The Hong Kong Polytechnic University

Department of Computing

Group-based Techniques for Identifying Top-K Degrees in Hidden Bipartite Graphs

by

Jianguo Wang

A thesis submitted in partial fulfillment of the requirements

for the degree of

Master of Philosophy

August 2012

(Temporary Binding for Examination Purposes)

CERTIFICATE OF ORIGINATLITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

. . . .

Jianguo Wang

August 2012

Abstract

Graphs are of fundamental importance in modeling data in various domains. Usually, graphs have both their vertices and edges available, which we refer to as *explicit* graphs. However, in applications such as bioinfomatics, graphs may only have vertices available (e.g., proteins), while the edges are *unknown* initially (e.g., interactions among proteins), which are called *hidden* graphs. Thus, the *edge probe tests* (e.g., biological experiments) are required to detect the presence of edges.

This work studies the kMCV (k most connected vertices) problem on a hidden bipartite graph $\mathcal{G}(\mathcal{B}, \mathcal{W})$ where $\mathcal{B}$ and $\mathcal{W}$ are two independent vertex sets. The kMCV problem aims to find the top k vertices in $\mathcal{B}$ that have the maximum degrees. It has applications in spatial databases, graph databases, and bioinformatics. There is a prior work on the kMCV problem, which is based on the "2-vertex test" model, i.e., an edge probe test can only reveal the existence of an edge between two individual vertices.

We study the kMCV problem, in the context of a more general edge probe test model called "group test". A group test can reveal whether there exists some edge between a vertex and a group of vertices. If the group test model is used properly, a single invocation of a group test can reveal as much information as multiple invocations of 2-vertex tests. We discuss the cases and applications where the group test model could be used, and make the following contributions.

1. We propose an algorithm, namely, GMCV, that adaptively leverages the group test model to solve the kMCV problem.
2. We derive cost models for our algorithm GMCV and the prior algorithm.
3. We conduct extensive experiments on both synthetic and real life datasets, and show that our GMCV outperforms the prior algorithm significantly.

List of Publications

- **J. Wang**, E. Lo, and M. L. Yiu. Identifying the Most Connected Vertices in Hidden Bipartite Graphs using Group Testing. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2012.

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to those who helped me during my studies.

First of all, I am grateful to have two excellent supervisors during my studies, Dr. Man Lung Yiu and Dr. Eric Lo. Not only because of their constant support and patient supervision, but also because they introduced me to the academic world. I am thankful for their insightful suggestions and guidance on how to proceed at every stage during my studies. Without them, the work would not have been completed!

Next, I would like to thank my many friends. In particular, I would like to thank Jeppe Thomsen, a very nice and helpful guy. Also, I thank my group mates Duncan Yung, Andy Ho and Yu Li.

Finally, I deeply thank my parents and my sister for their love, support and encouragement, which have made me even more motivated to complete the work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

A graph is called *hidden* if the edges are not explicitly given and *edge probe tests* are required to detect the presence of edges [18]. Recently, Tao et al. [29, 28] studied the k *most connected vertices* (kMCV) problem on hidden bipartite graphs. Specifically, given a hidden bipartite graph $\mathcal{G}$ with two independent vertex sets $\mathcal{B}$ (black vertex set) and $\mathcal{W}$ (white vertex set), the kMCV problem is to find the top k vertices in $\mathcal{B}$ that have the maximum degrees. Figure 1.1 shows a hidden bipartite graph $\mathcal{G}$, where $\mathcal{B} = \{b_1, b_2\}$ and $\mathcal{W} = \{w_1, w_2, \ldots, w_8\}$. The 1MCV returns vertex b_1 as the result since it has the largest degree. The problem is trivial on conventional bipartite graphs but not in the case of hidden graphs because edge probe tests are usually expensive operations (e.g., biological experiments, graph operations). The applications of finding the kMCV on a hidden bipartite graph include distance join on road networks, bioinformatics, and graph pattern matching [29, 28].

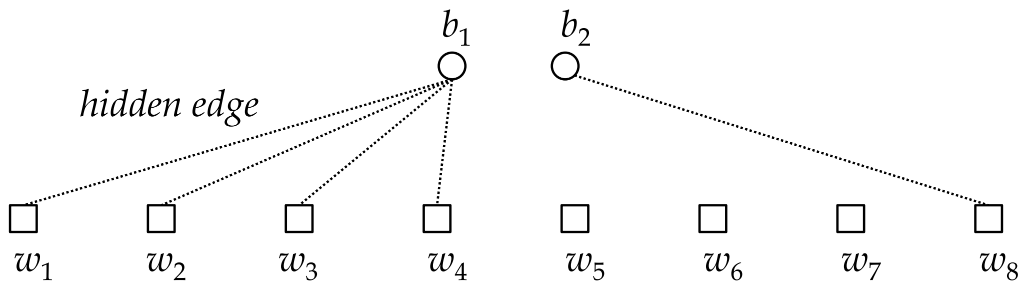


Figure 1.1. A (hidden) bipartite graph $\mathcal{G}(\mathcal{B}, \mathcal{W})$; edges are not explicitly given

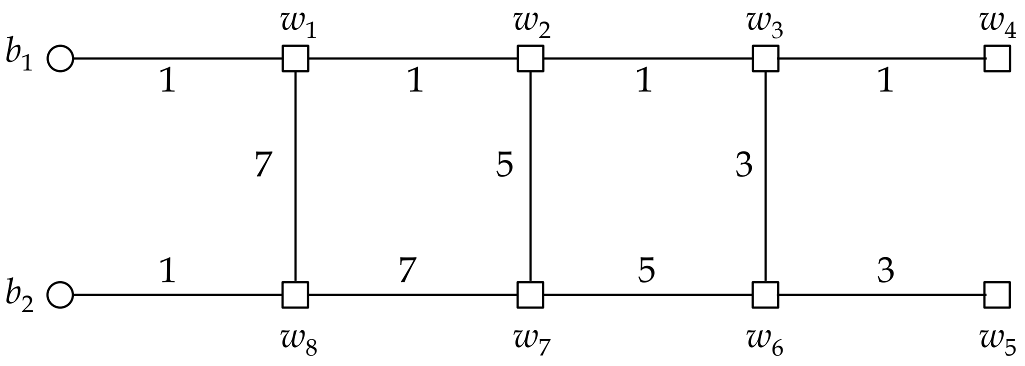


Figure 1.2. Example of a road network

Application 1: Distance Join on Road Networks.

Let $\mathcal{B}$ and $\mathcal{W}$ be the hotel set and scenic spot set, which constitute a bipartite graph $\mathcal{G}(\mathcal{B}, \mathcal{W})$. The distance join between $\mathcal{B}$ and $\mathcal{W}$ gives a collections of pairs $\langle b, w \rangle$ ($b \in \mathcal{B}$, $w \in \mathcal{W}$), where b and w satisfy a join predicate [23]. A hotel $b \in \mathcal{B}$ and a scenic spot $w \in \mathcal{W}$ has an edge if their distance is less than a threshold θ_{dist}, e.g., 5 km, where the distances are *shortest path distances*. Therefore, the kMCV problem could help discover the most convenient hotels. While the edges on $\mathcal{G}$ are not given initially, a shortest path algorithm could be executed to detect their presence. Figure 1.2 shows a road network. Figure 1.1 is the hidden graph representation of distance join on Figure 1.2, using $\theta_{dist} = 5$. To detect whether hotel b_1 and scenic spot w_2 have an edge connecting in Figure 1.1, we can run a *shortest path algorithm* as the edge probe test to find the shortest path between b_1 and w_2 in Figure 1.2. In this example, the shortest path distance between b_1 and w_2 is 2, thus after the execution of the shortest path algorithm, the edge that connects b_1 and w_2 in Figure 1.1 becomes explicit. Shortest path queries on large graphs are usually computationally expensive [30]. Therefore, the goal of kMCV is to find the answer using an efficient strategy.

Application 2: Bioinformatics.

In bioinformatics, interactions between proteins are often represented as graphs. Specifically, the interactions between bait proteins ($\mathcal{B}$) and prey proteins ($\mathcal{W}$), could form a hidden bipartite graph $\mathcal{G}(\mathcal{B}, \mathcal{W})$ [21, 22]. An edge (b, w) represents a bait protein b interacts with a prey protein w and this interaction could be discovered by carrying out an edge probe test in the form of a *biological experiment*, which may take hours or days [17]. The kMCV problem is to find the most active proteins. And it would be beneficial if there is a way to get the answer efficiently.

Application 3: Graph Pattern Matching.

Applications like drug discovery often need to identify the graph patterns that match the most number of data graphs [29, 28]. The discovery process usually involves testing whether a graph pattern b is a sub/super-graph of a data graph w. An edge is present if such a containment relationship exists between b and w. Such information, however, remains hidden unless an explicit sub/super-graph

Table 1.1. Applications that can apply group test

Application	**Meaning of the Black Vertex Set $\mathcal{B}$**	**Meaning of the White Vertex Set $\mathcal{W}$**	**Meaning of a Hidden Edge (b, w)**	**Meaning of a Group Test $Q(b, W)$**	**External Cost of a Group Test**
Distance join	locations (hotel sets)	locations (spot sets)	the distance of b and w is less than a threshold θ_{dist}	Run shortest path algorithm: the distance of b and at least one vertex in W is less than θ_{dist}	sub-linear to group size
Bioinformatics	bait proteins	prey proteins	b interacts with w	Conduct biological experiment: b interacts with at least one vertex in W	constant
Graph pattern matching	data graphs	data graphs	b is a sub/super-graph of w	Query on graph index: b is a sub/super-graph of at least one data graph in W	sub-linear to group size

containment test is carried out. Unfortunately, such testing is known to be expensive, e.g., a *subgraph isomorphism test* is NP-complete [9, 27]. Therefore, it is necessary to devise an efficient algorithm for the kMCV problem to speed up the drug discovery process.

As the pioneering work, [29, 28] developed an algorithm, SOE[1], to solve the kMCV problem. SOE is based on *2-vertex edge probe test*, or simply *2-vertex test* [7], i.e., each edge probe test $Q(b, w)$ takes as inputs *one* black vertex $b \in \mathcal{B}$ and *one* white vertex $w \in \mathcal{W}$, and returns 1 if b possesses an edge with w in the hidden bipartite graph $\mathcal{G}$ and 0 otherwise. In many applications [11, 13, 32, 7], the more general *vertex-group edge probe test* is used as a replacement of the 2-vertex model. Specifically, a vertex-group edge probe test, or simply, a *group test*, takes as inputs *one* black vertex $b \in \mathcal{B}$ and a *group* of white vertices $W \subseteq \mathcal{W}$, denoted as $Q(b, W)$, and returns 1 if there exists at least one white vertex $w \in W$

[1]Actually, [29, 28] proposed two algorithms: SS (Sample-and-Sort) and SOE (Switch-on-Empty). Since SOE outperforms SS in both theory and in practice, we therefore focus on SOE only.

possessing an edge with b in the hidden graph $\mathcal{G}$ and 0 otherwise. We observe that such a test model is also applicable to the kMCV problem (in above applications).

- In the distance join application, if a road network index [19, 25, 31] is available, a group test $Q(b, W)$ can be implemented by asking the road network index the nearest neighbor of a vertex b (denoted as w_{nn}) in a given *group of vertices* W. With the road network index (e.g., distance signature in [19]), we do not need to calculate the distance between b with every vertex in W, because the index can prune those unpromising results as we only concern about the nearest one. If $dist(b, w_{nn}) > \theta_{dist}$, we learn that all vertices in W are beyond θ_{dist} of b, therefore none of the vertices in the group W connects with b in the hidden graph, i.e., $Q(b, W) = 0$. Otherwise, we get $Q(b, W) = 1$.

- In bioinformatics, literature does show that many biological experiments can be set up to tell whether there are reactions between a protein b and a *set of proteins* W [22, 7].

- In the graph matching application, a graph index $I_{\mathcal{W}}$ (e.g., FG-index [9], cIndex [8], GPTree [33]) can be built on a *set of data graphs* $\mathcal{W}$. A group test $Q(b, W)$ can be regarded as a pattern query b on the set $W \subseteq \mathcal{W}$ to check whether there exists a data graph $w \in W$ such that b and w satisfy the containment relationship. If yes, then $Q(b, W) = 1$, and $Q(b, W) = 0$ otherwise. Notice that W corresponds to a particular subtree of the index $I_{\mathcal{W}}$. Thus, the group test can be implemented by issuing b as a graph query to the corresponding subtree of $I_{\mathcal{W}}$. In this case, the graph index can avoid the test between b and every vertex in W, because the index can prune those unpromising results.

Table 1.1 gives a summary of how the above applications associated with the kMCV problem in the context of the group test model.

The applicability of the group test model on the kMCV problem raises a very interesting research question: *Can we leverage the group test model to solve the kMCV problem more efficiently?* Specifically, a group test $Q(b, W)$ returning 0 is equivalent to revealing many hidden edges in a row: $Q(b, w_1) = 0$, $Q(b, w_2) = 0$, ..., $Q(b, w_i) = 0$, for all $w_i \in W$. If an algorithm can leverage it smartly and correctly, the number of tests can be significantly reduced. However, although the use of group test may reduce the number of tests in solving the kMCV problem, we have to ensure that the actual cost of solving the kMCV problem can essentially be reduced. That is because the cost (e.g., monetary cost, running time) of a group test execution, in which we call that as *external cost*, may be more than the external cost of a 2-vertex edge probe test execution, because the former may take more than two white vertices as input. Fortunately, in all of the applications that we concern, the external cost of a group test is indeed sub-linear to or even independent of the input size. For example, in the distance join application and the graph pattern matching application, it has been shown that the external cost (running time) of checking the nearest neighbor between a vertex b and a set of vertices W using a road network index, and the external cost (running time) of checking the containment relationship between a pattern b and a set of data graphs W using a graph index, are sub-linear to the size of W [19, 25, 31, 9, 8, 33], because of the indices' high pruning effectiveness. In bioinformatics, it is a well known fact that the external cost of a group test, no matter in terms of the monetary cost (e.g., the cost of the chemical used) or the time to finish an experiment, is independent of the number of input chemicals involved in the experiment [4, 5, 3, 15].

To leverage the group test model, we have to design the algorithm carefully because it is tricky to determine the input size of the white vertex set, i.e., $|W|$, for each group test. Even though the external cost of a group test is usually sub-linear to or independent of the group size, we still should not deliberately include a lot of vertices in each group test because that would increase the chance of the test result being 1. Such a result is actually not informative because it does not reveal any hidden edge between any pair of black vertex and white vertex. However, if a very small group size is used, the power of the group test model may not be well exploited. Therefore, it is challenging to leverage the group test model in a productive manner.

Based on the discussions above, we propose an algorithm, GMCV, that leverages the group test model to solve the kMCV problem. Note that if the group size $|W|$ is always set to 1, a group test is the same as 2-vertex test. Therefore, GMCV is more general than SOE. GMCV adaptively controls the group sizes based on the data characteristics during execution. For applications like distance join and graph pattern matching, GMCV can be regarded as a usual computer algorithm which aims to solve the kMCV problem efficiently. For applications like bioinformatics, GMCV can serve as an offline human-involving tool like [24] that assists human (scientists) in scheduling their actions (experiments) using the least amount of external resources. Specifically, GMCV can suggest a scientist what experiment should to be done after finishing the current experiment (which may take days).

1.2 Problem Definition

We formally define the kMCV (k most connected vertices) problem under the group test model.

Let $G = (\mathcal{B}, \mathcal{W}, \mathcal{E})$ be a bipartite graph, where $\mathcal{B}$ is a set of black vertices, $\mathcal{W}$ is a set of white vertices, and $\mathcal{E}$ is a set of edges connecting vertices in $\mathcal{B}$ and $\mathcal{W}$. G is hidden if $\mathcal{E}$ is not explicitly given. An edge probe test, or simply a test, can be carried out to detect the presence of edges.

Definition 1 (2-vertex test) *An edge probe test $Q(b, w)$ is called a* 2-vertex test *if it asks whether a black vertex $b \in \mathcal{B}$ connects with a white vertex $w \in \mathcal{W}$:*

$$Q(b, w) = \begin{cases} 1 \text{ , } \textit{if } (b, w) \in \mathcal{E} \\ 0 \text{ , } \textit{if } (b, w) \notin \mathcal{E} \end{cases}$$

The 2-vertex test method is used by SOE [29, 28]. As mentioned earlier, in many applications, e.g., distance join, protein-protein interaction, we can test a group of vertices together.

Definition 2 (group test) *Let W be a group of white vertices, an edge probe test $Q(b, W)$ is called a* group test *if it asks whether a black vertex $b \in \mathcal{B}$ connects with at least one white vertex $w \in W$:*

$$Q(b, W) = \begin{cases} 1 \text{ , } \textit{if } \exists w \in W, (b, w) \in \mathcal{E} \\ 0 \text{ , } \textit{if } \forall w \in W, (b, w) \notin \mathcal{E} \end{cases}$$

When $|W| = 1$, a group test is the same as a 2-vertex test. Hence, a 2-vertex test is a special case of a group test. Depending on the actual applications, the cost of a group test may or may not depend on the input sizes.

Definition 3 (external test cost β) *Let $Q(b, W)$ be a group test, the external cost (e.g., monetary cost, running time) of carrying out such a test is denoted as $\beta(b, W)$. Furthermore, we assume $\beta(b, W)$ is a function of its input size, i.e., $\beta(|W|)$.*

Definition 4 (kMCV) *Given a hidden graph $\mathcal{G} = (\mathcal{B}, \mathcal{W}, \mathcal{E})$, a user-threshold k, identify a minimal result set $R \subseteq \mathcal{B}$ such that:*

1. *$|R| \geq k$; and*
2. *$d_i > d_j$ for any $b_i \in R$ and $b_j \in \mathcal{B} \setminus R$, where d_i is the degree of b_i.*

The goal of this work is to minimize the total external test cost of solving the kMCV problem using the group test model.

1.3 Thesis Organization

The rest of the work is organized as follows. We review the related work in Chapter 2. Then, we present the technical contributions in the following order:

- First, we present the details of GMCV, a more general algorithm for solving the kMCV problem, in Chapter 3.
- Then, we present cost models of GMCV and SOE, in Chapter 4. Notice that the total external test cost of an execution of GMCV not only depends

on (i) the number of group tests executed, but also (ii) the input size to each group test and (iii) the implementation of the group test. For example, the time complexity of a group test in the distance join application is sub-linear to the input group size. However, in bioinformatics, a group test is an actual (chemical/biological) experiment, in which its cost (running time/monetary cost) is independent of the group size.

- Finally, we experimentally evaluate GMCV in Chapter 5. The evaluation is done on both real life datasets and synthetic datasets. The experimental results align with our theoretical results and show that GMCV is a good general alternative to SOE.

After presenting the above contributions, we conclude the work in Chapter 6 and include certain lemmas and their proofs in the Appendix (Chapter 7). Table 1.2 summarizes the symbols used in the subsequent chapters.

Table 1.2. Summary of notations

Symbol	Meaning
$\mathcal{B}$	black vertex set
$\mathcal{W}$	white vertex set
B	subset of $\mathcal{B}$
W	subset of $\mathcal{W}$
$W_i^j (W^j)$	the j-th test set of b_i (b)
R	result set
b	a black vertex
w	a white vertex
b_i	a black vertex in R
b_j	a black vertex not in R
d (d_i)	the degree of b (b_i)
τ	the k-th largest degree in R
μ	the maximum degree upper bound of vertices not in R
$Q(b, w)$	2-vertex test of b and w
$Q(b, W)$	group test of b and W
$\beta(b, w)$ or $\beta(1)$	external test cost of $Q(b, w)$
$\beta(b, W)$ or $\beta(\|W\|)$	external test cost of $Q(b, W)$

Chapter 2

Related Work

This work is related to hidden graph (Chapter 2.1), group testing (Chapter 2.2) and the prior work solving the kMCV problem (Chapter 2.3). We summarize the related work in Table 2.1.

Table 2.1. Summary of related work

	graph problem	non-graph problem
2-vertex test	kMCV [29, 28]	blood test [11]
group test	kMCV [this work] hamiltonian circuit [18] graph testing [16] graph reconstruction [3, 4, 5, 21, 7, 22]	denial-of-service [32] finding defective items [11, 6, 12, 26, 13]

2.1 Hidden Graph

A graph G is composed of two components: a vertex set V and an edge set E. Usually, both V and E are available for a given graph G. However, this is not the case in some applications, e.g., bioinformatics, where V denotes proteins and E denotes interactions among proteins. In this scenario, G is called a *hidden graph* because only V is available but E is *unknown* initially.

Hidden graph has been an active research topic in the computing theory community [18, 4, 3]. Applications of hidden graph are mostly bioinformatic related. One branch of hidden graph research is *graph testing*: given a hidden graph $\mathcal{G}$, the objective is to test whether $\mathcal{G}$ possesses a certain property (e.g., k-colorable [16]) using a minimal number of edge probe tests (e.g., biological experiments). Another branch of hidden graph research is *graph learning*: given a hidden graph $\mathcal{G}$, the objective is to reconstruct the whole graph using a minimal number of edge probe tests [18, 4, 3, 7, 15]. As argued by [29, 28], the kMCV problem is different from those work because it neither tests the possession of any property of the hidden graph, nor reconstructs the whole graph.

2.2 Group Test

2.2.1 General Group Test

Group test is motivated for uncovering the *unknown* items from a given collection of items. It was first proposed to solve the blood test problem [11] as follows. Suppose in a city with a large population, some people are infected with a kind of disease (called "defective" members). A blood sample is drawn from each member. The problem is to identify *all* the defective members with minimum number of blood tests [11]. The simplest way is to test one by one, which is very time consuming in practice, motivating the *group test* model. The idea is that, divide the large population into different *groups*, and (all the blood samples of) each group is subjected to testing. If the outcome is negative, it means that all members in the group are free from infection which could save many unnecessary tests. Otherwise, each member making up the group needs to be further examined. Interesting readers are recommended to [12] for more information. In our problem, the solid edges between the black vertex set $\mathcal{B}$ and the white vertex set $\mathcal{W}$ are considered as "defective" members. This is because, a blood test returning 1 means that the subjected blood must contain at least one *defective* member; while in our example, a group test returning 1 means there exist at least one *solid* edge. As mentioned above, we may not need to find out all the solid edges (i.e., defective members) to solve the problem.

2.2.2 Group Test on Hidden Graph

On hidden graphs, the edges are regarded as the *unknown* items. The group test model is also applicable to solve the hidden graph related problems, e.g., [12, 4, 5, 3, 7] aim to detect whether a set of vertices induce any edges. As mentioned above, previous research on hidden graphs are mostly on graph testing and learning, which are different from us, regardless of whether the group test model is used.

Furthermore, previous research on the group test model assume that the cost of a group test is a constant [12, 4, 5, 3, 7, 13, 32], i.e., independent of the group size, which is true in the applications they concerned, e.g., biological experiments. Thus, the costs of those algorithms are analyzed based on the number of tests they invoked. However, the assumption may or may not hold in our data engineering applications, like Applications 1 and 3 in the introduction. Therefore, we consider the actual test costs with regard to the group size.

2.3 kMCV Problem on Hidden Bipartite Graph

Tao et. al [29, 28] is the first to study the applications of hidden graph in the data engineering domain. The algorithm SOE (Switch-on-Empty) in [29, 28] is built to solve the kMCV problem, upon the premise that each edge probe test can detect the presence of an edge between two vertices. Let $Q(b, w)$ be an edge probe test between a vertex $b \in \mathcal{B}$ and $w \in \mathcal{W}$. We get $Q(b, w) = 1$ if the test confirms that an edge (b, w) is present in the hidden graph $\mathcal{G}$, or get $Q(b, w) = 0$ if that edge is not present. The basic idea of SOE is that, it continues testing a black vertex b with an unseen white vertex w until $Q(b, w) = 0$ and then switches to examine another black vertex. The algorithm is executed iteratively until enough information is gathered to answer the kMCV problem. We use Figure 1.1 as an example to illustrate how the SOE algorithm works.

The degree of each vertex b, denoted by $deg(b)$, is initialized to the range $[0, |\mathcal{W}|]$ (which represents the minimum and maximum possible value of $deg(b)$). SOE operates iteratively and gradually tightens the range of $deg(b)$. A new iteration starts when all vertices in $\mathcal{B}$ have been visited once.

- Iteration 1. Probe b_1 with w_1, i.e., $Q(b_1, w_1)$. Since $Q(b_1, w_1) = 1$ then SOE continues testing b_1 with other unseen white vertices until the test returns 0. Hence, the following tests are $Q(b_1, w_2)$, $Q(b_1, w_3)$, $Q(b_1, w_4)$ and $Q(b_1, w_5)$. As $Q(b_1, w_5) = 0$ (an empty edge detected), SOE switches to probe b_2. In this example, since $Q(b_2, w_1) = 0$, the first iteration finishes, with the following information: (i) the degree of b_1 is at most $8 - 1 = 7$ and at least 4, i.e., $deg(b_1) \in [4, 7]$; (ii) the degree of b_2 is at most $8 - 1 = 7$ and at least 0, i.e., $deg(b_2) \in [0, 7]$.

- Iteration 2. A new iteration starts with b_1 again and continues the probing between b_1 and the other white vertices in $\mathcal{W}$. Therefore, the probe sequence in this iteration is: $Q(b_1, w_6) = 0, Q(b_2, w_2) = 0$. And this iteration finishes with the following new information: (i) $deg(b_1) \in [4, 6]$; (ii) $deg(b_2) \in [0, 6]$.

- Iteration 3. The probe sequence is: $Q(b_1, w_7) = 0, Q(b_2, w_3) = 0$. The degree information after this iteration is: (i) $deg(b_1) \in [4, 5]$; (ii) $deg(b_2) \in [0, 5]$.

- Iteration 4. The probe sequence is: $Q(b_1, w_8) = 0, Q(b_2, w_4) = 0$. The degree information after this iteration is: (i) $deg(b_1) \in [4, 4]$; (ii) $deg(b_2) \in [0, 4]$.

- Iteration 5. In this iteration, since vertex b_1 has probed all white vertices, it starts with $Q(b_2, w_5)$. Since $Q(b_2, w_5) = 0$, meaning that $deg(b_2) \in [0, 3]$. SOE grabs this chance to prune b_2 because the degree of b_1 (which is 4) is larger than the upper degree bound of b_2 (which is 3). At this point, SOE terminates.

Table 2.2 illustrates the detailed execution steps.

SOE is based on the "2-vertex" model to solve the kMCV problem. This work aims to further improve it by using the group test model. Comparing with

Table 2.2. Detailed execution steps of SOE on Figure 1.1

Iterations	Vertex	Test Sequence	Discovered Vertices	Degree Bound
0 (initialization)	b_1	-	-	[0,16]
	b_2	-	-	[0,16]
1	b_1	$Q(b_1, w_1) = 1$ $Q(b_1, w_2) = 1$ $Q(b_1, w_3) = 1$ $Q(b_1, w_4) = 1$ $Q(b_1, w_5) = 0$	w_1, w_2 w_3, w_4 w_5	[4,7]
	b_2	$Q(b_2, w_1) = 0$	w_1	[0,7]
2	b_1	$Q(b_1, w_6) = 0$	w_6	[4,6]
	b_2	$Q(b_2, w_2) = 0$	w_2	[0,6]
3	b_1	$Q(b_1, w_7) = 0$	w_7	[4,5]
	b_2	$Q(b_2, w_3) = 0$	w_3	[0,5]
4	b_1	$Q(b_1, w_8) = 0$	w_8	[4,4]
	b_2	$Q(b_2, w_4) = 0$	w_4	[0,4]
5	b_1	-	-	[4,4]
	b_2	$Q(b_2, w_5) = 0$	w_5	[0,3]

SOE [29, 28], the use of the group test model raises at least two new technical aspects:

1. In terms of algorithm design, a kMCV algorithm that exploits the group test model has to determine the group size carefully, in which algorithms that based on the 2-vertex model do not.

2. In terms of solution analysis, the analysis has to base on the external test cost, which depends on (i) the number of executed group tests, (ii) the group size, and (iii) the cost function of various group test implementations.

Chapter 3

Algorithm

In this chapter, we present our GMCV algorithm that solves the kMCV problem by the use of group test, which aims to reduce the external test cost. We first put down the relevant definitions.

Definition 5 (hidden vertex & hidden edge) *For a vertex pair (b, w) where $b \in \mathcal{B}$ and $w \in \mathcal{W}$, w is a* hidden vertex *of b if the connection between b and w in the hidden graph $\mathcal{G}$ is unknown. If w is a hidden vertex of b, then (b, w) is a* hidden edge*.*

Definition 6 (solid & empty vertex) *For a vertex pair (b, w) where $b \in \mathcal{B}$ and $w \in \mathcal{W}$, if $(b, w) \in \mathcal{E}$, then w is a* solid vertex *of b; otherwise w is an* empty vertex *of b.*

Definition 7 (completed) *A black vertex b is* completed *if it has no hidden edges.*

GMCV finds the top k black vertices with the highest degree in iterations. In each iteration, it examines the black vertices $b_1, b_2, \cdots, b_{|\mathcal{B}|}$ in $\mathcal{B}$ one-by-one. For a black vertex b_i, some group tests are carried out between it and some white vertices $W \subseteq \mathcal{W}$ in order to tighten the degree bounds of b_i, except when b_i is completed, or when b_i is deliberately *skipped* in that iteration because of the poor chance for b_i being in the final result (more on this later). After one iteration, another iteration starts and the black vertices $b_1, b_2, \cdots, b_{|\mathcal{B}|}$ in $\mathcal{B}$ are examined once again. Similar to most top k processing algorithms (e.g., [14, 20]), GMCV maintains the degree upper bound (denoted as $b_i.maxDeg$) and lower bound (denoted as $b_i.minDeg$) of each black vertex $b_i \in B$ throughout the execution and stops when the following condition holds:

Property 1 (Stop condition) *Let τ be the k-th largest degree in the result set R, and μ be the maximum degree upper bound of vertices not in R, GMCV can stop and return R when $\tau > \mu$.*

With the skeleton of GMCV in place, we study the following research issues:

R1 In an iteration, when a black vertex b_i is being examined by GMCV, how to leverage the group test model in order to refine b_i's degree bounds? Specific issues include (a) *how to determine the group of white vertices that should be tested with b_i?* and (b) *when shall GMCV stop examining b_i in this iteration and switch to another black vertex?*

R2 Black vertices with low degrees are unlikely to be in the top k result set R, thus, the question is: *how to avoid unnecessary test for low-degree vertices?*

3.1 Dealing with Research Issue R1

GMCV follows the "switch-on-empty" principle [29, 28] to deal with research issue R1(b). Within an iteration, it continues to work on b_i until a test returns "empty", i.e., $Q(b_i, W) = 0$, or b_i becomes completed. For a black vertex b_i, let W^{CUR} be the set of white vertices that b_i is going to carry out the group test with, and W^{PRE} be the previous set of white vertices that b_i carried out the group test with.

To deal with research issue R1(a), GMCV adaptively identifies W^{CUR} based on W^{PRE} and the two possible "states" associated with b_i: *expanding*, and *identifying*. Initially, the state of every $b_i \in \mathcal{B}$ is *expanding*, W^{PRE} is set to empty, and W^{CUR} is set to one random white vertex. For other cases (except initialization), W^{CUR} is determined as follows:

When b_i is in the *expanding* state, the objective of the group test between b_i and a set of white vertices is to reveal as many hidden vertices of b_i as possible.

- **[Case EXP-(a)]**: if $Q(b_i, W^{PRE}) = 0$, the number of white vertices that should be involved in the upcoming group test, denoted as $|W^{CUR}|$, is set as twice the size of $|W^{PRE}|$, i.e., $|W^{CUR}| = 2 \cdot |W^{PRE}|$. This is called the *doubling strategy*, which is commonly used in problems to dynamically adjust the value of some unknown parameters [6, 10][1]. The rationale is that, if $Q(b, W^{PRE}) = 0$, it implies b_i might have a low degree. Thus,

[1]In fact, other strategies such as multiplying the group size by 3 [12] or 4 [26] do exist. However, the literature does emphasis on the doubling strategy [12] because of its stableness. I.e., in our problem, it performs well no matter the degree is low or high.

GMCV can aim higher in this test—set b_i to test with a larger group of white vertices and hope that can reveal even more hidden vertices of b_i. The set W^{CUR} is then randomly chosen from b_i's hidden vertices.

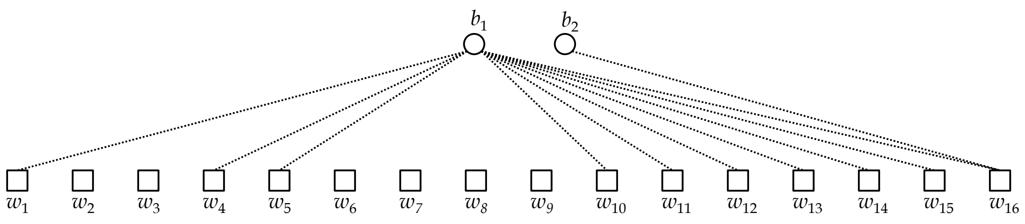
- **[Case EXP-(b)]**: if $Q(b_i, W^{PRE}) = 1$ and $|W^{PRE}| = 1$, it means b_i is a potentially high-degree vertex, so GMCV keeps $|W^{CUR}| = 1$.
- **[Case EXP-(c)]**: if $Q(b_i, W^{PRE}) = 1$ and $|W^{PRE}| > 1$, it implies that GMCV were too aggressive in the previous group test. In this case, b_i enters the *identifying* state.

When b_i is in the *identifying* state, the objective of the group test becomes to identify at least one of the solid vertices in W^{PRE} of b_i. Therefore,

- **[Case IDF-(a)]**: if $|W^{PRE}| > 1$ and $Q(b_i, W^{PRE}) = 1$, GMCV will devote some more tests to locate the white solid vertex in W^{PRE}. To do so, GMCV splits W^{PRE} into two halves: W_L^{PRE} and W_R^{PRE}, and sets W^{CUR} to be W_L^{PRE} and saves W_R^{PRE} as an *unexplored* set W^U.
- **[Case IDF-(b)]**: if $|W^{PRE}| = 1$ and $Q(b_i, W^{PRE}) = 1$, that means a white solid vertex of b_i in W^{PRE} has been identified; in this case, GMCV resets b_i's state back to the *expanding* state.
- **[Case IDF-(c)]**: if $Q(b_i, W^{PRE}) = 0$, GMCV explores the unexplored set by setting W^{CUR} to be W^U, but the test result of $Q(b_i, W^{CUR})$ is explicitly encoded as 1.

After identifying W^{CUR}, GMCV then executes such a group test $Q(b_i, W^{CUR})$. As mentioned, GMCV follows the switch-on-empty principle, so it may carry out a number of group tests, between b_i and a number of groups of white vertices, before it switches to another black vertex in the same iteration.

Figure 3.2 shows an example that illustrates some of the cases above. The corresponding input hidden graph is shown in Figure 3.1. In the first iteration, b_1 is first considered and $W^{CUR} = \{w_1\}$ (a random white vertex) (Iteration 1-a). After the first group test $Q(b_1, W^{CUR})$, it is found that w_1 is a solid vertex of b_1. This falls into [**Case EXP-(b)**] described above, resulting W^{CUR} is set to another random vertex w_2 (Iteration 1-b). After the next group test $Q(b_1, W^{CUR})$, it is found that w_2 is an empty vertex of b_1. So, GMCV follows the switch-on-empty principle and considers b_2 (Iteration 1-c). Since b_2 is first visited by GMCV, its W^{CUR} is set as $\{w_1\}$, like what happened to b_1. After the group test $Q(b_2, W^{CUR})$, it is found that w_1 is an empty vertex of b_2. Therefore, GMCV has to switch to another vertex, leading to Iteration 2, which considers b_1 again (Iteration 2-a). At that point, for b_1, $W^{PRE} = \{w_2\}$ (refer to Iteration 1-b), so, it falls into [**Case EXP-(a)**] described above, causing the size of W^{CUR} to be doubled (Iteration 2-a). After the group test $Q(b_1, W^{CUR})$, it is found that w_3, or w_4, or both, are solid vertices of b_1, so, it falls into [**Case EXP-(c)**] described above, b_1's state is thereby switched to *identifying* (Iteration 2-b). At that point, for b_1, $W^{PRE} = \{w_3, w_4\}$, so it falls into [**Case IDF-(a)**] described above, resulting W^{CUR} is set as $\{w_3\}$. After the group test $Q(b_1, W^{CUR})$, it is found that w_3 is an empty vertex of b_1 (which then also implies w_4 is a solid vertex of b_1), which triggers GMCV to switch to b_2 (Iteration 2-c). After the group test $Q(b_2, W^{CUR})$, it is found that both w_2 and w_3 are empty vertices of b_2, making GMCV switches to b_1 again (Iteration 3-a). By that time, although $Q(b_1, W^{CUR})$ supposes to test with w_4, it falls into the case of [**Case IDF-(c)**],



b_1
b_2
w_1 w_2 w_3 w_4 w_5 w_6 w_7 w_8 w_9 w_{10} w_{11} w_{12} w_{13} w_{14} w_{15} w_{16}

Figure 3.1. A hidden bipartite graph

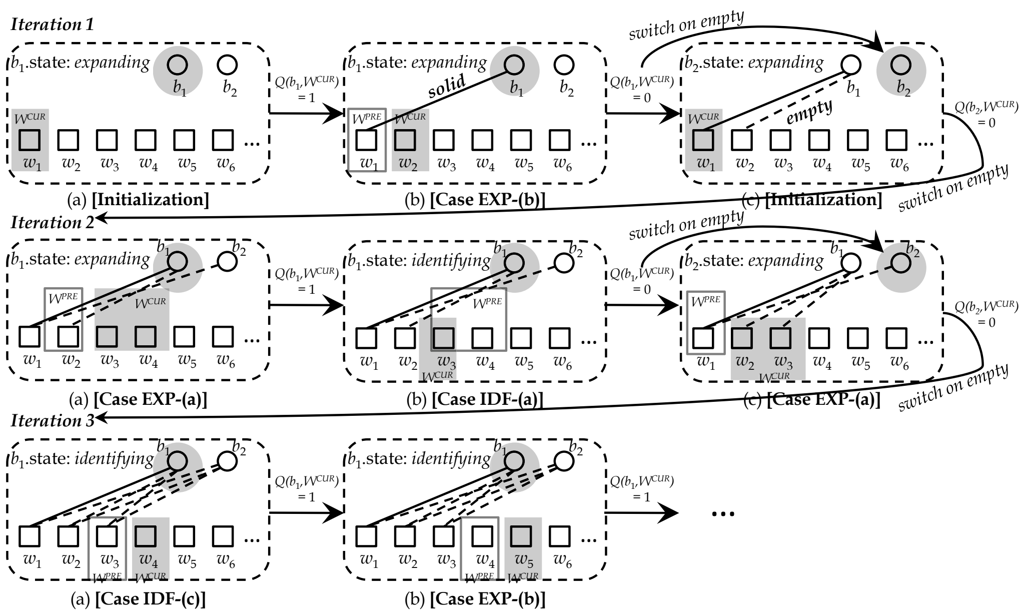


Figure 3.2. Running example

in which the test result is already encoded as 1 without even testing. So, after that, GMCV continues testing between b_1 and another white vertex w_5 (Iteration 3-b), and the process goes on until the stopping condition (Property 1) holds.

3.2 Dealing with Research Issue R2

For each black vertex $b_j \notin R$, the "necessary" tests are to reduce its degree upper bound, until below τ. In other words, it *should not* have any further tests once its degree upper bound below τ, as it is not part of the result set. However, the value of τ is unknown in advance, therefore, b_j may get redundant tests even if $b_j.maxDeg$ is really less than τ during the execution.

Thus, the question is, for any $b_j \notin R$ (i.e., low-degree vertex), how to prevent it from any further *unnecessary* tests even though τ is unknown beforehand? In other words, how to guarantee for any $b_j \notin R$, it does not have any unnecessary tests once $b_j.maxDeg < \tau$?

GMCV employs a *skipping policy* to achieve the goal. If $Q(b_j, W^{CUR}) = 0$, then, b_j is skipped for a *skip factor* of $|W^{CUR}| - 1$ iterations. E.g., if at iteration i, $Q(b, \{w_1, w_2, w_3\}) = 0$, then, GMCV skips b in the iterations $i+1$ and $i+2$. In Theorem 1, we will show that, with our skipping policy, vertices not in the result set do not have unnecessary tests. Then, we will show in Lemma 4 that the skip factor $|W^{CUR}| - 1$ is the optimal one among all the possible choices, so GMCV will use that as the skip factor. In the following, we first present the algorithm GMCV.

3.3 Algorithm: GMCV

The pseudo-code of GMCV is listed below. It is self-explanatory. It employs a skip factor of $|W^{CUR}| - 1$. Each black vertex b is associated with a field *skip*, which gets incremented whenever a group test has identified a group of b's empty vertices in a single group test, resulting in the skipping of processing b in a number of subsequent iterations.

Algorithm *GMCV*

```
     Input
     $\mathcal{G}(\mathcal{B},\mathcal{W})$: Hidden bipartite graph; $k$: User-threshold
     Output
     $R$: $k$ black vertices that have the maximum degree
1  $\tau$: the degree of the $k$-th ranked vertex in $R$
2  $\mu$: the maximum degree upper bound for those vertices not in $R$, i.e., $\max_{b \notin R} b.maxDeg$
3  $R$ is initialized to $k$ dummy vertices with degree $-1$
4  for each $b \in \mathcal{B}$ do
5      $b.minDeg \leftarrow 0$ /*degree lower bound*/
6      $b.maxDeg \leftarrow |\mathcal{W}|$ /*degree upper bound*/
7      $b.skip \leftarrow 0$ /*implement the skip policy*/
8  repeat
     /*start an iteration*/
9      for each $b \in \mathcal{B}$ do
10         if $b$ is completed then continue
11         if $b.skip > 0$ then /*skip policy*/
12             $b.skip \leftarrow b.skip - 1$
13             continue
14         find a group of white vertices $W^{CUR}$ to test /*Chapter 3.1*/
15         if $Q(b, W^{CUR}) = 0$ then /*external test*/
16             $b.maxDeg \leftarrow b.maxDeg - |W^{CUR}|$
```

```
17            $b.skip \leftarrow b.skip + (|W^{CUR}| - 1)$
18        **else**
19            **if** $|W^{CUR}| = 1$ **then**
20                $b.minDeg \leftarrow b.minDeg + 1$
21            **goto** line 10
22    let $C$ be the completed vertices in this iteration
23    $R \leftarrow R \cup C$
24    update $\tau$ /*$k$-th largest degree in $R$*/
25    $R \leftarrow \{b_i \in R : d_i \geq \tau\}$ /*update the result set $R$*/
26    update $\mu$ /*upper-bound score of vertices not in $R$*/
27 **until** $\mu < \tau$
```

Table 3.1 shows the detailed execution steps of GMCV in finding the 1MCV of the hidden graph presented in Figure 3.1. The *final* τ value is 10, which is the degree of b_1 but is unknown till the end of GMCV. After the fourth iteration, $b_2.maxDeg = 9$, which is below τ. Since then, b_2 is skipped for any further tests, until the end of GMCV.

Next, we show the correctness of our algorithm (regardless whether it applies skipping or not) in Lemma 1 and the choice for the skip factor. We show in Table 3.2 the purposes of the related theorems and lemmas.

Table 3.1. Detailed execution steps of GMCV on Figure 3.1

Iterations	Vertex	Test Sequence	Discovered Vertices	Skip	Degree Bound
0 (initialization)	b_1	-	-	0	[0,16]
	b_2	-	-	0	[0,16]
1	b_1	$Q(b_1, w_1) = 1$ $Q(b_1, w_2) = 0$	w_1, w_2	0	[1,15]
	b_2	$Q(b_2, w_1) = 0$	w_1	0	[0,15]
2	b_1	$Q(b_1, w_3w_4) = 1$ $Q(b_1, w_3) = 0$	w_3	0	[1,14]
	b_2	$Q(b_2, w_2w_3) = 0$	w_2, w_3	1	[0,13]
3	b_1	$Q(b_1, w_5) = 1$ $Q(b_1, w_6) = 0$	w_4, w_5, w_6	0	[3,13]
	b_2	-	-	0	[0,13]
4	b_1	$Q(b_1, w_7w_8) = 0$	w_7, w_8	1	[3,11]
	b_2	$Q(b_2, w_4w_5w_6w_7) = 0$	w_4, w_5 w_6, w_7	3	[0,9]
5	b_1	-	-	0	[3,11]
	b_2	-	-	2	[0,9]
6	b_1	$Q(b_1, w_9w_{10}w_{11}w_{12}) = 1$ $Q(b_1, w_9w_{10}) = 1$ $Q(b_1, w_9) = 0$	w_9	0	[3,10]
	b_2	-	-	1	[0,9]
7	b_1	$Q(b_1, w_{11}) = 1$ $Q(b_1, w_{12}) = 1$ $Q(b_1, w_{13}) = 1$ $Q(b_1, w_{14}) = 1$ $Q(b_1, w_{15}) = 1$ $Q(b_1, w_{16}) = 1$	w_{10}, w_{11} w_{12}, w_{13} w_{14}, w_{15}, w_{16}	0	[10,10]
	b_2	-	-	0	[0,9]

Table 3.2. Summary of the purposes of theorems and lemmas

Name	Purpose
Lemma 1	correctness of GMCV
Theorem 1	do not have unnecessary tests
Lemma 4	optimal skip factor

Lemma 1 *GMCV correctly reports the results, i.e., black vertices with top k maximum degrees.*

Proof. We show that (by Definition 4), R ($R \subseteq \mathcal{B}$) is a minimal set of vertices that satisfy (1) $|R| \geq k$; and (2) for for any $b_i \in R$ and $b_j \in \mathcal{B} \setminus R$, $d_i > d_j$.

First, we show that R satisfies the above two conditions. (1) $|R| \geq k$ is trivial as it already contains the vertex with the k-th largest degree τ; (2) The stopping condition $\mu < \tau$ (Property 1) guarantees that, for any vertices not in R will not have a higher degree than those in R.

Next, we show that R is a *minimal* set of vertices satisfying the two conditions. Let R' be any set of vertices satisfying the two conditions, we show that $R \subseteq R'$. Equivalently, for any black vertex b, if $b \in R$, then $b \in R'$. We prove it by contradiction. If $b \notin R'$, by the condition (2), the degree of b $deg(b)$ is smaller than any vertices in R', i.e., $deg(b) < \tau$. However, since $b \in R$, $deg(b) \geq \tau$, which is a contradiction. Thus, R is a minimal set of set of vertices satisfying the two conditions. □

THEOREM 1 *In GMCV, a black vertex $b_j \notin R$ stops any further tests, once its degree upper bound is just smaller than the final τ.*

Proof. The statement is equivalent to, any black vertex $b_j \notin R$ stops for any further tests *once* the number of empty vertices it has detected is greater than or equal to $|\mathcal{W}| - (\tau - 1)$. Let $\theta = |\mathcal{W}| - (\tau - 1)$.

Formally, let $\mathcal{E}_{b_j}$ be the number of empty vertices detected with b_j during GMCV, then $\mathcal{E}_{b_j}$ is increasing during the execution of the algorithm. Let $\mathcal{E}^m_{b_j}$ be the value of $\mathcal{E}_{b_j}$ after the m-th *change* of $\mathcal{E}_{b_j}$. (Thus, $\mathcal{E}^m_{b_j} \leq \mathcal{E}^{m+1}_{b_j}$). Let $\mathcal{E}^z_{b_j}$ be the value of $\mathcal{E}_{b_j}$ of the last change of $\mathcal{E}_{b_j}$ before GMCV terminates, we have (I) $\mathcal{E}^z_{b_j} \geq \theta$ and (II) $\mathcal{E}^{z-1}_{b_j} < \theta$.

We prove (I) by contradiction. At the end of GMCV, if $\mathcal{E}_{b_j} < \theta$ (i.e., $\mathcal{E}^z_{b_j} < \theta$), we have $b_j.maxDeg = |\mathcal{W}| - \mathcal{E}^z_{b_j} > |\mathcal{W}| - \theta = \tau - 1$. In order words, $b_j.maxDeg \geq \tau$. According to the stop condition of GMCV (Property 1), $\mu < \tau$, where μ is the the maximum degree upper bound of vertices not in R, meaning that $b_j.maxDeg < \tau$, which is a contradiction.

Next, we will prove (II) $\mathcal{E}^{z-1}_{b_j} < \theta$ by contradiction. Let us assume

$$\mathcal{E}^{z-1}_{b_j} \geq \theta \tag{3.1}$$

We state the supplementary Lemmas 2 and 3, which are proved in the appendix.

Lemma 2 *Let the* $(z-1)$*-th change of* $\mathcal{E}_{b_j}$ *value occurs at the end of iteration-*$\mathcal{I}$ *of GMCV, if* $b_j.skip = 0$*, then iteration-*$\mathcal{I}$ *is the last iteration of GMCV.*

Lemma 3 *Let the* $(z-1)$*-th change of* $\mathcal{E}_{b_j}$ *value occurs at the end of iteration-*$\mathcal{I}$ *of GMCV, if* $b_j.skip > 0$*, then at the end of the iteration-*$(\mathcal{I} + b_j.skip)$*, GMCV must have terminated.*

With Lemma 2 proven, it implies that the $(z-1)$-th change of $\mathcal{E}_{b_j}$ is the last change of $\mathcal{E}_{b_j}$, which contradicts the fact that $\mathcal{E}^z_{b_j}$ is the last change of $\mathcal{E}_{b_j}$.

With Lemma 3 proven, and together with the fact that the value of $\mathcal{E}_{b_j}$ does not change between iteration-$\mathcal{I}$ and iteration-$(\mathcal{I} + b_j.skip)$ (because by that time

$b_j.skip > 0$ and thus b_j is skipped), so the value of $\mathcal{E}_{b_j}$ at iteration-($\mathcal{I}+b_j.skip$) is equal to the value of $\mathcal{E}_{b_j}$ at the end of the iteration-$\mathcal{I}$, which is equal to $\mathcal{E}_{b_j}^{z-1}$. So, if we can prove that GMCV has terminated by that time, it implies that $\mathcal{E}_{b_j}^{z-1}$ is the value of $\mathcal{E}_{b_j}$ before GMCV terminates, which contradicts the fact that $\mathcal{E}_{b_j}^{z}$ is the last change of $\mathcal{E}_{b_j}$.

With Lemmas 2 and 3 proven, we can conclude that the assumption $\mathcal{E}_{b_j}^{z-1} \geq \theta$ (3.1) is false and the proof is completed. □

Lemma 4 *Setting the skip factor to be* $|W^{\text{CUR}}| - 1$ *is optimal in GMCV.*

Proof. Each $b \in \mathcal{B}$ has a sequence of group tests and stops when the stopping condition (Property 1) is met. Obviously, for a b_i in the final result set R, its whole sequence of group tests must be carried out. So, we care about only those b_j not in the final result set R. For $b_j \notin R$, its degree upper bound, denoted as $b_j.maxDeg$, gets reduced along the iterations when more tests are done.

Its corresponding aggregated external test cost is the minimum if its test sequence stops once $b_j.maxDeg$ is just smaller than τ. We denote that cost as $minC^{b_j}$, which is proved in Theorem 1. □

Chapter 4

Cost Model

Although SOE is proven to be instance-optimal (i.e., for any given problem instance, it incurs at most a constant factor of tests of the optimal solution), it is not applicable to the context with group test. In SOE, minimizing the number of tests is equivalent to minimizing the total external test cost because the external cost of a 2-vertex test function is a constant. However, the overall external cost of a group test function depends not only on the number of tests invoked, but also on the input size to each test as well as the *implementation of the group test*.

In this Chapter, we provide cost models to capture the total external test costs of GMCV (Chapter 4.1) and SOE (Chapter 4.2), and compare their external costs based on different group test cost functions (Chapter 4.3). For every black vertex b_i, we assume that its degree $d_i \neq 0$ and $d_i \neq |\mathcal{W}|$, as it is trivial to deal with these two cases.

4.1 External Test Cost of GMCV

In an execution of GMCV, a particular black vertex $b_i \in R$ is associated with a series of *expanding*-and-*identifying* processes that may span across multiple iterations. Initially, a test $Q(b_i, W_i^1)$ is carried out. If $Q(b_i, W_i^1) = 0$, another group test $Q(b_i, W_i^2)$ is carried out. The expanding phase $Q(b_i, W_i^1) = 0, Q(b_i, W_i^2) = 0, \cdots$, continues until the s-th test in which $Q(b_i, W_i^s) = 1$ (while all the previous tests return 0), where s is called the *turning point* in the process. After that, the identifying phase starts: $Q(b_i, W_i^{s+1}), \cdots, Q(b_i, W_i^{2s-1})$, i.e., recursively drill into the set W_i^s to locate the solid vertex.

Lemma 5 *Let $\mathcal{C}_i^j$ be the external test cost of the j-th expanding-and-identifying process of b_i, and s be the turning point, then $\mathcal{C}_i^j = 2\sum_{j=1}^{s-1} \beta(2^{j-1}) + \beta(2^{s-1})$.*

Proof. Note that the size of the vertex set W_i^j has the following property

$$|W_i^j| = \begin{cases} 2^{j-1} \ , 1 \leq j \leq s \\ 2^{2s-j-1} \ , s < j \leq 2s-1 \end{cases}$$

As $\mathcal{C}_i^j$ denotes the external test cost of the j-th expanding-and-identifying process of b_i, then $\mathcal{C}_i^j = \sum_{j=1}^{2s-1} \beta(|W_i^j|) = 2\sum_{j=1}^{s-1} \beta(2^{j-1}) + \beta(2^{s-1})$. □

Lemma 6 *For $b_i \in R$, the total external test cost $Cost(b_i)$ associated with b_i is:*

$$Cost(b_i) = d_i \cdot \mathcal{C}_i^j$$

where d_i is the degree of b_i.

Proof. Lemma 5 gives the external test cost of any expanding-and-identifying process. Since in GMCV, every black vertex b_i in R is completed, i.e., it has the exact degree d_i, and each expanding-and-identifying process locates one solid vertex, the cost of $b_i \in R$ is thus $d_i \cdot \mathcal{C}_i^j$. □

Next, we discussion about how to set the turning point s in an expanding-and-identifying process $\mathcal{C}_i^j$. An expanding-and-identifying process reveals 1 solid vertex plus at least $\sum_{j=1}^{s-1} 2^{j-1} = 2^{s-1} - 1$ empty vertices, a total of at least 2^{s-1} vertices. Let $\omega = 2^{s-1}$. Since GMCV algorithm *randomly* picks white vertices to carry out the group test on b_i, ω can be approximated as $|\mathcal{W}|/d_i$. So, we have $s - 1 = \lfloor \lg \frac{|\mathcal{W}|}{d_i} \rceil$, i.e., $s = \lfloor \lg \frac{|\mathcal{W}|}{d_i} \rceil + 1$. We use randomized rounding here because s is an integer.

Next, we derive $Cost(b_j)$, the external test cost associated with a vertex $b_j \notin R$. Before that, we define $A(t)$ be the accumulated external test cost in order to identify t empty vertices through a series of group tests whose results are all zero (i.e., the external test costs spent on the doubling strategy during the expanding phase). It is thus trivial to see that $A(t) = \sum_{j=0}^{\lfloor \lg t \rfloor} \beta(2^j)$.

Lemma 7 *For $b_j \notin R$, the external test cost $Cost(b_j)$ associated with b_j is:*

$$Cost(b_j) = \lambda_j \cdot \mathcal{C}_i^j + A(\theta - \lambda_j \cdot (2^s - 1))$$

where $\theta = |\mathcal{W}| - \tau + 1$, $\lambda_j = \lfloor \frac{\theta}{\frac{|\mathcal{W}|}{d_j} - 1} \rfloor$, and d_j is the degree of b_j.

Proof. In Theorem 1, we show that a black vertex $b_j \notin R$ does not need any further test in GMCV, once its degree upper bound is smaller than τ. Meaning that b_i needs to detect $|\mathcal{W}|-(\tau-1)$ empty vertices. Let $\theta = |\mathcal{W}|-(\tau-1)$. Next, the analysis is redirected to analyze the external test cost of detecting θ empty vertices for $b_j \notin R$.

As mentioned, an expanding-and-identifying process discovers 1 solid vertex plus at least $2^{s-1}-1$ empty vertices, where $s = \lfloor \lg \frac{|\mathcal{W}|}{d_i} \rceil + 1$. Thus, in order to detect θ empty vertices, it requires $\lfloor \frac{\theta}{2^{s-1}-1} \rfloor = \lfloor \frac{\theta}{\frac{|\mathcal{W}|}{d_i}-1} \rfloor$ (denoted as λ) expanding-and-identifying processes.

For the remaining $\theta - \lambda \cdot (2^s - 1)$ empty vertices, it requires a follow-up expanding phrase, which costs $\mathcal{A}(\theta - \lambda \cdot (2^s - 1))$.

Summing up the external test cost gives the result, which completes the proof. □

THEOREM 2 *The external test cost of GMCV is:*

$$Cost_{GMCV} = \sum_{b_i \in R} Cost(b_i) + \sum_{b_j \notin R} Cost(b_j) \tag{4.1}$$

where $Cost(b_i)$ and $Cost(b_j)$ are defined in Lemma 6 and Lemma 7 respectively.

4.2 External Test Cost of SOE

According to [29, 28], the number of tests $\mathcal{N}_{SOE}$ consumed by SOE for a hidden partite graph with $|\mathcal{B}|$ black vertices and $|\mathcal{W}|$ white vertices is

$$\begin{aligned}\mathcal{N}_{SOE} &= |R| \cdot |\mathcal{W}| + \sum_{i=|R|+1}^{|\mathcal{B}|} \frac{(|\mathcal{W}| - \tau + 1)(|\mathcal{W}| + 1)}{l_i \cdot |\mathcal{W}| + 1} \\ &= |R| \cdot |\mathcal{W}| + \sum_{i=|R|+1}^{|\mathcal{B}|} \frac{\theta(|\mathcal{W}| + 1)}{|\mathcal{W}| - d_i + 1}\end{aligned} \tag{4.2}$$

where $l_i = 1 - \frac{d_i}{|\mathcal{W}|}$.

Since each 2-vertex test has the cost of $\beta(1)$, the external test cost of SOE is:

$$Cost_{SOE} = \mathcal{N}_{SOE} \cdot \beta(1) \tag{4.3}$$

4.3 Cost Comparison

We compare the external test costs of GMCV and SOE based on the cost models established in Equations (4.1) and (4.3). Following [29, 28], we assume the degrees of the bipartite graph follow power-law distribution such that for each $b \in \mathcal{B}$, its degree equals d (between 0 and $|\mathcal{W}|$) has the probability:

$$Pr(d) = \frac{1/(d+1)^{\gamma}}{\sum_{i=0}^{|\mathcal{W}|} 1/(i+1)^{\gamma}} \tag{4.4}$$

where γ is the skewness factor to control the sparseness of a graph ($\gamma > 0$). The smaller the γ is, the denser the graph is.

We consider four group test implementations:

(I) Const, where $\beta(|W|) = \beta(1)$

(II) Log, where $\beta(|W|) = \lg |W| \cdot \beta(1)$

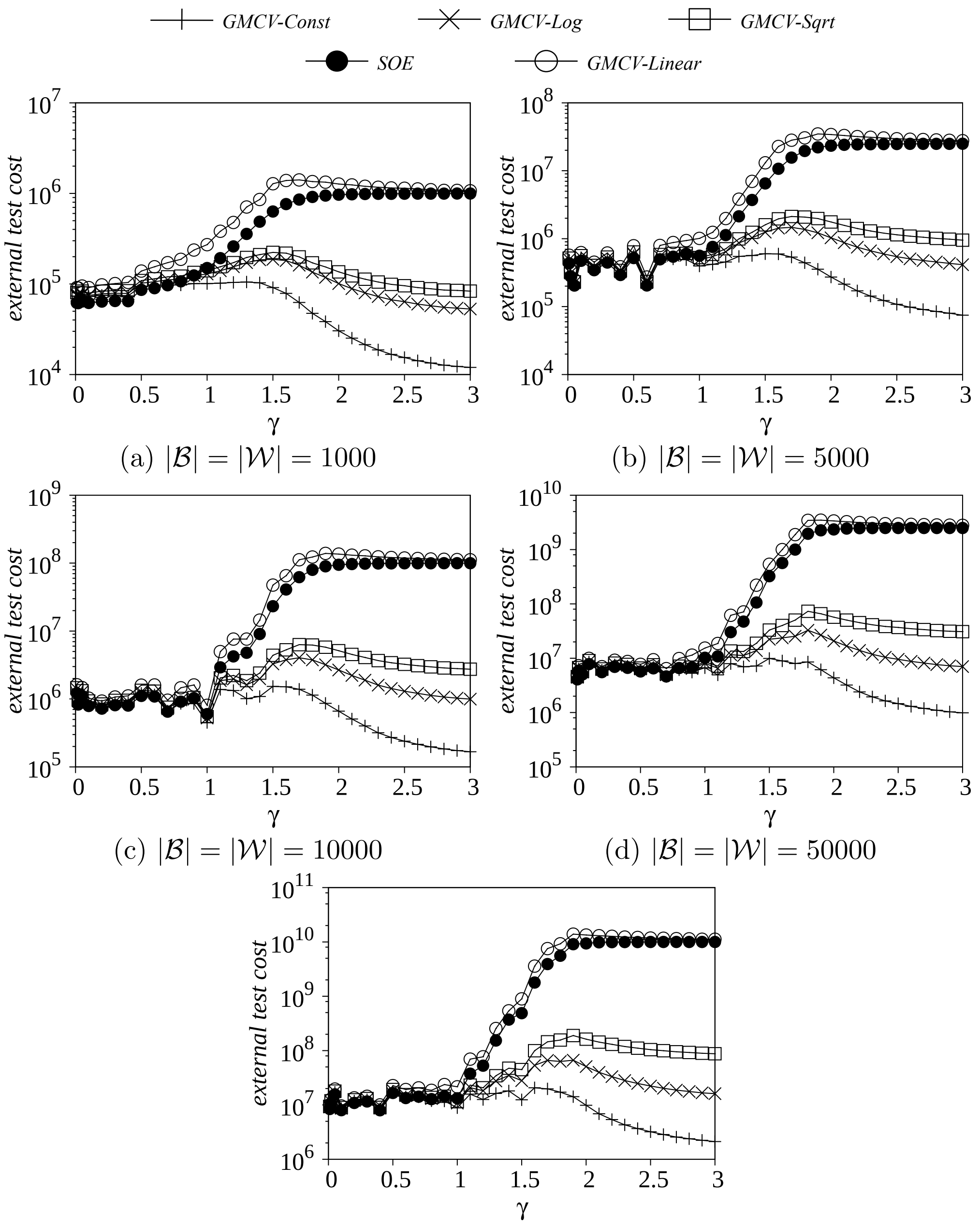
(III) Sqrt, where $\beta(|W|) = \sqrt{|W|} \cdot \beta(1)$

(IV) Linear, where $\beta(|W|) = |W| \cdot \beta(1)$

The Const implementation is to simulate the group test implementation in the biological domain, in which both the monetary cost and the running time of an experiment is a constant [17]. The Log and the Sqrt implementations are to simulate the group test implementations in the graph pattern matching and distance join applications, where the external cost (running time) is sub-linear to the input size. Applications for the Linear group test implementation are not clear; however, we include it in our study to show that GMCV should not be misused in applications where the external cost of a group test is (super) linear to its input size.

Figure 4.1 plots the external test costs of GMCV and SOE ($k = 10$) based on Equations (4.1) and (4.3), on hidden partite graphs of varying sizes ($|\mathcal{B}| = |\mathcal{W}|$) and different sparseness γ. It can be seen that GMCV outperforms SOE in almost all graph sizes and graph sparseness, except when the graphs are unusually dense (γ is close to 0)[1] or when GMCV is deliberately misused on applications where the external cost of a group test is (super) linear to the size of the input. In those cases, we found GMCV and SOE have comparable performance.

[1]Normally, γ is larger than 2.0 in real graphs [1, 2].



(a) $|\mathcal{B}| = |\mathcal{W}| = 1000$

(b) $|\mathcal{B}| = |\mathcal{W}| = 5000$

(c) $|\mathcal{B}| = |\mathcal{W}| = 10000$

(d) $|\mathcal{B}| = |\mathcal{W}| = 50000$

(e) $|\mathcal{B}| = |\mathcal{W}| = 100000$

Figure 4.1. Derived costs of SOE and GMCV

Chapter 5

Experiments

In this chapter, we evaluate GMCV on both real life datasets and synthetic datasets.

PPI[1]. It consists of the interactions between Yeast proteins, where $\mathcal{B}$ and $\mathcal{W}$ represent all the proteins. Particularly, a protein $b \in \mathcal{B}$ connects with $w \in \mathcal{W}$ if they can interact with each other.

Germany[2]. It is a real road network from Germany. In our problem setting, $\mathcal{B}$ and $\mathcal{W}$ contains all the nodes. A vertex $b \in \mathcal{B}$ and a vertex $w \in \mathcal{W}$ has an edge if their distance (in terms of the shortest path distance) is less than a predefined threshold, which is set to 10km by default.

Actor-W[3]. It is an actor collaboration network data based on IMDB[4]. In which, $\mathcal{B}$ and $\mathcal{W}$ include all the actors. In particular, two actors b and w have

[1] http://turing.cs.iastate.edu/PredDNA/dataset.html

[2] www.maproom.psu.edu/dcw

[3] http://www.datatang.com/DataRes/Detail.aspx?id=1624

[4] http://www.imdb.com

an edge if they have co-appeared in at least one movie.

Actor-D, available from [29, 28]. It is derived from the actor collaboration social network data by extracting 10,000 actors that have the largest number of collaborators, i.e., $\mathcal{B}$ and $\mathcal{W}$. Two actors b and w have an edge if they have 2-hop relationship, i.e., either they appeared in at least one common movie, or they have a common collaborator.

Table 5.1 summarizes the properties of the four real datasets above. Actor-D is unusually dense—in a hidden graph with only 10,000 black and 10,000 white vertices, a black vertex connects to more than 7,000 white vertices on average. In fact, Actor-D does not follow power-law distribution as its $\gamma < 0$. Furthermore, for a better understanding of the datasets used, we plot the degree distributions in Figure 5.1, Figure 5.2, Figure 5.3 and Figure 5.4.

Table 5.1. Statistics of real graphs

dataset	# black vertices	# white vertices	# edges	avg. deg.	raw data size
PPI	2,617	2,617	11,855	4.53	68KB
Germany	28,867	28,867	30,429	1.05	113KB
Actor-W	392,340	392,340	29,088,772	74.14	189MB
Actor-D	10,000	10,000	73,801,472	7,380	352MB

Synthetic Data. We follow [29, 28] to generate graphs of different sizes and sparseness. By default, $|\mathcal{B}| = |\mathcal{W}| = 5,000$.

Following [29, 28], we simulate the implementation of a (group) test. We use the four group test functions Const, Log, Sqrt, and Linear mentioned in Chapter 4.3. For example, we regard the external cost of a group test with an input of 4 vertices is 2, if the Sqrt group test function is used. The experimental results are reported in terms of external test cost.

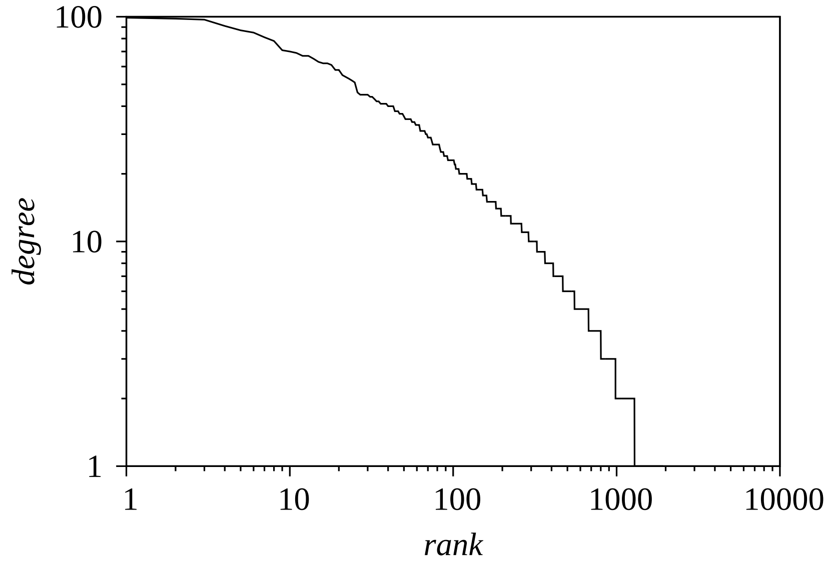


Figure 5.1. Degree distribution for PPI data

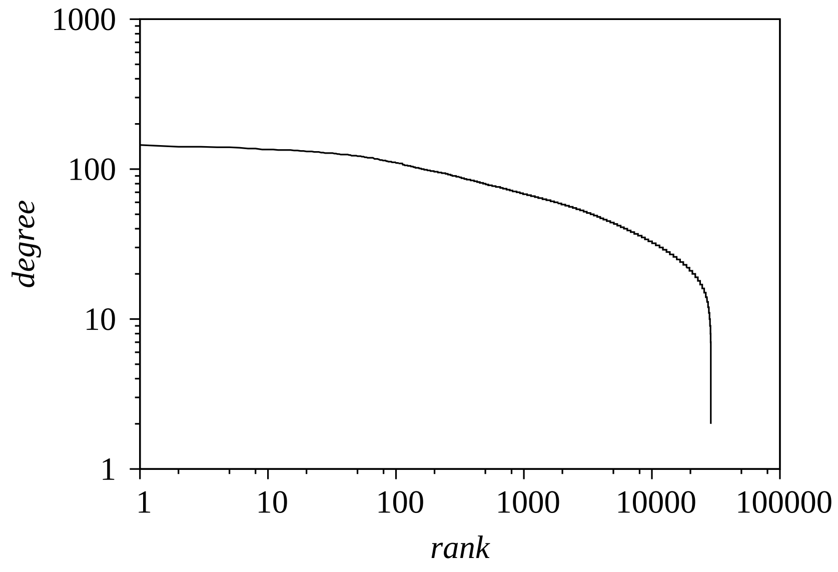


Figure 5.2. Degree distribution for Germany data

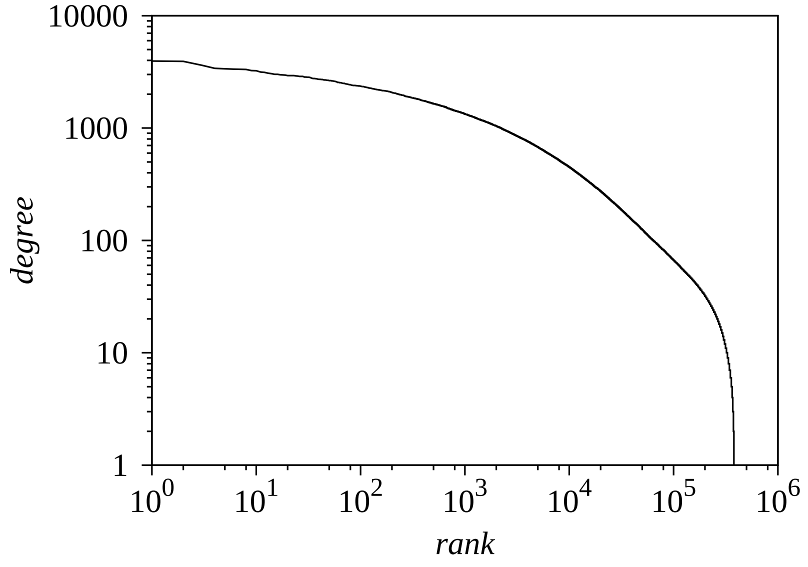


Figure 5.3. Degree distribution for Actor-W data

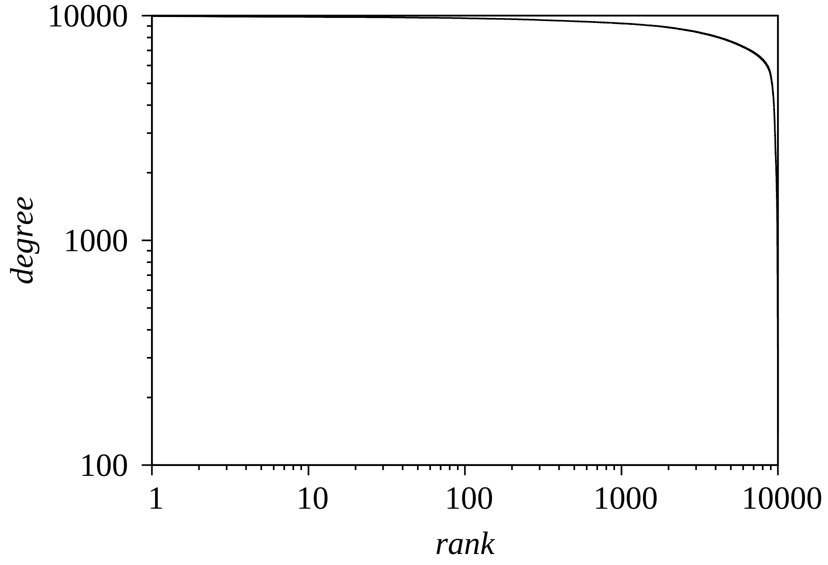


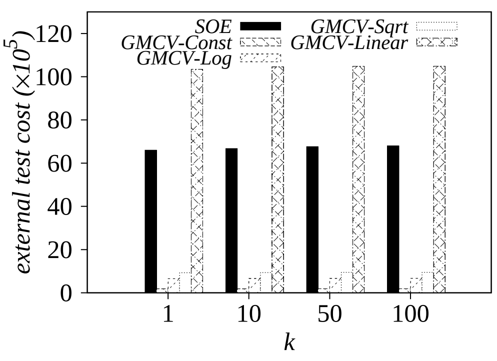
Figure 5.4. Degree distribution for Actor-D data

5.1 Experimental Results on Real Datasets

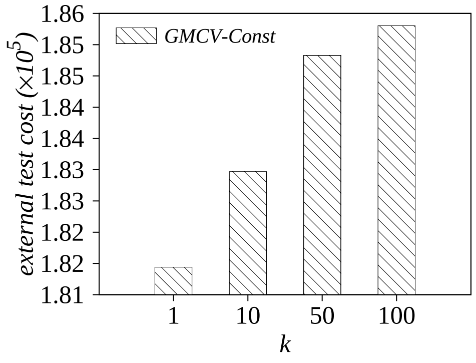
Figure 5.5 shows the external test costs of GMCV (based on different group test cost functions) and SOE of different k values, on the PPI dataset. It is clear that, GMCV outperforms SOE significantly, except when the inappropriate Linear group test is deliberately used. Specifically, the costs of GMCV are 36 times (Const), 10 times (Log), and 7 times (Sqrt) less than SOE, respectively. Since their costs differ so much and we cannot see the effect of k when putting them together in one graph, so we plot their individual costs as well (smaller graphs). We can observe that all methods scale well with the value of k.

The experimental results on Germany and Actor-W datasets are are shown in Figure 5.6 and Figure 5.7. We can also observe that GMCV outperforms SOE significantly, again except when the Linear group test function is deliberately used.

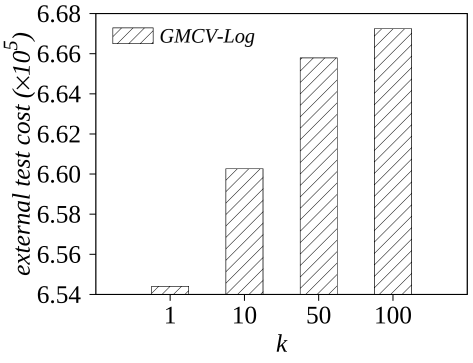
Figure 5.8 shows the external test costs of GMCV and SOE on Actor-D. We can see that, even on such an unusually dense dataset, SOE and GMCV have comparable performance. This is because, GMCV uses the doubling strategy to adaptively determine the group size based on the outcome of the previous test, i.e., double the group size if the previous test result is 0 and halve the group size otherwise. On dense graphs, however, a group test has a high chance to return 1. Therefore, GMCV seldom employs the doubling strategy, which makes GMCV behave like SOE, but with a little overhead.



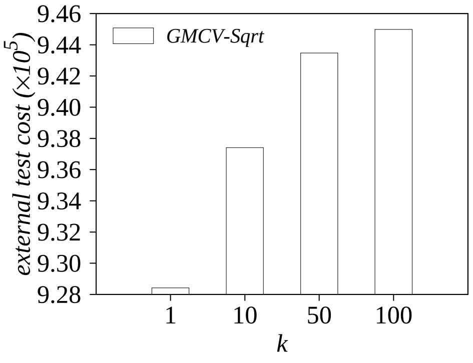
(a) All



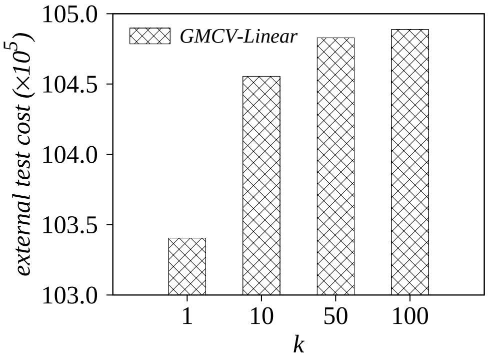
(b) GMCV-Const



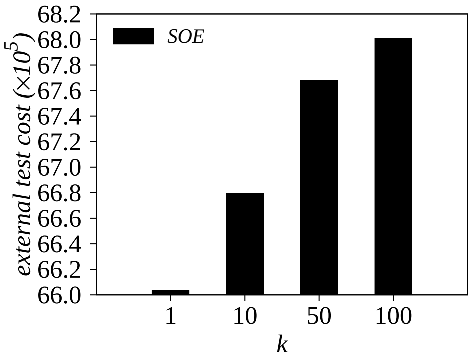
(c) GMCV-Log



(d) GMCV-Sqrt

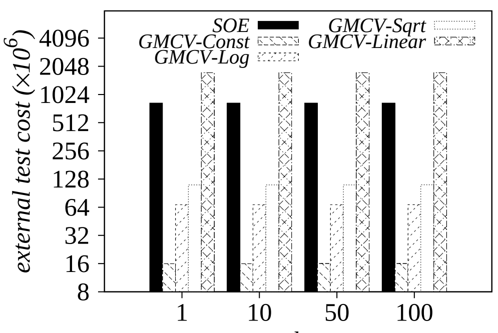


(e) GMCV-Linear

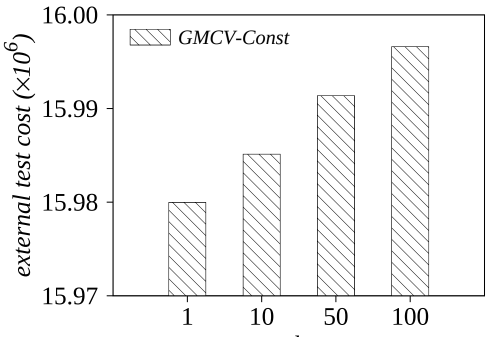


(f) GMCV-SOE

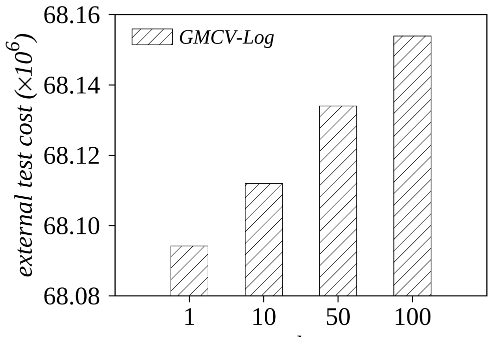
Figure 5.5. Results on PPI data, varying k



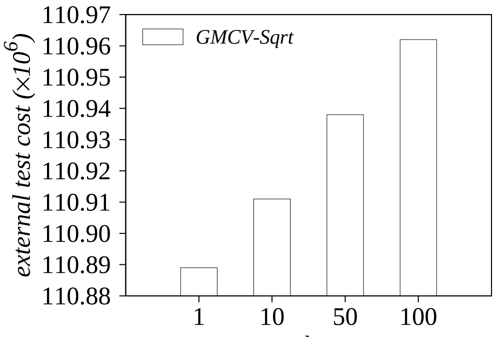
(a) All



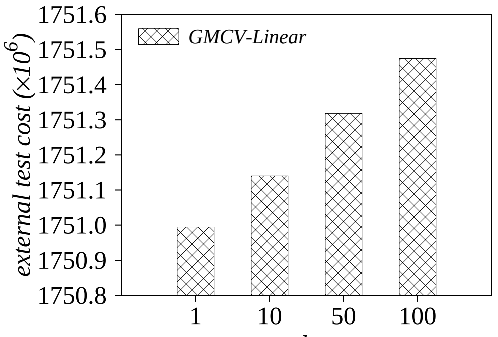
(b) GMCV-Const



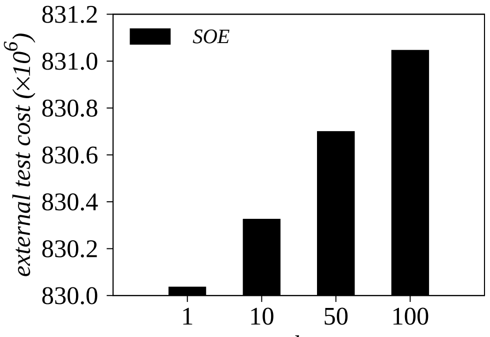
(c) GMCV-Log



(d) GMCV-Sqrt

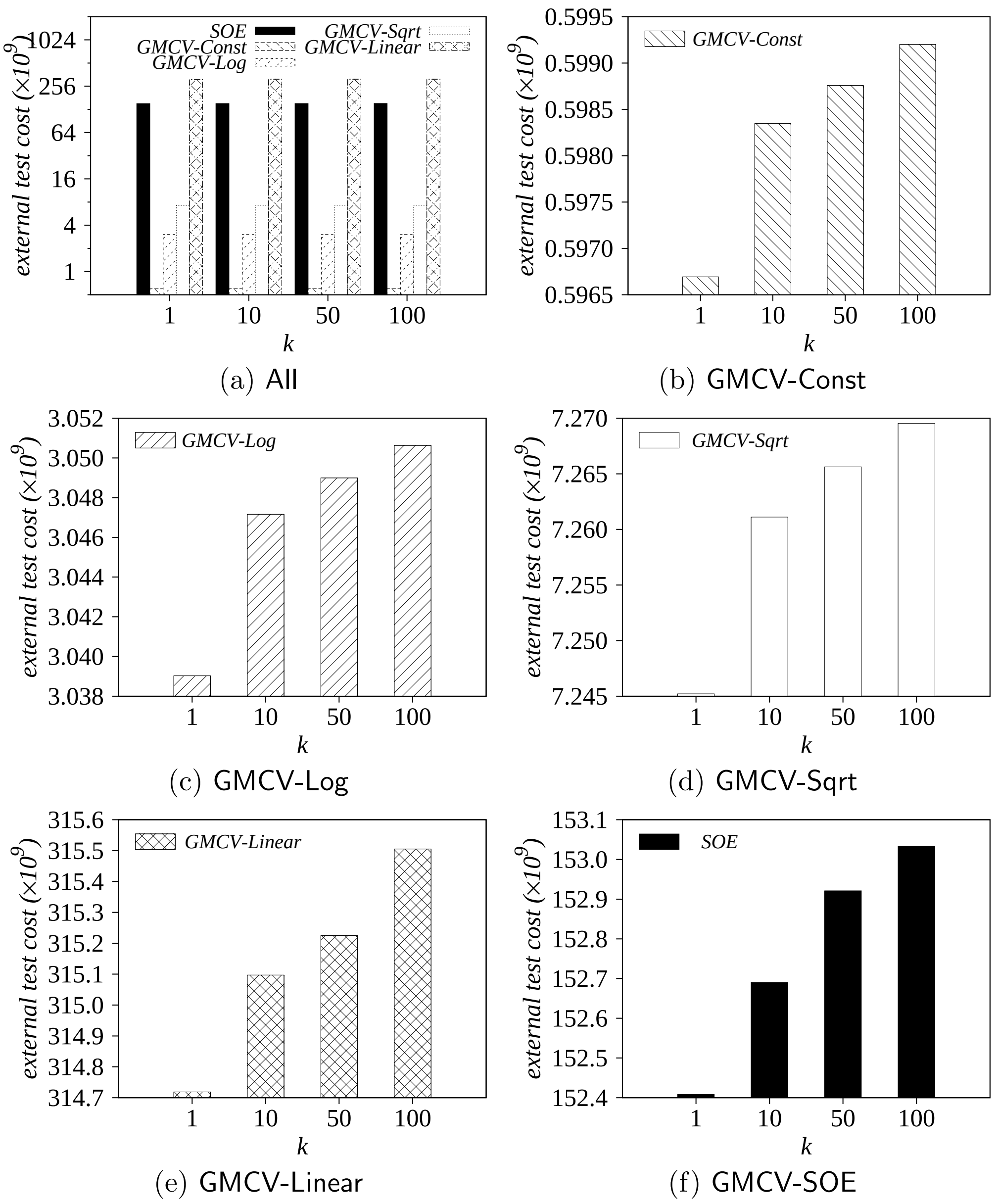


(e) GMCV-Linear



(f) GMCV-SOE

Figure 5.6. Results on Germany data, varying k



(a) All (b) GMCV-Const

(c) GMCV-Log (d) GMCV-Sqrt

(e) GMCV-Linear (f) GMCV-SOE

Figure 5.7. Results on Actor-W data, varying k

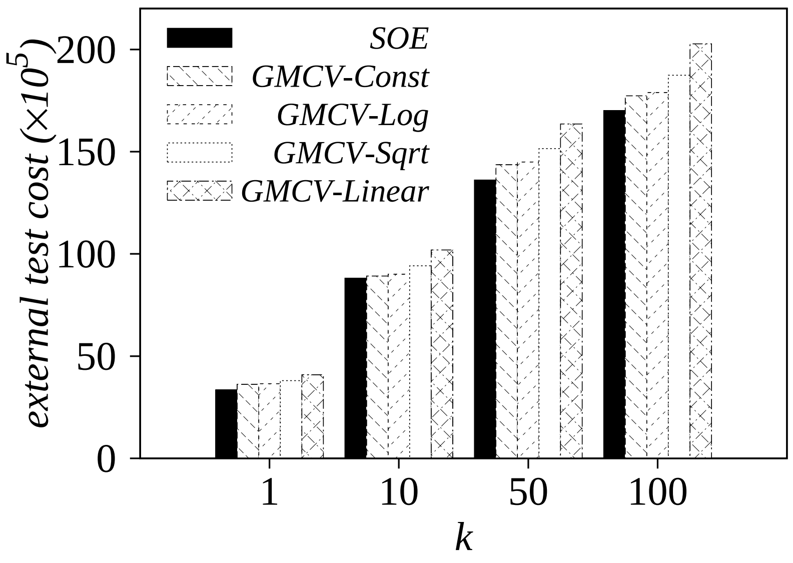


Figure 5.8. Results on Actor-D data

5.2 Experimental Results on Synthetic Datasets

5.2.1 Sparseness

Figure 5.9 shows the external test costs of GMCV and SOE running on synthetic graphs of different sparseness. The skewness factor γ ranges from 0.1 (average degree is 2,389) to 4.0 (average degree is 0.108). We can see that GMCV outperforms SOE from sparse to dense graphs, except when the improper Linear group test function is deliberately used. SOE is comparable with GMCV only when the graph is extremely dense ($\gamma = 0.1$).

5.2.2 Scalability

In this experiment, we evaluate the scalability of GMCV on synthetical graphs of different sizes (from 5,000 black vertices and 5,000 white vertices to

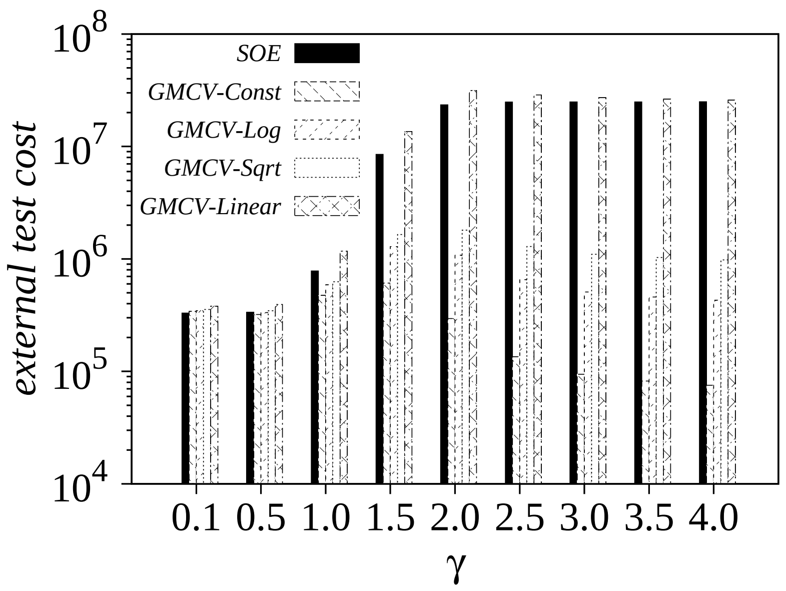


Figure 5.9. Varying graph sparseness with $k = 10$

500,000 black vertices and 500,000 white vertices). The graphs here are generated using $\gamma = 2.0$, which is found in many real life graph data [1, 2]. Figure 5.10 shows the external test costs of GMCV running on synthetic graphs of different sizes. We can see that GMCV scales well on graphs of different sizes.

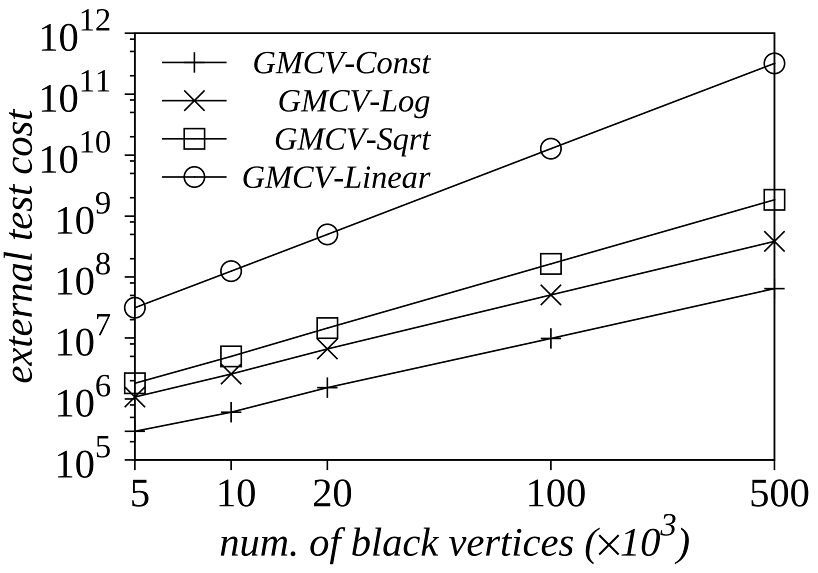


Figure 5.10. Varying num. of black vertices $|\mathcal{B}|$, with $|\mathcal{B}| = |\mathcal{W}|$ and $k = 10$

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This work studies the kMCV (k most connected vertices) problem on hidden bipartite graphs in the context of the group test model. Group test is a common test model in hidden graph literature. Instead of testing the presence of edge between only two vertices (which is called the 2-vertex test model), a group test takes as input a group of vertices and returns whether there is any edge among them. If the group test model is used properly, a single group test can reveal the same information as multiple 2-vertex tests. Therefore, if the external cost of a group test is constant to or sub-linear of the input size, the external cost of solving an kMCV problem can be significantly reduced.

To that end, an algorithm that based on group test, called, GMCV, is developed. GMCV adaptively determines the size of the vertices to be input to each group test based on the data characteristics. Our cost analysis as well as experimental results show that GMCV outperforms SOE, a 2-vertex test based kMCV algorithm, except in some extreme cases (e.g., when the linear implementation of group test is deliberately used or the graphs are unusually dense). In those cases, GMCV still has comparable performance with SOE, making GMCV a robust and more effective choice than SOE in the usual settings.

6.2 Future Work

Next, we present some possible research directions on this topic.

6.2.1 Generalized Group Testing Model for kMCV

In this work, each group test takes as inputs *one* black vertex and multiple white vertices. While, an even more general group testing model can test *multiple* (instead of *one*) black vertices and multiple white vertices within one group test.

Formally, let B ($B \subseteq \mathcal{B}$) and W ($W \subseteq \mathcal{W}$) be a group of black and white vertices, a test $Q(B, W)$ can reveal whether there exist some edge between B and W:

$$Q(B,W) = \begin{cases} 1 \text{ , if } \exists b \in B, \exists w \in W; (b,w) \in \mathcal{E} \\ 0 \text{ , if } \forall b \in B, \forall w \in W; (b,w) \notin \mathcal{E} \end{cases}$$

The generalized group testing model is even more powerful, however, it adds new challenges in solving the kMCV problem *efficiently.* Of course, one can simply choose $|B| = 1$ every time, which is exactly the situation considered in our work.

We present some preliminary ideas via building a *search tree* T to solve the problem with the generalized group testing model, assuming $|\mathcal{B}| \leq |\mathcal{W}|$. Next, we describe how to construct of the search tree T.

Each node in the tree represents a group test. The root of the tree T is the group test $Q(\mathcal{B}, \mathcal{W})$, i.e., it tests the entire black and white vertices. If the test result of a node say $Q(B, W)$ returns 1, split it into two child nodes as follows. Suppose $|B| \leq |W|$, split W into two halves $W1$ and $W2$ (otherwise, halve B). Then, the two child nodes are $Q(B, W1)$ and $Q(B, W2)$. A node is a leaf node is the test result returns 0 or the size of both B and W are 1.

After that, we can execute the tests by traversing the search tree in depth search manner or breadth search manner.

We use an example to illustrate it. Figure 6.2 is the corresponding search tree built for Figure 6.1.

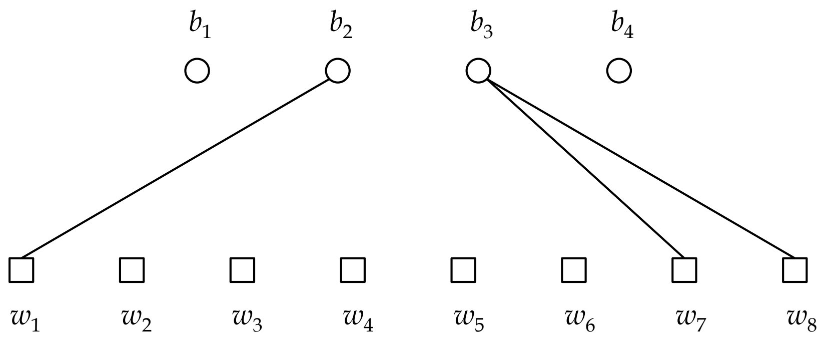


Figure 6.1. Example of a hidden graph

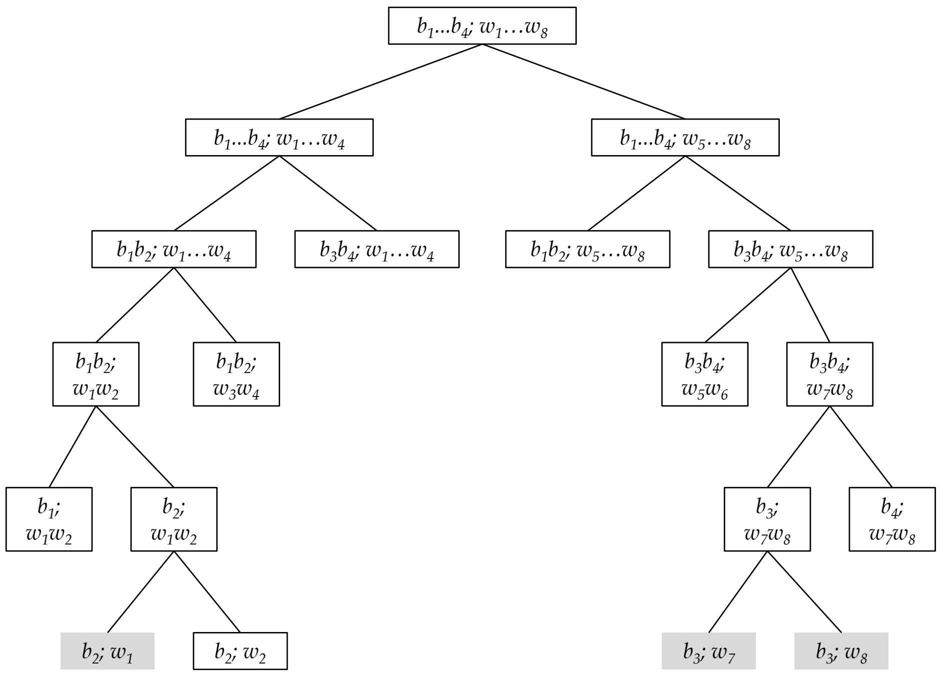


Figure 6.2. Search tree built for Figure 6.2

6.2.2 Incremental Maintenance of kMCV with respect to Updates

This work assumes the underlying graph is static, what if the graph is evolving with nodes and edges added or deleted? E.g., Figure 6.3 addes a black vertex b_5 and three edges associated with b_5 compared with Figure 6.1. Initially, the 1MCV in Figure 6.1 gives b_3. However, after the graph is updated in Figure 6.3, the 1MCV becomes b_5.

The research question is, how to efficiently maintain the kMCV on dynamic graphs? The naive solution is to re-compute kMCV from scratch. However, it is unnecessary to do so. E.g., the results of some previous tests (applied on Figure 6.1) are still applicable to the updated graph.

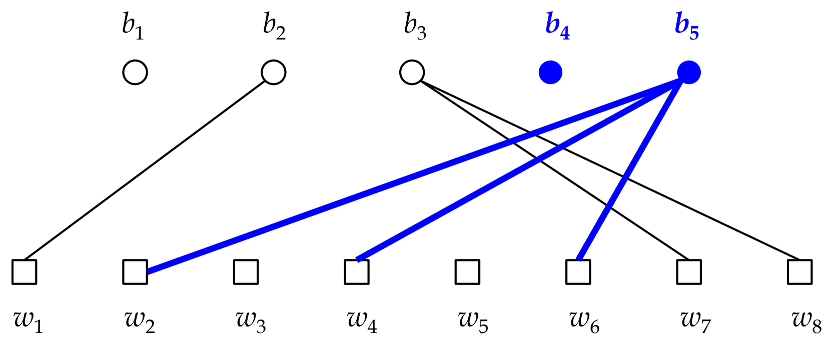


Figure 6.3. Adding vertices and edges

We need to differentiate different cases. (1) Only add nodes to $\mathcal{B}$; (2) Only delete nodes from $\mathcal{B}$; (3) Only add nodes to $\mathcal{W}$; (4) Only Delete nodes from $\mathcal{W}$; (5) Add nodes to both $\mathcal{B}$ and $\mathcal{W}$; (6) Delete nodes from both $\mathcal{B}$ and $\mathcal{W}$; (7) Add nodes to $\mathcal{B}$, delete nodes from $\mathcal{W}$; (8) Add nodes to $\mathcal{W}$, delete nodes from $\mathcal{B}$.

Chapter 7

Appendix

7.1 Proof of Lemma 2

Proof. We further state one more supplementary Lemma 8:

Lemma 8 *At the end of the iteration-$\mathcal{I}$ of GMCV, for a black vertex b_j that is not completed, (a) if $b_j.skip = 0$ then $\mathcal{I} = \mathcal{E}_{b_j}$; and (b) if $b_j.skip > 0$ then $\mathcal{I} < \mathcal{E}_{b_j}$. Thus, for any case, we have $\mathcal{I} \leq \mathcal{E}_{b_j}$.*

In order to prove that iteration-$\mathcal{I}$ is the last iteration of GMCV, we show that, the result set R contains the top k vertices, otherwise, the algorithm will still continue. It suffices to show that the k-th ranked vertex b^k is in R by the end of iteration-$\mathcal{I}$. According to GMCV, b^k is inserted into R when (I) it is completed; and (II) $deg(b^k) > \mu$, where $deg(b^k)$ is the degree of the vertex b^k.

For (I), if b^k is not completed, by Lemma 8,

$$\mathcal{I} \leq \mathcal{E}_{b^k} \tag{7.1}$$

While $b_j.skip = 0$, by Lemma 8, we have

$$\mathcal{I} = \mathcal{E}_{b_j}^{z-1} \tag{7.2}$$

Thus,

$$\begin{aligned} deg(b^k) &\leq |\mathcal{W}| - \mathcal{E}_{b^k} \\ &\leq |\mathcal{W}| - \mathcal{I} \qquad \triangleright \text{ by } (7.1) \\ &= |\mathcal{W}| - \mathcal{E}_{b_j}^{z-1} \qquad \triangleright \text{ by } (7.2) \\ &\leq |\mathcal{W}| - \theta \qquad \triangleright \text{ by } (3.1) \\ &= |\mathcal{W}| - (|\mathcal{W}| - \tau + 1) \qquad \triangleright \text{ by definition of } \theta \\ &= \tau - 1 \end{aligned} \tag{7.3}$$

We now reach a contradiction because $deg(b^k) = \tau$, i.e., the k-th degree. Therefore, we can conclude that b^k is completed.

For (II), we have $deg(b^k) = \tau$, which means, we have to show $\tau > \mu$. Recall that μ is the maximum degree upper bounds for vertices not in R, which consists of two parts: vertices in $\mathcal{B}$ but not yet completed; and vertices are completed but not yet in R. Let μ_1 and μ_2 be the maximum degree upper bounds for the former and the latter cases, respectively. Then $\mu = \max\{\mu_1, \mu_2\}$. We have $\tau > \mu_2$ (by line 25 of GMCV). Next, we show $\tau > \mu_1$.

Let b be a vertex that is not completed yet. By Lemma 8, we have,

$$\mathcal{I} \leq \mathcal{E}_b \tag{7.4}$$

Furthermore,

$$\begin{aligned} b.maxDeg &= |\mathcal{W}| - \mathcal{E}_b \\ &\leq |\mathcal{W}| - \mathcal{I} \qquad \triangleright \text{ by (7.4)} \\ &= \tau - 1 \qquad \triangleright \text{ by (7.3)} \end{aligned}$$

Thus, $\mu_1 = \max_{b \in \mathcal{B}} b.maxDeg \leq \tau - 1$, i.e., $\mu_1 < \tau$, together with $\mu_2 < \tau$, thus, $\mu = \max\{\mu_1, \mu_2\} < \tau$, which completes the proof of (II). □

7.2 Proof of Lemma 3

Proof. We have to prove that the result set R contains the top k vertices. Similar to Lemma 2, we need to show that the k-th rank vertex b^k is in R by the end of iteration-($\mathcal{I} + b_j.skip$). So, we have the following two statements: (I) b^k is completed otherwise, it cannot be in R; and (II) $deg(b^k) > \mu$.

For (I), at the end of the iteration-($\mathcal{I}+b_j.skip$), the number of empty vertices discovered is $\mathcal{E}_{b_j}^{z-1}$, hence

$$\begin{aligned} deg(b^k) &\leq |\mathcal{W}| - \mathcal{E}_{b_j}^{z-1} \\ &\leq |\mathcal{W}| - \theta \qquad \triangleright \text{ by (3.1)} \\ &= \tau - 1 \qquad \triangleright \text{ by definition of } \theta \end{aligned}$$

We reach a contradiction because $deg(b^k)$ is supposed to be τ. Thus, we conclude that b^k is completed.

Similar to the case (II) in the proof of Lemma 2, we can prove (II). $\square$

7.3 Proof of Lemma 8

Proof. We prove it by induction. Let b_j be a vertex not completed yet, first, we prove Lemma 8(a) if $b_j.skip = 0$, we have $\mathcal{I} = \mathcal{E}_{b_j}$.

Let t be the number of times that $b_j.skip = 0$ by the end of the iteration-$\mathcal{I}$ during GMCV. We show that, when $t = 1$, $\mathcal{I} = \mathcal{E}_{b_j}$ (initialization). We assume that $\mathcal{I} = \mathcal{E}_{b_j}$ holds when $t > 1$. We prove that at the $(t + 1)$-th time that $b_j.skip = 0$, $\mathcal{I} = \mathcal{E}_{b_j}$ still holds.

Proof of Lemma 8(a) via induction. When $t = 1$, i.e., by the end of the first iteration (because every vertex has to identify one empty vertex unless it is completed), here $\mathcal{I} = \mathcal{E}_{b_j} = 1$. That is, the equation holds for the initialization; Assume it is true during the t-th time that $b_j.skip$ becomes 0 while GMCV is at the end of iteration-$\mathcal{I}$; Next, we show it holds for the $(t + 1)$-th time. Suppose during the iteration-$(\mathcal{I}+1)$, x empty vertices were detected of b_j ($x \geq 1$ unless b_j is completed). We consider the case when $x > 1$ as is trivial for $x = 1$. Meaning that, $\mathcal{E}_{b_j}$ is increased by x. While according to GMCV, its skip is increased by $x - 1$, hence, from iteration-$(\mathcal{I} + 2)$ to iteration-$(\mathcal{I} + x)$, b_j (since $b_j.skip > 0$) is skipped and no new empty vertices are detected, therefore, $\mathcal{E}_{b_j}$ does not change compared to the end of the iteration-$(\mathcal{I} + 1)$. Therefore, we still have $\mathcal{I} = \mathcal{E}_{b_j}$ since both are increased by x. Therefore, it holds for the $(t + 1)$-th time.

Next, we prove Lemma 8(b) that if $b_j.skip > 0$ at the end of iteration-$\mathcal{I}$, then $\mathcal{I} < \mathcal{E}_{b_j}$. First, before iteration-$\mathcal{I}$, we know there must be an iteration-$\mathcal{I}'$ (just before iteration-$\mathcal{I}$) where $b_j.skip = 0$. Furthermore, let $\mathcal{E}'_{b_j}$ be the number of empty vertices detected at the end of iteration-$\mathcal{I}'$, from (a), we have

$$\mathcal{I}' = \mathcal{E}'_{b_j} \tag{7.5}$$

Let x be the number of empty vertices detected during the iteration-$(\mathcal{I}' + 1)$ (i.e., $b_j.skip = x - 1$). Then, at the end of iteration-$(\mathcal{I}' + 1)$, we have

$$\mathcal{E}_{b_j} = \mathcal{E}'_{b_j} + x \tag{7.6}$$

Since at the end of iteration-$\mathcal{I}$, $b_j.skip > 0$, we have

$$\begin{aligned} \mathcal{I} &< \mathcal{I}' + x \\ &= \mathcal{E}'_{b_j} + x \qquad \rhd \text{ by (7.5)} \\ &= \mathcal{E}_{b_j} \qquad \rhd \text{ by (7.6)} \end{aligned}$$

This completes the proof. $\square$

Bibliography

[1] L. A. Adamic and B. A. Huberman. Power-law distribution of the world wide web. *Science*, 287:2115, 2000.

[2] R. Albert, H. Jeong, and A. L. Barabasi. The diameter of the world wide web. *Nature*, 401:130–131, 1999.

[3] N. Alon and V. Asodi. Learning a hidden subgraph. *SIAM Journal on Discrete Mathematics (SIDMA)*, 18:697–712, 2005.

[4] N. Alon, R. Beigel, S. Kasif, S. Rudich, and B. Sudakov. Learning a hidden matching. *SIAM Journal on Computing (SICOMP)*, 33:487–501, 2004.

[5] D. Angluin and J. Chen. Learning a hidden graph using O(log n) queries per edge. *Journal of Computer and System Sciences (JCSS)*, 74:546–556, 2008.

[6] A. Bar-Noy, F. K. Hwang, I. Kessler, and S. Kutten. A new competitive algorithm for group testing. *Discrete Applied Mathematics*, 52(1):29–38, 1994.

[7] M. Bouvel, V. Grebinski, and G. Kucherov. Combinatorial search on graphs motivated by bioinformatics applications: A brief survey. In *International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 16–27, 2005.

[8] C. Chen, X. Yan, P. S. Yu, J. Han, D.-Q. Zhang, and X. Gu. Towards graph containment search and indexing. In *Proceedings of Very Large Data Bases (VLDB)*, pages 926–937, 2007.

[9] J. Cheng, Y. Ke, W. Ng, and A. Lu. Fg-index: towards verification-free query processing on graph databases. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 857–872, 2007.

[10] E. D. Demaine, A. López-Ortiz, and J. I. Munro. Adaptive set intersections, unions, and differences. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 743–752, 2000.

[11] R. Dorfman. The detection of defective members of large populations. *The Annals of Mathematical Statistics*, 14:436–440, 1943.

[12] D. Du and F. K. Hwang. *Combinatorial group testing and its applications.* World Scientific Press, 2000.

[13] D. Eppstein, M. T. Goodrich, and D. S. Hirschberg. Improved combinatorial group testing algorithms for real-world problem sizes. *SIAM Journal on Computing (SICOMP)*, 36:1360–1375, 2006.

[14] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 102–113, 2001.

[15] W. I. Gasarch and C. H. Smith. Learning via queries. *Journal of the ACM (JACM)*, 39:649–674, 1992.

[16] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM (JACM)*, 45:653–750, 1998.

[17] E. Golemis and P. Adams. *Protein-protein interactions: a molecular cloning manual.* Cold Spring Harbor Laboratory Press, 2005.

[18] V. Grebinski and G. Kucherov. Reconstructing a hamiltonian cycle by querying the graph: application to DNA physical mapping. *Discrete Applied Mathematics*, 88:147–165, 1998.

[19] H. Hu, D. L. Lee, and V. C. S. Lee. Distance indexing on road networks. In *Proceedings of Very Large Data Bases (VLDB)*, pages 894–905, 2006.

[20] I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys (CSUR)*, 40(4):1–58, 2008.

[21] N. T.-M. Laurent, L. Trilling, and J. louis Roch. A novel pooling design for protein-protein interaction mapping, 2004.

[22] Y. Li, M. T. Thai, Z. Liu, and W. Wu. Protein-protein interaction and group testing in bipartite graphs. *International Journal of Bioinformatics Research and Applications*, 1:414–419, 2005.

[23] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query processing in spatial network databases. In *Proceedings of Very Large Data Bases (VLDB)*, 2003.

[24] A. Parameswaran, A. D. Sarma, H. Garcia-Molina, N. Polyzotis, and J. Widom. Human-assisted graph search: it's okay to ask questions. *Proceedings of the VLDB Endowment (PVLDB)*, 4(5):267–278, 2011.

[25] H. Samet, J. Sankaranarayanan, and H. Alborzi. Scalable network distance browsing in spatial databases. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 43–54, 2008.

[26] J. Schlaghoff and E. Triesch. Improved results for competitive group testing. *Combinatorics, Probability and Computing*, 14:191–202, 2005.

[27] H. Shang, Y. Zhang, X. Lin, and J. X. Yu. Taming verification hardness: an efficient algorithm for testing subgraph isomorphism. *Proceedings of the VLDB Endowment (PVLDB)*, 1:364–375, 2008.

[28] C. Sheng, Y. Tao, and J. Li. Exact and approximate algorithms for the most connected vertex problem. *ACM Transactions on Database Systems (TODS)*, 37(2):1–39, 2012.

[29] Y. Tao, C. Sheng, and J. Li. Finding maximum degrees in hidden bipartite graphs. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 891–902, 2010.

[30] J. R. Thomsen, M. L. Yiu, and C. S. Jensen. Effective caching of shortest paths for location-based services. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 313–324, 2012.

[31] F. Wei. TEDI: Efficient shortest path query answering on graphs. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 99–110, 2010.

[32] Y. Xuan, I. Shin, M. Thai, and T. Znati. Detecting application denial-of-service attacks: A group-testing-based approach. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, pages 1203–1216, 2010.

[33] S. Zhang, J. Li, H. Gao, and Z. Zou. A novel approach for efficient supergraph query processing on graph databases. In *Proceedings of Extending Database Technology (EDBT)*, pages 204–215, 2009.