MCS-GPM: Multi-Constrained Simulation Based Graph Pattern Matching in Contextual Social Graphs

Guanfeng Liu, Yi Liu, Kai Zheng, An Liu, Zhixu Li, Yang Wang, *Senior Member, IEEE*, and Xiaofang Zhou, *Fellow, IEEE*

Abstract—Graph Pattern Matching (GPM) has been used in lots of areas, like biology, medical science, and physics. With the advent of Online Social Networks (OSNs), recently, GPM has been playing a significant role in social network analysis, which has been widely used in, for example, finding experts, social community mining, and social position detection. Given a query which contains 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 end-to-end constraints of the social contexts, like social relationships, social trust, and social positions on edges in G_Q, which are commonly found 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 identifying SSCs, and propose a novel index method and a graph compression method for SSC. Moreover, we devise a multithreading heuristic algorithm, called M-HAMC, to bidirectionally search the MC-GPM results in parallel without decompressing graphs. An extensive empirical study over five real-world large-scale social graphs has demonstrated the effectiveness and efficiency of our approach.

Index Terms—Graph pattern matching, social graph

1 Introduction

1.1 Background

Graph Pattern Matching (GPM) has been widely used in social network analysis [1], [2], [3], which 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. However, as shown in [2], the conventional subgraph isomorphism problem is too strictly defined to find useful patterns in real-world social graphs. Moreover, due to the NP-complete time complexity, it is hard to apply graph 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. In graph simulation, a matching of an edge in a query graph could be a path in a data graph, if the start vertex and the end vertex of the path have the same label with the start vertex and the end vertex of the edge respectively. 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 edges (A, B) 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 predefined 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 the $*$, illustrating the requirement of the matching path length is no greater than k or no requirement for the path length respectively. Graph matching in terms of the 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.

1.2 Problem and Challenges

The bounded simulation based GPM only considers the bounded path length of an edge when matching the edges,

- *G. Liu, Y. Liu, A. Liu, and Z. Li are with the School of Computer Science and Technology, Soochow University, Suzhou, Jiangsu 215000, China. E-mail: {gfliu, anliu, zhixuli}@suda.edu.cn.*
- *K. Zheng is with the University of Electronic Science and Technology of China. E-mail: zhengkai@uestc.edu.cn.*
- *Y. Wang is with the Department of Computing, Macquarie University, Sydney, NSW 2109, Australia. E-mail: yan.wang@mq.edu.au.*
- *X. Zhou is with the School of Information Technology and Electrical Engineering, The University of Queensland, St Lucia, QLD 4072, Australia. E-mail: zxf@itee.uq.edu.au.*

Manuscript received 15 May 2017; revised 28 Oct. 2017; accepted 11 Dec. 2017. Date of publication 21 Dec. 2017; date of current version 27 Apr. 2018. (Corresponding author: Kai Zheng.)
Recommended for acceptance by P. Cui.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2017.2785824

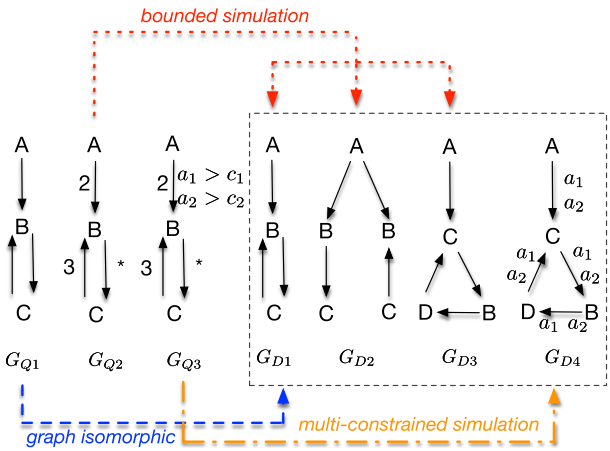


Fig. 1. Pattern graphs and data graphs.

which greedily finds the subgraph that has the minimal diameter. However, social graphs have many social contexts associated with vertices and edges, like the Contextual Social Graph (CSG) [7], where each vertex has the social role information in a specific domain (e.g., *a professor* in *data mining*), and each edge has the social relationship between participants (e.g., a father and his son) and the social trust information between participants (e.g., Tom trusts Bob in car repairing). In a variety of GPM based applications in social graphs, e.g., traveller selection in crowd-sourcing travel [8], study group selection (classroomsalon.com), and the expert selection in social graphs [7], in addition to the bounded path length, people are willing to incorporate the constraints of the intimacy social relationship and social trust between people in the identified subgraph in terms of GPM in a CSG, which have significant influence on people's collaborations and decision making [9].

Example 1. 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_1 and a_2 that can represent the *social trust* and the *social relationships* between people in CSGs. In real applications on CSGs, the constraints of social trust value and social intimacy 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 a social network based CRM (Customer Relation Management) system. In addition, the constraints of the social intimacy and the social trust can be specified between a traveler A and an accommodation provider B in crowdsourcing travel [8] to find trustworthy travel groups, or between students in a study group selection (classroomsalon.com) to find a trustworthy study group. 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 attribute a_1 and attribute a_2 must satisfy the *multiple constraints*, i.e., $a_1 > c_1$ and $a_2 > c_2$ (c_1 and c_2 are constants).

This example illustrates that a new type of *Multi-Constrained Simulation (MCS)* [10] is significant in many GPM based applications in social graphs. The relations between these different GPM methods are shown in Fig. 2. GPM with multiple constraints needs to match each vertex with multiple constraints given in a query graph to a path in a data graph, which covers the Multi-Constrained Path (MCP) problem which is NP-Complete [11], [12]. The

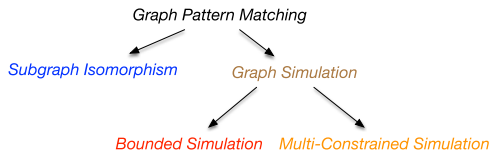


Fig. 2. The relations between different types of GPM.

traditional bounded simulation based GPM method supports only one constraint, i.e., the bounded path length, in matching. When facing multiple constraints on edges, it has to enumerate all possible matchings of each constraint and then find the intersections of the matchings, which is very time consuming. The detailed discussions of the drawbacks of the bounded simulation based GPM in MC-GPM will be given in Section 7. In addition, the existing methods for Regular Path Query (RPQ) in graphs [13], [14] deliver a path between vertices with a specified regular expression on the edges, where only one expression needs to be satisfied for each query without considering a multiple regular expressions for different attributes at the same time, and thus cannot be applied into the MC-GPM. Our previous algorithm HAMC [10] is an effectiveness and efficiency method for answering the NP-Complete MC-GPM query. However, as indicated in [15], [16], the less the sum of the path lengths in a GPM result, the better the quality of the result. HAMC did not consider the path length of the returned result, and thus usually cannot deliver high quality answer. Therefore, it is critical to develop an effective and efficient method to find high quality MC-GPM result, i.e., the find the pattern graph with the minimal the sum of the path length of all the edges, which subsumes the classical NP-Complete multi-constrained optimal path selection problem [11], and thus is NP-Complete as well. Our contributions are summarized as follows.

1.3 Contributions

(1) We first propose a new notion of *Multiple-Constrainted 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 then propose a concept called Strong Social Component (**SSC**), which consists of participants who have strong social connections, and propose an approach to identifying **SSC**s. As the social connections in **SSC** usually stay stable in a very long period of time [17], 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 Multithreading Heuristic Algorithm for the MC-GPM, called M-HAMC. In M-HAMC, based on a novel objective function, we propose a bidriectional search method to bidirectionally investigate if a match is included in a data graph, and find the matching result with the minimal bounded path length. M-HAMC has the time complexity of $\mathcal{O}(E_Q N_D log N_D + E_Q E_D)$, 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 over five large-scale real-world social graphs has demonstrated the superiority of our proposed M-HAMC in effectiveness and efficiency than the most promising existing algorithm, HAMC [10] in answering MC-GPM queries.

The rest of this paper is organized as follows. We first review the related work on GPM in Section 2. Then we introduce the necessary concepts and formulate the focal problem of this paper in Section 3. The identification of strong social components is presented in Section 4, followed by the graph compression methods and index structures proposed in Sections 5 and 6, respectively. Section 7 presents our proposed M-HAMC algorithm, Section 8 reports the experimental findings, and Section 9 concludes the paper.

2 Related Work

Based on the properties of the graph pattern matching strategies, the existing studies can be categorized into (1) isomorphism-based GPM to match each of the vertices and edges exactly in G_D, and (2) simulation-based GPM to simulate the vertices and edges pattern matching in G_D. Below we analyze each of the categories in detail.

2.1 Isomorphism-Based GPM

In order to reduce the complexity of the isomorphism-based GPM, this type of methods usually precompute some graph structure information to build up edge index, frequent subgraph index and/or reachability index. For example, Cheng et al., [18] propose an R-Join index structure that index the nodes within 2-hops away. In [19], Zou et al., propose a Distance-Join graph pattern matching method, where a constraint of the shortest distance between two nodes can be given in a query graph. Sun et al., [20] propose an efficient subgraph matching method based on subgraph isomorphism in large-scale web graphs. This method adopts a graph exploration method to improve the efficiency of subgraph joint processing in graph matching. Furthermore, in order to improve the efficiency of finding the top-k answers that have the top-k shortest edge lengths, Cheng et al., [21] propose a query optimization approach where they build 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 G_Q, usually it is not realistic to find a subgraph (or a few subgraphs) in G_D that contains the whole query graph. Then, Yan et al., [22] propose a *similarity based GMP method*. In their model, a distance is defined to compute the total number of the matched edges in the Maximal Common Subgraph between the query graph and the database graphs. If the distance is less than a specified threshold, then the subgraph is returned as a solution. Shang et al., [23] further improve the similarity-based GPM by proposing an index method, called GrafD-Index, to index graphs according to their similarity to the features in G_Q. Furthermore, Zhu et al., [24] propose a method, where they divide a G_D into several groups of similar graphs and index these graphs to support effective pruning, which can improve the efficiency of similarity based GPM.

Although the Distance Join GPM and similarity-based GPM methods relax the strict constraints in subgraph isomorphism, and increase the probability of finding a match, they are still NP-Complete [2]. In order to efficiently find matching subgraphs in large data graphs, some recent works adopt parallel and distributed GPM methods. For example, In [25], [26] a pattern graph is decomposed into small ones (edges), and the subgraphs can be extracted for each small pattern graph and join the intermediate results finally. In addition, in [27], they find the matched subgraphs based a parallel framework in a large data graph. Furthermore, in [28] a large data graph is decomposed into several small fragments based on a distributed GPM method.

The isomorphism-based GPM is important in many applications, e.g., 3D object matching [29] and protein structure matching [30]. However, such GPM suffers from expensive computation cost as it is NP-Complete.

2.2 Simulation-Based GPM

In order to relax the constraints of isomorphism-based GPM to meet the GPM requirements in some real applications, based on the *graph simulation* [31], Fan et al., [2] propose a bounded simulation in GPM. In their model, the label of each vertex is not unique and a bounded length can be specified on the edge between two vertices in query graphs. Then, the bounded simulation will deliver the matching whose any vertex having the same labels and the edges within the bounded length. This type of GPM can be conducted in cubic-time. Based on bounded simulation, Ma et al., [32] propose a *strong simulation*, where a GPM not only meets the requirements of the bounded length, but also preserves the topology of a query graph. It is to find a small set of matches whose topologies are more similar with the query graph than that in bounded simulation. In addition to the labeled vertices in bounded simulation, Fan et al., [33] further consider the requirements of different types of edges in GPM, and develop a social expert finding system based the bounded simulation GPM [3]. In order to improve the efficiency of the simulation-based GPM, Fan et al., [34] propose a graph pattern view based bounded simulation. In their model, a set of views are defined in a data graph, and they develop a method to estimate which view can be used to answer a specific query. In addition, they propose a resource-bounded query [35] where a fraction of a data graph that has a high probability of containing the query graph is extracted. In the application of community finding, Fang et al., [36] propose a method which aims to return an attributed community for an attributed graph, in which the attributed community is a subgraph which satisfies both structure cohesiveness and keyword cohesiveness. In [37], Yang et al., study the problem of diversified subgraph querying in a large graph, which is to find k subgraphs that match a given query graph with the maximum coverage based on the graph simulation. In [38], Fan et al., propose incremental algorithms for four types of typical pattern graphs, these incremental algorithms can reduce the incremental computations on big graphs to small data and minimize unnecessary re-computation.

Summary. 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 [8], study group selection (classroomsalon.com), and social network based e-commerce [7]. Therefore, the existing methods cannot support the significant multi-constrained graph pattern matching in many applications.

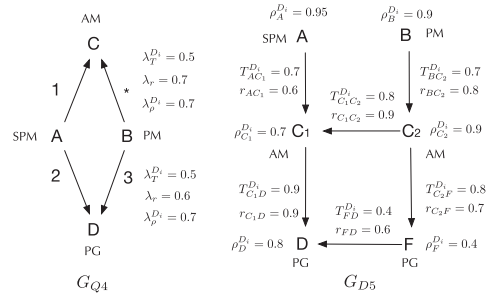


Fig. 3. Multiple-Constrained GPM in CSGs.

3 PRELIMINARIES

3.1 Data Graph

3.1.1 Contextual Social Graph

A Contextual Social Graph [7] 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 edge (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 2. G_{D5} in Fig. 3 is a CSG, where each vertex $v_i \in V$ is associated with a *role impact factor*, denoted as $\rho_{v_i}^{D_i} \in [0, 1]$, to illustrate the impact of participant v_i in domain i, which is determined by the expertise of v_i. $\rho_{v_i}^{D_i} = 1$ indicates that v_i is a domain expert in domain i while $\rho_{v_i}^{D_i} = 0$ indicates that v_i has no knowledge in that domain. Moreover, each edge (v_i, v_j) is associated with *social trust*, denoted as $T_{v_i,v_j}^{D_i} \in [0, 1]$, 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 ρ are called *social impact factors*, whose values can be extracted by using the data mining techniques [39], [40], [41].

Based on theories in *Social Psychology* [17], 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 [7]. The aggregated values of a path p in domain i is denoted as $\mathcal{AS}^{D_i}(p) = < AT^{D_i}(p), Ar(p), A\rho^{D_i}(p) >$.

Definition 1 (Path Domination). *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 $p \geqslant_{\mathrm{DOM}}^{D_i} p'$.*

3.2 Pattern Graph

A *Pattern Graph* is defined as a tuple $G_Q = < V_q, E_q, f_v, f_e, s_e >$, where

- V_q and E_p are the set of vertices and the set of directed edges, respectively;
- f_v is a function defined on V_q such that for each vertex u, $f_v(u)$ is the vertex label of u;
- f_e is a function defined on E_q such that for each edge (u, u'); $f_e(u, u')$ is the bounded length of (u, u') which is either a positive integer k or the symbol *;
- s_e is a function defined on E_q such that for each edge (u, u'), $s_e(u, u')$ is the multiple constraints of the aggregated social impact factor values of (u, u') represented by a tuple of $< \lambda_T^{D_i}, \lambda_r, \lambda_\rho^{D_i} >$, which are in the scope [0,1];

From G_{Q4} in Fig. 3, we can see the constraints, i.e., $\lambda_T^{D_i}, \lambda_r$ and $\lambda_\rho^{D_i}$ on edges (B, C) and (B, D), respectively. The value of these weights can be specified by users to illustrate their different requirements in different applications. For example, in the domain of crowdsourcing travel, a user could give a large value to $\lambda_r^{D_i}$ if he/she believes the social relationships between people are more important, while in the domain of employment, a user could give a large value to $\lambda_\rho^{D_i}$ if he/she believes the social impact of a people is more important.

3.3 Multi-Constrained Graph Pattern Matching (MC-GPM)

In this section, we introduce MC-GPM via Multi-Constrained Simulation in CSGs.

Bounded Simulation [2]. Given a data graph $G = (V, E, LV)$ and a pattern graph $Q = (V_q, E_q, f_v, f_e)$, a data graph G matches a pattern graph Q via *bounded simulation*, denoted as $Q \trianglelefteq_{\mathrm{sim}}^{\mathrm{B}} G$, if there exists a binary relation $S \subseteq V_Q \times V$ such that

- for all $u \in V_Q$, there exists $v \in V$ such that $(u, v) \in S$;
- for each pair $(u, v) \in S$,
  - $u \sim v$ (vertices u and v have the same table), and
  - for each edge (u, u') in E_Q, there exists a nonempty path p from v to v' in G such that $(u', v') \in S$, and the shortest path length $Slen(p) \leqslant k$ if $f_e(u, u') = k$.

Then S is a match in G for Q via bounded simulation.

Multi-Constrained Simulation. MCS is a nontrivial extension of bounded simulation. Consider a data graph $G_D = (V, E, LV, LE)$ and a pattern graph $G_Q = (V_q, E_q, f_v, f_e, s_e)$. G_D matches G_Q via MCS, denoted by $G_Q \trianglelefteq_{\mathrm{sim}}^{\mathrm{MC}} G_D$, if there exists a binary relation $S \subseteq V_Q \times V$ such that

- for all $u \in V_Q$,there exists $v \in V$ such that $(u, v) \in S$;
- for each pair $(u, v) \in S$,
  - $u \sim v$, and
  - for each edge (u, u') in E_Q, there exists a nonempty path p from v to v' in G such that $(u', v') \in S$, and $Slen(p) \leqslant k$, if $f_e(u, u') = k$;
  - $AT^{D_i}(v, v') \geqslant \lambda_T$, $Ar(v, v') \geqslant \lambda_r$ and $A\rho^{D_i}(v, v') \geqslant \lambda_\rho$, if $s_e(u, u') = \{\lambda_T, \lambda_r, \lambda_\rho\}$;

Then S is a match in G_D for G_Q via *multi-constrained simulation*.

If an edge (u, u') in G_Q is mapped to a nonempty path p from v to v' in G_D based on MCS, then (v, v') is an *edge pattern matching* (u, u') in G_D (denoted as $(v, v', G_D) \simeq (u, u', G_Q)$), and $(u, v) \in S$. If for each edge in G_Q, there is a matching edge in G_D, then an MC-GPM answer is returned (denoted as $G_M = (V, E, LV, LE)$, $G_M \subseteq G_D$).

Example 3. Suppose G_{Q4} in Fig. 3 is a query given by a user to select a group of participants from a CSG to finish a project. Based on data graph G_{D5}, we can get the MC-GPM answer as (1) vertex SPM (i.e., A, a Senior Project Manager) and vertex PM (i.e., B, a Project Manager) in G_{Q4} can be mapped to the same vertices SPM and PM in G_{D5}, which is part of *subgraph isomorphism*; (2) the vertex AM (i.e., C, an Assistant Manager) in G_{Q4} corresponds to multiple AMs (i.e., C_1 and C_2) in G_{D5}. This relationship

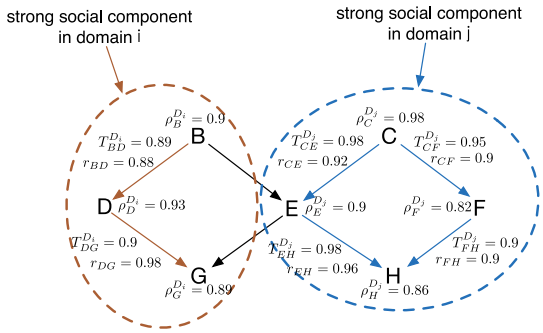


Fig. 4. An example of a strong social component.

can be captured by using *graph simulation*; (3) the edge with a bounded length in G_{Q4} can be mapped to a path length in G_{D5} by using *bounded simulation*; and (4) the edge with multiple constraints can be mapped to the aggregated social impact factor values of a path in G_{D5} by using *multi-constrained simulation*.

4 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 [42], 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.

Example 5. In an **SSC**, suppose the T, r and ρ values associated with each of the vertices and edges should be greater than 0.8. Fig. 4 depicts a graph that has two strong social components in domain i and domain j respectively.

Based on the theories in *Social Psychology* [17], 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***s* in a specific domain subsumes the classical NP-Complete maximum clique problem [42], which is very time consuming. Alternatively, we can identify up to K **SSC***s* for MC-GPM by first randomly selecting K vertices that are associated with high role impact factor values as the seeds, and then adopting 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 $\mathcal{O}(N_D E_D)$.

5 Context-Preserved Graph Compression for SSC

In this section, based on the existing graph compression method for bounded simulation [43], 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.

5.1 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 two compressed 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 $V = \{A, B_1, \ldots, B_n, C_1, \ldots, C_n, D_1, D_2\}$ and $E = \{(A, B_1), \ldots, (A, B_n), (A, C_1), \ldots, (A, C_n), (B_1, D_1), \ldots, (B_n, D_1), (C_1, D_2), \ldots, (C_n, D_2), (D_1, D_2), (D_2, D_1)\}$. $G_D^R = (V, E)$, where is $V = \{A, B_{1\ldots n} C_{1\ldots n}, D_1, D_2\}$ and $E = \{(A, B_{1\ldots n} C_{1\ldots n}), (B_{1\ldots n} C_{1\ldots n}, D_1), (D_1, D_2)\}$. For a reachability query $G_D^R = (V, E)$, where $V = \{A, B_i, D_j\}$, $i \in [1, n]$ and $j \in [1, 2]$, G_D^R is not reachability preserved compression if and only if A and B_i, or B_i and D_j are not reachable to each other, which contradicts $E = \{(A, B_{1\ldots n} C_{1\ldots n}), (B_{1\ldots n} C_{1\ldots n}, D_1), (D_1, D_2)\}$ in G_D^R. Therefore, *Theorem 1 is proven.* □

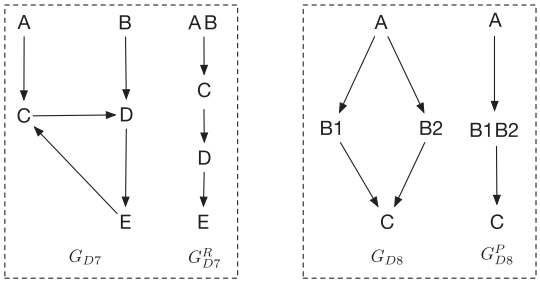
Example 6. Fig. 5 contains two groups of graphs.[1] Consider Fig. 5a, where G_{D7} is the original data graph and G_{D7}^R is the compressed graph. From G_{D7}, we can see that both A and B do not have any ancestors, and they can reach the same descendants (C, D and E). Therefore, A and B can be compressed as one vertex in G_{D7}^R, where the reachability of A and B to other vertices is preserved.

5.2 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. 3, we propose a graph compression method that can preserve such graph patterns, which is called *graph pattern preserved compression*, denoted as G_D^P. The property of the compression method is captured by *Theorem 2*.

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

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.



(a) reachability preserved (b) graph pattern preserved

Fig. 5. Reachability preserved and graph pattern perserved compressions.

Proof. Suppose $B_1, \ldots, B_n$ 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 $B_1, \ldots, B_n$ are compressed as a single vertex B. If $G_Q = (V, E)$, where $V = \{A, B, C\}$ and $E = \{(A, B), (B, C)\}$, $G_Q \trianglelefteq^{\mathrm{B}}_{\mathrm{sim}} G_D$. 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 $(u', v') \notin S, (S \subseteq G_Q \times G_D)$. Namely, there exists vertex B_i such that $(A, B_i, G_D) \not\simeq (A, B, G^P_D)$ or $(B_i, C, G_D) \not\simeq (B, C, G^P_D)$. This contradicts the assumption that $B, \ldots, B_n$ have the same label. Therefore, *Theorem 2 is proven.* □

Example 7. Consider the example shown in Fig. 5 that contains a data graph G_{D8} and the corresponding compressed graph G^P_{D8}. In G_{D8}, 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_{D8} is a graph pattern preserved compression of G_{D8}.

We can see that the graph pattern preserved compression is reachability preserved as its compression condition is *more strict* than that of reachability preserved compression.

5.3 Compression for Social Contexts

In order to support MC-GPM, e.g., (B, D) of G_{Q4} in Fig. 3, 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 of the compression method is captured by *Theorem 3*.

Theorem 3. *The compressed graph is social context preserved when two compressed vertices have the same label, the same ancestors, the same descendants, and the aggregated social impact factor values of the path via one of the vertex dominates that of the other one.*

Proof. Suppose $B_1, \ldots, B_n$ have the same ancestor A and the same descendant C in G_D. $\mathcal{AS}(A, C)$ via B_i $(1 \le i \le n)$ dominates others. Then, G_D is compressed as $G^S_D = (V, E)$, where $V = \{A, B_1 \ldots B_n, C\}$, $E = \{(A, B_1 \ldots B_n), (B_1 \ldots B_n, C)\}$ and $\mathcal{AS}(A, C)$ via $B_1 \ldots B_n$ equals to $\mathcal{AS}(A, C)$ 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 $Slen(A, C) = 2$ via any B_k $(1 \le k \le n)$. If G^S_D is not social context preserved, then the edge matching $(A, C, G_Q) \simeq (A, C, G_D)$ is missing in G^S_D, namely one of the social impact factors in $\mathcal{AS}(A, C)$ via B_j $(1 \le j \le n,$ and $i \ne j)$ in G_D is *greater* than that of (A, C) in G^S_D (i.e., $\mathcal{AS}(A, C)$ via B_i). This contradicts the assumption that $\mathcal{AS}(A, C)$ via B_i $(1 \le i \le n)$ dominates others. Therefore, *Theorem 3 is proven.* □

Example 8. Consider the example shown in Fig. 6 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, $\mathcal{AS}(A, D)$ via C_1 *dominates* $\mathcal{AS}(A, D)$ via C_2. Namely, each of the aggregated social impactor values of the path from A to D via C_1 is greater than the corresponding social impact value of the path from A to 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 $\mathcal{AS}(A, D)$ via C_1C_2 in G^S_{D9} equals to the dominated one in G_{D9}, i.e., $\mathcal{AS}(A, D)$ via C_1. Then G^S_{D9} is social context preserved.

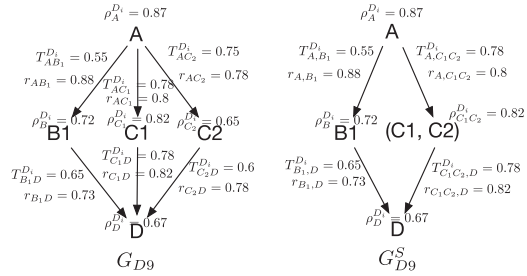


Fig. 6. Social context preserved compression.

The social context preserved compression is graph pattern preserved and reachability preserved as its compression condition is more *strict* than that of graph pattern preserved compression.

6 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 compressed graphs.

6.1 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, building the reachability index is not computationally expensive [44].

Example 9. Fig. 7 is an example of our index for the **SSC** in domain j of the graph depicted in Fig. 4. From the figure, we can see that the indices of each vertex include three parts: 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

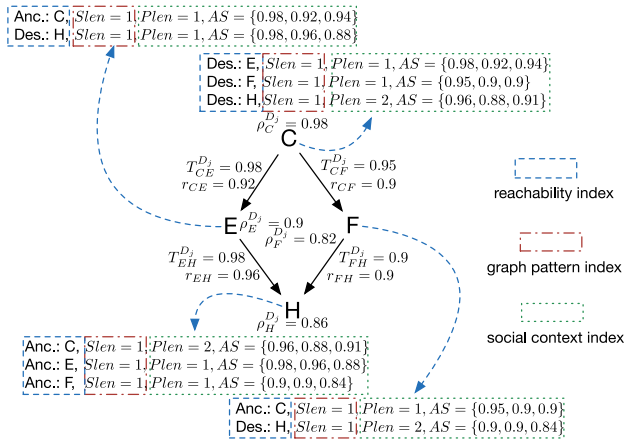


Fig. 7. The index of an **SSC** in domain j.

(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.

6.2 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 10. In Fig. 7. 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., $Slen = 1$). Given a query of a graph pattern with the bounded length, e.g., (A, D) of G_{Q4} in Fig. 3, 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.

6.3 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 ρ 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 ρ values respectively.

Example 11. In Fig. 7, we take 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 $\mathcal{AS}(p1_{(C,E,H)})$ dominates $\mathcal{AS}(p2_{(C,F,H)})$, we index $\mathcal{AS}(p1_{(C,E,H)})$ = {0.96,0.88,0.91} and its path length $Plen(p1_{(C,E,H)}) = 2$ at C. Given a graph pattern query with multiple constraints, e.g., (B, D) of G_{Q4} in Fig. 3, based on the social context index, we can quickly investigate if there exists an edge pattern match in the data graph, and thus saving query processing time.

6.4 Summary

The above three indices record important information of the graph in an **SSC**, which can be used to quickly investigate if there is an edge pattern match, 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 index construction is $\mathcal{O}(N_D log N_D + E_D)$. Furthermore, as mentioned in Section 4, the structure and the social contexts of the graph in an **SSC** usually stay stable in a very long period of time [17]. 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 [43] to first establish the matrices of the shortest path length, the ancestors and descendants, and the aggregated social impact factor values between vertices. Then if an edge is removed from or added into an SSC, we could check the matrix to update the shortest paths information that is affected by the change of reachability due to the change of edges. Then based on the updated shortest path information, we could update the social context information in the indices. The index maintenance in dynamic graphs is another challenging research topic and thus it is not discussed in this paper.

7 MC-GPM Algorithm

7.1 Our Previous HAMC Algorithm

In our previous work [10], we have proposed an approximation algorithm called HAMC, which supports the NP-Complete MC-GPM. However, HAMC has the following disadvantages, motiving us to develop a new Multithreading Heuristic Algorithm for the MC-GPM, called M-HAMC.

Disadvantage 1. Although HAMC supports MC-GPM, it does not support the distributed computing structure. Thus HAMC cannot adopt multi-core processors to parallel processing the NP-Complete MC-GPM that has exponential time complexity, and thus it can hardly deliver good efficiency in MC-GPM.

Disadvantage 2. As indicated in [15], [16], the less the sum of the length of the matching paths, the better the quality of the GPM result. However, HAMC considers the constraints of the matching path only without taking care of minimizing the matching path length, and thus it can hardly deliver the GPM results with good quality.

In order to overcome the above mentioned disadvantages in HAMC, based on the compressed data graph and the indices in **SSC**s, we propose a novel Multithreading Heuristic Algorithm for the MC-GPM, called M-HAMC, with our proposed novel heuristic search strategies. M-HAMC supports multithreading processing, and looks for the good quality of the GPM results by considering the matching path length. In experiments, we will investigate the performance of HAMC and M-HAMC in solving the MC-GPM problem.

7.2 Algorithm Overview

Based on the compressed and indexed SCC structures in data graphs, our proposed M-HAMC first (1) finds the matching from the data graph for each of the edges in a query graph, and then (2) joins the matching of each edge based on the topology of the query graph.

- First, in each of the edge matching, M-HAMC first bidirectionally performs the *Feasible Edge Pattern Matching (F-EPM)* procedure in parallel based on the novel objective function in Eq. (1) proposed in the below Section 7.3. This procedure can investigate if there is an edge matching in the data graph which satisfies the multiple constraints on the edges given in the query graph.
- Second, as indicated in [15], [16], the less the length of the matched edge, the better the quality of the edge matching result. Therefore, after finding the matching by F-EPM, M-HAMC then bidirectionally performs the *Optimal Edge Pattern Matching (O-EPM)* procedure proposed in the below Section 7.4 to find the edge matching with the minimal bounded path length in the data graph.
- Finally, M-HAMC will link the above delivered matching results together to return a GPM result. In the literature, there are two popular methods to answer a GPM query based on edge matching. In

order to quickly answer an MC-GPM query, we propose an Exploration-Based Graph Pattern Matching (EB-GPM) method in Section 7.5 to combine these edge matching results.

Below are the details of each of the procedures of M-HAMC.

7.3 Feasible Edge Pattern Matching (F-EPM)

Feasible Edge Pattern Matching (F-EPM) performs the *forward search process* from the start vertex and the *backward search process* from the end vertex in parallel to investigate if an edge pattern query in G_Q can be mapped into a path in the data graph G_D. The details of F-EPM are as follows,

Step 1: From the start vertex (denoted as v_s) and the end vertex (denoted as v_t), F-EPM bidirectionally performs the Dijkstra's algorithm to deliver the path with the minimal value of the objective function in Eq. (1).

Step 2: F-EPM records the aggregated social impact factor values, and the path length of the paths with the minimal δ value from v_s and v_t delivered by the forward and backward search processes respectively.

Step 3: For the forward search process, if a vertex has not been accessed by the *backward search process*, F-EPM continues the search based on Dijkstra's algorithm. Otherwise, it records the aggregated social impact factor values of the path from the current expansion vertex to v_t, and stops continuing search for the vertex accessed by the backward search process. When no further vertex can be selected as a forward expansion vertex, this process terminates. The backward search process performs the similar process and terminates when no further vertex can be selected as a *backward expansion vertex*.

Step 4: When both the forward and backward search processes terminate, F-EPM terminates.

$$\delta(p) \triangleq max\left\{\left(\frac{\lambda_{T_p}^{D_i}}{AT_p^{D_i}}\right), \left(\frac{\lambda_{r_p}}{Ar_p}\right), \left(\frac{\lambda_{\rho_p}^{D_i}}{A\rho_p^{D_i}}\right), \left(\frac{Plen}{Blen}\right)\right\}, \quad (1)$$

where $AT_p^{D_i}$, Ar_p and $A\rho_p^{D_i}$ are the aggregated social impact factor values of path p; $\lambda_{T_p}^{D_i}$, λ_{r_p} and $\lambda_{\rho_p}^{D_i}$ are the corresponding constraints; $Blen$ and $Plen$ 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, then $\delta(p) \leqslant 1$. Otherwise $\delta(p) > 1$. Based on this property, M-HAMC adopts Dijkstra's algorithm to identify the path with the minimal δ value (denoted as δ_{min}). If $\delta_{min}(p) \leqslant 1$, there is an edge pattern match. The pseudo-code of MC-EPM is shown in *Algorithm 1*.

Example 12. Based on the query edge (B, D) in G_{Q4} in Fig. 3 and the data graph G_{D5} in Fig. 8, F-EPM bidirectionally computes the minimal δ value from the start vertex B and the end vertex D. After one step of forward search, as C_2 is the only decendent of B, and $\delta(p(B, C_2)) = 0.78$, C_2 is selected as a forward expansion vertex and at C_2, F-EPM records $\mathcal{AS}^{D_i}(p(B, C_2)) = \{AT^{D_i}(p(B, C_2)) = 0.7$, $Ar(p(B, C_2)) = 0.8$, $A\rho^{D_i}(p(B, C_2)) = 0.9\}$. At the same time, after one step of backward search, we obtain $\delta(p(C_1, D)) = 0.875$ and $\delta(p(F, D)) = 1.25$. Then C_1 is selected as the *backward expansion vertex*, and at C_1, F-EPM records $\mathcal{AS}^{D_i}(p(C_1, D)) = \{AT^{D_i}(p(C_1, D)) = 0.9$, $Ar(p(C_1, D)) = 0.9$, $A\rho^{D_i}(p(C_1, D)) = 0.8\}$. F-EPM continues the search from the two expansion vertices C_1 and C_2 in parallel, and after the second search step, as vertex A does not have any outgoing edges, and all the rest of vertices, including C_1, C_2 and F are accessed by both of the forward and backward search process, F-EPM terminates. Then the path $p(B, C_2, C_1, D)$ with $\delta(p(B, C_2, C_1, D)) = 1$ is identified as the the one with the minimal δ value between B and D.

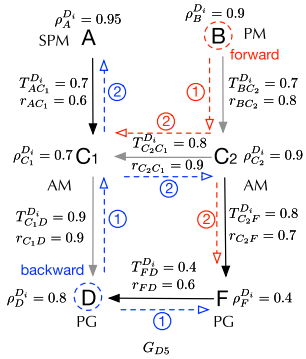


Fig. 8. The process of F-EPM.

Below *Theorem 4* illustrates that F-EPM is an effective method to find feasible EPM in MC-GPM.

Theorem 4. *F-EPM process can return an edge pattern match if one exists in the data graph.*

Proof. Assume $G_Q = (V, E)$, and $(v_i, v_j) \in E$ is an edge pattern query. Let p^* be a path from v_i to v_j in G_D with the *minimal* δ at v_i returned by the M-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 $(v_i, v_j, G_Q) \not\simeq (p^*, G_D)$, then $\exists \varphi \in \{T, r, \rho\}$ that $\mathcal{AS}\varphi_{p^*} < \lambda_{\varphi(v_i,v_j)}^{D_i}$ or $Plen(p^*) > Blen(v_i, v_j)$. Hence, $\delta(p^*) > 1$. Since p^{**} is an edge pattern match, then $\delta(p^{**}) \leq 1$ and $\delta(p^*) > \delta(p^{**})$. This contradicts $\delta(p^*) \leq \delta(p^{**})$. Therefore, $(v_i, v_j, G_Q) \simeq (p^*, G_D)$. *Theorem 4 is proven.* □

7.4 Optimal Edge Pattern Matching (O-EPM)

As there can be many paths matching an edge in a data graph, and the less the path length of an edge match, the better of the quality of the edge pattern match [15], [16]. In our M-HAMC, if there is a feasible edge pattern match in a data graph (i.e., $\delta_{min} \leqslant 1$), we perform the *Optimal Edge Pattern Matching (O-EPM)* method to bidirectionally find an edge pattern match in parallel by minimizing the bounded path length. The details of O-EPM are as follows,

Step 1: Start from v_s and v_t, O-EPM bidirectionally performs Dijkstra's algorithm to deliver the shortest path.

Step 2: O-EPM investigates the aggregated social impact factor values of the two *foreseen paths* for the current two expansion vertices v_i and v_j identified by O-EPM from v_s and v_t respectively. One is the combination of the current path with the shortest path length identified by the O-EPM from v_s (denoted as $SP_{v_s}(v_i)$), and the path that is saved at v_i identified by F-EPM from v_t (denoted as $FP_{v_t}(v_i)$). The other one is formed by $SP_{v_t}(v_j)$ and $FP_{v_s}(v_i)$. At this step, if both of the two searches visit the same vertex, the aggregated social impact factor values combined with each partial path are investigated to check whether the combined path is feasible.

Step 3: If the both of them are feasible edge pattern matching, O-EPM continues to searches the next vertex.

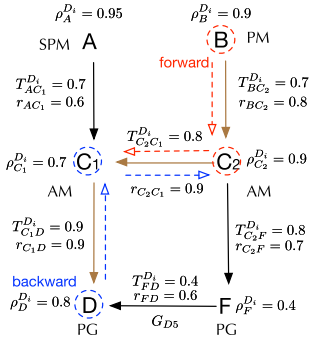


Fig. 9. The process of O-EPM.

Otherwise, O-EPM will find another expansion vertex from the neighbours of the current expansion vertex

Step 4: When there is an expansion vertex which has been selected by both the forward and backward search processes, O-EPM terminates. Then the path linked by the two paths identified by the two search processes is the query result of the edge pattern matching.

Example 13. In Fig. 9, O-EPM bidirectionally searches the graph from B and D in parallel. After one step search, C_1 and C_2 are selected as the expansion vertices of the forward and backward search respectively, and there can be two *foreseen paths*, one is $FS_{B,D}(C_1) = SP_B(C_1) + FP_D(C_1)$, and the other one is $FS_{D,B}(C_2) = SP_D(C_2) + FP_B(C_1)$. As they are feasible, O-EPM continues to bidirectionally search the shortest path between B and D from C_1 and C_2 respectively. Then C_2 and C_1 are selected as the next expansion vertices and $\delta(FS_{B,D}(C_2)) = 1$ and $\delta(FS_{D,B}(C_1)) = 1$. As both C_1 and C_2 are selected as the expansion vertices by the two processes, O-EPM terminates. The path $p(B, C_1, C_2, D)$ linking the source B and the target D is the optimal edge matching identified by O-EPM.

Below *Theorem 5* illustrates that O-EPM is an effective EPM optimization method in MC-GPM.

Theorem 5. *O-EPM process can return an edge pattern matching which is no worse than the one delivered by F-EPM.*

Proof. Assume $G_Q = (V, E)$, and $(v_i, v_j) \in E$ is an edge pattern query. Let $p^{\#}$ be a matching path delivered by O-EPM from v_i to v_j in G_D by using the Dijkstra's algorithm, and p^* is the matching path delivered by F-EPM. If there is only one path matching, based on *Theorem 4*, $p^{\#} = p^*$, and thus $Plen(p^{\#}) = Plen(p^*)$. Otherwise, if there are more than one matching path, based on the property of Dijkstra's algorithm, $p^{\#}$ is the shortest path starting from v_i where the corresponding constraints can be satisfied by the foreseen path ending at v_i. If $Plen(p^{\#}) > Plen(p^*)$, there is another matching path p^{**} delivered by F-EPM to be combined into the foreseen path, namely, $\delta(p^{**}) < \delta(p^*)$, which contradicts $\delta(p^*)$ has the minimal δ value. Therefore, *Theorem 5 is proven.* □

Summary. MC-EPM is the first part of M-HAMC, which is effective in MC-GPM as it can return an edge pattern match if one exists in G_D. In addition, M-HAMC employs Dijkstra's algorithm for M pairs of vertices in G_D. Therefore, the time complexity of MC-GPM is $\mathcal{O}(MN_D logN_D + ME_D)$. The pseudo code of MC-EPM algorithm is shown in *Algorithms 1, 2 and 3.*

7.5 Exploration-Based Graph Pattern Matching

In the literature, there are two popular methods to answer a GPM query based on edge pattern matching. They are the *join-based method* [2], [15], [34] and the *exploration-based method* [20], [21]. The join-based method aims to find a maximal match that contains all matching subgraphs in a data graph, while the exploration-based method aims to quickly answer a GPM query.

Algorithm 1. MC-EPM

Data: $(V, V'), (V, V') \in G_Q, G_D$
Result: $(V^*, V^{**}), (V^*, V^{**}) \in G_D$ and $(V^*, V^{**}, G_D) \simeq (V, V', G_Q)$
1 **begin**
2 $path = \emptyset$;
3 $F\text{-}EPM(G, V^*, V^{**})$;
4 $path = O\text{-}EPM(G, V^*, V^{**})$;
5 **return** $path$;
6 **end**

Algorithm 2. F-EPM

Data: V^*, V^{**}, G
Result: $dist\_z, dist\_f$
1 **begin**
2 **Spawn**
3 Initialize-Single-Source (G, V^*) //initialize the predecessors;
4 $S_1 = \emptyset$; $Q_1 = V[G]$; $dist\_f = \infty$;
5 **repeat**
6 u_1 = Extract-Min-Delta $(Q_1, dist\_f)$;
7 // extract the vertex with the minimal delta value from Q_1
8 $S_1 = S_1 \bigcup \{u_1\}$;
9 **for** *each vertex* $v_1 \in Adj[u_1]$ *and* $v_1 \notin S_2$ **do**
10 **Spawn**
11 $dist\_f$ = Update $(u_1, v_1, dist\_f)$;
12 // an update step to update the minimal δ value at $dist\_f$
13 **end**
14 **Sync**
15 **until** $Q_1 \neq \emptyset$ *and* $u_1 \neq null$ *and* $u_1 \notin S_2$;
16 **Spawn**
17 Initialize-Single-Source (G, V^{**})
18 $S_2 = \emptyset$; $Q_2 = V[G]$; $dist\_z = \infty$;
19 **repeat**
20 u_2 =Extract-Min-Delta $(Q_2, dist\_z)$;
21 $S_2 = S_2 \bigcup \{u_2\}$;
22 **for** *each vertex* $v_2 \in Adj[u_2]$ *and* $v_2 \notin S_1$ **do**
23 **Spawn**
24 $dist\_z$ = Update $(u_2, v_2, dist\_z)$;
25 **end**
26 **Sync**
27 **until** $Q_2 \neq \emptyset$ *and* $u_2 \neq null$ *and* $u_2 \notin S_1$;
28 **Sync**
29 **return** $dist\_f, dist\_z$;
30 **end**

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: Start from a *source vertex* (a vertex with indegree zero, denoted as s_v), EB-GPM first returns an edge pattern match 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.

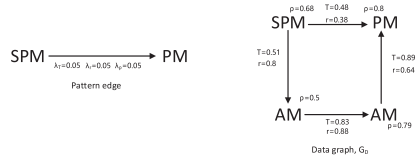


Fig. 10. A case study.

- 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 matches 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 pattern graph in Fig. 3, from vertex A, MC-EPM returns an edge pattern match in G_{D5} as $(A, C_1, D, G_{D5}) \simeq (A, D, G_{Q4})$. 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 match in G_{D5} as $(A, C_1, G_{D5}) \simeq (A, C, G_{Q4})$. 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 $V = \{A, B, C_1, C_2, D\}$ 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 edges with another unexplored matching edge in the data graph.

EP-GPM performs E_Q times of MC-EPM methods. The time complexity of M-HAMC is $\mathcal{O}(E_Q M N_D log N_D + M E_Q E_D)$. 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 M-HAMC is $\mathcal{O}(E_D N_D log N_D + E_Q E_D)$ which is the same as our previous HAMC method.

7.6 A Case Study

Fig. 10 contains a query edge and a contextual social graph. The query edge is from SPM (Senior Project Manager) to PM (Project Manager), where the constraints for social contexts are shown on the edge. In addition, the contextual social graph contains SPM, PM and two AMs (Assistant Manager). By using HAMC, we can get the edge matching as $SPM \rightarrow AM \rightarrow AM \rightarrow PM$, while M-HAMC will deliver the edge matching as $SPM \rightarrow PM$. Both of the two matchings are feasible. But based on the social psychology theories [17], the less the path length, the better the quality of a matching in social graphs. Namely, in this case, SPM and PM can better establish their collaboration relationships based on their direct interactions rather than the indirect interactions based on the path from SPM to PM via two AMs.

7.7 Summary

Our proposed M-HAMC algorithm is an efficient and effective method for the NP-Complete MC-GPM problem in large-scale contextual social graphs. Our method achieves $\mathcal{O}(E_D N_D log N_D + E_Q E_D)$ computation cost. Moreover, if the matching edges are included into the graphs of **SSC**s, M-HAMC achieves an outstanding computation cost in $\mathcal{O}(E_Q)$.

TABLE 1
The Pattern Graphs

Pattern ID	Vertices	Edges
1	5	6
2	10	12
3	15	18
4	20	24
5	25	30

Algorithm 3. O-EPM

Data: V^*, V^{**}, G
Result: $Path(V^*, V^{**}), \Pi$
1 **begin**
2 **Spawn**
3 Initialize-Single-Source (G, V^*);
4 $S_1 = \emptyset$; $Q_1 = V[G]$; $L_2 = \infty$;
5 **repeat**
6 u_1 = Extract-Min-Path (Q_1);
7 // extract the vertex with the minimal bounded path length from Q_1
8 **else**
9 **if** $dist\_z[u_1] < 1$ *or* $dist\_f[u_1] < 1$ **then**
10 $S_1 = S_1 \bigcup \{u_1\}$;
11 **for** *each vertex* $v_1 \in Adj[u_1]$ *and* $v_1 \notin S_2$ **do**
12 **Spawn**
13 UpdateLength (u_1, v_1, L_1);
14 **end**
15 **Sync**
16 **end**
17 **end**
18 **until** $Q_1 \neq \emptyset$ *and* $u_1 \neq null$ *and* $u_1 \notin S_2$;
19 **Spawn**
20 Initialize-Single-Source (G, V^{**});
21 $S_2 = \emptyset$; $Q_2 =$V[G]; $L_2 = \infty$;
22 **repeat**
23 $u_2 =$ Extract-Min-Path (Q_2);
24 // extract the vertex with the minimal value from Q_2
25 **if** $dist\_z[u_2] < 1$ *or* $dist\_f[u_2] < 1$ **then**
26 $S_2 \leftarrow S_2 \bigcup \{u_2\}$;
27 **for** *each vertex* $v_2 \in Adj[u_2]$ *and* $v_2 \notin S_2$ **do**
28 **Spawn**
29 UpdateLength (u_2, v_2, L_2);
30 **end**
31 **Sync**
32 **end**
33 **until** $Q_2 \neq \emptyset$ *and* $u_2 \neq null$ *and* $u_2 \notin S_1$;
34 **Sync**
35 **return** Get-Path (V^*, V^{**}, Π);
36 **end**

8 Experiments

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

8.1 Experiment Setting

Datasets. The five large-scale real-world social graphs we used are available at *snap.stanford.edu*, which have been widely used in the literature for graph pattern matching and

TABLE 2
The Social Datasets

Name	Vertices	Edges	Description
Epinions	75,879	508,837	A trust-oriented social network
DBLP	317,080	1,049,866	A co-author relationship network
Youtube	1,134,890	2,987,624	A video recommendation social network
Pokec	1,632,803	30,622,564	A general online social network
LiveJournal	4,847,571	68,993,773	A general online social network

TABLE 3
The Setting of λ_T, λ_r, and λ_ρ

Constraint ID	λ_T	λ_r	λ_ρ
1	0.015	0.015	0.015
2	0.025	0.025	0.025
3	0.05	0.05	0.05
4	0.075	0.075	0.075
5	0.01	0.01	0.01

social network analysis. The details of these datasets are shown in Table 2.

Graph Pattern Query and Paramater Setting.

- 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.
- 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 1. Moreover, a set of constraints are given in Table 3 from low to high values. Furthermore, the constraints of the bounded path length, *Blen*, is set as 4, 5 and 6 based on the *small-world* characteristic in social graphs [45].
- The setting of the number of threads depends on the number of available cores and the blocking coefficient of tasks. Usually, *Number of Threads = Number of Available Cores / (1 − Blocking Coefficient)*, where the blocking coefficient is between 0 and 1. In order to investigate the performance of M-HAMC under different numbers of threads on a PC with a Quad-Core Processor, we set the number of threads from 1 to 15.

Implementation. As we discussed in Section 2, there is no existing GPM method in the literature for the MC-GPM problem. Therefore, in the experiments, (1) we first implement our previous HAMC algorithm [10], which has been the most promising algorithm for MC-GPM; (2) we then implement our proposed M-HAMC algorithm to compare the effectiveness and efficiency with HAMC in MC-GPM; and (3) as returning all the MC-GPM answers included in a G_D is NP-Complete [46], we compare the performance of two algorithms in finding a certain number of answers.

All HAMC and M-HAMC algorithms are implemented using Scala 2.11 running on a PC with an Intel Core i5-3470 Quad-Core Processor 3.2 GHz, 16 GB RAM, Ubuntu14.04.1 operating system and MySql 5.6 database. All the experimental results are averaged based on five independent runs.

8.2 Experimental Results and Analysis

Exp-1: Effectiveness. This experiment is to investigate the effectiveness of our MC-GPM by (1) comparing the

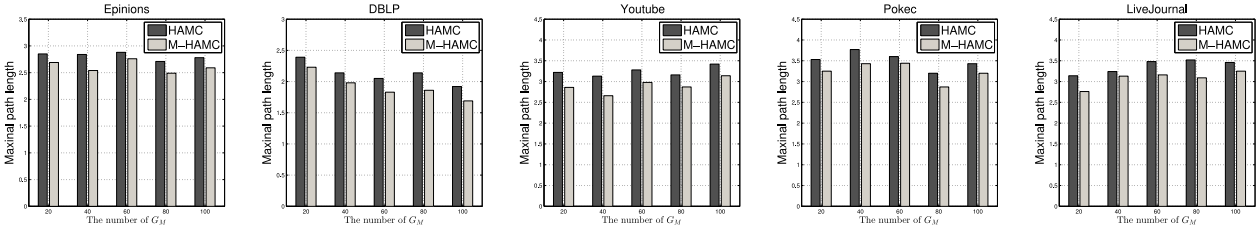


Fig. 11. The average maximal path length for different numbers of G_m.

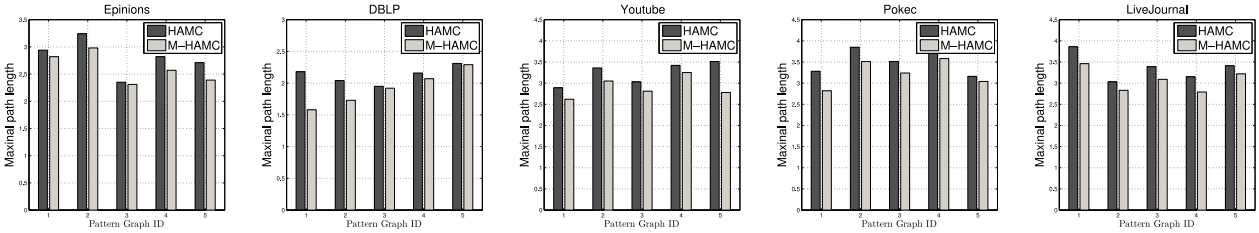


Fig. 12. The average maximal path length for different pattern graphs.

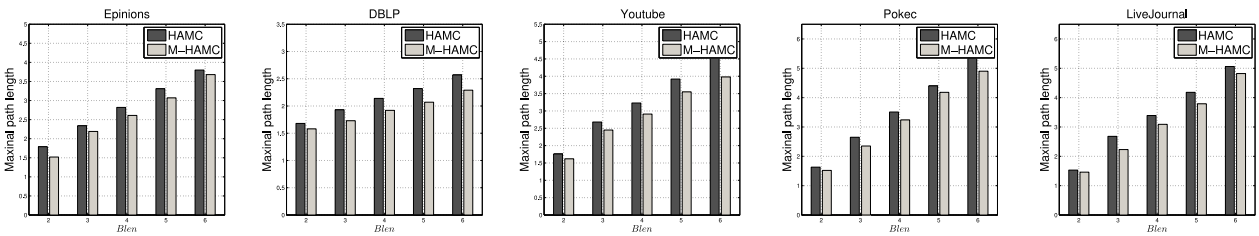


Fig. 13. The average maximal path length for different path lengths.

TABLE 4
The Average Maximal Path Length

Dataset	HAMC	M-HAMC	Comparison
Epinions	2.81	2.61	7.04% less
DBLP	2.13	1.92	9.87 % less
Youtube	3.24	2.90	10.49 % less
Pokec	3.51	3.24	7.64 % less
LiveJournal	3.37	3.08	8.61 % less

average path length for outputting different numbers of GPM answers, different pattern graphs, different number of bounded path lengths, different values of constrains, and (2) comparing the average sum of the path lengths of the GPM answers by the two methods.

Results. Figs. 11 to 14 depict the average maximal path length of all the edge pattern matching with different numbers of GPM answers, different pattern graphs and different bounded path lengths respectively, by each of HAMC and M-HAMC. From these figures, we can see that the average maximal path lengths returned by M-HAMC are always less than that of HAMC. The detailed experimental results are listed in Table 3. Statistically, on average, M-HAMC can return answers with a maximal bounded path length which is *8.73 percent less* than that of HAMC. In addition, Figs. 15 to 18 depict the averaged sum of the path length returned by M-HAMC and HAMC under different G_m, different pattern graphs and different bounded path lengths respectively. From these figures, we can see that the sum of the path lengths of the GPM answers returned by M-HAMC is always less than that of HAMC. The detailed experimental results are listed in Table 4 and Table 5. Statistically, on average, M-HAMC can return the answers with the sum of the path length which is *6.71 percent less* than that of HAMC. Thus, M-HAMC can return better quality GMP answers, and thus more effective than HAMC.

Analysis. The experimental results illustrate that (1) HAMC considers the feasibility of the MC-GMP only, but does not take the path length of the edge pattern match into consideration; and (2) as illustrated in on *Theorem 4* and *Theorem 5*, our M-HAMC can return an edge pattern match if there is one existing in a data graph, and M-HAMC considers to minimize the path length in answering the GPM query, which can effectively improve the quality of the query results.

Exp-2: Efficiency. This experiment is to investigate the efficiency of our MC-HAMC by (1) comparing the average query processing time of the two methods for outputting different numbers of answers, (2) under different pattern graphs, (3) under different number of bounded path lengths, (4) under different values of constrains, and (5) under different number of threads.

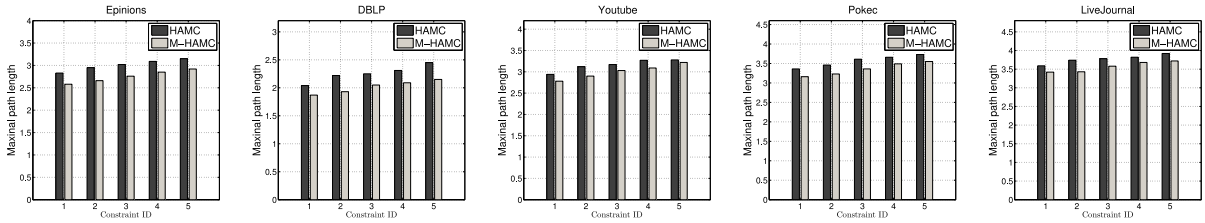


Fig. 14. The average maximal path length for different constraints.

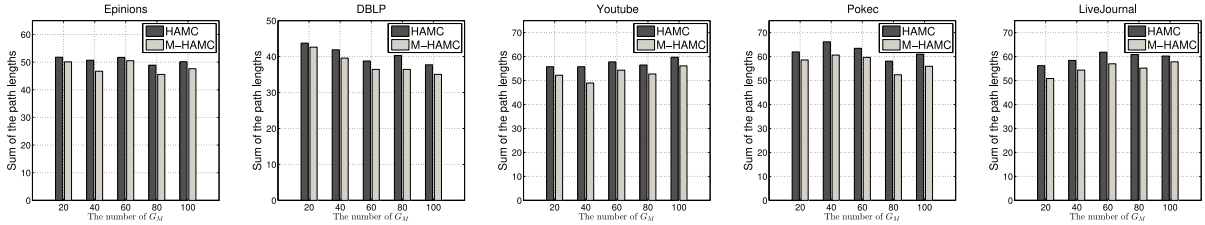


Fig. 15. The average sum of the path length for different numbers of G_m.

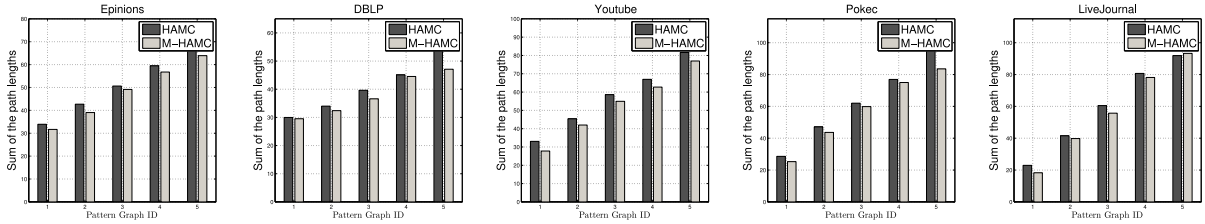


Fig. 16. The average sum of the path length for different pattern graphs.

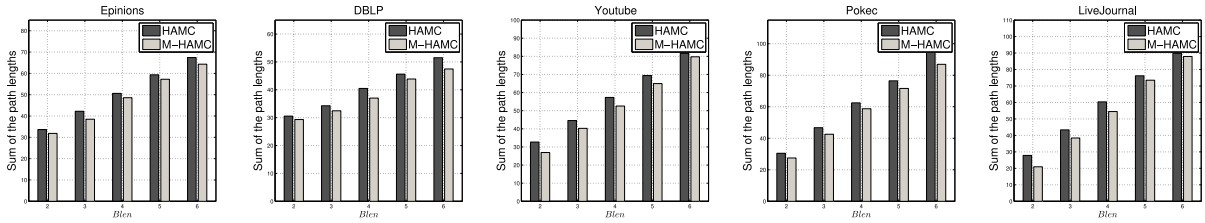


Fig. 17. The average sum of the path length for different path lengths.

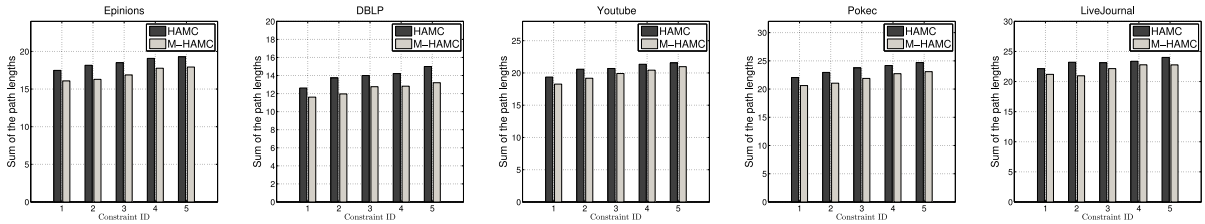


Fig. 18. The average sum of the path length for different constraints.

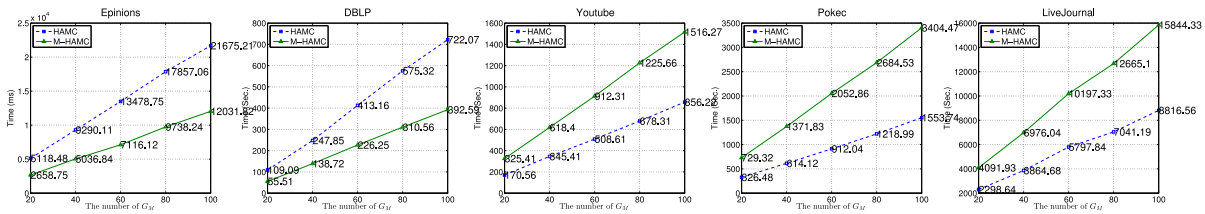


Fig. 19. The average query processing time of returning different numbers of G_m.

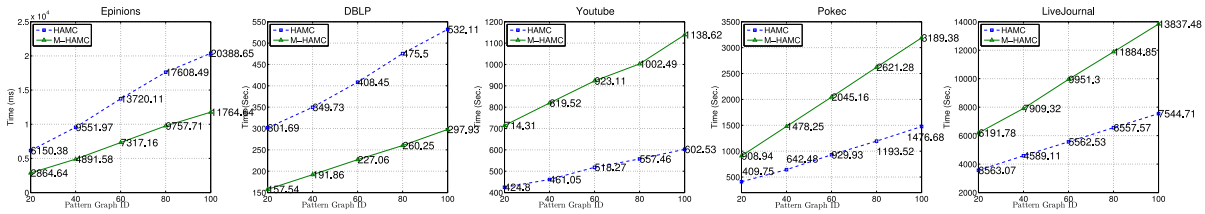


Fig. 20. The average query processing time of different pattern graphs.

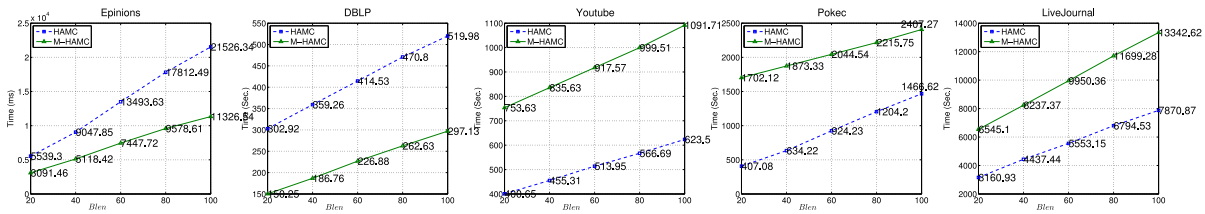


Fig. 21. The average query processing time of different path lengths.

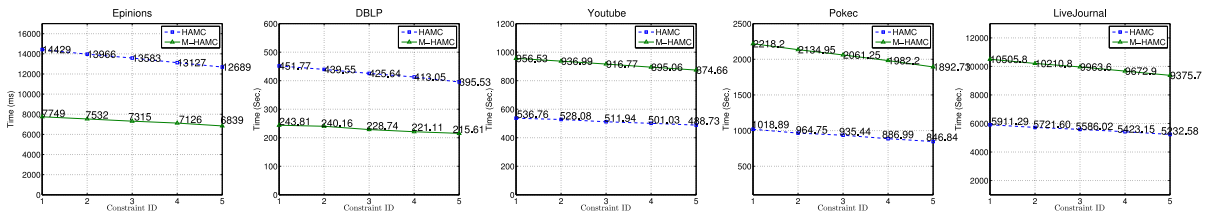


Fig. 22. The average query processing time of different constraints.

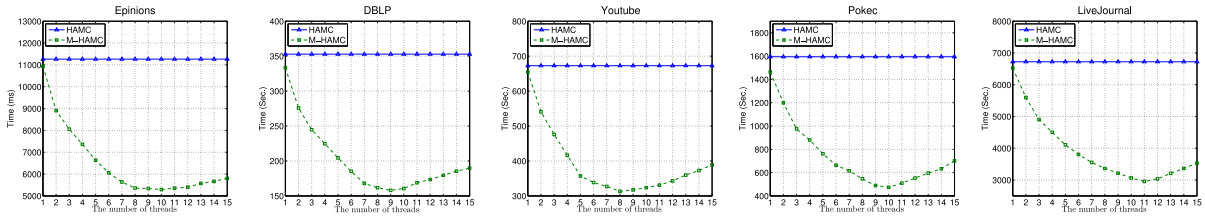


Fig. 23. The execution time with different numbers of threads.

Results. Figs. 19 to 23 depict the average query processing time of M-HAMC and HAMC in returning different numbers of answers (i.e., G_M), with different pattern graphs and different bounded path lengths respectively. From these figures, we can see that (1) when the number of answers increases, the total average query processing time of the two methods all linearly increases for with the increase of G_M, the scale of the pattern graph and the bounded path length; (2) with the increase of the number of threads, the execution time of M-HAMC can first decrease fast and increase when the number of threads is greater than 10; (3) M-HAMC has better efficiency than HAMC for the MC-GPM in all the cases in the five datasets. The detailed experimental results are listed in Table 6. Statistically, on average, the query processing time of M-HAMC is 46.8 percent less than that of HAMC.

TABLE 5
The Average of the Sum of Path Length

Dataset	HAMC	M-HAMC	Comparison
Epinions	50.65	48.10	5.04% less
DBLP	40.49	38.03	6.09% less
Youtube	57.12	52.89	7.42% less
Pokec	62.15	57.47	7.52% less
LiveJournal	59.52	55.05	7.50% less

TABLE 6
The Average Query Processing Time

Dataset	HAMC	M-HAMC	Comparison
Epinions	13.5	7.3	45.7% less
DBLP	413	226	45.3% less
Youtube	919	512	44.3% less
Pokec	2,048	930	54.6% less
LiveJournal	9,954	5,563	44.1% less

Analysis. The experimental results illustrate that (1) both M-HAMC and HAMC have the linear time complexity of the scale of the pattern graph, and thus they have good scalability; (2) with the increase of the number of threads, the efficiency can be improved due to the better usage of the capabilities of CPUs. But, due to the block and the switching between threads, the execution time of M-HAMC can increase after reaching a certain number of threads. This is consistent with the theory in multi-threading programming [47]; (3) with the increase of the value of constraints, the execution time of both M-HAMC and HAMC decreases. Because with the decrease of the number of feasible paths, the number of edges accessed by both M-HAMC and HAMC decreases, leading to less execution time; and (4) M-HAMC bidirectionally finds EPM in parallel for both F-EPM and O-EPM processes, which improves the efficiency of graph search. In addition, in the O-EPM of M-HAMC, the search terminates when both the forward and backward search processes access the same expansion vertex, which avoids visiting all the vertices and edges in a data graph. Thus, M-HAMC can greatly save the query processing time.

Summary. The above experimental results have demonstrated that the proposed heuristic edge pattern matching strategies adopted in M-HAMC provide an effective means to answer MC-GPM queries. In addition, with our proposed multithreading search strategies, M-HAMC can bidirectionally search the data graph in parallel, which greatly saves query processing time. Therefore M-HAMC significantly outperforms the previous algorithm HAMC in both effectiveness and efficiency. Therefore, M-HAMC is a very competitive algorithm for the new NP-Complete MC-GPM problem in social network based applications.

9 Conclusion

In this paper, we have proposed a new Multi-Constrained Simulation 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 multithreading heuristic algorithm, M-HAMC which employs our novel heuristic matching strategies for the NP-Complete MC-GPM problem. M-HAMC achieves $\mathcal{O}(E_D N_D log N_D + E_Q E_D)$ 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 and efficiency.

Acknowledgments

This work was partially supported by the Natural Science Foundation of China (Grant Nos. 61532018, 61402312, 61572336).

References

[1] J. Brynielsson, J. Hogberg, L. Kaati, C. Martenson, and P. Svenson, "Detecting social positions using simulation," in *Proc. Int. Conf. Adv. Soc. Netw. Anal. Mining*, 2010, pp. 48–55.
[2] W. Fan, J. Li, S. Ma, N. Tang, Y. Wu, and Y. Wu, "Graph pattern matching: From intractable to polynomial time," in *Proc. 30th Int. Conf. Very Large Data Bases*, 2010, pp. 264–275.
[3] W. Fan, X. Wang, and Y. Wu, "ExpFinder: Finding experts by graph pattern matching," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2013, pp. 1316–1319.
[4] M. R. Henzinger, T. Henzinger, and P. Kopke, "Computing simulations on finite and infinite graphs," in *Proc. 47th Annu. IEEE Symp. Found. Comput. Sci.*, 1995, pp. 453–462.
[5] R. Jin, Y. Xiang, N. Ruan, and D. Fuhry, "3-hop: A high compression indexing scheme for reachability query," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2009, pp. 813–826.
[6] L. Zou, L. Chen, and M. T. Ozsu, "Distance-join: Pattern match query in a large graph database," in *Proc. 30th Int. Conf. Very Large Data Bases*, 2009, pp. 886–897.
[7] G. Liu, Y. Wang, and M. A. Orgun, "Optimal social trust path selection in complex social networks," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2010, pp. 1391–1398.
[8] R. Milano, R. Baggio, and R. Piattelli, "The effects of online social media on tourism websites," in *Proc. Inf. Commun. Technol. Tourism*, 2011, pp. 471–483.
[9] G. Liu, Y. Wang, and M. A. Orgun, "Social context-aware trust network discovery in complex contextual social networks," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, pp. 101–107.
[10] G. Liu, et al., "Multi-constrained graph pattern matching in large-scale contextual social graphs," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2015, pp. 351–362.
[11] J. M. Jaffe, "Algorithms for finding paths with multiple constraints," *Netw.*, vol. 14, pp. 91–116, 1984.
[12] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Math. Operations Res.*, vol. 17, no. 1, pp. 36–42, 1992.
[13] L. Libkin and D. Vrgoc, "Regular path queries on graphs with data," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2011, pp. 74–85.
[14] J. Reutter, M. Romero, and M. Y. Vardi, "Regular queries on graph databases," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2015, pp. 1–18.
[15] W. Fan, X. Wang, and Y. Wu, "Diversified top-k graph pattern matching," in *Proc. 30th Int. Conf. Very Large Data Bases*, 2014, pp. 1510–1521.
[16] J. Cheng, X. Zeng, and J. X. Yu, "Top-k graph pattern matching over large graphs," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2013, pp. 1033–1044.
[17] P. Berger and T. Luckmann, *The Social Construction of Reality: A Treatise in the Sociology of Knowledge*. New York, NY, USA: Anchor Books, 1966.
[18] J. Cheng, J. X. Yu, B. Ding, P. S. Yu, and H. Wang, "Fast graph pattern matching," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2008, pp. 913–922.
[19] L. Zou, L. Chen, and M. T. Ozsu, "Distance-join: Pattern match query in a large graph database," in *Proc. 30th Int. Conf. Very Large Data Bases*, 2009, pp. 886–897.
[20] Z. Sun, H. Wang, H. Wang, B. Shao, and J. Li, "Efficient subgraph matching on billion node graphs," in *Proc. 30th Int. Conf. Very Large Data Bases*, 2012, pp. 788–799.
[21] J. Cheng, X. Zeng, and J. X. Yu, "Top-k graph pattern matching over large graphs," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2013, pp. 1033–1044.
[22] X. Yan, P. S. Yu, and J. Han, "Substructure similarity search in graph databases," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2005, pp. 766–777.
[23] H. Shang, X. Lin, Y. Zhang, J. X. Yu, and W. Wang, "Connected substructure similarity search," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2010, pp. 903–914.

[24] Y. Zhu, L. Qin, J. X. Xu, and H. Cheng, "Finding top-k similar graphs in graph databases," in *Proc. 15th Int. Conf. Extending Database Technol.*, 2012, pp. 456–467.
[25] F. N. Afrati, D. Fotakis, and J. D. Ullman, "Enumerating subgraph instances using map-reduce," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2013, pp. 62–73.
[26] J. Gao, C. Zhou, J. Zhou, and J. X. Yu, "Continuous pattern detection over billion-edge graph using distributed framework," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2014, pp. 556–567.
[27] Y. Shao, B. Cui, L. Chen, L. Ma, J. Yao, and N. Xu, "Parallel subgraph listing in a large-scale graph," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2014, pp. 625–636.
[28] J. Huang, K. Venkatraman, and D. J. Abadi, "Query optimization of distributed pattern matching," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2014, pp. 64–75.
[29] M. F. Demirci, "Graph-based shape indexing," *Mach. Vis. Appl.*, vol. 23, no. 3, pp. 541–555, 2012.
[30] Y. Tian and J. Patel, "TALE: A tool for approximate large graph matching," in *Proc. IEEE 24th Int. Conf. Data Eng.*, 2008, pp. 963–972.
[31] R. Milner, *Communication and Concurrency*. Upper Saddle River, NJ, USA: Prentice Hall, 1989.
[32] S. Ma, Y. Cao, W. Fan, J. Huai, and T. Wo, "Capturing topology in graph pattern matching," in *Proc. 30th Int. Conf. Very Large Data Bases*, 2011, pp. 310–321.
[33] W. Fan, J. Li, S. Ma, N. Tang, and Y. Wu, "Adding regular expressions to graph reachability and pattern queries," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2011, pp. 39–50.
[34] W. Fan, X. Wang, and Y. Wu, "Answering graph pattern queries using views," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2014, pp. 184–195.
[35] W. Fan, X. Wang, and Y. Wu, "Querying big graphs within bounded resources," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2014, pp. 301–312.
[36] Y. Fang, R. Cheng, S. Luo, and J. Hu, "Effective community search for large attributed graphs," *Proc. VLDB Endowment*, vol. 9, no. 12, pp. 1233–1244, 2016.
[37] Z. Yang, A. W.-C. Fu, and R. Liu, "Diversified top-k subgraph querying in a large graph," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2016, pp. 1167–1182.
[38] W. Fan, C. Hu, and C. Tian, "Incremental graph computations: Doable and undoable," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2017, pp. 155–169.
[39] F. L. S. Yoo, Y. Yang and I. Moon, "Mining social networks for personalized email prioritization," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 967–976.
[40] J. Tang, J. Zhang, L. Yan, J. Li, L. Zhang, and Z. Su, "ArnetMiner: Extraction and mining of academic social networks," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 990–998.
[41] G. Liu, Y. Wang, and M. A. Orgun, "Trust transitivity in complex social networks," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2011, pp. 1222–1229.
[42] N. Biggs, E. Lloyd, and R. Wilson, *Graph Theory*. Oxford, U.K.: Oxford University Press, 1986.
[43] W. Fan, J. Li, X. Wang, and Y. Wu, "Query preserving graph compression," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2012, pp. 157–168.
[44] H. Yildirim, V. Chaoji, and M. J. Zaki, "GRAIL: Scalable reachability index for large graphs," in *Proc. 30th Int. Conf. Very Large Data Bases*, 2010, pp. 276–284.
[45] S. Milgram, "The small world problem," *Psychology Today*, vol. 2, no. 60, pp. 60–67, 1967.
[46] J. M. Jaffe, "Algorithms for finding paths with multiple constraints," *Netw.*, vol. 14, pp. 95–116, 1984.
[47] V. Subramaniam, *Programming Concurrency on the JVM: Mastering Synchronization, STM, and Actors*. Raleigh, NC, USA: The Pragmatic Programmers, 2011.

Guanfeng Liu received the PhD degree in computer science from Macquarie University, Australia, in 2013. He is an associate professor in the School of Computer Science and Technology, Soochow University, China. His research interests include graph mining and social networks. He has published more than 40 papers in the most prestigious journals and conferences such as the AAAI, ICDE, CIKM, the *IEEE Transactions on Knowledge and Data Engineering*, the *IEEE Transactions on Services Computing*, and ICWS.

Yi Liu is working toward the master's degree in the Research Center on Advanced Data Analytics, Soochow University, China. His research interests include graph mining and social networks. He has published several papers in the *World Wide Web Journal*.

Kai Zheng received the PhD degree in computer science from The University of Queensland in 2012. He is a full professor with the University of Electronic Science and Technology of China. He has published over 50 papers in the most prestigious journals and conferences such as SIGMOD, ICDE, the *VLDB Journal*, ACM Transactions, and IEEE Transactions. He was the program committee co-chair of the 18th APWeb Conference and general co-chair of the 22nd DASFAA Conference.

An Liu received the PhD degree from the University of Science and Technology of China, Hefei, and the City University of Hong Kong, Hong Kong. He is an associate professor in the School of Computer Science and Technology, Soochow University, China. His research interests include the service-oriented computing and social networks. He served as the workshop co-chair of WISE 2017 and DASFAA 2015.

Zhixu Li received the BS and MS degrees from the Renmin University of China, and the PhD degree from the University of Queensland, in 2006, 2009, and 2013, respectively. He is an associate professor in the School of Computer Science and Technology, Soochow University, China. His research interests include data cleaning, big data applications, information extraction, and retrieval.

Yan Wang received the BEng, MEng, and the DEng degrees in computer science and technology from the Harbin Institute of Technology (HIT), P. R. China, in 1988, 1991, and 1996, respectively. He is currently an associate professor in the Department of Computing, Macquarie University, Sydney, Australia. His research interests include trust computing and social networks. He is a senior member of the IEEE.

Xiaofang Zhou received the BSc and MSc degrees in computer science from Nanjing University, China, and the PhD degree in computer science from The University of Queensland, Australia, in 1984, 1987, and 1994, respectively. He is a professor of computer science with The University of Queensland and adjunct professor in the School of Computer Science and Technology, Soochow University, China. His research interests include spatial and multimedia databases, high performance query processing, web information systems, data mining, bioinformatics, and e-research. He is a fellow of the IEEE.