Optimal Group Testing Strategies with Interval Queries and Their Application to Splice Site Detection

Ferdinando Cicalese[1,⋆], Peter Damaschke[2,⋆⋆] , and Ugo Vaccaro[3]

[1] Institut für Bioinformatik, Centrum für Biotechnologie (CeBiTec), Universität Bielefeld, 33594 Bielefeld, Germany
nando@cebitec.uni-bielefeld.de
[2] School of Computer Science and Engineering, Chalmers University, 41296 Göteborg, Sweden
ptr@cs.chalmers.se
[3] Università di Salerno, 84081 Baronissi (SA), Italy
uv@dia.unisa.it

Abstract. The classical Group Testing Problem is: Given a finite set of items $\{1, 2, \ldots, n\}$ and an unknown subset $P \subseteq \{1, 2, \ldots, n\}$ of up to p positive elements, identify P by asking the least number of queries of the type "does the subset $Q \subseteq \{1, 2, \ldots, n\}$ intersect P?". In our case, Q must be a subset of *consecutive* elements. This problem naturally arises in several scenarios, most notably in Computational Biology. We focus on algorithms in which queries are arranged in *stages*: in each stage, queries can be performed in parallel, and be chosen depending on the answers to queries in previous stages. Algorithms that operate in few stages are usually preferred in practice. First we study the case $p = 1$ comprehensively. For two-stage strategies for arbitrary p we obtain asymptotically tight bounds on the number of queries. Furthermore we prove bounds for any number of stages and positives, and we discuss the problem with the restriction that query intervals have some bounded length d.

1 Introduction and Contributions

In group testing, the task is to determine the "positive" members of a set of objects O by asking as few queries as possible of the form "does the subset $Q \subseteq O$ contain at least one positive object?". A negative answer to a query tells us that all items in Q are "negative". Group testing is a paradigm occuring in a variety of situations such as quality control, multiple access communication, computational molecular biology, and data compression, among the others (see [5, 6, 9, 7, 2, 10, 12] and the numerous references quoted therein).

⋆ Supported by DAAD-grant no. A/04/33535.
⋆⋆ Supported by the Swedish Research Council (Vetenskapsrådet), project "Algorithms for searching and inference in genetics", grant no. 621-2002-4574.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 1029–1037, 2005.

In this paper we consider a variant in which the set of objects is $O = \{1, 2, \ldots, n\}$, the unknown subset $P \subseteq O$ of positive elements has cardinality at most p, and queries (tests) are constrained to be intervals $\{i, i+1, \ldots, j-1, j\}$, for some $i, j \in \{1, 2, \ldots, n\}$, such that $j - i + 1 \leq d$, d being a parameter of the problem. This variant naturally arises in several important situations. One of them concerns the determination of exon-intron boundaries within a gene [11, 13]. Adopting a very simplified model, one can view a gene as consisting of several disjoint substrings within a long string representing the DNA molecule. These substrings are called *exons*, and the substrings separating them are called *introns*. Only the concatenation of exons codes for a protein, the biological role of introns is rather unclear. Each boundary point linking an exon and an intron is called a *splice site*. The determination of splice sites is often a critical point to search for mutations associated with a gene responsible for a disease, because only mutations in exons are relevant. Now, in laboratories it is possible to obtain a "purified" version of a gene by transcribing it to cDNA in which all introns are removed and the exons spliced together. Using standard experimental procedures (polymerase chain reaction, PCR) one can pick any two positions in the cDNA string and determine whether they are at the same distance as they were in the original genomic DNA string. If these distances do not coincide then at least one intron (and hence a splice site) must be present in the genomic DNA between the two picked positions. The formulation of splice sites identification as a group testing problem with interval queries is explicitly stated in [8, 11, 13], where parameter d takes into account technological limitations of the PCR procedures.[4] A group testing procedure actually finds the positions (base or group of bases) before (or after) which there was an intron that has been spliced.

In the area of searching two kinds of algorithms are usually considered: In *adaptive* algorithms the tests are performed one by one, and the outcome of previous tests are assumed known at the time of determining the current test. Conversely, in *non-adaptive* algorithms all tests must be specified in advance without knowing the outcomes of the other tests. An intermediate situation corresponds to algorithms in which tests can be arranged in *stages*: in each stage a certain number of tests is performed non-adaptively, while tests of a given stage can be determined depending on the outcomes of the tests in the previous stages. In many situations, for instance in the biological setting, "few"-stage procedures are by far the most preferable ones [6, 9]. According to [9] *... the technicians who implement the pooling strategies generally dislike even the* 3*-stage strategies that are often used [...]. The pools are either tested all at once or in a small number of stages (usually at most 2).* In this paper we shall concentrate on s-stage group testing procedures, particularly for $s = 1$ and $s = 2$.

[4] In [13] it is mentioned that current technology suggests also a lower bound on the test size. One can see (as also noticed in [13]) that in the worst case this additional constraint makes impossible the complete classification of each item as either positive or negative.

To the best of our knowledge our paper is the first to provide a rigorous algorithmic analysis of group testing with interval queries. In particular, it is interesting to know that a simple strategy for any number of positives is provably optimal in the 2-stage case.

The work [13] and the book [11] report about the experimental evaluation, on real data, of the algorithm ExonPCR, that finds exon-intron boundaries within a gene. The authors of [13] give also a simple asymptotic analysis of their $\Theta(\log n)$-stage algorithm, for the case in which no bound on the test size is assumed.

Due to the space limitation, almost all proofs are removed from this extended abstract. The interested reader is referred to the full paper [1] where other scenarios in which our group testing problem naturally arises are also described.

2 Non-adaptive Interval Group Testing Algorithms

Let us denote by $\mathcal{N}_s(n,p,d)$ the worst-case minimum number of tests that are necessary (and sufficient) to successfully identify all positives in a search space of cardinality n, under the hypothesis that the number of positives is at most p, the interval tests have size at most d, and s-stage algorithms are used.

Theorem 1. *For all non-negative integers n, d, p, it holds that*

$$\mathcal{N}_1(n,p,d) = \begin{cases} 2\lfloor n/3 \rfloor + (n \bmod 3) & \text{if } p=1 \text{ and } d=2, \\ \lceil (n+1)/2 \rceil & \text{if } p=1 \text{ and } d \geq 3, \\ n & \text{otherwise.} \end{cases} \tag{1}$$

Proof. The main argument for the lower bounds is that any strategy must perform at least one test discriminating between any pair of consecutive items.

The matching upper bound $\mathcal{N}_1(n,1,3) \leq \lceil (n+1)/2 \rceil$, is given by the following strategy:

For $i = 1, 2, \ldots, \lceil (n+1)/2 \rceil$, perform the test T_i defined by

$$T_i = \begin{cases} \{1,2\} & i = 1, n > 1, \\ \{2i-2, 2i-1, 2i\} & i = 2, 3, \ldots, \lceil (n+1)/2 \rceil - 1, \\ \{n-1, n\} & i = \lceil (n+1)/2 \rceil \text{ and } n \text{ is odd}, \\ \{n\} & i = \lceil (n+1)/2 \rceil \text{ and } n \text{ is even, or } n = 1. \end{cases}$$

It is easy to check that this strategy defines a one-to-one mapping from the set of all possible solutions to the set of tests' outcomes.

The other cases are similar. □

3 Multistage Interval Group Testing

We shall now focus on s-stage interval group testing algorithms consisting of a fixed number s of stages. In each stage tests are performed in parallel so that the outcomes of the tests are taken into account only for choices made in the following stages.

For any integer $s \geq 2$, an s-stage interval group testing algorithm $\mathcal{A}$ consists of s successive 1-stage algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_s$, confining the positives in a collection of smaller and smaller subintervals. The last stage returns subintervals of size one since it determines the exact positions of the positives. Our main interest will be on case $s = 2$.

3.1 At Most One Positive

We shall first consider s-stage algorithms for the special case when at most one positive item is present in the search space. The main result of this section is contained in Theorem 2.

Proposition 1. *Let $I = \{1, 2, \ldots, n\}$ contain at most one positive item. Let $\mathcal{A}$ be a set of q interval tests in I such that for all possible choices of the positive item (including the choice for which no item in I is positive), the outcomes of the tests of $\mathcal{A}$ reduce the space of items in I which are candidate to be the positive one to a set of cardinality at most a. Then $q \geq \lceil n/(2a) \rceil$.*

In the following, an interval group testing algorithm *for at most one positive* in a set I is an algorithm that either finds the positive in I or correctly decides that no positive is in I. An interval group testing algorithm *for exactly one positive* in I is an algorithm that finds the positive element in a set I already known to contain a positive. These two notions are related by an obvious statement:

Proposition 2. *Let I be a set of cardinality n and I' be a set of cardinality $n+1$. For any interval group testing algorithm $\mathcal{A}$ for at most one positive in I, there exists an interval group testing algorithm $\mathcal{B}$ for exactly one positive, in I', which performs the same number of tests as $\mathcal{A}$.*

Trivially $\mathcal{B}$ can run the algorithm $\mathcal{A}$ on the first n elements of I'. If $\mathcal{A}$ finds a positive then $\mathcal{B}$'s output will coincide with $\mathcal{A}$'s output, otherwise $\mathcal{B}$ will conclude that the positive is the $n+1$st element of I'.

The following lemma is the key tool for analyzing s-stage interval group testing algorithms for one positive.

Lemma 1. *Fix integers $s \geq 1$ and $q \geq s$. Let $t = \lfloor q/s \rfloor$ and $k = q \bmod s$, and let $n = 2^s t^{s-k}(t+1)^k - 1$. There exists an s-stage interval group testing algorithm $\mathcal{A}$ for at most one positive element in $I = \{1, 2, \ldots, n\}$, which performs q tests. Moreover, there exists no s-stage interval group testing algorithm $\mathcal{A}$ for at most one positive element in $I = \{1, 2, \ldots, n+1\}$ which performs q tests*

Proof. The existence part is by induction on s. For $s = 1$ and for all $q \geq 1$ we have $k = 0$, $t = q$ and $n = 2q - 1$. Thus the desired result directly follows by Theorem 1.

Let now $s \geq 2$, $q \geq s$ and $t = \lfloor q/s \rfloor$, $k = q \bmod s$. Let $n = 2^s t^{s-k}(t+1)^k - 1$ and let $I = \{1, 2, \ldots, n\}$.

Assume as an induction hypothesis that for all $q' \geq s-1$ and for $t' = \lfloor q'/(s-1) \rfloor$, $k' = q' \bmod (s-1)$ and $\hat{n} = 2^{s-1} t'^{s-1-k'}(t'+1)^{k'}$ there exists an

$(s-1)$-stage algorithm for at most one positive in a set of cardinality $\hat{n}-1$ which uses q' tests. Notice that by Proposition 2 this implies that there exists an $(s-1)$-stage algorithm for exactly one positive in a set of cardinality $\hat{n}$ which uses q' tests.

An algorithm $\mathcal{A}$ that achieves the claimed result is as follows. In the first stage $\mathcal{A}$ uses t queries to partition the search space into $2t-1$ intervals of size $u = 2^{s-1}t^{s-k-1}(t+1)^k$ leaving an additional interval of size $u-1$ uncovered. This can be easily achieved by distributing the queries as described in the first part of Theorem 1.

By induction hypothesis and Proposition 2 the remaining $q-t$ queries suffice to complete the search in one of the intervals of size u if at least one query has answered YES and then a positive is know to be present or in the only subinterval of size $u-1$ if no test has answered YES.

The second statement of the lemma is proved by contradiction (omitted). □

Theorem 2. *Fix an integer $s \geq 1$. For all integers $n \geq 1$ we have*

$$\mathcal{N}_s(n,1,n) = \begin{cases} \lceil \log n \rceil & \text{if } n < 2^s - 1. \\ \min\left\{ q \mid 2^s \left(\left\lfloor \frac{q}{s} \right\rfloor\right)^{s-(q \bmod s)} \left(\left\lceil \frac{q}{s} \right\rceil\right)^{q \bmod s} - 1 \geq n \right\} & \text{otherwise} \end{cases} \tag{2}$$

In particular, for $n > 2^s - 1$, we have $\mathcal{N}_s(n,1,n) = \frac{s}{2} n^{1/s} + O(s)$.

Proof. For $n < 2^s - 1$, the problem reduces to binary search. For $n \geq 2^s - 1$, the result follows straightforwardly by Lemma 1. □

3.2 More Than One Positive: Two-Stage Interval Group Testing

The aim of this section is to prove asymptotically tight upper and lower bounds on the query number of 2-stage interval group testing algorithms for *at most p positives*, for any fixed number p. Remember that p is typically much smaller than n.

Unlike the case $p = 1$, for $p \geq 2$ the possible answers to the first questions in the first stage do no longer partition the search space into mutually disjoint candidate subsets to contain the positive(s). The instantaneous description of the searcher's knowledge after one stage is considerably more complicated. Nonetheless an upper bound for the case $s = 2$ and $p \geq 2$ can be easily obtained.

Theorem 3. *For any $p \geq 2$ we have $\mathcal{N}_2(n,p,n) \leq 2\sqrt{p-1}\sqrt{n} + O(p)$.*

Proof. An obvious algorithm $\mathcal{A}$ achieves the upper bound. The t_1 queries in $\mathcal{A}_1$ split the search space in t_1 disjoint intervals of roughly equal length n/t_1. This causes $n(p-1)/t_1 + O(p)$ queries in the second stage $\mathcal{A}_2$: If p intervals say YES, there is one positive in each of them, hence $p(n/t_1)/2 + O(p) = np/2t_1 + O(p)$ queries are needed. If $p-1$ intervals say YES, each of them could contain both

positives. (Actually only one interval can, but the searcher does not see which.) Hence $\mathcal{A}_2$ needs $n(p-1)/t_1 + O(p)$ queries, which is obviously the worst case for this algorithm. The $O(p)$ term accounts for the fact that n/t_1 is in general not integer, so that every YES interval could be longer by up to 1. Choosing $t_1 = \sqrt{p-1}\sqrt{n}$ minimizes $t_1 + n(p-1)/t_1$. Consequently, $2\sqrt{p-1}\sqrt{n} + O(p)$ queries are sufficient. □

The challenge is in fact to prove that one cannot do essentially better than this obvious strategy. Assume we could show that an adversary can enforce at least $n(p-1)/t_1$ queries in the second stage for any set of t_1 intervals in the first stage. Clearly, this would imply a lower bound that misses the upper bound of Theorem 3 only by an additive $O(p)$ term. For $p = o(\sqrt{n})$ this is asymptotically tight.

We remark that, for given p, it suffices to consider case $t_1 = \omega(p)$, because if $t_1 = O(p)$, the query intervals cut the search space in $O(p)$ pieces. (Here, a piece means a maximal contiguous sequence of elements bounded by interval ends, and without further interval ends in between. The length of a piece is the number of elements in it.) Hence the p longest pieces have total length $\Omega(n)$. If the adversary places the positives in these pieces, $\Omega(n)$ queries are enforced in the second stage.

Actually, our general lower bound is slightly weaker than mentioned above. In certain cases we get only $n(p-1)/(t_1 + p/2 + 1)$, but this means merely a factor $1 + p/2t_1$ between upper and lower bound, and for $t_1 \approx \sqrt{p}\sqrt{n}$ this is $1 + \sqrt{p}/2\sqrt{n}$. Hence our final result is still asymptotically tight. The exact $\mathcal{N}_2(n, p, n)$ for any n and p remains an open problem.

We start with some notations and give a rough idea of our proof of the lower bound.

Let $\mathcal{Q}$ be a set of interval questions. As said above, the interval ends in $\mathcal{Q}$ cut the search space $\{1, 2, \ldots, n\}$ into *pieces*. For any piece a, we denote by $N^{\mathcal{Q}}(a)$ the set of query intervals in $\mathcal{Q}$ containing a.

Let $\pi_1, \ldots, \pi_\ell$ be the pieces determined by the intervals of $\mathcal{Q}$. Let x_i be the length of π_i. By a YES *set* for $\mathcal{Q}$ we understand a set of query intervals in $\mathcal{Q}$ that have answered YES. Given a YES set Y, we define the YES vector $\mathbf{w}^Y = (w_1, \ldots, w_\ell)$, where w_i is the *weight* assigned to the piece π_i's according to the following scheme:

- A piece gets weight 1/2 if it can contain a positive but not more than one.
- A piece gets weight 1 if it can contain more than one positive.

We denote with $w^Y(\mathcal{Q})$ the weighted sum of the lengths of the pieces cut by $\mathcal{Q}$ weighted according to the YES vector associated to Y. In formulas $w^Y(\mathcal{Q}) = \sum_{j=1}^{\ell} x_j w_j^Y$.

Assume now that $\mathcal{Q}$ is the set of interval questions asked in the first stage of a two stage group testing algorithm which finds more than one positive. By Theorem 1, if Y is the set of intervals in $\mathcal{Q}$ that answer YES then the number

of queries to be asked in the second stage in order to find all the positives is *at least* $w^Y(\mathcal{Q})$. Therefore, in order to prove the promised bound we show that for each possible set of interval questions $\mathcal{A}_1$ there exists a yes set Y such that $w^Y(\mathcal{A}_1) \geq n/|\mathcal{A}_1|$.

Let $\mathcal{A}_1$ be the set of questions asked in the first stage of a two stage interval group testing algorithm for p positives in the set $\{1, 2, \ldots, n\}$. Let ℓ be the number of pieces in which the search space is cut by the questions in $\mathcal{A}_1$. Let $x_1, \ldots, x_\ell$ be the length of these pieces. For each piece a we shall write $N(a)$ instead of $N^{\mathcal{A}_1}(a)$, tacitly assuming that $\mathcal{A}_1$ is the set of interval questions we are referring to.

W.l.o.g. we can assume that for each two pieces a and b determined by $\mathcal{A}_1$ it holds that $N(a) \neq N(b)$. Then, we also have that ℓ, the total number of pieces, is at most $2t_1$. In fact, the number of pieces covered by query intervals is at most $2t_1 - 1$ (this is trivial for $t_1 = 1$, and every new interval can create at most two new pieces by splitting) and w.l.o.g. at most one piece a is outside all query intervals ($N(a) = \emptyset$).

The next lemma, basically a duality argument, is the key to the lower bound.

Lemma 2. *Consider a multiset of k (not necessarily distinct!)* YES *sets, and for each $i = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, \ell$, let w_{ij} be the weight of the jth piece in the* YES *vector associated to the ith* YES *set. If there exist $r > 0$ such that for all $j = 1, 2, \ldots, \ell$, it holds that $\sum_{i=1}^{k} w_{ij} \geq r$, then an adversary can force at least $\frac{r}{k}n$ queries in $\mathcal{A}_2$.*

Our lower bound consists in proving that it is possible to select a multiset of YES sets such that for each piece cut by $\mathcal{A}_1$ the sum of the weights assigned by the YES sets to that piece is greater than $p - 1$. In particular, we show that in many cases it is possible to have such a multiset of cardinality $k \leq t_1$ and in general of $k \leq t_1 + p/2 + 1$.

This is done in a series of combinatorial lemmas and case distinctions that we cannot present within the limited space. See the full paper [1]. We finally get:

Theorem 4. *For $p = o(\sqrt{n})$, the 2-stage interval group testing problem for at most p positives needs $\mathcal{N}_2(n, p, n) = 2\sqrt{p-1}\sqrt{n} + O(p)$ queries, and they are also sufficient.*

3.3 More and More Stages

We can characterize the asymptotic complexity of an optimal strategy for any fixed number s of stages. So far we do not have the precise constant of the main term as, in fact, we have in the 2-stage case, but recent ideas give hope that this gap can be closed.

Theorem 5. *Let $n \geq 2^s(p-1)$ and let $I = \{1, \ldots, n\}$ contain at most $p \geq 2$ positives. There exists an s-stage interval group testing algorithm which finds all positives in I by performing a number of tests smaller than or equal to*

$$s(p-1)^{(s-1)/s}(n - (n \bmod (p-1)))^{1/s} + s + p + (n \bmod (p-1)) - 2. \quad (3)$$

3.4 Bounded Queries

We also extend the results of the previous sections to the case of strategies using bounded interval tests whose size is bounded to be not larger than a given threshold d, and we characterized the query number of the optimal strategies in many cases of interest. We cannot state these results here (see the full paper), but a general observation is in order. Since the whole search space I of cardinality n must be covered, we are forced to use $\geq \lceil n/d \rceil$ tests. Dividing the search space into that many intervals of size d in the first stage and collecting the answers to these tests, will tell which of these intervals contain any positives. Then the algorithms have to recur, but the constraint of d on the tests' size does not count anymore. Thus in the next stage an algorithm with unbounded test size will solve the problem. This is provably optimal in many cases.

References

1. F. Cicalese, P. Damaschke, U. Vaccaro, *Optimal Group Testing Strategies with Interval Queries and Their Application to Splice Site Detection*, Technical Report 2004-04, Universität Bielefeld, Technische Fakultät. Available at `http://www.cs.chalmers.se/~ptr/2004-04.pdf`.
2. G. Cormode, S. Muthukrishnan, *What's hot and what's not: Tracking most frequent items dynamically*, in: ACM Principles of Database Systems, 2003.
3. L. A. Cox, X. Sun, and Y. Qiu, *Optimal and Heuristic Search for a Hidden Object in one Dimension*, in: Proc. of IEEE Conf. on System, Man, and Cybernetics, pp. 1252–1256, 1994.
4. A. De Bonis, L. Gasieniec, U. Vaccaro, *Generalized Framework for Selectors with Applications in Optimal Group Testing*, in: ICALP '03, LNCS **2719**, pp. 81-96, Springer-Verlag, 2003.
5. D.Z. Du and F.K. Hwang, *Combinatorial Group Testing and its Applications*, World Scientific, Singapore, 2000.
6. M. Farach, S. Kannan, E.H. Knill, S. Muthukrishnan, *Group Testing with Sequences in Experimental Molecular Biology*, in: Proc. of Compression and Complexity of Sequences 1997, B. Carpentieri, A. De Santis, U. Vaccaro, J. Storer (Eds.), IEEE CS Press, pp. 357-367, 1997.
7. E. H. Hong and R.E. Ladner, *Group testing for image compression*, IEEE Transactions on Image Processing, **11(8)**, pp. 901-911, 2002.
8. R. Karp, *ISIT'98 Plenary Lecture Report: Variations on the theme of 'Twenty Questions'*, IEEE Information Theory Society Newsletter, vol. **49**, No.1, March 1999.
9. E. Knill, *Lower Bounds for Identifying Subset Members with Subset Queries*, in: Proceedings of Symposium on Discrete Algorithms 1995 (SODA 1995), pp. 369-377, 1995.
10. Hung 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*, DIMACS Series Discrete Math. Theoret. Computer Science, **55**, Amer. Math. Soc., pp. 171-182, 2000.

11. P.A. Pevzner, *Computational Molecular Biology, An Algorithmic Approach*, MIT Press, 2000.
12. J. Wolf, *Born again group testing: Multiaccess communications*, IEEE Trans. Information Theory, **IT-31**, pp. 185-191, 1985.
13. G. Xu, S.H. Sze, C.P. Liu, P.A. Pevzner, N. Arnheim, *Gene hunting without sequencing genomic clones: Finding exon boundaries in cDNAs*, Genomics, **47**, pp. 171-179, 1998.