Multi-Constrained Graph Pattern Matching in Large-Scale Contextual Social Graphs

Guanfeng Liu1,2, Kai Zheng3, Yan Wang4, Mehmet A. Orgun4, An Liu1,2, Lei Zhao1,2 and Xiaofang Zhou1,3

1School of Computer Science and Technology, Soochow University, Suzhou 215006, China
2Collaborative Innovation Center of Novel Software Technology and Industrialization, Jiangsu, China
3School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane 4072, Australia.
4Department of Computing, Macquarie University, NSW 2109, Australia.
{gfliu, anliu, zhaol}@suda.edu.cn; {kevinz, zzxf}@itee.uq.edu.au; {yan.wang, mehmet.orgun}@mq.edu.au

Abstract—Graph Pattern Matching (GPM) plays a significant role in social network analysis, which has been widely used in, for example, experts finding, social community mining and social position detection. Given a pattern graph $G_Q$ and a data graph $G_D$, a GPM algorithm finds those subgraphs, $G_M$, that match $G_Q$ in $G_D$. However, the existing GPM methods do not consider the multiple constraints on edges in $G_Q$, which are commonly exist in various applications such as, crowdsourcing travel, social network based e-commerce and study group selection, etc.

In this paper, we first conceptually extend Bounded Simulation to Multi-Constrained Simulation (MCS), and propose a novel NP-Complete Multi-Constrained Graph Pattern Matching (MC-GPM) problem. Then, to address the efficiency issue in large-scale MC-GPM, we propose a new concept called Strong Social Component (SSC), consisting of participants with strong social connections. We also propose an approach to identify SSCs, and propose a novel index method and a graph compression method for SSC. Moreover, we devise a heuristic algorithm to identify MC-GPM results effectively and efficiently without decompressing graphs. An extensive empirical study on five real-world large-scale social graphs has demonstrated the effectiveness, efficiency and scalability of our approach.

I. INTRODUCTION

A. Background

Online Social Networks (OSNs) have attracted billions of users worldwide, which have provided rich information for users to perform various tasks such as detecting social positions [1], finding experts [2], [3] and making travel plans [2]. These tasks are typically conducted by applying Graph Pattern Matching (GPM) which has been widely used in social network analysis. GPM is typically defined in terms of subgraph isomorphism, in which, given a data graph $G_D$ and a pattern graph $G_Q$ as input, it answers whether $G_D$ contains a subgraph that is isomorphic to $G_Q$.

Example 1: Fig. 1 contains three query pattern graphs $G_{Q1}$, $G_{Q2}$ and $G_{Q3}$, and four data graphs $G_{D1}$, $G_{D2}$, $G_{D3}$ and $G_{D4}$. $G_{Q1}$ is isomorphic to $G_{D1}$, but is not isomorphic to $G_{D2}$, $G_{D3}$ or $G_{D4}$.

However, as shown in [2], the conventional subgraph isomorphism is too strictly defined to find useful patterns in the real-world social graphs. Moreover, due to the NP-complete time complexity, it is hard to apply isomorphism test to large-scale social graphs.

In order to address the above-mentioned issues in subgraph isomorphism, graph simulation [4] has been proposed which has less restrictions but more capacity to extract more useful subgraphs with better efficiency. In contrast to subgraph isomorphism, graph simulation supports simulation relations instead of exact match of vertices. Recalling Fig. 1, $G_{Q1}$ is not isomorphic to $G_{D2}$, but it matches $G_{D2}$ via graph simulation as $B$ and $C$ in $G_{Q1}$ can be simulated to one out of $B$ and $C$ in $G_{D2}$ respectively. Graph simulation has been widely used in structural index and website classification, but it still needs to perform edge-to-edge mapping (e.g., the edge $(A,C)$ and $(B,C)$ in the case of graph simulation in Fig. 1). This is still too strict for some real applications that utilize the connectivity between vertex pairs via a path with arbitrary or pre-defined lengths [5], [6] (e.g., path lengths 2 and 3 in $G_{Q2}$).

To address this issue in graph simulation, Fan et al., [2] proposed bounded simulation, wherein each vertex has a label of a category, and each edge is labeled with either a constant $k$ or a *. Graph matching based on bounded simulation maps edges in a pattern graph to paths within bounded lengths in a data graph, instead of edge-to-edge mappings in subgraph isomorphism and graph simulation [2]. As shown in the example in Fig. 1, $G_{Q2}$ matches $G_{D1}$, $G_{D2}$ and $G_{D3}$ via bounded simulations. Based on bounded simulation, some
studies have been conducted to restrict the simulation structure [7], match with different edge types [8], and improve the efficiency of bounded simulation based GPM [9], [10].

B. Problem

The bounded simulation based GPM only considers the bounded path length of an edge when matching the edges, which greedily finds the subgraph that has the minimal diameter. However, some graphs have attributes on vertices and edges, like the Contextual Social Graph (CSG) [11], where each vertex has the social role information, and each edge has the social relationships and social trust information. In a variety of applications in social networks, e.g., crowd-sourcing travel [12], study group selection (classroomsalon.com), and social network based e-commerce [11], people are willing to incorporate multiple social contexts associated with edges in a CSG, which have significant influence on people’s collaborations and decision making [13].

Example 2: Consider $G_{Q3}$ and $G_{D4}$ in Fig. 1, where in addition to the traditional graph structure, each edge in $G_{D4}$ is associated with two attributes: $a$ and $b$ that can be social trust and social relationships between people in CSGs. In real applications on CSGS, the constraints of social trust (e.g., $a > m$ in $G_{Q3}$) and social relationships (e.g., $b < n$ in $G_{Q3}$) can be specified between, for example, a Project Manager $A$ and an Assistant Manager $B$ to find a trustworthy team, or between two customers $A$ and $B$ to help retailers find loyal customers in social network based CRM (Customer Relation Management) systems. In such multi-constrained graph pattern matching, a path in $G_{D4}$ is a match of $(A, B)$ in $G_{Q3}$, if the path length is no greater than 2, and the aggregated values of $a$ and $b$ can satisfy the multiple constraints, i.e., $a > m$ and $b < n$, ($m$ and $n$ are constants).

This example illustrates that a new type of Multi-Constrained GPM (MC-GPM) is an important issue in social graphs. MC-GPM subsumes the classical NP-Complete multi-constrained path selection problem [14], and thus is NP-Complete. So, the main challenge of our work is to design novel indexing structures and approximation techniques to effectively and efficiently support MC-GPM queries. To the best of our knowledge, all the existing methods of GPM do not support MC-GPM queries. Our contributions are summarized as follows.

C. Contributions

(1) We propose a new notion of Multi-Constrained Simulation (MCS) by extending bounded simulation. In contrast to its traditional counterpart, the MCS based MC-GPM is to find a graph pattern matching result, where each edge of the matching graph satisfies both the bounded path length and the multiple constraints on edges, which can better support many emerging social network based applications.

(2) We propose a concept called Strong Social Component (SSC), which consists of participants who have strong social connections, and propose an approach to identify SSCs. As the social connections in SSC usually stay stable in a very long period of time [15], we propose a novel index structure and a graph compression method for SSC with polynomial time complexity. Our method can match the pattern graph without any graph decompression, which can reduce storage consumption and improve efficiency.

(3) Based on the indices and compressed graph, we propose a Heuristic Algorithm for MC-GPM, called HAMC. In HAMC, a novel objective function is proposed to test if an edge matching is included in a data graph. HAMC has the time complexity of $O(E_{Q}N_{D}logN_{D} + E_{Q}E_{D})$, where $N_{D}$ and $E_{D}$ are the number of vertices and edges respectively in the data graph, and $E_{Q}$ is the number of edges in the query graph.

(4) An extensive empirical study based on five large-scale real-world social graphs has demonstrated the effectiveness, efficiency and scalability of our proposed algorithms.

The rest of this paper is organized as follows. We first review the related work on GPM in Section II. Then we introduce the necessary concepts and formulate the focal problem of this paper in Section III. The strong social component identification is presented in Section IV, followed by the graph compression methods and index structures proposed in Section V and Section VI respectively. Section VII presents our proposed MC-GPM algorithm, Section VIII reports the experimental observations, and Section IX concludes the paper.

II. RELATED WORK

In the literature, GPM methods have already been widely studied, which can be categorized into (1) the isomorphism-based GPM to match each of the vertices and edges in $G_{Q}$ exactly, and (2) the simulation-based GPM to simulate pattern matching of the vertices and edges in $G_{Q}$. Below we analyze them in detail.

Isomorphism-Based GPM: In the studies of isomorphism-based GPM, Zou et al., [6] index the shortest path length between any two vertices in a data graph to support the requirement of the connection length. In addition, Sun et al., [16] adopt a graph exploration method to improve the efficiency of subgraph joint processing in graph pattern matching. Furthermore, Cheng et al., [17] propose a top-k graph pattern matching approach which builds up a spanning tree of a cyclic graph query, and rank the answers by the sum of the edge lengths of an answer.

Given a query graph $G_{Q}$, usually it is not realistic to find a isomorphic subgraph as the strict matching. Yan et al.,[18] propose a similarity based method, where a distance is computed based on the total number of the matching edges between a query graph and the data graph. If the distance is less than a specified threshold, then an answer is returned. In addition, Shang et al., [19] further improve the similarity-based method by indexing graphs according to their similarity to the features in $G_{Q}$, which can prune those non-promising graphs. Furthermore, Zhu et al., [20] divide a $G_{D}$ into several groups of similar graphs and index these graphs to support effective non-promising graph pruning.
In order to improve the efficiency of the GPM in large data graphs, some paralleled and distributed GPM methods have been proposed recently. In [21], [22] a pattern graph is decomposed into several small patterns, and they find the subgraph matchings for each small pattern graph and join the intermediate results finally. In addition, in [23], they find multiple GPM answers based on a paralleled framework. Furthermore, in [24], a large data graph is decomposed into several small fractions based on a distributed GPM method, and then they predict the probability of each fraction to having a GPM answer.

**Simulation-Based GPM:** As isomorphism-based GPM is still too strict for some applications, based on graph simulation [4], Fan et al., [2] propose bounded simulation based GPM method, where the label of each vertex is not unique and a bounded length can be specified on the edges in a query graph. In bounded simulation, it is not necessary to exactly match each vertex and each edge as the subgraph isomorphism, instead, matching any vertex having the same labels and the path whose length is not greater than the bounded length in a data graph. This type of GPM can be conducted in cubic-time. In addition, based on bounded simulation, Ma et al., [7] propose strong simulation to find a small set of matches whose topologies are more similar with the query graph than bounded simulation. Moreover, Fan et al., [8] further consider the requirements of different types of edges in GPM. Furthermore, in order to improve efficiency, Fan et al., [10] later propose a graph pattern view based bounded simulation, where a set of views are defined in a data graph, and an estimation method has been proposed to predict which view can be used to answer a specific query. Moreover, they propose a resource-bounded query [25] where a fraction of a data graph that has a high probability of containing the query graph is extracted. Finally, Fan et al., [9] propose a method to find the top-k matching of the specific vertex patterns, where some patterns that contain important vertices and edges have high priority to be matched in a query graph.

**Summary:** The isomorphism-based GPM are important in many applications, e.g., 3D object matching [26] and protein structure matching [27]. The index and the paralleled and distributed methods are effective ways to improve the efficiency of GPM. However, such GPM is still suffering expensive computation cost as it is NP-Complete. Moreover, although the similarity-based method enhance the probability of returning an answer, it still too strict to use in some applications, e.g., finding social experts [2] and project organization [9]. Simulation-based GPM methods relax the restrictions of subgraph isomorphism and thus well address the GPM in these applications. But all the existing methods do not consider the multiple constraints on edges in a graph query. Such a query is popular and fundamental in many social network based applications, like crowd-sourcing travel [12], study group selection (classroomsalon.com), and social network based e-commerce [11]. Therefore, the existing methods cannot support the significant MC-GPM in many applications.

### III. PRELIMINARIES

#### A. Data Graph

1) **Contextual Social Graph:** A Contextual Social Graph (CSG) [11] is a labeled directed graph \( G = (V, E, LV, LE) \), where

- \( V \) is a set of vertices;
- \( E \) is a set of edges, and \((v_i, v_j) \in E\) denotes a directed edge from vertex \( v_i \) to vertex \( v_j \);
- \( LV \) is a function defined on \( V \) such that for each vertex \( v \in V \), \( LV(v) \) 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 link \((v_i, v_j) \in E\), \( LE(v_i, v_j) \) is a set of labels for \((v_i, v_j)\), like social relationships and social trust in a specific domain.

**Example 3:** We now give a specific example of a CSG from [11] to show how the vertex and edge labels can be defined and used. \( G_{DS} \) in Fig. 2 is a CSG, where each vertex \( v_i \in V \) is associated with a role impact factor, denoted as \( \rho_{Di} \in [0, 1] \), to illustrate the impact of participant \( v_i \) in domain \( i \), which is determined by the expertise of \( v_i \), \( \rho_{Di} = 1 \) indicates that \( v_i \) is a domain expert in domain \( i \) while \( \rho_{Di} = 0 \) indicates that \( v_i \) has no knowledge in that domain. Moreover, each edge \((v_i, v_j)\) is associated with social trust, denoted as \( T_{v_i,v_j} \in [0, 1] \), and social intimacy degree, denoted as \( r_{v_i,v_j} \in [0, 1] \), to illustrate trust and intimacy social relationships between participants. \( T, r \) and \( \rho \) are called social impact factors, whose values can be extracted by using the data mining techniques [28], [29], [30].

Based on the theories in Social Psychology [15], we adopt the multiplication method to aggregate \( T \) and \( r \) values of a path, and adopt the average method to aggregate the \( \rho \) values of the vertices in a path. The details of the aggregation method has been discussed in [11]. The aggregated values of a path \( p \) in domain \( i \) is denoted as \( AS^{Di}(p) = \{ AT^{Di}(p), Ar(p), A\rho^{Di}(p) \} \). If each of the aggregated social impact factor value of \( p \) is greater than the corresponding one of path \( p' \), then \( p \) dominates \( p' \) in domain \( i \), which is denoted as \( p \geq_{DOM} p' \).

#### B. Pattern Graph

A **Pattern Graph** is defined as \( G_Q = (V_q, E_q, f_v, f_e, s_e) \), where
Suppose \( Q \) is a query given by a 
baseline method (named as BaseLine) that extends 
The GPM algorithm via bounded simulation. The details of 
BaseLine are as follows,

**Step 1:** For each constrained edge in \( G_Q \), compute the 
mapped shortest path length, \( \text{Slen}(p) \), in \( G_D \).

**Step 2:** If \( \text{Slen}(p) \) is no greater than the bounded length 
(denoted as \( \text{Blen} \)) specified on the corresponding edge, i.e., 
\( \text{Slen}(p) \leq \text{Blen}(p) \), investigate if the aggregated 
social impact factor values of \( p \) can satisfy the corresponding 
constraints.

- If each of the constraints can be satisfied, an edge 
mapping is returned.
- Otherwise, compute the second shortest path length based 
on the top-k shortest path selection algorithm [31] and go 
to Step 2.

**Step 3:** After investigating all the edges in \( G_Q \), if there is 
an edge mapping for each of the edge in \( G_Q \), return an answer 
by the exploration based graph pattern matching method (see 
details in Section VII-B). Otherwise, there are no GPM answers 
included in \( G_D \).

**Example 5:** Consider the case of MC-GPM shown in 
Fig. 2. BaseLine can return the edge mapping answer, 
\((A, C_1, G_{D5}) \simeq (A, C, G_{Q4}), \) and \((A, C_1, D, G_{D5}) \simeq 
(A, D, G_{Q4})\). Then it returns the edge mapping answer 
of \((B, C)\) in \( G_{Q4} \) are \((B, C_2, G_{D5}) \simeq (B, C, G_{Q4})\), and 
the edge mapping of \((B, D)\) is \((B, D, G_{Q4}) \simeq (B, C_2, C_1, G_{D5})\) because 
the constraints cannot be satisfied by another path via \( B, C_2, F \) 
and \( D \), i.e., \( AT^{D_5}(P_{(B, C_2, F, D)}) < 0.6 \). As there is at 
least an edge mapping result for each edge in \( G_{Q4} \), an 
MC-GPM answer can be returned by using the exploration 
based GPM method. The matching graph is \( G_M = (V, E, LV, LE) \), 
where \( V = \{A, B, C_1, C_2, D\} \) and \( E = 
\{(A, C_1), (B, C_2), (C_2, C_1), (C_1, D)\}\). 

Suppose for each edge in \( G_Q \), BaseLine needs to perform 
the Dijkstra’s algorithm in \( G_D \) for \( N \) times. Then the time 
complexity of BaseLine is \( O(E_{Q} N N_{D} log N_D + N E_{Q} E_{D}) \).

**IV. STRONG SOCIAL COMPONENT**

In order to enhance the efficiency and effectiveness of our 
MC-GPM method, in this section, we propose a strong social 
component identification method. In graph theory [32], 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 Strong Social Component as below.

**Definition 1: Strong Social Component.** 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.

**Example 6:** In an SSC, suppose the $T, r$ and $\rho$ values associated with each of the vertices and edges should be greater than 0.8. Fig. 3 depicts a graph that has two strong social components in domain $i$ and domain $j$ respectively, where the $T, r$ and $\rho$ values are greater than 0.8.

Based on the theories in Social Psychology [15], 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 low update cost.

Identifying all the SSCs in a specific domain subsumes the classical NP-Complete maximum clique problem [32], which is very time consuming. Alternatively, we can identify up to $K$ SSCs for MC-GPM. We propose an SSC identification method which first randomly selects $K$ vertices that are associated with high role impact factor values as the seeds. Then from each of the seeds, our algorithm adopts the 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 $O(NDE_D)$. The pseudo-code of the algorithm is shown in Algorithm 1.

V. CONTEXT-PRESERVED GRAPH COMPRESSION FOR SSC

In this section, based on the existing graph compression method for bounded simulation [33], we propose a context-aware graph compression method, where the reachability, graph pattern and social contexts are preserved. Moreover, the graphs compressed by our approach can be directly queried without any decompression. In contrast, the existing compression methods are not designed for solving the MC-GPM problem, and thus they cannot preserve the social context information. Rather, the existing approaches have to restore the original graph from compact structures to answer a graph pattern query.

A. Compression for Reachability

A reachability query for a pair of vertices in a pattern graph $G_Q$ is to investigate if there exists at least one path linking the two vertices in a data graph, e.g., $(B, C)$ of $G_{Q3}$ in Fig. 1. The graph compression property captured by Theorem 1 preserves reachability information, which is called reachability preserved compression, denoted as $G_D^R$.

**Theorem 1:** The compressed graph is reachability preserved when the compressed two vertices have the same ancestors and can reach the descendants of each other.

**Proof:** Suppose there is a data graph $G_D = (V, E)$, where $V = \{A, B_1, ..., B_n, C_1, ..., C_n, D_1, D_2\}$ and $E = \{(A, B_1), ..., (A, B_n), (A, C_1), ..., (A, C_n), (B_1, D_1), ..., (B_n, D_1), (C_1, D_2), ..., (C_n, D_2), (D_1, D_2), (D_1, D_2)\}$.

$G_D^R = (V, E)$, where $V = \{A, B_1, ..., C_1, ..., D_2\}$ and $E = \{(A, B_{1...nC_{1...n}}), (B_{1...nC_{1...n}}, D_1), (D_1, D_2)\}$. For a reachability query $G_{D'}^R = (V, E)$, where $V = \{A, B_{1, D_{1}}, D_j\}, i \in [1, n]$ and $j \in [1, 2]$, $G_D^R$ is not reachability preserved compression if and only if $A$ and $B$, or $B_1$ and $D_j$ is not reachable, which contradicts $E_{D'}^R = \{(A, B_{1...nC_{1...n}}), (B_{1...nC_{1...n}}, D_1), (D_1, D_2)\}$ in $G_D^R$. Therefore, Theorem 1 is proven. 

**Example 7:** Fig. 4 contains two groups of graphs\(^1\). Consider Fig. 4 (a), where $G_{D7}$ is the original data graph and $G_D^R$ is the compressed graph. From $G_{D7}$, we can see that both $A$ and $B$ do not have any ancestors, and they can reach the same descendants ($C, D$ and $E$). Therefore, $A$ and $B$ can be compressed as one vertex in $G_D^R$, where the reachability of $A$ and $B$ to other vertices is preserved.

\(^1\)As the compressions do not consider any social context, in order to clearly display the graph structure, the social contexts of the graphs are not shown in this example.
B. Compression for Graph Pattern

In addition to reachability preserved compression, to support the graph pattern query, e.g., \((A, D)\) of \(G_{Q4}\) in Fig. 2, we propose a graph compression method that can preserve such graph patterns, which is called graph pattern preserved compression, denoted as \(G^P_{D}\). The property the compression method is captured by Theorem 2.

**Theorem 2:** The compressed graph is graph pattern preserved when the compressed two vertices have the same label, the same ancestors and the same descendants.

**Proof:** Suppose \(B_1,...,B_n\) are vertices in \(G_D\). \(A\) and \(C\) are their ancestor and descendant respectively. Then we can have a compressed graph \(G^P_{D}\), where \(B_1,...,B_n\) are compressed as a single vertex \(B\). If \(G_Q = (V,E)\), where \(V = \{A,B,C\}\) and \(E = \{(A, B), (B, C)\}\), \(G_Q \leq_{sim} G_D\). If \(G^P_{D}\) is not graph pattern preserved, for an edge \((u, u')\) in \(G_Q\), there exists an edge \((v, v')\) in \(G^P_{D}\) such that \((u, u') \notin S, (S \subseteq G_Q \times G_D)\). Namely, there exists vertex \(B_i\) such that \((A, B_i, G_D) \notin (A, B, G^P_{D})\) or \((B_i, C, G_D) \notin (B, C, G^P_{D})\). This contradicts the assumption that \(B_1,...,B_n\) have the same label. Therefore, Theorem 2 is proven. \(\square\)

**Example 8:** Consider the example shown in Fig. 4 that contains a data graph \(G_{DS}\) and the corresponding compressed graph \(G^P_{DS}\). In \(G_{DS}\), we can see that \(B_1\) and \(B_2\) have the same ancestor \(A\) and the same descendant \(C\). Therefore, based on Theorem 2, \(G^P_{DS}\) is a graph pattern preserved compression of \(G_{DS}\).

**Theorem 3:** The graph pattern preserved compression is reachability preserved

**Proof:** As illustrated in Theorem 1 and Theorem 2, the compression condition of graph pattern preserved compression is more strict than that of reachability preserved compression. Therefore, Theorem 3 is proven. \(\square\)

C. Compression for Social Contexts

In order to support MC-GPM, e.g., \((B, D)\) of \(G_{Q4}\) in Fig. 2, we propose a graph compression method that can preserve social context information, which is called social context preserved compression, denoted as \(G^S_{D}\). The property the compression method is captured by Theorem 4.

**Theorem 4:** The compressed graph is social context preserved when the compressed two vertices have the same label, the same ancestors, the same descendants, and \(AS\) of the path via one of the vertex dominates that of the other one.

**Proof:** Suppose \(B_1,...,B_n\) have the same ancestor \(A\) and the same descendant \(C\) in \(G_D\). \(AS(A,C)\) via \(B_i (1 \leq i \leq n)\) dominates others. Then, \(G_D\) is compressed as \(G^S_{D} = (V,E)\), where \(V = \{(A, B_1...B_n, C), E = \{(A, B_1...B_n)\}, AS(A,C) via B_1...B_n equals to \(AS(A,C)\) via \(B_i\). Given a query \(G_Q\) for \(A, C\) with \(Blen(A,C)\) and the multiple constraints of social impact factors on edge \((A, C)\), only the constraints on edge \((A, B)\) will be investigated as \(Blen(A,C) = 2\) via any \(B_k (1 \leq k \leq n)\). If \(G_{D}\) is not social context preserved, then the edge matching \((A, C, G_Q) \approx (A, C, G_{D})\) is missing in \(G^S_{D}\), namely one of the social impact factor in \(AS(A,C)\) via \(B_j (1 \leq j \leq n, and i \neq j)\) in \(G_{D}\) is greater than that of \((A,C)\) in \(G^S_{D}\) (i.e., \(AS(A,C)\) via \(B_i\)). This contradicts the assumption that \(AS(A,C)\) via \(B_i (1 \leq i \leq n)\) dominates others. Therefore, Theorem 4 is proven. \(\square\)

**Example 9:** Consider the example shown in Fig. 5 which contains a data graph \(G_{D9}\) and the corresponding compressed graph \(G^S_{D9}\). In \(G_{D9}\), we can see that \(C_1\) and \(C_2\) have the same ancestor \(A\) and same descendant \(D\). In addition, \(AS(A,D)\) via \(C_1\) dominates \(AS(A,D)\) via \(C_2\). Then, based on Theorem 4, \(G_{D9}\) is compressed as \(G^S_{D9}\), where \(C_1\) and \(C_2\) in \(G_{D9}\) are compressed as one vertex and \(AS(A,D)\) via \(C_1\) in \(G^S_{D9}\) equals to the dominated one in \(G_{D9}\), i.e., \(AS(A,D)\) via \(C_1\). Then \(G^S_{D9}\) is a social context preserved.

**Theorem 5:** The social context preserved compression is graph pattern preserved and reachability preserved

**Proof:** As illustrated in Theorem 3 and Theorem 4, the compression condition of social context preserved compression is more strict than that of that of graph pattern preserved compression. Therefore, Theorem 5 is proven. \(\square\)

D. Summary

Our graph compression method can preserve important information of a CSG, including reachability, graph pattern and social contexts information. Thus, a graph pattern query of MC-GPM can be answered based on the compressed data graph without any decompression. By compressing vertices and edges in the graph of an SSC, our graph compression method can greatly save the memory and query processing time (see details in experiments). In addition, in the worst case, our compression method needs to visit all the vertices.
and edges in a data graph. Therefore, the time complexity of our compression method is \( O(N_D E_D) \). As mentioned in Section IV, the structure and the social contexts of the graph in an SSC usually stay stable in a very long period of time [15]. Therefore, it is not necessary to update the compressed data graph frequently. When there are some changes of the social contexts and/or graph structure in an SSC, some efficient incremental graph compression methods in the literature can be adopted [10], [33]. These methods investigate if the changes affect the GPM in the compressed graph, and they only recompress a small subgraph that is affected by the changes, instead of recompressing the whole data graph.

VI. INDEX OF STRONG SOCIAL COMPONENTS

In order to improve the efficiency of MC-GPM, we propose a novel index structure to index the reachability, graph pattern and social contexts in the compressed graphs.

A. 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.

Example 10: Fig. 6 is an example of our index for the SSC in domain \( j \) of the graph depicted in Fig. 3. From the figure, we can see the the indices of each vertex include three parts: they are 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, e.g., \((B,C)\) of \( G_{Q3} \) in Fig. 1, if the query vertices are included in the SSC, we can investigate the reachability immediately, greatly saving query processing time.

B. Graph Pattern Index

After indexing the reachability information, we further index the graph pattern information to improve the efficiency of graph pattern queries. This index records the shortest path length between any two vertices in the graph of an SSC.

Example 11: Consider the graph pattern index shown in Fig. 6. For vertex \( E \), in addition to index 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 vertices.

Given a query of a graph pattern with the bounded length, e.g., \((A,D)\) of \( G_{Q4} \) in Fig. 2, based on the graph pattern index, we can investigate if the indexed path length is greater than the bounded length, and thus can efficiently answer a query.

C. Social Context Index

In order to improve the efficiency of MC-GPM, 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 each of the aggregated \( T \), \( r \) and \( \rho \) values of one of the paths between two vertices dominates others, we index that path length and the corresponding aggregated social impact factor values.
- Otherwise, we index up to three paths that have the maximal aggregated \( T \), \( r \) and \( \rho \) values respectively.

Example 12: Consider the social context index shown in Fig. 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 \( AS(p1(C,E,H)) \) dominates \( AS(p2(C,F,H)) \). Then, we index \( AS(p1(C,E,H)) \)={0.96,0.96,0.94} 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, e.g., \((B,D)\) of \( G_{Q4} \) in Fig. 2, based on the social context index, we can quickly investigate if there exists an edge pattern matching in the data graph, and thus saving query processing time.

D. Summary

The above three indices record important information of the graph in an SSC. Given an MC-GPM query, if the two vertices of a query edge can be mapped into a path in SSCs, this indexed information can be used to quickly investigate if there is an edge pattern matching, and thus greatly saving query processing time (see details in the experiments). In addition, in the worst case, we need to perform the Dijkstra’s algorithm four times, and thus the time complexity of the indices construction is \( O(N_D \log N_D + E_D) \). Furthermore, as mentioned in Section IV, the structure and the social contexts of the graph in an SSC usually stay stable in a very long period of time. Therefore, usually it is not necessary to update the indices frequently, which reduces the cost of index maintenance. Moreover, the index update methods have been well developed in the literature [16], [20]. Thus, we would not discuss it in detail.

VII. AN MULTI-CONSTRAINED GRAPH PATTERN MATCHING ALGORITHM

Based on the compressed data graph and the indices in SSCs, we propose a novel heuristic algorithm for MC-GPM, called HAMC, with our proposed novel heuristic search strategies. HAMC firstly performs a Multi-Constraint Edge
Algorithm 2: MC-EPM

Data: \((v, v')\), \((v, v') \in G_D\), \(G_D\), \(G_Q\)
Result: \((v, v', v) \in G_D\) and \(G_Q\)

1 begin
2 while \(\exists v^*, v^*.\text{visit} = \text{false}\) do
3 \quad Compute \(\text{mind}(v^*, v'^*)\);
4 \quad if \(\text{mind}(v^*, v'^*) > 1\) then
5 \quad \quad return \((v^*, v'^*)\);
6 \quad else
7 \quad \quad \(v^*.\text{visit} = \text{true}\);
8 \quad return \(\emptyset\);
9 end

Algorithm 3: Exploration-based graph pattern matching

Data: \(G_D, G_Q\)
Result: \(G_M\)

1 begin
2 \(G_M = \emptyset\) and \(G_{temp} = \emptyset\);
3 while \(\exists (v, v') \in G_Q\), \(v, v'\).\text{visit} = \text{false}\) do
4 \quad if \(MC - EPM(s_v, v_j) \neq \emptyset\) then
5 \quad \quad \(s_v, v_j).\text{explore} = \text{true};\)
6 \quad \quad \(G_{temp} = (V, E, LV, LE), V = \{v_a, v_b\}\) and \(E = \{(v_a, v_b)\};\)
7 \quad \quad \(G_M = G_M \cup G_{temp};\)
8 \quad \quad \(G_{temp} = \emptyset;\)
9 \quad \quad \text{if} \(v_j\) is not a leaf vertex then
10 \quad \quad \quad \(s_v, v_j).\text{explore} = \text{false and}\)
11 \quad \quad \quad \(MC - EPM(s_v, v_j) \neq \emptyset\) then
12 \quad \quad \quad \(v_a, v_b).\text{explore} = \text{true};\)
13 \quad \quad \quad \(G_{temp} = (V, E, LV, LE), V = \{v_a, v_b\}\) and \(E = \{(v_a, v_b)\};\)
14 \quad \quad \quad \(G_M = G_M \cup G_{temp};\)
15 \quad \quad \quad \(G_{temp} = \emptyset;\)
16 \quad \quad \quad \text{else}\)
17 \quad \quad \quad \(v_j, v_m).\text{explore} = \text{false and}\)
18 \quad \quad \quad \(MC - EPM(v_j, v_m) \neq \emptyset\) then
19 \quad \quad \quad \(v_a, v_b).\text{explore} = \text{true};\)
20 \quad \quad \quad \(G_{temp} = (V, E, LV, LE), V = \{v_a, v_b\}\) and \(E = \{(v_a, v_b)\};\)
21 \quad \quad \quad \(G_M = G_M \cup G_{temp};\)
22 \quad \quad \quad \(G_{temp} = \emptyset;\)
23 \quad \quad \quad return \(G_M;\)
24 end

Pattern Matching (MC-EPM) based on the novel objective function to investigate if there is an edge pattern matching in the data graph. If an answer is returned, HAMC then performs an exploration-based GPM to answer an MC-GPM query.

A. Multi-Constrained Edge Pattern Matching

Multi-Constrained Edge Pattern Matching (MC-EPM) method first investigates if an edge pattern query in \(G_Q\) can be mapped into a path in SSCs. If so, MC-EPM checks the indices of each vertex in an SSC to determine if there is an edge pattern matching. Otherwise, it performs the Dijkstra’s algorithm to investigate the minimal value of the objective function in Eq. (1), which can help find an edge matching if one exists in \(G_D\).

\[
\delta(p) \triangleq \max\{\frac{\lambda_{D_i}}{A_{p_i}}, \frac{\lambda_{r}}{A_{p_r}}, \frac{\lambda_{D_j}}{A_{p_j}}, (\frac{\lambda_{D_i}}{A_{p_i}}) \cdot (\frac{\lambda_{D_j}}{A_{p_j}}) \cdot (\frac{\lambda_{D_i}}{A_{p_i}}) \}
\]

where \(A_{D_i}, A_{r}, A_{p_j}\) are the aggregated social impact factor values of path \(p; \lambda_{D_i}, \lambda_{r}, \lambda_{p_j}\) are the corresponding constraints; \(Blen\) and \(Blen\) are the bounded path length and the identified path length respectively.

From the objective function, we can see that if an edge pattern query can be mapped into a path \(p\) in a data graph, \(\delta(p) \leq 1\). Otherwise \(\delta(p) > 1\). Based on this property, HAMC adopts the Dijkstra’s algorithm to identify the path with the minimal \(\delta\) value (denoted as \(\delta_{\text{min}}\)). If \(\delta_{\text{min}}(p) \leq 1\), there is an edge pattern matching. The pseudo-code of MC-EPM is shown in Algorithm 2.

Example 13: Consider the example shown in Fig. 2, where the multiple social impact factor constraints and the bounded length are given to edge \((B, D)\) in \(G_Q\). In order to return an edge pattern matching answer in \(G_{D3}\), MC-EPM computes \(\delta_{\text{min}}(B, D) = 1\) in \(G_{D5}\). Therefore, there is an edge pattern matching in \(G_{D5}\) for \((B, D)\) in \(G_Q\). The path with edges \((B, C_2), (C_2, C_1)\) and \((C_1, D)\) has \(\delta_{\text{min}}\), and thus \((B, C_2, C_1, D)\) is another path in \(G_{D5}\) that \(\delta_{\text{min}}(p) \leq 1\).

Below \(\text{Theorem 6}\) establishes that HAMC is an effective algorithm in MC-GPM.

\(\text{Theorem 6: HAMC can return an edge pattern matching answer if one exists in the data graph.}\)

\(\text{Proof:}\) Assume \(G_Q = (V, E), (v_i, v_j) \in E\) is an edge pattern query. Let \(p^*\) be a path from \(v_i\) to \(v_j\) in \(G_D\) with the minimal \(\delta\) at \(v_i\) returned by the HAMC, and \(p^*\) is another path between \(v_i\) and \(v_j\) in \(G_D\), where \((v_i, v_j, G_Q) \simeq (p^*, G_D)\). Then, assume \((v_i, v_j, G_Q) \neq (p^*, G_D)\), then \(\exists p \in \{T, r, \rho\}\) that \(\lambda_{D_i} < \lambda_{p_i} < \frac{\lambda_{D_i}}{A_{p_i}}\) and \(\text{Plen}(p^*) > \text{Plen}(p^*) > \text{Blen}(v_i, v_j)\). Hence, \(\delta(p^*) > 1\). Since \(p^*\) is an edge pattern matching, then \(\delta(p^*) = 1\) and \(\delta(p^*) > \delta(p^*)\). This contradicts \(\delta(p^*) < \delta(p^*)\). Therefore, \((v_i, v_j, G_Q) \simeq (p^*, G_D)\). \(\text{Theorem 6}\) is proven.

Summary: MC-EPM is the first part of HAMC, which is effective in MC-GPM as it can return an edge pattern matching answer if one exists in \(G_D\). In addition, in the worst case there is no vertex included into any SSCs of \(G_D\). Then, for an edge pattern query in \(G_Q\), HAMC employs a Dijkstra’s algorithm for \(M\) pairs of vertices in \(G_D\). Therefore, the time complexity of MC-GPM is \(O(MN_{DlogN_D} + ME_D)\).

B. Exploration-Based Graph Pattern Matching

In the literature, there are two popular methods to answer a GPM query based on the edge pattern matching. They are the join-based method [2], [9], [10] and the exploration-based method [16], [17]. The join-based method aims to find a maximal matching that contains all matching subgraphs in a data graph, while the exploration-based method aims to quickly answer a GPM query.

As MC-GPM is NP-Complete, it is computationally infeasible to find all the matching subgraphs in \(G_D\). In order to quickly answer an MC-GPM query, we propose
an Exploration-Based Graph Pattern Matching (EB-GPM) method, presented below.

**Step 1:** Starting from a source vertex (a vertex with indegree zero, denoted as $s_v$), EB-GPM firstly returns an edge pattern matching result based on the above introduced MC-EPM.

**Step 2:** Mark the matching edge as explored and investigate if the end point of the edge (denoted as $e_p$) is a leaf vertex in the query.

- If the end point of the edge is a leaf vertex, EB-GPM rolls back to the start point of the edge (denoted as $s_p$) and matching another unmatched edge starting from $s_v$ in $G_Q$.
- Otherwise, EB-GPM continues to investigate another unexplored edge from $e_p$.

**Step 3:** If $s_p = s_v$, and each of the edge pattern queries in $G_Q$ corresponds to an explored edge in $G_D$, an MC-GPM query answer is returned.

**Example 14:** Consider the MC-GPM example in Fig. 2, which contains two source vertices, $A$ and $B$. Then from $A$, MC-EPM returns an edge pattern matching result in $G_{D5}$ as $(A, C_1, D, G_{D5}) \approx (A, D, G_{Q4})$. As $D$ is a leaf vertex in $G_{Q4}$, EB-GPM rolls back to source vertex $A$ to investigate if an edge from $A$ has not been explored in $G_{Q4}$. As $(A, C)$ is unexplored, MC-EPM returns another edge pattern matching result in $G_{D5}$ as $(A, C_1, D, G_{D5}) \approx (A, C, G_{Q4})$. Since all the edges from $A$ are explored, EB-GPM then completes the same process from the other source vertex $B$ by MC-EPM. Then EB-GPM can return an MC-GPM answer as $G_M = (V, E, LV, LE)$, where $V \in \{A, B, C_1, C_2, D\}$ and $E = \{(A, C_1), (B, C_2), (C_2, C_1), (C_1, D)\}$.

If there are more than one MC-GPM results included in a data graph, we can use EB-GPM to return other results by replacing one of the explored matching edge with another unexplored matching edge in the data graph.

The pseudo-code of EP-GPM method is shown in Algorithm 3. EP-GPM performs $E_Q$ times of MC-EPM methods. The time complexity of HAMC is $O(E_QM_NDlogND + ME_QED)$. But $E_Q$ and $M$ have an inverse relation as $M = \frac{E_D}{E_Q}$ [4]. Namely, when $E_Q$ has a large value, e.g., $E_Q = E_D$, $M = 1$, and vice versa. Therefore, the time complexity of our HAMC is $O(E_DNDlogND + E_QED)$.\n
**C. Summary**

Our proposed HAMC algorithm is an efficient and effective method for the NP-Complete MC-GPM problem in large-scale contextual social graphs. Our method achieves in $O(E_DNDlogND + E_QED)$ computation cost. Moreover, if the matching edges are included into the graphs of SSCs, HAMC achieves the outstanding $O(E_Q)$ computation cost.

**VIII. EXPERIMENTS**

We conduct experiments on five large-scale real-world social graphs to evaluate (1) the performance our algorithm in answering MC-GPM queries; (2) the effectiveness of our index for SSC in improving the efficiency of MC-GPM; and (3) the effectiveness of our graph compression approach in reducing memory usage and improving the efficiency of MC-GPM.

**A. Experiment setting**

**Datasets:**

We use five large-scale real-world 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 I.

**Graph Pattern Query and Parameter Setting:**

- As we discussed in Section III, the social context impact factor values (i.e., $T$, $r$ and $\rho$) 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 value 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.

- We use a simple MC-GPM query that has the same graph structure as the graph in Fig. 6. Moreover, a set of relative low constraints are specified as $\lambda^{D} = 0.05, \lambda_{p} = 0.05, \lambda_{r} = 0.2$, and $Blen = 4$. These settings ensure the high possibility of returning MC-GPM answers in a data graph. Otherwise, no or only few answers might be returned by all the algorithms, making it difficult to investigate their performance.

**Implementation:**

As we discussed in Section II, there is no existing GPM method in the literature for the MC-GPM problem. Therefore, in the experiments, (1) we first implement BaseLine extended from bounded simulation [2] to illustrate a baseline performance of MC-GPM; (2) we then implement our HAMC algorithm without graph compression (denoted as HAMC-NoCompression) to investigate the influence of indices on MC-GPM; (3) we implement HAMC algorithm with both indices and graph compression to further investigate the influence of graph compression on MC-GPM; and (4) As returning all the MC-GPM answers included in a $G_D$ is NP-Complete [34], we compare the performance of three algorithms in finding a certain number of answers.

All BaseLine, HAMC-NoCompression and HAMC algorithms are implemented using Visual C++ running on a PC with Intel Core i5-3470 3.20GHz CPU, 16GB RAM, Windows 7 operating system and MySql 5.6 database. All the experimental results are averaged based on five independent runs.

<table>
<thead>
<tr>
<th>Name</th>
<th>Vertices</th>
<th>Edges</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epinions</td>
<td>75,837</td>
<td>508,837</td>
<td>A trust-oriented social network</td>
</tr>
<tr>
<td>DBLP</td>
<td>317,080</td>
<td>1,049,866</td>
<td>A co-author relationship network</td>
</tr>
<tr>
<td>Youtube</td>
<td>1,134,890</td>
<td>2,987,624</td>
<td>A video recommendation social network</td>
</tr>
<tr>
<td>Pokec</td>
<td>1,632,903</td>
<td>80,622,584</td>
<td>A general online social network</td>
</tr>
<tr>
<td>LiveJournal</td>
<td>4,847,571</td>
<td>68,993,773</td>
<td>A general online social network</td>
</tr>
</tbody>
</table>
Exp-1: Effectiveness. This experiment is to investigate the effectiveness of our HAMC by comparing the average number of GPM answers returned by the three methods. 

Results: Fig. 7 depicts the average number of GPM answers returned by each of BaseLine, HAMC-NoCompression and HAMC after visiting 1,000 mapped paths on the five datasets. From the figure, we can see that (1) the number of answers returned by BaseLine is always less than that of HAMC-NoCompression and HAMC; and (2) HAMC returns the same number of answers as HAMC-NoCompression. Statistically, on average, HAMC and HAMC-NoCompression can return 37.8% more answers than BaseLine. Thus, they are more effective than BaseLine.

Analysis: The experimental results illustrate that (1) BaseLine ineffectively visits many paths that do not corresponding to any matching edge in GPM, while based on Theorem 6, our HAMC and HAMC-NoCompression can return an edge pattern matching result if one exists in a data graph, showing they are more effective in answering MC-GPM queries than BaseLine; and (2) As illustrated in Theorem 5, our compression method is reachability preserved, graph pattern preserved and social context preserved. Therefore, HAMC and HAMC-NoCompression can visit the same mapped path based on the same MC-EPM, thus they can return the same number of answers based on EP-GPM.

Exp-2: Efficiency. This experiment is to investigate the efficiency of our HAMC by (1) comparing the average query processing time of the three methods for outputting different numbers of answers, and (2) comparing their average query processing time under different numbers of SSCs when returning 100 query answers.

Result-1: Fig. 8 depicts the average query processing time of BaseLine, HAMC-NoCompression and HAMC in returning the answers (i.e., $G_M$) of the MC-GPM query on the five datasets. From the figure, we can see that (1) when the number of answers increases, the total average query processing time of the three methods increases for datasets; (2) BaseLine spends more query processing time than HAMC-NoCompression and HAMC in all the cases; and (3) HAMC has the best efficiency for the MC-GPM in all the five datasets. Statistically, on average, the query processing time of HAMC is 21.6% less than that of BaseLine, and 5.7% less than that of HAMC-NoCompression.

Analysis-1: Result-1 illustrates (1) BaseLine needs more query processing time than others as it spends much time in each edge pattern query; (2) our novel heuristic search strategy and the proposed index structure can greatly save the query processing time, as some of the edge pattern matching can be returned via searching indexed information; and (3) the proposed graph compression reduces the index search space, and thus HAMC is more efficient than the other two methods.

Result-2: Fig. 9 depicts the average query processing time when returning 100 answers for the MC-GPM query with different numbers of SSCs on the five datasets. From the figure, we can see that (1) with the increase of the number of SSC, the average query processing time of both HAMC-NoCompression and HAMC decreases in all the cases; (2) BaseLine needs more query processing time than others even at SSC = 20 on the five datasets; and (3) under a certain number of SSCs, HAMC has the minimal query processing time among all the three methods. The detailed experimental results are listed in Table II. Statistically, on average, the query processing time of HAMC is 21.4% less than that of BaseLine and 4.2% less than that of HAMC-NoCompression.

Analysis-2: Result-2 illustrates (1) the large query processing time of BaseLine is consistent with the first analysis in Analysis-1; (2) the more the SSCs, the higher the probability of finding the matched edges based on the indexed information, and thus the query processing time decreases with the increase of the number of SSC; and (3) as analysed in Analysis-1, our graph compression reduces the index search space, and thus HAMC has the minimal query processing time of all methods, which is consistent with Analysis-1.

Exp-3: Scalability. This experiment is to investigate the scalability of our HAMC by comparing the average query processing time of returning a query answer in social graphs with different scales.

Results: Fig. 10 depicts the average query processing time of the MC-GPM query under different scales of social graphs with a certain number of SSCs. From the figure, we can see that (1) both HAMC-NoCompression and HAMC scales better than BaseLine; (2) The larger the number of SSCs, the better the scalability of our method, and (3) HAMC has the best scalability among all the three methods. The detailed results

### Table II: The Comparison of the Average Query Processing Time Between HAMC and Others with Different Number of SSC

<table>
<thead>
<tr>
<th>SSC</th>
<th>Comparison with BaseLine</th>
<th>Comparison with HAMC-NoCompression</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>13.2% less</td>
<td>4.67% less</td>
</tr>
<tr>
<td>40</td>
<td>19.89% less</td>
<td>4.76% less</td>
</tr>
<tr>
<td>60</td>
<td>24.06% less</td>
<td>5.60% less</td>
</tr>
<tr>
<td>80</td>
<td>27.64% less</td>
<td>6.00% less</td>
</tr>
<tr>
<td>100</td>
<td>32.19% less</td>
<td>4.27% less</td>
</tr>
</tbody>
</table>

### Table III: The Comparison of the Average Query Processing Time Between HAMC and Others on Five Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Comparison with BaseLine</th>
<th>Comparison with HAMC-NoCompression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epinions</td>
<td>8.11% less</td>
<td>2.09% less</td>
</tr>
<tr>
<td>DBLP</td>
<td>16.9% less</td>
<td>7.82% less</td>
</tr>
<tr>
<td>Youtube</td>
<td>35.88% less</td>
<td>8.84% less</td>
</tr>
<tr>
<td>Pokec</td>
<td>39.67% less</td>
<td>5.59% less</td>
</tr>
<tr>
<td>LiveJournal</td>
<td>34.18% less</td>
<td>2.81% less</td>
</tr>
</tbody>
</table>
are listed in Table III. The scalability experiment illustrates that (1) HAMC has the best scalability in all the three methods, which is consistent with the time complexity analysis, i.e., $O((|V|+|E|)\log|V|+|E|)$ for the other two methods; (2) the size of the original data graph; has the best scalability as graph compression can reduce the size of the original data graph; and (3) HAMC has the best scalability as graph compression can reduce the size of the original data graph.

Analysis: The scalability experiment illustrates that (1) HAMC has the best scalability in all the three methods, which is consistent with the time complexity analysis, i.e., $O((|V|+|E|)\log|V|+|E|)$ for the other two methods; (2) the size of the original data graph; has the best scalability as graph compression can reduce the size of the original data graph; and (3) HAMC has the best scalability as graph compression can reduce the size of the original data graph.

Exp 4: Memory Usage. This experiment is to investigate the memory usage of three methods under different numbers of SSCs on the five datasets. From the figure, we can see that (1) with the increase of the number of SSCs, the memory usage of both HAMC and HAMC-NoCompression increases; (2) the memory usage of HAMC is lower than that of HAMC-NoCompression in most cases; and (3) HAMC-NoCompression consumes less memory than HAMC in the average. The memory usage of HAMC can save 1.577 KB memory than HAMC-NoCompression for a single dataset.

Analysis: The experimental results of memory usage illustrate that (1) with the increase of the number of SSCs, the memory usage of both HAMC and HAMC-NoCompression increases; (2) the memory usage of HAMC is lower than that of HAMC-NoCompression in most cases; and (3) HAMC-NoCompression consumes less memory than HAMC in the average. The memory usage of HAMC can save 1.577 KB memory than HAMC-NoCompression for a single dataset.
trate that (1) HAMC-NoCompression needs to index the graph structures and social context in an SSC, thus it consumes more memory than BaseLine; (2) HAMC compresses the graph in an SSC, thus can reduce the memory of storing and indexing the vertices and edges that are removed by the graph compression method. In addition, the larger the number of SSC, the higher the probability of compressing more vertices and edges. Thus HAMC consumes less memory than HAMC-NoCompression; and (3) All the social impact factors in an SSC are required to have high values. Usually, only a few vertices and edges can be included and compressed in an SSC, and thus the saved memory by graph compression is usually less than the usage of storing indices. So, in most of the cases of SSC, HAMC consumes more memory than BaseLine.

Summary:

The above experimental results have demonstrated that the proposed heuristic matching strategies adopted in HAMC provide an effective means to answer MC-GPM queries. In addition, with our proposed index structure and graph compression methods, HAMC can greatly save query processing time, which makes HAMC significantly outperform BaseLine in effectiveness, efficiency and scalability with low memory usage. Therefore, our HAMC is a very competitive algorithm for the new NP-Complete MC-GPM problem in social network based applications.

IX. CONCLUSION

In this paper, we have proposed a new Multi-Constrained Simulation (MCS) to support a new type of Multi-Constrained Graph Pattern Matching (MC-GPM) that is a corner stone for many social network based applications. Then, we have developed a novel concept, strong social component, upon which we have designed a novel index structure and a context-preserved graph compression method. Finally, we have proposed a heuristic algorithm, HAMC employs our novel heuristic matching strategies for the NP-Complete MC-GPM problem. HAMC achieves \( O(E_D N_D \log N_D + E_Q E_D) \) in time cost, and the experiments conducted on five real-world large-scale social graphs have demonstrated the superiority of our proposed approaches in terms of effectiveness, efficiency, scalability and memory cost.

ACKNOWLEDGEMENT

This work was partially supported by Natural Science Foundation of China (Grant Nos. 61232006, 61303019, 61003044), Australian Research Council DP (DP140103171), Doctoral Fund of Ministry of Education of China (201332011200012), and Collaborative Innovation Center of Novel Software Technology and Industrialization, Jiangsu, China.

---

<table>
<thead>
<tr>
<th>SSC</th>
<th>HAMC</th>
<th>HAMC-NoCompression</th>
<th>Save</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>4,649,427 KB</td>
<td>4,650,041 KB</td>
<td>614 KB</td>
</tr>
<tr>
<td>40</td>
<td>4,649,917 KB</td>
<td>4,650,848 KB</td>
<td>931 KB</td>
</tr>
<tr>
<td>60</td>
<td>4,653,297 KB</td>
<td>4,652,779 KB</td>
<td>1,288 KB</td>
</tr>
<tr>
<td>80</td>
<td>4,658,315 KB</td>
<td>4,652,647 KB</td>
<td>1,924 KB</td>
</tr>
<tr>
<td>100</td>
<td>4,651,275 KB</td>
<td>4,653,153 KB</td>
<td>1,878 KB</td>
</tr>
</tbody>
</table>

---

TABLE IV

The comparison of memory usage between HAMC and HAMC-NoCompression with different numbers of SSC

REFERENCES