Multi-Fuzzy-Constrained Graph Pattern Matching with Big Graph Data

Guliu Liu[a,b], Lei Li[a,b,*], and Xindong Wu[c,a,b]
[a]Key Laboratory of Knowledge Engineering with Big Data (Hefei University of Technology), Ministry of Education, Hefei, 230009, China
[b]School of Computer Science and Information Engineering, Hefei University of Technology, 230009, Anhui, China
[c]Mininglamp Academy of Sciences, Mininglamp Technologies, 100084, Beijing, China
adjworld@outlook.com, lilei@hfut.edu.cn, xwu@hfut.edu.cn

*Correspondence information:
Lei Li
School of Computer Science and Information Engineering,
Hefei University of Technology, 230009, Anhui, China
+8618356055575, lilei@hfut.edu.cn

Abstract

Graph pattern matching has been widespread used for protein structure analysis, expert finding and social group selection, ect. Recently, the study of graph pattern matching using the abundant attribute information of vertices and edges as constraint conditions has attracted the attention of scholars, and multi-constrained simulation has been proposed to address the problem in contextual social networks. Actually, multi-constrained graph pattern matching is an NP-complete problem and the fuzziness of constraint variables may exist in many applications. In this paper, we introduce a multi-fuzzy-constrained graph pattern matching problem in big graph data, and propose an efficient first-k algorithm Fuzzy-ETOF-K for solving it. Specifically, exploration-based method based on edge topology is adopted to improve the efficiency of edge connection, and breadth-first bounded search is used for edge matching instead of shortest path query between two nodes to improve the efficiency of edge matching. The results of our experiments conducted on three datasets of real social networks illustrate that our proposed algorithm Fuzzy-ETOF-K significantly outperforms existing approaches in efficiency and the introduction of fuzzy constraints makes our proposed algorithm more efficient and effective.

Keywords:graph pattern matching,big graph data, multi-fuzzy-constrained

1 Introduction

In recent years, Big data become increasing important in acadamic and industry, It has been widely studied by scholars. Big data are generated from multi-source, heterogeneous and autonomous data [1], and usually have the characteristics of distributed and decentralized control. Furthermore, extensive application scenario makes it more changeable and huge in scale, which brings great challenges to its processing and analysis [2]. Therefore, how to represent and mine useful information become a major issue in big data research. As an important data structure, graphs are used to represent big data with complex relationships. We will call it big graph later, such as biological networks, social networks and knowledge graphs. In order to conduct effective analysis over those graphs, various queries have been investigated, such as reachability [3], shortest path query [4], frequent subgraph mining [5, 6, 7, 8], subgraph matching [9, 10, 13, 14, 33], keyword search [11] and Graph Pattern Matching (GPM) [12, 16, 23, 35, 41], etc.

Early, for the search of graphs, researchers studied the reachability and the shortest path query in graphs. Subgraph queries for small-scale static graphs have also been extensively studied [13, 14]. However, with the continuous increase of data size and the increasing complexity of data relationships, the query and analysis of graph data are facing new challenges. Subgraph matching , which refers to finding subgraphs that is isomorphic to a given pattern graph G_P from the data graph G_D, has attracted the attention of researchers due to its wide applications. For example, there is a protein of unknown properties. How do we know his nature? The protein interaction network of known properties can be used as the pattern graph, and the protein interaction network of the unknown property protein as the data graph. Then the same interactive network can be matched through the pattern graph, and used to guess the properties of the unknown protein [15].

However, this isomorphic subgraph matching does not adapt well to some applications in social networks, such as expert finding [16], social group discovery [17], personalized recommendation [18, 19, 20, 21], ect. In order to extend the application of graph pattern matching in social networks, Fan et al. [22] proposed bounded simulation, which was based on binary correspondence between nodes and relaxes strict edge-to-edge mathcing to length bounded path matching. But they still didn't take the rich information on the nodes and edges in the big graph data to get better query results. Therefore, Liu et al. [23] proposed Multi-Constrained Graph Pattern Matching (MC-GPM) problem based on Contextual Social Graph (CSG), which required matching more effective results by multiple constraints on vertices and edges. In addition, there are often fuzzy attribute constraints that are not considered in existing MC-GPM methods, such as the social influence of participants, the degree of trust between social participants, ect.

In this paper, we present the problem of Multi-Fuzzy-Constrained Graph Pattern Matching (MFC-GPM), which makes MC-GPM more suitable for applications in general big graph data. In addition, based on the Baseline algorithm proposed by Liu et al. [23], we propose a more efficient Edge Topologically Ordered First-K(ETOF-K) algorithm, and expand it to Fuzzy-ETOF-K alogrithm for solving MFC-GPM problem. Finally, we experiment on three real-world social networks of different sizes to compare the efficiency of our algorithm with

two existing algorithms, which validates that our proposed algorithm significantly outperforms existing approaches and introducing fuzzy to MC-GPM is necessary.

The rest of this paper is organized as follows. We first review the related work on GPM in Section 2. Then, the introduction of necessary concepts and the definition of MFC-GPM are presented in Section 3. Section 4 proposes our Edge Topologically Ordered First-K algorithm. Section 5 presents our experimental sets and results, and section 6 concludes this paper.

2 Related Work

Graph pattern matching has developed for many years. At first, It is often used for isomorphic matching tasks, such as protein property detection [15]. It requires matching subgraphs to have the same topology structure as the pattern graph. However, the continuous increase of data scales and the increasingly complex relationship between data, the traditional graph pattern matching method based on isomorphism cannot meet the applications of emerging fields due to its high algorithm complexity. Simulation based graph pattern matching [22, 23, 24, 35, 36] has attracted the attention of researchers in recent years because of its ability to return matching results in cubic time.

Isomorphism-based GPM Isomorphism-based graph pattern matching is to find subgraphs that are isomorphic to the pattern graph, it is also called subgraph matching. The earliest Ullmann [13] proposed a depth-first-based matching algorithm, which is a brute-force enumeration search method. In order to improve the efficiency of the algorithm, Cordella et al. [14] proposed the VF2 algorithm by improving the pruning strategy of the Ullmann algorithm. More pruning strategies based on pattern graph semantics and structure in [25, 26] were studied. But isomorphic graph pattern matching is an NP-complete problem, in order to further improve the efficiency of matching, Yan et al. [27] proposed a GIndex algorithm based on frequent subgraph mining and indexing. This algorithm significantly improves the performance of graph pattern matching. Path-based indexing algorithm was also proposed by Shasha et al. [28]. More approaches about index-based matching can be found in the literature [29, 30, 31].

In recent years, with the rapid growth of data scale and the increasing complexity of data relationships, the above mentioned methods can not adapt to the exact matching of pattern graph on very large scale graph data. Afrati et al. [32] proposed a distributed parallel computing method based on map-reduce. They decomposed the pattern graph into serveral subgraphs, and then obtained the final matching subgraphs through distributed computing and join operations. However, the connection operation is very computationally intensive, which seriously affects the efficiency of the matching. Shao et al. [33] proposed a parallel computing framework PSgL which obtained matching results by iterative computation of intermediate matching results. In addition, incremental-based matching [18, 26, 30] and top-k algorithms [34] have also been studied in response to the dynamic growth of graphs and the need for real-time applications, respectively.

Simulation-based GPM Although the graph pattern matching based on isomorphism can be used to detect isomorphic structures of chemical substances and predict the interaction between proteins [15], and it can also be applied to expert discovery [16], social group query [17], personalized recommendation [18, 19], etc. However, isomorphic-based matching is too strict for the applications of social networks. Henzinger et al. [24] proposed graph simulation, which obtained a set of simulated nodes by calculating the similarity of nodes, and the computation can be completed in polynomial time. But this is still based on edge-to-edge matching, and its flexibility does not meet the needs of new applications such as in social networks. Fan et al. [22] proposed bounded simulation, which requires that the edges satisfy the bounded length path matching and can be completed in cubic time. Then, in order to adapt to specific applications, some improved methods based on bounded simulation have been proposed, such as graph pattern view [35], resource-bounded query [36] ect. Moverover, a strong simulation [37] is proposed for matching the pattern graph topological structure that was not well retained by bounded simulation. Exactly, strong simulation uses the duality to ensure the topological relationship between node and it's adjacent points, and then further secure it with the locality of the pattern graph. However, all the existing work has failed to use the abundant vertices and edges information contained in the big graph data. Shemshadi et al. [38] considered multi-label information on vertices but did not consider the constraints on the edges. Liu et al. [23] proposed the MC-GPM problem, put constraints on nodes and edges to match more accurate results, and proposed a Baseline algorithm based on exploration-based method and a heuristic algorithm(HAMC) based on compressed index of data graph. MC-GPM is useful for contextual social networking applications, such as crowdsourcing travel [21, 39, 39], social network based e-commerce [40], ect. Liu et al. [41, 42] also studied the top-k algorithm and the parallel algorithm. However, these tasks have not substantially improved the performance of the Baseline algorithm and the HAMC algorithm, but only use more computing resources in exchange for shortening the matching time. In this paper, we will propose an ETOF-K algorithm to improve the performance of the algorithm from the aspects of edge matching and edge joining, respectively. And the general mode of multi-fuzzy-constrained graph pattern matching in big graph data is proposed.

3 Multi-Fuzzy-Constrained Graph Pattern Matching (MFC-GPM)

In this section, first, we will give the definition of the data graph and the pattern graph in the graph pattern matching. Then the definition of the graph simulation is given. Finally, we introduce our proposed MFC-GPM based on graph simulation.

3.1 Data Graph and Pattern Graph

Data Graph A data graph is a labeled directed graph and can be represented by $G_D = (V, E, f_v^D, f_e^D)$, where

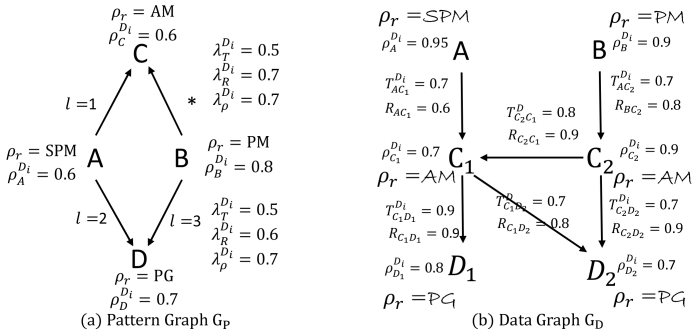


Fig. 1: Data Graph and Pattern Graph in a CSG

- 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;
- f_v^D is a function defined on V such that for each vertex $v \in V$, $f_v^D(v)$ is a set of attributes of v. As in a CSG, for a vertex v, $f_v^D(v)$ may represent the social role ρ_r of domain D_i to which v belongs and the social influence $\rho_v^{D_i}$ of v in domain D_i;
- f_e^D is a function defined on E such that for each directed edge $(v_i, v_j) \in E$, $f_e^D(v_i, v_j)$ is a set of attribute of (v_i, v_j). As in a CSG, $f_e^D(v_i, v_j)$ may denote social relationship $R_{v_iv_j}$ and social trust $T_{v_iv_j}$ between v_i and v_j.

Pattern Graph A pattern graph is defined as a labeled directed graph and is denoted as $G_P = (V_P, E_P, f_v^P, f_e^P, f_l^P, f_m^P)$, where

- V_P and E_P are the set of vertices and the set of directed edges, respectively;
- f_v^P is a function defined on V_P such that for each vertex $u \in V_P$, $f_v^P(u)$ is a set of attribute constraints of u. As in a CSG in Fig.1, each vertex has a social role constraint ρ_r and a social influence constraint $\rho_v^{D_i}$;
- f_e^P is a function defined on E_P such that for each directed edge $(u_i, u_j) \in E_P$, $f_e^P(u_i, u_j)$ is a set of attribute constraints of (u_i, u_j);
- f_l^P is a function defined on E_P such that for each edge (u_i, u_j), $f_l^P(u_i, u_j)$ is the bounded length of (u_i, u_j) which is either a positive integer k or a symbol $*$, indicates that the interval length between u_i and u_j cannot exceed k or there is no requirement for it, respectively;
- f_m^P is a set of membership functions defined on each attribute constraints.

Example 1. As shown in Fig.1, in the data graph G_D, each vertex represents a social participant and has a label ρ_r that represents its social role and an aggregated attribute $\rho_v^{D_i}$ that represents its influence in a particular domain D_i. Each edge has two aggregated attributes $R_{v_iv_j}$ and $T_{v_iv_j}$, which represent the

social relationship and social trust between the two vertices v_i and v_j ,respectively. In the pattern graph G_P, each vertex has constraints on its social role and influence in a particular domain. Each edge has multiple aggregated social constraints λ_ρ, λ_T and λ_R, which respectively represent the social influence, social trust and social relationship, and constraint l that limit the length of the matching path. It should be noted that all the vaule of aggregated attributes can be mined from the existing social networks. Moveover, The membership functions of all aggregated attributes can be defined as uniform, such as equation 1.

$$f_m = \begin{cases} \frac{f_D}{f_P} & 0 < f_D < f_P \\ 1 & f_P \leq f_D \end{cases} \tag{1}$$

where f_m represents the uniform membership function, f_P and f_D represent the constraints in the pattern graph and the corresponding aggregated matching values obtained in the data graph, respectively. In addition, the set of the membership constraint values of all membership functions can be set to 0.9.

3.2 Graph Simulation

Graph Simulation proposed by Henzinger et al. [24]. They used it to calculate the similarity of the nodes for verification and refinement of the reaction system. Fan et al. [22]improved the graph simulation, proposed bounded simulation, and applied it to graph pattern matching. The definition of the graph simulation is as follows:

Consider a data graph $G_D = (V, E, f_v^D)$ and a pattern graph $G_P = (V_P, E_P, f_v^P, f_l^P)$. A subgraph of data graph G_D matches pattern graph G_P via graph simulation denoted by $G_P \trianglelefteq^B G_D$. If there exists a binary relation $S_B \subseteq V_P \times V$ such that

- for all $u \in V_P$, there exists $v \in V$ such that $(u, v) \in S_B$, and the attributes $f_v^D(v)$ of v satisfies the constraint function $f_u^P(v)$ of u;
- for each pair $(u, v) \in S$,
  - $u \sim v$, and
  - for each edge (u, u') in E_P, there exists a nonempty path p from v to v' in G_D such that $(u', v') \in S_B$, and $length(p) \leq k$, if $f_l^P(u, u') = k$;

Then S_B is a match in G_D for G_P via graph simulation.

It should be noted that this type of graph simulation matching results not only include nodes that match the vertices in the pattern graph, but also include nodes on the edge matching path. In addition, $u \sim v$ means that u is similar to v. That is, u and v satisfy the same node labels or the v-labels contain all the labels of u, and within the path length constraint, all subsequent nodes of u are included in the successor node of v, and these corresponding successor nodes are also similar. This is the weakest constraint graph simulation definition, proposed by Fan et al. [22]. In the original simulation proposed by Henzinger et al. [24], the matching path length is defined as 1, which is an edge-to-edge matching. Later, Ma et al. [37] further requested on the basis of the bounded simulation. If $u \sim v$, then the parent nodes of v contain all the father nodes of u. That is, $\forall u', (u', u) \in E_P$ there exist a vertex $v', (v', v) \in E$ and $(u', v') \in S_B$, if $(u, v) \in S_B$. Liu et al. [23] proposed Multi-Constrained Simulation (MCS) based on bounded simulation to adapt to multi-constrained matching.

3.3 MFC-GPM

The simulation-based graph pattern matching is more flexible than isomorphism-based and can complete matching queries in polynomial time, so it is more suitable for social group discovery [22], personalized recommendation [18,19,20,21], etc. in social networks with higher real-time requirements. However, big graph data usually contains a large amount of vertex and edge information, which can be used to match more accurate results. Liu et al. [23] proposed MC-GPM problem and proposed two matching method based on MCS. MC-GPM can be used for crowd-sourcing travel [39], social network-based e-commerce [40], etc. However, in the practical application of big graph data, there are usually constraints that are inherently fuzzy, such as the social influence on the vertex and the social trust on the edge in a CSG. Obviously, MC-GPM does not take this fuzzy property of constraints into account. As a result, some actual useful match results be lost. For example, the results that only one attribute is slightly below the constraint and other constraints are well satisfied are lost. Therefore, in this paper, the MFC-GPM problem is proposed, in which all constraints are set to a membership function. For constraints without fuzzy property, set its membership value to be equal to 1. Moreover, we can set different membership functions for different constraints. But for the sake of illustration, in this paper, all constraints are set to the same membership function.

When a vertex $v \in V$ satisfy the node label constraint of vertex $u \in V_P$, we use the membership function defined on vertex constraints to calculate the membership value of other constraints defined on u, and if the membership constraint is satisfied, the matching vertex v is saved. Moreover, when a path p satisfies the length constraint defined on its corresponding pattern edge, we use the membership function defined on it to calculate the membership value of other constraints defined on the pattern edge. The edge matching result is saved when all constraints satisfy the membership constraints. When all the vertices and edges are matched, a subgraph matching result will be returned. The specific definition of MFC-GPM is as follows:

Consider a data graph $G_D = (V, E, f_v^D, f_e^D)$ and a pattern graph $G_P = (V_P, E_P, f_v^P, f_e^P, f_l^P, f_m^P)$, where $f_m^P = \{f_v^m, f_T^m, f_R^m, f_\rho^m\}$, indicate that the membership functions defined on corresponding attribute constaints. G_D matches G_P via MFC-GPM, denoted by $G_P \trianglelefteq^{MFC} G_D$, if there exists a binary relation $S \subseteq V_P \times V$ such that

- for all $u \in V_P$, there exists $v \in V$ such that $(u, v) \in S$, where $(u, v) \in S$ means that there is a vertex v that matches u, that is, v satisfies $f_v^P(u)$ or $\rho_v^{D_i}$ satisfies f_v^m, if $f_v^P(u)$ contains attribute constraint $\rho_v^{D_i}$ and f_v^m represent membership function defined on $\rho_v^{D_i}$;
- for each pair $(u, v) \in S$,
  - $u \sim v$, and
  - for each edge (u, u') in E_P, there exists a nonempty path p from v to v' in G_D such that $(u', v') \in S$, and $length(p) \leq k$, if $f_l^P(u, u') = k$;
  - $AT^{D_i}(v, v') \geq f_T^m$, $AR(v, v') \geq f_R^m$ and $A\rho^{D_i}(v, v') \geq f_\rho^m$, where f_T^m, f_R^m, f_ρ^m are the membership functions of $\lambda_T, \lambda_R, \lambda_\rho$, respectively. If $f_e^P(u, u') = \{\lambda_T, \lambda_R, \lambda_\rho\}$ and $AT^{D_i}(v, v'), AR(v, v'), A\rho^{D_i}(v, v')$ represent aggregated attributes of path (v, v');

then S is a match in G_D for G_P via MFC-GPM.

Example 2. See pattern graph in Fig.1, we can easily get matching preselected set for each vertex, where vertex SPM (i.e., A, a Senior Project Manager), vertex PM (i.e., B, a Project Manager) in G_P can be mapped to the same vertices SPM and PM in G_D, respectively; vertex AM (i.e., C, an Assistant Manager) in G_P corresponds to multiple AMs (i.e., C_1 and C_2) in G_D and vertex PG (i.e., D, a Programmer) in G_P corresponds to multiple PGs (i.e., D_1 and D_2) in G_D. For edge matching we can easily get (A, C) in G_P to match (A, C_1) in G_D (denoted as $(A, C_1, G_D) \simeq (A, C, G_P)$), and others edge matching $(A, C_1, D_1, G_D) \simeq (A, D, G_P)$, $(A, C_1, D_2, G_D) \simeq (A, D,\ G_P)$, $(B, C_2, C_1, G_D) \simeq (B, C, G_P)$ and $(B, C_2, G_D) \simeq (B, C, G_P)$. For edge (B, D) in G_P, if the matching is performed by the conventional MC-GPM, the path (B, D_1) can be obtained. However, since $AT^{D_i}(B, D_2) = 0.49$, which is less than the constraint value 0.5 of the corresponding pattern edge (B, D) to λ_T, path (B, D_2) is discarded. In our proposed MFC-GPM, since the path (B, D_2) satisfies the fuzzy constraint, it is saved.

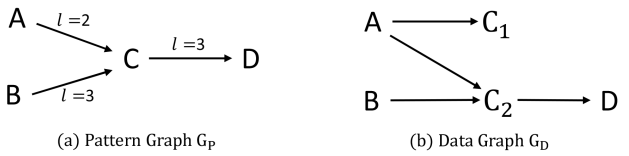
4 Edge Topologically Ordered First-K (ETOF-K) Algorithm

In this section, the ETOF-K algorithm will be introduced first. Then, the multi-constrained edge matching and the topologically ordered edge connection are explained in detail.

4.1 Methodology

MC-GPM is an NP-complete problem. In order to solve this problem effectively, Liu et al. [23], proposed two First-K algorithms based on MCS. First, they proposed a Baseline algorithm based on bounded simulation, which performs the edge connection based on the depth-first traversal of the graph, and the edge matching is performed by the shortest path query between nodes. Then they proposed a heuristic algorithm (HAMC) based on the compression and indexing of strongly linked subgraphs (they called Strong Social Component, SSC) in data graph, which also uses the depth-first traversal method of the graph for edge connection. But for the edge matching, the query is first performed in the SSC index. If it exists, the edge matching result is returned and if it does not exist, the Dijkstra algorithm is used to search in the data graph.

In this paper, An Edge Topologically Ordered First-K algorithm has been proposed based on Baseline algorithm. First, ETOF-K algorithm performs edge matching by breadth-first bounded search based on the current matching vertex, instead of matching the shortest path between two vertices. This will greatly reduce the query of the shortest path between unnecessary vertices. Second, the edge topologically ordered sequence is used to edge connection processing instead of the graph depth-first search. Because when we match an edge, if there is still an edge pointing to its starting vertex that has not yet matched, then the current matching may be invalid. Therefore, the topologically ordered edge join method can achieve pruning by filtering the vertices before matching the edges starting from it.



(a) Pattern Graph G_P (b) Data Graph G_D

Fig. 2: Pattern and Data Graph for Illustrating Edge Connection (Note that we omit the attributes and attribute constraints on the nodes and edges in the graph, which are the same as in Fig.1.)

Example 3. Consider a pattern graph in Fig.2, for edge join operations, if we already get $(A, C_1, G_D) \simeq (A, C, G_P)$ and $(A, C_2, G_D) \simeq (A, C, G_P)$. Then, if using the depth-first method of Baseline algorithm, the match edge (C, D) will be started from vertex C_1. But obviously, there is no path from C_1 to vertex D, so this step should be reduced. By using topologically ordered edge join method, when the matching result of edge (A, C) is obtained, the edge (B, C) should be matched. It then connects the edges (A, C) and (B, C) through the vertex C, and filters the matching results of the two edges. Therefore, when matching edge (C, D), only the matching path with C_2 as the starting vertex needs to be considered. For edge matching, our proposed ETOF-K algorithm will not need to consider whether there is a shortest path between the vertex B and C_1 that satisfies the bounded length. Instead, we start from B to conduct a breadth-first bounded search for matching edges that satisfy the constraints.

For fast return matching results, the ETOF-K algorithm proved to be very effective by our experiments. It is a process of edge-joining while edge matching. First, we need to get the indegree of all the vertices in the pattern graph, and then use the topological sorting algorithm to get the order of accessing the edges. We call it the topologically ordered sequence of edges. Then we perform edge matching and join operations based on this sequence.

4.2 Multi-Constrained Edge Pattern Matching

Multi-constrained edge pattern matching is the main part of our algorithm. It is usually a query in the data graph to match the edges or paths that satisfy all the constraints defined on the edges in the pattern graph. In practical applications, it usually means to query whether a set of specified conditions can be satisfied between two participants. In this paper, A pattern edge matching algorithm based on candidate node breadth-first depth bounded search is proposed. First, we conduct a breadth first depth bounded search from a candidate vertex $v \in V$ of u, and get all matching paths $(v, v') \in M_{path}$ that satisfy a bounded length of (u, u'), where $u' \sim v'$. Then, whether these matching paths in M_{path} satisfy the multiple constraints defined on the corresponding pattern edge is checked. If there are matching paths in M_{path} that satisfy all constraints, add them to the list $M^i_{e_p}$ and return the edge matching results list $M^i_{e_p}$. Otherwise, it returns an empty set $\emptyset$.

The detailed steps of the algorithm are shown in Algorithm 1, where path$w$ represents an empty path with a starting point of v, an end point of v, and a length of zero. e_p represents the pattern edge currently to be matched, and v represents the candidate node of the starting node of the edge e_p. We use the

Algorithm 1 Muti-Constrained Edge Pattern Matching, MC-EPM

Input: $e_p \in E_P, v \in V and\, G_D$
Output: path set $M^i_{e_p}$

```
1: Push(Q, pathw)
2: while !QueueEmpty(Q) do
3:    pathj = Pop(Q)
4:    if pathj.end satisfy constraint f_v^P(e_p.end) then
5:       Push(M_path, pathj)
6:    end if
7:    if pathj.length < e_p.boundedlength then
8:       p = G.vertices[pathj.end].firstarc
9:       while p != NULL do
10:         pathi = pathj.push(p)
11:         Push(Q, pathi)
12:         p = p -> nextarc
13:      end while
14:   end if
15: end while
16: while l ++ < M_path.length do
17:   if M_path[l] satisfy constraints f_e^P(e_p) then
18:      Push(M^i_{e_p}, M_path[l])
19:   end if
20: end while
21: return M^i_{e_p}
```

queue Q to help perform a breadth-first search on the data graph, as shown in line 2-15. When the path length is less than the bounded length, all edges starting from the end vertex of the current path are added to path$i$, and then added to the queue Q, as shown by line 7-14. If the end vertex of the current path is similar to the end vertex of the edge e_p, then the current path is added to the candidate path list M_{path}, as shown by line 4-6. Finally, it is looped to determine whether the path in the paths set M_{path} satisfies a plurality of constraints defined on the corresponding pattern edge e_p, as shown by line 16-20.

For different applications, the multiple constraints defined on the edge may be different. For example, in CSG, the multiple constraints defined on the edges are often social influence factors, social trust, and social relationships; in medical big data, it can be reliability determined by the doctor's experience, data quality determined by data sources and data imbalance, ect. In addition, for our proposed MFC-GPM, whether a path satisfies multiple constraints will be determined by its membership function and corresponding membership value defined on different constraints, such as example 2.

4.3 Exploration-Based Topologically Ordered Edge Connection

Our proposed ETOF-K algorithm is a exploration-based method for graph pattern matching, which aims to quickly answer a GPM query. The matching

Algorithm 2 ETOF-K Algorithm

Input: G_D, G_P
Output: results set M_{sub}

1: Get the indegree of all vertices in G_P, $indegree[V_P]$.
2: Get the topological ordered sequence of edges in G_P, $Edge\_T[E_P]$.
3: Matches candidate vertex sets V_i^m for all vertices with $indegree[v_i] = 0$.
4: **while** there exists $v \in V_0^m$, $v.visited$=False **do**
5: $M_{e_p}^0$ = MC-EPM $((u, u'), u \sim v, G_D)$
6: **if** $M_{e_p}^0 \neq \phi$ **then**
7: **while** $l{+}{+} < M_{e_p}^0.length$ **do**
8: $M_{temp}.add(M_{e_p}^i[l])$
9: $M'_{sub} = EC(G_D, Edge\_T[E_P], M_{e_p}, i+1)$
10: $M_{sub}.add(M'_{sub})$
11: **end while**
12: **else**
13: continue
14: **end if**
15: **end while**
16: **return** M_{sub}

process can be described in two parts. In the previous section, we have already introduced the edge matching process. Therefore, in this section, the edge connection method will be introduced. For our proposed MC-GPM algorithm ETOF-K, before performing edge matching and connection, it is necessary to obtain the indegree of all vertices $indegree[V_P]$, the topological ordered sequence of edges $Edge\_T[E_P]$ and the candidate sets V_i^m of vertices with zero indegrees in the pattern graph. Then, starting from the first edge in $Edge\_T[E_P]$, each edge is matched in turn, and after each edge matching set is obtained, the Edge Connection (EC) method is used for the edge connection. Note that we need to save a list of matching results for each matched edge. The pseudo code of the ETOF-K algorithm and its EC method are shown in Algorithm 2 and Algorithm 3, respectively.

In Algorithm 2, V_0^m represents the candidate matching node of the starting node of the first edge in $Edge\_T[E_P]$. The loop matches each candidate node in V_0^m as shown in line 4-15. First, the MC-EPM algorithm is used to get the matching results $M_{e_p}^0$ of the first pattern edge (u, u') in $Edge\_T[E_P]$. Then, Recursive matching is performed for each matching edge in $M_{e_p}^0$, as shown in line 7-11. The detailed process of edge recursive matching is shown in Algorithm 3.

In Algorithm 3, the matching results $M_{e_p}^i$ of the current pattern edge (u, u') should be obtained firstly. If $M_{e_p}^i$ is empty, return to the previous level; otherwise, judge the list of matched edges whether there is an edge (u'', u') with the same end node as the current pattern edge, as shown in line 3-4; if it exists, the matching result of the edge and the matching result of the current edge are filtered first, and then the next pattern edge is matched; If not, the next pattern edge is matched directly , as shown in line 10-13. If the Edge currently being matched is the last pattern Edge in $Edge\_T[E_P]$, the matching result is recorded in the list M_{sub}, and return to the previous matching edge in $Edge\_T[E_P]$ to

Algorithm 3 Edge Connection, EC

Input: G_D, M_{e_p}, $Edge\_T[E_P]$, $(u, u') \in E_P$
Output: result set M_{sub}

1: $M^i_{e_p}$ = MC-EPM($(u, u'), u \sim v, G_D$)
2: **if** $M^i_{e_p} \neq \emptyset$ **then**
3: **if** there is a matching result of (u'', u') **then**
4: Join edges (u, u') and (u'', u') to get candidate sets for vertex u', and update the candidate set $M^i_{e_p}$ of the current matching edge.
5: **end if**
6: **if** Edge (u, u') is the last edge in $Edge\_T[E_P]$ **then**
7: Record the current matching result in M_{sub}.
8: Return to the previous edge to match other results.
9: **end if**
10: **while** $l{+}{+} < M^i_{e_p}.length$ **do**
11: $M_{temp}.add(M^i_{e_p}[l])$
12: Match the next edge in $Edge\_T[E_P]$ by EC.
13: **end while**
14: **else**
15: **return** ϕ
16: **end if**
17: **return** M_{sub}

matching other results, as shown in line 6-9. You can find an example of a matching process in Example 4.

Example 4. Consider MC-GPM example in Fig.1. First, the indegree of all vertices $indegree[V_P]$ ($indegree[A] = 0$, $indegree[B] = 0$, $indegree[C] = 2$ and $indegree[D] = 2$) should be calculated, then the topological ordered sequence $Edge\_T[E_P]$ ((A, C), (A, D), (B, C), (B, D)) of the pattern edges should be obtained by topological sorting algorithm based on $indegree[V_P]$, and then, the set of candidate vertices $V^m_A = \{A\}$ and $V^m_B = \{B\}$ for vertices A and B can be getting. Then, the MC-EPM method is used to perform multiple constraints matching on the edges in $Edge\_T[E_P]$ in turn. When matching the third edge (B, C), the edge matching $(A, C_1, G_D) \simeq (A, C, G_P))$, $(A, C_1, D_1, G_D) \simeq (A, D, G_P)$ and $(A, C_1, D_2, G_D) \simeq (A, D, G_P)$ have to be completed, so when getting the edge matching result $(B, C_2, C_1, G_D) \simeq (B, C, G_P)$ and $(B, C_2, G_D) \simeq (B, C, G_P)$, the connection of the matching results of the edges (A, C) and (B, C) should be performed to get the candidate edge $(B, C_2, C_1, G_D) \simeq (B, C, G_P)$ of (B, C), and the candidate vertex C_1 of C. In a similar way, the edge connection operations on edges (A, D) and (B, D) are performed. Finally, we will get two matching results $M_{sub1} = (V_m, E_m, LV, LE)$, where $V_m = \{A, B, C_1, C_2, D_1\}$ and $E_m = \{(A, C_1), (B, C_2), (C_2, C_1), (C_1, D_1)\}$ by using MC-GPM and $M_{sub2} = (V_m, E_m, LV, LE)$, where $V_m = \{A, B, C_1, C_2, D_2\}$ and $E_m = \{(A, C_1), (B, C_2), (C_2, C_1), (C_1, D_2), (C_2, D_2)\}$ by using MFC-GPM.

5 Experiments

In this section, we conduct experiments on three real-world social graphs. The details of the three datasets are shown in Table 1. The Baseline and HAMC algorithm have been implemented based on source code of Liu et al. [23] and an algorithm, we call it the Baseline2 algorithm, has been implemented by using our edge matching method instead of the original one of the Baseline algorithm. Then, our proposed ETOF-K algorithm has been implemented by ourself. Finally, we compare the efficiency of the Baseline, HAMC, Baseline2 and ETOF-K algorithms. The effectiveness of introducing fuzzy constraints into MC-GPM is proved by comparing the ETOF-K algorithm and the Fuzzy-ETOF-K algorithm.

5.1 Experiment Settings and Implementation

The datasets used for our experiments contain only vertices and edges, which are available at snap.stanford.edu. The aggregated attributes of the vertices and edges mentioned earlier can be mined from the existing social networks, which is another very challenging problem. In our experiments, without loss of generality, we randomly generate them with the function *rand*() in SQL. In Exp-1, the constraint $\rho_v^{D_i}$ of the nodes is set to 0.01, and the multi-constraint attribute settings on the edges are as follows, $\lambda_T^{D_i} = 0.01$, $\lambda_r^{D_i} = 0.01$, $\lambda_\rho^{D_i} = 0.01$. The length constraint f_l^P of the edges is set to 3. The settings of those parameters will ensure that all algorithms have a result return, which is necessary for comparing the efficiency of the algorithms. In Exp-2, the constraint $\rho_v^{D_i}$ of the nodes is set to 0.1, and the multi-constraint attribute settings on the edges are as follows, $\lambda_T^{D_i} = 0.1$, $\lambda_r^{D_i} = 0.1$, $\lambda_\rho^{D_i} = 0.1$. The length constraint f_l^P of the edges is set to 3. These settings of those parameters are for us to easily observe the impact of fuzzy constraints on the matching, while ensuring the return of matching results. In addition, we use equation 1 as the membership function of all aggregated constraints in a pattern graph, and set the membership value to 0.9. For the generation of strong linked components (SSC) in the HAMC algorithm, we require that all attributes of the vertices and edges selected into the SSC must be greater than or equal to 0.2, and the number of SSC acquisitions is set to 100. For a more detailed understanding of SSC, you can refer to [23].

Since MC-GPM is an NP-complete problem, it is impossible to get all the matching results in G_D. We compare the efficiency of these algorithms by the number of results returned over a certain period of time. All the Baseline, HAMC, Baseline2, ETOF-K and Fuzzy-ETOF-K algorithms are implemented using Visual C++ and running on a PC with Intel(R) Core(TM) i7-8700K CPU @3.70GHz,48GB RAM, Windows 10 operating system and Mysql 5.6 database.

5.2 Experimental Results and Analysis

Exp-1: This experiment is to investigate the efficiency of our ETOF-K algorithm by comparing the number of matching results returned in the same time of four algorithms.

As shown in Fig.3-Fig.5. (1) The Baseline and HAMC algorithms return almost the same number of results at all statistical time for all three datasets. This is because the matching process between the HAMC algorithm and the

Table 1: The detail information of three real-world data graph

Dataset	Vertices	Edges	Description
Epinions	75879	508837	A trust-oriented social network
DBLP	317080	1049866	A co-author relationship network
Pokec	1632803	30622564	A general online social network

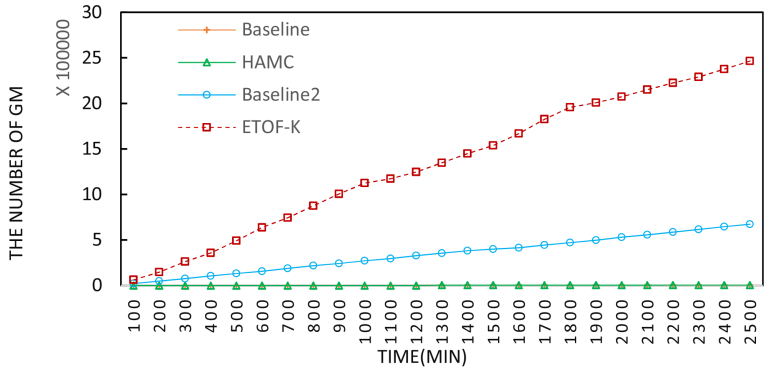


Fig. 3: The number of matching results G_M in different time on Epinions

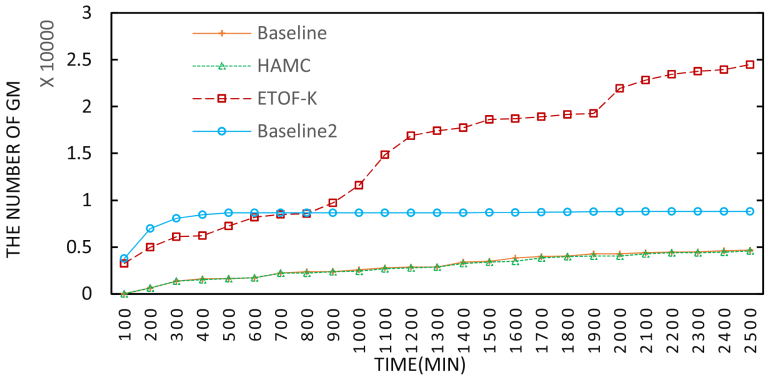


Fig. 4: The number of matching results G_M in different time on DBLP

Baseline algorithm is almost the same, except that the HAMC algorithm can speed up the matching process of the partial edges by using the compression and indexing of the SSC subgraph. However, as a whole, the HAMC algorithm first queries the index file when performing edge matching, so the overall efficiency is not as good as the Baseline algorithm. (2) Baseline2 algorithm that replaces the shortest path query's edge matching method with our edge matching method can get more matching results than Baseline and HAMC algorithms at same

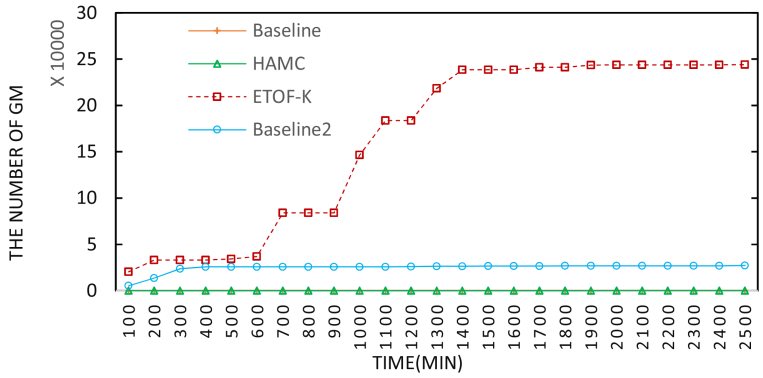


Fig. 5: The number of matching results G_M in different time on Pokec

statistical time. This proves that our vertex-based breadth-first bounded edge matching method is more efficient than the previous edge matching method. (3) And the number of matching results obtained by our ETOF-K algorithm is much higher than the previous three algorithms at three datasets. This proves that the edge matching process based on edge topologically ordered sequences can effectively prun the edge matching process, thus improving the matching efficiency.

In addition, the statistics of the specific number of matching subgraphs in Tables 2-4. T represents the algorithm execution time in minutes. From those tables we can see that the Baseline algorithm and the HAMC algorithm perform very poorly on the Epinions dataset and the Pokec dataset. On the Epinions dataset, the two algorithms do not return matching results after 20 hours of execution. On the Pokec dataset, the Baseline algorithm did not return a matching result for 5 hours, and the HAMC algorithm did not return a matching result after 30 hours of execution. Moreover, the number of matching results does not increase linearly with the execution time of the algorithm, but rising in a fold line. This is because the social complexity of different candidate nodes in the network is different. When the social relationship of a candidate node is complex, we need to judge the matching of all its complex social relationships, this results in a significant increase in the computational complexity of its matching. If it is finally determined that the candidate node does not meet the requirements, no matching result will be returned for a long time.

Exp-2: This experiment is to investigate the efficiency and effectiveness of introduce fuzzy constraints to our ETOF-K algorithm. Due to the fact that we cannot get all match subgraph, the efficiency of the two algorithms is compared by using the method in Exp-1 and the effectiveness of two algorithms is compared by the number of results that concerned to the same first edge in the topological ordered sequence of pattern edges after running 2500 minutes.

As shown in Fig.6 and Fig.8, on the Epinions and Pokec datasets, the Fuzzy-ETOF-K algorithm can return more matching results than the ETOF-K algorithm at all the same statistical times. On the DBLP dataset, the number of results returned by the Fuzzy-ETOF-K algorithm and the ETOF-K algorithm

Table 2: Details of the number of results returned at different time points on the Epinions dataset

	T=100	T=200	T=300	T=400	T=500	T=600	T=700	T=800	T=900	T=1000	T=1100	T=1200	T=1300
Baseline	0	0	0	0	0	0	0	0	0	0	0	0	3429
HAMC	0	0	0	0	0	0	0	0	0	0	0	0	3429
Baseline2	22334	48498	77008	105241	132615	156850	189364	218323	243366	271907	296954	329934	354038
ETOF-K	62543	147842	261741	359301	491191	637043	744322	876327	1004723	1124644	1171168	1247555	1347302

	T=1400	T=1500	T=1600	T=1700	T=1800	T=1900	T=2000	T=2100	T=2200	T=2300	T=2400	T=2500
Baseline	3429	3429	3429	3429	3429	3429	3429	3429	5322	5322	5322	5322
HAMC	3429	3429	3429	3429	3429	3429	3429	3429	5322	5322	5322	5322
Baseline2	382500	400533	416104	444975	470518	498989	529692	558707	586977	618085	646753	673075
ETOF-K	1449619	1537764	1668963	1827056	1955824	2007173	2073885	2150678	2224766	2290701	2377292	2464516

Table 3: Details of the number of results returned at different time points on the DBLP dataset

	T=100	T=200	T=300	T=400	T=500	T=600	T=700	T=800	T=900	T=1000	T=1100	T=1200	T=1300
Baseline	21	647	1358	1603	1642	1729	2222	2359	2359	2582	2793	2865	2865
HAMC	21	647	1358	1526	1642	1729	2222	2222	2359	2425	2681	2793	2865
Baseline2	3789	6992	8066	8470	8655	8655	8655	8655	8655	8655	8655	8655	8655
ETOF-K	3260	4992	6105	6216	7262	8201	8496	8572	9722	11605	14863	16890	17430

	T=1400	T=1500	T=1600	T=1700	T=1800	T=1900	T=2000	T=2100	T=2200	T=2300	T=2400	T=2500
Baseline	3412	3476	3853	3998	4046	4273	4285	4385	4460	4488	4593	4664
HAMC	3253	3412	3476	3853	3998	4046	4050	4273	4385	4385	4460	4593
Baseline2	8670	8689	8689	8732	8760	8776	8792	8805	8805	8805	8805	8805
ETOF-K	17753	18625	18722	18911	19163	19280	21964	22832	23450	23780	23944	24490

Table 4: Details of the number of results returned at different time points on the Pokec dataset

	T=100	T=200	T=300	T=400	T=500	T=600	T=700	T=800	T=900	T=1000	T=1100	T=1200	T=1300
Baseline	0	0	0	185	185	185	185	185	185	185	185	185	185
HAMC	0	0	0	0	0	0	0	0	0	0	0	0	0
Baseline2	5315	13557	23668	25614	25614	25614	25614	25614	25614	25614	25614	26107	26473
ETOF-K	20413	33073	33073	33073	34187	37008	84234	84234	84234	146686	183800	183800	218525

	T=1400	T=1500	T=1600	T=1700	T=1800	T=1900	T=2000	T=2100	T=2200	T=2300	T=2400	T=2500
Baseline	185	185	185	185	185	185	185	185	185	185	185	185
HAMC	0	0	0	0	0	0	185	185	185	185	185	185
Baseline2	26473	26522	26522	26522	27002	27002	27002	27002	27002	27019	27019	27203
ETOF-K	238354	238354	238354	241182	241283	243620	243683	243860	243860	243860	243910	244021

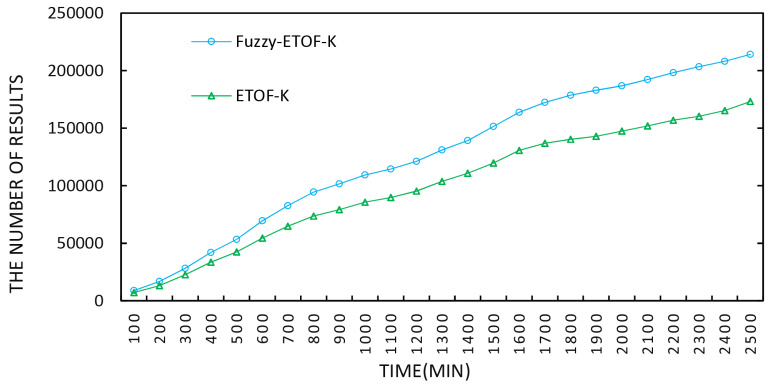


Fig. 6: The number of matching results G_M in different time on Epinions

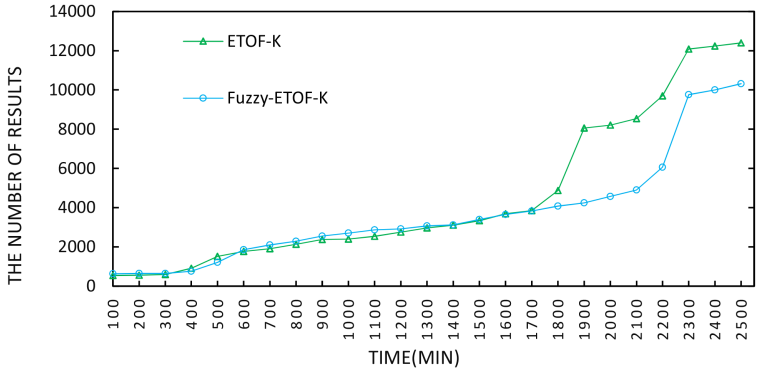


Fig. 7: The number of matching results G_M in different time on DBLP

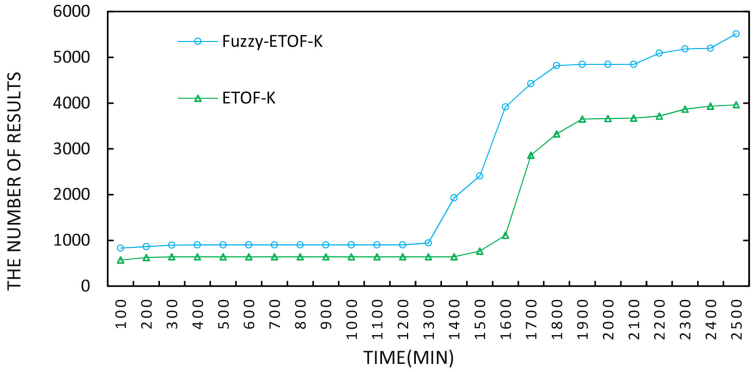


Fig. 8: The number of matching results G_M in different time on Pokec

is similar in the first 30 hours, and the ETOF-K algorithm is superior to the Fuzzy-ETOF-K algorithm, as shown in Fig.7. This is because fuzzy constraints

Table 5: The comparison of the number of results between ETOF-K and Fuzzy-ETOF-K on three datasets

Dataset	Matching edges	ETOF-K	Fuzzy-ETOF-K	Percentage
Epinions	2	136626	176116	28.90%
DBLP	140	11322	13680	20.83%
Pokec	11	4005	5253	31.16%

require membership calculation for each constraint condition, so the access of Fuzzy-ETOF-K algorithm to candidate nodes is slightly slower than that of ETOF-K algorithm as a whole. These experimental results prove that although Fuzzy-ETOF-K algorithm is not better than ETOF-K algorithm in all cases, but it is not much worse than ETOF-K algorithm even if it is poor. This also proves that the influence of the fuzzy calculation on the matching efficiency is slightly less than the effect of the effective matching result discard on the matching efficiency.

In addition, as shown in Table 5, after 2500 minutes of algorithm execution, on the Epinions dataset, we can get the two matching results of the first edge and all the 136626 matching subgraphs associated with the two matching edges by ETOF-K algorithm and 175289 matching subgraphs associated with the same two matching edges by introducing fuzzy into ETOF-K algorithm. It is 28.90 % more effective than ETOF-K. Similarly, we can see that the Fuzzy-ETOF-K algorithm is 20.83% and 31.16% more efficient than ETOF-K on the DBLP dataset and on the Pokec dataset, respectively. This proves that it is necessary and more effective to introduce fuzzy constraints into MC-GPM.

6 Conclusion

In this paper, we presented a Multi-Fuzzy-Constrained Graph Pattern Matching (MFC-GPM) problem and then proposed a Fuzzy-ETOF-K algorithm to solve it. For Multi-Constrained Graph Pattern Matching (MC-GPM) problem, we proposed an ETOF-K algorithm to improve its efficiency and conducted validation experiments on three real social network datasets by comparing Baseline, HAMC, Baseline2 and ETOF-K algorithms. Furthermore, we proved the necessity and effectiveness of introducing fuzzy constraints into MC-GPM problem by comparing ETOF-K and Fuzzy-ETOF-K algorithms.

In the future, we will further study and improve the Fuzzy-ETOF-K algorithm, and consider the dynamic graph and super-large graph pattern matching problem in combination with parallel computing and distributed computing techniques. Research combined with practical applications is also under consideration.

Acknowledgments

This work has been supported by the National Key Research and Development Program of China under Grant No. 2016YF-B1000901, the National Natural

Science Foundation of China under Grant No. 91746209, and the Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT) of the Ministry of Education under Grant No. IRT17R32.

References

[1] X. Wu, X. Zhu, G. Wu, Data mining with big data, IEEE Transactions on Knowledge and Data Engineering 26(1) (2014), 97-107.

[2] L. Li, J. He, M. Wang, X. Wu, Trust agent-based behavior induction in social networks, IEEE Intelligent Systems 31(1) (2016), 24-30.

[3] H. Wei, J. X. Yu, C. Lu, R. Jin, Reachability querying: an independent permutation labeling approach, The VLDB JournalThe International Journal on Very Large Data Bases 27(1) (2018), 1-26.

[4] D. Eppstein, Finding the k shortest paths, SIAM Journal on computing 28(2) (1998), 652-673.

[5] A. Ray, L. B. Holder, A. Bifet, Efficient frequent subgraph mining on large streaming graphs, Intelligent Data Analysis 23(1) (2019), 103-132.

[6] S. Zhang, Z. Du, J. T. L. Wang, H. Jiang, Discovering frequent induced subgraphs from directed networks, Intelligent Data Analysis 22(6) (2018), 1279-1296.

[7] S. Liu, and C.K. Poon, On mining approximate and exact fault-tolerant frequent itemsets, Knowledge and Information Systems, 55 (2) (2018), 361-391.

[8] Z. Peng, T. Wang, W. Lu, H. Huang, X. Du, F. Zhao, and A.K.H. Tung, Mining frequent subgraphs from tremendous amount of small graphs using MapReduce, Knowledge and Information Systems, 56 (3) (2018), 663-690.

[9] W. Zheng, L. Zou, X. Lian, H. Zhang, W. Wang, D. Zhao, SQBC: An efficient subgraph matching method over large and dense graphs, Information Sciences 261 (2014), 116-131.

[10] D. Natarajan, and S. Ranu, Resling: A scalable and generic framework to mine top-k representative subgraph patterns, Knowledge and Information Systems, 54 (1) (2018), 123-149.

[11] T. Tran, H. Wang, S. Rudolph, P. Cimiano, Top-k exploration of query candidates for efficient keyword Search on graph-shaped (RDF) data, IEEE 25th International Conference on Data Engineering, IEEE, Shanghai, China, 2009, pp. 405-416.

[12] S. G. Baek, D. H. Kang, S. Lee, Y. I. Eom, Efficient graph pattern matching framework for network-based in-vehicle fault detection, Journal of Systems and Software 140 (2018), 17-31.

[13] J. R. Ullmann, An algorithm for subgraph isomorphism, Journal of the ACM (JACM) 23(1) (1976), 31-42.

[14] L. P. Cordella, P. Foggia, C. Sansone, M. Vento, A (sub) graph isomorphism algorithm for matching large graphs, IEEE transactions on pattern analysis and machine intelligence 26(10) (2004), 1367-1372.

[15] J. T. Hu, A. L. Ferguson, Global graph matching using diffusion maps, Intelligent Data Analysis 20(3) (2016), 637-654.

[16] W. Fan, X. Wang, Y. Wu, Expfinder: Finding experts by graph pattern matching, IEEE 29th International Conference on Data Engineering, IEEE, Brisbane, QLD, Australia, 2013, pp. 1316-1319.

[17] D. Tang, Y. Zhang, Y. He, Z. Xiao, Research and application on crime rule based on graph data mining algorithm, Computer Technology and Development 11 (2011), 89-91.

[18] W. Fan, X. Wang, Y. Wu, Incremental graph pattern matching, ACM Transactions on Database Systems (TODS) 38(3) (2013), 18.

[19] M. Lenin, E. William, M. Maitrayi, Personalized news recommendation using graph-based approach, Intelligent Data Analysis 22(4) (2018), 881-909.

[20] C. Li, H. Chen, R. Chen, and H. Hsieh, On route planning by inferring visiting time, modeling user preferences, and mining representative trip patterns, Knowledge and Information Systems, 56 (3) (2018), 581-611.

[21] K.H. Lim, J. Chan, C. Leckie, and S. Karunasekera, Personalized trip recommendation for tourists based on user interests, points of interest visit durations and visit recency, Knowledge and Information Systems, 54 (2) (2018), 375-406.

[22] W. Fan, J. Li, S. Ma, N. Tang, Y. Wu, Y. Wu, Graph pattern matching: from intractable to polynomial time, Proceedings of the VLDB Endowment 3(1-2) (2010), 264-275.

[23] G. Liu, K. Zheng, Y. Wang, M. A. Orgun, A. Liu, L. Zhao, X. Zhou, Multi-constrained graph pattern matching in large-scale contextual social graphs, IEEE 31st International Conference on Data Engineering, IEEE, Seoul, South Korea, 2015, pp. 351-362.

[24] M. R. Henzinger, T. A. Henzinger, P. W. Kopke, Computing simulations on finite and infinite graphs, IEEE 36th Annual Foundations of Computer Science, IEEE, Milwaukee, WI, USA, 1995, pp. 453-462.

[25] S. Zhang, S. Li, J. Yang, GADDI: Distance index based subgraph matching in biological networks, Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, ACM, Saint Petersburg, Russia, 2009, pp. 192-203.

[26] C. Wickramaarachchi, R. Kannan, C. Chelmis, V. K. Prasanna, Distributed exact subgraph matching in small diameter dynamic graphs, IEEE International Conference on Big Data, IEEE, Washington, DC, USA, 2017, pp. 3360-3369.

[27] X. Yan, P. S. Yu, J. Han, Graph indexing: a frequent structure-based approach, Proceedings of the 2004 ACM SIGMOD international conference on Management of data, SIGMOD, Paris, France, 2004, pp. 335-346.

[28] D. Shasha, J. T. L. Wang, R. Giugno, Algorithmics and applications of tree and graph searching, Acm Sigmod-sigact-sigart Symposium on Principles of Database Systems, ACM, Madison, Wisconsin, 2002.

[29] J. Cheng, Y. Ke, W. Ng, A. Lu, Fg-index: towards verification-free query processing on graph databases, Proceedings of the 2007 ACM SIGMOD international conference on Management of data, ACM, Beijing, China, 2007, pp. 857-872.

[30] J. Mondal, A. Deshpande, CASQD: continuous detection of activity-based subgraph pattern queries on dynamic graphs, Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems, ACM, Irvine, California, 2016, pp. 226-237.

[31] H. Tran, J. Kim, B. He, Fast subgraph matching on large graphs using graphics processors, Database Systems for Advanced Applications, DASFAA, Hanoi, Vietnam, 2015, pp. 299-315.

[32] F. N. Afrati, D. Fotakis, J. D. Ullman, Enumerating subgraph instances using map-reduce, IEEE 29th International Conference on Data Engineering, IEEE, Brisbane, QLD, Australia, 2013, pp. 62-73.

[33] Y. Shao, B. Cui, C. Lei, M. Lin, J. Yao, X. Ning, Parallel subgraph listing in a large-scale graph, Acm Sigmod International Conference on Management of Data, ACM, Snowbird, Utah, USA, 2014, pp. 625-636.

[34] J. Gao, B. Song, P. Liu, W. Ke, J. Wang, X. Hu, Parallel top-k subgraph query in massive graphs: Computing from the perspective of single vertex, IEEE International Conference on Big Data, IEEE, Washington, DC, USA, 2017, pp. 636-645.

[35] W. Fan, X. Wang, Y. Wu, Answering graph pattern queries using views, IEEE 30th International Conference on Data Engineering, IEEE, Chicago, IL, USA, 2014, pp. 184-195.

[36] W. Fan, X. Wang, Y. Wu, Querying big graphs within bounded resources, Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, 2014, pp. 301-312.

[37] S. Ma, Y. Cao, W. Fan, J. Huai, T. Wo, Strong simulation: Capturing topology in graph pattern matching, ACM Transactions on Database Systems (TODS) 39(1) (2014), 4.

[38] A. Shemshadi, Q. Z. Sheng, Y. Qin, Efficient pattern matching for graphs with multi-Labeled nodes, Knowledge-Based Systems 109 (2016), 256-265.

[39] R. Milano, R. Baggio, R. Piattelli, The effects of online social media on tourism websites, ENTER2011 18th International Conference on Information Technology and Travel & Tourism, Congress und Messe Innsbruck GmbH, Innsbruck, Austria, 2011, pp. 471-483.

[40] G. Liu, Y. Wang, M. A. Orgun, Optimal social trust path selection in complex social networks, AAAI Conference on Artificial Intelligence, AAAI, Atlanta, GA, 2010.

[41] Q. Shi, G. Liu, K. Zheng, A. Liu, Z. Li, L. Zhao, X. Zhou, Multi-constrained top-K graph pattern matching in contextual social graphs, IEEE International Conference on Web Services, IEEE, Honolulu, HI, USA, 2017, pp. 588-595.

[42] G. Liu, Y. Liu, K. Zheng, A. Liu, Z. Li, Y. Wang, X. Zhou, MCS-GPM: Multi-constrained simulation based graph pattern matching in contextual social graphs, IEEE Transactions on Knowledge and Data Engineering 30(6) (2018), 1050-1064.