5th Emléktábla Workshop

Combinatorial Search 08. 12-15. 2013.

Preliminary Schedule

Day 1:
9:29 Welcome
9:30 - 10:15 Ferdinando Cicalese, Universita degli Studi di Salerno
10:30 - 11:15 Peter Damaschke, Chalmers University
11:30 - 12:15 Ding-Zhu Du, University of Texas at Dallas
Lunch Break
14:00 - 14:45 Gyula O.H. Katona, Rényi Institute
15:30 from in front of Rényi: Traveling together to Erdőtarcsa by private bus.

Other Days:
9:29 Waking up
8:30 - 9:30 Breakfast
9:30 Partitioning to Groups of 3-5 for the day
9:30 - 12:30 Work in Groups of 3-5
12:30 - 14:00 Lunch Break
14:00 Optional Repartitioning for the afternoon
14:00 - 17:00 Work in Groups of 3-5
17:00 - 18:30 Discussion of Results
18:30 - Dinner and other activities

Last Day:
Discussion from after lunch and then Return to Budapest starting 16:00 (ETA 17:30).

List of Participants

Fabricio Benevides, Universidade Federal do Ceará
Endre Csóka, University of Warwick
Michelle Delcourt, University of Illinois at Urbana-Champaign
Victor Falgas-Ravry, Umea University
Dániel Gerbner, Rényi Institute
Dániel Korándi, UCLA and ETHZ
Balázs Keszegh, Rényi Institute
Younjin Kim, Korea Advanced Institute of Science and Technology
Teeradej Kittipassorn, University of Memphis
Sebastian Koch, Cambridge University
Mohit Kumbhat, Sungkyunkwan University, Korea
Shoham Letzter, Cambridge University
Bernard Lidicky, University of Illinois at Urbana-Champaign
Zoltán Nagy, Rényi Institute
Bhargav Narayanan, Cambridge University
Cory Palmer, University of Illinois at Urbana-Champaign
Dömötör Pálvölgyi, Eötvös University
Balázs Patkós, Rényi Institute
Alexey Pokrovskiy, London School of Economics
Ago-Erik Riet, University of Tartu
Blerina Sinaimeri, INRIA Lyon
Ágnes Tóth, BUTE and Rényi Institute
Tomas Valla, Czech Technical University in Prague
Máté Vizer, Rényi Institute
Dominik Vu, University of Memphis
Gábor Wiener, BUTE

Tree Search Problems

by Ferdinando Cicalese

Introduction

Given a set of objects $U = \{u_1, \ldots, u_n\}$, a probability distribution $p(\cdot)$ on U, a set of tests $\{t_1, \ldots, t_m\}$, with $t_i \subseteq U$, a cost function $c : \{t_1, \ldots, t_m\} \mapsto \mathbb{R}^+$ and a "hidden" marked object, the Binary Identification Problem (BIP) consists of defining a strategy (decision tree) for adaptively selecting a sequence of tests that uniquely identify the marked object minimizing the total cost spent in the worst case (or, alternatively, the expected cost when the marked object is chosen according to the the distribution p) [11]. A test t incurs a cost $c(t)$ and allows to determine whether the marked object is in the set t or in $U \setminus t$. The BIP is an $\mathcal{NP}$-Complete problem [13] that does not admit an $o(\log n)$-approximation unless $\mathcal{P} = \mathcal{NP}$ [13, 18]. On the other hand, a simple greedy algorithm attains an $O(\log n)$-approximation [1].

When we impose some structure in the set of tests we have interesting particular cases. If the set of tests consists of all the subsets of U (i.e., 2^U), then the strategy that minimizes the expected cost is a Huffman tree. Let G be a DAG with vertex set U. If the set of tests is $\{t_1, \ldots, t_n\}$, where $t_i = \{u_j | u_i \rightsquigarrow u_j \text{ in } G\}$, then we have the problem of searching in a poset [20, 16]. When G is a directed path we have the alphabetic coding problem [12].

We will focus on the version of the BIP in which the underlying space of objects and tests can be represented by the set of vertices and edges of a tree, respectively. Hence, c is a cost assignment to the edges $E(T)$ of T, i.e., $c : e \in E(T) \mapsto c(e) \in \mathbb{R}_0^+$. A decision tree for such an instance is a binary tree recursively defined as follows: if the tree T has only one vertex, then the decision tree is a single leaf labeled with the only vertex in T. If T has at least one edge, a decision tree for T has its root r labeled with one edge $e = \{u, v\}$ in T, and the subtrees rooted at the children of r are decision trees for the connected components T_u and T_v of $T - e$.

For the sake of distinguishing between the input tree and the decision tree, we shall reserve the term *node* to the decision tree and the term *vertex* to the input tree.

A decision tree D naturally defines a strategy for identifying an initially unknown vertex x from T via edge queries. If node w of D is labeled with the edge $e = \{u, v\}$ of T, we map w to the question "Is x in T_u or in T_v?", where T_u (resp. T_v) denotes the component of $T - e$ which contains u (resp. v). The search strategy now consists in starting with the query at the root of D and then recursively continuing with the subtree being a decision tree for the component indicated in the answer. Accordingly, each leaf ℓ of D is then labeled with the vertex of T uniquely identified by the sequence of questions and answers corresponding to the path from the root of D to ℓ.

Given a decision tree D, the cost $cost^D(u)$ of identifying a vertex u following the strategy defined by D is defined as the sum of the costs of the tests associated to the nodes on the unique path from the root of D to the leaf associated to u.

We consider two different measures of performance for a decision tree: The *worst identification cost* of D, is defined as the maximum over all $u \in U$ of the cost of identifying u using the decision tree D, i.e., in formulae:

$$worstcost(D) = \max_{u \in U} cost^D(u).$$

We also consider the *expected identification cost* of D defined as the expected cost of identifying an objects chosen in accordance to the distribution $\mathbf{p}(.)$ when we use the strategy associated to D,

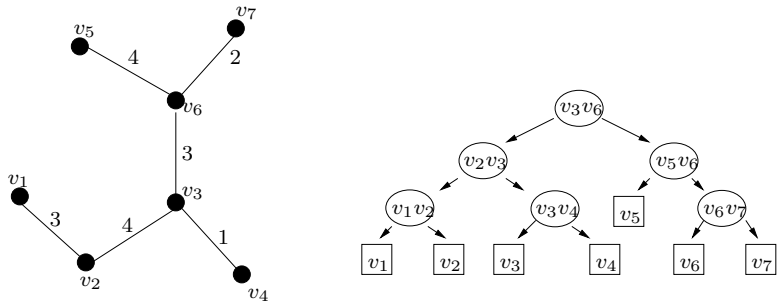

Figure 1: A tree T with edge costs (left hand side) and a decision tree D for it (right hand side). For example, identifying the vertex v_3 is done by queries to the edges $\{v_3, v_6\}$, $\{v_2, v_3\}$, and $\{v_3, v_4\}$. This identification process has a cost of $3 + 4 + 1 = 8$. The maximum cost of 10 has to be payed for the identification of v_2, and this is the worst identification cost of the depicted decision tree.

i.e., in formulae:

$$avgcost(D) = \sum_{u \in U} p(u) \cdot cost^D(u).$$

We refer to Figure 1 for an example.

BIP on Trees, The Edge Ranking Problem and Applications. The BIP for trees is equivalent to the edge ranking problem for trees. An edge ranking of T is an assignment to each edge e of T of an integer $r(e)$ (the rank of e) s.t. for any two edges $e_1, e_2 \in T$ if $r(e_1) = r(e_2)$, then the path connecting e_1 and e_2 contains an edge e with $r(e) > r(e_1)$. The cost of an edge ranking of a tree T with edge costs c, denoted by $\text{rankcost}(T, c)$ is defined as follows: If T has only one vertex, then $\text{rankcost}(T, c) = 0$. Otherwise, $\text{rankcost}(T, c) = c(e^*) + \max\{\text{rankcost}(T_u, c), \text{rankcost}(T_v, c)\}$, where $e^* = \{u, v\}$ is the edge with maximum rank in T and T_u (resp. T_v) is the connected component of $T - e$ that contains u (resp. v). Given a tree T with edge cost c the edge ranking problem asks for the minimum cost ranking. The equivalence to the decision tree problem is easily seen (see also [10]).

The BIP is a basic problem in computer science and has applications in many different scenarios. The BIP for trees arises when one has to identify the faulty component of a system. As an example, a system is represented by a network (in our case a tree) and its faulty component (vertex) has to be found. Different points of the network might require more or less expensive operations for the inspection. Inspecting one spot (edge) in the network reveals only directional information about the location of the failure w.r.t. the inspected point. One such problem is described, e.g., in [20] as searching for holes in an oil pipeline. In [23], the problem of finding a bug in a software application is mentioned.

The edge ranking problem arises in the context of multi-part product assembly [9, 14]. Assume that each edge represents the operation of assembling two parts of a product and the weight of an edge represents the time necessary to complete the corresponding assembly operation. Each product part can only participate in one assembly operation at a time, which means that, whenever two edges share an endpoint, the corresponding operations are dependent and cannot be performed simultaneously. An edge ranking provides a scheduling of the assembly operations with the guarantee that only independent operations are scheduled simultaneously. Moreover, the cost of the edge ranking is the total time necessary for completely assembling the product.

The Minimization of the Worst Identification Cost

The minimization of the worst identification cost for the BIP on trees with uniform edge costs has been extensively studied in the context of searching and edge ranking [14, 8, 19, 2, 22, 23]. Linear time algorithms that construct an optimal decision tree for the case of uniform costs are presented in [19, 22].

The case of non-uniform edge costs was also first studied in the context of edge ranking in [9], where the the problem was proved to be NP-complete already for the class of instances of diameter at most 10. In the same paper, an $O(\log n)$ approximation algorithm was also provided. In fact, the $O(\log n)$ approximation can be attained for the general version of the BIP (not restricted to tree instances), via a simple greedy procedure [4].

When the tree is a path, the BIP with non-uniform costs is equivalent to the problem of searching in an ordered array with costs depending on the position probed. A natural DP approach solves this problem in $O(n^3)$ time. A linear time algorithm with constant approximation factor is presented in [17]. In [3], Charikar *et al.* consider this problem from a competitive analysis perspective.

The complexity of finding a decision tree with minimum worst identification cost for trees is well characterized in terms of both the maximum degree and the diameter of the input tree as summarized in the following proposition.

Proposition 1. *The problem of determining a search strategy with minimum worst identification cost for diameter 6 trees is strongly NP-hard. The same complexity holds for degree 3 trees.*

There is an $O(n^2)$ time algorithm that finds a decision tree with the minimum worst identification cost for the problem of searching in a path.

For trees of diameter at most 5 there exists a polynomial time algorithm which determines a decision tree with minimum worst identification cost.

Although these result set the boundaries of tractability of the problem, the question about approximability in the general case remains wide open. The best result about algorithms for arbitrary trees is given by the following.

Theorem 2. *There is an $O(\log n/\log\log\log n)$-approximation algorithm for the minimization of the worst identification costs that runs in $O(n^2 \log n)$ times.*

This last result suggests a separation between the BIP for trees with non-uniform costs on the tests (queries) and the general BIP since the latter does not admit an $o(\log n)$ approximation unless $P = NP$. This motivates the following open problem.

Open Problem 1. *Is there a constant factor approximation algorithm for the problem of minimizing the worst case cost for general trees?*

The Minimization of the Average Number of Tests

We now consider the measure of performance based on the expected identification cost. Here we replace the probability distribution with a non-negative weight function $w : V \to \mathbb{R}^+$ that gives the likelihood of a vertex being the one marked and we focus on the case of unitary (or uniform) costs. Then, instead of the expected identification cost, we speak of the *average number of tests*, where the (weighted) average is with respect to the function w. It turns out that even if we assume unitary costs, the problem is $\mathcal{NP}$-complete and it is even hard to approximate in the more general case of posets.

Theorem 3. *[5, 6] The problem of minimizing the average number of tests for searching in bounded degree trees is $\mathcal{NP}$-complete. Moreover, in general posets the problem does not admit an $o(\log n)$ approximation unless $\mathcal{NP} \subseteq DTIME(n^{\log\log n})$.*

For bounded degree trees there is an FPTAS [7]: Let (T, w) denote an instance of the problem on the tree T where w is the likelihood function. Let $\Delta(T)$ denote the maximum degree of a vertex of T. Let $w(T)$ be the total weight of T i.e., the sum of the weights of all vertices of T and $w_{\min}$ be the minimum weight among all vertices of T. Then the following holds.

Theorem 4. *[7] There is a poly$(n \cdot \frac{w(T)}{w_{\min}})$-time algorithm for finding optimal search trees for the class of instances (T, w) with $\Delta(T) = O(1)$. In addition, for any $\epsilon > 0$, there is a poly(n/ϵ)-time algorithm for computing a decision tree whose average number of tests is at most a factor $(1 + \epsilon)$ from the minimum average number of tests achievable by any decision tree for the same instance.*

For this, we show a non-trivial Dynamic Programming based algorithm to computes the best possible search tree, among the search trees with height at most H, in $O(n2^{2H})$ time. Then, one shows that every tree T admits a minimum cost search tree whose height is at most $4\Delta(\log(w(T)/w_{min}) + 1) + (\Delta + 1)\log n$, where Δ is the maximum degree of T, $w(T)$ is the sum of the weights of the vertices in T and w_{min} is the minimum positive weight among all vertices of T. This bound allows us to execute the DP algorithm with $H = 4\Delta(\log(w(T)/w_{min}) + 1) + (\Delta + 1)\log n$, obtaining a pseudo-polynomial time algorithm for trees with bounded degree. By scaling the weights w in a fairly standard way we obtain the FPTAS.

It should be noticed that the bound on the height of optimal search trees generalizes the known logarithmic bound on the height of optimal search trees for totally ordered sets, and it is nearly tight since for a complete tree of degree Δ, where only the leaves have weight 1 and all other vertices have weight 0, it is not difficult to see that any search tree has height $\Omega\left(\frac{\Delta}{\log\Delta}\left(\log n + \log w(T)\right)\right)$.

Open Problem 2. *A natural question which remains open concerns the limit of approximability in the general case where we do not assume bounded degree tree instances: Is there a PTAS for unbounded degree instances or does the problem become APX-hard?*

Connected to this question is the performance of a natural greedy strategy.

The approximation of the Greedy Approach. For general trees and uniform costs, a simple greedy approach provides a 1.62-approximation of the minimum average number of queries.

The greedy choice is to always select an edge $e = \{u, v\}$ which minimizes $\max\{w(T_u), w(T_v)\}$, where T_u and T_v are the connected components of $T - e$.

It has been conjectured that Greedy guarantees approximation factor equal to 3/2. In fact, we can show that 3/2 is a lower bound on the approximation provided by Greedy. To this aim, we consider the class of input trees with the following structure. Fix an integer $m \geq 1$. The input tree T_m will have $2m + 1$ vertices. We denote the root of T_m by r. The root r has m children, which we denote by $a_1, \ldots, a_m$. For each $i = 1, \ldots, m$, the vertex a_i has only one child, which we denote by b_i. The root r has weight 1, and each vertex a_i $(i = 1, \ldots, m)$ has weight $\epsilon > 0$ while, for each $i = 1, \ldots, m$, the vertex b_i has weight 2^{i-1}.

In Fig. 2 we show the search tree produced by Greedy, denote by D_m (the rightmost tree) and an alternative search tree D^*_m, (the central tree) which has a smaller cost.

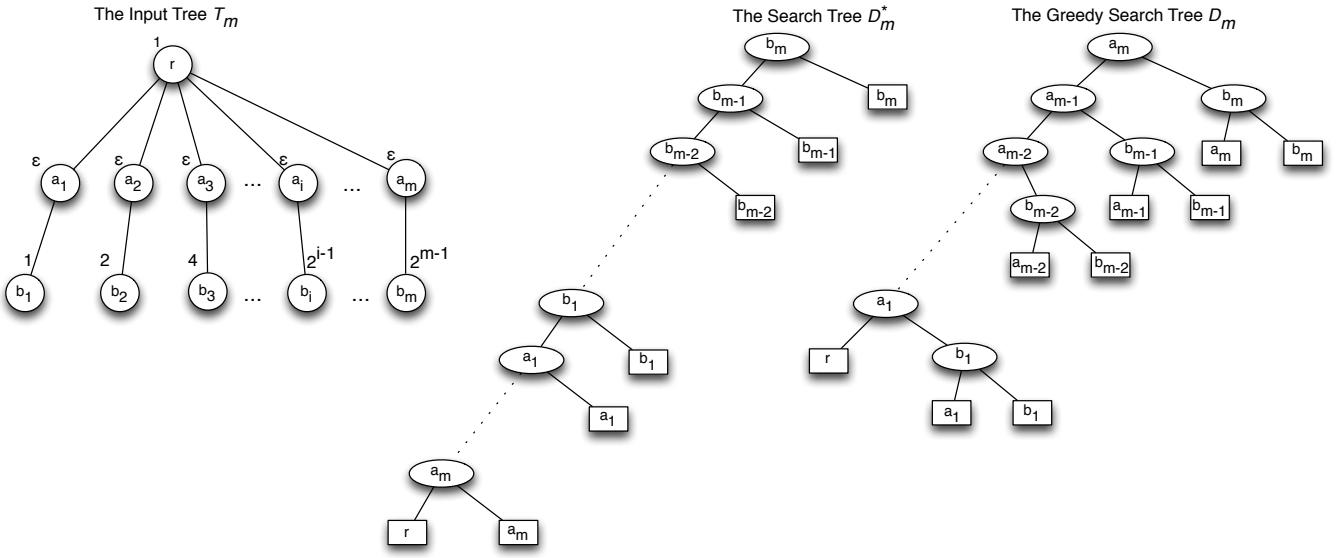

Figure 2: The input tree T_m (on the left), together with the search trees D_m produced by Greedy (on the right) and the search tree D^*_m (center) used to show the lower bound on the approximation of Greedy. In the decision trees, a test on the edge $e = \{u, v\}$, is indicated by the vertex in the edge queries which is further from the root of the input tree.

We have

$$cost(D_m) = \sum_{i=1}^{m}(i+1)(2^{m-i}+\epsilon) + m = 3 \cdot 2^m + \frac{\epsilon}{2}(m^2+3m) - 3 \quad (0.1)$$

$$cost(D^*_m) = \sum_{i=1}^{m} i2^{m-i} + \epsilon \sum_{i=m+1}^{2m} i + 2m = 2^{m+1} + \frac{\epsilon}{2}(3m^2+m) + m - 2. \quad (0.2)$$

Since $cost(D^*_m)$ is lower bounded by the cost of an optimal search tree for T_m, we have that $cost(D_m)/cost(D^*_m)$ is a lower bound on the approximation of Greedy. The claimed 3/2-approximation follows by considering the limit of the ratio $cost(D_m)/cost(D^*_m)$ for $\epsilon \to 0$ and $m \to \infty$.

Open Problem 3. *Can we prove that the approximation guarantee of the greedy strategy is* 3/2*?*

References

[1] E. M. Arkin, H. Meijer, J. S. B. Mitchell, D. Rappaport, and S. S. Skiena. Decision Trees for Geometric Models. *Intl. J. Comp. Geom. and Appl.*, 8(3):343–364, 1998.

[2] Y. Ben-Asher, E. Farchi, and I. Newman. Optimal search in trees. *SIAM Journal on Computing*, 28(6):2090–2102, 1999.

[3] M. Charikar, R. Fagin, V. Guruswami, J. M. Kleinberg, P. Raghavan, and A. Sahai. Query strategies for priced information. *J. of Comp. and System Sc.*, 64(4):785–819, 2002.

[4] F. Cicalese, T. Jacobs, E. Laber, and M. Molinaro. On Greedy Algorithms for Decision Trees In *Proc. of ISAAC 2010*, LNCS 6507, pp. 206–217, 2010.

[5] F. Cicalese, T. Jacobs, E. Laber, and M. Molinaro. On the Complexity of Searching in Trees: Average-Case Minimization In *Proc. of ICALP 2010*, LNCS 6198, pp. 527-539, 2010.

[6] F. Cicalese, T. Jacobs, E. S. Laber, and M. Molinaro. On the complexity of searching in trees and partially ordered structures. *Theoretical Computer Science*, 412:6879–6896, 2011.

[7] F. Cicalese, T. Jacobs, E. S. Laber, and M. Molinaro. Improved Approximation Algorithms for the Average-Case Tree Searching Problem. *Algorithmica*, to appear, DOI 10.1007/s00453-012-9715-6

[8] P. de la Torre, R. Greenlaw, and A. Schäffer. Optimal edge ranking of trees in polynomial time. *Algorithmica*, 13(6):592–618, 1995.

[9] D. Dereniowski. Edge ranking of weighted trees. *DAM*, 154:1198–1209, 2006.

[10] D. Dereniowski. Edge ranking and searching in partial orders. *DAM*, 156:2493–2500, 2008.

[11] M. Garey. Optimal binary identification procedures. *SIAM J. Appl. Math.*, 23(2):173–186, 1972.

[12] T. Hu and A. Tucker. Optimal computer search trees and variable-length alphabetic codes. *SIAM Journal on Applied Mathematics*, 21(4):514–532, 1971.

[13] L. Hyafil and R. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976.

[14] A. Iyer, H. Ratliff, and G. Vijayan. On an edge ranking problem of trees and graphs. *Discrete Applied Mathematics*, 30(1):43–52, 1991.

[15] D. Knuth. Optimum binary search trees. *Acta. Informat.*, 1:14–25, 1971.

[16] S. R. Kosaraju, T. M. Przytycka, and R. S. Borgstrom. On an optimal split tree problem. In *Proc. of WADS '99*, pp. 157–168, London, UK, 1999. Springer-Verlag.

[17] E. Laber, R. Milidiú, and A. Pessoa. On binary searching with non-uniform costs. In *Proc. of SODA'01*, pp. 855–864, 2001.

[18] E. Laber and L. Nogueira. On the hardness of the minimum height decision tree problem. *Discrete Applied Mathematics*, 144(1-2):209–212, 2004.

[19] T. W. Lam and F. L. Yue. Optimal edge ranking of trees in linear time. In *Proc. of SODA '98*, pp. 436–445, 1998.

[20] M. Lipman and J. Abrahams. Minimum average cost testing for partially ordered components. *IEEE Transactions on Information Theory*, 41(1):287–291, 1995.

[21] K. Makino, Y. Uno, and T. Ibaraki. On minimum edge ranking spanning trees. *J. Algorithms*, 38:411–437, February 2001.

[22] S. Mozes, K. Onak, and O. Weimann. Finding an optimal tree searching strategy in linear time. In *Proc. of SODA '08*, pp. 1096–1105, 2008.

[23] K. Onak and P. Parys. Generalization of binary search: Searching in trees and forest-like partial orders. In *Proc. of FOCS '06*, pp. 379–388, 2006.

A question on additivity of pool numbers in group testing

by Peter Damaschke

We are given n elements, where an unknown subset of elements are *positive*, also called *defective*. Let P denote this unknown subset of positive elements. A *group test* using a *pool* (subset) Q answers positively if $P \cap Q \neq \emptyset$, and negatively if $P \cap Q = \emptyset$. We also say that the pool Q is positive or negative, respectively. A *pooling design* is simply a set of pools. In the *strict group testing* (SGT) problem, some number d is also given beforehand, and a searcher has to identify P whenever $|P| \leq d$. In the case $|P| > d$ the searcher is only supposed to report the fact that $|P| > d$. The group tests may be arranged in s *stages*, where all pools of the pooling design within a stage must be tested simultaneously. We define $t(n, d, s)$ to be the optimal number of tests needed to solve the SGT problem on n elements, with the specified d, in s stages. In [1] we made an effort to figure out as many as possible $t(n, d, s)$ values.

The searcher's instantaneous knowledge prior to a stage is captured by the *response hypergraph* defined as follows. Its vertices are the *candidate elements*, i.e., those elements that have not yet appeared in negative pools. Its hyperedges are the positive pools seen so far (but only the candidate elements therein are kept). The *candidate hypergraph*, defined on the same vertex set, has as hyperedges all sets of size at most d which may still be the unknown P. We call them the *candidate sets*. (Note that there may exist candidate elements which belong to no candidate set!) The candidate sets are exactly the hitting sets, of size at most d, of the response hypergraph.

In the following we are concerned with the situation after $s - 1$ stages, that is, prior to the last stage. Given the candidate hypergraph G after $s - 1$ stages, we are obviously interested in the smallest pooling design that eventually solves the SGT problem in the final stage s. This is an important subtask in the calculation of $t(n, d, s)$ values. Let us call a pooling design *sufficient* for G if the answers to the group tests uniquely determine which candidate set equals P (or allow to report that P is none of them). Sufficient pooling designs are easy to characterize: A pooling design is sufficient for G if and only if the complement of every candidate set equals the union of some of the pools. (Another characterization in terms of certain graph colorings is given in [1].) We remark that this generalizes the well-known d-disjunctness property for the case $s = 1$.

In the analysis of $t(n, d, s)$ especially for "large" ratios d/n we have apparently often to deal with *pooling designs on disjoint subsets of elements*. This leads us to the **problem:** *Are the test numbers in the last stage additive, for two "independent" instances of SGT?* Below we state our problem more formally and detailed.

Let G_i be some given candidate hypergraph on a set V_i of elements, where $i = 1, 2$ and $V_1 \cap V_2 = \emptyset$. Let t_i be the optimal number of pools in a sufficient pooling design on G_i. We define the *product* $G = G_1 \times G_2$ to be the candidate hypergraph on $V_1 \cup V_2$ whose candidate sets are exactly all unions $C_1 \cup C_2$ of candidate sets $C_1 \subseteq V_1$ and $C_2 \subseteq V_2$. Let t be the optimal number of pools in a sufficient pooling design on G. Trivially we have $t \leq t_1 + t_2$. Is it always true that $t = t_1 + t_2$? If not, is it true up to some exceptions that we can characterize?

Intuitively one would expect the test numbers to be additive, but the catch is that the searcher is free to use mixed pools with elements from both V_1 and V_2, and perhaps save some tests in this way. (Also, the conjecture is not true for the adaptive variant of SGT.)

A solution to the problem would nicely extend our lower-bound toolbox in [1]. If the problem turns out to be easy, plan B is to continue on [1] and figure out more $t(n, d, s)$ values ...

References

[1] Peter Damaschke, Azam Sheikh Muhammad: A toolbox for provably optimal multistage strict group testing strategies. In: D.Z. Du, G. Zhang (eds.), *19th International Computing and Combinatorics Conference COCOON 2013*, Hangzhou, *LNCS* 7936, 446–457.

Pooling Design and Probe Selection

by Ding-Zhu Du and Weili Wu

1 Introduction

A few year ago, a newspaper reported the following: "The study suggests that some genes start working less hard after age 40, at least in some people." After age 40, "about 400 genes showed significant changes in how hard they had been working to instruct cell to make certain proteins." This is a typical research work about gene functions. The study on gene functions is getting more and more attentions. Computer technology has been extensively involved in such a study. Usually, this study requires a DNA library of high quality, which obtained through a large amount of testing and screening. Consequently, the efficiency of testing and screening becomes a crucial issue to the success and ultimately the impact of the study of gene functions. Indeed, some efficient method makes a big difference. For example, the Life Science Division of Los Alamos National Laboratories [71] reported that they were dealing with 220,000 clones and hence, testing those clones individually requires 220,000 tests. However, with a smart method, this number was reduced to 376 tests. The difference is really significant.

This smart method is called the *pooling design*. The pooling design is a special type of group testing, which is known as a mathematical search method since 1943 [27, 1]. Given a set of n items with at most d positive ones, the group testing tests subsets of items, called *pools*. For example, in the above mentioned testing at Los Alamos National Laboratories, each pool contains about 5,000 clones. The outcome of a test on a pool is *positive* if the pool contains a positive item, and is *negative* if the pool contains no positive item. When d/n is small, the possibility of a pool being negative is quite large so that one test can identify many items. This is the basic idea of group testing.

Group testing was studied starting from the Wassernan-type blood test in World War II [27] and obtained many developments later due to its various applications, such as multi-channel excess, chemical testing, software testing, etc. Especially, its significant applications in the study of molecular biology get more attentions [2, 4, 5, 6, 7, 8, 9, 10, 13, 14, 20, 28, 30, 37, 41, 43, 44, 47, 48, 50, 52, 54, 60, 63, 68, 69, 70, 73, 74, 75, 78, 81, 83, 84, 85, 86, 87, 89, 90, 91].

Consider a set of 16 items with at most one positive item. We can design the following two group testing algorithms.

The First Algorithm. Arrange 16 items into a 4×4 matrix. Each row or column represents a pool. Since there is at most one positive item, there exists at most one positive row and at most one positive column; if those positive row and column exist, then their intersection is the positive item. Therefore, we need 8 tests to identify the positive one (Fig. 1(a)).

The Second Algorithm. Bisect 16 items into two pools of size 8. Test both pools. If both are negative, then all items are negative. If there is a positive pool, then bisect the positive pool into two parts A and B and test A. If test-outcome is negative, then B is positive; otherwise, A is positive. Bisect the positive one again until the positive item is singled out. Thus, we need at most 5 tests to identify the positive one (Fig. 1(b)).

The first one is a nonadaptive algorithm. The second one is sequential. In general, a group testing is *nonadaptive* if all tests are arranged in a single stage, that is, no information on test outcomes is available for determining the composition of another test [29, 30]. The sequential group testing may use less number of tests than nonadaptive group testing while the nonadaptive group

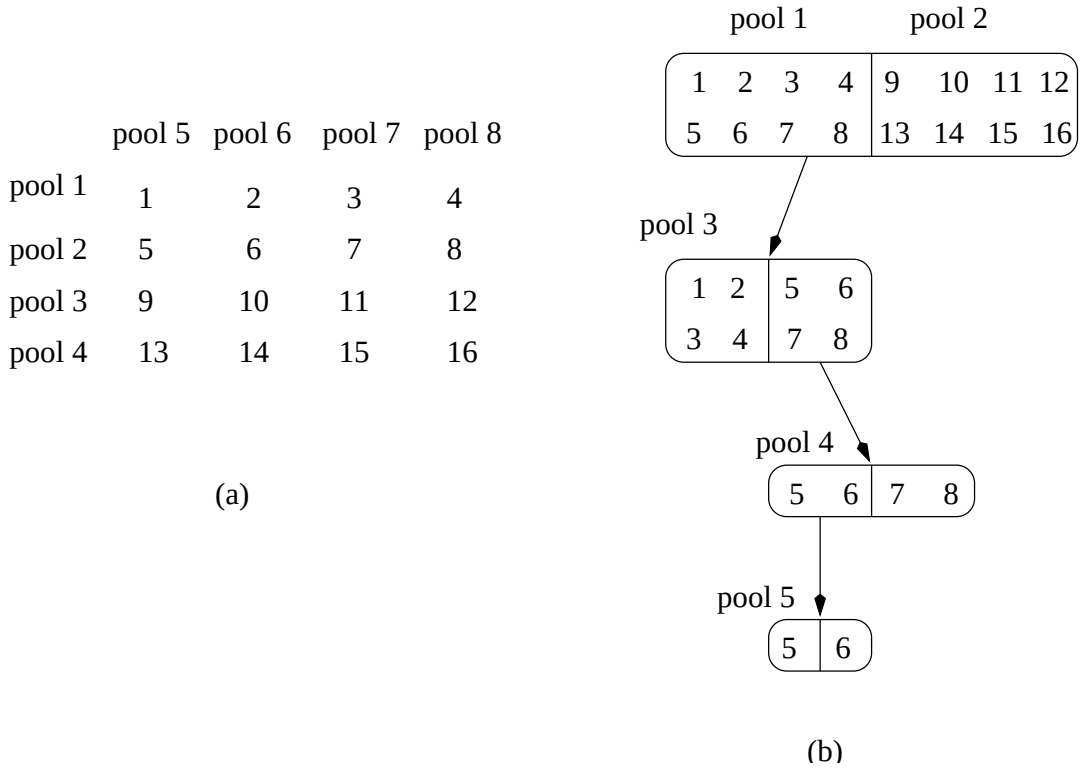

Figure 1: Nonadaptive and sequential group testing (assume item 5 is positive).

testing saves time. Since testing in molecular biology is often time-consuming, a pooling design is usually a nonadaptive group testing or a group testing with a small number of stages [6, 13, 67].

A nonadaptive group testing is often represented by a binary matrix with rows indexed with pools and columns indexed with items. A cell (i, j) contains a 1-entry if and only if the ith pool contains the jth item. This binary matrix is called the *incidence matrix* of the represented group testing.

Note that a pool is positive if it contains a positive item. If each column (item) is considered as a set of pools with 1-entry in the column, then the union of all columns indexed with positive items is the set of all positive pools, called *test-outcome*. Therefore, for a binary matrix representing a nonadaptive group testing which can identify up to d positive items, all unions of up to d columns should be distinct. The binary matrix with this property is said to be $\bar{d}$*-separable*.

Finding optimal or near-optimal algorithm to minimize the number of tests is a classical research problem in mathematical search theory. For group testing, such a problem is very hard to solve. Indeed, the complexity of finding optimal group testing is a long-standing open problem. We only know that the decision version of group testing belongs to PSPACE and is conjectured to be PSPACE-complete [33].

Given a test-outcome with a $\bar{d}$-separable matrix, how to determine all positive items? A naive way is to compare the given test-outcome with all unions of up to d columns. This takes $O(n^d)$ time. Ming Li (mentioned in [29]) showed that no (decoding) algorithm for $\bar{d}$-separable matrix can be done in polynomial-time with respect to n and d unless NP=P. This means that this naive way is actually the best way.

Indeed, decoding is equivalent to finding a subset of at most d items hitting every positive pool. Therefore, its computational complexity is reduced to the well-known NP-hard hitting set problem. However, the input size of this hitting set problem is controlled by the union of negative pools since

every item in a negative pool is negative and hence can be removed from consideration of decoding. When no column can be contained by a d-union, every negative item must appear in a negative pool. Therefore, decoding will become quite easy since all items not appearing in the union of all negative pools are positive. The binary matrix with the above property is said to be *d-disjunct*. For $t \times n$ d-disjunct matrix, the decoding time is $O(nt)$. A good design of d-disjunct matrix may give $t = O(d^2 \log n)$.

Recently, there is a surprising development in decoding of test-outcome obtained from nonadaptive group testing. This development was made by Indyk, Ngo and Rudra [57] through the study of concatenated code which has a close relationship with transversal design (see next section). They designed a type of nonadaptive group testing whose outcome can be decoded within time in a polynomial with respect to t, the number of tests. This design is called *efficiently decodable* group testing. Here, we would like to propose the study of efficiently decodable group testing with connection to pooling design and probe selection as follows.

- In efficiently decodable two or three-stage group testing, introduce the competitiveness in order to reduce the number of tests or to increase efficiency of pooling design.
- Extend the efficiently decodable nonadaptive group testing to complex model which has many application in molecular biology.
- For an efficiently decodable nonadaptive group testing in hand, how to reduce the number of tests further. This problem is closely related to the probe section problem.

In the following sections, we will give a detail description for proposed research and background in each identified problem.

2 Transversal Design and Efficiently Decoding

Nonadaptive group testing has many applications in molecular biology, such as DNA library screening, gene detection, physical mapping and contig sequencing [30]. In many cases, an item is a clone (a small piece of DNA) and the test is performed through hybridization.

In fact, a DNA consists of two *strands*. Each strand is a sequence of four types of nucleotides, A, C, G, T. The two strands stay together in double helix with a complementary relation $A \leftrightarrow T$ and $C \leftrightarrow G$, that is, one strand can be obtained from the other by changing A to T, T to A, C to G and G to C. The two strands can be separated under proper heating, but they have a tendency to bind to each other when put together. This binding tendency is known as *hybridization*.

The hybridization can be used to find out whether a DNA strand D contains a specific substrand S. Let S^{-1} denote the complementary strand of S. Mix D and S^{-1}. If D contains S, then S^{-1} will hybridize with S. This hybridization effect is magnified by the PCR (polymer chain reaction) technique so that it becomes observable or measurable. A PCR is essentially a technology to make copies of existing DNA pieces by using hybridization. The availability of multiple copies is a basic condition for application of group testing.

A nonadaptive pooling design is called a *transversal design* if all pools can be divided into disjoint families, each of which is a partition of all items (pools in different parts are disjoint). In a transversal design, all items need the same number of copies. Thus, it is easy to implement and hence is used very frequently in practice.

A transversal design can be represented by a q-nary matrix. A matrix is q-nary if every entry is chosen from $\{0, 1, ..., q-1\}$. Let rows be indexed by families and columns indexed by items. A cell (i, j) contains entry k if and only if item j belongs to the kth pool in the ith family. This matrix representation is called a *transversal matrix* of the transversal design. For example, matrix

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 2 & 2 \\ 2 & 1 & 0 & 1 & 0 & 2 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

represents transversal design

$$\begin{array}{lll} \{1,2\}, & \{3,4\}, & \{5,6\}; \\ \{3,5\}, & \{2,4\}, & \{1,6\}; \\ \{1,4,5\}, & \{2,3,6\}. & \end{array}$$

Its incidence matrix is

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

A transversal design is d-disjunct if and only if its transversal matrix satisfies the following property: no column is contained in the union of d other columns where the union of d column vectors is defined to be a column vector with each component being the union of the corresponding components of those d column vectors. Therefore, a q-nary matrix with this property is also called a *d-disjunct* q-nary matrix.

In [32], Du *et al.* gave a construction of the d-disjunct q-nary matrix as follows: Let $GF(q)$ be a finite field of order q and k a positive integer satisfying

$$n \leq q^k \tag{2.1}$$
$$f = d(k-1)+1 \leq q. \tag{2.2}$$

Construct an $f \times n$ matrix $M(d, n, q, k)$ by labeling columns with n distinct polynomials of degree $k-1$ over the finite field $GF(q)$ and labeling rows with f distinct elements of $GF(q)$, and by putting entry $g(x)$ into cell (x, g). Then $M(d, n, q, k)$ is a d-disjunct q-nary matrix. In fact, suppose to the contrary that $M(d, n, q, k)$ is not d-disjunct. Then it has a column g_0 contained in the union of other d columns $g_1, ..., g_d$, that is, for each row index x_i, $g_0(x_i) = g_j(x_i)$ for some j. Note that there are $d(k-1)+1$ rows. Thus, there exists a g_j $(1 \leq j \leq d)$ such that $g_0(x_i) = g_j(x_i)$ for at least k row indices x_i. By Lagrange Interpolation Formula, a polynomial of degree $k-1$ can be completely determined by its values at k distinct places. It follows that $g_0 = g_j$, a contradiction.

The following theorem is established in [32].

Theorem 1. *For above chosen q and k, $M(d, n, q, k)$ is a d-disjunct $f \times n$ q-nary matrix where*

$$f \leq q_0 = (1 + o(1)) \frac{d \log_2 n}{\log_2(d \log_2 n)}$$

and

$$q \leq 2q_0.$$

Therefore, $M(d, n, q, k)$ represents a transversal design with at most

$$(2 + o(1))(\frac{d \log_2 n}{\log_2(d \log_2 n)})^2$$

tests.

This is significantly better than other constructions for transversal designs in the literature, such as the k-dimensional grid design [4,48,53], in which the number of tests can be $O(dn^{1/k})$, the Chinese remainder sieve [43], in which the number of tests is $O(\frac{(d \ln n)^2}{\ln(2d \ln n)})$, and constructions in [74, 78].

In [32], Du *et al.* also proposed a multiplication theorem as follows.

Theorem 2. *If there exist a d-disjunct $f \times n$ q-nary matrix M_1 and a d-disjunct $f' \times q$ q'-nary matrix M_2, then there exists a d-disjunct $ff' \times n$ q'-nary matrix M_3.*

Actually, M_3 can be constructed from M_1 and M_2 by labeling columns of M_2 with $0, 1, ..., q-1$ and replacing each entry of M_1 by a corresponding column of M_2, denoted by $M_3 = M_1 \times M_2$. Note that M_3 gives a transversal design with $ff'q'$ tests. If $f'q' < q$, then $ff'q' < fq$. Du and Hwang [30] indicated that by using this multiplication theorem repeatedly, the number of tests may be reduced to approach the lower bound

$$\frac{d^2 \log_2 n}{\log_2 d}$$

for nonadaptive group testing [38,39,40].

Indyk, Ngo and Rudra [57] introduced a new concept, *list disjunct* matrix. A binary matrix is (d, ℓ)-*list disjunct* if for any two disjoint column subsets S and T with $|S| \leq d$ and $|T| \geq \ell$, there exists a row on which every column in S has entry 0 and at least one column in T has entry 1. Clearly, the (d, ℓ)-list disjunctness for $\ell \geq 2$ is weaker than the d-disjunctness. Indyk, Ngo and Rudra proved the following interesting facts about the list disjunctness.

1. There exist (d, d)-list disjunct binary matrices with $O(d \log n)$ rows.
2. There exists a d-disjunct $t \times n$ matrix $M_3 = M_1 \times M_2$ such that M_2 is a (d, d)-list disjunct $O(d \log q) \times q$ binary matrix and M_1 is a q-nary matrix with $O(\frac{d \log n}{\log q})$ rows.

Moreover, M_1 can be list recoverable[1], which enable M_3 to have a decoding algorithm with running time $poly(t)$, i.e., in polynomial with respect to t. Therefore, they proved the following.

Theorem 3. *There exist d-disjunct $t \times n$ matrices with decoding time $poly(t)$.*

This is a surprising result compared with classic decoding time $O(nt)$ for the d-disjunct matrix. This efficiently decodable nonadaptive group testing has been applied to data security [57] and compressed sensing [77]. It has also been extended to the error-tolerant case [76].

3 Proposed Research

In this section, we propose some research problems on efficiently decodable group testing related to applications in molecular biology.

[1]This concept is from coding theory. Actually, a q-nary $t \times n$ matrix can be seen as a q-nary code of n items; each code word has length t.

3.1 Highly Nonadaptive Competitive Group Testing

The competitive group testing was first proposed by Du and Hwang [31] in 1993. For a sample of n items with d positive ones, let $M(d,n)$ denote the minimum number of tests required in the worst case when d is known. Let $M_A(d|n)$ denote the number of tests performed by Algorithm A in the worst case when applied on such a sample of n items with d positive ones. Then algorithm A is called a *c-competitive* group testing if

$$M_A(d|n) \leq cM(d,n) + a$$

for all $0 \leq d < n$ where a and c are two constant. The constant c is called the *competitive ratio*. The advantage of the competitive group testing is that without knowing the actual number of positive items or its upper bound, the number of tests is competitive with the worst case possible upper bound $M(d,n)$. Since $M(d,n) = d\log\frac{n}{d} + o(1)$, the number of tests is decreased as d, the number of positive items in an actual sample, becomes smaller.

The algorithm proposed by Du and Hwang [31] is a 2.75-competitive algorithm. Bar-Noy *et al.* [3] gave a 2-competitive algorithm. The competitive ratio was further improved by Du *et al.* [36] to 1.65 and by Schlaghoff and Triesch [83] to $1.5+\varepsilon$ where $0 < \varepsilon < 0.01$. If randomized algorithms are considered, Damaschke and Sheikh Muhammad [24] showed that one can achieve nearly $d\log n$ tests even with highly nonadaptive group testing strategy. This motivates us to study the following problem.

Problem 1. *Can we design efficiently decodable 2-stage or 3-stage competitive group testing algorithms? Or, can we design efficiently decodable highly nonadaptive randomized group testing?*

From the previous section, we know that for applications in molecular biology, we favor nonadaptive group testing. However, there does not exist a competitive nonadaptive group testing since in nonadaptive group testing, the number of tests is fixed at the beginning which depends on the estimation of the upper bound of the number of positive items. If this estimation is far from the actual number, we may waste a lot of tests. Therefore, it may be worth to add one or two stages or a certain level of randomness to introduce the competitiveness into efficiently decodable group testing.

Du and Park [35] introduced another type of competitiveness, the strongly competitive group testing. They also gave two strongly competitive algorithms. Cheng *et al.* [15,18] proposed some improvements. Introducing the strong competitiveness into efficiently decodable group testing may also be an interesting research subject.

3.2 Error-Tolerant Testing

We may also involve the error-tolerant issue in the study of Problem 1. It was Macula [65] who first proposed the d^e-disjunct matrix, an error-tolerant version of d-disjunct matrix, which has attracted a lot of attention [55,66,78,91,76].

In fact, error-tolerance is an important issue in coding theory [59,80]. So it is in pooling designs [55,66,76,78,91] since biological tests may not be 100% reliable. Actually, a nonadaptive pooling design is also called a superimposed code [38,40,51,61]. It was not too hard to extend the efficiently decodable nonadaptive group testing to error-tolerant one [76]. However, it may be a challenging job to involve the presence of inhibitors [25,28,30]. Indeed, in the presence of inhibitors, decoding is an important issue and classic decoding methods are already not so simple. Therefore, the following problem is interesting to study.

Problem 2. *Can we extend the efficiently decodable method to the case when inhibitors are present?*

3.3 Extend to Complex Model

When the positiveness of a test is caused by a subset of items, but not a single item, we need to consider complex model [30]: Given a set of n items and at most d positive subsets of items, identify all positive subsets with the minimum number of tests; each test is on a pool (subset of items) and the test outcome is positive if and only if the pool contains a positive subset.

A pooling design is *d-disjunct* for a complex model if for any sample, every negative subset is contained in at least one negative pool, that is, all positive subsets can be identified by excluding all subsets contained in negative pools. The complex model is said to be *r-uniform* if all positive subsets have the same cardinality r. For r-uniform complex model, a pooling design is d-disjunct if and only if its binary incidence matrix satisfies the following conditions: For any r columns $C_1, ..., C_r$ and any other d columns $C_{r+1}, ..., C_{r+d}$, there exists a row R_i such that $C_1, ..., C_r$ have 1-entries in row R_i and $C_{r+1}, ..., C_{r+d}$ have 0-entries in row R_i. A binary matrix satisfying such a property is called a *$(d, r]$-disjunct* matrix [18]. We may also introduce the transversal designs to the complex model as well the $(d, r]$-disjunct q-nary matrix corresponding necessary and sufficient condition for it to represent a $(d, r]$-disjunct transversal design.

Problem 3. *Can we extend the efficiently decodable nonadaptive group testing to the r-uniform complex model?*

This is not an easy job. Actually, we may need first to solve the following problem.

Problem 4. *Can we extend the construction of Du* et al. *for the d-disjunct matrix to the r-uniform complex model? Can we extend the construction of Indyk-Ngo-Rudra for the (d, d)-list disjunct matrix to the r-uniform complexity model?*

Eextending the construction of Du *et al.* for the d-disjunct matrix to the $(d, r]$-disjunct matrix is already not an easy job. Gao *et al.* [44] made an attempt. The result is not satisfied because the number of tests is far from a known lower bound. Actually, we intend to find a new extension.

Indeed, there is another possible interesting way. To explain it clearly, let us consider 2-uniform complex model, that is, each positive subset contains two items. The 2-uniform complex model is exactly the group testing in graphs [1,21,22,44,46,58,63]. It has an application in protein-protein interactions [63,87].

Let M be a $(d, 2]$-disjunct q-nary matrix M. Then M should have the following property: For any $d+1$ distinct pairs of columns $\{C_1, C_1'\}, ..., \{C_{d+1}, C_{d+1}'\}$, there exists a row at which the union of C_1 and C_1' is disjoint from the union of C_2 and C_2', ..., the union of C_{d+1} and C_{d+1}'.

Let $\mathcal{P}$ be a collection of polynomial of degree $k-1$ on a finite field $GF(q)$ such that all products of two polynomials in the family are distinct. Construct a q-nary matrix M by using polynomials in $\mathcal{P}$ for column labels and $2kd+1$ distinct elements of $GF(q)$ for row labels; for each cell (x, g), put entry $g(x)$. Then M satisfies required property.

How do we design family $\mathcal{P}$? An idea is to replace polynomials of degree $k-1$ by polynomials of degree k with the coefficient of the first term being 1. Since such a polynomial of degree k can be determined by its k roots and the product of two polynomials is determined by the union of two root sets, we can reduce the problem to find a collection of k-multi-subsets of $GF(q)$ such that all unions of two k-multi-subsets in the collection are different. For simplicity, we may find a collection of k-subsets of $GF(q)$ such that all unions of two k-subsets in the collection are different. This means that the transpose of the incident matrix of the collection is a 2-separable matrix. Therefore, if we design a 2-disjunct $q \times h$ matrix with every column having a constant weight k, then we obtain a family $\mathcal{P}$ of h members.

It is not so easy to apply the above idea to general r, which is why it is interesting to study.

3.4 Probe Selection

A probe is a short oligonucleotide of size 8-25, used for identifying viruses (or bacteria) in a biological sample through hybridization, where a biological sample is a biological object, such as blood, containing a subset of viruses. When each probe hybridizes to a unique virus, identification is straightforward.

Although temperature and salt concentration are very helpful for a probe to hybridize uniquely to its intended virus, finding unique probes for given set of viruses is still a difficult task, especially for closely related virus subtypes. Schilep, Torney and Rahman [83] proposed a group testing method to use non-unique probes to find which viruses are in a biological sample. They consider a set of viruses hybridized to a probe as a pool and the incidence matrix of probes and viruses as a pooling design.

A pool yields a positive outcome if it hybridizes to a virus in the biological sample. The problem here differs from those in previous chapters in the fact that the candidate set of probes is more or less determined, not subject to our design as in previous chapters. What we can do is to construct a pooling design by selecting a subset of the candidate set.

If the incidence matrix is $\bar{d}$-separable, then the presence of up to d viruses in a given biological sample can be determined with those non-unique probes. Their methodology contains three steps:

Step 1. Find a large set of non-unique probes.

Step 2. From the set of probes obtained in Step 1, find a minimum subset of probes to identify up to d viruses.

Step 3. Decode the presence or absence of viruses in the given biological sample from testing outcomes.

The minimization problem in Step 2 can be formulated as follows:

> MIN-$\bar{d}$-SS (Minimum $\bar{d}$-Separable Submatrix). Given a $\bar{d}$-separable binary matrix M, find a minimum $\bar{d}$-separable submatrix with the same number of columns.

Since for $\bar{d}$-separable matrix, it is hard to decode at Step 3, we may use d-disjunct matrix or efficiently decodable d-disjunct matrix instead, which introduce the following two minimization problem:

> MIN-d-DS (Minimum d-Disjunct Submatrix). Given a d-disjunct binary matrix M, find a minimum d-disjunct submatrix with the same number of columns.

> MIN-d-EDDS (Minimum Efficiently decodable d-Disjunct Submatrix). Given an efficiently decodable d-disjunct binary matrix M, find a minimum efficiently decodable d-disjunct submatrix with the same number of columns.

These two problems are unlikely polynomial-time solvable. In fact, for any fixed d, MIN-d-DS is NP-hard. However, the complexity of MIN-d-EDDS is hard to determine. Therefore, we have a research problem as follows.

Problem 5. *What is the computational complexity of* MIN-d-EDDS*? Design good approximations for it.*

There are many research works already existing in the literature on MIN-$\bar{d}$-SS, MIN-d-DS and a related problem as follows.

MIN-d-SS (Minimum d-Separable Submatrix). Given a d-separable binary matrix M, find a minimum d-separable submatrix with the same number of columns.

We may get some ideas from those existing works to study MIN-d-EDDS.

For $d = 1$, MIN-d-SS is exactly the well-known minimum test cover problem [26] (also called the minimum test set problem [45] or the minimum test collection [49]). Minimum test cover problem has a greedy approximation with performance $1 + 2\ln n$ where n is the number of items [45]. This suggests us to study greedy approximations for MIN-d-EDDS.

For MIN-d-DS, it is not hard to obtain greedy approximations with performance ratio $1 + (d + 1)\ln n$. In fact, consider the collection $\mathcal{S}$ of all possible pairs (C, D) of one column C and a subset D of d columns. Clearly $|\mathcal{S}| < n^{d+1}$. A row is said to *cover* such a pair (C, D) if at this row, the entry of column C is 1 and all entries of columns in D are 0. Now, we choose rows one by one to maximize the total number of pairs newly covered by the row. This is a special case of the minimum set cover problem. It is well-known that the greedy algorithm for the minimum set cover has performance ratio $1 + \ln|\mathcal{S}| < 1 + (d + 1)\ln n$.

This greedy algorithm works well only for small d because its running time is $O(n^{d+1})$. When d is large, it is too slow. Therefore, we must look for other smart ways. Schilep, Torney and Rahman [84] proposed greedy algorithm which adds probe one by one until the incidence matrix with considered viruses form a $\bar{d}$-separable matrix. This doesn't work for large d, neither. In fact, if d is not bounded, then testing whether a binary matrix is d-separable, or $\bar{d}$-separable, or d-disjunct is co-NP-complete. There exist other methods [62] in the literature, which work well for small d. However, no efficient method has been found to produce good solutions for larger d. This situation suggests that the following problem is challenge.

Problem 6. *Design good greedy approximation algorithms for* MIN-d-EDDS*, especially in case that d is larger.*

In some applications, the pool size cannot be too big due to the sensitivity of tests. For example, UNH suggested in ADS testing, each pool should not contain more than five blood samples [80]. When the pool size is bounded, the problem becomes easier. For instance, let us consider the case that every pool has size at most 2 so that all pools of size 2 together with items form a graph G where pools are edges and item are vertices. Halldórsson *et al.* [49] and De Bontridder *et al.* [26] proved that in this case, MIN-1-SS is still APX-hard, which means that there is no polynomial-time approximation scheme for it unless NP=P. They also showed that MIN-1-SS in this case has a polynomial-time approximation with performance ratio $7/6 + \varepsilon$ for any fixed $\varepsilon > 0$.

An interesting result was showed by Wang *et al.* [89] that a subgraph H of G represents a d-disjunct matrix if and only if every vertex in H has degree at least $d + 1$ and hence finding such an H with minimum number of edges is polynomial-time solvable. What about the case when all pools have size 3? Wang *et al.* proved that in this case MIN-d-DS is still NP-hard. Motivated from above research, we propose to study the following problem.

Problem 7. *When the pool size is bounded, is there a good approximation for* MIN-d-EDDS*?*

References

[1] M. Aigner, Search problems in graphs, *Disc. Appl. Math.*, 14 (1986) 215-230.
[2] D. J. Balding, et al., A comparative survey of non-adaptive pooling designs, *Genetic Mapping and DNA Sequencing*, 133-155, IMA Volumes in Mathematics and its Applications, Springer, Berlin, 1995.

[3] A. Bar-Noy, F.K. Hwang, I. Kessler, and S. Kutten, Competitive group testing in high speed networks, *Discrete Applied Mathematics* 52 (1994) 29-38.

[4] E. Barillot, B. Lacroix and D. Cohen, Theoretical analysis of library screening using a N-dimensional pooling strategy, *Nucleic Acids Res.* 19 (1991 6241-6247.

[5] R. Beigel, N. Alon, M. S. Aprydin, L. Fortnow and S. Kesif, An optimal procedure for gap closing in whole genome shotgun sequencing, *Proc. RECOMB*, ACM Press, 2001, pp. 22-30.

[6] T. Berger, J.W. Mandell and P. Subrahmanya, Maximally efficient two-stage screening, *Biometrics* 56 (2000) 833-840.

[7] Morgan A. Bishop, Anthony J. Macula, Thomas E. Renz, Vladimir Ufimtsev: Hypothesis group testing for disjoint pairs. *J. Comb. Optim.* 15(1): 7-16 (2008)

[8] W. J. Bruno, et al., Design of efficient pooling experiments, *Genomics* 26 (1995) 21-30.

[9] P. Berman, B. Dasgupta and M.-Y. Kao, Tight approximability results for test set problems in bioinformatics, *Journal of Computer and System Sciences*, 71 (2005) 145-162.

[10] W.W. Cai, C.W. Chow, S. Damani, G. Simon and A. Bradley, A SSLD anchored BAC framework map of the mouse genome, *Nat. Genet.*, to appear.

[11] Yongxi Cheng, An efficient randomized group testing procedure to determine the number of defectives, *Operations Research Letters*, 39 (2011) 352-354.

[12] Yongxi Cheng, **Ding-Zhu Du**, Guohui Lin: On the upper bounds of the minimum number of rows of disjunct matrices, *Optimization Letters* 3(2): 297-302 (2009).

[13] Yongxi Cheng, **Ding-Zhu Du**: New Constructions of One- and Two-Stage Pooling Designs *Journal of Computational Biology* 15(2): 195-205 (2008).

[14] Yongxi Cheng, **Ding-Zhu Du**: Efficient Constructions of Disjunct Matrices with Applications to DNA Library Screening, *Journal of Computational Biology* 14(9): 1208-1216 (2007).

[15] Yongxi Cheng, **Ding-Zhu Du** and Yinfeng Xu, A zig-zag approach for competitive group testing, to appear in *INFORMS Journal on Computing*.

[16] Yongxi Cheng, Ker-I Ko, **Weili Wu**: On the complexity of non-unique probe selection. *Theor. Comput. Sci.* 390(1): 120-125 (2008)

[17] Yongxi Cheng and Yinfeng Xu, An efficient FPRAS type group testing procedure to approximate the number of defectives, *Journal of Combinatorial Optimization* online 2012.

[18] Yongxi Cheng, **Ding-Zhu Du** and Feifeng Zheng, A new strongly competitive group testing algorithm with small sequentiality, submitted for publication.

[19] C. Colbourn, D. Hoffman and R. Reed, A new class of group divisible designs with block size three, *J. Combin. Thy.* 59 (1992) 73-89.

[20] C. Colburn, A. Ling and M. Tompa, Construction of optimal quality control ogilio arrays, *Bioinformatics* 18 : 4 (2002) 529-535.

[21] G. J. Chang and F. K. Hwang, A group testing problem, *SIAM J. Alg. Disc. Meth.*, 1 (1980) 21-24.

[22] T. Chen and F. K. Hwang, A competitive algorithm in searching for many edges in a hypergraph, 2003, preprint.

[23] P. Damaschke, A tight upper bound for group testing in graphs, *Disc. Math.*, 48 (1994) 101-109.

[24] P. Damaschke and A. Sheikh Muhammad, Randomized group testing both query-optimal and minimal adaptive, *Proc. 38th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2012)*, LNCS Vol 7147, pp. 214-225.

[25] A. De Bonis and U. Vaccari, Improved algorithms for group testing with inhibitors, *Inform. Proc. Let.* 65 (1998) 57-64.

[26] K.M.J. De Bontridder, B.V. Halldórsson, M.M. Halldórsson, C.A.J. Hurkens, J.K. Lenstra, R. Ravi and L. Stougie, Approximation algorithms for the test cover problem, *Mathematical Programming*, 98 (2003) 477-491.

[27] R. Dorfman, The detection of defective members of a large population, *Ann. Math. Stat.*, 14 (1943) 436-440.

[28] **Ding-Zhu Du** and F.K. Hwang, Identifying d positive clones in the presence of inhibitors, to apprea in *International Journal of Bioinformatics Research and Applications*.

[29] **Ding-Zhu Du** and F. K. Hwang, *Combinatorial Group Testing and Its Applications (2nd ed.)*, World Scientific, Singapore, 1999.

[30] **Ding-Zhu Du** and F. K. Hwang, *Pooling Designs and Nonadaptive Group Testing: Impotant Tools for DNA Sequencing*, World Scientific, 2006.

[31] **Ding-Zhu Du** and F.K. Hwang, Competitive group testing, *Discrete Applied Mathematics* 45 (1993) 221-232.

[32] **Ding-Zhu Du**, F. K. Hwang, **Weili Wu** and T. Znati, A new construction of transversal designs, *Journal of Computational Biology*, 13 (2006) 990-995.

[33] **Ding-Zhu Du** and K.-I Ko, Some completeness results on decision trees and group testing, *SIAM Algebraic and Discrete Methods*, 8 (1987) 762-777.

[34] **Ding-Zhu Du**, Ker-I Ko, Xiaodong Hu, *Design and Analysis of Approximation Algorithms*, Springer 2011.
[35] **Ding-Zhu Du** and H. Park, On competitive group testing, *SIAM Journal of Computing* 23 (1994) 1019-1025.
[36] **Ding-Zhu Du**, G.-L. Xue, S.-Z. Sun, and S.-W. Cheng, Modifications of competitive group testing, *SIAM J. Computing* 23 (1994) 82-96.
[37] A.G. D'yachkov, F.K. Hwang, A.J. Macula, P.A. Vilenkin, and C.-W. Weng, A construction of pooling designs with some happy surprises, preprint, 2004.
[38] A. G. Dyachkov and V. V. Rykov, Bounds of the length of disjunct codes, *Problems Control Inform. Thy.* 11 (1982), 7-13.
[39] A. G. Dyachkov and V. V. Rykov, A survey of superimposed code theory, *Problems. Control Inform. Thy.* 12 (1983) 1-13.
[40] A. G. Dyachkov, V. V. Rykov and A. M. Rashad, Superimposed distance codes, *Problems Control Inform. Thy.* 18 (1989) 237-250.
[41] A. G. D'ychkov, A. J. Macula, D. C. Torney, and P. A. Vilenkin, Two models of nonadaptive group testing for designing screening experiments, *Proc. 6th Int. Workshop on Model-Orented Designs and Analysis*, 2001, pp. 63-75.
[42] D. Eppstein, M. T. Goodrich, and D. S. Hirschberg, Improved combinatorial group testing for real-world problem sizes, *SIAM J. on Computing* vol 36, no 5 (2007) 1360-1375.
[43] M. Farach, S. Kannan, E. Knill, S. Muthukrishnan, Group testing problem with sequences in experimental molecular biology, *Proc. Compression and Complexity of Sequences*, 1997, pp. 357-367.
[44] H. Gao, F.K. Hwang, M. Thai, **Weili Wu** and T. Znati, Construction of disjunct matrices for group testing in the complex model, *Journal of Combinatorial Optimization* 12(3): 297-301 (2006).
[45] M.R. Garey and D.S. Johnson, *Computers and Intractability*, (W.H. Freeman, San Francisco, 1979).
[46] V. Grebinski and Kucherov, Optimal reconstruction of graphs under additive model, *Algorithmica* 28 (2000) 104-124.
[47] V. Grebinski and G. Kucherov, Reconstructing a Hamiltonian cycle by querying the graph: Application to DNA physical mapping, *Disc. Appl. Math.* 88 (1998) 147-145.
[48] E.D. Green and M.V. Olson, Systematic screening of yeast artificial chromosone libraries by use of the polymer chain reaction, *Proc. Nat. Acad. Sci., USA* 87 (1990) 1213-1217.
[49] B.V. Halldórsson, M.M. Halldórsson and R. Ravi, On the approximability of the minimum test collection problem, *Lecture Notes in Computer Science*, 2161 (2001) 158-169.
[50] H.M. Huang, F.K. Hwang and J. F. Ma, Using transforming matrices to generate DNA clone grids, *Disc. Appl. Math.* 129(2-3): 421-431 (2003).
[51] T. Huang and C.-W. Weng, A note on decoding of superimposed codes, *Journal of Combinatorial Optimization*, 7:4 (2003).
[52] F. K. Hwang and W. D. Lin, The incremental group testing model for gap closing in sequencing long molecules, *J. Combin. Opt.* 7:4 (2003).
[53] F. K. Hwang, An isomorphic factorization of the complete graph, *Journal of Combinatorial Theory* 19 (1995) 333-337.
[54] F. K. Hwang and Y. C. Liu, Error tolerant pooling designs with inhibitors, *J. Comput. Biol.* 10 (2003) 231-236.
[55] F.K. Hwang, On Macula's error-correcting pooling design, *Discrete Mathematics* 268(1-3): 311-314 (2003).
[56] F. K. Hwang and V. T Sós, Non-adaptive hypergeometric group testing, *Studia Scient. Math. Hungarica* 22 (1987) 257-263.
[57] P. Indyk, H.Q. Ngo, and A. Rudra, Efficiently decodable non-adaptive group testing, *SODA 2010*, pp. 1126-1142
[58] P. Johann, A group testing problem for graphs with several defective edges, *Disc. Appl. Math.* 117 (2002) 99-108.
[59] S. M. Johnson, A new upper bound for error correcting codes, *IEEE Trans. Inform. Thy.* 8 (1962) 203-207.
[60] R. M. Karp, R. Stougton and K. Y. Yeung, Algorithms for choosing differential gene expression experiments, *Proceedings of the Third Annual International Conference on Computational Molecular Biology*, 1999, pp. 208-217.
[61] W. H. Kautz and R. R. Singleton, Nonrandom binary superimposed codes, *IEEE Trans. Inform. Thy.* 10 (1964) 363-377.
[62] G. Klau, S. Rahmann, A. Schliep, M. Vingron, and K. Reinert, Optimal robust non-unique probe selection using integer linear programming, *Bioinformatics*, 20 (2004) I186-I193.
[63] Y. Li, M. Thai, Z. Liu and **Weili Wu**, Protein-to-protein interactions and group testing in bipartite graphs, *International Journal of Bioinformatics and Applications*, 1 (2005) 414-419.

[64] A.J. Macula, A simple construction of d-disjunct matrices with certain constant weights, *Discrete Mathematics* 162 (1996) 311-312.
[65] A. J. Macula, Error correcting nonadaptive group testing with d^e-disjunct matrices, *Discrete Applied Mathematics* 80 (1997) 217-222.
[66] A. J, Macula, P. Vilenkin, and D. Torney, Two stage group testing in the presence of errors, *DIMACS Proceedings of Medical Applications of Discrete Mathematics*, DIMACS Series in Discrete Math and Theoretical Computer Science, Vol. 55, American Mathematical Society, 2000, pp. 145-157.
[67] A. J. Macula, Trivial two-stage group testing with high error rates, *Journal of Combinatorial Optimization*, 7 (2003) 361-368.
[68] A. J. Macula, V. V. Rykov, and S. Yekhanin, Trivial two-stage testing for complexes using almost disjunct matrices, *Discrete Applied Mathematics*, 137 (2004) 97-107.
[69] Anthony J. Macula, Susannah Gal, Cheryl P. Andam, Morgan A. Bishop, Thomas E. Renz: Pcr Nonadaptive Group Testing of DNA Libraries for Biomolecular Computing and Taggant Applications, *Discrete Math., Alg. and Appl.* 1(1): 59-70 (2009)
[70] Anthony J. Macula, Alexander Schliep, Morgan A. Bishop, Thomas E. Renz: New, Improved, and Practical k-Stem Sequence Similarity Measures for Probe Design. *Journal of Computational Biology* 15(5): 525-534 (2008)
[71] M. V. Marathe, A. G. Percus, and D. C. Torney, Combinatorial optimization in biology, manuscript, 2000.
[72] B. M. E. Moret and H. D. Shapiro, On miniming a set of tests, *SIAM J. Sci. Statist. Comput.* 6 (1985) 983-1003.
[73] Y. Mutoh, M. Jimbo and H.L. Fu, A resolvable $r \times c$ grid-block packing and its application to DNA library screening, preprint.
[74] H. Q. Ngo and **Ding-Zhu Du**, New constructions of non-adaptive and error-tolerance pooling designs, *Discrete Mathematics* 243 (2002) 161-170.
[75] H. Q. Ngo and **Ding-Zhu Du**, A survey on combinatorial group testing algorithms with applications to DNA library screening, in *Discrete mathematical problems with medical applications* (New Brunswick, NJ, 1999), pp. 171–182, DIMACS Ser. Discrete Math. Theoret. Comput. Sci., 55, Amer. Math. Soc., Providence, RI, 2000.
[76] Hung Q. Ngo, Ely Porat, Atri Rudra: Efficiently Decodable Error-Correcting List Disjunct Matrices and Applications - (Extended Abstract), *ICALP* (1) 2011: 557-568.
[77] Hung Q. Ngo, Ely Porat, Atri Rudra: Efficiently Decodable Compressed Sensing by List-Recoverable Codes and Recursion. *STACS 2012*: 230-241
[78] H. Park, **Weili Wu**, Z. Liu, X. Wu and H. Zhao, DNA screening, pooling designs, and simplicial complex, *Journal of Combinatorial Optimization* 7:4 (2003).
[79] W. W. Paterson, *Error Correcting Codes*, (MIT Press, Cambridge, Mass. 1961).
[80] P.A. Perzner, *Computational Molecular Biology: an Algorithmic Approach*, MIT Press, Massachusetts, 2000.
[81] R.M. Phatarfod and A. Sudbury, The use of a square array scheme in blood testing, *Statistics in Medicine* 13 (1994) 2337-1343.
[82] J. Schlaghoff and E. Triesch, Improved results for competitive group testing, *Combinatorics, Probability and Computing* 14 (2005) 191-202.
[83] A. Schliep, D. C. Torney and S. Rahmann, Group testing with DNA chips: generating designs and decoding experiments, *Proceedings of the 2nd IEEE Computer Society Bioinformatics Conference*, 2003.
[84] H. Tettelin, D. Radune, S. Kasif, H. Khousi and S. L. Salyberg, Optimized multiplex PCR: Efficiently closing a whole geneme shotgun sequencing project, *Genomics* 62 (1999) 500-507.
[85] My T. Thai, David MacCallum, Ping Deng, **Weili Wu**: Decoding algorithms in pooling designs with inhibitors and error-tolerance. *IJBRA* 3(2): 145-152 (2007).
[86] N. Thierry-Mieg, L. Trilling, and J.-L. Roch, Anovel pooling design for protein-protein interaction mapping, manuscript, 2004.
[87] D.C. Torney, Sets pooling designs, *Ann. Combin.* 3(1999) 95-101.
[88] E. Triesch, A group testing problem for hypergraphs of bounded rank, *Disc. Appl. Math.* 66 (1996) 185-188.
[89] Feng Wang, David Hongwei Du, Xiaohua Jia, Ping Deng, **Weili Wu**, David MacCallum: Non-unique probe selection and group testing. *Theor. Comput. Sci.* 381(1-3): 29-32 (2007).
[90] **Weili Wu**, C. Li, X. Wu and X. Huang, Decoding in pooling designs, *Journal of Combinatorial Optimization* 7:4 (2003).
[91] **Weili Wu**, Y. Huang, X. Huang and Y. Li, On error-tolerant DNA screening, *Discrete Applied Mathematics* 154 (12): 1753-1758 (2006).

Combinatorial Search Problems

by Gyula O.H. Katona

1

Problem 1. *Exactly two (at most two?) distinguished elements x and y of $[n]$ are sought by testing subsets of $[n]$. Only subsets of size at most k can be used. The result of the test A is YES, if and only if at least one of x and y is in A. Determine the minimum number of tests in both cases: adaptive, non-adaptive.*

The case when there is exactly one distinguished element was more or less solved in [1]. The adaptive case was easy: one could ask disjoint k-element sets until the element was caught or the remaining part was of size at most $2k$. Then the usual halving finished the algorithm.

The non-adaptive case was less easy, [1] gave only lower and upper estimates. But it determined the form of the matrix of the optimal search (0,1 matrix where the rows represent the test sets). It was described by its columns: for some integer parameter r, the matrix contains all distinct columns with at most r 1's, some columns with $r+1$ 1's, and no columns with more than $r+1$ 1's. If k is small, namely $n \geq \binom{k+1}{2}+1$ then the exact minimum of the number of tests is $\left\lceil 2\frac{n-1}{k+1}\right\rceil$. A new, nice approach can be found in [2].

2

A graph $G(V,E)$ is given with one distinguished edge. Paths of the graph can be tested if the distinguished edge is along the path or not. Let $\alpha(G)$ and $\nu(G)$ denote the minimum number of tests in an adaptive and non-adaptive algorithm, finding the distinguished edge, respectively. Since single edges are also paths, $\alpha(G) \leq \nu(G) \leq |E|$.

Problem 2. *Determine*

$$\alpha(n,e) = \max_{|E|=e} \alpha(G)$$

and

$$\nu(n,e) = \max_{|E|=e} \nu(G).$$

If G is the complete graph K_N then the results of [1] can be well applied in the determination of $\alpha(K_N)$ and $\nu(K_N)$ since the number of possible distinguished elements is $\binom{N}{2}$ and sizes of the test sets are bounded by $N-1$. In the adaptive case there must be a small difference at the end of the algorithm, since there is no path of length more than $N-1$. But this small gap can probably bridged. We only have to decompose K_N into paths of length $N-1$ and one shorter. On the other hand, in the non-adaptive case, the upper bound obtained by [1] can be probably attained by constructing the appropriate paths.

It is probably easier to determine the values

$$\max_e \alpha(n,e), \max_n \alpha(n,e), \max_e \nu(n,e), \max_n \nu(n,e).$$

3

Consider the set S of 0,1 sequences of length n. Introduce the *Hamming distance* between two such sequences as the the number of different digits. If $u \in S$ is a sequence, R is a non-negative integer then $B_r(u)$ is the ball with center u and radius r.

Problem 3. *We want to find one unknown element $x \in S$ by asking balls of radius at most k if x is in the ball or not. Both the adaptive and non-adaptive cases seem to be interesting. Even the case when $k \geq \frac{n}{2}$ seems to be non-trivial.*

4

Suppose that there are exactly two unknown elements r and g of $[n]$. Asking a subset $A \subset [n]$ the answer tells us

RED, if $r \in A, g \notin A$,
GREEN, if $g \in A, r \notin A$,
WHITE, otherwise.

Problem 4. *Determine the minimum lengths of the adaptive and the non-adaptive algorithms.*

References

[1] G. Katona, On separating systems of a finite set, *J. Combin. Theory* **1**(1966) 174-194.

[2] Éva Hosszu, János Tapolcai and Gábor Wiener, On a problem of Rényi and Katona, Proc. of the 8th Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications (eds. A. Frank, A. Recski, G. Wiener), pp. 229-232..

Contributed Problems

The plurality problem with three colors

by Gábor Wiener

This is a problem of Martin Aigner [1]. We are given n balls of (at most) three colors. We say that a ball is in plurality if its color class contains more balls than any other color classes (i.e. if we have 4 red, 3 green, and 3 yellow balls then the red balls are in plurality, while if we have 5 red, 5 green, and 0 yellow balls then there is no plurality). We have to find a ball of plurality color or to prove that there is no plurality using pairwise comparisons of the balls. Such a comparison tells us only whether the two balls have the same color or not. Our aim is to find the minimum number of comparisons $p(n)$ needed to find a plurality ball or to prove that there is no plurality. For two colors instead of three, the problem was solved by Saks and Werman [3] who showed that the number of comparisons needed is $n - \nu(n)$, where $\nu(n)$ is the number of 1's in the binary representation of n. Aigner, De Marco, and Montangero showed [2] that $p(n) \geq 3\lfloor \frac{n}{2} \rfloor - 2$ and Aigner proved [1] that $p(n) \leq \lfloor \frac{5n}{3} \rfloor - 2$. Both bounds are the best known ones and it is believed (as Aigner reports) that the upper bound is the correct answer. For $c \geq 4$ colors it is known that the answer is $\Theta(cn)$ [1]. For related problems and more literature also see [1].

References

[1] M. Aigner: Two colors and more, *in: G. O. H. Katona, I. Csiszár, G. Tardos (eds.): Entropy, Search, Complexity; Springer* (2007), 9-26.

[2] M. Aigner, G. De Marco, and M. Montangero: The plurality problem with three colors and more, *Theoretical Computer Science* **337** (2005), 319-330.

[3] M. E. Saks and M. Werman: On computing the majority by comparisons, *Combinatorica* **11** (1991), 383-387.

Geometric search

by Dániel Gerbner

Let us consider the simplest group testing model. We are given an underlying set, and we know that one of its elements is defective. We are also given a family $\mathcal{F}$ of its subsets, and we can ask a member of $\mathcal{F}$ as a test; the answer shows if the set contains the defective element or not. Our goal is to identify the defective element using as few questions as possible.

In our case the points are on the plane, and $\mathcal{F}$ is defined by some geometric properties. Let us consider the case when $\mathcal{F}$ is the family of convex sets, and the points are in general position. It was shown in [1] that the number of tests in a non-adaptive algorithm is between $n/(2\log n + 2)$ and $20n \log\log n/\log n$.

Can this gap be closed? Also, it might be interesting to consider other geometric properties or other search theoretic questions. Only the most simple ones have been studied.

References

[1] D. Gerbner, G. Tóth, Separating families of convex sets, COMPUTATIONAL GEOMETRY: Theory and Applications, *accepted*.

The path is more important than the goal, is it?

by Balázs Keszegh

The *pyramid graph* $Py(n)$ is a directed graph defined in the following way. $Py(n)$ has $N = n(n+1)/2$ vertices on $n+1$ levels, for $1 \leq i \leq n+1$ the ith level having i vertices $v_{i,1}, v_{i,2} \dots v_{i,i}$, and from every vertex $v_{i,j}$ where $1 \leq i \leq n$ and $1 \leq j \leq i$, there is a *left outgoing edge* going to $v_{i+1,j}$ (its *left child*) and a *right outgoing edge* going to $v_{i+1,j+1}$ (its *right child*). $Py(n)$ has one *root* at the top, $v_{1,1}$ and $n+1$ *sinks* at the bottom, the vertices on the $(n+1)$th level.

For each non-sink vertex exactly one of its two outgoing edges is labeled. The labeled edges define a unique path P from the root to a sink. The search problem we regard is the following: At the beginning we don't know anything about which edge is labeled. Determine the (a) path P or only (b) the endvertex of P (which is a sink) using queries of the following type: in one query we ask k non-sink vertices of the graph and the answer is the labeled edge going out from each of these vertices.

Fixing a natural number k, pa_k is the minimal number of rounds we need to determine the path, if in each round we can ask k questions. Similarly, si_k is the minimal number of rounds we need to determine the sink of P, if in each round we can ask k questions. Determine these parameters.

For $k = 1, 2, 3$ we know the exact answer [1] but for bigger k the problem is open.

Is it true that $pa_k = si_k$ for all k?

This problem can of course be generalized to other directed (acyclic) graphs, for example for complete rooted d-ary trees we know the minimal number of queries for every k.

This problem was originally proposed by Soren Riis.

References

[1] D. Gerbner, B. Keszegh, Path-search in a pyramid and in other graphs, Journal of Statistical Theory and Practice 6(2), (2012), 303-314.

Counting induced subgraphs of a bipartite graph by size

by Bhargav P. Narayanan

Dömötör Pálvölgyi asked (see below) if every bipartite graph on 2^k edges contains an induced subgraph on exactly half the number of edges. It would be nice to answer the following question which is also about sizes of induced subgraphs of a bipartite graph.

Given a graph G, let S_G be the set of sizes of all the induced subgraphs of G, i.e.

$$S_G = \{e(H) : H \text{ is an induced subgraph of } G\}.$$

Problem. Is there an absolute constant C such that for every bipartite graph G on m edges, $|S_G| = \Omega(\frac{m}{(\log m)^C})$?

There's been a large body of work on counting induced subgraphs of a given graph where we distinguish induced subgraphs by different parameters - for instance, up to isomorphism. The size is a natural parameter to count induced subgraphs by.

The problem is particularly interesting on bipartite graphs because it is a natural generalisation of the Erdős multiplication table problem. Estimating the size of S_G when $G = K_{n,n}$ is exactly the same question as computing the number of distinct products ab with both $a, b \leq n$; the multiplication table problem has only recently been resolved but it follows easily from the prime number theorem that $|S_{K_{n,n}}| = \Omega(\frac{n^2}{(\log n)^2})$.

Does every bipartite graph with 2^k edges have an induced subgraph with 2^{k-1} edges?

by Dömötör Pálvölgyi

Suppose we have a (simple) bipartite graph with 2^k edges. Is it true that there is a subset of the vertices such that their induced subgraph has exactly 2^{k-1} edges?

The answer is no for general graphs, since you can take a K_6 plus a disjoint edge. If we don't require the number of edges to be a power of 2, the answer is again no as shown by a $K_{5,9}$ plus a disjoint edge. I suspect that the answer to my question is also no.

The question is an equivalent formulation of the following problem. Suppose we want to find the unique defective elements in two disjoint sets in the (adaptive) group testing model and some questions have been already asked. Is it true that if there are at most 2^k possibilities left, then we can find both elements with k further questions?

For complete bipartite graphs (no questions asked before we start) this has been solved in G.J. Chang and F.K. Hwang, A group testing problem on two disjoint sets, SIAM J. Alg. Disc. Methods 2 (1981) 35–38. `http://epubs.siam.org/doi/abs/10.1137/0602005`