in complex social networks. IEEE Transactions on Services Computing (TSC) **6**(2), (2013)

17. Liu, G., Zheng, K., Wang, Y., Orgun, M.A., Liu, A., Zhao, L., Zhou, X.: Multi-constrained graph pattern matching in large-scale contextual social graphs. In: ICDE, pp. 351–362 (2015)
18. Liu, G., Zheng, K., Wang, Y., Orgun, M.A., Liu, A., Zhao, L., Zhou, X.: Multi-constrained graph pattern matching in large-scale contextual social graphs. In: ICDE'15, pp. 351–362 (2015)
19. Milano, R., Baggio, R., Piattelli, R.: The effects of online social media on tourism websites. In: ENTER, pp. 471–483 (2011)
20. Modiano, E.: Traffic grooming in wdm networks. IEEE Commun. Mag. **39**(7), 124–129 (2001)
21. Morris, M.R., Teevan, J., Panovich, K.: What do people ask their social networks, and why?: a survey study of status message q&a behavior. In: CHI., pp. 1739–1748 (2010)
22. Rauch, M., Thomas, H., Peter, K.: Computing simulations on finite and infinite graphs. In: Annual Symposium o Foundations of Computer Science, pp. 453–462 (1995)
23. Raymond, W.J., Willett, P.: Maximum common subgraph isomorphism algorithms for the matching of chemical structures. J. Comput. Aided Mol. Des. **16**(7), 521–533 (2002)
24. Schenkel, R., Crecelius, T., Kacimi, M., Michel, S., Neumann, T., Parreira, J., Weikum, G.: Efficient top-k querying over social-tagging networks. In: SIGIR, pp. 523–530 (2008)
25. Song, J., Gao, L., Liu, L., Zhu, X., Sebe, N.: Quantization-based hashing: a general framework for scalable image and video retrieval. Pattern Recogn. **75**, 175–187 (2018)
26. Song, J., Gao, L., Nie, F., Shen, H.T., Sebe, N.: Optimized graph learning using partial tags and multiple features for image and video annotation. IEEE Trans. Image Process. **25**(11), 4999–5011 (2016)
27. Song, J., Shen, H.T., Wang, J., Huang, Z., Sebe, N., Wang, J.: A distance-computation-free search scheme for binary code databases. IEEE Trans. Multimedia **18**(3), 484–495 (2016)
28. Tang, J., Zhang, J., Yan, L., Li, J., Zhang, L., Su, Z.: Arnetminer: Extraction and mining of academic social networks. In: KDD, pp. 990–998 (2008)
29. Tian, Y., Patel, P.M.: Tale: A tool for approximate large graph matching. In: ICDE, pp. 963–972 (2008)
30. Ullmann, R.J.: An algorithm for subgraph isomorphism. J. ACM **23**(1), 31–42 (1976)
31. Wang, J., Zhang, T., Song, J., Sebe, N., Shen, H.T.: A survey on learning to hash. IEEE TPAMI **August**(99), 1–1 (2017)
32. Yildirim, H., Chaoji, V., Zaki, M.J.: Grail: Scalable reachability index for large graphs. In: VLDB, pp. 276–284 (2010)
33. Yoo, S., Yang, Y., Lin, F., Moon, I.: Mining social networks for personalized email prioritization. In: KDD, pp. 967–976 (2009)
34. Zhu, X., Zhang, L., Huang, Z.: A sparse embedding and least variance encoding approach to hashing. IEEE Trans. Image Process. **23**(9), 3737 (2014)
35. Zhu, Y., Qin, L., Yu, J.X., Cheng, H.: Finding top-k similar graphs in graph databases. In: EDBT, pp. 456–467 (2012)