Noname manuscript No.
(will be inserted by the editor)

Privacy and Efficiency Guaranteed Social Subgraph Matching

Kai Huang · Haibo Hu · Shuigeng Zhou · Jihong Guan · Qingqing Ye · Xiaofang Zhou

Received: date / Accepted: date

Abstract Due to the increasing cost of data storage and computation, more and more graphs (e.g., web graphs, social networks) are outsourced and analyzed in the cloud. However, there is growing concern on the privacy of these outsourced graphs at the hands of untrusted cloud providers. Unfortunately, simple label anonymization cannot protect nodes from being re-identified by adversary who knows the graph structure. To address this issue, existing works adopt the k-automorphism model, which constructs $(k-1)$ symmetric vertices for each vertex. It has two disadvantages. First, it significantly enlarges the graphs, which makes graph mining tasks such as subgraph matching extremely inefficient and sometimes infeasible even in the cloud. Second, it cannot protect the privacy of attributes in each node. In this paper, we propose a new privacy model (k,t)-privacy that combines the k-automorphism model for graph structure with the t-closeness privacy model for node label generalization. Besides a stronger privacy guarantee, the paper also optimizes the matching efficiency by (1) an approximate label generalization algorithm $TOGGLE$ with $(1+\epsilon)$ approximation ratio, and (2) a new subgraph matching algorithm PGP on succinct k-automorphic graphs without decomposing the query graph.

Keywords (k,t)-privacy · label generalization · subgraph matching

Kai Huang, Shuigeng Zhou
Shanghai Key Lab of Intelligent Information Processing
School of Computer Science, Fudan University, China.
E-mail: khuang14, sgzhou@fudan.edu.cn

Haibo Hu, Qingqing Ye
Department of Electronic and Information Engineering
Hong Kong Polytechnic University, Kowloon
Hong Kong and PolyU Shenzhen Research Institute.
E-mail: haibo.hu, qqing.ye@polyu.edu.hk

Jihong Guan
Department of Computer Science and Technology
Tongji University, China.
E-mail: jhguan@tongji.edu.cn

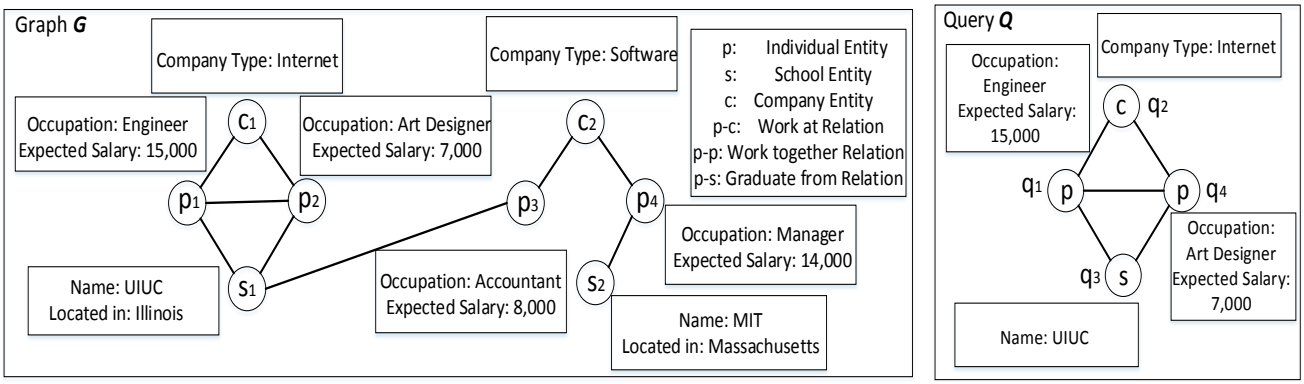
Xiaofang Zhou
Department of Computer Science
The University of Queensland, Australia.
E-mail: zxf@itee.uq.edu.au

1 Introduction

Attributed graph, a subtype of graph where each node contains a set of attributes (as shown in Figure 1(a)), has become increasingly popular to model web and social networks. Many graph queries are developed to analyze and retrieve the rich semantic and structural information of these graphs. Among them the subgraph matching, which retrieves all subgraphs isomorphic to a given query graph, is fundamental to many graph data analytics [1]. However, as graph data size continues to grow, storing and processing them imposes expensive upfront infrastructure costs on users. As such, many cloud service providers such as Amazon, Alibaba, and Microsoft Azure offer graph outsourcing services by storing graphs owned by users and executing mining tasks on behalf of them. GraphLab [2] even provides a graph-based Software-as-a-Service (SaaS).

However, the cloud server is considered as "honest-but-curious" (a.k.a., semi-honest), which is consistent with most related works in the literature [3,43,4,5]. On the one hand, the cloud server acts in an "honest" fashion, i.e., correctly follows the designated protocol specification such as HIPAA compliance [1] and always offers correct computations without cheating. On the other hand, it is "curious" to infer the sensitive information of the graph data by analyzing index structures, requested queries, and query results. This assumption does not reduce the need for storing and processing

[1] https://aws.amazon.com/compliance/hipaa-compliance/



(a) G (b) Q

Fig. 1 Original data graph G and query Q.

graphs on the untrusted cloud provider. For example, Paysafe, a Fintech company, needs to store and analyze the payments data including attributes and interactions of customers and their payments (which are modeled as a large graph), but it is unwilling to spend expensive upfront infrastructure costs on it. Hence, the company resorts to a graph-based cloud service, *Graph Database and Graph Analytics* [2] provided by ORACLE [44], even it is assumed that the cloud provider is "honest-but-curious". In general, cloud servers can learn the attribute values (a.k.a., labels) and structural information of the outsourced graphs. Privacy breaches on these graphs are known to disclose sensitive information [3], which can be grouped to two categories: *content disclosure* and *identity disclosure*. *Content disclosure* compromises sensitive label information, such as salary, social security number, or medical history of a user. To guard against content disclosure, three classic privacy preserving techniques have been proposed, namely, k-anonymity [6], ℓ-diversity [7] and t-closeness [8]. They aim to generalize several labels into a single one (also known as an equivalence class) to hide the sensitive information of a single label so that the attacker can only see the generalized label. In particular, k-anonymity requires that each equivalence class contains at least k records so that each record is indistinguishable with at least $k-1$ other records. ℓ-diversity requires that the distribution of a sensitive attribute in each equivalence class has at least ℓ "well-represented" values. While t-closeness requires the label distribution in each equivalence class is no more than t distance away from that in the whole set of labels. It is known that t-closeness is able to resist more attacks (such as similarity attack) than k-anonymity and ℓ-diversity, and is the most stringent privacy metric among the three.[3] However, when it comes to protecting labels in attributed graphs, while [3] and [10] adopt k-anonymity and ℓ-diversity respectively, there is no work that adopts t-closeness. *Identity disclosure* [11,12] compromises the location of a target node in the graph even after removing the node's identity. This disclosure can be caused by various structural attacks [13,11,14] such as degree attack, 1-neighbor-graph attack, subgraph attack and hub-fingerprint attack. To guard against these attacks, many structure-privacy-preserving techniques [15–17] have been developed to enforce symmetry in an outsourced graph. In particular, [15] proposes the k-automorphism model. Given a graph G, it transforms G into a k-automorphic graph G^k by introducing some noise edges and vertices, where each vertex has at least $(k-1)$ other symmetric vertices. Hence, there are no structural differences between any vertex and its $(k-1)$ symmetric vertices. In other words, the attacker can not distinguish a vertex from the other $(k-1)$ symmetric vertices. And it is known that k-automorphism strategy can defend against any structure-based attack [15].

Example 1 *Consider the graph G in Figure 1(a) where each vertex represents an entity. There are three entities, individual entity (p), school entity (s) and company entity (c). The edges in G represent the relations between two entities, such as "Work at" relation, "Work together" relation and "Graduate from" relation. If an adversary knows that one person has a 1-degree neighbor node in the graph G, he can immediately know that node p_4 is that person and the related attributes of node p_4 are revealed. k-automorphism model can be applied to prevent such structure attack by introducing noise edges to construct a k-automorphic graph G^k. For example, G^k in Figure 2(a) is a k-automorphic graph of G where $k = 2$. In this figure, the noise edges are shown by black dashed lines. Note that each node contains a set of attributes where sensitive information (e.g. salary) are contained. In this case, the privacy model for structural privacy (e.g. k-automorphism) is not sufficient to protect the label privacy of each node. For example, even for the k-*

[2] https://www.oracle.com/database/graph/

[3] One can argue that differential privacy [9] is more stringent than t-closeness as the former is defined regardless of the underlying dataset or apriori knowledge. However, it is infeasible in subgraph matching where *exact* matchings are desirable.

automorphic graph G^k, if the labels are not anonymized, the adversary can use some prior knowledge or background knowledge (e.g. the "Occupation" attribute in the G, Figure 1(a)) to identify the location of an individual. Suppose the adversary can identify an individual as the node p_1 or p_4 of G in Figure 1(a), then he can conclude that this person's salary is high ("14000" or "15000") without exactly reidentifying the node. Therefore, when sensitive labels are considered, the t-closeness should be adopted for graphs to generalize each label into a generalized one as shown in Figure 2(a).

The example above shows the necessity of preserving both structural privacy and label privacy. However, except for [3,59], which only support k-anonymity, all existing works on attributed graphs consider either structural or label privacy. In this paper we propose (k,t)-privacy, an integrated privacy model for outsourced graphs. Any graph that satisfies (k,t)-privacy must satisfy both k-automorphism and t-closeness for each generalized label of all node attributes. Intuitively, the (k,t)-privacy model can preserve both structural privacy and label privacy.

Once the graph G is anonymized, a straightforward method is to upload the k-automorphic graph G^k to the cloud and perform subgraph queries over G^k. However, there are some major limitations. Firstly, a lot of noise edges and vertices will be introduced to the original graph G when constructing the k-automorphic graph G^k. This results in a much larger graph on the cloud side, which leads to much more expensive storage cost and much higher query cost. Although Chang et al. [3] propose to store only a succinct version of k-automorphic graphs, query decomposition is adopted during subgraph matching, which causes an extra result-joining step and incurs a large number of false positive results. Secondly, while imposing t-closeness on the labels in a graph G, different outputs of label generalization have different impacts on not only the privacy strength but also the search space (i.e. number of vertices or matchings to explore for a query, see Section 4). Intuitively, the larger search space will cause expensive query cost. The second drawback is illustrated with the following example.

Example 2 *Consider the attribute "Expected Salary" of the graph G in Figure 1(a), the original set of labels $l = (7000, 8000, 14000, 15000)$. There are two possible label generalizations. The first generalizes $(7000, 8000)$ as a generalized label "A" and $(14000, 15000)$ as another generalized label "B". In this case, given the query Q in Figure 1(b) and its anonymized form, there are two matchings, (c_1, p_1, p_2, s_1) and (c_2, p_3, p_4, s_2). The second generalization is to generalize $(7000, 15000)$ as "A" and $(8000, 14000)$ as "B". In this case, there is only one matching (c_1, p_1, p_2, s_1). Obviously, the first generalization will lead to a false positive matching (i.e., (c_2, p_3, p_4, s_2)).*

To address subgraph matching efficiency problem due to the enlarged graph size caused by $k-$automorphism and the enlarged search space caused by the generalization using $t-$closeness, we model the cost of subgraph matching and reduce the problem of optimal generalization of labels for $t-$closeness to the *General Set Partitioning Problem* [18]. Based on this, we propose an efficient approximation algorithm $TOGGLE$ with a bounded approximation ratio of $(1+\epsilon)$. Furthermore, as the cloud only stores a succinct version of k-automorphic graphs, query decomposition is adopted during subgraph matching [3], which causes an extra result-joining step and incurs a large number of false positive results. We design a new subgraph matching algorithm PGP that can directly work on such graphs without decomposition. To summarize, our main contributions are as follows:

(i) We propose (k,t)-privacy for outsourced graphs, which is, to the best of our knowledge, the most stringent generalization based privacy model for both graph structure and label generalization.
(ii) We propose a t-closeness label generalization algorithm $TOGGLE$ that optimizes costs for subgraph matching. It is proved to have an approximation ratio of $(1+\epsilon)$.
(iii) We also develop a partial-graph-based subgraph processing algorithm PGP without query decomposition. It exploits the symmetry of the outsourced graph and limits search scope to a localized region.
(iv) We conduct an empirical study on our algorithms in all three datasets ever experimented in the literature and show their high efficiency under various parameter settings.

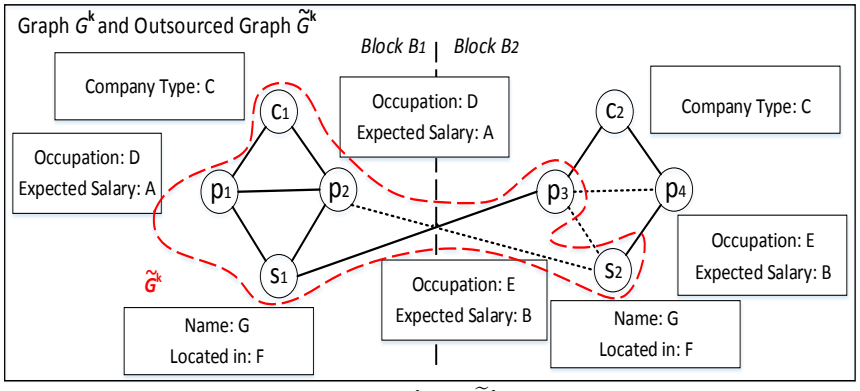
The remainder of this paper is organized as follows. Section 2 presents the background and problem statement. Section 3 overviews the graph outsourcing and subgraph matching framework with a baseline solution adapted from [3]. In Section 4, we present the label generalization algorithm $TOGGLE$ and in Section 5, we introduce the PGP subgraph matching algorithm. Experiments and related works are presented in Sections 6 and 7, respectively, followed by a conclusion in Section 8. Formal proofs of lemmas are in the Appendix A.

2 Background and Problem Statement

In this section, we first introduce the background of two privacy models in (k,t)-privacy, namely t-closeness and k-automorphism. We then present the privacy attacks on graphs and summarize them as the threat model for this paper. Finally, the formal definition of privacy-preserving subgraph matching with (k,t)-privacy is presented. Table 1 lists the key notations and acronyms used in this paper.

Table 1 List of key notations.

Notation	Description
G, G^k, $\widetilde{G}^k$	a graph, k-automorphic graph, outsourced graph
LCT	Label Correspondence Table
Q, $\widetilde{Q}$	a query graph, outsourced query
EMD	Earth Mover's Distance
l, n	labels, number of labels
y_j, m	label group, number of label groups
τ, n_τ, m_τ	vertex type, number of labels and number of label groups of type τ
$F_G(\tau)$, $F_Q(\tau)$	probability of vertices in a graph G (resp. query Q) being the type τ



(a) G^k and $\widetilde{G}^k$

Label Group	Labels
A	7000, 15000
B	8000, 14000
C	Internet, Software
D	Engineer, Art Designer
E	Accountant, Manager
F	Illinois, Massachusetts
G	UIUC, MIT

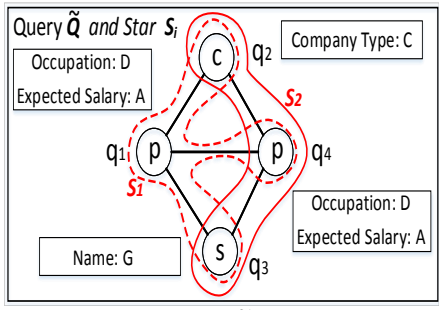
(b) LCT

Fig. 2 k-automorphic data graph G^k, outsourced graph $\widetilde{G}^k$ and Label Correspondence Table (LCT).

Alignment Vertex Table	
p_1	p_4
p_2	p_3
s_1	s_2
c_1	c_2

Automorphic Function
$F_1(p_1) = p_4$; $F_1(p_4) = p_1$

(a) AVT



(b) $\widetilde{Q}$

Fig. 3 Alignment Vertex Table and Outsourced query $\widetilde{Q}$.

2.1 t-Closeness

Generalization, which combines several values into a single one (also known as an equivalence class), has been a popular anonymization technique for labels [19,20,3]. The output of a label generalization algorithm is a *label correspondence table* which maps this correspondence. t-closeness [8] is a privacy metric that imposes a constraint on each equivalence class.

Definition 1 ***(t-closeness)*** *An equivalence class satisfies t-closeness if and only if the label distribution in this class is no more than t distance away from that in the whole set of labels. If all equivalence classes satisfy t-closeness, the label generalization algorithm satisfies t-closeness.*

The distance is measured by the Earth Mover's Distance (EMD) [21], which is based on the minimum workload to transform one distribution to another. Specifically, each distribution is viewed as a mass of earth; and in each step of the transformation, some earth is moved from one place to another. The moved mass multiplied by the ground distance of this move is the workload of this step and added to the total workload. To find the minimum workload, the EMD computation relies on the well-known combinational optimization problem — balanced transportation. Formally, let $\mathcal{P} = (p(v_1), p(v_2), ..., p(v_n))$ and $\mathcal{P}' = (p(v'_1), p(v'_2), ..., p(v'_m))$ denote the two distributions, and $d_{i,j}$ be the ground distance between v_i and v'_j, [4] the balanced transportation finds a mass flow $F = \{f_{i,j}\}, i \in [1,n], j \in [1,m]$ such that $\sum_{i=1}^{n}\sum_{j=1}^{m} d_{i,j} \times f_{i,j}$ is minimum, subjects to $f_{i,j} \geq 0$, and $\sum_{i=1}^{n}\sum_{j=1}^{m} f_{i,j} = \sum_{i=1}^{n} p(v_i) = \sum_{j=1}^{m} p(v'_j) = 1$. Solving the mass flow F, we can obtain the Earth Mover's

[4] We will directly use the notation $(v_1, v_2, ...v_i, ...v_j...v_n)$ to denote the uniform distribution where each value is equally likely. For sorted numerical values $(v_1, v_2, ...v_i, ...v_j...v_n)$, the ground distance of v_i and v_j is $\frac{|i-j|}{n-1}$ [8].

Distance between distributions $\mathcal{P}$ and $\mathcal{P}'$ as: $EMD(\mathcal{P}, \mathcal{P}') = \sum_{i=1}^{n} \sum_{j=1}^{m} d_{i,j} \times f_{i,j}$.

Example 3 *Figure 2(b) shows a label correspondence table. For attribute "Expected Salary" of graph G in Figure 1(a), the original set of labels* $l = (7000, 8000, 14000, 15000)$. *There are two possible label generalizations. The first combines* $(7000, 8000)$ *as a label group "A" and* $(14000, 15000)$ *as another group "B". In this case,* $EMD(l, A) = (0/3 + 1/3 + 1/3 + 2/3) * 1/4 = 1/3$, *where* $0/3$ *is the ground distance* $d_{0,0}$ *and* $1/4$ *is the mass* $f_{0,0}$. *Similarly,* $EMD(l, B) = 1/3$. *As such, this method satisfies* $1/3$*-closeness. The second generalization, as illustrated in Figure 2(b), combines* $(7000, 15000)$ *as "A" and* $(8000, 14000)$ *as "B". In this case,* $EMD(l, A)$ *and* $EMD(l, B)$ *are both* $1/6$. *Therefore ,this method satisfies* $1/6$*-closeness, which preserves more privacy than the first generalization.*

2.2 k-Automorphism

k-automorphism is a graph privacy model that can defend existing structural attacks [15], including degree attack, 1-neighbor-graph attack, subgraph attack and hub-fingerprint attack [13,11,14]. The idea is to construct $(k-1)$ symmetric blocks for each block (i.e., a subset of vertices and their corresponding edges) in a graph, so that a vertex cannot be distinguished from other $(k-1)$ vertices in those symmetric blocks. The resulted graph is a *k-automorphic graph*. Converting a graph G to a *k*-automorphic graph G^k involves three steps — graph partition, graph (block) alignment, and edge copy. First, the vertices in G are partitioned into k blocks by adopting the METIS algorithm [22]. Second, graph (block) alignment selects and aligns those vertices which have the largest degree in each block, and then aligns all other vertices in the same block with those vertices in other blocks by their breath-first search (BFS) traversal order. The result is an *alignment vertex table (AVT)* where each column corresponds to one block and each row denotes the mapping of k vertices. The same mapping can also be recorded by an *Automorphic Function*. Third and finally, based on AVT, symmetry edges are inserted in other $(k-1)$ blocks for each edge in one block. Crossing edges between two blocks are also copied accordingly.

Example 4 *Graph G in Figure 1(a) is first partitioned into* $k = 2$ *blocks,* B_1=$\{p_1, p_2, s_1, c_1\}$ *and* B_2=$\{p_4, p_3, s_2, c_2\}$ *(see Figure 2(a)). By graph alignment, each vertex in one block is aligned with that in the other. For example,* p_1 *is aligned with* p_4, *so* (p_1, p_4) *is inserted to the alignment vertex table in Figure 3(a). Equivalently, their mapping is recorded by a k-dimensional automorphic function* F_1. *After all vertices are aligned, the missing symmetry edges between* p_3 *and* p_4, *and between* p_3 *and* s_2 *(the dashed line in Figure 2(a)) are inserted. Finally, the crossing edge between* s_1 *and* p_3 *is copied between* p_2 *and* s_2. *The resulted k-automorphic graph* G^k *is shown in Figure 2(a).*

2.3 Graph Privacy Attacks and Threat Model

In this paper, we study attributed graph, which is an undirected graph with attributes on each node [3].

Definition 2 ***(Attributed Graph)*** *An attributed graph is defined as* $G = (V, E, T, \Gamma, L)$ *where* V *is a vertex set;* E *is an edge set; and* T, Γ, L *denote the sets of vertex types, attributes and labels, respectively. Each vertex has a unique type and one or more attributes (depending on the type). Each attribute takes one from a set of labels as its value.*

Example 5 *The data graph G in Figure 1(a) is an attributed graph with three vertex types, namely, "Individual", "University" and "Company". "Individual" contains two attributes, namely "Occupation" and "Expected Salary". The label value of "Expected Salary" of vertex* p_1 *is "15,000".*

In the literature, privacy attacks on outsourced graphs lead to identity disclosure (*structural attacks*) and content disclosure (*label attacks*). Structural attacks include degree attack, 1-neighbor-graph attack, subgraph attack and hub-fingerprint attack [13,11,14]. Label attacks include background-knowledge attack, homogeneity attack, skewness attack and similarity attack [7,8]. A common approach in these attacks is to first identify the target node and then unveil its sensitive labels. The following definition abstracts the capability of all known attacks as above, and serves as the threat model of this paper.

Definition 3 ***(Threat Model of Graph Privacy Attacks)*** *An attacker has the complete structure information about the target node, including degree, neighbor list, and shortest-distances from known nodes (a.k.a., hubs). She also has the complete label information of all nodes except for the target node. Based on such information, she attempts to match the target node in the outsourced graph and then to unveil its label.*

2.4 Problem Statement

Our problem of privacy-preserving subgraph matching with (k, t)-privacy involves two sub-problems.

Definition 4 ***(Graph Outsourcing Problem with*** (k, t)***-Privacy)*** *Given a data graph G, to compute an outsourced graph to the cloud that satisfies the following two privacy metrics.*

(i) Preserving label privacy by enforcing t-closeness for its labels,

(ii) Preserving structure privacy by enforcing k-automorphism for its structure.

Privacy Guarantee of (k,t)-Privacy. For any outsourced graph that satisfies (k,t)-privacy, the probability that an attacker correctly re-identifies a target node using any structural information is at most $1/k$, as the k-automorphism model ensures each node is structurally indistinguishable from at least $k-1$ other nodes. Even within the $1/k$ probability of which the attacker re-identifies the target node, she can only learn limited information about the node's true label as this label has been generalized to a label group whose distribution of underlying labels is at most t distance away from the distribution of all labels in this graph.

Definition 5 ***(Subgraph Matching Problem on Outsourced Graph)*** *Given a data graph G and its corresponding outsourced graph, for a query Q, to retrieve all subgraphs $\{g_i\}$ of G, each of which is subgraph isomorphic to Q and vice versa. $Q = (V_1, E_1, T_1, \Gamma_1, L_1)$ is subgraph isomorphic to $g_i = (V_2, E_2, T_2, \Gamma_2, L_2)$ if and only if there exists at least one injection function $f : Q \rightarrow g_i$ such that*

(i) $\forall u \in V_1, \exists f(u) \in V_2$ such that $T_1(u) = T_2(f(u))$ and $L_1(u) \subseteq L_2(f(u))$.

(ii) $\forall (u,v) \in E_1, (f(u), f(v)) \in E_2$.

Example 6 *Figure 1(b) presents a query Q. A UIUC spin-off company wants to hunt two employees in a professional social network. The searching criteria are as follows: (1) they both graduated from UIUC, (2) they are working for the same Internet company, and (3) one is an engineer whose expected salary is $15,000$/mo, and the other is an art designer whose expected salary is $7,000$/mo.*

Theorem 1 *Both graph outsourcing problem with (k,t)-privacy and subgraph matching problem on outsourced graph are NP-Hard.*

3 Solution Overview

In this section, we overview our solution to graph outsourcing and subgraph matching on outsourced graphs. The workflow of our solution is depicted in Figure 4. Given a data graph G, the client user anonymizes it to satisfy (k,t)-privacy (step ①). The result consists of a k-automorphism graph G^k and a label corresponding table LCT. Since each vertex in G^k has $(k-1)$ symmetric vertices in the other $(k-1)$ blocks, the client user only needs to upload a succinct version of k-automorphism graph $\widetilde{G}^k$ to cloud (step ②), which is also called an outsourced graph [3]. $\widetilde{G}^k$ comprises of vertices and edges in one block together with their 1-hop neighbouring vertices and edges. For example, the one inside red dashed circle in Figure 2(a) is the succinct $\widetilde{G}^k$.

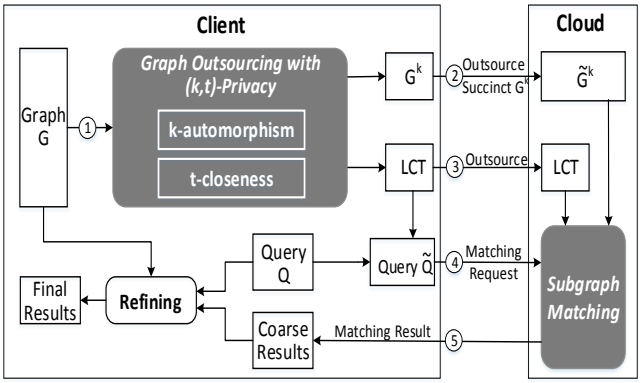


Fig. 4 Workflow of subgraph matching on outsourced graph

Upon receiving a matching query $\widetilde{Q}$ (step ④) transformed from the client's original query Q (e.g., Figure 3(b) from Figure 1(b)), the cloud evaluates it based on $\widetilde{G}^k$ and LCT. The results are sent back to the client (see step ⑤) for further refining to filter those false positive results. In the rest of this paper, we only focus on two key components highlighted in Figure 4, namely, **graph outsourcing with (k,t)-privacy** and **subgraph matching**.

In the rest of this section, we propose a baseline solution by adapting existing work [3] to generalize labels into label groups. Then we discuss its limitations, based on which we propose two new algorithms for label generalization and subgraph matching in Sections 4 and 5, respectively.

3.1 Baseline Solution

3.1.1 Graph Outsourcing with (k,t)-Privacy

The idea is to work on k-automorphism and t-closeness separately. We first transform the original graph G to a k automorphic graph G^k, and then adapt existing label generalization algorithm in [3] to satisfy t-closeness metric. Given a vertex type that contains n vertex labels $l = (l_1, l_2, ..., l_n)$, we propose Algorithm 1 to generalize n labels into m groups. It first permutates the set of labels into $P = (l_{p_1}, l_{p_2}, ..., l_{p_n})$ and then sequentially divides P into m groups, each of which contains n/m labels and satisfies t-closeness.

In particular, Algorithm 1 initializes the global $minScore$ and the permutation $\{y_j\}$ (Lines 2 and 3) where y_j is a label group and $j \in \{1, 2, ..., m\}$. Then the function $groupEnum$ is called recursively to find the optimal permutation by gradually expanding the candidate permutation P (Line 4).

In this $groupEnum$ function, it first checks whether the current P is a complete permutation of size n (Line 6). If so, its cost (e.g., a estimated search space) is calculated by $getCost(P)$ (Line 7). After that, P is updated to $\{y_j\}$ if the cost is lower than the current lowest (Lines 8 and 9). Otherwise, it greedily enumerates n/m labels as a label group

Algorithm 1 A Baseline Method for Label Generalization

Input: Labels l, number of labels n, group number m, threshold t

Output: An optimal label generalization into m label groups

```
1:  procedure BasSol(l, n, m, t)
2:      minScore ← +∞;
3:      {y_j} ← {∅};
4:      groupEnum({y_j}, P ← ∅, l, n, m, t);
5:  function groupEnum({y_j}, P, l, n, m, t)
6:      if |P|== n then
7:        if getCost(P) < minScore then
8:          minScore ← getCost(P);
9:          {y_j} ← P;
10:         return {y_j};
11:     enumerate n/m labels from l to P_s;
12:     while y'_j ∈ P_s do
13:       if checkTclo(y'_j, t) then
14:         move y'_j from l to P ;
15:         groupEnum({y_j}, P, l, n, m, t);
16:         move y'_j from P to l;
17:         return {y_j};
```

from l and records them in the set P_s (Line 11). For each label group y'_j in this set, a $checkTclo$ routine calculates EMD by solving a transportation problem [21] and then checks whether y'_j satisfies t-closeness (Line 13). If it is true, y'_j will be appended to the candidate permutation P and removed from the labels l (Line 14).

Complexity Analysis. The space complexity of Algorithm 1 is $O(n)$. As for time complexity, let t_1 denote the time complexity of $checkTclo$, which is invoked at most $\frac{n!}{((n/m)!)^m}$ times. As such, the time complexity is $O(\frac{n!t_1}{((n/m)!)^m})$, which is exponential to n.

3.1.2 Subgraph Matching

As described in [3], for a subgraph matching query Q at the client, the client first generalizes its vertex labels by the label correspondence table and sends the generalized query $\widetilde{Q}$ to the cloud. Since the cloud can only access the succinct graph $\widetilde{G}^k$, which consists of one block of G^k together with its 1-hop neighbours, it uses a special star-based subgraph matching algorithm. On the cloud side, the algorithm consists of three steps.

(i) *Query decomposition*. The cloud first decomposes query $\widetilde{Q}$ (e.g., Figure 3(b)) into a set of star shapes $\{S_i\}$ (see Figure 3(b)), each of which has a root vertex together with its adjacent edges and neighbouring vertices.

(ii) *Star matching*. The cloud then retrieves matchings for each decomposed star S_i in succinct graph $\widetilde{G}^k$, denoted by $R(S_i, \widetilde{G}^k)$, and leverages the symmetry of G^k to retrieve the matchings for S_i in G^k, denoted by $R(S_i, G^k)$.

(iii) *Star joining*. The cloud starts with a $R(S_i, G^k)$ and iteratively computes its natural join with $R(S_j, G^k)$, until all $j \neq i$ are joint up. The results $R(\widetilde{Q}, G^k)$ are the matchings for $\widetilde{Q}$ over G^k.

On the client side, the algorithm filters false positives in $R(\widetilde{Q}, G^k)$ using the original graph G and query Q, and obtains the final subgraph matchings $R(Q, G)$.

3.2 Limitations

Though the baseline solution can support privacy-preserving subgraph matching with (k, t)-privacy, it has two main drawbacks. First, the time complexity of label generalization is exponential to the number of labels as all permutations need to be considered. Second, the star-based subgraph matching is also inefficient because: (1) due to query decomposition it cannot narrow down the search scope of the query to a localized region in the graph; and (2) the natural join is computationally intensive. We illustrate the second drawback with an example.

Example 7 *Figure 3(b) shows a generalized query graph $\widetilde{Q}$. The baseline algorithm decomposes $\widetilde{Q}$ into two stars, S_1 and S_2. Three matchings are found for S_1 on $\widetilde{G}^k$, namely, (p_1, c_1, s_1, p_2), (p_2, c_1, s_1, p_1) and (p_2, c_1, s_2, p_1). Similarly, three matchings are found for S_2 on $\widetilde{G}^k$, namely, (c_1, s_1, p_1) , (c_1, s_1, p_2) and (c_1, s_2, p_2). As such, the natural join needs to join all six matchings for both stars. Since each matching of S_1 should join with all matchings of S_2, nine join operations are needed.*

To overcome the first drawback, in Section 4, we propose the ***TOGGLE*** algorithm (***T***-closeness-***O***ptimized ***G***raph ***G***eneralization on ***L***abel ***E***xtension) for label generalization. It significantly reduces the search space for (sub)-optimal label generalization. For the second drawback, we present the ***PGP*** algorithm (***P****artial-****G****raph-based subgraph* ***P****rocessing*) in Section 5. It exploits the symmetry of the outsourced graph and eliminates the need for query decomposition.

4 TOGGLE for Label Generalization

As we discussed in Section 1, different outputs of label generalization have different impacts on the search space and the larger search space will cause expensive query cost. Hence, $TOGGLE$ aims to generalize labels into groups to minimize the search space, which refers to the total number of vertices to explore for a query. To this end, we first study how to estimate the search space of subgraph matching (Section 4.1). We then introduce the optimal $TOGGLE$ for label generalization that minimizes the search space by formu-

lating the minimization problem into a combinational optimization problem with constraints (i.e., *TOGGLE* problem, Section 4.2). We further present an approximate $TOGGLE$ algorithm with a bounded error in Section 4.3.

4.1 Estimating Search Space for Subgraph Matching

To estimate the search space for a query $\widetilde{Q}$ over an outsourced graph $\widetilde{G}^k$, we assume a general expansion-based graph search [28,3] as follows. The first vertex q of $\widetilde{Q}$ is selected based on degree and neighborhood signature. q is then matched on $\widetilde{G}^k$ with any vertex that contains the same vertex type and label group as q. After that, other vertices of $\widetilde{Q}$ are matched with neighbours of those already matched in $\widetilde{G}^k$. For the first vertex q, the number of vertices to explore can be estimated by the number of vertices in $\widetilde{G}^k$ multiplied by the probability of these vertices being the same types and having the same labels as q. For other vertices of $\widetilde{Q}$, the number of vertices to explore in $\widetilde{G}^k$ are limited to neighbours of those already matched in $\widetilde{G}^k$. In the end, the search space of $\widetilde{Q}$ over $\widetilde{G}^k$ is the product of all these numbers. Formally, for the τ-th vertex type, let n_τ, m_τ and $p_{r_\tau(j-1)+i}$ denote the number of labels, the number of label groups, and the i-th position in the j-th label group, respectively. $r_\tau = n_\tau/m_\tau$. We also use $F_G(\tau)$ to denote the probability of vertices in a graph G (e.g., G^k, $\widetilde{Q}$) being the type τ. Similarly, $F^l_G(\tau,i)$ and $F^g_G(\tau,j)$ denote the probability of the i-th label and the j-th label group (after the label generalization) being this type, respectively.

For first vertex q, its number of matchings can be estimated by the number of vertices in $\widetilde{G}^k$ (approximately $\frac{|V(G^k)|}{k}$) multiplies the matching probability of q over $\widetilde{G}^k$. Formally,

$$\frac{|V(G^k)|}{k}\sum_{\tau=1}^{\mathbb{T}}\Big[F_{G^k}(\tau)F_{\widetilde{Q}}(\tau)\sum_{j=1}^{m_\tau}F^g_{G^k}(\tau,j)F^g_{\widetilde{Q}}(\tau,j)\Big] \quad (1)$$

where $\mathbb{T}$ is the number of vertex type.

For other vertices, their matches are restricted to neighbours of those already matched in $\widetilde{G}^k$. Let $|\widetilde{Q}|$ denote the number of vertices in $\widetilde{Q}$, and the average degree $D(G^k)$ approximately represent the number of neighbours of the vertex in G^k, the number of matches for the other $|\widetilde{Q}|-1$ vertices can be estimated as follows.

$$\Big\{D(G^k)\sum_{\tau=1}^{\mathbb{T}}\Big[F_{G^k}(\tau)F_{\widetilde{Q}}(\tau)\sum_{j=1}^{m_\tau}F^g_{G^k}(\tau,j)F^g_{\widetilde{Q}}(\tau,j)\Big]\Big\}^{|\widetilde{Q}|-1} \quad (2)$$

Therefore, the search space of subgraph query $\widetilde{Q}$ is the product of Expressions (1) and (2), which directly correlates with $F_{G^k}(\tau)F_{\widetilde{Q}}(\tau)\sum_{j=1}^{m_\tau}F^g_{G^k}(\tau,j)F^g_{\widetilde{Q}}(\tau,j)$. Notably, each vertex in G^k has the union of label groups of all its $(k-1)$ symmetric vertices, which implies that $F^g_{G^k}(\tau,j)$ will increase by a factor of no more than $k-1$. In other words, assuming that the j-th label group of the τ-th vertex type contains r_τ labels, $\{l_{\tau,p_{r_\tau(j-1)+i}}|i\in[1,r_\tau]\}$, where $l_{\tau,p_{r_\tau(j-1)+i}}$ is the label locating at position i of this label group, $F^g_{G^k}(\tau,j)\le k\sum_{i=1}^{r_\tau}F^l_G(\tau,p_{r_\tau(j-1)+i})$ is therefore obtained. Additionally, $F^g_{\widetilde{Q}}(\tau,j)=\sum_{i=1}^{r_\tau}F^l_Q(\tau,p_{r_\tau(j-1)+i})$ can be easily derived. $F_{G^k}(\tau)F_{\widetilde{Q}}(\tau)\sum_{j=1}^{m_\tau}F^g_{G^k}(\tau,j)F^g_{\widetilde{Q}}(\tau,j)$ is therefore bounded by the product of $kF_{G^k}(\tau)F_{\widetilde{Q}}(\tau)$ and

$$\sum_{j=1}^{m_\tau}\Big[\sum_{i=1}^{r_\tau}F^l_G(\tau,p_{r_\tau(j-1)+i})\Big]\Big[\sum_{i=1}^{r_\tau}F^l_Q(\tau,p_{r_\tau(j-1)+i})\Big]. \quad (3)$$

Given a vertex type τ, $kF_{G^k}(\tau)F_{\widetilde{Q}}(\tau)$ is a constant.

4.2 Optimal TOGGLE for Label Generalization

Given the vertex labels $l=(l_{\tau,1},l_{\tau,2},...,l_{\tau,n_\tau})$ for the τ-th vertex type, label generalization combines them into m_τ groups, each with r_τ labels. It is equivalent to finding a permutation $P=(l_{\tau,p_1},l_{\tau,p_2},...,l_{\tau,p_{n_\tau}})$ of l and then sequentially dividing the permutated labels into m_τ groups, each satisfying t-closeness. Formally,

$$EMD_j(l,\{l_{\tau,p_{r_\tau(j-1)+i}}|i\in[1,r_\tau]\})\le t,\forall j\in[1,m_\tau].$$

Meanwhile, we want *TOGGLE* to minimize Expression (3) so that this label generalization can lead to a minimum search space. Combining the above, *TOGGLE* can be formulated as a *combinational optimization problem with constraints*.

TOGGLE $\quad \underset{P}{\operatorname{argmin}}\sum_{j=1}^{m_\tau}\Big[\sum_{i=1}^{r_\tau}F^l_G(\tau,p_{r_\tau(j-1)+i})\Big]^2$

subjects to

$$EMD_j(l,\{l_{\tau,p_{r_\tau(j-1)+i}}|i\in[1,r_\tau]\})\le t,j\in[1,m_\tau].$$

Note that the objective function as above is a simplified version of Expression (3) by assuming query graphs and data graphs are independent and identically distributed. And the vertex type τ is omitted whenever it is fixed.

This optimization problem is challenging as there are a huge number of permutations, and in each permutation the Earth's Mover Distance of all m_τ label groups need to be calculated. To address this, we reduce our optimization problem to the *General Set Partitioning Problem (GSPP)* [18]. Given a universe of n elements $(1,2,...,n)$, there is a rule $\mathbb{Y}$ that generates feasible subsets y_j of these elements, $\mathbb{J}=\{y_j\}$. Each subset y_j has a cost $c(y_j)$, and GSPP divides all elements into subsets with the minimum cost. Formally,

Definition 6 ***(General Set Partitioning Problem)*** *Let* $\lambda_j \in \{0,1\}$ *denote whether subset* y_j *is a result subset.* $y_{i,j}$*=1 if* y_j *contains element* i*, and 0 otherwise. Then GSPP finds* $\operatorname{argmin} \sum_{y_j \in \mathbb{J}} c(y_j)\lambda_j$ *subjects to* $\sum_{y_j \in \mathbb{J}} y_{i,j}\lambda_j = 1, i = 1, 2, ..., n$.

In $TOGGLE$, the universe has n labels denoted as l. Each permutation P partitions the labels into label groups, each has r=n/m labels. The j-th label group $\{l_{p_{r(j-1)+i}} | i \in [1, r]\}$ is a subset y_j. The rule $\mathbb{Y}$ is t-closeness, $EMD_j(l, y_j) \leq t$. The cost is the search space, i.e., $c(y_j) = (\sum_{i=1}^{n} F_G^l(y_{i,j}))^2$, where $F_G^l(y_{i,j})$ is the probability of label i in y_j. Specifically, if $y_{i,j} = 1$, $F_G^l(y_{i,j}) = F_G^l(i)$, otherwise $F_G^l(y_{i,j}) = 0$.

Therefore, the optimization problem of *TOGGLE* can be reduced to *GSPP* as follows:

$$\underline{\textbf{TOGGLE } \textit{under} \textbf{ GSPP}} \quad \min \sum_{y_j \in \mathbb{J}} \lambda_j \underbrace{c(y_j)}_{(\sum_{i=1}^{n} F_G^l(y_{i,j}))^2} \tag{4}$$

subjects to $\sum_{y_j \in \mathbb{J}} y_{i,j}\lambda_j = 1, i = 1, 2, ..., n$, and $\lambda_j \in \{0,1\}$. The original $TOGGLE$ problem has $\frac{n!}{((n/m)!)^m}$ permutations, wherein the $GSPP$ reduction effectively shrinks the size $|\mathbb{J}|$ to $\frac{n!}{(n/m)!(n-n/m)!}$.

4.3 Sub-Optimal TOGGLE for Label Generalization

Since the asymptotic size of $GSPP$ is still exponential in terms of n, finding the optimal partition is only feasible for small n.[5] In what follows, we propose a sub-optimal solution with a theoretical guarantee. In particular, we first transform the original $GSPP$ optimization problem in Expression (4) into Linear Programming Master (LPM) problem by relaxing the integer constraint. Then, the initial solution for LPM problem is obtained, based on which an iterative process consisting of a master problem and sub problem is presented. The process terminates until the desirable solution is derived.

Algorithm 2 outlines the sub-optimal solution. Inspired by *Column Generation* [18], the algorithm first transforms the original $GSPP$ optimization problem, denoted by **OP**, into **LPM** problem (Line 1) by relaxing the integer constraint $\lambda_j \in \{0,1\}$ to $\lambda_j \in [0,1]$.

$$\underline{\textbf{LPM Problem}} \qquad \min \sum_{y_j \in \mathbb{J}} c_j \lambda_j$$

subjects to $\sum_{y_j \in \mathbb{J}} y_{i,j}\lambda_j = 1, i = 1, ..., n$ and $\lambda_j \in [0,1]$, where $c_j = c(y_j) = (\sum_{i=1}^{n} F_G^l(y_{i,j}))^2$ and y_j is a subset and is also called a *column*.

[5] When n is small, we can enumerate all feasible subsets and relax the constraint $y_{i,j} \in \{0,1\}$ to $y_{i,j} \in [0,1]$. Then we apply the *Simplex* method to solve this linear programming problem. Finally, we employ the *Branch-and-Bound* method to obtain the integer solution [29].

Algorithm 2 *Sub-Optimal TOGGLE for Label Generalization*

Input: Original optimization problem **(OP)**, labels l, number of labels n, group number m and threshold t
Output: A sub-optimal partition of labels

1: $LPM \leftarrow reform(OP, \emptyset)$;
2: **if** $t \geq \frac{m-1}{2(n-1)}$ **then**
3: **for** $j \in [m]$ **do**
4: $y'_j \leftarrow \{l_j, l_{j+m}, ..., l_{j+(n/m-1)m}\}$;
5: $y_j \leftarrow genCol(y'_j)$;
6: **else**
7: $\{y_j\} \leftarrow ModifiedBasSol(l, n, m, t)$;
8: $return\ \{y_j\}$;
9: $y_j^* \leftarrow \emptyset$;
10: $\delta \leftarrow +\infty$;
11: **while** $\delta \geq 1.0e-4$ **do**
12: $RLPM \leftarrow reform(LPM, \{y_j\} \cup y_j^*)$;
13: $\{\lambda_j\}, \mu \leftarrow Simplex(RLPM)$;
14: $QKP \leftarrow conSubP(\mu)$;
15: $y_j^*, \delta \leftarrow QKPSolver(QKP)$;
16: $\{y_j\} \leftarrow Branch(RLPM)$;
17: $return\ \{y_j\}$;

Since it is impossible to obtain all $|\mathbb{J}|$ feasible columns at once for LPM problem, the algorithm first generates $|\mathbb{J}'|$ ($\leq |\mathbb{J}|$) feasible columns as an initial solution (Lines 2 to 8). With these columns, a master problem is formulated and solved by the Simplex method (Lines 12 to 13). The optimal solution is then used to formulate a dual sub problem, generate a new feasible column y_j^* (Lines 14 to 15), and append it to the resulted columns (Line 12). The algorithm then iteratively solves the master problem and sub problem until there is no column with a desired reduced cost (Line 11). Finally, it applies the branch and bound algorithm [29] to obtain the integer solution to the partition of labels (Line 16).

In what follows, we elaborate them and prove the theoretical error bound of this approximation algorithm.

4.3.1 Initial Solution (Lines 2 to 8)

Let $l = (l_1, l_2, ..., l_n)$ be n labels sorted by their values and $(1/n, 1/n, ..., 1/n)$ be their distribution masses. Each feasible column y_j should contain n/m labels $(e_1, ..., e_\alpha, ..., e_{n/m})$ with evenly distributed mass $(m/n, m/n, ..., m/n)$. So it should generate m feasible columns "aligned" with l to calculate EMD. Here, "aligned" means e_α transports distribution mass to l_i. We further define an *Alignment Group* as a group of consecutive l_i in l. If both l and e_α are sorted by their values, Lemma 2 shows their alignment relationship.

From Lemma 2, we can derive that if we select an element e_α for y_j from one of the positions $[(\alpha-1)m+1, \alpha m]$

of l, the ground distance between e_α and alignment group α is at most $\frac{n^2-n(n/m)}{2(n/m)^2(n-1)}$. Based on this lemma, we propose the following heuristic to obtain the initial solution. When $t < \frac{m-1}{2(n-1)}$, we design a procedure $ModifiedBasSol$ to find the exact optimal partition that satisfies t-closeness by exhaustive search (Line 7). The procedure is similar to the baseline solution in Algorithm 1 but it uses Lemma 2 to calculate EMD. When $t \geq \frac{m-1}{2(n-1)}$, we group $\{l_j, l_{j+m}, ..., l_{j+(n/m-1)m}\}$ into a label group y'_j and further generate y_j based on y'_j by setting $y_{i,j} = 1$ if $l_i \in y'_j$, otherwise $y_{i,j} = 0$ (Lines 2 to 6).

4.3.2 Master Problem (Lines 12 to 13)

With the generated $|\mathbb{J}'|$ columns, we further reformulate the LPM problem to a Restricted Linear Programming Master (**RLPM**) problem and obtain an optimal solution to it and its dual problem (Lines 12 and 13).

RLPM Problem $\min \sum_{y_j \in \mathbb{J}'} c_j \lambda_j$

subjects to $\sum_{y_j \in \mathbb{J}'} y_{i,j}\lambda_j = 1, i = 1, ..., n$ and $\lambda_j \in [0, 1]$, where $c_j = c(y_j) = (\sum_{i=1}^{n} F_G^l(y_{i,j}))^2$ and y_j is a column generated from the rule $\mathbb{Y}$. Note that y_j is limited to $\mathbb{J}'$ instead of $\mathbb{J}$.

RLPM problem is a simple linear programming problem with only $|\mathbb{J}'|$ feasible columns, so it can be easily solved by employing the Simplex method (Line 13).

4.3.3 Sub Problem (Lines 14 to 15)

Let μ denote the dual optimal solution to the RLPM problem. Sub problem is to find a new feasible column y_j^* that minimizes the reduced cost $\{c(y_j) - \mu y_j\}$ and meets the rule $\mathbb{Y}$, or formally

$$y_j^* = \underset{y_j}{\operatorname{argmax}}\{\mu y_j - c(y_j)\}$$

subjects to $EMD(l, y_j) \leq t$, where $c(y_j) = (\sum_{i=1}^{n} F_G^l(y_{i,j}))^2$.

The sub problem is obviously NP-hard. We propose a sub-optimal solution by reducing the sub problem to a $0-1$ Quadratic Knapsack Problem (QKP) (Line 14):

0 – 1 QKP $\max\{\mu y_j - c(y_j)\}$

subjects to

QKP Constraints $\sum_{\beta=1}^{m} y_{(\alpha-1)m+\beta,j} = 1, \quad \alpha = 1, ..., \frac{n}{m}$

The $0-1$ QKP can be solved by CPLEX optimizer [30] as a quadratic programming problem (Line 15).

4.3.4 Analysis on $TOGGLE$

In this sub-section, we give a detailed theoretical analysis for correctness, performance guarantee and complexity.

Correctness Analysis: We can theoretically prove that the generalized labels generated by $TOGGLE$ satisfy t-closeness. To this end, we need to prove that any generalized label in the initial solution (i.e., y_j) and that in the sub problem (i.e., y_j^*) achieve t-closeness.

Theorem 2 *When $t \geq \frac{m-1}{2(n-1)}$, the column y_j or y_j^* generated as above satisfies t-closeness, where n and m are number of labels and label groups, respectively.*

To sketch this proof, we first introduce two lemmas. In particular, Lemma 1 paves the way for the proof of Lemma 2 and the latter proves the range of the ground distance. Their proofs are given in the Appendix A.

Lemma 1 *The α-th element e_α of y_j is aligned with α-th alignment group of l, and e_α needs to transport $1/n$ distribution mass to each element in alignment group α.*

Lemma 2 *The ground distance between e_α and the alignment group α (denoted by g_α) is*

$$Dist(e_\alpha, g_\alpha) = \frac{1}{n-1} \sum_{\beta=1}^{m} |i - (\alpha-1)m - \beta|.$$

For $\forall i \in [(\alpha-1)m+1, (\alpha-1)m+m]$, if m is odd, $Dist(e_\alpha, g_\alpha) \in [\frac{n^2 - \frac{n}{m}^2}{4(n-1)\frac{n}{m}^2}, \frac{n^2-\frac{n^2}{m}}{2(n-1)\frac{n}{m}^2}]$. Otherwise, $Dist(e_\alpha, g_\alpha) \in [\frac{n^2}{4(n-1)\frac{n}{m}^2}, \frac{n^2-\frac{n^2}{m}}{2(n-1)\frac{n}{m}^2}]$.

Performance Guarantee: We can also prove that the (sub-)optimal $TOGGLE$ have a theoretical guarantee. In particular, if $t < \frac{m-1}{2(n-1)}$, (sub-)optimal TOGGLE can find the exact solution. Otherwise, it can provide a $(1 + m/5)$-near optimal solution.

Theorem 3 *When $t \geq \frac{m-1}{2(n-1)}$, sub-optimal $TOGGLE$ provide a $(1+\epsilon)$-approximate solution to the original problem where ϵ is approximately $m/5$.*

To sketch this proof, we introduce two lemmas. The first gives an approximation bound to the initial solution, the second gives an approximation bound to the sub problem.

Lemma 3 *When $t \geq \frac{m-1}{2(n-1)}$, the initial solution has an approximation ratio $(1 + \frac{m^2-m+2}{(m+1)(ln(n)+0.5772+1/(2n))})$ to the optimal solution.*

Lemma 4 *When $t \geq \frac{m-1}{2(n-1)}$, the reduction to $0-1$ QKP provides a $\frac{4}{(ln(n)+0.5772+1/(2n))m}$-approximate solution to the original subproblem.*

Complexity Analysis: We can also prove that the space complexity is linear to number of labels n, and worst case time complexity is directly proportional to $\frac{n!}{(n/m)!(n-n/m)!}$.

In particular, the space complexity is $O(n)$. As for time complexity, if $t \geq \frac{m-1}{2(n-1)}$, let ω, t_1 and t_2 denote the number of iterations, the time cost for one Simplex invocation, and the time cost for one QKP solver, respectively. Then the total time complexity of $TOGGLE$ is $O(\omega(t_1 + t_2))$. Otherwise, the time complexity is $\frac{n!t_3}{(n/m)!(n-n/m)!}$, where t_3 is the time cost to calculate EMD using Lemma 2. Note that the time complexity of (sub-)optimal $TOGGLE$ is far less that of the baseline solution (i.e., $O(\frac{n!t_1}{((n/m)!)^m})$, see Section 3), which is exponential to n. Moreover, the value of $\frac{n!}{(n/m)!(n-n/m)!}$ is relatively small as n is the number of labels in a single attribute whose value is small in real-world dataset (see Section 6).

5 PGP Subgraph Matching Algorithm

Although subgraph matching has been intensively studied [31,32,1], the only algorithm that can work on an outsourced succinct graph $\widetilde{G}^k$ is the star-based algorithm [3] described in Section 3. However, this algorithm needs to decompose the original query into multiple sub-queries, which is significantly inefficient. In this section, we propose the partial-graph-based subgraph processing algorithm PGP that no longer needs query decomposition. In the outsourced graph $\widetilde{G}^k$, **boundary nodes** are those "1-hop neighboring nodes" and **interior nodes** are the others. The challenge in processing query on $\widetilde{G}^k$ is to retrieve those neighbors for boundary nodes whose neighbors are partially in the succinct graph. To address this problem, PGP exploits the symmetry of the outsourced graph to retrieve matchings for those nodes. Its pseudo-codes are shown in Algorithm 3. Note that T and T_p denote the final matched subgraphs and the partial embedding of a single complete matching, respectively.

In Algorithm 3, it first initializes T and T_p with empty sets, and marks all data nodes in $\widetilde{G}^k$ as *unvisited* (Line 2). Then all matching candidates are generated for each query node of $\widetilde{Q}$ by comparing their labels, degrees, and neighbours' labels with those of interior nodes in $\widetilde{G}^k$, and expanding them to G^k (Line 3). Specifically, for each query node u of $\widetilde{Q}$, its interior-node candidate v is selected if all three conditions are met: (1) v contains the same vertex type and label group as u, (2) u's degree is no more than that of v, and (3) labels of u's neighbours are subset of those of v's neighbours. Next, the algorithm expands the candidates to the other blocks of G^k and forms the candidate set. After that, it generates a sorted list L_q to store the access order for each query node of $\widetilde{Q}$ according to its selectivity, the ratio between the number of candidate nodes and its degree (Line 4). Finally, it triggers the sub-routine $PartialSQ$ to retrieve all matched subgraphs.

Algorithm 3 *PGP* Algorithm for Subgraph Matching

Input: Outsourced query $\widetilde{Q}$, outsourced data graph $\widetilde{G}^k$
Output: All matched subgraphs T

1: **procedure** $PGP(\widetilde{Q}, \widetilde{G}^k)$
2: Initialize T, T_p and $\widetilde{G}^k$;
3: Generate candidate vertices for vertices of $\widetilde{Q}$;
4: Generate an ordered list L_q and father nodes for vertices of $\widetilde{Q}$;
5: $PartialSQ(0, T_p)$;
6: **function** $PartialSQ(d, T_p)$
7: **if** $|T_p| == |\widetilde{Q}|$ **then**
8: add T_p to T;
9: **return** T;
10: Find current vertex u and its father node u_f;
11: $C_s \leftarrow refineV(u, u_f, T_p)$;
12: **for** candidate $v \in C_s$ **do**
13: **if** $derJoin(u, v, T_p)$ **then**
14: Add pair $< u, v >$ to T_p;
15: $PartialSQ(d+1, T_p)$
16: Remove pair $< u, v >$ from T_p;
17: **return** T_p;

This sub-routine first determines whether the partial embedding T_p is a complete matching by comparing the size of T_p with that of $\widetilde{Q}$. If so, T_p is inserted to T (Lines 7 to 8). Then it recursively processes the query (Lines 10 to 17) as follows. It finds the query node u in depth d from L_q with its farther node u_f, and refines the candidates for matching with u by calling subroutine $refineV$ (Lines 10 to 11). $refineV$ confines $u's$ candidate v to the neighbours of v_f which is already matched with u_f. Specifically, if v_f is an interior node, v is selected as the candidate vertex when v is one of neighbours of v_f. If v_f is a boundary node, there are three steps: (1) finding the symmetry vertex v'_f from interior nodes for v_f, (2) retrieving all neighbours of v'_f; (3) determining if v can be mapped with one of those neighbours by automorphic function (if so, v is a candidate vertex of u). Then for each candidate $v \in C_s$, it determines whether v can be matched with u in current partial embedding T_p by judging whether all paired vertices of u's neighbours in T_p are also the neighbours of v (Lines 12 to 13). If v can be matched, the paired u and v will be inserted into T_p (Line 14). Then $PartialSQ$ is recursively called to access the next vertex in $\widetilde{Q}$ (Line 15).

Utility Analysis. As for the utility, two metrics should be considered. One is recall, i.e., the fraction of correct results among all true results for the query. Fortunately, all matchings retrieved by PGP can cover all true matchings of Q over G, as (k,t)-privacy model does not drop any edge

Table 2 Datasets

Dataset	$\|V\|$	$\|E\|$	N_t	N_a	N_l
Web-NotreDame	$0.3M$	$1.1M$	1	1	200
DBpedia	$3.2M$	$8.6M$	86	101	6300
UK-2002	$18M$	$261M$	2500	2500	20000

or node. As such, the recall is always 100%. The other is precision, i.e., the fraction of correct results among the retrieved matchings. The precision of all matchings retrieved by PGP is the same as that of the baseline solution.

Complexity Analysis. The space complexity is $O(|T|)$ where $|T|$ is the number of matchings. The time complexity is $O(|C_s|^{|V(\widetilde{Q})|})$ where $V(\widetilde{Q})$ denotes the set of vertices of query $\widetilde{Q}$.

6 Experimental Results

6.1 Experimental Setup

We evaluate the performance of proposed $TOGGLE$ (denoted by TOG) and PGP algorithms against the baseline solution for label generalization (denoted by BAS) and star-based subgraph processing (denoted by SGP) described in Section 3 (codes provided by courtesy of the authors of [3]). To evaluate the privacy and utility, we further compare our methods with state-of-the-art techniques. All algorithms are written in C++ except for *TOGGLE*, which is implemented in Matlab. The client is a desktop computer with Intel(R) Core(TM) i5-6600 CPU machine and 16GB RAM running Windows 10 Pro. The cloud server is a Windows Server 2016 Datacenter with 4 CPU cores and 64GB RAM.

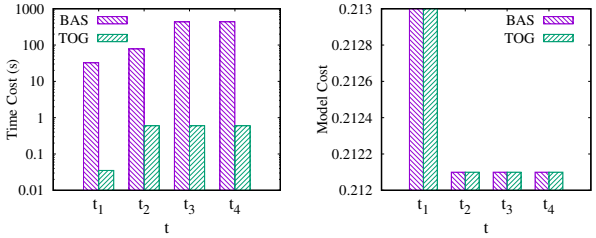
Datasets. We use three real graph datasets with increasing sizes: Web-NotreDame, DBpedia and UK-2002. Their statistics are given in Table 2 where N_t, N_a and N_l indicate number of types, attributes and labels, respecitvely. In particular, the maximum numbers of labels in a single attribute are 200, 985 and 1000, respectively. Query graphs are generated from the data graph by the random walk scheme [33]. For each dataset, 1000 query graphs with sizes in the range of [4–30] are randomly generated.

Performance Metrics. In each experiment, we measure the *Time Cost*, *Model Cost*, and *Approximation ratio*. *Time Cost* is the clock time to complete the experiment. *Model Cost* is the value of the objective function (Expression (4)) for a label generalization algorithm. *Approximation ratio* is the ratio of *Model Cost* of a sub-optimal solution to that of the optimal solution.

Parameter Settings. Unless otherwise stated, we use the default parameter values in Table 3. We choose 4 values $t_i, i \in \{1, 2, 3, 4\}$ for t in order to cover all possible cases of approximation ratio as stated in the proof of Theorem 2.

Table 3 Parameter Settings for Experiments

Parameter	Symbol	Default Value
#-automorphism	k	2
# of labels, groups, group size	n, m, $\frac{n}{m}$	12, $n/2$, 2
#-closeness	t_1	$0.9 * t_2$
	t_2	$\frac{m-1}{2*(n-1)}$
	t_3	$t_2 + 0.5(t_4 - t_2)$
	t_4	$\frac{2mn-2n-mm+m}{2(n-1)m}$



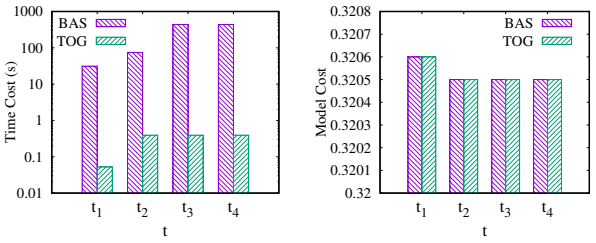
(a) Time Cost (b) Model Cost

Fig. 5 Time Cost and Model Cost on UK-2002(12).

6.2 Performance of Sub-Optimal TOGGLE

We first compare the performance of $TOGGLE$ with that of BAS under default settings. As we will see, since BAS is very time-consuming, we use only 12 labels in our default settings. Figure 5(a) shows the time cost on UK-2002 with 12 labels (denoted by UK-2002(12)). From this figure, we find that the time cost of TOG is far less than that of BAS regardless of t. In addition, we observe a dramatic increase of time cost of BAS as t increases from t_1 to t_4. This is because the number of label groups satisfying t-closeness increases when t is relaxed from t_1 to t_4. From Figure 5(b), we observe that the model cost of TOG is the same as that of BAS, which indicates that approximation ratio of TOG on UK-2002(12) can reach 1. Figure 6 shows similar results on the Dbpedia(12) dataset. This shows that $TOGGLE$ not only has a theoretical approximation bound, but is also close to the optimal in practice. In what follows, we further evaluate its performance by varying group size and number of labels.

Impacts of group size. Figure 7 shows the time cost on UK-2002(12) by varying the group size. In this figure, we use TOG_1 to denote the time cost of TOG when $t = t_1$; for $t = t_2$, t_3 or t_4, we use TOG as it acts the same. Similarly, BAS_i denotes the time cost of BAS when $t = t_i$. We observe that the time cost of TOG is far less than that of BAS especially for $t = t_3$ and t_4. When the group size increases, the time of BAS decreases because its search space is proportional to the number of different permutations. On the



(a) Time Cost (b) Model Cost

Fig. 6 Time Cost and Model Cost on Dbpedia(12).

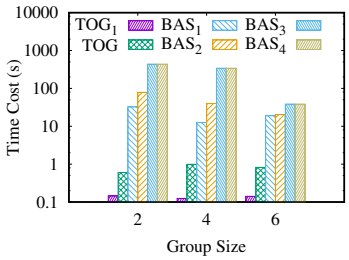


Fig. 7 Time Cost vs. Group Size on UK-2002(12).

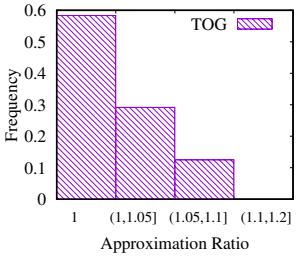


Fig. 8 $Freq$ on UK-2002(12) and Dbpedia(12).

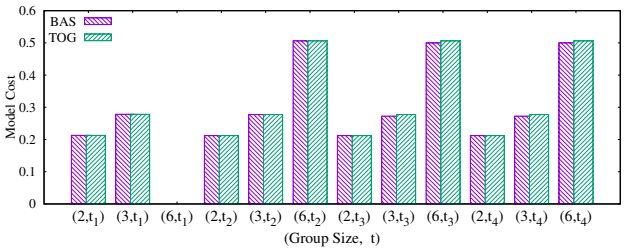


Fig. 9 Model Cost vs. Group Size on UK-2002(12).

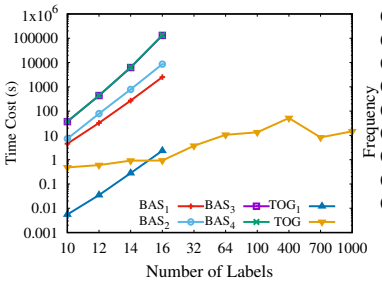


Fig. 10 Time Cost vs. Number of Labels on UK-2002.

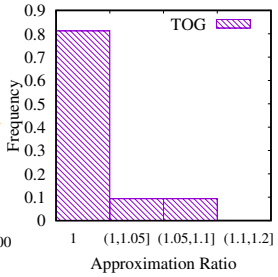


Fig. 11 $Freq$ on UK-2002 and Dbpedia.

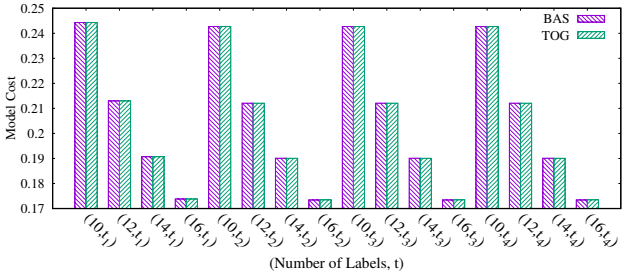


Fig. 12 Model Cost vs. Number of Labels on UK-2002.

other hand, TOG fluctuates within a narrow range within 1 second.

Figure 9 presents the corresponding model costs on UK-2002(12). There is one missing result at $(6, t_1)$ because there is no feasible label group in this setting. It can be seen from this figure that increasing group size will lead to the increase of model cost. Nonetheless, the model cost of TOG is almost identical to that of BAS_i.

We also plot the frequency distribution of approximation ratio (denoted by $Freq$) on various group sizes and t setting for both datasets in Figure 8. Overall even the worst case leads to an approximation ratio lower than 1.1. As for the best case, more than 50% of cases can achieve a ratio equal to 1.

Impacts of label size. We vary the number of labels from 10 to 1000 (i.e., the maximum number of labels in a single attribute) on UK-2002 and compare their results. As shown in Figure 10, the time cost of BAS is rising exponentially with respect to the number of labels, and it already becomes prohibitively costly (e.g., more than 1.5 days) for only 16 labels at $t \geq t_3$. On the other hand, TOG is less sensitive to the increase of number of labels. In fact, the time cost of TOG reaches its peak at 400 and slowly decreases afterwards, which indicates that TOG can easily scale to even more labels. Figure 12 presents the corresponding model costs on UK-2002. It can be seen from this figure that the model cost of TOG is almost equivalent to that of BAS. This shows TOG can achieve robust and desirable performance irrespective of label size.

As for the approximation ratio, we plot the frequency distribution of approximation ratios (denoted by $Freq$) of TOG for both UK-2002 and Dbpedia in Figure 11. We find the approximation ratios are no more than 1.1 and 80% of them are exactly 1.

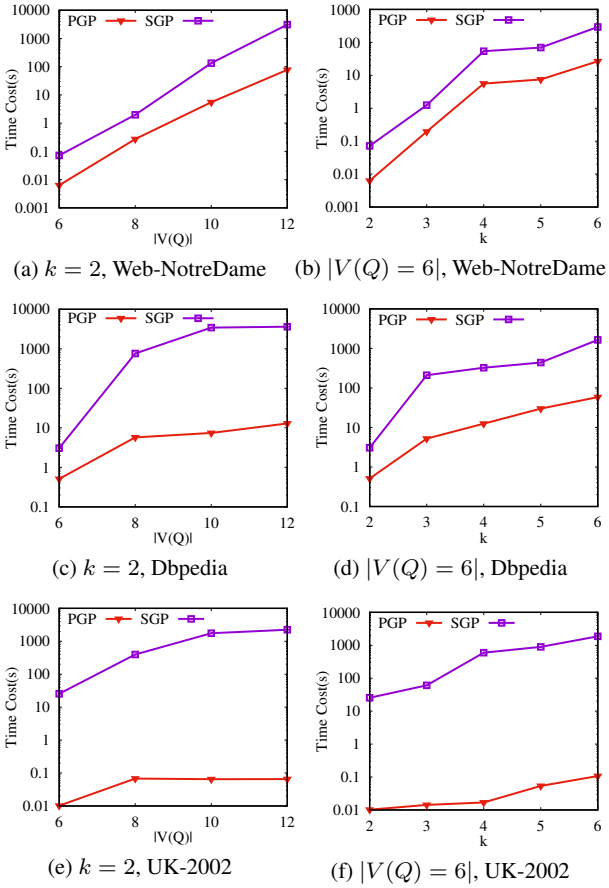
To sum up, our privacy-preserving label generalization algorithm $TOGGLE$ achieves high efficiency, effectiveness and scalability.

6.3 Performance of PGP Algorithm

We compare our PGP algorithm with SGP, the star-based subgraph processing algorithm. Figure 13(a) presents the time cost for the Web-NotreDame dataset. We observe that as the query size increases, the time cost of PGP grows much slower than that of SGP. This is because the number of star matchings undergoes a huge rise as shown in Table 4. We then fix the query size to 6 and vary the value of k from 2 to 6. As shown in Figure 13(b), the time cost increases with k, because a larger k introduces more redundant edges to the k-automorphic data graph, which leads to more matchings for a single query. Nonetheless, PGP takes

Table 4 # of Star Matchings on Web-NotreDame (k=2)

Query Size $\|V(Q)\|$	6	8	10	12
Number of Star Matchings	880	5280	24999	59967



(a) $k = 2$, Web-NotreDame (b) $|V(Q) = 6|$, Web-NotreDame

(c) $k = 2$, Dbpedia (d) $|V(Q) = 6|$, Dbpedia

(e) $k = 2$, UK-2002 (f) $|V(Q) = 6|$, UK-2002

Fig. 13 Time Cost vs. $|V(Q)|$ and Time Cost vs. k.

Table 5 Success ratio under similarity attack and majority attack

k	Similarity Attack		Majority Attack	
	BAS[3]	TOG	BAS[3]	TOG
2	53%	0%	1.1%	1.2%
4	48%	0%	1.3%	1.1%
6	52%	0%	0.9%	1.0%

less time cost than SGP, since the latter incurs significant time to compute natural join for a large number of matched stars. Figures 13(c) through (f) show similar results for the other two datasets. As such, we conclude that PGP always outperforms SGP under various k and query sizes. Furthermore, the gain of PGP becomes more evident for larger k or query sizes.

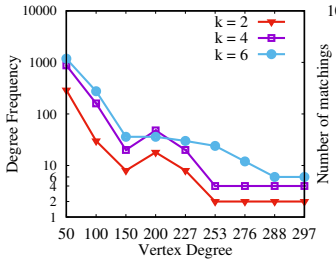


Fig. 14 Degree frequencies.

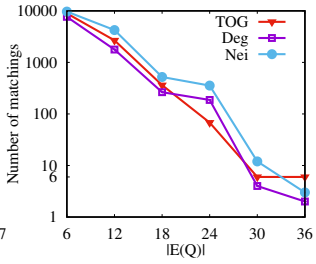


Fig. 15 Number of matchings (k=6).

6.4 Performance of Privacy-Utility

We further compare our work with some existing techniques in terms of privacy and utility on Web-NotreDame.

Power of Privacy Protection. We compare anonymized graphs produced by our work (TOG) and other classical anonymization methods [11, 13, 3] under different attacks such as degree attack, subgraph attack and similarity attack. For any vertex degree d of the anonymized graph produced by our work, we report its frequency of vertices with degree d. As depicted in Figure 14, the minimal degree frequency is k, which indicates that our method can guarantee privacy under degree attack. We further test our method under subgraph attacks. To this end, we randomly extract some subgraphs from the original graph as query graphs, and retrieve the matchings for each query to determine if the number of matching is at least k. As shown in Figure 15, the number of matchings for each subgraph query over the anonymized graph produced by our work is at least k. This indicates that our method can make the anonymized graph defend against subgraph attack. Similarly, we can also find that Deg [11] and Nei [13] suffer from subgraph attack. The reason is that these two algorithms only consider a single type of attack, i.e. degree-attack or 1-neighbor-graph attack, respectively. We further compare TOG with BAS [3] under similarity attack and majority attack (i.e., simply predict the target node's label based on the majority label of the neighbors). Table 5 presents the success ratios under similarity attack and majority attack (denoted by $SucRatio(SimAttack)$ and $SucRatio(MajAttack)$), which refer to the ratios of that one attacker can infer the sensitive label using the attack techniques. We find that the success ratio of the similarity attack on anonymized graph produced by TOG is 0, while that on anonymized graph produced by BAS is close to 50% . We can also find that $SucRatio(MajAttack)$ of both TOG and BAS are close to 1.0% , which equal to the probability that randomly selecting a label (from 100 labels) as the label of a target node. Therefore, we can conclude that TOG can guarantee privacy under degree attack, subgraph attack and similarity attack.

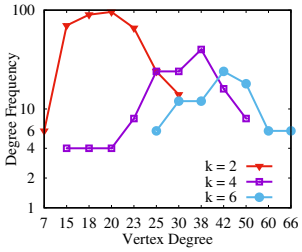


Fig. 16 Degree frequencies on $ERNet$.

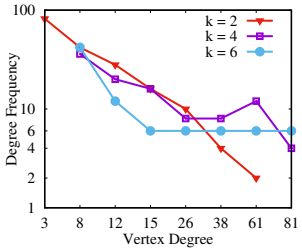


Fig. 17 Degree frequencies on $SFNet$.

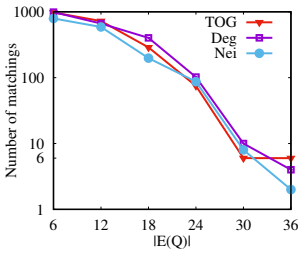


Fig. 18 Number of matchings on $ERNet$ (k=6).

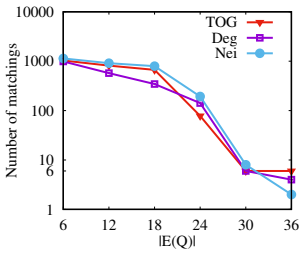


Fig. 19 Number of matchings on $SFNet$ (k=6).

Table 6 Success ratio under similarity attack and majority attack on $ERNet$

k	Similarity Attack		Majority Attack	
	BAS[3]	TOG	BAS[3]	TOG
2	49%	0%	1.1%	1.0%
4	51%	0%	1.0%	1.1%
6	50%	0%	1.1%	1.0%

In addition, we further evaluate our methods using synthetic data sets, and compare them with the existing state-of-the-art techniques. We use a software called $Pajek$ (http://vlado.fmf.unilj.si/pub/networks/pajek/) to generate two kinds of random graphs, Erdos Renyi Graph and Scale-Free Graph.

1. Erdos Renyi Graph (denoted by $ERNet$): This graph can be generated by a random graph model, which defines a random graph as N vertices connected by M edges that are chosen randomly from the $N(N-1)$ possible edges. $Pajek$ can generate it by setting the number of vertices N and average degree d. In our experiments, we set $N = 1000$ and $d = 10$.
2. Scale-Free Graph (denoted by $SFNet$): A scale-free network is a network whose degree distribution follows or asymptotically follows a power law. In our experiments, we set the number of vertices to be 1049.

As depicted in Figure 16 and 17, for any vertex degree d of the anonymized graph produced by our method, its frequency of vertices with degree d is at least k. It indicates that our method can defend against degree attack. We further test our method under subgraph attacks by retrieving the matchings for each query. As presented in Figure 18 and 19, the number of matchings for each subgraph query over the anonymized graph produced by our work is at least k. This indicates that our method can make the anonymized graph defend against subgraph attack. Similarly, we can also find that both Deg [11] and Nei [13] suffer from subgraph attack. We further compare TOG with BAS [3] under similarity attack and majority attack on our synthetic data set $ERNet$. Table 6 presents the success ratios under similarity attack and majority attack (denoted by $SucRatio(SimAttack)$ and $SucRatio(MajAttack)$). We find that the success ratio of the similarity attack on anonymized graph produced by TOG is 0, while that on anonymized graph produced by BAS is close to 50% . Observe that $SucRatio(MajAttack)$ of both TOG and BAS are close to 1.0% , which equal to the probability that randomly selecting a label (from 100 labels) as the label of a target node. Therefore, we can conclude that TOG can guarantee privacy under degree attack, subgraph attack and similarity attack.

Table 7 Running Time and Storage Space (PGP V.S. SQ)

k	Running Time (S)		Storage Space (MB)	
	PGP	SQ [33]	PGP	SQ [33]
2	0.01	0.12	19.18	37.80
3	0.19	0.48	19.56	54.92
4	5.60	6.19	20.00	72.41
5	7.47	12.36	21.25	91.22
6	26.69	65.57	21.82	107.95

Utility Evaluation. To evaluate the utility of our method (PGP), we further compare it with classical subgraph matching methods over the entire graph. Since both of $\widetilde{G}^k$ and Alignment Vertex Table (AVT) are outsourced to the cloud during pre-processing, the cloud actually knows all the information of the original $k-$automorphic graph G^k. Hence, most subgraph matching algorithms [25,34,33] can be extended to work on the outsourced graph $\widetilde{G}^k$ by following steps: 1) *Graph Extensions.* Given a vertex p of $\widetilde{G}^k$, its symmetric vertices P on other blocks can be found by retrieving from Alignment Vertex Table (AVT). Similarly, the symmetric vertices Q of another vertex q of $\widetilde{G}^k$ can be easily found. If there is an edge between p and q, an edge should be inserted between vertices pair of p' and q' where $p' \in P$ and $q' \in Q$. 2) *Index Construction.* Then the indices for degree and neighborhood signature filtering for candidates as in previous works such as [28] can be constructed. 3) *Subgraph Matching.* Given the extended graph of $\widetilde{G}^k$ and the indices, classical subgraph matching algorithms such as [25, 34,33] can be applied to retrieve subgraph matchings. The results of PGP and that of the classical subgraph match-

Table 8 Recall

Methods	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$
PGP	100%	100%	100%	100%	100%
SQ	100%	100%	100%	100%	100%
SGP	100%	100%	100%	100%	100%

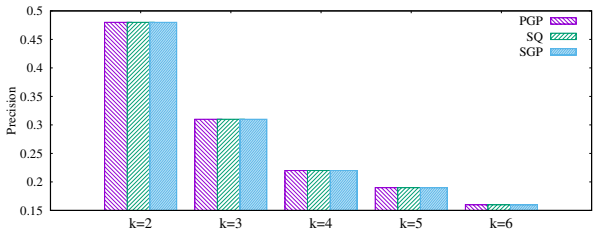


Fig. 20 Precision.

ing algorithm [33] (denoted by SQ) on Web-NotreDame are shown in Table 7. Under the same privacy (the same k), we can find that PGP outperforms SQ w.r.t both running time and storage space. In particular, the time and space savings are up to 10 and 5 folds, respectively.

We further evaluate the recall and precision of PGP, SQ, and SGP. Recall that recall is the fraction of correct results among all true results for the query and precision is the fraction of correct results among the retrieved matchings. As reported in Table 8, recall are always 100% since none of edges and nodes are dropped in the generalization process. As for precision, these methods obtain the same performance due to that all of them can be deemed as subgraph processing on a $k-$automorphic graph G^k (Figure 20). In addition, the precision decreases as the value of k increases, since more dummy edges will be introduced as the value of k increases. The observations further justify the utility analysis in Section 5 (Utility Analysis). In general, more stringent privacy will be obtained as the value of k increases. Meanwhile, more running time and storage space are taken for subgraph processing, and worse precision is obtained.

7 Related Work

The most germane to this research is privacy-preserving graph data publication and anonymization, and privacy-preserving graph query processing in the cloud, which is summarized in Table 9.

Privacy preserving graph data publication and anony mization. Many structural privacy-preserving mechanisms have been developed [15–17] by exploiting the symmetry of the graph data to guard against various attacks such as degree attack, subgraph attack and hub-fingerprint attack [13, 11, 14]. In particular, the *k*-automorphism method by Zou *et al.* constructs $(k-1)$ symmetric vertices for each existing vertex and is claimed to defend against any graph structural attacks [15]. When sensitive labels are considered, the k-anonymity and ℓ-diversity have been adopted for graphs [10, 3]. In addition, they can also preserve the structure privacy. However, [10] can only defend against some specific structural attack (e.g., degree attack) or label attack on simple labels. They are both vulnerable to advanced attacks, such as similarity attack. Differential privacy (DP) [9] as a more stringent privacy model has been widely studied to protect against the privacy disclosure of statistical databases. It ensures that query results on a dataset are insensitive to the change of a single record. Owing to its unique strengths, it has been applied to graph data analysis, which can be grouped into two categories: edge-DP [45–48] and node-DP [51–53, 50] based methods. In edge-DP (resp. node-DP), two graphs are neighboring if they differ on a single edge (resp. node). In particular, [45] publishes degree sequence of a graph under edge-DP by adopting the Laplace mechanism where sensitive is 2. [46, 47] have also considered publishing the degree sequence or distribution by extending the technique to synthetic generation. [48] proposes a framework for graph metric estimation with local differential privacy. When the setting moves to node-DP, the sensitivity of degree distribution becomes $2(|V|-1)$ since removing a node may have effect on other $|V|-1$ nodes. Hence, [53] explores the projection approach to reduce the sensitivity when publishing the degree distribution of a graph under node-DP. While [52] adopts a truncation approach to reduce the sensitivity. Instead of publishing degree sequence or distribution, differentially private subgraph counting queries under node-DP have been studied [51, 53]. For example, [53] develops a novel method for differentially private triangle counting in large graphs. Recent efforts have investigated the problem of publishing statistics of attributed graphs or spatio-temporal graphs [35, 49, 54]. The neighboring data in [35] is the records that differ in the presence of a single edge or in the attribute vector associated with a single node. In contrast, neighboring data in [54] is defined on infinite series data. As the perturbation is required to satisfy differential privacy, noises such as Laplace noise [54, 45] needs to be injected into graphs or their statistics. This makes DP-based techniques insufficient or even infeasible in subgraph matching where *exact* matchings are desirable.

There are a lot of other graph data anonymization and publication techniques [36–38, 14]. For example, to protect a graph from link re-identification, Zheleva & Getoor propose five different privacy preservation strategies, which vary in terms of the amount of data removed and the amount of privacy preserved [36]. While Campan & Truta tries to mask the graph data according to the k-anonymity model, in terms of both nodes' attributes and nodes' associated structural information [37]. However, they can not apply to our case since 1) some of them can not protect both structural and

Table 9 Summary on Related Work

Topic	Ref	Focuses	Cons
Privacy preserving graph data publication and anonymization	[15–17]	Exploiting the symmetry of the graph data to guard against various attacks such as degree attack, subgraph attack and hub-fingerprint attack [13, 11, 14].	Can not protect label privacy since only structure privacy is considered.
	[10, 3]	Adopting the k-anonymity and ℓ-diversity for labels of graphs. In addition, they can also preserve the structure privacy.	Can only defend against some specific structural attack (e.g., degree attack) or label attack on simple labels.
	[45–54, 35]	Utilizing differential privacy for publishing graph statistics without disclosing graph privacy.	Insufficient or even infeasible in subgraph matching where *exact* matchings are desirable since data perturbation is required.
	[36–38, 14]	Some other techniques on graph data anonymization and publication.	Some of them can not protect both structural and label privacy, or need to delete edges from the original data graph.
Privacy preserving graph query processing in cloud	[39]	An asymmetric structure-preserving subgraph query processing method which is the first practical private approach for subgraph query services.	The data graph is publicly known.
	[42, 40, 56, 57]	Privacy-preserving schemes for (approximate) shortest path queries or top-k critical vertices query on shortest path.	Can not process subgraph matching studied in this paper.
	[43]	Exploring the problem of processing subgraph matching over a set of encrypted small graphs in cloud.	Support only subgraph matching over a set of small graphs instead of a single large attributed graph.
	[58]	A novel k-decomposition algorithm and a new information loss matrix designed for utility measurement in massively large graph datasets.	Can not protect label privacy since only structure privacy is considered.
	[3, 59]	Privacy preserving subgraph matching on large attributed graphs in cloud.	Vulnerable to similarity attack and suffering from low efficiency.
	[60, 61]	Techniques on other privacy-preserving graph queries such as reachability query and kNN query.	Can not be adapted to subgraph matching on a single large attributed graph.

label privacy (e.g., [36] can only protect the graph from link re-identification). 2) some of them need to delete edges from the original data graph [36, 37]. Hence, they are infeasible in subgraph matching where *exact* matchings are desirable.

Privacy preserving graph query processing in cloud. There are a lot of privacy-preserving methods or frameworks for diverse queries. In particular, Fan et al. [39] proposes an asymmetric structure-preserving subgraph query processing method which is the first practical private approach for subgraph query services, where the data graph is publicly known and the query structure/topology is kept secret. [40] develops specific schemes for shortest path queries, which achieve much better performance and scalability with a strong privacy property in practical scenarios. [42] proposes a method to efficiently compute the shortest distance in large outsourced graphs without compromising their sensitive information. Ma et al. [56] investigate the shortest path sketch by defining a top-k critical vertices query on the shortest path. A novel graph encryption scheme that enables approximate constrained shortest distance querying is studied in [57]. However, all of them are privacy-preserving schemes for (approximate) shortest path queries or top-k critical vertices query on shortest path, which do not apply to our case where the goal is to find subgraph matches on a single large attributed graph. [43] presents a method to answer the subgraph matching over a set of encrypted small graphs instead of a single large graph. [58] develops a novel k-decomposition algorithm and a new information loss matrix designed for utility measurement in massively large graph datasets. However, it cannot protect label privacy since only structure privacy is considered. The most germane to this research is [3, 59] which develop privacy-preserving subgraph matching methods on large attributed graphs in cloud. Unfortunately, they are vulnerable to similarity attack and suffer from low efficiency. Other techniques on other privacy-preserving graph queries such as reachability query [60] and kNN query [61]. Nevertheless, they cannot be adapted to subgraph matching on a single large attributed graph.

8 Conclusion

In this paper, we propose a graph label generalization algorithm and an efficient subgraph matching algorithm in cloud with t-closeness and k-automorphism privacy. The former achieves a theoretically guaranteed approximation ratio of $(1+\epsilon)$ where ϵ is approximately 0.2 times the number of label groups. The latter exploits the symmetry of the generalized graph and limits the search scope to a localized region. As for future work, we plan to extend this solution framework to a wide range of classic graph queries, such as maximal clique search and best region search.

9 Acknowledgment

This work was supported by National Natural Science Foundation of China (Grant No: U1636205, 61572413), the Research Grants Council, Hong Kong SAR, China (Grant No: 15238116, 15222118, 15218919 and C1008-16G), and the Australian Research Council (Grants No. DP200103650).

Appendix A Proofs

In this section, we present the formal proofs of theorems and lemmas.

A.1 Proof of Theorem 1

Given a data graph G, the graph outsourcing problem with t-closeness and k-automorphism is to compute an outsourced graph G', where t-closeness for its labels is required. Since t-closeness is a known NP-Hard problem [55] and can be reduced to our graph outsourcing problem, the graph outsourcing problem with t-closeness and k-automorphism is NP-Hard. In addition, our subgraph matching problem is NP-Hard since it involves subgraph isomorphism testing, which is a classical NP-Hard problem [25–27]. Overall, both graph outsourcing problem with (k,t)-privacy and subgraph matching problem on outsourced graphs are NP-Hard.

A.2 Proof of Lemma 1

Let $l=(l_1,l_2,...,l_n)$ be ordered labels and $(1/n,1/n,...,1/n)$ their distribution masses. We define α-th *Alignment Group* (denoted by g_α) as m consecutive labels in l, i.e., $g_\alpha = (l_{(\alpha-1)m+1},\ l_{(\alpha-1)m+2},\ ...,l_{(\alpha-1)m+\beta},...,l_{\alpha m})$(Figure 21). In addition, let feasible column y_j be ordered labels $(e_1,...,e_\alpha,...,e_{n/m})$ with evenly distributed mass $(m/n,m/n,...,m/n)$. Since labels are ordered, according to [8], the minimal workload of $EMD(l,y_j)$ can be achieved by satisfying all elements of l sequentially, i.e., sequentially move distribution masses from y_j to l. In particular, as depicted in Figure 21, e_α should transport $\frac{1}{n}$ distribution mass to each label in $g_\alpha = (l_{(\alpha-1)m+1},\ l_{(\alpha-1)m+2},\ ...,l_{(\alpha-1)m+\beta},...,l_{\alpha m})$. In short, each element e_α in y_j should be "aligned" with the α-th alignment group (i.e., transport distribution mass to elements in α-th alignment group), and e_α should transport $\frac{1}{n}$ distribution mass to each element of α-th alignment group.

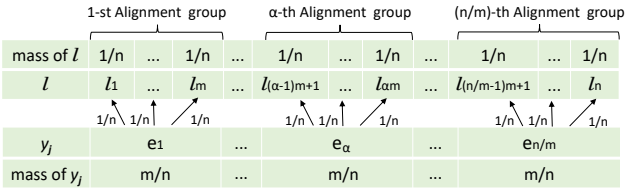


Fig. 21 Alignment Group

A.3 Proof of Lemma 2

According to Lemma 1, each element e_α in y_j is aligned with α-th alignment group (i.e., g_α). In addition, observe that the subscripts of elements in alignment group α are $(\alpha-1)m+1,(\alpha-1)m+2,...,(\alpha-1)m+\beta,...,(\alpha-1)m+m$, respectively, and the ground distance between e_α and β-th element in alignment group α is $\frac{|i-(\alpha-1)m-\beta|}{n-1}$ where i is the position of e_α in l. Therefore, we derive that the ground distance between e_α and α-th alignment group is

$$\frac{1}{n-1}\sum_{\beta=1}^{m}|i-(\alpha-1)m-\beta|$$

where i is $e'_\alpha s$ position in l. To estimate its domain, three cases should be considered:

(1) If $i \le (\alpha-1)m+1$,

$$\begin{aligned} Dist(e_\alpha,g_\alpha) &= \frac{1}{n-1}((\alpha-1)m^2+\frac{(1+m)m}{2}-im) \\ &= \frac{2n^2\alpha-2(n^2/m)i-2n^2+(n/m+n)n}{2(n-1)(n/m)^2}. \end{aligned}$$

$Dist(e_\alpha,g_\alpha) \in [\frac{n^2-n^2/m}{2(n-1)(n/m)^2}, \frac{2n^3/m-2n^3/m^2-n^2+n^2/m}{2(n-1)(n/m)^2}]$.

(2) If $(\alpha-1)m+1 \le i \le (\alpha-1)m+m$,

$$\begin{aligned} Dist(e_\alpha,g_\alpha) &= \frac{1}{n-1}(\sum_{\beta_1=1}^{\beta}(\beta-\beta_1)+\sum_{\beta_2=1}^{m-\beta}\beta_2) \\ &= \frac{2\beta^2\alpha-2(1+m)\beta+m+m^2}{2(n-1)}. \end{aligned}$$

If m is odd, $Dist(e_\alpha,g_\alpha) \in [\frac{n^2-n^2/m^2}{4(n-1)(n/m)^2}, \frac{n^2-n^2/m}{2(n-1)(n/m)^2}]$, otherwise, $[\frac{n^2}{4(n-1)(n/m)^2}, \frac{n^2-n^2/m}{2(n-1)(n/m)^2}]$.

(3) If $i \geq (\alpha-1)m+m$,

$$Dist(e_\alpha, g_\alpha) = \frac{1}{n-1}(im-(\alpha-1)m^2-\frac{(1+m)m}{2}) = \frac{\frac{2n^2}{mi}-2(\frac{n}{m})^2 n-2n^2\alpha+2n^2-(\frac{n}{m}+n)n}{2(n-1)(n/m)^2},$$

$Dist(e_\alpha, g_\alpha) \in [\frac{n^2-n^2/m}{2(n-1)(n/m)^2}, \frac{2n^3/m-2n^3/m^2-n^2+n^2/m}{2(n-1)(n/m)^2}]$.

Therefore, for $\forall i \in [(\alpha-1)m+1, (\alpha-1)m+m]$, $Dist(e_\alpha, g_\alpha) \in [\frac{n^2-\frac{n}{m}^2}{4(n-1)\frac{n}{m}^2}, \frac{n^2-\frac{n^2}{m}}{2(n-1)\frac{n}{m}^2}]$ (if m is odd) or $Dist(e_\alpha, g_\alpha) \in [\frac{n^2}{4(n-1)\frac{n}{m}^2}, \frac{n^2-\frac{n^2}{m}}{2(n-1)\frac{n}{m}^2}]$ (if m is even).

A.4 Proof of Theorem 2

Lemma 2 proved that $\forall i \in [(\alpha-1)m+1, (\alpha-1)m+m]$, if m is odd, $Dist(e_\alpha, g_\alpha) \in [\frac{n^2-\frac{n}{m}^2}{4(n-1)\frac{n}{m}^2}, \frac{n^2-\frac{n^2}{m}}{2(n-1)\frac{n}{m}^2}]$. Otherwise, $Dist(e_\alpha, g_\alpha) \in [\frac{n^2}{4(n-1)\frac{n}{m}^2}, \frac{n^2-\frac{n^2}{m}}{2(n-1)\frac{n}{m}^2}]$. Each element e_α of y_j generated by initial solution is selected from the i-th position of l, where $i \in [(\alpha-1)m+1, (\alpha-1)m+m]$. In addition, Lemma 1 shown that each element e_α in y_j is supposed to transport $1/n$ distribution mass to each element in $\alpha-$th alignment group. Based on those two observations, we derive that $EMD(l, y_j) \leq \sum_{\alpha=1}^{n/m} \frac{n^2-n^2/m}{2(n-1)(n/m)^2} \times \frac{1}{n} = \frac{mn^2-n^2}{2(n-1)n^2} = \frac{m-1}{2(n-1)}$. Therefore, when $t \geq \frac{m-1}{2(n-1)}$, the column y_j satisfies $t-$closeness. Similarly, each column generated in sub problem also satisfies $t-$closeness. By the way, we can adopt the similar way to prove that the EMD between l and any other column is bounded by $\frac{2mn-2n-m^2+m}{2(n-1)m}$.

A.5 Proof of Lemma 3

Let l=$(l_1, l_2, ..., l_n)$ be n labels ordered by their values, and a the *Euler-Mascheroni* constant ($\approx \frac{1}{ln(n)+0.5772+1/2n}$), the frequency of l can be represented by $(\frac{a}{1}, \frac{a}{2}, ..., \frac{a}{n})$, since the frequencies of labels roughly obey the Zipf's law [3]. When $t \geq \frac{m-1}{2(n-1)}$, sub-optimal $TOGGLE$ generates the initial partition $\{y_j | j \in [1, m]\}$ where $y_{i,j} = 1$ if i locates in $\{j, j+m, ..., j+(n/m-1)m\}$ or $y_{i,j} = 0$, otherwise. Let the sum of label frequencies of y_j be

$$s_j = \frac{a}{j} + \frac{a}{j+m} + ... + \frac{a}{j+(n/m-1)m},$$

the cost of y_j is obviously s_j^2. Therefore, the total cost of the initial solution is $s_1^2+s_2^2+...+s_m^2$ where $s_1+s_2+...+s_m = 1$ and $s_1 > s_2 > ... > s_m$. Due to

$$\begin{aligned} s_1 &= \frac{a}{1} + \frac{a}{1+m} + ... + \frac{a}{1+(n/m-1)m} \\ &\leq \frac{a}{1} + \frac{a}{m} + ... + \frac{a}{(n/m-1)m} \\ &\leq \frac{1}{m} + \frac{(m^2-m+2)a}{m^2+m}, \end{aligned}$$

we can derive that $\frac{1}{m} + \frac{(m^2-m+2)a}{m^2+m} \geq s_1 > s_2 > ... > s_m$, and

$$\begin{aligned} \sum_{i=1}^m s_i^2 &= (\sum_i^m s_i)^2 - s_1(\sum_{i\neq 1}^m s_i) - s_2(\sum_{i\neq 2}^m s_i) - ... - s_m(\sum_{i\neq m}^m s_i) \\ &\leq \frac{1}{m} + \frac{(m^2-m+2)a}{m^2+m}. \end{aligned}$$

Therefore, the model cost of initial solution under $t-$closeness constraints is at most $\frac{1}{m} + \frac{(m^2-m+2)a}{m^2+m}$. To estimate the approximate ratio R_1 of our model cost to the exact model cost, we first relax the $t-$closeness constraint to find the minimum model cost. Formally, for any $\{X_i\}$ subjecting to $\sum_{i=1}^m X_i = 1$, we need to estimate the lower bound of $\sum_{i=1}^m X_i^2$. According to $Cauchy$-$Schwarz$ inequality, for $X_i, Y_i \in \mathcal{R}$, $(\sum_{i=1}^m X_i Y_i)^2 \leq (\sum_{i=1}^m X_i^2)(\sum_{i=1}^m Y_i^2)$. Let $Y_i = 1$, we derive that $\frac{1}{m} \leq \sum_{i=1}^m X_i^2$. Therefore, the minimum model cost, $s_1^2 + s_2^2 + ... + s_m^2$, is no less than $\frac{1}{m}$. The approximate ratio $R_1 \leq \frac{1/m+(m^2-m+2)a/(m^2+m)}{1/m} \leq 1 + \frac{(m^2-m+2)a}{m+1}$. The approximation is good since the approximate ratio is approximately liner to $m \cdot a$.

A.6 Proof of Lemma 4

From Lemma 3, we observe that the sum of labels frequencies of y_j is $s_j = \frac{a}{j} + \frac{a}{j+m} + ... + \frac{a}{j+(n/m-1)m}$ where $s_1 + s_2 + ... + s_m = 1$ and $s_1 > s_2 > ... > s_m$. If we denote the first dual solution of the master problem as $\mu = [s_1^2, s_2^2, ..., s_m^2, 0, 0, ..., 0]$, the objective values of the original sub problem and the reduced problem can be formulated as $J_2 = min(c(y_j) - \mu y_j)$, *s.t.*, $EMD(l, y_j) \leq t$ and $J_2' = min(c(y_j) - \mu y_j)$, *s.t., QKP Constraints*, respectively. Intuitively, we can derive that

$$\begin{aligned} \frac{J_2}{J_2'} &\leq \frac{\sum_{i=1}^n s_i^2 y_{i,j} - (\sum_{i=1}^n \frac{a y_{i,j}}{i})^2}{s_1^2 - (\frac{a}{1} + \sum_{i=2}^{n/m} \frac{a}{im})^2} \\ &\leq \frac{\sum_{i=1}^n s_i^2 y_{i,j}}{s_1^2 - (\frac{a}{1} + \sum_{i=2}^{n/m} \frac{a}{im})^2} \leq \frac{\sum_{i=1}^n s_i^2 y_{i,j}}{2s_1 \times (\frac{a}{1} + \sum_{i=2}^{n/m} \frac{a}{im})} \\ &\leq \frac{\sum_{i=1}^n s_i^2 y_{i,j}}{2s_1 \times s_m} \leq \frac{\sum_{i=1}^{n/m} s_i^2}{2s_1 \times s_m} = \frac{\sum_{i=1}^{n/m} s_i \times s_1}{2s_1 \times s_m} \\ &= \frac{1}{2}(\frac{s_1}{s_1}\frac{s_1}{s_m} + \frac{s_2}{s_1}\frac{s_2}{s_m} + ... + \frac{s_{n/m}}{s_1}\frac{s_{n/m}}{s_m}) \\ &\leq \frac{1}{2}(\frac{s_1}{s_1}\frac{s_1}{s_m} + \frac{s_2}{s_1}\frac{s_1}{s_m} + ... + \frac{s_{n/m}}{s_1}\frac{s_1}{s_m}) \\ &\leq \frac{1}{2}(\frac{s_1}{s_1} + \frac{s_2}{s_1} + ... + \frac{s_{n/m}}{s_1})\frac{s_1}{s_m} \\ &\leq \frac{1}{2}(\frac{s_1+s_2+...+s_{n/m}}{s_1})\frac{s_1}{s_m} \leq \frac{1}{2}\frac{1}{s_1}\frac{s_1}{s_m} \leq \frac{m}{4a}. \end{aligned}$$

Therefore, the approximate ratio of J_2' to J_2 is no less than $4a/m$ where a is the *Euler-Mascheroni* constant.

A.7 Proof of Theorem 3

Let the optimal solution to original problem be OPT, and the initial solution $R_1 \times$ OPT, if the first reduced cost of the column generation method is J_2, then OPT $= R_1 \times$ OPT $- \gamma \times J_2$ where $\gamma \geq 1$ and $\gamma = (R_1 - 1)$ OPT$/J_2$. Similarly, if the first reduced cost of the sub optimal method is J_2', we can derive the objective value $X = R_1 \times$ OPT $- \gamma' \times J_2'$.

On the basis of those two lemmas we can prove that if $\gamma \leq \gamma'$, the approximate ratio is OPT$/X \geq$ OPT$/(R_1 \times$ OPT $- \gamma \times J_2') =$ OPT$/(R_1 \times$ OPT $- ((R_1 - 1)$OPT$/J_2) \times J_2') \geq 1/(1+(m^3-5m^2+6m-8)a/(m^2+m))$. Otherwise, OPT$/X =$ OPT$/(R_1 \times$ OPT $- \gamma' \times J_2') \geq$ OPT$/(R_1 \times$ OPT$) \geq 1/(1+(m^2-m+2)a/(m+1))$. Therefore, OPT $\leq X \leq (1 + (m^3 - 5m^2 + 6m - 8)a/(m^2 + m))$OPT or $(1 + (m^2 - m + 2)a/(m + 1))$OPT $\approx (1 + 0.2m)$OPT.

References

1. F. Bi, L. Chang, X. Lin, L. Qin, and W. Zhang, "Efficient subgraph matching by postponing cartesian products," in *SIGMOD*, 2016, pp. 1199–1214.
2. Y. Low, J. E. Gonzalez, A. Kyrola, D. Bickson, C. E. Guestrin, and J. Hellerstein, "Graphlab: A new framework for parallel machine learning," *arXiv preprint arXiv:1408.2041*, 2014.
3. Z. Chang, L. Zou, and F. Li, "Privacy preserving subgraph matching on large graphs in cloud," in *SIGMOD*, 2016, pp. 199–213.
4. H. Hu, J. Xu, Q. Chen, et al, "Authenticating location-based services without compromising location privacy," in *SIGMOD*, 2012, pp. 301–312.
5. J. Xu, P. Yi, B. Choi, et al, "Privacy-preserving reachability query services for massive networks," in *CIKM*, 2016, pp. 145–154.
6. L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.
7. A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," in *ICDE*, 2006, pp. 24–24.
8. N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *ICDE*, 2007, pp. 106–115.
9. C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *TCC*, 2006, pp. 265–284.
10. M. Yuan, L. Chen, S. Y. Philip, and T. Yu, "Protecting sensitive labels in social network data anonymization," *TKDE*, vol. 25, no. 3, pp. 633–647, 2013.
11. K. Liu and E. Terzi, "Towards identity anonymization on graphs," in *SIGMOD*, 2008, pp. 93–106.
12. C.-H. Tai, P.-J. Tseng, S. Y. Philip, and M.-S. Chen, "Identity protection in sequential releases of dynamic networks," *TKDE*, vol. 26, no. 3, pp. 635–651, 2014.
13. B. Zhou and J. Pei, "Preserving privacy in social networks against neighborhood attacks," in *ICDE*, 2008, pp. 506–515.
14. M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis, "Resisting structural re-identification in anonymized social networks," *PVLDB*, vol. 1, no. 1, pp. 102–114, 2008.
15. L. Zou, L. Chen, and M. T. Özsu, "K-automorphism: A general framework for privacy preserving network publication," *PVLDB*, vol. 2, no. 1, pp. 946–957, 2009.
16. J. Cheng, A. W.-c. Fu, and J. Liu, "K-isomorphism: privacy preserving network publication against structural attacks," in *SIGMOD*, 2010, pp. 459–470.
17. W. Wu, Y. Xiao, W. Wang, Z. He, and Z. Wang, "K-symmetry model for identity anonymization in social networks," in *EDBT*, 2010, pp. 111–122.
18. C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations research*, vol. 46, no. 3, pp. 316–329, 1998.
19. X.-Y. Li, C. Zhang, T. Jung, J. Qian, and L. Chen, "Graph-based privacy-preserving data publication," in *INFOCOM*, 2016, pp. 1–9.
20. S. Hajian, J. Domingo-Ferrer, and O. Farràs, "Generalization-based privacy preservation and discrimination prevention in data publishing and mining," *DMKD*, vol. 28, no. 5-6, pp. 1158–1188, 2014.
21. Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *IJCV*, vol. 40, no. 2, pp. 99–121, 2000.
22. G. Karypis and V. Kumar, "Analysis of multilevel graph partitioning," in *ICS*, 1995, p. 29.
23. S. Schrenk, G. Finke, and V.-D. Cung, "Two classical transportation problems revisited: pure constant fixed charges and the paradox," *Mathematical and computer modelling*, vol. 54, no. 9-10, pp. 2306–2315, 2011.
24. J. Žerovnik, "Heuristics for np-hard optimization problems-simpler is better!?" *Logistics & Sustainable Transport*, vol. 6, no. 1, pp. 1–10, 2015.
25. W.-S. Han, J. Lee, and J.-H. Lee, "Turboiso: towards ultrafast and robust subgraph isomorphism search in large graph databases," in *SIGMOD*, 2013, pp. 337–348.
26. H. Shang, Y. Zhang, X. Lin, and J. X. Yu, "Taming verification hardness: an efficient algorithm for testing subgraph isomorphism," *PVLDB*, vol. 1, no. 1, pp. 364–375, 2008.
27. M. R. Garey and D. S. Johnson, *Computers and intractability*. freeman San Francisco, 1979, vol. 174.
28. H. He and A. K. Singh, "Graphs-at-a-time: query language and access methods for graph databases," in *SIGMOD*, 2008, pp. 405–418.
29. E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Operations research*, vol. 14, no. 4, pp. 699–719, 1966.
30. I. ILOG, "Cplex optimizer," *Available: https://www.ibm.com/cn-zh/marketplace/ibm-ilog-cplex*, 2012.
31. B. Du, S. Zhang, N. Cao, and H. Tong, "First: Fast interactive attributed subgraph matching," in *SIGKDD*. ACM, 2017, pp. 1447–1456.
32. M. Qiao, H. Zhang, and H. Cheng, "Subgraph matching: on compression and computation," *PVLDB*, vol. 11, no. 2, pp. 176–188, 2017.
33. Z. Yang, A. W.-C. Fu, and R. Liu, "Diversified top-k subgraph querying in a large graph," in *SIGMOD*, 2016, pp. 1167–1182.
34. G. Zhu, X. Lin, K. Zhu, W. Zhang, and J. X. Yu, "Treespan: efficiently computing similarity all-matching," in *SIGMOD*, 2012, pp. 529–540.
35. Z. Jorgensen, T. Yu, and G. Cormode, "Publishing attributed social graphs with formal privacy guarantees," in *SIGMOD*, 2016, pp. 107–122.
36. E. Zheleva and L. Getoor, "Preserving the privacy of sensitive relationships in graph data," in *International Workshop on Privacy, Security, and Trust in KDD*, 2007, pp. 153–171.
37. A. Campan and T. M. Truta, "Data and structural k-anonymity in social networks," in *International Workshop on Privacy, Security, and Trust in KDD*, 2008, pp. 33–54.
38. S. Bhagat, G. Cormode, B. Krishnamurthy, and D. Srivastava, "Class-based graph anonymization for social network data," *PVLDB*, vol. 2, no. 1, pp. 766–777, 2009.
39. Z. Fan, B. Choi, J. Xu, and S. S. Bhowmick, "Asymmetric structure-preserving subgraph queries for large graphs," in *ICDE*, 2015, pp. 339–350.

40. D. Xie, G. Li, B. Yao, X. Wei, X. Xiao, Y. Gao, and M. Guo, "Practical private shortest path computation based on oblivious storage," in *ICDE*, 2016, pp. 361–372.
41. K. Nayak, X. S. Wang, S. Ioannidis, U. Weinsberg, N. Taft, and E. Shi, "Graphsc: Parallel secure computation made easy," in *S&P*, 2015, pp. 377–394.
42. J. Gao, J. X. Yu, R. Jin, J. Zhou, T. Wang, and D. Yang, "Neighborhood-privacy protected shortest distance computing in cloud," in *SIGMOD*, 2011, pp. 409–420.
43. N. Cao, Z. Yang, C. Wang, K. Ren, and W. Lou, "Privacy-preserving query over encrypted graph-structured data in cloud computing," in *ICDCS*, 2011, pp. 393–402.
44. Available at: `https://www.oracle.com/a/tech/docs/sg-oow2019-using-graph-analysis-and-fraud-detection-in-fintech-industry.pdf`.
45. M. Hay, C. Li, G. Miklau, and D. Jensen,"Accurate estimation of the degree distribution of private networks," in *ICDM*, 2009, pp. 169–178.
46. V. Karwa, S. Raskhodnikova, A. Smith, and G. Yaroslavtsev, "Private analysis of graph structure," *PVLDB*, vol. 4, no. 11, pp. 1146–1157, 2011.
47. J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao, "Private release of graph statistics using ladder functions," in *SIGMOD*, 2015, pp. 731–745.
48. Q. Ye, H. Hu, M. H. Au, X. Meng, and X. Xiao, "LF-GDPR:Graph Metric Estimation with Local Differential Privacy," in *TKDE*, 2020, doi: 10.1109/TKDE.2020.3047124.
49. H. Jiang, J. Pei, D. Yu, et al, "Applications of Differential Privacy in Social Network Analysis: A Survey," *TKDE*, 2021.
50. X. Ding, S. Sheng, S. Zhou, et al, "Differentially Private Triangle Counting in Large Graphs," *TKDE*, 2021.
51. S. Chen and S. Zhou, "Recursive mechanism: Towards node differential privacy and unrestricted joins," in *SIGMOD*, 2013, pp. 653–664.
52. S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith, "Analyzing graphs with node differential privacy," in *TCC*, 2013, pp. 457–476.
53. W. Y. Day, N. Li, and M. Lyu, "Publishing graph degree distribution with node differential privacy," in *SIGMOD*, 2016, pp. 123–138.
54. Q. Wang, Y. Zhang, X. Lu, et al, "Real-time and spatio-temporal crowd-sourced social network data publishing with differential privacy," *TDSC*, vol. 15, no. 4, pp. 591–606, 2016.
55. H. Liang, H. Yuan. On the complexity of t-closeness anonymization and related problems. in *DASFAA*, 2013, pp. 331–345.
56. J. Ma, B. Yao, X. Gao, et al. Top-k critical vertices query on shortest path. *TKDE*, vol. 30, no. 10, pp. 1999–2012, 2018.
57. M. Shen, B. Ma, L. Zhu, et al. Cloud-based approximate constrained shortest distance queries over encrypted graphs with privacy protection. *TIFS*, vol. 13, no. 4, pp. 940–953, 2017.
58. X. Ding, C. Wang, K. K. R. Choo, et al. A novel privacy preserving framework for large scale graph data publishing. *TKDE*, vol. 33, no. 2, pp. 331–343, 2019.
59. J. Gao, et al. A privacy-preserving framework for subgraph pattern matching in cloud, in *DASFAA*, 2018, pp. 307–322.
60. J. Jiang, P. Yi, B. Choi, et al. Privacy-preserving reachability query services for massive networks, in *CIKM*, 2016, pp. 145–154.
61. S. Yang, S. Tang, X. Zhang. Privacy-preserving k nearest neighbor query with authentication on road networks. *JPDC*, vol. 134, pp. 25–36, 2019.