Identifying the Most Connected Vertices in Hidden Bipartite Graphs using Group Testing

Jianguo Wang, Eric Lo, and Man Lung Yiu

Abstract—A graph is called *hidden* if the edges are not explicitly given and *edge probe tests* are required to detect the presence of edges. This paper studies the *k*MCV (*k* most connected vertices) problem on hidden bipartite graphs, which has applications in spatial databases, graph databases, and bioinformatics. There is a prior work on the *k*MCV problem, which is based on the "2-vertex testing" model, i.e., an edge probe test can only reveal the existence of an edge between two individual vertices. We study the *k*MCV problem, in the context of a more general edge probe test model called "group testing". A group test can reveal whether there exists some edge between a vertex and a group of vertices. If group testing 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 group testing could be used, and present an algorithm, namely, GMCV, that adaptively leverages group testing to solve the *k*MCV problem.

Index Terms—H.2.4.h Query processing; E.1.d Graphs and networks

1 Introduction

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 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 aims to identify the vertex b_1 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].

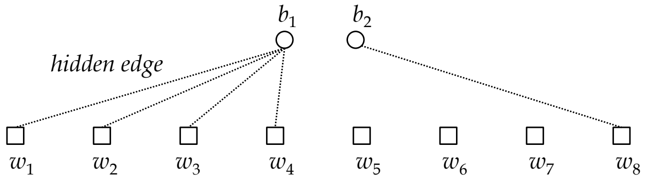


Fig. 1. A (hidden) bipartite graph $\mathcal{G}(\mathcal{B}, \mathcal{W})$; edges are not explicitly given

• *J. Wang, E. Lo and M. L. Yiu are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong.*
E-mail: {csjgwang, ericlo, csmlyiu}@comp.polyu.edu.hk

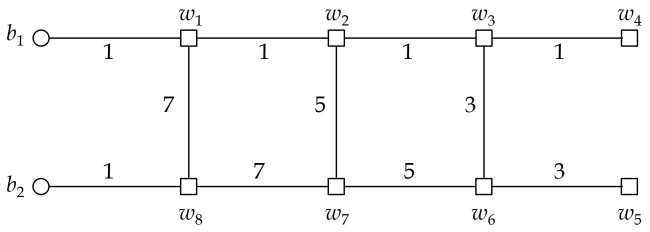


Fig. 2. Example of a road network

Example 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 $\mathcal{G}(\mathcal{B}, \mathcal{W})$. 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 2 shows a road network. Figure 1 is the hidden graph representation of distance join on Figure 2, using $\theta_{dist} = 5$. To detect whether hotel b_1 and scenic spot w_2 have an edge connecting in Figure 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 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 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.

Example 2: Bioinformatics.
In bioinformatics, interactions between proteins are often represented as graphs. Specifically, the interac-

tions 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.

Example 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 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 testing*, or simply *2-vertex testing* [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 and w possesses an edge in the hidden bipartite graph $\mathcal{G}$ and 0 otherwise. In many applications [11], [13], [32], [7], the more general *vertex-group edge probe testing* 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 $w \in W$ 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*. 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 $I_{\mathcal{W}}$.

Table 1 gives a summary of how the above applications associated with the kMCV problem in the context of group testing.

The applicability of group testing on the kMCV problem raises a very interesting research question: *Can we leverage group testing 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 group testing, 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 testing. 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

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.

TABLE 1
Applications that can apply group testing

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

increase the chance of the testing 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 group testing 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 group testing 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 testing. 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 an 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 [23] 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 do next after finishing the current experiment (which may take days).

The rest of the paper is organized as follows. We review the related work in Section 2. We formally define the problem in Section 3. 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 Section 4.
- Then, we present cost models of GMCV and SOE, in Section 5. Notice that the total external testing 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 testing 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 Section 6. The evaluation is done on both real life datasets and synthetic datasets. The experimental results show that GMCV is a good general alternative to SOE.

After presenting the above contributions, we conclude the paper in Section 7. Table 2 summarizes the symbols used in the subsequent sections.

TABLE 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 testing of b and w
$Q(b, W)$	group testing of b and W
$\beta(b, w)$ or $\beta(1)$	external testing cost of $Q(b, w)$
$\beta(b, W)$ or $\beta(\lvert W\rvert)$	external testing cost of $Q(b, W)$

2 Related Work

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.

The works above are all based on the 2-vertex edge probe test model [7]. They assume that the cost of a 2-vertex test is a constant. So, the costs of those algorithms are analyzed based on the number of tests they invoked. Thus it is natural that those works focus on reducing the number of tests. Recently, the more general vertex-group edge probe test model is used in both graph testing and graph learning [13], [32], [7], because in those applications the cost of a group test is independent of the group size. This paper aims to investigate the use of group testing in solving the kMCV problem.

Comparing with SOE [29], [28], the use of group testing 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 testing cost, which depends on (i) the number of executed group tests, (ii) the group size, and (iii) the cost function of various group testing implementations.

3 Problem Definition

We formally define the kMCV (k most connected vertices) problem under the group testing 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 testing). *An edge probe test $Q(b, w)$ is called* 2-vertex testing *if it asks whether a black vertex $b \in \mathcal{B}$ connects with a white vertex $w \in \mathcal{W}$:*

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

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

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

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

Definition 3 (external testing 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 $\beta(b, W)$. For simplicity, we represent $\beta(b, W)$ using the 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 paper is to minimize the total external testing cost of solving the kMCV problem using group testing. For ease of presentation, we assume there is no tie on the vertex's degree such that there is exactly k vertices in the result set R. Our techniques can be easily extended to handle the tie case.

4 Algorithm GMCV

In this section, we present our GMCV algorithm that solves the kMCV problem by the use of group testing, which aims to reduce the external testing 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 group testing 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 testing for low-degree vertices?*

4.1 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 group testing with, and W^{PRE} be the previous set of white vertices that b_i carried out group testing 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 group testing 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]2. The rationale is that, if $Q(b, W^{PRE}) = 0$, it implies b_i might have a low degree. Thus, 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.

2. 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 because of its stableness.

When b_i is in the *identifying* state, the objective of group testing 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^{PRE}_L and W^{PRE}_R, and sets W^{CUR} to be W^{PRE}_L and saves W^{PRE}_R 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 testing $Q(b_i, W^{CUR})$. 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 4 shows an example that illustrates some of the cases above. The corresponding input hidden graph is shown in Figure 3. 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)]**, 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.

4.2 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 testing 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* testing even though τ is unknown beforehand? In other words, how to guarantee for any $b_j \notin R$, it does not have any unnecessary testing 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 (Section 4.3), we will show that, with our skipping policy, vertices not in the result set do not have unnecessary testing. Then, we will show in Lemma 4 (Section 4.3) 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.

4.3 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
            /*Section 4.1*/
15          if $Q(b, W^{CUR}) = 0$ then /*external testing*/
16              $b.maxDeg \leftarrow b.maxDeg - |W^{CUR}|$
17              $b.skip \leftarrow b.skip + (|W^{CUR}| - 1)$
18          else
19              if $|W| = 1$ 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 shows the detailed execution steps of GMCV in finding the 1MCV of the hidden graph presented in Figure 3. 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.

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

Proof: 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. □

Theorem 1. *In GMCV, a black vertex $b_j \notin R$ stops any further testing, 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 testing *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 $\mathcal{E}_{b_j}^m$ be the value of $\mathcal{E}_{b_j}$ after the m-th *change* of $\mathcal{E}_{b_j}$. (Thus, $\mathcal{E}_{b_j}^m \leq \mathcal{E}_{b_j}^{m+1}$). Let $\mathcal{E}_{b_j}^z$ be the value of $\mathcal{E}_{b_j}$ of the last change of $\mathcal{E}_{b_j}$ before GMCV terminates, we have (I) $\mathcal{E}_{b_j}^z \geq \theta$ and (II) $\mathcal{E}_{b_j}^{z-1} < \theta$.

We prove (I) by contradiction. At the end of GMCV, if $\mathcal{E}_{b_j} < \theta$ (i.e., $\mathcal{E}_{b_j}^z < \theta$), we have $b_j.maxDeg = |\mathcal{W}| - \mathcal{E}_{b_j}^z > |\mathcal{W}| - \theta = \tau - 1$. 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) $\mathcal{E}_{b_j}^{z-1} < \theta$ by contradiction. Let us assume

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

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

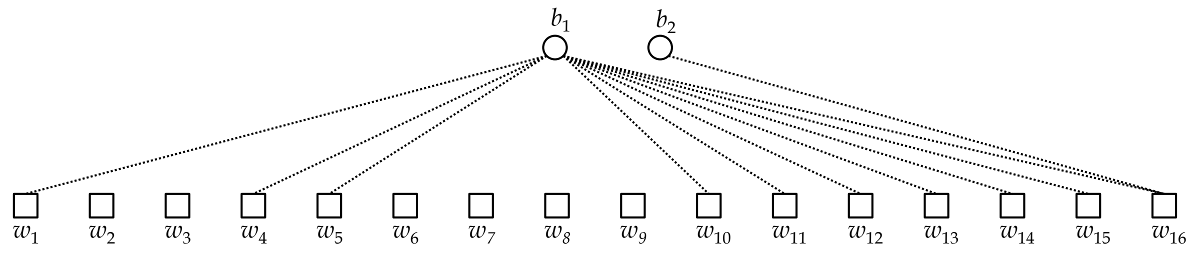

b_1
b_2
w_1 w_2 w_3 w_4 w_5 w_6 w_7 w_8 w_9 w_{10} w_{11} w_{12} w_{13} w_{14} w_{15} w_{16}

Fig. 3. A hidden bipartite graph

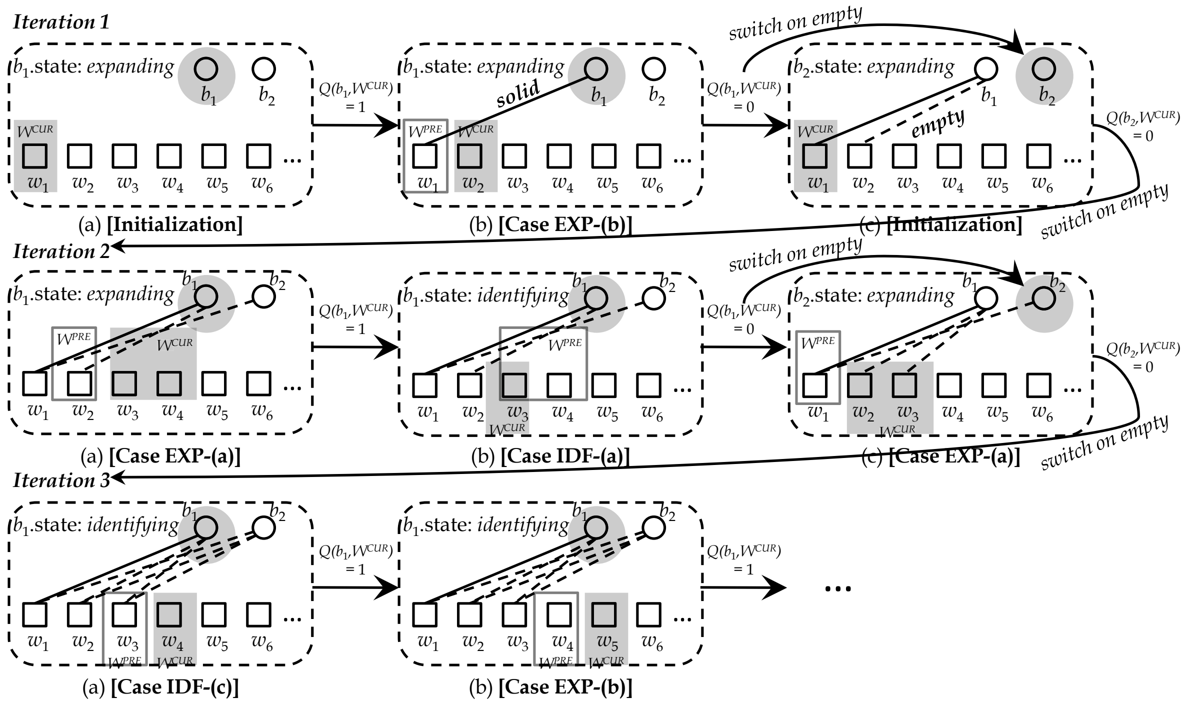


Fig. 4. Running example

TABLE 3
Detailed execution steps of GMCV on Figure 3

Iterations	Vertex	Testing 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]

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 $\mathcal{E}^{z-1}_{b_j}$. So, if we can prove that GMCV has terminated by that time, it implies that $\mathcal{E}^{z-1}_{b_j}$ is the value of $\mathcal{E}_{b_j}$ before GMCV terminates, which contradicts the fact that $\mathcal{E}^z_{b_j}$ is the last change of $\mathcal{E}_{b_j}$.

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

Lemma 4. *Let C_s be the external testing cost of our solution with skip factor of s, then, for any s, $C_{|\mathcal{W}^{\text{CUR}}|-1} \leq C_s$.*

Proof: First, 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 testing cost is the minimum if its test sequence stops once $b_j.maxDeg$ is just smaller than τ. We denote that cost as $minC^{b_j}$.

Let $C^{b_j}_s$ be the external testing cost of b_j with any skip factor s. We first show that $C^{b_j}_s = minC^b$ when $s = |\mathcal{W}^{CUR} - 1|$. That is, we show $C^{b_j}_{|\mathcal{W}^{\text{CUR}}-1|} = minC^{b_j}$. If that is proved, then it is straightforward to deduce the total cost of all $b_j \notin R$ as minimum and thereby proved the lemma.

As mentioned above, if $C^{b_j}_{|\mathcal{W}^{\text{CUR}}-1|} = minC^{b_j}$, it implies GMCV stops processing b_j once its degree upper bound $b_j.maxDeg$ gets refined so that it is smaller than τ, which is proved in Theorem 1. □

5 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 testing. In SOE, minimizing the number of tests is equivalent to minimizing the total external testing 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 section, we provide cost models to capture the total external testing costs of GMCV (Section 5.1) and SOE (Section 5.2), and compare their external costs based on different group test cost functions (Section 5.3). 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.

5.1 External Testing 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 $Q(b_i, W^1_i)$ is carried out. If $Q(b_i, W^1_i) = 0$, another group test $Q(b_i, W^2_i)$ is carried out. The expanding phase $Q(b_i, W^1_i) = 0, Q(b_i, W^2_i) = 0, \cdots$, continues until the s-th test in which $Q(b_i, W^s_i) = 1$ (while all the previous tests return 0), where s is called the *turning point* in the process. After that, the identifying phase starts: $Q(b_i, W^{s+1}_i), \cdots, Q(b_i, W^{2s-1}_i)$, i.e., recursively drill into the set W^s_i to locate the solid vertex.

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

Proof: Note that the size of the vertex set W^j_i has the following property

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

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

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

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

where d_i is the degree of b_i, and the value of turning point s in $\mathcal{C}^j_i$ is set as $\lfloor \lg \frac{|\mathcal{W}|}{d_i} \rceil + 1$ ($\lfloor x \rceil$ denotes the randomized rounding [24] of x).

Proof: Lemma 5 gives the external testing 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 $d_i \cdot \mathcal{C}^j_i$. The value of s in $\mathcal{C}^j_i$ is derived as follows.

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 group testing 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 testing cost associated with a vertex $b_j \notin R$. Before that, we define $A(t)$ be the accumulated external testing cost in order to identify t empty vertices through a series of group tests whose results are all zero (i.e., the external testing 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 testing 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 $\theta = |\mathcal{W}| - \tau + 1$, $\lambda_j = \lfloor \frac{\theta}{\frac{|\mathcal{W}|}{d_j} - 1} \rfloor$, $s = \lfloor \lg \frac{|\mathcal{W}|}{d_j} \rceil + 1$, 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 testing 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 testing 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 testing cost gives the result, which completes the proof. □

Theorem 2. *The external testing cost of GMCV is:*

$$Cost_{GMCV} = \sum_{b_i \in R} Cost(b_i) + \sum_{b_j \notin R} Cost(b_j) \qquad (2)$$

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

5.2 External Testing 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

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

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

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

$$Cost_{SOE} = \mathcal{N}_{SOE} \cdot \beta(1) \qquad (3)$$

5.3 Cost Comparison

We compare the external testing costs of GMCV and SOE based on the cost models established in Equations (2) and (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:

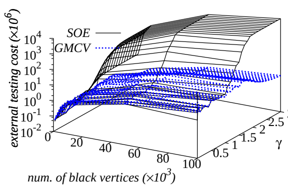
$$Pr(d) = \frac{1/(d+1)^\gamma}{\sum_{i=0}^{|\mathcal{W}|} 1/(i+1)^\gamma} \qquad (4)$$

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.

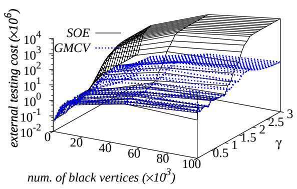
We consider four group testing implementations:

(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 $\beta(|W|) = |W| \cdot \beta(1)$

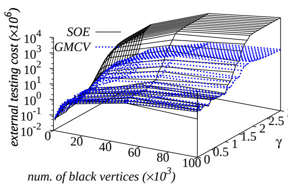
The Const implementation is to simulate the group test implementation in the biological domain, in which both the monetary cost and the running time of an experiment is a constant [17]. The Log and the Sqrt implementations are to simulate the group test implementations in the graph pattern matching and 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.



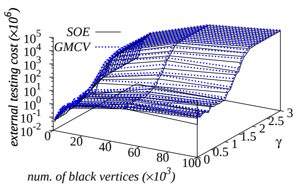
(a) Const



(b) Log



(c) Sqrt



(d) Linear

Fig. 5. Derived Costs of SOE and GMCV

Figure 5 plots the external testing costs of GMCV and SOE ($k = 10$) based on Equations (2) and (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)[3] or when GMCV is deliberately misused on applications where the external cost of a group testing is (super) linear to the size of the input. In those cases, we found GMCV and SOE have comparable performance.

6 Experiments

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

PPI[4]. 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[5]. 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[6]. It is an actor collaboration network data based on IMDB (http://www.imdb.com). In which, $\mathcal{B}$ and $\mathcal{W}$ include all the actors. In particular, two actors b and w have 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 4 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 $\gamma < 0$.

TABLE 4
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

3. Normally, γ is larger than 2.0 in real graphs [1], [2].
4. http://turing.cs.iastate.edu/PredDNA/dataset.html
5. www.maproom.psu.edu/dcw
6. http://www.datatang.com/DataRes/Detail.aspx?id=1624

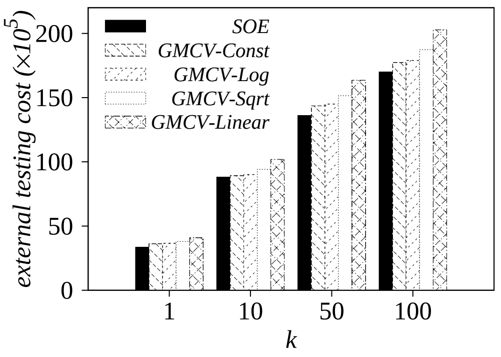


Fig. 9. Results on Actor-D data

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 testing functions Const, Log, Sqrt, and Linear mentioned in Section 5.3. 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 testing cost.

6.1 Experimental Results on Real Datasets

Figure 6 (the bigger graph) shows the external testing costs of GMCV (based on different group testing 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 testing 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 7 and Figure 8. We can also observe that GMCV outperforms SOE significantly, again except when the Linear group testing function is deliberately used.

Figure 9 shows the external testing 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 testing, 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 testing 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.

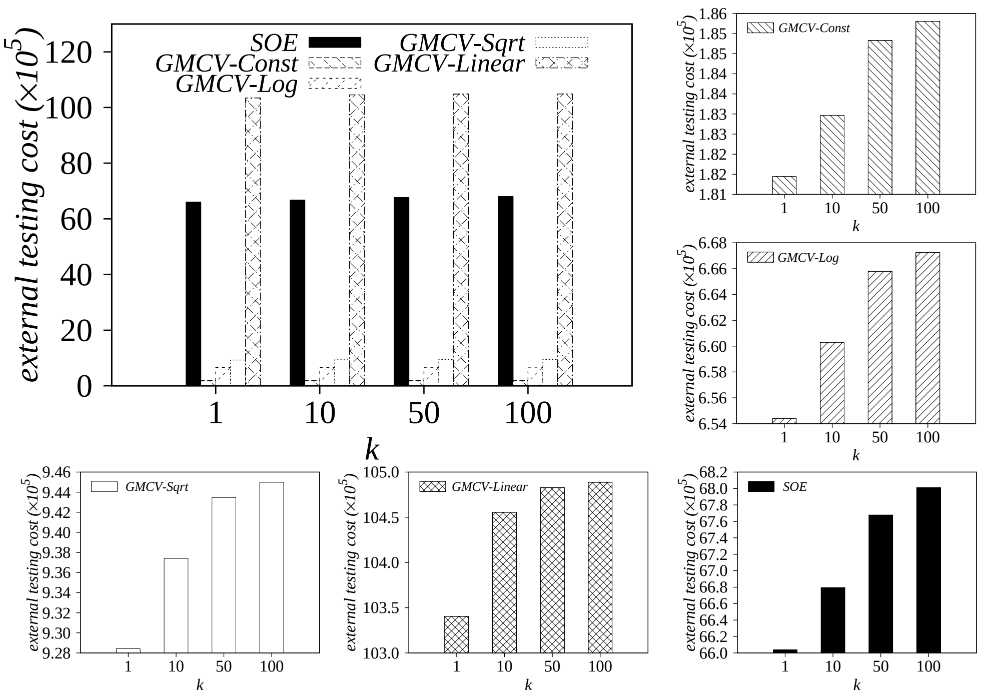


Fig. 6. Results on PPI data, varying k

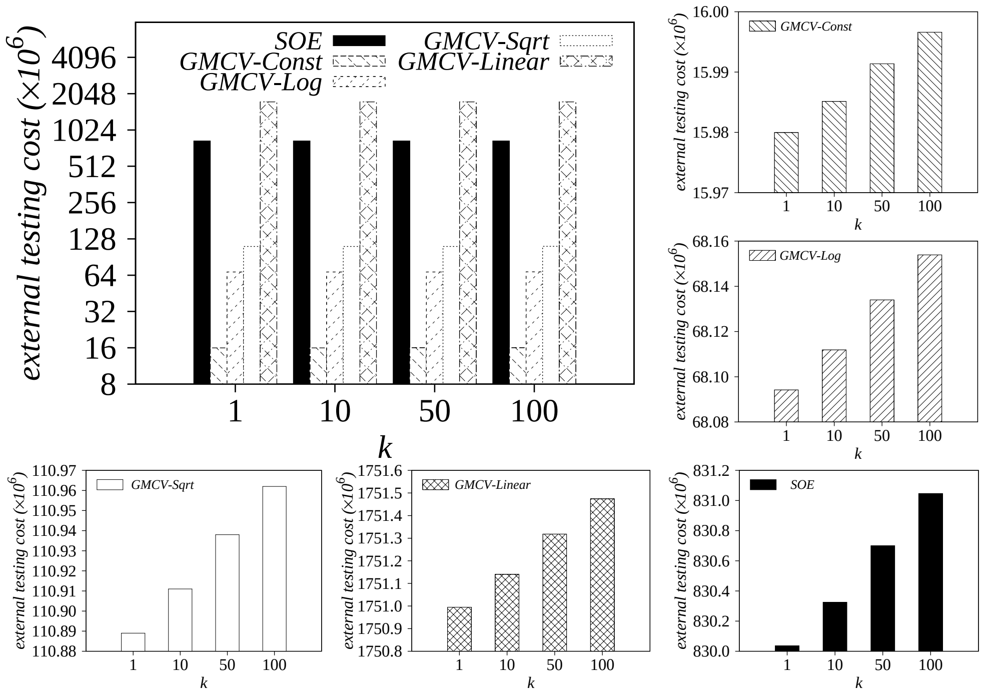


Fig. 7. Results on Germany data, varying k

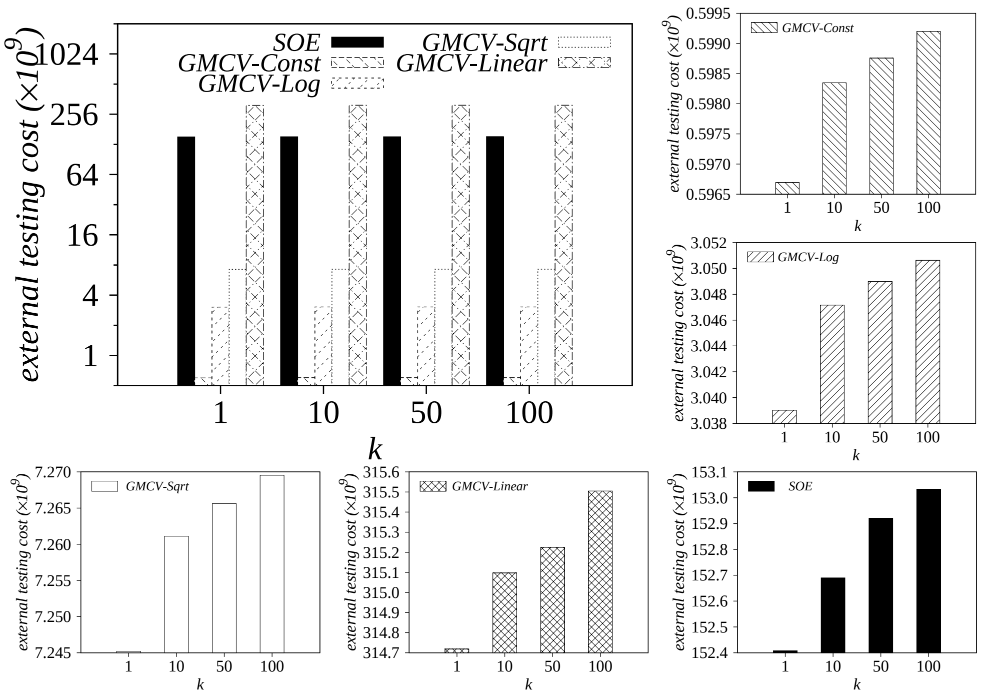


Fig. 8. Results on Actor-W data, varying k

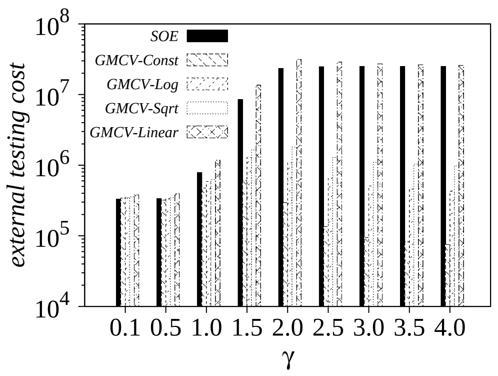


Fig. 10. Varying graph sparseness; $k = 10$

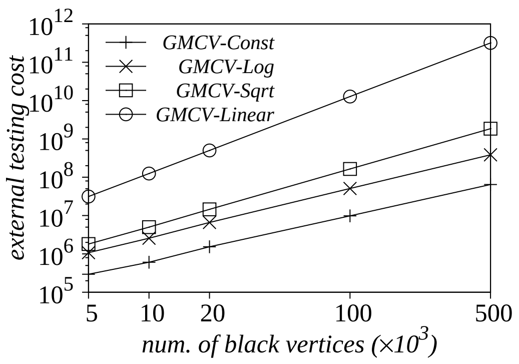


Fig. 11. Varying num. of black vertices $|\mathcal{B}|$, with $|\mathcal{B}| = |\mathcal{W}|$; $k = 10$

6.2 Experimental Results on Synthetic Datasets

6.2.1 Sparseness

Figure 10 shows the external testing 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 ($\gamma = 0.1$).

6.2.2 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 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 11 shows the external testing costs of GMCV running on synthetic graphs of different sizes. We can see that GMCV scales well on graphs of different sizes.

7 Conclusions

This paper studies the kMCV (k most connected vertices) problem on hidden bipartite graphs in the context of group testing. Group testing is a common testing model in hidden graph literature. Instead of testing the presence of edge between only two vertices (which is called the 2-vertex testing model), a group test takes as input a group of vertices and returns whether there is any edge among them. If group testing 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 testing, 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 testing based kMCV algorithm, except in some extreme cases (e.g., when the linear implementation of group testing is deliberately used or the graphs are unusually dense). In those cases, GMCV still has comparable performance with SOE, making GMCV is a robust and more effective choice than SOE in the usual settings.

REFERENCES

[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. Hwang K. *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] 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.

[24] P. Raghavan and C. D. Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[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.

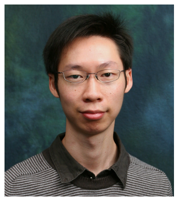
[30] J. R. Thomsen, M. L. Yiu, and C. S. Jensen. Effective caching of shortest paths for location-based services. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 313–324, 2012.

[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.

Jianguo Wang received the bachelor's degree in 2009 from Zhengzhou University, China. He is currently an MPhil student in the Hong Kong Polytechnic University, Hong Kong, under the supervision of Dr. Man Lung Yiu. His research interests include information retrieval and database.



Eric Lo received his PhD degree in 2007 from ETH Zurich. He is currently an assistant professor in the Department of Computing, Hong Kong Polytechnic University. He research interests include query processing, query optimization, and large-scale data analysis.

Man Lung Yiu received the bachelor's degree in computer engineering and the PhD degree in computer science from the University of Hong Kong in 2002 and 2006, respectively. Prior to his current post, he worked at Aalborg University for three years starting in the Fall of 2006. He is now an assistant professor in the Department of Computing, Hong Kong Polytechnic University. His research focuses on the management of complex data, in particular query processing topics on spatiotemporal data and multidimensional data.