Two Counting Problems Related to Graph Automorphism Problem

Miao Benjie
516030910509
bjmiao@sjtu.edu.cn

Li Jieyu
516030910503
ljy9792@hotmail.com

***Abstract*—In this work we discussed two counting problems related to automorphism problem. We analyze the problems themselves, design an approximating algorithm to the problems and analyze the algorithm. Finally, we do experiments to verity our result.**

***Index Terms*—graph theory, automorphism, orbit, counting problem**

I. Introduction

Graph Automorphism(GA) problem is one of the most classical problems in graph theory. Intuitively, Graph automorphism find a bijection from the graph node set to itself, judging whether there is a symmetry topology in a problem. Dozens of works [2], [3], [4] have discussed this problem and there seems never an end in the process of exploring this problem.

The time complexity of this problem is somehow a mystery. Tracing back to 1979, [5] have found that GA is unlikely to be either in P problem or in NP-complete problem. In fact, due to the large possibility to pruning, the search space can be largely reduced, making this problem even more practical than we theoretically expected.

Automorphism can actually capture the **symmetry** property of a graph. Consider a complete graph, of which all its permutations are automorphism, and all the nodes can be mapped to some other nodes by some automorphism; on the other hand, Figure 1, known as the smallest asymmetric graph, has no nontrivial automorphism. The capability to describe symmetry makes it useful in the field of graph matching and de-anonymization. In fact, the motivation of our work comes from the application of de-anonymization problem in the social network field.

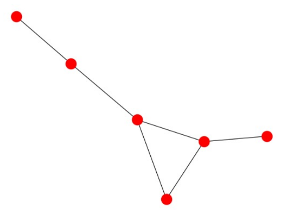

Fig. 1. An Example of Graph Automorphism

In this work we are interested in **two counting problems** related to graph automorphism: first, we want to know how many nodes are *symmetric*, that is, how many nodes can be mapped to another node by some nontrivial automorphism. Second, we want to know how many **orbits** are in the given graph. We will show latter that the answer of these two problems are valuable, since they can directly show symmetry of a given graph, which will lead to an upper bound of de-anonymization algorithm performance on this given graph.

We will also show that these two counting problems are also GA-complete (i.e. they have the same time complexity with the classical GA problem). However, since they are counting problems, there is possibility to using some **approximating algorithm** to get a **sub-optimal** solution. We actually do **design an approximating algorithm** of those two problems. And we both theoretically and experimentally prove the effectiveness of our algorithm, especially in the real-world networks.

The rest of the paper goes as follows: Section 2 shows related works. Section 3 presents the fundamental concepts about the problems we focused on. Section 4 is the analysis of problems themselves and in Section 5 we shows our approximating algorithm and the analysis of the algorithm. Section 6 is the experiment part and we conclude in the last section.

II. Related Works

Graph Automorphism is a classical graph theory problem. [2], [3], [4] are some of the previous works.

Orbit is an concept from abstract algebra, and is closely related to the permutation group. [6], [7] are some of the works.

Also, the motivation of our work comes from a social network problem : de-anonymization. [8] is the most related work.

III. Preliminary Definitions and Concepts

Definition 1 (Graph Isomorphism & Automorphism): For two graph $G_1 = (V_2, E_2)$ and $G_2 = (V_2, E_2)$, if a bijection $f : V_1 \to V_2$ such that

$$\forall (v_i, v_j) \in E_1 \leftrightarrow (f(v_i), f(v_j)) \in E_2$$

then we say G_1 and G_2 are *isomorphic*, and f is an *isomorphism* of G_1 onto G_2.

If $G_1 = G_2$, then f is an *automorphism* of G onto itself.

Formally, an automorphism f is a **permutation** σ on the vertex set V. For Figure2, one automorphism can be

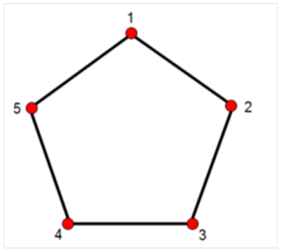

Fig. 2. An Example of Graph Automorphism

represented as :

$$\left\{ \begin{array}{cccccc} V: & 1 & 2 & 3 & 4 & 5 \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ f(V): & 2 & 3 & 4 & 5 & 1 \end{array} \right\}$$

In fact, automorphism is a concept from the scope of abstract algebra, which is used commonly to represent the symmetry of a group or a set.

Our other focus in this work uses another concept from abstract algebra : **Orbit**. Here, for the sake of simplicity, we slightly abuse the original concept of orbit, using a simpler concept to define it.

Definition 2 (Orbit): In graph $G = (V, E)$, for $v \in V$, the orbit that contains v, $ORB(v)$, is defined as follows :

$$ORB(v) = \{v_j \in V | \exists f, f(v) = v_j\}$$

where f is some automorphism permutation.

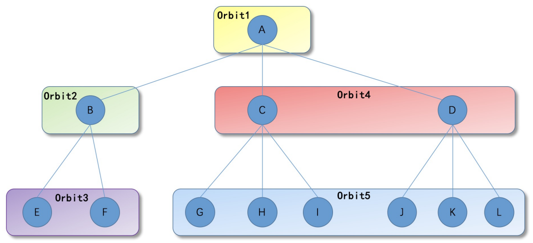

Fig. 3. An Illustration of Orbit

Intuitively, an orbit contains all the nodes that are *symmetric*.

IV. Problem Formulation

Based on our preliminary definitions and concepts, we can now give our formal problem formulation.

- *Problem 1 (Graph Automorphism Vertex Counting):* (#GA-VERTEX) Given a graph $G = (V, E)$, find $|V_{aut}|$, where $V_{aut} = \{v_i \in V | \exists f, f(v_i) = v_j\}$, where f is any automorphism permutation.
- *Problem 2 (Orbit Counting Problem):* (#ORB) For graph $G = (V, E)$, find the number of orbits it contains. That is, let a partition of V is $\{V_1, V_2, ..., V_k\}$. If for each $V_i (i = 1, 2, ..., k)$, $\forall v_{i_1}, v_{i_2} \in V_i$, there exists f , an automorphism of G, such that $v_{i_1} = f(v_{i_2})$, then the size of partition is the answer.

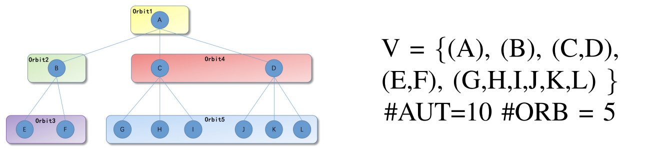

Fig. 4. An Illustration of #AUT and #ORB

V. Problem Analysis

In this section, we do some analysis on both two problems. We first **analyze their time complexity in comparison with the classical automorphism problem**. Second, we **discuss the significance** of these two problems by introducing an important application using the result of the two problems we focused on. In the last part of the section, we **deduce some important property** related to the problems, on which we based when designing our efficient algorithm to those two problems.

A. Complexity of #AUT problem and #ORB problem

In this part we show the **equivalence of #AUT, #ORB with the traditional graph automorphism(GA) problem in time complexity**. Here we use the notation $A \leq_P B$ to denote that problem A can be polynomially reduced to B.

Theorem 5.1: #AUT $\leq_P$ GA, #ORB $\leq_P$ GA.

Proof: Using the result of GA, we can count the number of all nontrivial automorphism, count all the nodes in nontrivial automorphism group. That is the answer of #AUT.

For #ORB problem, we can use a Find-Union data structure to get all the orbits in the given graph. ■

Actually, another version of graph automorphism problem is to judge whether there exists any nontrivial automorphism in a given graph (we denote it as GA-Judge problem). Clearly, GA-Judge $\leq_P$ #AUT and GA-Judge $\leq_P$ #ORB.

Fortunately, [9] have proved that GA and GA-Judge can be polynomially reducible. Therefore, we come to the conclusion that #AUT, #ORB are equivalent to GA problem.

B. Value of #AUT problem and #ORB problem

To show the value of those two counting problems, we have to introduce an important problem in the social network: **de-anonymization** problem. Typically, a de-anonymization problem is to match the nodes of two graphs : one is labeled with name, and the other is anonymous. Figure 5 is an illustration.

Problem 3 (De-anonymization Problem): Let $G = (V, E, \mathbf{M})$ be the underlying social network, where V is the set of nodes, E is the set of edges and $\mathbf{M}$ denotes the adjacency matrix of G.We define $G_1(V_1, E_1, \mathbf{A})$ as the published network and $G_2(V_2, E_2, \mathbf{B})$ the auxiliary network. G_1, G_2 have the same node of G, but edges are sampled (with a certain rate) from G. Given the published network G_1 and the auxiliary network G_2, the problem of social network de-anonymization aims to find a bijective mapping $\pi : V_1 \mapsto V_2$ that reveals the correct correspondence of the

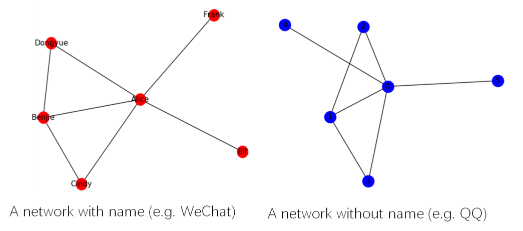
Frank
Dongyue
Alice
Benjie
Elf
Cindy
A network with name (e.g. WeChat)
4
2
0
5
1
3
A network without name (e.g. QQ)

Fig. 5. De-anonymization Problem

nodes in the two networks. Equivalently, a mapping π can be represented as a permutation from V_1 to V_2.

Then we can show some useful property of the problems we are focused on.

Theorem 5.2 (Upper Bound of Guaranteed De-anonymization): Given a G, $|V| - |V_{aut}|$ is the upper bound of the node which are guaranteed to be correctly matched.

Proof: For those in $|V_{aut}|$, we can do nothing better than a guess. ■

Theorem 5.3 (Upper Bound of De-anonymization Expectation): Given a graph G, $|Orb|$ is the upper bound of expectation of correctly matched node.

Proof: Similar to the problem **Guessing Card**. Due to the linearity of expectation, the expectation of each orbit at most one. ■

This part is not the main thread of this work. The only thing we want to show here is that these two problem are indeed useful.

C. Useful Definition and Property

In this part we have some additional definitions to introduce.

Definition 3 (Automorphic pair): For a pair of nodes v_i, v_j in graph $G = (V, E)$. If there exists an automorphism mapping f such that $f(v_i) = v_j$, we say such pair of node *automorphic vertex pair*, or *automorphic pair*.

Definition 4 (Automorphic subset): For a subset $V_i \subseteq V$, if for each $v_{i1}, v_{i2} \in V_i$, (v_{i1}, v_{i2}) is an automorphic pair, we say such pair of node *automorphic vertex subset*, or *automorphic subset*.

We can easily find the relationship between orbit and automorphism subset.

Proposition 5.4: Each orbit is an automorphic subset. In fact, for any vertex v, the automorphic subset containing v, $aut(v) \subseteq orb(v)$.

Definition 5 (Intra-edge & Inter-edge): For a subset $V' \subseteq V$, consider an edge $e = (v_i, v_j)$:

(1) If $v_i \in V', v_j \in V'$, then e is an intra-edge of V'.

(2) If $v_i \in V', v_j \in V - V'$(or vice versa), then e is an inter-edge of V.

This definition is essential, because two conditions of judging automorphism result comes from it.

Proposition 5.5 (Condition for automorphic pair, I): If for a vertex pair $V' = (v_i, v_j)$, if all inter-edge of edge $E' = \{(v_i, v_k), (v_j, v_k)\}$ then (v_i, v_j) is an automorphic pair.

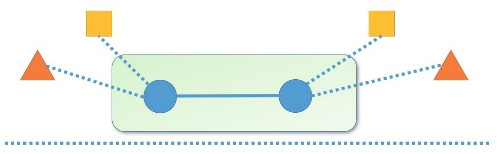

Fig. 6. Intra- and inter- edge

Proposition 5.6 (Condition for automorphic pair, II): If for a vertex pair $V' = (v_i, v_j)$:

(1) the numbers of intra-edge on v_i and v_j are equal, and the outer vertex are one-one matched. That is, $\forall v_k$ that is adjacent to v_i, there contains a v_l adjacent to v_j and (v_k, v_l) is an automorphic pair.

Then we can claim that (v_i, v_j) is a automorphic pair.

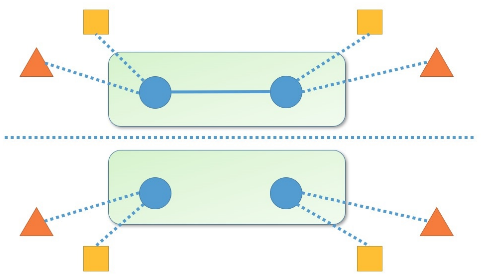

Fig. 7. Automorphism condition for Automorphic pair

VI. Algorithm Design & Analysis

By observing the real-world network, we find that many automorphism have small searching depth. Thus it comes to our mind that we can use a depth-limited search as our approximating algorithm.

Here we need a new definition: chordless cycle.

Definition 6 (Chordless Cycle): For any two vertexes v_i and v_j in a cycle $C = \{v_1, v_2, ..., v_n\}$, if v_i and v_j are adjacent if and only if $|i - j| \equiv 1(mod\ n)$. Then C is a **Chordless Cycle**.

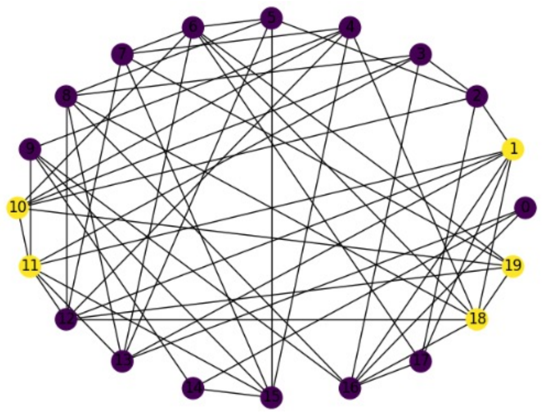

Fig. 8. An illustration of chordless cycle

A. Algorithm Design

We can now give out our approximating algorithm. Basically it is a depth-limited search. See algorithm 12 for detail.

```
input : $d$-search depth limit, $k$-max chordless cycle size.
output: The answer of #AUT and #ORB
1  Find a chordless cycle with size no larger than k (vertex pair when k=2)
2  Set $V' = \{v_1, v_2, \ldots, v_k\}$ I ← all inter-edges of V. If search depth exceeds d, return #AUT, #ORB.
3  if all the vertexes have same number of adjacent inter-edge then
4      if All inter-edge is connected to one single node then
5          $V'$ is an automorphism subset
6          #AUT ← #AUT + $|V'|$
7          #ORB ← #ORB ∪ V'
8      end
9      if All inter-edge is connected to another vertex subset $V''$ 'symmetrically' then
10         Let $V''$ to be the new to-be-determined vertex subset.
11     end
12 end
```

B. Algorithm Analysis

Here we need some definition and lemmas before we analyze our algorithm.

Definition 7 (Symmetric Center): For a subgraph $G' \subset G$. The vertex set of G' is $V' = \{v_1, v_2, ..., v_n\}$. Then G' is a **Symmetric Center** if and only if there is an automorphism transformation with a factor $(v_{i_1}, v_{i_2}, ..., v_{i_k})$, where $\{v_{i_1}, v_{i_2}, ..., v_{i_k}\}$ is a subset of V'. Notably, symmetric center is not always a chordless cycle.

Lemma 6.1: For a chordless cycle $C = \{v_1, v_2, ..., v_n\}$. d_i is the degree of v_i. If there isn't a bijection between $\{d_i\}$ and itself so that $(d_1, d_2, ...d_n) = (d_{i_1}, d_{i_2}, ..., d_{i_n})$ where the latter is the vector after mapping, then C isn't automorphism.

Lemma 6.2: For any cycle $C = \{v_1, v_2, ..., v_n\}$, if it is a symmetric center and has at least 1 inter-chords, then the spanning chordless cycle is also a symmetric center.

Obviously, if a chordless cycle is a symmetric center, then it will not be a symmetric center when adding some inter-chord. On the other hand, if a cycle is a symmetric center, then it must be a symmetric center when deleting all of its inter-chords. So the lemma is correct. Further, if the spanning chordless cycle is not a symmetric center, then the cycle must not be a symmetric center.

Based on what Lemma6.2 shows, we can simplify the problem by giving a hypothesis that there at least 1 chorless cycle symmetric center. Because we can use the spanning chordless cycle to take place a symmetric center if there isn't a chorless cycle symmetric center. Actually, in local search algorithm for this problem, it's usually need one time's search to exclude a chordless cycle if it's not a symmetric center. According to Lemma 6.1, we just need to justify weather it has the bijection and this problem can be incorporated into a combination counting problem.To provide this phenomenon, we can simplify this subproblem as assuming the out-degree of the vertex in the chordless cycle is less than 2. In this case, it increases the possibility of finding a symmetric structure drastically because of the short ranges of degree. Assume the chordless cycle has n vertexes and m vertexes of them has 1 degree. Then the possibility of finding a symmetric structure in this graph is showing below.

$$P = \frac{C_{\frac{n}{2}}^{\frac{m}{2}}}{C_n^m} \tag{1}$$

And if we use Stirling's approximation to approximate it

$$P = \frac{\sqrt{\pi n}(\frac{n}{2e})^{\frac{n}{2}}\sqrt{2\pi m}(\frac{m}{e})^m\sqrt{2\pi(n-m)}(\frac{n-m}{e})^{n-m}}{\sqrt{2\pi n}(\frac{n}{e})^n\sqrt{\pi m}(\frac{m}{2e})^{\frac{m}{2}}\sqrt{\pi(n-m)}(\frac{n-m}{2e})^{\frac{n-m}{2}}} \tag{2}$$

Then we can simplify it as

$$P = \frac{\sqrt{2}m^{\frac{m}{2}}(n-m)^{\frac{n-m}{2}}}{n^{\frac{n}{2}}} \tag{3}$$

Assume the probability of there exists an edge between two vertexes is p, then

$$E(m) = pn \tag{4}$$

$$P \approx \sqrt{2}p^{\frac{pn}{2}}(1-p)^{\frac{1-p}{2}n} \leqslant \frac{1}{2^{\frac{n-1}{2}}} \tag{5}$$

The possibility will be smaller in real-word problem. Because when the ranges of degree get large, the number of the arrangement of degree increases and the arrangement with symmetric structure decreases. As a result, the possibility will be smaller.

However, it does not mean there is not a symmetric structure in this graph. In a simple view, there maybe exists an exactly the same subgraph in this graph and both of them are holding on a symmetric center. In this case, there is a possibility that the graph still has a symmetric structure.

C. Complexity

On the first step of the local search algorithm, we need to focus on a subgraph of the huge graph. Assume the subgraph has n vertexes. Firstly, we can use DFS to find a cycle. This process has $O(n)$ time complexity. And then use DFS continuously until we find a chordless cycle. To produce the time complexity, we can choose the vertex with the large degree in this cycle as the start vertex and check weather the vertex is adjacent with the start vertex at first in every step. In this case, the time complexity of this process is also $O(n)$. Then we come back to the original graph and check weather the chordless cycle is a possible-symmetric-center. This process also has $O(n)$ time complexity because it can be incorporated into bracket matching problem. So the time complexity of this algorithm is $O(kn)$, where k is the search times.

VII. Experiment

We have implemented our algorithm and done some experiments to testify our theoretical result. Since we are focused on real-world network, so we use Erdos-Renyi [1] model, the most common model to simulate real-world network.

We use different p, the probability between each two edge. We run the algorithm using Python3, and plot the result as follows.

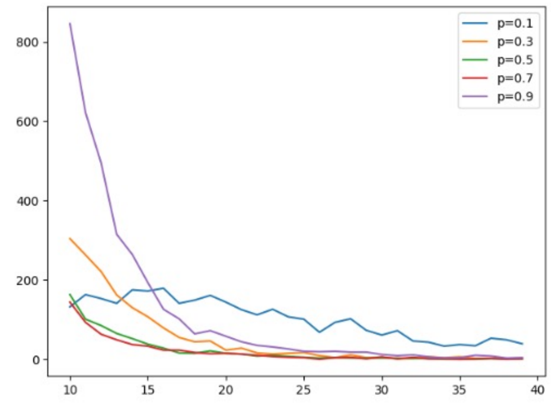

Fig. 9. Result

We can see that the algorithm can have good performance as the graph goes larger. That is because as the graph becomes larger, the graph are less likely to have large automorphism structure.

VIII. Conclusion

In this work we discussed two counting problem related to automorphism problem. We analyze the problems themselves, design an approximating algorithm and analyze the algorithm.

References

[1] Erdos P, Rnyi A. On the evolution of random graphs[J]. Publ. Math. Inst. Hung. Acad. Sci, 1960, 5(1): 17-60.
[2] Colbourn C J, Booth K S. Linear time automorphism algorithms for trees, interval graphs, and planar graphs[J]. SIAM Journal on Computing, 1981, 10(1): 203-225.
[3] Lpez-Presa J L, Chiroque L N, Anta A F. Novel Techniques for Automorphism Group Computation[C]//International Symposium on Experimental Algorithms. Springer, Berlin, Heidelberg, 2013: 296-307.
[4] Hopcroft J E, Wong J K. Linear time algorithm for isomorphism of planar graphs (preliminary report)[C]//Proceedings of the sixth annual ACM symposium on Theory of computing. ACM, 1974: 172-184.
[5] Garey M R, Johnson D S. Computers and intractability: a guide to NP-completeness[J]. 1979.
[6] Seress . Permutation group algorithms[M]. Cambridge University Press, 2003.
[7] Dixon J D, Mortimer B. Permutation groups, GTM 163[J]. 1996.
[8] Fu L, Fu X, Hu Z, et al. De-anonymization of Social Networks with Communities: When Quantifications Meet Algorithms[J]. arXiv preprint arXiv:1703.09028, 2017.
[9] Gl A, Halevi S, Lipton R J, et al. Computing from partial solutions[C]//Computational Complexity, 1999. Proceedings. Fourteenth Annual IEEE Conference on. IEEE, 1999: 34-45.