Reconstructing a Hamiltonian Cycle by Querying the Graph: Application to DNA Physical Mapping [1]

Vladimir Grebinski and Gregory Kucherov

INRIA-Lorraine and CRIN/CNRS
Campus Scientifique
615, rue du Jardin Botanique, BP 101
54602 Villers-lès-Nancy, France

email: {grebinsk,kucherov}@loria.fr

[1] A preliminary version of this paper was presented at the 5th Israeli Symposium on Theory of Computing and Systems, Ramat-Gan, Israel, June 17-19, 1997

Abstract

This paper studies four mathematical models of the multiplex PCR method of genome physical mapping described in [19]. The models are expressed as combinatorial group testing problems of finding an unknown Hamiltonian cycle in the complete graph by means of queries of different type. For each model, an efficient algorithm is proposed that matches asymptotically the information-theoretic lower bound.

Keywords: genome physical mapping, combinatorial group testing, algorithms, asymptotic complexity.

1 Introduction

1.1 Biological Motivation

This paper studies several mathematical models of the multiplex PCR method of genome physical mapping described in [19]. Physical mapping is the central stage in genome exploration which consists in creating some *landmarks* or *tags* throughout the DNA molecule. These landmarks are some specific nucleotide sequences associated with their positions on the molecule. They are usually obtained through the process of *cloning* which allows to extract some fragments of the molecule that can be then replicated to create multiple copies. These copies are then used to reconstruct the clone layout.

There are different methods of physical mapping [20]. The one we refer to in this paper (see [19]) differs from the others in that the cloned fragments are not supposed to cover the whole DNA molecule contiguously but rather occur on it with a frequency sufficient to bound, with a high probability, the gap between any two adjacent clones by some threshold value. Some clones can overlap forming longer continuous fragments (*contigs*). For example, the physical mapping project described in [19] dealt with a 170kb region of the circular *Bacillus subtilis* genome cloned in yeast artificial chromosomes (YAC) with 500 clones that formed 32 contigs of length from 0.5 to 15kb and gaps between them ranging from 6 to 18kb. Projects of much larger scale using this method can be envisaged.

The aim of such a physical mapping project is to reconstruct a *physical map* – the mutual placement of the contigs, that is their order and the lengths of the gaps between them. A tool for doing this is the multiplex LA PCR (*Long Accurate Polymerase Chain Reaction*) hereafter called simply *experiment* or *reaction*. The input to an experiment is a set of *primers*, which are short nucleotide sequences (of length about 20 nucleotides in project [19]) that characterize the ends of the contigs (see also [17]). Whenever the input set contains two primers corresponding to adjacent ends of neighboring contigs (like primers A and B on Figure 1(a)), the experiment outputs a PCR product the size of which corresponds to the length of the gap between the contigs. An additional condition for the product to come out is that its length should be smaller than some threshold distance, which is achieved by choosing a sufficiently big number of clones (contigs) at the previous stage of the method. Moreover, an experiment can yield several distinct products if the input set contains several corresponding pairs of primers. Thus, the number of products suggests the number of primer pairs corresponding to adjacent ends of two contigs. However, this information has a limited value, as in practice only a restricted small number of products can be distinguished and, in addition, distinct products of similar length can be visible as a single one.

The advantage of multiplexing is that many primers can be put into one reaction which increases the probability of obtaining at least one positive output, and on the other hand, eliminates many order possibilities in case when no output is obtained. On the other hand, no information is available

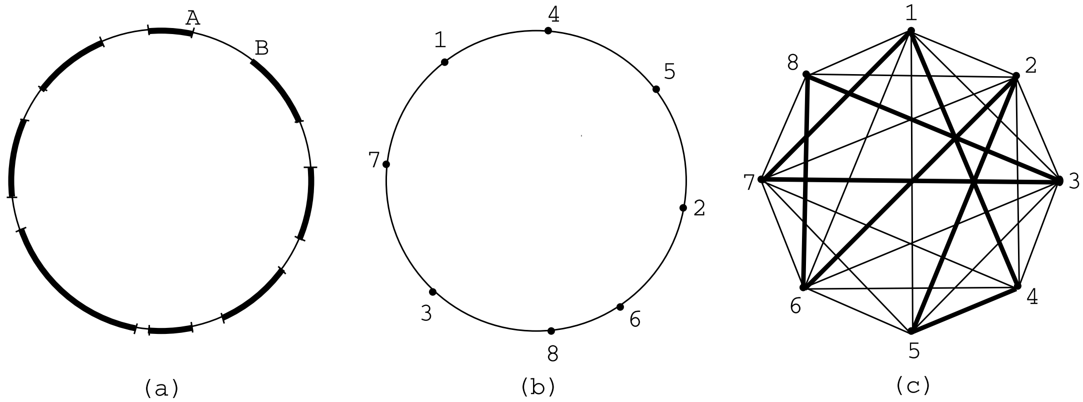

Figure 1: (a) Placement of contigs on a circular genome
(b) Placement of points on the circle (contigs are assimilated to points)
(c) Hamiltonian cycle corresponding to the order of points

on which pair of primers actually produced the output. Therefore, if the number of primers is big, little information is given by the output.

In this paper we give several mathematical formalizations of the physical mapping method described above. For all mathematical models we address the following combinatorial problem: what is the optimal strategy of conducting experiments in order to obtain a physical map using minimal number of them?

1.2 Mathematical Formulation

Firstly, we assume that once a primer corresponding to one end of a contig occurs in the input set, the one corresponding to another end of this contig does too. Thus, primers always come in pairs, each pair characterizing two ends of a contig. This assumption leads to a more simple and apparently less powerful mathematical model. However, from theoretical point of view this assumption is justified. As it will be shown in the paper, in most cases the asymptotic lower bounds can be reached under this assumption. In a practical implementation, however, manipulating individual primers is useful.

Secondly, since our purpose is to reconstruct the order of contigs and the distances between them, it is clear that the length and the internal structure of contigs are irrelevant. Thus, we assume that contigs have "no length" and hereafter we simply reduce contigs to points.

Thirdly, it is readily seen that between the problems of determining the order of points and determining the distances between them, the former one is essential. An obvious argument is that given an order of points, the distances between them can be determined easily in $O(n)$ experiments. In what follows we focus on the problem of order reconstruction.

Due to the above observations, we can model our physical mapping problem as follows:

> Assume that n points $\{1, 2, \ldots, n\}$ are placed on a circle in an unknown order (see Figure 1(b)). We are allowed to make *queries* about adjacency of some points. Determine the order of the points by making as few queries as possible.

Of course, the solution and its efficiency will strongly depend on the type of queries we are allowed to ask. Before making this precise, we reformulate the problem in terms of graphs. Clearly, an order of n points on the circle can be uniquely associated with a Hamiltonian cycle in the complete undirected graph K_n (we assume that the direction on the circle is irrelevant). Figure 1(c) illustrates the Hamiltonian cycle corresponding to the order of Figure 1(b). Thus, we reformulate the problem as follows:

> Consider the complete graph K_n with vertices $\{1, 2, \dots, n\}$. Assume that some Hamiltonian cycle HC is fixed in K_n, that is not known to us. We are allowed to make *queries* about adjacency of some vertices in HC. Determine HC by making as few queries as possible.

In this paper we study four types of queries which lead to four different mathematical models.

Multi-vertex model. For a set of vertices $\{a_1, \dots, a_m\}$, ask whether $K_{\{a_1,\dots,a_m\}} \cap HC$ is non-empty, where $K_{\{a_1,\dots,a_m\}}$ is the complete graph on the set of vertices $\{a_1, \dots, a_m\}$.

Quantitative multi-vertex model. For a set of vertices $\{a_1, \dots, a_m\}$, ask what the number of edges in $K_{\{a_1,\dots,a_m\}} \cap HC$ is.

k-vertex model. Assume that a constant k is predefined. For a set of vertices $\{a_1, \dots, a_m\}$, where $m \leq k$, ask whether $K_{\{a_1,\dots,a_m\}} \cap HC$ is non-empty.

Quantitative k-vertex model. For a set of vertices $\{a_1, \dots, a_m\}$, where $m \leq k$, what is the number of edges in $K_{\{a_1,\dots,a_m\}} \cap HC$?

In the multi-vertex model we ask whether at least one edge from a given set belongs to the Hamiltonian cycle. However, this set has a special structure – we ask about the edges of a complete subgraph. The multi-vertex model is strengthened in the quantitative multi-vertex model. Now we are allowed to *count* the number of edges of the Hamiltonian cycle in a complete subgraph. In terms of the biological method, this reflects our ability to count the number of neighboring primer pairs in the experiment. This is the most powerful model. The k-vertex model and quantitative k-vertex model are restrictions of the multi-vertex model and quantitative multi-vertex model respectively. They are motivated by the important practical constraint that only a limited number of primers can be submitted to an experiment.

1.3 Summary of the Results

In this paper we study the complexity of each of the four models and design asymptotically optimal algorithms for all of them. Note that the complexity is understood as worst-case query complexity, that is the number of queries necessary to reconstruct the order in most unfavorable circumstances. The complexity bounds are summarized in Figure 2. The last column indicates an important property of the algorithm – its adaptiveness. In adaptive algorithms, a query generally depends on the answers to the previous queries. In non-adaptive algorithms, all queries are independent and can be made in parallel. Often in practice an algorithm consists of two stages (see [10] for examples in molecular biology applications). For such algorithms, the type of each stage is shown.

We also discuss the multiplicative constants hidden in the O-notation, that are of course important for practical applicability of the algorithms. Finally, we describe a prototype implementation for both k-vertex models that are most close to biological reality.

Model	lower bound	algorithm performance	type of algorithm	
			first stage	second stage
multi-vertex model	$\Omega(n \log n)$	$O(n \log n)$	adaptive	
quantitative multi-vertex model	$\Omega(n)$	$O(n)$	adaptive	non-adaptive
k-vertex model	$\Omega(\frac{n^2}{k^2})$	$(1+o(1))\frac{n^2}{k^2}$	non-adaptive	adaptive
quantitative k-vertex model	$\Omega(\frac{n^2}{k^2})$	$(1+o(1))\frac{n^2}{k^2}$	non-adaptive	non-adaptive

Figure 2: Result summary

2 Multi-vertex Model

There are $(n-1)!/2$ Hamiltonian cycles in K_n. Since every query yields one bit of information, by the standard information-theoretic argument the lower bound $\log((n-1)!/2) = \Omega(n \log n)$ [2] can be immediately obtained. Note that the same lower bound stays for the average complexity since the average length of a branch in a binary tree with $(n-1)!/2$ leaves is $\Omega(n \log n)$.

Let us make now the following observation. Assume that only two vertices and not any number of them can be tested at a time. In other words, each query tests whether an individual edge belongs or not to the Hamiltonian cycle. It is known (see [1, exercise 3.5.5]) that in this case at least $n^2/4 - n/2 - 1 = \Omega(n^2)$ queries must be made in the worst case. In our model we are able to simultaneously ask about many edges. However, this set of edges has a special structure – it is a complete subgraph rather than any subgraph. Recall that in case of positive answer, we have no information about the vertices in the set that are actually adjacent. Therefore, it is not immediately clear if we can benefit from the possibility of testing many edges at once – if the number of vertices in the set is big, the value of the positive answer is decreased since we need further tests to identify the pair(s) which produced this answer. In contrast, the value of the negative answer is increased but its probability is little if the number of vertices is big.

In this section we show that the lower bound $\Omega(n \log n)$ can be achieved. Below we propose an algorithm that matches this bound.

Let HC be a Hamiltonian cycle and assume we have already discovered some of its edges. These edges form a set of disjoint paths that will be our main data structure.

Definition 1 *Let HC be a Hamiltonian cycle in K_n. A* chain *c is a sequence of vertices $< a_1, \ldots, a_t >$, $t \geq 1$, such that $\forall j$, $1 \leq j \leq t-1$, $(a_j, a_{j+1}) \in HC$. Note that degenerate one-vertex chains are allowed. For $c =< a_1, \ldots, a_t >$, define $left(c) = a_1$, $right(c) = a_t$. For a set of chains $C = \{c_1, \ldots, c_k\}$, define $left(C) = \{left(c_1), \ldots, left(c_k)\}$ and $right(C) = \{right(c_1), \ldots, right(c_k)\}$. A set of chains is* independent *if for every $c_1, c_2 \in C$, the edges $(left(c_1), left(c_2))$, $(left(c_1), right(c_2))$, $(right(c_1), right(c_2))$ don't belong to HC.*

The idea of the algorithm is to process all vertices one-by-one storing them in the independent set of chains C. The main observation is that a new vertex may have neighbors only in the sets $left(C), right(C)$. Assume we have an independent set of chains C.

The following procedure inserts vertex i into C.

[2] Throughout the paper the logarithms are binary unless the base is indicated.

```
INSERT-VERTEX$(C, i)$
1    query $left(C) \cup \{i\}$ and $right(C) \cup \{i\}$
2    if both answers are no then add one-vertex chain $< i >$ to $C$
3    if $left(C) \cup \{i\}$ yields yes and $right(C) \cup \{i\}$ yields no
4        then find in $left(C)$ one or two vertices adjacent to $i$
5            if one such vertex was found then append $i$ to the corresponding chain
6            if two such vertices $a'_1, a''_1$ were found then
7                replace chains $< a'_1, \dots, a'_{t'} >, < a''_1, \dots, a''_{t''} >$ by chain $< a'_{t'}, \dots, a'_1, i, a''_1, \dots, a''_{t''} >$
8    if $right(C) \cup \{i\}$ yields yes and $left(C) \cup \{i\}$ yields no
9        then proceed symmetrically to the previous case
10   if both $left(C) \cup \{i\}$ and $right(C) \cup \{i\}$ yields yes
11       then find in $left(C)$ a vertex $a'_1$ adjacent to $i$
12           find in $right(C)$ a vertex $a''_{t''}$ adjacent to $i$
13               if $a'_1 \neq a''_{t''}$
14                   then replace chains $< a'_1, \dots, a'_{t'} >, < a''_1, \dots, a''_{t''} >$
                                         by chain $< a''_1, \dots, a''_{t''}, i, a'_1, \dots, a'_{t'} >$
15                   else replace chain $< a'_1 >$ by chain $< a'_1, i >$
16   return $C$
```

The algorithm starts with the set $C = \{< 1 >\}$ and iterates INSERT-VERTEX(C, i) for vertices $i = 2, \dots, n$. Clearly, when all vertices have been processed, the set C consists of the Hamiltonian cycle HC. It is straightforward that INSERT-VERTEX maintains an independent set of chains. The **if – then – else** operator on line 13 checks for a particular case when i has a single neighbor in C which is the element of a singleton chain. To estimate the total number of queries, we have to estimate the number of queries of steps 4 and 11-12. By simple binary search, step 4 can be done in $\lceil 2 \log n \rceil$ queries. Similarly, steps 11, 12 can be done in $\lceil \log n \rceil$ each. Thus, INSERT-VERTEX(C, i) makes at most $\lceil 2 + 2 \log n \rceil$ queries and the whole algorithm RECONSTRUCT-MULTI(n, HC) makes at most $\lceil (2 + 2 \log n) \rceil n = O(n \log n)$ queries which matches the lower bound.

3 Quantitative Multi-vertex Model

This model extends the multi-vertex model by the possibility of counting pairs of adjacent vertices in the query set. The first observation is that this feature reduces the information-theoretic lower bound. Since each query has potentially $n + 1$ distinct answers, at least $\log_{n+1} (n - 1)!/2 = \Omega(n)$ queries in the worst case must be made by any algorithm. In this section we prove that surprisingly enough, this linear bound can be achieved and propose an algorithm that matches this bound.

The algorithm has two main stages:

```
RECONSTRUCT-QUANTITATIVE$(n, HC)$
split the set of vertices $\{1, \dots, n\}$ into three disjoint subsets such that any two vertices
    from the same subset are not adjacent in $HC$
find all edges between these subsets
```

The first stage is easy to accomplish in $O(n)$ queries:

```
SPLIT($n, HC$)
1   initialize three empty sets $S_1, S_2, S_3$
2   for $i := 1$ to $n$ do
3       if querying $S_1 \cup \{i\}$ yields no
4           then add $i$ to $S_1$
5           elseif querying $S_2 \cup \{i\}$ yields no
6               then add $i$ to $S_2$
7               else add $i$ to $S_3$
```

Step 7 makes use of the fact that if i has a neighbor in both S_1 and S_2, it cannot have one in S_3. Clearly, SPLIT(n, HC) makes at most $2n = O(n)$ queries.

The second stage deals with three bipartite graphs formed by the edges of HC between vertices of each of the three subsets. Consider such a graph. This is a bipartite graph in which the degrees of vertices have the values $\{0, 1, 2\}$. The problem now is to reconstruct the graph by querying its subgraphs, where the output of a query is the number of edges in the subgraph.

Let C_1, C_2 be the two independent vertex sets of the bipartite graph, each of size at most n. As the first step, consider the problem of determining the degree of each vertex in C_1 by querying different subsets of C_1 together with the whole set C_2. This can be trivially done in $|C_1|$ steps by querying, for each $i \in C_1$, the set $\{i\} \cup C_2$ and getting immediately the degree of i. However, better is possible.

The degree reconstruction problem can be reformulated as follows: reconstruct an unknown vector $(a_1, \ldots, a_n)$, where $a_i \in \{0, 1, 2\}$, by means of querying, for a set of positions $I = \{i_1, \ldots, i_l\} \subseteq \{1, \ldots, n\}$, for the sum $\sum_{j \in I} a_j$.

Let us focus for a moment on the restricted case when $a_i \in \{0, 1\}$, and consider the following definition. Given n, a $k \times n$ $\{0, 1\}$-matrix $\mathcal{A}$ is called a *separating matrix* for all $\{0, 1\}$-vectors of length n, if for any two such vectors v_1, v_2, $\mathcal{A}v_1 \neq \mathcal{A}v_2$ provided that $v_1 \neq v_2$. In other terms, arithmetic sums of different subsets of columns of $\mathcal{A}$ are all different. Associating columns to objects and rows to queries, it is easy to see that such a matrix provides an algorithm for the vector reconstruction problem with $a_i \in \{0, 1\}$, that makes k queries. The following result is known.

Theorem 1 *A separating $k \times n$ matrix can be effectively constructed with asymptotic number of rows $k = 2n/\log n$ which is also the asymptotically minimal number.*

Theorem 1 has an interesting history. The problem of minimizing the number of rows in a separating matrix was first studied by Erdős and Rényi [6] who proved the lower bound of $2n/\log n$. Several other proofs of this lower bound were obtained later with various methods. Probably the latest one is the elegant proof from [12] using the Kolmogorov complexity (other references can be also found in [12]). As for the upper bound of $2n/\log n$, it was independently proved by Lindström [13] and Cantor and Mills [3]. Later in [15], Lindström proposed a tricky construction of separating matrix with $2n/\log n$ rows asymptotically, based on elementary methods.

It is important to note that the separating matrix provides an algorithm, such that any query it makes does not depend on the answers to the previous queries. In other words, all queries are independent and can be specified before any answer is known. Such algorithms are called *non-adaptive*. Although non-adaptive algorithms are obviously less powerful in general, they often admit "nicer" mathematical formulations which allow to use more powerful mathematical methods.

Another important remark about the algorithm implied by Theorem 1 is that its complexity is twice the information-theoretic lower bound $\log_{n+1} 2^n = n/\log(n+1)$. Thus, in spite of the restriction of non-adaptiveness, the algorithm reaches, up to a constant factor of two, the absolute lower bound. Moreover, no adaptive algorithm with a better complexity is known.

Let us now turn back to our case when $a_i \in \{0,1,2\}$. The following extension of Theorem 1 holds.

Theorem 2 *A separating $k \times n$ matrix for the n-vectors with elements $\{0,1,2\}$ can be effectively constructed with the asymptotic value of $k = 4n/\log n$.*

Theorem 2 follows from the general bound $k = 2\lceil\log(d+1)\rceil n/\log n$ for the case when the elements of the unknown vector take the values $\{0,1,\ldots,d\}$. The proof of this bound is an extension of Lindström's proof from [15] of Theorem 1 and is given in Appendix 1.

Let us now return to the bipartite graph problem. As the second step, consider the following problem: Given a vertex $i \in C_1$, find its two adjacent vertices in C_2 by querying subsets of C_2. The simplest way to do it (see also Section 2) is to find the two vertices in $2\log n$ queries using binary search. However, binary search is a strongly adaptive method, and for the reason that will become clear in a moment, we need a non-adaptive algorithm.

Note that this problem as well as the vector separation problem above is typically studied in the area of Combinatorial Group Testing. We refer the reader to [5, 1] for overviews of the area, and to [9] for a nice account of some results for the case of two "defective objects" to be identified. In our case, two "defective objects" among n should be identified in a non-adaptive manner and querying a subset outputs the number (0,1 or 2) of "defective objects" in it. To define such a non-adaptive algorithm amounts to constructing a $k \times n$ $\{0,1\}$-matrix such that the component-wise sums of two different pairs of columns are all different. Let us call such a matrix a *2-separating $k \times n$ matrix.* The following result is from [14].

Theorem 3 ([14]) *A 2-separating $k \times n$ matrix can be effectively constructed with the asymptotic value of $k = 2\log n$.*

A proof of Theorem 3, more general than the proof of [14], is given in Appendix 2.

Note that although Theorem 3 provides the same bound as the naive binary search method, its proof is non-trivial since the non-adaptiveness is a serious restriction here. For comparison, the optimal adaptive algorithm for finding two "defective objects" in the model with counting was proved to make $C\log n$ queries, where $1.26 \leq C \leq 1.44$ [9].

Now we are in position to give an efficient algorithm for the bipartite graph problem that combines the two non-adaptive algorithms above.

Consider the non-adaptive algorithm based on Theorem 3 for finding the two adjacent vertices in C_2 for a given vertex $i \in C_1$. This algorithm is simply a collection of subsets $P_1,\ldots,P_k \subseteq C_2$ such that the numbers of adjacent vertices of i in $P_1,\ldots,P_k$ identify uniquely the two adjacent vertices of i in C_2. Since P_j's don't depend on i, we will ask about each P_j for all $i \in C_1$ "at once" by applying the separating matrix of Theorem 2.

```
RECONSTRUCT-BIPARTITE($C_1, C_2$)
    for $j := 1$ to $k$ do
        apply the separating matrix from Theorem 2 to find, for each $i \in C_1$,
          the number of adjacent vertices in $P_j$
```

Clearly, after the whole run of RECONSTRUCT-BIPARTITE(C_1, C_2) the number of adjacent vertices of each $i \in C_1$ in each P_j will be known, and therefore the adjacent vertices themselves can be determined. We conclude that RECONSTRUCT-BIPARTITE reconstructs a bipartite graph with n vertices in each component asymptotically in $(2\log n)(4n/\log n) = 8n = O(n)$ queries.

Returning back to algorithm RECONSTRUCT-QUANTITIVE, solving the initial Hamiltonian cycle reconstruction problem, we summarize the complexity in the following final theorem.

Theorem 4 *Reconstructing a Hamiltonian cycle in the quantitative multi-vertex model can be done in $O(n)$ queries.*

Proof: Consider the algorithm RECONSTRUCT-QUANTITIVE. The first step (algorithm SPLIT) requires $2n$ queries. The second step can be done by three applications of algorithm RECONSTRUCT-BIPARTITE. Therefore, the overall query complexity can be bounded from above by $2n + 3 \cdot 8n = 26n = O(n)$. □

In conclusion we note that the quantitative model studied in this section (also called the *additive* model) presents an interesting theoretical framework for reconstructing graphs of more general classes than Hamiltonian cycles. Results in this direction are presented in [7].

4 k-vertex and Quantitative k-vertex Models

In practical experiments only a limited number of primers can be submitted to one reaction. The technology described in [19] restricts this number to be smaller than 16. Reactions with a bigger number of primers don't give reliable outputs because of reaction inhibiting effects. In terms of our mathematical model, the number of vertices that can be queried is bounded by some predefined constant. This restriction cannot be captured within the methods described in the previous sections as all of them essentially require querying an unbounded number of vertices at some stages. This led us to consider in this section the k-vertex and quantitative k-vertex models which restrict the multi-vertex and quantitative multi-vertex models respectively by that every queried complete subgraph of K_n has at most k vertices.

The first observation is that each experiment with k vertices can be simulated by $\binom{k}{2} = \frac{k(k-1)}{2}$ experiments with 2 vertices. Since the lower bound for the 2-vertex model is asymptotically $n(n-2)/4$ (see Section 2), any algorithm that solves the problem in the k-vertex model (even in the quantitative one) makes at least $\frac{n(n-2)}{2k(k-1)}$ queries. Therefore, the focus of this section is to reduce the multiplicative constant in the quadratic complexity bound. For both models, we reduce this constant to one, that is we propose an algorithm which makes $\frac{n(n-1)}{k(k-1)}(1 + o(1))$ queries. Note that this is twice the lower bound, but is the best we could expect since the lower bound $n(n-2)/4$ for the 2-point model is not known to be tight – actually, no better algorithm than querying all the $n(n-1)/2$ edges is known.

The central idea is to cover the complete graph K_n by subgraphs $G_1, \ldots, G_M$, where each G_i is a complete graph K_m, $m \leq k$, such that every edge (i,j) of K_n belongs to exactly one G_i. Assume that we have constructed such a covering. Querying each G_i, we find at most n of them which contain edges of the Hamiltonian cycle HC. In each such G_i, we can identify the edges of HC using the techniques developed in the previous sections. (Of course, $G_i \cap HC$ does not form a Hamiltonian cycle of G_i, but it can be seen that the results of Sections 2 and 3 still apply to such graphs.) Processing one G_i then requires $O(k \log k)$ queries for the k-point model (Section 2) and $O(k)$ for the quantitative k-point model (Section 3), and the overall complexity of the method is respectively $M + O(nk \log k)$ and $M + O(nk)$. Thus, the main problem is to minimize M, that is to cover the graph K_n by a minimal number of graphs K_m, $m \leq k$, such that every edge of K_n occurs in only one of them. In the rest of the Section we describe how it can be done.

In terms of sets, our problem is to construct a minimal number of subsets of the set $\{1, \ldots, n\}$ such that every subset has at most k elements and every pair i, j, $1 \leq i, j \leq n$, occurs in exactly one subset. The problems of arranging objects of a set into some number of (intersecting) subsets of a given size such that each object and each pair of objects occur in a specified number of subsets is a well-established area in combinatorics called *design theory* or *block design* (see e.g. [8, 2, 4]). However, most of the results there present conditions for such an arrangement to exist and don't consider algorithmic aspects of its construction. Furthermore, subsets are usually required to have one or several specified cardinalities. This requirements are too strong for our purpose, as we allow subgraphs of any size smaller than k and we look for an algorithm approximating the minimal number of subgraphs and not for an exact solution.

Another link that should be mentioned here is the Theorem of Rödl [18] that insures that there exists a covering family of complete subgraphs K_k the number of which tends to $\frac{n(n-1)}{k(k-1)}$ for n going to infinity. However, we need stronger properties – the construction should be "efficient" and should guarantee that no edge is covered many times.

We present below a simple and easy-to-implement algorithmic solution to this problem. This solution has some relationships with classical design theory results (see the methods of *affine block design* in [2]), but we will not discuss them here. Instead, we present it in a self-contained way and focus on algorithmic aspects and complexity analysis.

Lemma 1 (Exact Design) *Consider the complete graph K_n. Let $n \geq k^2$ and assume that the set of vertices $V = \{1, \ldots, n\}$ is divided into k disjoint subsets $S_1, \ldots, S_k$ of n/k elements each. If $gcd(n/k, (k-1)!) = 1$, then $(n/k)^2$ subgraphs K_k can be effectively constructed such that every edge between S_i and S_j, $i \neq j$, occurs in exactly one of the subgraphs.*

Proof: Consider a $(k \times n/k)$-table A^0 where the vertices of S_j are placed (in any order) in row j. We construct a sequence of $(k \times n/k)$-tables $A^1, A^2, \ldots, A^{n/k-1}$ according to the following formula: $A^t(i,j) = A^0(i, j + (i-1)t \mod n/k)$, $1 \leq t \leq n/k - 1$. Intuitively, at each iteration each row i is circularly rotated by $(i-1)$. We claim that for every two vertices $x \in S_{i_1}, y \in S_{i_2}$, $i_1 \neq i_2$, there is exactly one column in $A^0, A^1, \ldots, A^{n/k-1}$ containing both x and y. Indeed, assume that two tables A^{t_1}, A^{t_2}, $t_1 \neq t_2$, contain a column with x, y. Then t_1, t_2 must be solutions of the equation $j_1 + (i_1 - 1)t \equiv j_2 + (i_2 - 1)t \mod n/k$, that is $j_1 - j_2 \equiv (i_2 - i_1)t \mod n/k$. Value $(i_2 - i_1)$ belongs to the set $[-(k-1), -1] \cup [1, k-1]$. Since n/k is relatively prime to $(k-1)!$, it is relatively prime to all numbers from $[-(k-1), -1] \cup [1, k-1]$. Therefore, $(i_2 - i_1)$ has an inverse element in ring $\mathbb{Z}/(n/k)\mathbb{Z}$, and there is only one solution $t = (i_2 - i_1)^{-1}(j_1 - j_2)$ of the equation above in $\mathbb{Z}/(n/k)\mathbb{Z}$. This shows that if there are two columns containing x, y, they must belong to the same table A^t. However, in each A^t there is only one column containing x. There are n/k tables each containing n/k columns. Associating every column to a complete subgraph, the lemma follows. □

The proof above gives a simple effective procedure of constructing subgraphs K_k. If n/k is not relatively prime to $(k-1)!$, we extend the set V of vertices by dummy vertices V' such that $(|V| + |V'|)/k$ verifies the property. Thus, the following corollary holds.

Corollary 1 *Under the conditions of Lemma 1, if $r > n/k$ and $gcd(r, (k-1)!) = 1$, then r^2 subgraphs K_m, $m \leq k$ can be effectively constructed such that every edge between S_i and S_j, $i \neq j$, occurs in exactly one of the subgraphs.*

Corollary 1 allows to construct subgraphs that cover all edges connecting vertices of distinct subsets of the partition. To cover the whole graph, we apply the construction recursively to each subset S_j. This leads to the following algorithm that constructs a covering of the set of vertices V, $|V| = n$.

```
DESIGN($V, k$)
    if $n \leq k$ then output $K_n$
    else find the smallest number $q \geq \max(k, \lceil n/k \rceil)$ such that $gcd(q, (k-1)!) = 1$
        divide the vertices of $V$ together with $qk - n$ dummy vertices into $k$ disjoint subsets $S_1, \ldots, S_k$
            by applying Corollary 1 find $q^2$ subgraphs $K_m$, $m \leq k$ covering each edge between
            distinct subsets $S_i$ and $S_j$ exactly once
        for $j := 1$ to $k$
            DESIGN($S_j, k$)
```

To estimate the total number M of subgraphs K_m, we need an estimate of the smallest number $q \geq \max(k, \lceil n/k \rceil)$ such that $gcd(q, (k-1)!) = 1$, i.e. q is not divisible by any prime number smaller than k. For the purpose of complexity estimation, we assume that instead of such q we choose p, where p is the smallest prime number $p \geq \max(k, \lceil n/k \rceil)$. Obviously, $p \geq q$ and p is not divisible by any number smaller than k. From Number Theory, it is known that the asymptotic bound $nextprime(n) - n \leq n^{\frac{11}{20}}$ holds (see e.g. [16]), where $nextprime(n) = min\{p \text{ is prime} | p \geq n\}$. Let $M = f(n, k)$ be the total number the subgraphs constructed by DESIGN(n, k). Then

$$f(n,k) = \begin{cases} 1 & \text{if } n \leq k \\ nextprime(k)^2 + k & \text{if } k < n < k^2 \\ nextprime(n/k)^2 + k \cdot f(n/k, k) & \text{if } n \geq k^2 \end{cases}$$

To estimate the asymptotic behavior of $f(n, k)$ for $n \to \infty$, we consider only the last case.

$$\begin{aligned} f(n,k) &\leq \sum_{i=1}^{\log_k n} k^{i-1} \cdot \left(\frac{n}{k^i} + \left(\frac{n}{k^i} \right)^{11/20} \right)^2 \\ &= \frac{n}{k} \left(\frac{n-1}{k-1} + 2 \left(n^{\frac{11}{20}} - 1 \right) \left(k^{\frac{11}{20}} - 1 \right)^{-1} + \frac{n^{1/10} - 1}{k^{1/10} - 1} \right) \\ &= \frac{n(n-1)}{k(k-1)} + o\left(\frac{n^2}{k^2} \right) \end{aligned}$$

Since the overall query complexity of the method is $f(n, k) + O(nk \log k)$ for the k-point model and $f(n, k) + O(nk)$ for the quantitative k-point model, we obtain the final result.

Theorem 5 *Reconstructing a Hamiltonian cycle in the k-vertex model can be done in $\frac{n^2}{k^2}(1 + o(1))$ queries.*

5 Practical Aspects and Computer Experiments

In this concluding section we discuss the applicability of our results to real-life physical mapping projects, and describe some computer experiments. Since the k-vertex model is most close to the reality of biological experiments, we concentrate on this model.

Note that algorithm DESIGN can be implemented very efficiently. Number q at line 2 can be found by simply testing numbers $\max(k, \lceil n/k \rceil), \max(k, \lceil n/k \rceil)+1, \ldots$ successively for the property of being relatively prime to all numbers smaller than k. How many numbers do we have to try before we find a good one? An attempt to give a rigorous answer would lead to deep issues of number theory. Without undertaking this analysis we note that this number of tries is small enough. Even if we look for the smallest prime number greater than n/k, the estimation $nextprime(n/k) - n/k \leq (n/k)^{\frac{11}{20}}$ from the proof above is very pessimistic. We will have to try only $\ln(n/k)$ numbers "on average" as the number of primes smaller than n is asymptotically $n/\ln n$. If we look for a number which is possibly not prime but relatively prime to all numbers $1, \ldots, k-1$, then the number of tries will be even smaller. Considering the set of numbers relatively prime to $1, \ldots, k-1$, it can be shown, using some number theory arguments that we omit, that the average gap between consecutive numbers from this set is $e^{\gamma} \log k \approx 1.781 \ln k$ asymptotically when $k \to \infty$ ($\gamma = 0.577...$ is Euler's constant). This confirms that the procedure of line 2 takes very reasonable computation time.

Note also that the case when $k < n < k^2$ can be greatly optimized. The source for optimization is that if n is close to k, then only a small part of the $(k \times q)$-table ($q \geq k$) will be filled with "real" vertices. Therefore, it might be better to create a table with a number of rows smaller than k (e.g. of order $\sqrt{n}$) that would be densely filled. This optimization improves considerably the performance of the algorithm for small n.

Algorithm DESIGN has been implemented in C++ and tested on various argument values. It appeared that for values of n, k arising in practical situations, DESIGN(n, k) runs very fast. The following table summarizes some computations on a SPARC-10 workstation.

n	k	Number of subgraphs	Ratio to $\frac{n(n-1)}{k(k-1)}$	time (sec)
500	8	6524	1.464	0.81
500	16	2113	2.032	0.49
1000	8	20055	1.124	3.01
1000	16	6394	1.536	1.47
5000	8	465211	1.042	73.54
5000	16	112689	1.082	33.9
10000	8	1837697	1.029	296.5
10000	16	435441	1.045	134.34

The ratio of the number of subgraphs constructed to $\frac{n(n-1)}{k(k-1)}$ indicates the quality of the design. As expected, the ratio approaches 1 when the recursive depth of the algorithm increases, the latter being equal to $\frac{\log n}{\log k}$. Note however that this is a somewhat pessimistic characteristic of the design quality since $\frac{n(n-1)}{k(k-1)}$ is the absolute minimum of the number of subgraphs, which is not always achievable.

Finally, let us discuss the lower order term in the complexity bound of Theorem 5. This term is of order $nk \log k$ for the k-vertex model and nk for the quantitative k-vertex model. For practical data (e.g. $n \sim 1000$, $k \sim 16$) this value may be even bigger than the number of subgraphs in the design (see above). Recall that this term corresponds to finding actual edges of the Hamiltonian cycle after localizing them in at most n complete subgraphs of size at most k. Fortunately, as n is big and k small, the expected time for this work is considerably smaller. Actually, as $(n/k)^2 \gg n$, most of "positive subgraphs" will contain, with high probability, only one edge of HC. Such an edge can be found in about $2 \log k$ queries ($\lceil \log \binom{k}{2} \rceil + 1$ in the worst case, see [1, 5]). Therefore, a good estimation (from above) for this work is of order $2n \log k$ queries. (Note that quantitative

k-vertex model makes no difference here.) Computer simulations show that for $n = 1000$, $k = 16$ there are around 911 positive subgraphs. If we are satisfied with finding only one edge from each positive subgraph we need about 12690 queries (including determination of positive subgraphs).

It must be noted that mathematical formalizations studied in this paper must be considered as the first step for modeling real-life physical mapping projects. We didn't take into account some further biological constraints arising in practice. One such constraint is related to errors that are always present in biological experiments. Another complicating feature is implied by the fact that in practice, the contigs which yield the positive outcome of the PCR are not those that are just adjacent but rather those that are situated within some threshold distance. This leads to a graph which is an extension of the Hamiltonian cycle such that a vertex is connected to several vertices in its "neighborhood" and not just to one vertex "on each side". However, we believe that our procedure for design construction is a useful basis that can be further refined into mathematical models better adapted to particular practical situations.

To conclude, we note that combinatorial search problems are very common in molecular biology (see [11]), screening clone libraries being a typical example. Therefore, we believe that the techniques developed here can be successfully applied to other biological tasks too.

Acknowledgments We are greatly indebted to Alexei Sorokin and Alexandre Bolotin from the Laboratoire de Génétique Microbienne of the Institute National de la Recherche Agronomique at Jouy-en-Josas (France), for explaining us the underlying biological problem, as well as for helpful discussions and comments. We also thank anonymous referees for their valuable remarks. We are grateful to Paul Zimmermann for commenting the manuscript and to Guillaume Hanrot for consultation on some number theory questions.

References

[1] M. Aigner. *Combinatorial Search.* John Wiley & Sons, 1988.

[2] T. Beth, D. Jungnickel, and H. Lenz. *Design Theory.* Cambridge University Press, 1986.

[3] D.G. Cantor and W.H. Mills. Determination of a subset from certain combinatorial properties. *Can. J. Math.*, 18:42–48, 1966.

[4] J. H. Dinitz and D. R. Stinson, editors. *Contemporary Design Theory.* John Wiley & Sons, 1992.

[5] D.-Z. Du and F. K. Hwang. *Combinatorial Group Testing and its applications*, volume 3 of *Series on applied mathematics.* WorldScientific, 1993.

[6] P. Erdős and A. Rényi. On two problems of information theory. *Publ. Hungar. Acad. Sci.*, 8:241–254, 1963.

[7] Vladimir Grebinski and Gregory Kucherov. Optimal reconstruction of graphs under the additive model. In Rainer Burkard and Gerhard Woeginger, editors, *Proceedings of 5th Annual European Symposium on Algorithms (ESA'97)*, Lecture Notes on Computer Science, pages 246–258, mars 1997.

[8] M. Hall. *Combinatorial Theory.* Blaisdell Publishing Company, 1967.

[9] F.K. Hwang. A tale of two coins. *American Mathematical Monthly*, 94:121–129, Feb. 1987.

[10] E. Knill. Lower bounds for identifying subset members with subset queries. In *Proc. 6th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 369–377, 1995.

[11] E. Knill and S. Muthukrishnan. Group testing problems in experimental molecular biology. Technical Report LAUR-95-1503, Los Alamos National Laboratory, March 1995.

[12] Ming Li and Paul Vitányi. Kolmogorov complexity arguments in combinatorics. *Journal of Combinatorial Theory, Series A*, 66:226–236, 1994.

[13] Bernt Lindström. On a combinatorial problem in number theory. *Canad. Math. Bull.*, 8:477–490, 1965.

[14] Bernt Lindström. Determination of two vectors from the sum. *Journal of Combinatorial Theory*, A6:402–407, 1969.

[15] Bernt Lindström. Determining subsets by unramified experiments. In J.N. Srivastava, editor, *A Survey of Statistical Designs and Linear Models*, pages 407–418. North Holland, Amsterdam, 1975.

[16] C.J. Mozzochi. On the difference between consecutive primes. *J. of Number Th.*, 24:181–187, 1986.

[17] E. Port, F. Sun, D. Martin, and M. Waterman. Genomic mapping by end-characterized random clones: A mathematical analysis. *Genomics*, 26:84–100, 1995.

[18] V. Rödl. On a packing and covering problem. *European Journal of Combinatorics*, 5:69–78, 1985.

[19] A. Sorokin, A. Lapidus, V. Capuano, N. Galleron, P. Pujic, and S. D. Ehrlich. A new approach using multiplex long accurate PCR and yeast artificial chromosomes for bacterial chromosome mapping and sequencing. *Genome Research*, 6:448–453, 1996.

[20] M. S. Waterman. *Introduction to Computational Molecular Biology. Maps, sequences and genomes.* Chapman & Hill, 1995.

Appendix 1

Recall that a separating matrix $\mathcal{A}$ for a class of vectors V is such that $\mathcal{A}v_1 \neq \mathcal{A}v_2$ for any $v_1, v_2 \in V$, $v_1 \neq v_2$. Equivalently, given vector $b = \mathcal{A}\mathbf{x}$ for some $\mathbf{x} \in V$, vector $\mathbf{x}$ can be uniquely reconstructed. We will prove the following generalization of Theorems 1 and 2.

Theorem 6 *A separating $k \times n$ matrix for the n-vectors with elements $\{0, 1, \ldots, d\}$ can be effectively constructed with the asymptotic value of $k = 2\lceil \log(d+1) \rceil n / \log n$.*

Since the proof is an extension of the Lindström's construction for Theorem 1, we first sketch the proof of Theorem 1 below.

The construction is based on the following key lemma.

Lemma 2 *Let $\mathcal{F}$ be a collection of sets such that for all B in $\mathcal{F}$ and $A \subset B$ we have $A \in \mathcal{F}$. Let the function $f(A)$ be defined for all $A \in \mathcal{F}$ with $f(A) = 0$ or $f(A) = 1$ and such that for some fixed $S \in \mathcal{F}$*

$$f(A \cap S) = f(A), \quad \textit{when } A \in \mathcal{F}.$$

If $C \in \mathcal{F}$ and C is not a subset of S, then we have ($|A|$ is the number of elements in A)

$$\sum_{A \subseteq C,\, |A| \ odd} f(A) = \sum_{A \subseteq C,\, |A| \ even} f(A).$$

Proof: Since $C \not\subseteq S$ there is $a \in C \setminus S$. Since $a \notin S$ we have $(A \cup \{a\}) \cap S = A \cap S$ and by the property of f,

$$f(A \cup \{a\}) = f(A), \text{when } A \in \mathcal{F}.$$

If $A \subset C$ and $a \notin A$ then $A \cup \{a\} \subset C$, also one of $|A|$ and $|A \cup \{a\}|$ is odd and the other is even. It follows that the the terms in the two sums above are pairwise equal, hence the sums are equal too. □

Now we describe how to construct a separating matrix $\mathcal{A}$ for the vectors with elements $\{0, 1\}$. Take a natural n. The rows of $\mathcal{A}$ will be labeled by the subsets of $\{1, 2, \ldots, n\}$. The columns are divided into blocks, one block for each non-empty subset $\{a_1, a_2, \ldots, a_i\}$. Such a block consists of i columns. For each block $\{a_1, a_2, \ldots, a_i\}$ we fill the columns of the matrix as follows:

- Fix any sequence $1 \leq n_1 < n_2 < \cdots < n_i \leq 2^{i-1}$, of natural numbers with the property that different subsets of $\{n_1, \ldots, n_i\}$ sum up to different numbers. We will take $n_k = 2^{k-1}$ for this purpose.
- For the column corresponding to a_k put arbitrarily n_k 1's in the rows labeled by subsets of $\{a_1, \ldots, a_i\}$ of odd cardinality.
- Put 0's to the rows corresponding to the other subsets of $\{a_1, \ldots, a_i\}$.
- Fill all other entries using the rule that value at the row marked by a subset A is the value of the entry at the row labeled by $A \cap \{a_1, \ldots, a_i\}$.

Take any block corresponding to a subset C. Consider a vector resulting from the following linear combination of the rows of the matrix: add up all rows labeled by the subsets of C of odd cardinality and subtract those labeled by the subsets of C of even cardinality. By Lemma 2, all entries in the columns that correspond to blocks S such that $C \not\subseteq S$ will turn to 0. For the current block, the sum will be $(2^0, \ldots, 2^{i-2}, 2^{i-1})$, and the equation

$$2^0 \cdot x_1 + \ldots + 2^{i-2} \cdot x_{i-1} + 2^{i-1} \cdot x_i = b \tag{1}$$

has a unique solution since $x_k = 0$ or $x_k = 1$. This leads to the following strategy of determining $\mathbf{x}$. Order all blocks $S_1, \ldots, S_{2^n-1}$ such that if $S_i \subseteq S_j$ then $i \leq j$. Start from S_{2^n-1}. Take the linear combination as explained above, obtain an equation (1), and determine all the unknowns corresponding to this block. Then proceed to the previous block in the order. At each step, the unknowns corresponding to the bigger blocks in the order have been already determined, those corresponding to the smaller blocks will have coefficient 0 in the corresponding linear combination, and therefore, those corresponding to the current block can be determined uniquely.

The matrix considered has $n \cdot 2^{n-1}$ columns and 2^n rows, which proves Theorem 1.

Now we generalize the construction above to prove Theorem 6. Note that the value of k in Theorem 6 matches the theoretic-information lower bound.

Proof of Theorem 6: The construction of the matrix is essentially the same as in the proof of Theorem 1. The place where this proof fails is the choice of $n_1, n_2, \ldots, n_i$. We cannot generalize this idea directly to the case when x_i's take values other than 0 or 1. Choosing bigger numbers is impossible since at most 2^{i-1} 1's can be put to a column. To cope with this difficulty, we use the following observation about the matrix $\mathcal{A}$ above. Assume that some column in that matrix is filled with 0's. The only consequence of this will be that the corresponding coefficient in (1) will turn to 0, and the corresponding variable value cannot be recovered. The other properties of the matrix are preserved and the other variables can be determined.

If now the variables range over $\{0, 1, \ldots, d\}$, then they can be determined from $p = \lceil \log(d+1) \rceil$ equations:

$$\begin{array}{ccccccc}
2^0 \cdot x_0 & + & 2^p \cdot x_p & + & 2^{2p} \cdot x_{2p} + \ldots & = & b_0 \\
2^1 \cdot x_1 & + & 2^{p+1} \cdot x_{p+1} & + & 2^{2p+1} \cdot x_{2p+1} + \ldots & = & b_1 \\
 & & \ldots & & & & \\
2^{p-1} \cdot x_{p-1} & + & 2^{2p-1} \cdot x_{2p-1} & + & 2^{3p-1} \cdot x_{3p-1} + \ldots & = & b_{p-1}
\end{array}$$

This gives a solution to the problem: we construct a $(\lceil \log(d+1) \rceil 2^n) \times (n2^{n-1})$-matrix by copying $\mathcal{A}$ p times, where the i-th column in a block is presented only in the $(i \mod p)$-th copy and replaced by a zero columns in all the others. Each copy allows to obtain one equation of the system above. This completes the proof. □

Theorem 2 now follows as the particular case of Theorem 6 when $d = 2$.

Appendix 2

We describe an effective method of constructing a d-separating matrix for any fixed d.

Definition 2 *A* $(0,1)$-*matrix* $\mathcal{A}$ *of size* $k \times n$ *is called a* d-separating *matrix iff the sum of any up to d columns is unique. (The sum is over* $\mathbb{Q}$*).*

In our situation, given n, we are looking for a matrix with minimal k. We will present such a matrix with $k = d \log n$ asymptotically. The construction has two stages: First we construct such a matrix over a field Z_p for sufficiently large p that depends on k and tends to infinity as n does. After this we will code each element of this field as a binary vector.

We need a well known lemma about symmetric polynomials:

Lemma 3 (Newton's formulas) *Consider two sets of symmetric polynomials:*

$$p_i = \sum_{j=1}^{n} {x_j}^i \qquad (i = 1 \dots d)$$

and

$$s_i = \sum_{1 \leq j_1 < j_2 \cdots < j_i \leq n} x_{j_1} x_{j_2} \dots x_{j_i} \qquad (i = 1 \dots d).$$

Then there exist polynomials $q_i \in \mathbb{Q}(x_1, \dots, x_d)$ such that

$$s_i = q_i(p_1, \dots, p_d).$$

The standard compactness argument shows that for each d, there exists a prime number p such that the set of polynomials q_i can be found in $\mathbb{Z}_p(x_1, \dots, x_d)$. Moreover, this holds for almost all primes p. We define p_d to be the smallest prime such that for all primes p, $p \geq p_d$ implies the existence of the set $\{q_i \in \mathbb{Z}_p(x_1, \dots, x_d)\}$.

Now, given n and d, we construct matrix $\mathcal{A}$ as follows:

1. Fix some prime p, $p \geq p_d$ and suppose that $n = p^m$ for some m.
2. Consider each $i \in (1 \dots n-1)$ as an element of $\mathbb{GF}(p^m)$. Let the i-th column of $\mathcal{A}$ be the vector $\begin{pmatrix} i \\ i^2 \\ \dots \\ i^d \end{pmatrix}$. (Here each entry is interpreted in $\mathbb{GF}(p^m)$).
3. Fix any basis $v_1, \dots, v_m$ of $\mathbb{GF}(p^m)$. Transform each entry $w = \alpha_1 v_1 + \cdots + \alpha_m v_m$, $\alpha_i \in \mathbb{Z}_p$, of the matrix $\mathcal{A}$ into the m-tuple $(\alpha_1, \dots, \alpha_m)$.
4. Finally, transform each entry $w \in \mathbb{Z}_p$ to the vector (over $\mathbb{Z}$) of its binary representation.

It is easy to estimate the height of the matrix $\mathcal{A}$:

$$d * m * \lceil \log p \rceil = d \frac{\log n}{\log p} * \lceil \log p \rceil = d \log n \frac{\lceil \log p \rceil}{\log p} \to d \log n \qquad \text{when } n \to \infty.$$

Here we suppose that $p \to \infty$ when $n \to \infty$.

Lemma 4 *The matrix $\mathcal{A}$ is d-separating.*

Proof: Consider Step 2. We will show that the sum of up to d columns defines uniquely these columns. Let $v_{a_1}, v_{a_2}, \dots, v_{a_d}$ be a subset of columns $a_1, a_2, \dots, a_d$. Assume that

$$v_{a_1} + v_{a_2} + \cdots + v_{a_d} = \begin{pmatrix} w_1 \\ w_2 \\ \dots \\ w_d \end{pmatrix}$$

Let $\beta_i = (-1)^i s_i(w_1, w_2, \dots, w_d)$. It is well known that a_i are the roots of the polynomial $g(x) = (x^d + \sum_{i=1}^{d} \beta_i x^{d-i})$ and since the polynomial g cannot have more than $deg(g) = d$ roots in the field, they are uniquely defined. If the sum of $d' < d$ columns is taken, we have 0 as the root of degree $d - d'$, and the non-zero components are again uniquely defined.

Step 3 cannot change the property of the matrix since addition in $\mathbb{GF}(p^m)$ can be viewed as addition of m-tuples with elements in $\mathbb{Z}_p$.

Step 4 also keeps the matrix d-separating. Indeed let $v_1, v_2, ..., v_{k'}, w_1, w_2, ..., w_{k''} \in \mathbb{Z}_p$ and $\vec{v_1}, \vec{v_2}, \ldots, \vec{v_{k'}}, \vec{w_1}, \vec{w_2}, \ldots, \vec{w_{k''}}$ be vectors of their binary representation. What remains to show is that

$$\sum_{i=1}^{k'} \vec{v_i} = \sum_{i=1}^{k''} \vec{w_i} \qquad (\text{sum in } \mathbb{Z}^{\lceil \log p \rceil})$$

implies

$$\sum_{i=1}^{k'} v_i = \sum_{i=1}^{k''} w_i \qquad (\text{ sum in } \mathbb{Z}_p) \ .$$

But this is immediate since

$$v \equiv \overrightarrow{(1, 2, 4, \ldots, 2^{\lceil \log p \rceil - 1})} \cdot \vec{v} \qquad (\mathbb{Z}_p)$$

This proves the Lemma. □

When $d = 2$, the height of the matrix $\mathcal{A}$ is $2 \log n$ which proves Theorem 3.