Optim Lett (2014) 8:2341–2348
DOI 10.1007/s11590-014-0751-9

ORIGINAL PAPER

Learning a hidden graph

Huilan Chang · Hung-Lin Fu · Chih-Huai Shih

Received: 29 March 2013 / Accepted: 8 May 2014 / Published online: 28 May 2014

Abstract We study the problem of learning a hidden graph by edge-detecting queries, each of which tells whether a set of vertices induces an edge of the hidden graph or not. We provide a new information-theoretic lower bound and give a more efficient adaptive algorithm to learn a general graph with n vertices and m edges in $m \log n + 10m + 3n$ edge-detecting queries.

Keywords Combinatorial search · Graph learning · Competitive group testing · Complex model

1 Introduction

In a *graph learning problem*, a hidden graph G is known to belong to a given family $\mathcal{G}$ of labeled graphs on vertex set $[n] := \{1, 2, \cdots, n\}$. Referring to the information of "belonging to $\mathcal{G}$", we wish to identify G by *edge-detecting queries*, each of which tells whether a subset of $[n]$ induces an edge of G. Such a problem is motivated by applications in DNA physical mapping [14]. In the strategy proposed by Sorokin et al. [19] for physical mapping, some cloned fragments are produced to cover the

H. Chang
Department of Applied Mathematics, National University of Kaohsiung,
Kaohsiung 811, Taiwan, ROC
e-mail: huilan0102@gmail.com

H.-L. Fu · C.-H. Shih (✉)
Department of Applied Mathematics, National Chiao Tung University,
Hsinchu 30010, Taiwan, ROC
e-mail: skyking.shih@gmail.com

H.-L. Fu
e-mail: hlfu@math.nctu.edu.tw

whole DNA molecule with some gaps and then some of them are assembled to form longer continuous fragments (contigs). However, the information about the mutual placement of the contigs on the DNA sequence is lost. Multiplex PCR is a tool to tell whether there are two neighbouring contigs among a set of contigs so as to identify the relative replacements of the contigs. For example, if a DNA molecule is circular, then identifying every pair of close contigs can provide the order of contigs along the circular molecule; this identification work can be modeled by a graph learning problem on a family of Hamiltonian cycles and each Multiplex PCR experiment plays an edge-detecting role in graph learning problem.

Learning a hidden graph is also a subarea of combinatorial search on graphs [1]. Many conditions on $\mathcal{G}$ have been considered in the literature [3,7,8,14,15]: labeled graphs in $\mathcal{G}$ have prescribed topology such as Hamiltonian cycle, matching, tree, star, and clique; labeled graphs in $\mathcal{G}$ have bounded degree. The problem of learning a general graph is to deal with labeled graphs that have no restricted topology. Specifically, $\mathcal{G}$ collects all simple graphs on $[n]$. Angluin and Chen [6] proved that a hidden general graph can be identified in $12m \log n$ edge-detecting queries ($\log := \log_2$) where m (unknown) is the number of edges in the hidden graph. This is the best known result to the best of our knowledge.

Learning a hidden general graph can be viewed as a variant of group testing which is a well-known field of combinatorial search. Given a set of items, each of which is either positive or negative, a *group test* on a subset of items determines whether it contains any positive item. The main task of *classical group testing problem* is to identify positive items by group tests. An extension of classical group testing is *the complex model*, where a set of complexes, each of which is a subset of basic items, is given and the property (positive or negative) of each complex is not yet determined; the corresponding query to identify positive complexes answers whether a subset of basic items contains at least one positive complex. In classical group testing, the number of positive items is usually assumed upper bounded while for *competitive group testing* [12], no information on the number of positive items is given. Learning a hidden graph is identical to a subcase of complex model where every set of two items is a complex; positive complexes correspond to edges and negative ones correspond to all pairs of items that are not edges. Many studies have been done on the classical competitive group testing (see [10,11,18] for recent development); however, to the best of our knowledge, very little is known about the competitive group testing on complex model.

An *affine plane* of order p contains p^2 points and $p^2 + p$ blocks of size p such that each pair of points occur together in exactly one of the blocks. Such an affine plane exists whenever p is a prime power [4]. The *affine plane method* proposed in [14,20] and then employed in [9,16] is to take an affine plane with the point set containing $[n]$ and then identify each subgraph induced by a block. In this paper, we concentrate on the study of learning a hidden general graph. We improve the trivial information-theoretic lower bound (see Sect. 2). Furthermore, by exploiting the affine plane method along with a fundamental algorithm, we can learn a general graph G on $[n]$ in $m \log n + 10m + 3n$ edge-detecting queries where m (unknown) is the number of edges of G, which is an improvement of the upper bound in [6] (see Sect. 3). Our algorithm performs as well as the algorithms in [9] for some families of graphs of known topology.

2 Lower bound

Angluin and Chen [6] proved that for any $0 < \epsilon < 2$, $\epsilon m(\log n - 2)$ edge-detecting queries are required to identify a hidden graph G drawn from the family $\mathcal{G}$ of all graphs with n vertices and $m = n^{2-\epsilon}$ edges. In the following, we deal with the problem that the hidden G is drawn from the family of all graphs with n vertices, that is, the number of edges in G is unknown. We provide the following information-theoretic lower bound.

Theorem 2.1 $\left\lceil \log\left(\sum_{i=0}^{m}\binom{\binom{n}{2}}{i}\right)\right\rceil$ *edge-detecting queries are required to identify a graph G drawn from the family of all graphs with n vertices if G has exactly m edges.*

Before proving our main result on the lower bound, we need to establish some notions and a lemma. A query can be represented by a subset S of $[n]$ and we use $q(S)$ to denote its outcome while $q(S) = 1$ means the outcome is "yes" and $q(S) = 0$ means otherwise. For simplicity, we call a pair of vertices a *non-edge* if it does not induce an edge of the hidden graph. Let T_A denote the computation tree of an adaptive algorithm A that solves the problem of learning a general graph on $[n]$. Then each leaf represents a graph G on $[n]$ and in T_A the path from the root to the leaf G represents the query history of the hidden graph G under A. While applying an algorithm, pairs of vertices are determined as non-edges or edges gradually. For any query $S \subseteq [n]$ in the query history of G, let E_S^- (resp. E_S^+) collect all pairs of vertices that are determined as non-edges (resp. edges) in the partial history from the beginning to S. Let $\binom{S}{2} := \{\{i, j\} : i \neq j \in \mathcal{S}\}$. Then $E_S^* := \binom{[n]}{2} \setminus (E_S^- \cup E_S^+)$ collects all pairs of vertices that are not determined right after S. Let $l_A(G)$ be the depth of a leaf G in T_A and $l_A(m) = \max\{l_A(G) : \text{G has } m \text{ edges}\}$. First of all, we have the following result.

Lemma 2.2 *Let $\mathcal{G}$ be the family of all graphs on $[n]$. Then for any algorithm A that solves the hidden graph problem on $\mathcal{G}$, there exists an algorithm A_m that also solves the hidden graph problem on $\mathcal{G}$ and satisfies $l_{A_m}(m) = l_A(m)$ and $l_{A_m}(r) \geq l_{A_m}(r-1)$ for any $r \leq m$.*

Proof Let A_m be a procedure obtained by adjusting A as follows: Suppose that G is the hidden graph in $\mathcal{G}$. While applying algorithm A, let S be the first query satisfying $|E_S^+ \cup E_S^*| = |\binom{[n]}{2} \setminus E_S^-| \leq m - 1$ and S' be the query preceding S. Since the number of edges in the hidden graph is unknown, a positive query does not provide any information to declare any pair of vertices as non-edge. Hence if $q(S) = 1$, then $|\binom{[n]}{2} \setminus E_{S'}^-| = |\binom{[n]}{2} \setminus E_S^-| \leq m - 1$, contradicting the selection of S. Therefore, $q(S) = 0$ and thus $\binom{S}{2} \cap E(G) = \emptyset$. Furthermore, assume without loss of generality that $\binom{S}{2} \nsubseteq E_{S'}^-$ since each pair of vertices in $E_{S'}^-$ has been determined as non-edge after S'. Now, we have an $e \in \binom{S}{2} \setminus E_{S'}^-$ that is an element in $E_{S'}^* \setminus E(G)$. In algorithm A_m, we turn to treat e as an edge and $q(S) = 1$; whenever a query after S contains e, the outcome is designated to be yes. After each pair of vertices is determined as edge or non-edge, return the hidden graph with e removed. Namely, we identify G by identifying some graph G' with $E(G') = E(G) \cup \{e\}$ where e is not an edge of G and is designated in the middle of excuting the algorithm A. It is clear that A_m also solves the hidden graph problem on $\mathcal{G}$.

Next, suppose that the hidden graph G has m edges. Then it is clear that any query S in the query history of the hidden graph G under algorithm A satisfies $|\binom{[n]}{2} \setminus E_S^-| \geq m$. Hence the query history is preserved to A_m. This implies $l_{A_m}(m) = l_A(m)$.

For $r \leq m$, let G_{r-1} be the worst case for A_m among all graphs with $r-1$ edges; namely, $l_{A_m}(G_{r-1}) = l_{A_m}(r-1)$. Let S be the first query satisfying $|E_S^+ \cup E_S^*| \leq m-1$ along the query history of G_{r-1} under A. Let G_r be the graph with $E(G_r) = E(G_{r-1}) \cup \{e\}$ where e is selected by the algorithm A_m as above. Then the query history of G_{r-1} is a part of G_r's under A_m. Hence, $l_{A_m}(r) \geq l_{A_m}(G_r) \geq l_{A_m}(G_{r-1}) = l_{A_m}(r-1)$. □

We are now ready to prove the main result on the lower bound.

Proof of Theorem 2.1 According to Lemma 2.2, $l_A(m) = l_{A_m}(m) \geq l_{A_m}(r)$ for any $r \leq m$. Then in T_{A_m} the depth of each leaf representing a hidden graph with at most m edges is at most $l_{A_m}(m)$. We have $l_A(m) = l_{A_m}(m) \geq \log\left(\sum_{i=0}^{m} \binom{\binom{n}{2}}{i}\right)$. Namely, $\left\lceil \log\left(\sum_{i=0}^{m} \binom{\binom{n}{2}}{i}\right) \right\rceil$ queries are required to identify a graph (with m edges) drawn from the family of all graphs with n vertices. □

Theorem 2.1 shows that the lower bound $\epsilon m(\log n - 2)$ cannot be achieved by any algorithm when the number of edges in the hidden graph is large.

3 Efficient learning of a general graph

To introduce our algorithm for identifying a hidden general graph, we start with some basic routines. Angluin and Chen [5] proved that

Lemma 3.1 *For any hidden hypergraph G on $[n]$ with each edge of size at most r, there is an algorithm that identifies an edge of G in $r\lceil \log n \rceil$ edge-detecting queries.*

A *binary splitting algorithm* [13] introduced to solve the classical group testing problem is as follows. Test the set of items. If the outcome is positive, test a set containing half of items, then we know that its outcome indicates either the tested half or the other half contains a positive item. Then iteratively halve the one with positives and test one of them until a positive item is obtained. Therefore, a binary splitting algorithm can identify all positive items in $d(\lceil \log n \rceil + 1) + 1$ group tests where d (unknown) is the number of positive items and the last 1 is contributed by the last test which ensures that there is no more positive item.

Let $G[S]$ denote the subgraph of G induced by a vertex set S and $G[X, Y]$ denote the bipartite subgraph of G induced by partite sets X and Y. A graph is called *non-trivial* if it contains at least one edge. The following result extended from the binary splitting algorithm plays a fundamental role in our main algorithm.

Lemma 3.2 *A bipartite graph G with partite sets X, Y can be identified in $|X| + m(\log |Y| + 2)$ edge-detecting queries where m (unknown) is the number of edges in*

Algorithm 1 IDENTIFY-HIDDEN-GRAPH

```
Step 1  Identify a maximal matching:
        Find a maximal matching $M$ of $G$ by iteratively identifying an edge and removing its two vertices
        from $G$ until no more edge can be found. There remains an independent set, say $I$.
        **Return**: $M$ and $I$.
Step 2  Decompose $G[[n] \setminus I]$ into bipartite subgraphs:
        **Initialization**: Take an edge $e = \{u, v\}$ from $M$ and let $B_1 = G[\{u\}, \{v\}]$ and $k = 1$.
        Remove $e$ from $M$.
        **Iteration**: If $M = \emptyset$, then stop the iteration; otherwise, remove a new edge $e = \{u', v'\}$ from $M$
        and let $i = 1$.
        $(*)$ If $i > k$, then let $k \leftarrow k + 1$ and $B_k := G[\{u'\}, \{v'\}]$, and go back to the beginning of the
        iteration; otherwise, for $B_i := G[X, Y]$, apply queries on $X \cup \{u'\}$, $X \cup \{v'\}$, $Y \cup \{u'\}$ and $Y \cup \{v'\}$.
        If either $q(X \cup \{u'\}) = 0 = q(Y \cup \{v'\})$ or $q(X \cup \{v'\}) = 0 = q(Y \cup \{u'\})$, then include $u'$, $v'$ into
        $X$ and $Y$ correspondingly such that $B_i = G[X, Y]$ is bipartite and then go back to the beginning
        of the iteration; otherwise, let $i \leftarrow i + 1$ and go back to $(*)$.
        **Return**: $B_1, \cdots, B_k$.
Step 3  Identify each $B_i := G[X_i, Y_i]$ by applying IDENTIFY-BIPARTITE on $G[\{u\}, Y_i \setminus \{v\}]$ for each
        $u \in X_i$ where $\{v, u\}$ is an edge in $M$.
Step 4  For any $B_i = G[X, Y]$ and $B_j = G[X', Y']$ with $1 \le i < j \le k$, apply IDENTIFY-BIPARTITE
        to $G[X, \{u\}]$, $G[Y, \{u\}]$ $G[X, \{v\}]$, and $G[Y, \{v\}]$ for every $u \in X'$ and $v \in Y'$ with $\{u, v\} \in M$.
Step 5  Apply IDENTIFY-BIPARTITE to $G[\{v\}, I]$ for each $v \in [n] \setminus I$.
```

G. In particular, if $1 = |X| \le |Y|$ and G is known to be non-trivial, $m(\log|Y| + 2)$ edge-detecting queries are sufficient.

Proof G can be identified by the following procedure: for each $u \in X$ apply binary splitting algorithm on Y with u included in each test. Overall, it takes at most $|X| + m(\lceil \log |Y| \rceil + 1)$ queries. For $1 = |X| \le |Y|$ and G is known to be non-trivial, a query on $\{u\} \cup Y$ can be saved if $X = \{u\}$. □

We denote the procedure that identifies edges in a bipartite graph mentioned in the proof of Lemma 3.2 by IDENTIFY-BIPARTITE. We now give a five-step algorithm IDENTIFY-HIDDEN-GRAPH (see Algorithm 1) to identify an arbitrary hidden graph G even though the number of edges in G is unknown. Our main strategy is to partition $[n]$, the set of vertices of G, into independent sets (see Step 1–2) and then identify the hidden subgraphs induced by all pairs of independent sets (see Step 3–5).

Theorem 3.3 *For the family $\mathcal{G}$ of all graphs on $[n]$, a hidden graph G drawn from $\mathcal{G}$ can be identified in $m(\log n + 4) + \alpha'(G)(\log n + 5) + 1$ edge-detecting queries where m is the number of edges in G and $\alpha'(G)$ is the maximum size of matching in G.*

Proof Denote the number of edges identified in Step i by m_i. Step 1 clearly returns a maximal matching M and an independent set I. Since identifying an edge can be done in $2\lceil \log n \rceil (\le 2\log n + 2)$ queries (Lemma 3.1), M is identified in $|M|(2\log n + 2) + 1$ queries where the last test is to ensure that the remaining graph is trivial after M is identified.

Before each iteration in Step 2, some bipartite graphs $B_1, B_2, \cdots, B_k$ are built. In the iteration, a new edge $e = \{u', v'\}$ is removed from M, and we find the smallest index i such that $B_i \cup \{e\}$ is bipartite; if no such a B_i exists, then the edge e is isolated as a new bipartite graph. Determining whether $B_i \cup \{e\}$ is bipartite for a bipartite graph $B_i = G[X, Y]$ can be accomplished by using the outcomes of the four queries $X \cup \{u'\}$,

$X \cup \{v'\}$, $Y \cup \{u'\}$ and $Y \cup \{v'\}$. Finally, Step 2 returns bipartite graphs $B_1, \cdots, B_k$, and $[n] \setminus I$ is successfully partitioned into independent sets which are partite sets of $B_1, \cdots, B_k$. To analyze the number of queries spent in Step 2, we consider the following two cases of pairs of e and B_i in the iteration:
Case 1 (e, B_i)s where e is not merged into B_i. Then w.l.o.g. $q(X \cup \{u'\}) = 1 = q(Y \cup \{u'\})$ or $q(X \cup \{u'\}) = 1 = q(X \cup \{v'\})$. In either case, at least two edges containing u' or v' will be identified in Step 4. Therefore, the total number of queries spent in this case for all possible pairs of B_i and e is at most $2m_4$.
Case 2 (e, B_i)s where e is merged into B_i. Since each edge e is merged into B_i at most once, at most $4|M|$ queries are spent in this case.

Hence, there are at most $2m_4 + 4|M|$ queries spent in Step 2.

After Step 1 and Step 2, all the remaining hidden edges are between I and the partite sets of $B_1, \cdots, B_k$ and can be classified into three types:

Type 1: Edges in some B_i but not in M,
Type 2: Edges in $G[X, X']$, $G[X, Y']$, $G[Y, X']$ and $G[Y, Y']$ for some $B_i := G[X, Y]$ and $B_j := G[X', Y']$,
Type 3: Edges containing a vertex in I.

Step 3–5 clearly identify edges of Type 1–3, respectively. Therefore, the correctness of IDENTIFY-HIDDEN-GRAPH is obtained. It remains to analyze the number of queries spent in Step 3–5. In Step 3, since $|Y_i \setminus \{v\}| < |M| \leq \frac{n}{2}$ for $i = 1, \cdots k$, by Lemma 3.2 the total number of queries spent here is at most $m_3(\log \frac{n}{2} + 2) + |X_1| + |X_2| + \cdots + |X_k| = m_3(\log n + 1) + |M|$. In Step 4, according to the construction of B_i and B_j (see Step 2), at least two of $G[X, \{u\}]$, $G[Y, \{u\}]$, $G[X, \{v\}]$, and $G[Y, \{v\}]$ are known to be non-trivial and thus at most two of them are trivial. Therefore, in Step 4 the number of trivial subgraphs that IDENTIFY-BIPARTITE is applied to is at most m_4. Hence by Lemma 3.2 Step 4 takes at most $[m_4(\log \frac{n}{2} + 2)] + m_4 \leq m_4 \log n + 2m_4$ queries. Finally, since $|[n] \setminus I| = 2|M|$, by Lemma 3.2 at most $2|M| + m_5(\log n + 2)$ queries are spent in Step 5.

Overall, IDENTIFY-HIDDEN-GRAPH identifies the hidden graph in

$$\begin{aligned}&(m_3 + m_4 + m_5)(\log n + 4) + |M|(2\log n + 9) + 1\\ &= m(\log n + 4) + |M|(\log n + 5) + 1\end{aligned}$$

queries where $m_3 + m_4 + m_5 + |M| = m$. Therefore, the theorem follows. □

Recall that an affine plane of order p contains p^2 points and $p^2 + p$ blocks of size p such that each pair of points occur together in exactly one of the blocks. Such an affine plane exists whenever p is a prime power. The affine plane method is to take an affine plane with the point set containing $[n]$ and then identify each subgraph induced by a block. Since each pair of points occurs together in exactly one of the blocks, the hidden graph can be determined by identifying the subgraphs induced by blocks. Each subgraph induced by a block has exactly p vertices. Therefore, to make the method efficient, we need a small prime power p satisfying $p^2 \geq n$. For $n > 24^2$, there is a prime p such that $n \leq p^2 \leq 2n$ (obtained from [17]; see [9,16]) and for $n \leq 24^2$, a prime power p satisfying $n \leq p^2 \leq 2n$ can be found easily. Hence,

Lemma 3.4 *For $n > 1$, there exists a prime power p such that $n \le p^2 \le 2n$.*

We then identify the graph induced by a block by using IDENTIFY-HIDDEN-GRAPH.

Theorem 3.5 *For the family $\mathcal{G}$ of all graphs on $[n]$, a hidden graph G drawn from $\mathcal{G}$ can be identified in $m \log n + 10m + 3n$ edge-detecting queries where m is the number of edges in G.*

Proof By the definition of affine plane, we know that each hidden edge appears in exactly one of the subgraphs induced by blocks. Thus the total number of edges identified from blocks is exactly m. Let $\mathcal{B}$ be the set of blocks. Since $\sum_{B\in\mathcal{B}} \alpha'(G[B]) \le m$ and the subgraph induced by a block has p ($\le \sqrt{2n}$) vertices, by Theorem 3.3 the hidden graph can be identified in

$$\begin{aligned}&\sum_{B\in\mathcal{B}} \big(||G[B]||(\log p + 4) + \alpha'(G[B])(\log p + 5) + 1\big)\\ &\quad \le m(\log p + 4) + m(\log p + 5) + (p^2 + p)\\ &\quad \le m \log n + 10m + 2n + \sqrt{2n}\end{aligned}$$

queries where $||G[B]||$ is the number of edges in $G[B]$. □

4 Conclusion

For the graph learning problem on the family of all graphs with n vertices, we first provide a better information-theoretic lower bound $\log\left(\sum_{i=0}^{m}\binom{\binom{n}{2}}{i}\right)$ (m is the number of edges in the hidden graph). We further provide a deterministic algorithm, IDENTIFY-HIDDEN-GRAPH, together with the affine plane method to identify the hidden graph in $m \log n + 10m + 3n$ edge-detecting queries. Furthermore, for some families of graphs of known topology, our algorithm exhibits the same economic performance as the algorithms presented in [9] do (see the following table).

	Hamiltonian cycles	Perfect matchings
Information lower bound	$n \log n$	$(1 + o(1))(\frac{n}{2} \log n)$
Chang et al. [9]	$n \log n + 10n$	$\frac{n}{2} \log n + 4n$
IDENTIFY-HIDDEN-GRAPH with affine plane method	$n \log n + 13n$	$\frac{n}{2} \log n + 8n$

Acknowledgments The authors would like to express their gratitude to the referees for their valuable comments and suggestions in improving the presentation of this paper. Partially supported by National Science Council, Taiwan under Grant NSC 99-2811-M-009-056 (H.-L. Fu and C.-H. Shih) and 100-2115-M-390-004-MY2 (H. Chang).

References

1. Aigner, M.: Combinatorial Search. Wiley, New York (1988)
2. Alon, N., Asodi, V.: Learning a hidden subgraph. SIAM J. Discrete Math. **18**, 697–712 (2005)
3. Alon, N., Beigel, R., Kasif, S., Rudich, S., Sudakov, B.: Learning a hidden matching. SIAM J. Comput. **33**, 487–501 (2004)
4. Anderson, I.: Combinatorial designs: construction methods. Ellis Horwood, New York (1990)
5. Angluin, D., Chen, J.: Learning a hidden hypergraph. J. Mach. Learn. Res. **7**, 2215–2236 (2006)
6. Angluin, D., Chen, J.: Learning a hidden graph using $O(\log n)$ queries per edge. J. Comput. Syst. Sci. **74**, 546–556 (2008)
7. Beigel, R., Alon, N., Apaydin, M.S., Fortnow, L., Kasif, S.: An optimal procedure for gap closing in whole genome shotgun sequencing. In: Proceedings of 2001 RECOMB, pp. 22–30. ACM Press, New York
8. Bouvel, M., Grebinski, V., Kucherov, G.: Combinatorial search on graphs motivated by bioinformatics applications: a brief survey. WG, LNCS 3787, pp. 16–27 (2005)
9. Chang, H., Chen, H.-B., Fu, H.L., Shih, C.H.: Reconstruction of hidden graphs and threshold group testing. J. Comb. Optim. **22**, 270–281 (2011)
10. Damaschke, P., Sheikh Muhammad, A.: Competitive group testing and learning hidden vertex covers with minimum adaptivity. Discrete Math. Algebra Appl. **2**, 291–311 (2010)
11. Damaschke, P., Sheikh Muhammad, A.: Bounds for nonadaptive group tests to estimate the amount of defectives. Discrete Math. Algebra Appl. **3**, 517–536 (2011)
12. Du, D.Z., Hwang, F.K.: Combinatorial Group Testing and its Applications, 2nd edn. World Scientific, Singapore (2000)
13. Du, D.Z., Hwang, F.K.: Pooling Designs and Nonadaptive Group Testing: Important Tools for DNA Sequencing. World Scientific, Singapore (2006)
14. Grebinski, V., Kucherov, G.: Reconstructing a Hamiltonian cycle by querying the graph: application to DNA physical mapping. Discrete Appl. Math. **88**, 147–165 (1998)
15. Grebinski, V., Kucherov, G.: Optimal reconstruction of graphs under the additive model. Algorithmica **28**, 104–124 (2000)
16. Hwang, F.K., Liu, Y.C.: Error-tolerant pooling designs with inhibitors. J. Comput. Biol. **10**, 231–236 (2003)
17. Nagura, J.: On the interval containing at least one prime number. In: Proceedings of the Japan Academy Series A, vol. 28, pp. 177–181 (1952)
18. Schlaghoff, J., Triesch, E.: Improved results for competitive group testing. Comb. Prob. Comput. **14**, 191–202 (2005)
19. Sorokin, A., Lapidus, A., Capuano, V., Galleron, N., Pujic, P., Ehrlich, S.D.: A new approach using multiplex long accurate PCR and yeast artificial chromosomes for bacterial chromosome mapping and sequencing. Genome Res. **6**, 448–453 (1996)
20. Tettelin, H., Radune, D., Kasif, S., Khouri, H., Salzberg, S.L.: Optimized multiplex PCR: efficiently closing a whole-genome shotgun sequencing project. Genomics **62**, 500–507 (1996)