ELSEVIER

Discrete Applied Mathematics 88 (1998) 291–314

DISCRETE APPLIED MATHEMATICS

Genotyping of pooled microsatellite markers by combinatorial optimization techniques

Giuseppe Lancia[a], Mark Perlin[b,*]

[a] *Dipartimento di Elettronica e Informatica, University of Padua, Italy*
[b] *School of Computer Science, CMU, Pittsburgh, PA 15213, USA*

Received 20 August 1996; received in revised form 30 November; accepted 4 December 1997

Abstract

An important everyday task for geneticists and molecular biologists is that of isolating and analyzing some particular DNA regions (markers), each drawn from a limited and known set of possible values (alleles). This procedure is called genotyping and is based on DNA amplification and size separation. In order to increase the throughput of genotyping, recently a new experiment has been proposed which tries to analyze many different markers of similar size at once. We study the mathematical problem corresponding to this model and give a branch-and-bound algorithm for its solution. We show that by using the techniques described in this paper, genotyping of pooled markers can be computed effectively, thus potentially achieving a considerable reduction in time and expense. 1998 Published by Elsevier Science B.V. All rights reserved.

1. Introduction

In this paper we investigate the possibility of using combinatorial optimization techniques to increase the rate at which genotyping is currently performed on individuals. A brief simplified description of the situation is as follows.

1.1. Genetic markers

Human DNA is organized into 23 chromosome pairs, with one chromosome copy inherited from each parent. Along the chromosomal DNA sequences there are many sites that are highly polymorphic – i.e. there is tremendous variability in the DNA content at the site. Such sites can be used as *genetic markers*, and the different DNA sentences appearing at the site are known as the *alleles* of that marker. While useful to geneticists, highly polymorphic marker sequences typically do not code for proteins. Genetic polymorphism finds use in:

[*] Corresponding author. Email: perlin@cs.cmu.edu.

PII S0166-218X(97)00076-6

- *Genetic fingerprinting*: Forensic science exploits the allelic differences between individuals at multiple polymorphic markers to form a unique DNA signature for an individual [9].
- *Localization of genetic disease*: Genetics researchers use genetic markers to trace the pattern of inheritance of chromosomal DNA. These data can help determine shared chromosomal regions in related individuals affected by a disease, which in turn can roughly localize the causative gene on a chromosome [1]. There are robust statistical methods [12, 15, 20, 26] that are routinely used to rigorously implement this genetic localization.
- *Diagnosis of genetic disease*: Clinical geneticists use genetic markers to trace the pattern of inheritance in particular affected families. This analysis can help assess the probability of transmission of the disease gene to a given family member [22].

1.2. Genetic maps

Sampling an individual's genome with many highly polymorphic genetic markers can provide a high-resolution genotype "snapshot" in a digital form that can be compared with the genotypes of other relatives. With the advent of dense *genetic maps* having close to 1 Mb resolution [3, 6, 13], such genetic marker snapshots have become the mainstay of both regional and genome-wide (3000 Mb) searches for genes [2, 11]. See [24] for a general introduction to the construction and use of genetic maps, and [10] for some of the associated combinatorial problems.

1.3. Microsatellite markers

Genetic maps are largely comprised of *microsatellite markers* [25], which are abundant in the human genome, highly polymorphic, and based on the *polymerase chain reaction* (PCR). The older genetic markers (e.g., restriction fragment length polymorphisms, or RFLPs) are now rarely used. The microsatellite family includes di-, tri-, and tetranucleotide repeats that are DNA words of the form "PR_nS", where P is a fixed prefix string, S is a fixed suffix string, R is the nucleotide unit of the repetitive sequence R_n with the length of R small (e.g., 2, 3, or 4), and n is the number of tandem copies of R.

A key advantage of microsatellites is that an allele corresponds to the number of repeated units (n, in PR_nS), and therefore the *length* of the PR_nS DNA sequence. Thus, genotyping can be performed by simply determining the size (i.e., not the sequence) of the amplified PCR products. Such size determinations are done by physically separating DNA fragments using *gel electrophoresis*. The number of repetitions varies widely among individuals, and for this reason microsatellites are also called *length polymorphism* markers.

With R="CA", the CA-repeat unit forms a *dinucleotide repeat* microsatellite marker. There are believed to be over 100 000 CA-repeats present in the human genome. These dinucleotide repeat markers have sequences of alternating C and A nucleotides, denoted as $(CA)_n$ where n is the number of repeats. Due to their high polymorphism, these markers can be used for the identification of Mendelian disorders [17, 19]. For simplicity, in the remainder of this paper we will primarily use CA-repeats as the markers of choice; however, our results hold for other length-polymorphic markers as well.

1.4. PCR amplification

The PCR amplification process [14] creates millions of copies of each CA-repeat template DNA sequence. This amplification process is not completely error-free, and some of the copies will have different length than the original (almost invariably shorter). However, most of the copies will have the correct length, and the number of wrong copies of size l will rapidly go to zero as l decreases. This stutter artifact may be due to slipped strand mispairing [8] during replication within the DNA sequence's repeat region. When alleles are closely spaced, or individuals are pooled together, this stutter artifact obscures the data and precludes automated scoring by simple inspection. Interestingly, the stutter artifact indicates where the true data lies, and, without stutter artifact, spurious peaks are readily mistaken for alleles [21].

The PCR fragments must be labeled so that they can be detected later on. Originally, radioactively labeled nucleotide precursors were used. With the advent of fluorescently labeled nucleotide precursors and automated fluorescent DNA sequencers, nonradioactive fluorescent labeling is now the common practice [28].

For the sake of simplicity consider the following example: by PCR amplification, copies of a $(CA)_{20}$ repeat are made; of these, 54% have length 40 base pairs (bp), 37.6% have length 38 bp (i.e. a CA pair was skipped by the PCR process), 6.4% have length 36 bp, 2% have length 34 bp and no copy has any of the other possible lengths. If we plot this distribution of lengths in a graph, the resulting curve is generally similar to an exponential decay with the peak corresponding to the correct length (Fig. 1). We call this distribution the *stutter pattern* for the allele. As described in [17, 18], under known fixed PCR conditions (e.g. enzyme, cycle times, number of cycles, template and primer concentrations, and buffers), PCR amplification of a given allele will, typically, result in the same stutter pattern (modulo some unavoidable measurement error). Therefore, a stutter pattern can be regarded as the signature of the corresponding allele, and we can build and store a library of the stutters for all the alleles of a given marker.

1.5. DNA size separation

The alleles of a marker (and their stutter patterns) are observed by size separation of the PCR products on an electorphoretic gel. Gel electrophoresis is an experiment in which all the DNA copies are put in a gel and subjected to an electric field. Under the influence of the field, the DNA molecules migrate in the gel, moving with speed inversely proportional to their size. DNA fragments of the same size, moving at equal speed, will form a band on the gel, of intensity proportional to the number of fragments. By comparing the position of a band to that of some sample molecules of known size,

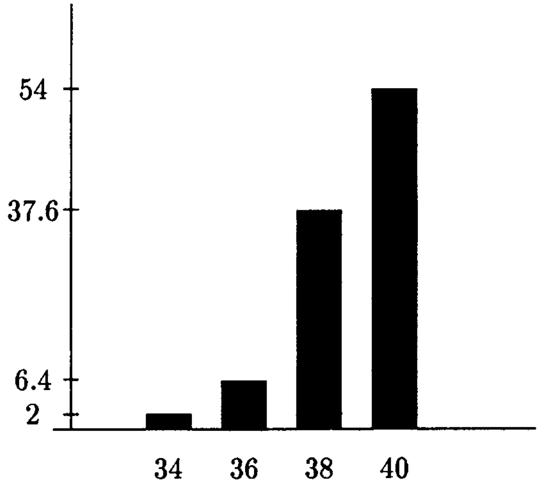

Fig. 1. Example of stutter pattern for a $(CA)_{20}$ repeat.

we can infer the length of all the molecules in that band. New technologies for DNA size separation, such as mass spectroscopy [27], are also being developed.

Gel electrophoresis is a key bottleneck in CA-repeat marker genetic analysis. To increase throughput, therefore, the PCR products from multiple experiments are loaded onto a single gel. Current multiplexing dimensions include:

- *Lane*: There are typically 24–64 lanes on one gel.
- *Size*: Since a given genetic marker produces its DNA bands within some limited size range, multiple CA-repeat genetic markers (typically 3–6) having disjoint size ranges can be loaded together within one lane.
- *Color*: On fluorescent DNA sequencers, multiple fluorescent dyes (typically 2–8) can be used to produce virtually independent data images. Using commercially available CA-repeat marker panels (e.g., for the applied biosystems automated four-color DNA sequencers), each lane usually combines 15 different CA-repeat markers by using five different size windows per color, and three different fluorescent colors. A fourth fluorescent color is used for size standards, rather than for marker data. This multiplexing reads out roughly 500 CA-repeat marker experiments per gel (15 markers/lane × 34 lanes/gel) [19].

Automation is feasible for virtually every step in the genotyping pipeline: robotic sample preparation, PCR amplification, electrophoretic sizing on automated DNA sequencers, allele determination, and computer data entry [7]. However, the allele determination step has thus far remained the *key bottleneck* eluding full automation. The reason is that although one might expect to observe two data bands on the gel from one marker's typing of an individual (i.e., in correspondence with the marker's two alleles, one from each of the two inherited chromosomes), the intrinsic PCR stutter artifact produces more complex signals.

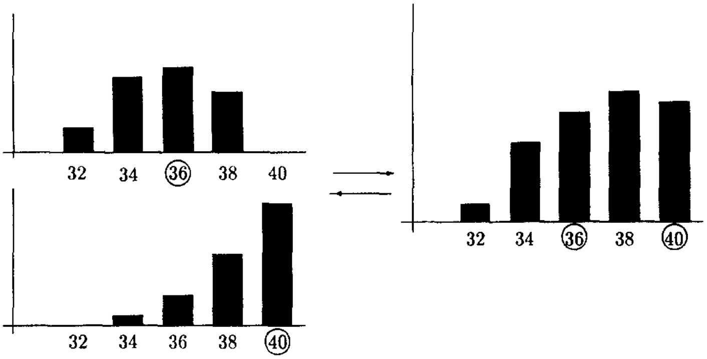

Fig. 2. In this example, calling the alleles for the higher peaks would result in error. Left: stutters for $(CA)_{18}$ and $(CA)_{20}$ repeats. Right: stutter for pooled markers.

1.6. PCR stutter encoding

Ideally, without PCR stutter, the two alleles of an individual's genetic marker would produce (up to) two distinct bands within the marker's size window on the gel. Note that using 30–50 bp as the size window for each marker in order to observe just two bands is not an optimal information use of valuable gel territory. It would be far better if somehow multiple (say 5–10) different markers were tagged as distinct, pooled within one size window, and the tagged readout measurements then mathematically demultiplexed into the allele data for all the markers within the window. This multiplexing of similarly-sized tagged markers might then produce considerable improvement in throughput, to say 2500–5000 experiments per gel.

PCR stutter artifact is an intrinsic property of microsatellite markers that may provide the requisite tagging, and allow of multiplexing markers that have similar sizes. It was suggested in [17, 18] that appropriate computer deconvolution algorithms might exploit the intrinsic PCR "artifact" as a useful tagging mechanism. This is not a trivial task, since numerous factors (e.g. differential PCR amplification, stutter variation, measurement noise) can confound the analysis. None the less, given the potential order of magnitude improvement in genetic marker throughput, this computational method warrants further study.

When running a gel on many markers at once whose stutters overlap, the result can be a complex pattern of bands, given by the sum of the bands for all the markers (Fig. 2). The goal of this paper is to study a mathematical model of superimposed PCR stutters from multiple length-polymorphic (e.g. CA-repeat) markers, and to devise efficient algorithms to deconvolve these data, thereby determining the alleles for all the superimposed markers. Our model will also account for the unavoidable presence of measurement errors in the readings.

The main biological assumption underlying the technique of pooling microsatellite markers is that when the stutter patterns for all the alleles are known and the PCR conditions are kept fixed, the stutter resulting from pooling many markers in the same gel will simply be the sum of the individual stutters for each single marker (modulo experimental error and noise) [18]. Under this assumption we show an efficient algorithm which can solve exactly the deconvolution problem for as many as seven pooled markers in a matter of 3–5 min. Considering that there are n^{14} possible solutions when each marker has n alleles (and in real cases n can be, for instance, as large as 20) we see that the running times are more than acceptable when compared to the size of the search space.

1.7. Organization

The remainder of the paper is organized as follows. Section 2 introduces the mathematical model for the problem. In Section 3 we address the computational complexity of the problem and transform it into an equivalent problem. Section 4 describes a fast global search procedure based on branch and bound. In Section 5 we focus on the monodimensional version of the problem, for which we give a pseudopolynomial dynamic programming and a branch and bound algorithm. Section 6 reports the computational results. A few conclusions are drawn in Section 7.

2. The mathematical model

The following parameters characterize the pooled genotyping problem:

- k: the number of *markers* to be pooled
- n_i: the number of *alleles* for marker $i=1,\dots,k$
- D_j^i: the *domain* of the stutter for allele $j=1,\dots,n_i$, marker $i=1,\dots,k$. By domain we denote the set of sizes, in bp, of the copies created by PCR amplification of the allele j for marker i. Let $\alpha_j^i := \min D_j^i$ be the smallest detectable size for which the PCR produces some copy of the allele, and $\beta_j^i := \max D_j^i$ be the largest size (normally, corresponding to the correct value for the allele). Ideal PCR conditions would amount to $\alpha_j^i = \beta_j^i$, i.e. $|D_j^i| = 1$. More generally, $|D_j^i| > 1$, and the interval $[\alpha_j^i,\dots,\beta_j^i]$ corresponds to a "window" on the gel where the stutter will be localized. However, typically not all the positions within this window will be achieved by some bands of a stutter. For example, generally an allele of a d-nucleotide repeat will have a domain equal to $D_j^i = \{\alpha_j^i, \alpha_j^i + d, \dots, \beta_j^i - d, \beta_j^i\}$. Therefore, by defining the domain as a subset of the size window, we can reduce the dimensionality of the problem without losing any information.

For actual data, these assumptions are reasonable [17, 23]:

Fact 1. *$\beta_j^i \neq \beta_h^i$ whenever $j \neq h$, as the correct allele values are localized in correspondence to the rightmost extreme of the stutters range.*

Fact 2. $|D_j^i| \simeq |D_h^i|$ *(i.e. the ranges covered are roughly the same for each allele) and this value is generally quite small* ($\leqslant 10$).

- m: the number of DNA sizes, at which any allele, of any marker, can produce a detectable band, i.e. $m = |D|$ where $D = \bigcup_{\substack{i=1,\dots,k \\ j=1,\dots,n_i}} D_j^i$ is the *global domain*. If $\alpha = \min D$ and $\beta = \max D$, then the window of DNA sizes on the gel where bands may appear is the range $[\alpha, \dots, \beta]$. However, within this window of length $\beta - \alpha$, only m positions are actually possible, and therefore we look only at those m.
- A^i: the $m \times n_i$ *stutter-matrix* for marker $i = 1, \dots, k$. Each column describes the stutter of an allele for the marker and each row corresponds to a DNA size, decreasing from the largest (first row) to the smallest (last row). For actual data, entries can be assumed to be nonnegative integers, in the range $1, \dots, 1000$, representing the fraction of segments amplified for each size, as a multiple of $1/1000$. Clearly, this value depends on the instruments resolution, which is seldom more accurate than one part in a thousand. Referring to our previous example, let us assume that another allele for the marker in question is $(\mathrm{CA})_{23}$, with the following stutter: 68.5% at 46 bp, 24.6% at 44 bp, 11.2% at 42 bp, 5.6% at 40 bp. The domains for the stutters are $D_1^1 = \{34, 36, 38, 40\}$ and $D_2^1 = \{40, 42, 44, 46\}$. Further, assume that there are some other markers with alleles whose domain contains the sizes 32 and 48 bp, and that the global domain is $D = \{32, 34, 36, 38, 40, 42, 44, 46, 48\}$. Then $m = 9$, and the stutter matrix for this marker will be

$$A^i = \begin{pmatrix} 0 & 0 \\ 685 & 0 \\ 246 & 0 \\ 112 & 0 \\ 56 & 540 \\ 0 & 376 \\ 0 & 64 \\ 0 & 20 \\ 0 & 0 \end{pmatrix}.$$

In this matrix the top row corresponds to molecules of size 48 bases, and the bottom to molecules of size 32. In this example, passing from a row to the next, the size is decreased by 2, since we are looking at dinucleotide repeats. By our definitions, it should be clear, however, that it is not necessarily true that consecutive rows correspond to DNA sizes which are even and/or whose difference is a multiple of some fixed number. In fact, the mapping of matrix rows to DNA sizes will depend on the particular markers and alleles in the pooling. Of course, it will be possible to pool dinucleotide repeats with trinucleotide repeats. Actually, our model does not rely on the markers being repeats at all, but only on being length-polymorphic.

Remark 1. Note that because of Fact 1, possibly after reordering the columns, each A^i can be assumed to be lower triangular (i.e. the entries above the main diagonal are zero); this property will be exploited in the algorithm to be described later.

Let us define $A = (A^1 A^2 \cdots A^k)$.

- b: the $m \times 1$ vector representing the outcome of the gel measurement, i.e. the sum of the stutters for the k pooled markers.
- x: the solution. $x = (x^1, x^2, \ldots, x^k)$ where each $x^i \in \{0,1,2\}^{n_i}$ is a vector of n_i components. In our model, x^i_j represents how many copies of the allele j for marker i are present in the pooling. For each marker there would normally be two alleles (one per chromosome) either equal (homozygous) or different (heterozygous). However, PCR may in some occasions fail to amplify one or both alleles for several reasons. In order to account for this possibility, we will expect the amplification of *at most* 2 alleles per marker, i.e. $\sum_{j \in 1,\ldots,n_i} x^i_j \leqslant 2$. Note that a solution to the problem can be seen as a selection of either 0, 1 or 2 columns from each matrix A^i.

2.1. *The noiseless version of the problem*

Assume that the reading of the gel is done without measurement error. In this purely theoretical situation, because of the very nature of the experiment, there must exist (although not necessarily unique) x as above satisfying $Ax = b$.

We then define the noiseless genotyping problem (NGP) as the following:
(NGP): Given $A = (A^1 \cdots A^k) \in Z^{m \times (n_1 + \cdots + n_k)}$, $b \in Z^m$, find $x = (x^1, x^2, \ldots, x^k) \in Z^{n_1 + \cdots + n_k}$,
s.t.

(i) $Ax = b$,

(ii) $\sum_{j=1}^{n_i} x^i_j \leqslant 2, \quad i = 1, \ldots, k,$

(iii) $x^i_j \in \{0,1,2\}, \; i = 1, \ldots, k, \; j = 1, \ldots, n_i.$

2.2. *Accounting for experimental error*

More realistically, we will have to consider experimental error and accept the case in which no solution is feasible for $Ax = b$, and we are interested in the solution that better fits the data at hand. This would be achieved by removing the constraint (i) and introducing the objective min $\|Ax - b\|$.

Note that if we know the error rates and the error behaviour of the instrument, we can improve our model by helping it with additional constraints. In particular, we may assume that an error function[1] $\varepsilon(v) : Z \mapsto [0,1]$ exists such that for each value v representing the reading of an unknown amount of DNA, $\varepsilon(v)$ is the maximum relative error that we may have encountered. Therefore, if $\widehat{b}$ is the underlying noiseless result of the experiment, from the observed one b we may compute lower (b^-) and upper (b^+) bounds on the possible values of $\widehat{b}$. Then, knowing that $b^- \leqslant \widehat{b} \leqslant b^+$, we may impose the additional constraints $b^- \leqslant Ax \leqslant b^+$ to the model.

To exploit the knowledge of the error, we could decide what the error function is, fix it once and for all and make it part of the model. Or, as we do here, we

[1] Typically, given in the form of a table, describing the instrument accuracy at different resolutions.

could assume that the error function is known, possibly, by a separate module, which prepares the input for our problem. This module will derive b^- and b^+ from b. If no knowledge of the error is assumed, the module will simply set $b_i^- = 0$ and $b_i^+ = +\infty$. In Section 6 on computational results, we will describe the model for the error adopted in our simulations. Note that by considering b^- and b^+ to be input parameters of the problem, we are able to make the problem independent from the particular error function used.

We therefore add the following parameters to the model:

- b^-, b^+: The values b^- and b^+, depending on b and on the error rate of the experiment, are such that $b^- \leqslant \hat{b} \leqslant b^+$ with high probability.

Summarizing, in the presence of errors, the problem of determining the genotyping for a pooled set of markers can be stated as an optimization problem with the objective of minimizing the euclidean distance of the solution from the observed value b, with the further requirement that the solution must lie within a known m-dimensional interval. Formally, we define the genotyping problem (GP) as:

(GP): Given $A = (A^1 \cdots A^k) \in Z^{m \times (n_1 + \cdots + n_k)}$, $b, b^-, b^+ \in Z^m$, find $x = (x^1, x^2, \ldots, x^k) \in Z^{n_1 + \cdots + n_k}$,

s.t.

(i) $b^- \leqslant Ax \leqslant b^+$,

(ii) $\sum_{j=1}^{n_i} x_j^i \leqslant 2, \quad i = 1, \ldots, k,$

(iii) $x_j^i \in \{0, 1, 2\} \quad i = 1, \ldots, k, \; j = 1, \ldots, n_i.$

and $\|Ax - b\|$ is minimum.

The noiseless genotyping problem is a special case of (GP) occurring when $b^- = b^+ = b$. Note that in the presence of errors it is not necessarily true that the solution minimizing the Euclidean distance is the correct one. In fact, for a significantly large error that solution will very likely be wrong. This problem is inherent to this situation and no algorithm can evade it: if the error rate is too large, the data are meaningless and the experiment must be redone. However, for error rates usually encountered in experiments (e.g. ±20%) our algorithm has proved very robust. We also suggest the possibility of finding not just the best solution, but a set of good candidate solutions.

3. The complexity of the problem

In this section we address the computational complexity of the problems [5]. Since (GP) is clearly at least as difficult as (NGP), each negative complexity result for the latter translates automatically to the former.

Theorem 2. *The problem (NGP) is strongly* NP-*complete.*

Proof. We will describe a reduction from the satisfiability problem (SAT) to (NGP).

Let $C_1 \wedge C_2 \wedge \cdots \wedge C_p$ be an instance of (SAT) of p clauses over r boolean variables $z_1, \ldots, z_r$.

We construct $k = r + p$ matrices of $m = r + 2p$ rows each. We label the rows as "selector constraints" (rows $1, \ldots, r + p$) and "clause satisfiers" (rows $r + p + 1, \ldots, r + 2p$). We also name the matrices as "truth assignments" (matrices $1, \ldots, r$) and "fillers" (matrices $r + 1, \ldots, r + p$).

Each truth assignments matrix A^i, $i = 1, \ldots, r$ has two columns, one corresponding to setting $z_i = \text{TRUE}$ and the other for $\bar{z}_i = \text{TRUE}$. In the column for z_i there is a 1 in row i and in all the clause satisfiers rows corresponding to clauses where the literal z_i appears. Analogously, in the column for $\bar{z}_i$ there is a 1 in row i and in all the clause satisfiers rows corresponding to clauses where the literal $\bar{z}_i$ appears.

The filler matrices A^i, $i = r+1, \ldots, r+p$ have r columns each, labeled $0, 1, \ldots, r-1$. In each column j there are only two nonzero entries, namely a 1 in row i and a j in the clause satisfier row corresponding to clause C_{i-r}.

Finally, we set $b_i = 1$ for all selector constraints components and $b_i = r$ for all clause satisfier components. This completes the reduction. An example follows:

$$C_1 \wedge C_2 \wedge C_3 := (z_1 \vee \bar{z}_2 \vee z_3) \wedge (z_2 \vee \bar{z}_3) \wedge (\bar{z}_1 \vee \overline{z_3})$$

becomes

$$
\begin{array}{c}
\\ \\ \\ \\ \\ \\ \\ C_1 \\ C_2 \\ C_3
\end{array}
\begin{array}{c}
z_1 \quad \bar{z}_1 \\
\begin{pmatrix} 1 & 1 \\ & \\ & \\ & \\ & \\ & \\ 1 & \\ & \\ & 1 \end{pmatrix}
\end{array}
\begin{array}{c}
z_2 \quad \bar{z}_2 \\
\begin{pmatrix} & \\ 1 & 1 \\ & \\ & \\ & \\ & \\ & 1 \\ 1 & \\ & \end{pmatrix}
\end{array}
\begin{array}{c}
z_3 \quad \bar{z}_3 \\
\begin{pmatrix} & \\ & \\ 1 & 1 \\ & \\ & \\ & \\ 1 & \\ & 1 \\ & 1 \end{pmatrix}
\end{array}
\begin{array}{c}
\\
\begin{pmatrix} & & \\ & & \\ & & \\ 1 & 1 & 1 \\ & & \\ & & \\ 0 & 1 & 2 \\ & & \\ & & \end{pmatrix}
\end{array}
\begin{array}{c}
\\
\begin{pmatrix} & & \\ & & \\ & & \\ & & \\ 1 & 1 & 1 \\ & & \\ & & \\ 0 & 1 & 2 \\ & & \end{pmatrix}
\end{array}
\begin{array}{c}
\\
\begin{pmatrix} & & \\ & & \\ & & \\ & & \\ & & \\ 1 & 1 & 1 \\ & & \\ & & \\ 0 & 1 & 2 \end{pmatrix}
\end{array}
\begin{array}{c}
\\
x
\end{array}
\begin{array}{c}
\\
\begin{array}{cc} = & 1 \\ = & 1 \\ = & 1 \\ = & 1 \\ = & 1 \\ = & 1 \\ = & 3 \\ = & 3 \\ = & 3. \end{array}
\end{array}
$$

Now, we claim that the derived (NGP) is feasible if and only if the boolean formula is satisfiable. In fact, since $b_i = 1$ for each selector constraint row, a solution to the (NGP) will have to pick one and only one column from each matrix. In particular, restricted to the first r matrices, this pick can be seen as setting the truth value for all the r Boolean variables. Also, since $b_i = r$ for each clause satisfier row but a filler column can account only for at most $r - 1$ in that row, it must be that one of the columns picked in $A^1, \ldots, A^r$ has a 1 corresponding to row C_i, i.e. the truth assignment for a variable satisfies clause C_i. Since this is true for all clauses, the solution identifies a satisfying truth assignment.

Conversely, given a satisfying truth assignment, a solution to (NGP) is obtained as follows. For each matrix i pick column z_i if z_i is true, and column $\bar{z}_i$ otherwise. Further, for each clause C_i let

$$s_i = |\{z_j\colon\ z_j \in C_i, z_j = \text{TRUE}\} \cup \{z_j\colon\ \bar{z}_j \in C_i, z_j = \text{FALSE}\}|.$$

Since C_i is true it must be $s_i \geqslant 1$; then pick from the ith filler the column corresponding to $r - s_i$.

This completes the proof. Note that since the largest number we built in the reduction was r, and $r < m$, this shows that (NGP) is strongly NP-complete. □

The (NGP) bears many similarities with the (SUBSET-SUM) problem [5]. In (SUBSET-SUM) we are given a set of integers J and a target t, and we look for a subset J' s.t. $\sum_{j \in J'} j = t$. Although NP-complete, (SUBSET-SUM) admits a pseudopolynomial algorithm, so that we could have expected a similar algorithm exists for solving (NGP). Theorem 2 rules out the possibility of such an algorithm for the general case, but we will show that in the monodimensional case ($m = 1$) there is an efficient pseudopolynomial dynamic programming solution. Note that in the monodimensional case the matrices A^i can be thought of as (multi)sets of integers, and b is an integer; the problem is then to pick either 0, 1 or 2 integers from each set so that their sum is b.

Theorem 3. *The monodimensional version of (NGP) is* NP-*complete.*

Proof. The proof is very similar to that for Theorem 2, so we will omit the details. Referring to the previous example, now think of each column as a single number, written in base $2r$. The base is large enough so that there can be no carry in any digits of the sum. Then the proof goes exactly as in Theorem 2. □

Even if the monodimensional (NGP) is NP-complete, now the existence of a pseudopolynomial algorithm cannot be ruled out, since the numbers built in the proof are exponentially large in the length of the input. In fact, such an algorithm exists and will be described in Section 5.

It should also be pointed out that some (NGP) instances have exponentially many solutions; as a simple example, take a monodimensional problem with $A^i = \{0\}$, $i = 1, \ldots, k$ and $b = 0$. This problem has 3^k solutions, since we can pick 0 either never, once or twice from each set and still get 0 in the sum. However, these artificial examples are of little or no interest when it comes to real data, for which the set of feasible solutions is generally empty for the (NGP) and rather small for the (GP).

Although the problem in which k is variable is NP-complete, in practical applications it would be impossible to pool together more than a relatively small number of markers, because of technical (maximum capacity of instruments) and biological (the markers would interfere with each other and the assumption of independence would fail) limitations. Therefore, we are more interested in the case where k is fixed, say $2 \leqslant k \leqslant 10$. Of course, when k is fixed the problem becomes polynomial; if $n = \max_{i=1,\ldots,k} n_i$, there are only $O(n^2)$ ways to choose two columns in a submatrix and $O(n^{2k})$ possible solutions to the overall problem. However, for cases of interest to us, n can be as large as 20 and we see that even for $k = 5$, a solution space of 20^{10} points is far too large to be explored by complete enumeration.

3.1. A simple transformation

For simplicity in the following analysis of the problem, we will reduce it to an equivalent one in which we pick just one column from each submatrix. This can be easily achieved. First, add a zero column to each A^i. Then, compute matrices $\hat{A}^i$, $i=1,\dots,k$, where each $\hat{A}^i$ is $m\times\hat{n}_i$, with $\hat{n}_i=n_i(n_i+1)/2$, and each column of $\hat{A}^i$ is the sum of two (not necessarily distinct) columns of A^i. $\hat{A}^i$ will be called the *extended stutter-matrix* for marker i. Note that each column of $\hat{A}^i$ corresponds to one of the four possibilities for the genotyping restricted to marker i, i.e.

- *zero column*: no alleles were amplified.
- *sum of the zero and a nonzero columns of* A^i: only one allele was amplified.
- *sum of two different nonzero columns of* A^i: the individual is heterozygus.
- *sum of two equal nonzero columns of* A^i: the individual is homozygous.

The pre-processing of the data can be done in time $O(kmn^2)=O(mn^2)$ when k is fixed. We therefore may assume that pre-processing is always performed. For the sake of notational simplicity, we will drop the hat on the variables, and restate the problem in its final form as

(GP1): We are given k extended stutter-matrices $A^1,\dots,A^k\in Z^{m\times(n_1+\cdots+n_k)}$, and vectors $b,b^-,b^+\in Z^m$. Find $x=(x^1,\dots,x^k)\in Z^{n_1+\cdots+n_k}$,

s.t.

(i′) $b^-\leqslant Ax\leqslant b^+$,

(ii′) $\sum_{j=1}^{n_i} x_j^i=1,\quad i=1,\dots,k,$

(iii′) $x_j^i\in\{0,1\},\quad i=1,\dots,k,\ j=1,\dots,n_i$

and $\|Ax-b\|^2$ is minimum.

In the following, we will always refer to the problems in this latter form. Note that in order to deal only with integer quantities, we have squared the objective function. This is done without loss of generality, since $u\leqslant v\Leftrightarrow\sqrt{u}\leqslant\sqrt{v}$.

4. A divide and conquer strategy

In this section, we first describe a search strategy that finds all the feasible solutions (see Fig. 3) for (i′), (ii′) and (iii′); then we turn this strategy into a branch and bound [16] algorithm which solves the problem (GP1).

To solve the multidimensional problem, we will reduce it to a sequence of monodimensional ones, starting from row 1 up to row m. Let $y=Ax$; the idea is to first find all the solutions such that $b_1^-\leqslant y_1\leqslant b_1^+$, then, within these, keep those for which $b_2^-\leqslant y_2\leqslant b_2^+$, and so forth. This approach turns out to be practical because of the particular structure of the problem. In fact, we claim that *all the monodimensional problems solved will be instances in which there are only either* 1,2 *or* 3 *numbers per*

```
PROCEDURE DAC($\mathcal{N}$,$r$)
/* finds all the feasible solutions to the genotyping problem */
  if $r=m$ then print all solutions in $F^1(\mathcal{N})\times\cdots\times F^k(\mathcal{N})$ and return, else
  for each $i=1,\dots,k$ do
    Restrict the submatrix $A^i$ to the columns in $F^i(\mathcal{N})$
    Let the set of different entries in the $r$th row be $S^i=\{a_1^i,a_2^i,a_3^i\}$
    Partition the columns as $F^i(\mathcal{N})=F^i_{a_1}(\mathcal{N})\cup F^i_{a_2}(\mathcal{N})\cup F^i_{a_3}(\mathcal{N})$ accordingly.
  Solve a feasibility monodimensional problem with data $S^1,\dots,S^k$, $b_r^-$, $b_r^+$.
  Let $X\subseteq\{0,1\}^{|S^1|+\cdots+|S^k|}$ be the set of all the solutions to this problem.
  if $X=\emptyset$ return, else
  for each solution $x=(x^1,\dots,x^k)\in X$ do
    for $i=1,\dots,k$, let $a_i=S^ix^i$ /* $a_i$ is the value picked from the set $S^i$ */
    Create a new node $\mathcal{N}'$ with $\mathcal{F}(\mathcal{N}')=F^1_{a_1}(\mathcal{N})\times\cdots\times F^k_{a_k}(\mathcal{N})$.
    DAC($\mathcal{N}'$,$r+1$) /* depth-first recursive call */

end
```

Fig. 3. A divide and conquer (DAC) search strategy for (GP1).

set. This will be accomplished by removing all the multiple occurrences of a number in each set.

Before making this statement formal, let us look at an example. A submatrix relative to a marker is lower triangular (by Fact 1, Section 2) and ends with a zero column, e.g.

$$\begin{array}{ccccc}
c_1 & c_2 & c_3 & c_4 & c_5\\
1 & 0 & 0 & 0 & 0\\
2 & 5 & 0 & 0 & 0\\
3 & 6 & 8 & 0 & 0\\
4 & 7 & 9 & 10 & 0.
\end{array}$$

After the transformation described in Section 3.1, this submatrix becomes

$$\begin{array}{ccccccccccccccc}
c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & c_{22} & c_{23} & c_{24} & c_{25} & c_{33} & c_{34} & c_{35} & c_{44} & c_{45} & c_{55}\\
2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
4 & 7 & 2 & 2 & 2 & 10 & 5 & 5 & 5 & 0 & 0 & 0 & 0 & 0 & 0\\
6 & 9 & 11 & 3 & 3 & 12 & 14 & 6 & 6 & 16 & 8 & 8 & 0 & 0 & 0\\
8 & 11 & 13 & 14 & 4 & 14 & 16 & 17 & 7 & 18 & 19 & 9 & 20 & 10 & 0.
\end{array}$$

Looking at the first row, we note that the columns are divided into 3 groups. The monodimensional problem corresponding to the first row has for this marker data set $\{2,1,\dots,1,0,\dots,0\}$ which we can then contract into $\{2,1,0\}$. Now, if the solution to this problem picks a 0 from this set, this corresponds implicitly to picking any of the columns in $\{c_{22},\dots,c_{55}\}$ and discarding those in $\{c_{11},\dots,c_{15}\}$. Analogously, choosing the 1 will select the columns $\{c_{12},\dots,c_{15}\}$ and discard $\{c_{11},c_{22},\dots,c_{55}\}$. The best case is when the solution picks c_{11}, so that we can discard all but one column.

Further, once the selection is made, we may restrict our attention to the selected columns and look at the second row. Again, this row has only a few noncoincident entries. For instance, if we are restricted to columns in $\{c_{22},\dots,c_{55}\}$ then the monodimensional problem for the second row has for this marker data set $\{10,5,0\}$. Analogously, if we are looking at columns in $\{c_{12},\dots,c_{15}\}$ the new set has only two values in it, namely $\{7,2\}$. Finally, for column c_{11} the new data set is $\{4\}$. This way of grouping columns permits to consider (and discard) many solutions at once, and yields a very efficient search algorithm.

Because of constraint (ii′), a solution x can be seen as the incidence vector of a set of columns (alleles) one from each A^i (marker). If $N^i := \{1,\dots,n_i\}$, the set of feasible solutions to (GP1) is a subset $\mathscr{F}$ of $N^1 \times \cdots \times N^k$. We will denote as $\mathscr{F}_r$ the set of solutions feasible for (i′) restricted to rows $1,\dots,r$, so that $\mathscr{F} = \mathscr{F}_m$. The algorithm can be best described as a depth-first visit of a search tree of m levels. Each node $\mathscr{N}$ of level r corresponds to a set $\mathscr{F}(\mathscr{N}) = F^1(\mathscr{N}) \times \cdots \times F^k(\mathscr{N}) \subseteq \mathscr{F}_{r-1}$, where each $F^i(\mathscr{N}) \subseteq N^i$. At the root node $\mathscr{R}$ we have $\mathscr{F}(\mathscr{R}) = N^1 \times \cdots \times N^k$, while the nodes at level m will represent complete solutions to (GP1). When visiting a node $\mathscr{N}$ at level r, we will partition the candidate solutions at $\mathscr{N}$ by splitting each $F^i(\mathscr{N})$ into at most three sets, as $F^i(\mathscr{N}) = F^i_1(\mathscr{N}) \cup F^i_2(\mathscr{N}) \cup F^i_3(\mathscr{N})$ (where some of the $F^i_j(\mathscr{N})$ can be empty) so that

$$\mathscr{F}(\mathscr{N}) = (F^1_1(\mathscr{N}) \cup F^1_2(\mathscr{N}) \cup F^1_3(\mathscr{N})) \times \cdots \times (F^k_1(\mathscr{N}) \cup F^k_2(\mathscr{N}) \cup F^k_3(\mathscr{N})).$$

Then for each $\pi \in \{1,2,3\}^k$ and nonempty set $F^1_{\pi_1}(\mathscr{N}) \times \cdots \times F^k_{\pi_k}(\mathscr{N})$ which is contained in $\mathscr{F}_r$ (i.e. feasible for (i') also at row r) we will create, and visit, a new node $\mathscr{N}'$ at level $r+1$, with $F^i(\mathscr{N}') = F^i_{\pi_i}(\mathscr{N})$ for $i = 1,\dots,k$. If otherwise $\mathscr{F}(\mathscr{N}) \not\subseteq \mathscr{F}_r$, the node $\mathscr{N}$ will be killed and the search resumed from its parent.

The algorithm can be implemented recursively as sketched in Fig. 3.

To start the search we will create a root node $\mathscr{R}$ with $\mathscr{F}(\mathscr{R}) = N^1 \times \cdots \times N^k$ and call DAC($\mathscr{R}$, 1). The fact that in the algorithm each S^i has at most three elements is justified by the following analysis.

Definition 1. We say that a matrix is *ternary* if

- in the first row there are (at most) three different entries, $\{a_1,a_2,a_3\}$; partition the set of columns accordingly in $C_{a_1}, C_{a_2}, C_{a_3}$.
- each nonempty submatrix obtained by keeping the columns in C_{a_j} and rows $2,\dots,m$ is ternary.

Some results are immediate: a column vector is a ternary matrix; a lower triangular matrix is ternary; adding the same constant vector to each column of a matrix does not change it being ternary or not.

Let M^i, $m \times n$ stutter-matrix for a marker and A^i the extended stutter-matrix obtained after transformation 3.1. Then we have

```
PROCEDURE BAB($\mathcal{N}, r, lb$)
/* finds the optimal solution to the genotyping problem. $lb$ is the lower bound for node $\mathcal{N}$*/
begin
  if $lb \geqslant \widehat{v}$ then return, else
  if $r = m$ then
    for all $x \in F^1(\mathcal{N}) \times \cdots \times F^k(\mathcal{N})$ do
      if $\|Ax - b\|^2 < \widehat{v}$ then        /* update the optimum */
        $\widehat{v} := \|Ax - b\|^2$
        $\widehat{x} := x$
    return
  else
    for each $i = 1, \ldots, k$ do:
      Restrict the submatrix $A^i$ to the columns in $F^i(\mathcal{N})$.
      Let the set of different entries in the $r$th row be $S^i = \{a_1^i, a_2^i, a_3^i\}$.
      Partition the columns as $F^i(\mathcal{N}) = F_{a_1}^i(\mathcal{N}) \cup F_{a_2}^i(\mathcal{N}) \cup F_{a_3}^i(\mathcal{N})$ accordingly.
    Solve a feasibility monodimensional problem with data $S^1, \ldots, S^k$, $b_r^-$, $b_r^+$.
    Let $X \subseteq \{0,1\}^{|S^1| + \cdots + |S^k|}$ be the set of all the solutions to this problem.
    if $X = \emptyset$ return, else
    Sort $X$ by nondecreasing $|Sx - b_r|$.
    for each solution $x = (x^1, \ldots, x^k) \in X$ do
      for $i = 1, \ldots, k$, let $a_i = S^i x^i$ /* $a_i$ is the value picked from the set $S^i$ */.
      Create a new node $\mathcal{N}'$ with $\mathcal{F}(\mathcal{N}') = F_{a_1}^1(\mathcal{N}) \times \cdots \times F_{a_k}^k(\mathcal{N})$.
      BAB $\left(\mathcal{N}', r+1, lb + (\sum_{i=1}^k a_i - b_r)^2\right)$ /* depth-first recursive call */

end
```

Fig. 4. The branch-and-bound procedure for (GP1).

Theorem 4. *A^i is ternary.*

Proof. By induction on m.

Since M^i is lower triangular it has only one nonzero entry in its first row, say a; then the entries in the first row of A^i are $\{2a, a, 0\}$. So, if $m = 1$, A^i is ternary. Otherwise, let us assume that if M^i has $m-1$ rows it generates a ternary A^i. For a general $m > 1$, partition the columns into C_{2a}, C_a, C_0. Refer to the figure in the previous example: C_{2a} is a column vector and henceforth ternary. Subtracting the first column of M^i from C_a leaves a triangular matrix, again ternary. Finally, the submatrix in columns C_0, rows $2, \ldots, m$ is the transformation of M^i restricted to columns $2, \ldots, n$, rows $2, \ldots, m$ and hence, by induction, is ternary. □

Corollary 5. *Each time the algorithm DAC is entered, the matrices A^i restricted to columns $F^i(\mathcal{N})$, rows $r, \ldots, m$, are ternary.*

The main consequence of this analysis is that each monodimensional problem solved in step 3 of the algorithm is relatively easy, and the resulting search is very fast. This is also because on average there are only a few solutions to each monodimensional problem. This follows again from the nature of the PCR stutter data, which has very different entries in each set that are not easily interchangable in different solutions. This relative uniqueness of PCR stutter patterns can be increased by using different PCR experimental conditions that accentuate each marker's stuttern pattern. In our test runs, most of the time we found only one or no solution to each monodimensional problem, and never more than 5. The resulting search tree has therefore a low-bounded degree, and depth m, so that we can say that algorithm DAC, satisfactorily solves the problem.

It is straightforward to turn this search algorithm into a branch-and-bound, by adding the computation of a lower bound at each node $\mathcal{N}$ of level $r \in \{1,\dots,m\}$. Let $y = Ax$, where x is the incidence vector of any solution in $\mathscr{F}(\mathcal{N})$. Then for all $x \in \mathscr{F}(\mathcal{N})$, $y_1,\dots,y_{r-1}$ are fixed, and $\sum_{h=1}^{r-1}(y_h - b_h)^2$ is a lower bound to the value at the node. If B^i are the matrices resulting from A^i restricted to the columns in $F^i(\mathcal{N})$, the optimal value at the node is $\sum_{h=1}^{r-1}(y_h - b_h)^2 + \min \sum_{h=r}^{m}(B_h x - b_h)^2$, where the min is taken over all $x \in \mathscr{F}(\mathcal{N})$ and B_h denotes the hth row of B. So the node corresponds to a problem similar to the original but restricted to rows $r,\dots,m$. We then branch as in *DAC*, but now we first order the subproblems of $\mathcal{N}$ by nonincreasing $(y_r - b_r)^2$ in order to explore the most promising subproblems first.

Let $\widehat{x}$ be the best solution found at any point of the search and $\widehat{v} = \|Ax - b\|^2$ be its value. The branch-and-bound algorithm is described in Fig. 4. The main program will set $\widehat{v} := +\infty$ and call BAB($\mathscr{R}$,1,0). Finally, we note that the branch-and-bound algorithm can be easily modified so that it returns not just one solution but the p best solutions, where p is a user input parameter. From our experiments it turns out that $p \leqslant 5$ is usually enough to retrieve the correct solution among the p best. Retrieving p solutions instead of just one may be preferable when the noise is high or in particularly crucial experiments. In such situations, the algorithm may be used as a tool for an expert to finally decide on the correct genotype (possibly by a consensus method) or re-execute the experiment to reduce the noise.

5. The monodimensional problem

We will be interested in the following form of the monodimensional genotyping problem:

(MGP): Given k sets (row vectors) $A^1,\dots,A^k$, where $|A^i| = n_i$, and two bounds $b^- \leqslant b^+ \in Z$, find all $x = (x^1, x^2, \dots, x^k) \in Z^{n_1 + \cdots + n_k}$

s.t.

(i) $b^- \leqslant Ax \leqslant b^+$,

(ii) $\sum_{j=1}^{n_i} x_j^i = 1, \; i = 1,\dots,k$,

(iii) $x_j^i \in \{0,1\}, \; i = 1,\dots,k, \; j = 1,\dots,n_i$.

As shown in the previous section, each time we solve an (MGP) there are at most three numbers per set; the solution space has at most 3^k points, so that even an exaustive search may be conceivable when k is small. However, for $k \geqslant 7$ exaustive search starts to be impractical, also because the subroutine for the monodimensional problem is called several times within the algorithm DAC.

Therefore, in this section we describe two efficient algorithms for (MGP). The first has mainly theoretical interest and it is based on a dynamic programming approach [4]. The second is faster (expecially when the entries of A and b are large), and is based on a branch-and-bound approach.

5.1. A pseudopolynomial dynamic programming algorithm

In order to solve (MGP) we can build a table $T(r,s)$ with $r=1,\dots,k$, $s=1,\dots,b^+$. Each entry $T(r,s)$ is a boolean variable representing the event "there is a partial solution $x^1,\dots,x^r$ such that $\sum_{i=1,\dots,r} A^i x^i = s$". Recursively, this can be written as

$$T(r,s) = \begin{cases} \text{TRUE} & \text{if } r=1 \text{ and } s\in A^1, \\ \text{FALSE} & \text{if } r=1 \text{ and } s\notin A^1, \\ \bigvee_{a_j\in A^r} T(r-1, s-a_j) & \text{if } r>1. \end{cases}$$

In parallel with the construction of table T, we keep a table B of backpointers that allow us to recover all the solutions that yield a certain sum. The final solution will be obtained by looking at all the entries $j \leqslant b^+$ that are TRUE. The algorithm has complexity $O(kb^+)$ i.e. pseudopolynomial. It can be further speeded up by avoiding to compute all those entries of T which beforehand we know are going to be FALSE. Let $m_i = \min A^i$, $M_i = \max A^i$; then the only possible TRUE entries in the $(r+1)$th row of T are within columns $c_r + m_{r+1}$ and $C_r + M_{r+1}$ where c_r (C_r) is the minimum (maximum) TRUE entry of row r (clearly $c_1 = m_1$ and $C_1 = M_1$).

5.2. A branch-and-bound algorithm

In this section we describe a simple branch-and-bound strategy to explore the search space for the monodimensional genotyping problem.

As in section 4, let $N^i := \{1,\dots,n_i\}$. The problem amounts to finding all k-tuples $(j_1,\dots,j_k) \in N^1 \times \cdots \times N^k$ such that $b^- \leqslant \sum_{i=1}^k a^i_{j_i} \leqslant b^+$. We build these k-tuples incrementally, starting from component $d=1$ up to component $d=k$. At a generic step, if $d<k$, then $\sigma = \sum_{i=1}^d a^i_{j_i}$ is the partial sum built so far. Therefore, $\sigma^- = \sigma + \sum_{i=d+1}^k \min A^i$ and $\sigma^+ = \sigma + \sum_{i=d+1}^k \max A^i$ are a lower and an upper bound on the value of the final solution. If $\sigma^+ < b^-$ or $\sigma^- > b^+$ we can prune the node. Otherwise we can branch by trying recursively all the possible values for the $(d+1)$th component of the k-tuple.

The search tree has at most k levels. In order to prune as many nodes as possible, we can reorder the sets so that $i<j \Rightarrow |A^i| \leqslant |A^j|$. Similarly, the computation of maxima and minima for all sets can be done in a preliminar step before starting the search. Due

to its simplicity and to the fact that k is small and $n_i \leqslant 3$ for $i = 1, \ldots, k$, this algorithm has proved very fast and was used as a subroutine for the general procedure.

6. Computational results

The procedures were coded in C++ and run on a Sun Workstation. To test the algorithm we have performed different simulations, varying the parameters of the problem, the error rates, and the way the data were generated. The test problems are described in detail later, but depending on how we generated them, at a top-level, we can classify the problems into two main classes: (semi) "real" and "random". In generating the problems, we followed these common steps:

1. *Choice of markers and number of alleles*: For the "real" problems we obtained information on actual microsatellite markers and corresponding alleles. For the "random" problems, we randomly generated number and size of the alleles for each marker.
2. *Choice of stutter patterns*: For both class of problems we used some real stutter patterns. Since we did not have many samples of real stutters, we generated new ones by addding random perturbations to the known ones. Further, for the "random" problems, we created more new stutters by generating random distributions in the form of exponential decays.
3. *Choice of genotypes*: The problems were finally generated by picking random genotypes, i.e. choosing the alleles for each marker. In the "real" problems, we used the published frequencies to decide if the individual is heterozygous or homozygous, and which alleles are more likely to be chosen. In the "random" problems, these choices were made at random.
4. *Model for the error*: Once a genotype (called the *correct* or *true* solution) had been chosen, its noiseless value $\widehat{b}$ was computed. Finally, b was obtained from $\widehat{b}$ by introducing some random error in each component.

   In order to do this, we have adopted the following model for the error. Since we can expect the error rates to be dependent on the magnitude of the values measured (i.e. the relative error in reading a weak signal should be greater than that of reading a strong signal), we consider two error parameters: ε_m, the experimental error when reading "small" amounts of DNA (e.g. $\varepsilon_m = \pm 50\%$) and ε_M, the experimental error for "large" amounts of DNA (e.g. $\varepsilon_M = \pm 10\%$). As we mentioned in Section 2.2, the error rates are generally given as a table, with entries in the range $\varepsilon_M, \ldots, \varepsilon_m$. We obtain this table by interpolating logarithmically between ε_M and ε_m. Let v_m (v_M) be the minimum (maximum) over all positive entries of $\widehat{b}$. For any DNA amount $v_m \leqslant v \leqslant v_M$, our error function is given by $\varepsilon(v) = \varepsilon_M + E(\mathrm{e}^{\lambda u} - 1)$ where $E = (\varepsilon_m - \varepsilon_M)/(\mathrm{e}^{\lambda} - 1)$, $u = (v_M - v)/(v_M - v_m)$ and λ is a parameter that determines how smooth the change from one error limit to the other is.

   To generate b from $\widehat{b}$, we assume that the error has a normal distribution, with zero mean. Therefore, we draw each b_i from the normal distribution $\mathrm{N}(\widehat{b}_i, \sigma)$ where

$\sigma = \varepsilon(\widehat{b}_i)\widehat{b}_i/4$ is chosen so that the probability $P[(1-\varepsilon(\widehat{b}_i))\widehat{b}_i \leqslant b_i \leqslant (1+\varepsilon(\widehat{b}_i))\widehat{b}_i] > 99\%$. The lower and upper bounds are $b_i^- = (1-\varepsilon(\widehat{b}_i))\widehat{b}_i$ and $b_i^+ = (1+\varepsilon(\widehat{b}_i))\widehat{b}_i$.

For our tests we have defined three error functions, which we call *small-noise*, *medium-noise* and *large-noise*. This classification is based on the error allowed in the data. In particular, we have $\varepsilon_M = \pm 3\%$, $\varepsilon_m = \pm 20\%$ for small-noise problems, $\varepsilon_M = \pm 5\%$, $\varepsilon_m = \pm 30\%$ for medium-noise problems and $\varepsilon_M = \pm 10\%$, $\varepsilon_m = \pm 50\%$ for large-noise problems. We used large-noise error for the "real" problems, while the "random" problems were generated for all three classes of error. The reason is that the "real" problems had on average fewer alleles per marker than the "random", thereby resulting simpler to solve.

In the tests that follow, we have set the number of solutions p to be found by the algorithm to 5. We consider a problem solved if the correct solution is within the p solutions returned by the algorithm. If the least-squares solution is indeed the true solution, we say that the problem was *unbiased*. Note that being biased or not is a property of the problem and not of the algorithm. The algorithm will always find the correct solution for unbiased problems, while for biased problems, there may be wrong solutions which achieve better objective function value than the true one. Clearly, this depends on the amount of bias in the data, which in turn depends on the experimental error. As we remarked at the end of Section 4, the algorithm could be used as a decision support system, and the output be interpreted by an expert. If more solutions have a very similar value, it may be the case that too much error was introduced and the experiment must be redone. However, when the error rate is small, real-life problems are unbiased and the algorithm can be trusted to return always the true solution.

6.1. "Real" problems from CHLC

We generated a first set of problems by using as much as possible real data for our simulations. In particular, we accessed through the web a data base available from CHLC[2] (Cooperative Human Linkage Center), which reports many informations about real CA-repeat markers and corresponding allele distributions. We will call this data base DB1. A typical entry of DB1 is shown in Table 1, and contains for each marker the frequency with which an individual is heterozygous, the number of alleles, and for each allele its size and relative frequency. The data base contains data for 263 markers, whose alleles vary in size from 65 to 333 bp. In Table 2 we show the distribution of alleles sizes in the data base. The markers in the data base have a minimum of 2 alleles and a maximum of 18; the average number of alleles is 7.8. The problems were generated for $k = 5, 6, \ldots, 10$ markers. For each value of k we generated 10 problems in the following way. We randomly pick, in uniform way, k markers in the data base. For each marker we randomly decide the genotype (i.e. homozygous or heterozygous) with a probability given by the published frequency. Finally, we choose, respectively, one or two alleles, where the probability of an allele to be chosen is equal to its published

[2] at `http://www.chlc.org/ChlcMarkers.html`

Table 1
A record of the CHLC data base DB1

```
Marker: MFD008 initial typing data
Locus: D8S84
Map: 8q13-q21.2
Accession: M23608
Accession2: M23386
Primerno: 027,028
Hetero: 0.58
Noalleles: 8
Allele1: 195 = 0.01 193 = 0.03 191 = 0.07 189 = 0.02
Allele2: 187 = 0.27 185 = 0.54 183 = 0.04 181 = 0.01
Repsequenc: ACCTGAGTTT (AC)20.5 GTACAGGGTA
Geno133101: 193,185
Geno133102: 185,185
Reference: AM J HUM GEN 44:388-396, 1989
```

Table 2
Total number of alleles for each size. Data from data base DB1

Allele size	61 to 80	81 to 100	101 to 120	121 to 140	141 to 160	161 to 180	181 to 200	201 to 220	221 to 240	241 to 260	261 to 280	281 to 300	301 to 320	321 to 340
No. alleles	58	235	331	356	314	283	210	115	68	28	28	8	2	2

relative frequency. As far as the stutters are concerned, we used real stutters from some actual markers [17]. All of these stutters have domains of cardinality between 5 and 9. Of these stutters we had 21. Extra stutters for our problems were obtained by first picking uniformly one of the real stutters, and then changing the entries by adding to each a random perturbation.

The error function used was the most difficult, i.e. large-noise. The results for this class of problems are shown in Table 3. All the problems were solved to optimality (there were no biased problems). For each set of 10 problems, we report average, minimum and maximum value of global domain cardinality, number of alleles per marker and running time. Further, we indicate the percentage of solved and biased problems. The problems turned out to be relatively easy, since the data base contains markers whose alleles sizes range within a window of 280 bp and therefore a random choice of markers may result in a problem with little overlap. To test the algorithm in a more constrained case, we extracted from the original data base DB1 two more data bases: DB2 consisting of 162 markers all of whose alleles have sizes within 101 and 200 bp; and DB3, consisting of 89 markers with allele sizes ranging in 101 and 160 bp. In Table 4 we report a description of the data bases. We generated 60 + 60 more problems in the same way as before, only that now we used the information contained in DB2 (*dense* problems) and DB3 (*very dense* problems) to generate the data. The results are reported in Tables 5 and 6. Again, all the problems were solved and the average running times are smaller than one minute. There were three biased

Table 3
Problems from data base DB1

No. markers k	Perc. solved	Perc. biased	Global domain cardinality m			No. alleles per marker			Time (s)		
			Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
5	10/10	0/10	60.6	49	73	8.5	4	13	3.3	2.6	6.2
6	10/10	0/10	73.4	55	85	8.9	4	16	5.5	3.7	12.7
7	10/10	0/10	75.4	61	85	8.8	4	15	6.9	3.9	12.7
8	10/10	0/10	79.6	76	84	9.0	6	13	7.4	6.3	9.3
9	10/10	0/10	86.4	77	95	9.2	4	16	8.8	6.7	11.3
10	10/10	0/10	93.0	85	95	8.2	3	13	20.4	10.4	37.0

Table 4
The data bases used for tests

Data base	No. markers	Range (bp)	Avg no. alleles per marker	Min no. alleles per marker	Max no. alleles per marker
DB1	263	65–333	7.8	2	18
DB2	162	101–200	7.9	3	18
DB3	89	101–160	7.7	3	14

Table 5
Problems from data base DB2

No. markers k	Perc. solved	Perc. biased	Global domain cardinality m			No. alleles per marker			Time (s)		
			Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
5	10/10	0/10	55.6	42	69	8.2	5	13	3.0	2.1	4.3
6	10/10	0/10	64.3	45	82	9.1	5	19	4.3	3.4	6.7
7	10/10	0/10	64.6	57	77	8.5	4	12	6.7	4.0	10.2
8	10/10	0/10	61.7	52	77	8.5	4	15	10.4	6.6	12.4
9	10/10	0/10	70.0	58	81	8.6	5	13	21.7	7.5	40.3
10	10/10	0/10	79.2	70	83	8.8	5	19	26.1	9.5	72.2

problems. We remind that for biased problems, the best solution was not the true one, but the true solution was still retrieved within the p best. By inspection, we noted that the number of wrongly called alleles in the solution which achieved the best objective value was never larger than two.

6.2. "Random" problems

After solving the "real" problems, we decided to generate some new, more difficult ones. In particular, difficult problems are obtained when the markers are very similar to each other, each marker has a large number of alleles, and they all span a small global domain. We created a set of problems with these features, by fixing the number of alleles to 15 for each marker and generating the stutters within a global domain of

Table 6
Problems from data base DB3

No. markers k	Perc. solved	Perc. biased	Global domain cardinality m			No. alleles per marker			Time (s)		
			Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
5	10/10	0/10	38.0	29	49	8.6	5	13	6.7	3.8	9.8
6	10/10	0/10	48.0	43	51	9.1	5	13	5.2	4.0	6.7
7	10/10	1/10	51.8	46	55	8.3	4	13	11.3	5.5	31.4
8	10/10	1/10	44.8	35	55	8.9	4	15	30.4	6.4	85.9
9	10/10	1/10	56.0	48	65	8.5	4	15	32.5	7.7	61.2
10	10/10	0/10	51.2	43	59	8.3	4	15	55.7	17.5	91.4

Table 7
Random problems on $k = 5,6,7$ markers, 15 alleles per marker

No. markers k	Noise	Perc. solved	Perc. biased	Avg no. wrong alleles	Global domain cardinality m			Time (min:s)		
					Avg	Min	Max	Avg	Min	Max
5	Small	10/10	0/10	0	26.8	23	28	0:49	0:39	1:00
5	Medium	10/10	2/10	0.5	25.8	24	28	0:56	0:49	1:06
5	Large	10/10	2/10	0.4	26.6	24	28	1:03	0:55	1:11
6	Small	10/10	1/10	0.4	25.8	24	28	1:45	1:35	1:55
6	Medium	10/10	1/10	0.2	25.9	24	28	1:57	1:48	2:10
6	Large	9/10	5/10	2.2	26.8	25	28	2:33	2:18	2:52
7	Small	10/10	0/10	0	27.4	26	28	3:19	3:01	3:35
7	Medium	10/10	0/10	0	26.5	25	28	3:40	3:22	4:02
7	Large	9/10	5/10	1.4	25.9	22	28	4:15	3:44	4:44

small cardinality ($25 \leqslant m \leqslant 28$ on average). As expected, these problems turned out to be quite harder, and we considered only $k = 5,6$ or 7 markers. For each value of k we generated 3 sets of 10 problems each, one set per each error function (small, medium and large noise). The stutters were generated in two ways. Again, some were obtained by slightly randomly modifying some stutters coming from real data [17]. Others were generated completely at random, by simulating an exponential decay and normalizing the result (i.e. all the columns are scaled so that the sum of the entries is a constant. This reflects the fact that in the actual experiment a constant amount of DNA is used for each marker).

The results are reported in Table 7. It should be noted that in $\frac{88}{90}$ cases the algorithm found the correct solution and $\frac{74}{90}$ times this was actually the best in the least-squares sense (i.e. the problems were unbiased). For completeness, we also report the average number of alleles that are wrongly called by the best solution (i.e. which are in the least square solution but not in the correct solution). In all but three instances the correct solution was indeed among the two best and hence a smaller value for p could have been used, this way speeding up the search process. The running times are in the order of minutes, showing the procedure is effective and could possibly be adopted in actual lab experiments.

7. Conclusions

In this paper, we focused on a key bottleneck in current genetic analysis: the gel electrophoresis readout step in multiplexed length-polymorphic (e.g. CA-repeat) marker studies. The authors' previous work [18] with deconvolution methods for exploiting PCR stutter artifact had suggested a novel solution to this bottleneck. Specifically, by using each marker's PCR stutter artifact as a unique signature for that marker, different markers corresponding to the *same* size window could be loaded together onto the gel, and then later deconvolved using a computer program. This paper explored branch-and-bound and dynamic programming algorithms that could perform this stutter-based deconvolution analysis. Our theoretical analysis and computer simulations suggest that an order of magnitude improvement in gel throughput may be computationally feasible.

The ultimate proof of our computational method will be in its efficacy on laboratory data. We are currently working with laboratory-based collaborators to develop quantitative fluorescent data on an Applied Biosystems automated DNA sequencer for stutter-multiplexed markers. A key challenge will be handling differential PCR amplification of differently sized alleles. Our recent work on software systems for fully automated microsatellite genotyping (see FAST-MAP at `http://www.cs.cmu.edu/~genome/FAST-MAP.html` or TrueAllele^{TM} at `http://www.cybergenetics-inc.com`) suggests that precalibration of these differential amplification ratios may suffice. We expect to refine the combinatorial optimization algorithms presented here as these algorithms are assessed on actual genetic marker data.

References

[1] F. Collins, Positional cloning: let's not call it reverse anymore, Nature Genetics 1(1) (1992) 3–6.

[2] J.L. Davies, Y. Kawaguchi, S.T. Bennett, J.B. Copeman, H.J. Cordell, L.E. Pritchard, P.W. Reed, S.C.L. Gough, S.C. Jenkins, S.M. Palmer, K.M. Balfour, B.R. Rowe, M. Farrall, A.H. Barnett, S.C. Bain, J.A. Todd, A genome-wide search for human type 1 diabetes susceptibility genes, Nature 371(6493) (1994) 130–136.

[3] C. Dib, S. Faure, C. Fizames, D. Samson, N. Drouot, A. Vignal, P. Millasseau, S. Marc, J. Hazan, E. Seboun, M. Lathrop, G. Gyapay, J. Morissette, J. Weissenbach, A comprehensive genetic map of the human genome based on 5,264 microsatellites, Nature 380 (1996) 152–154.

[4] S.E. Dreyfus, A.M. Law, The Art and Theory of Dynamic Programming, Academic Press, New York, 1977.

[5] M.R. Garey, D.S. Johnson, Computers and Intractability, a Guide to the Theory of NP-Completeness, W.H. Freeman and Co., San Fransisco, CA, 1979.

[6] G. Gyapay, J. Morissette, A. Vignal, C. Dib, C. Fizames, P. Millasseau, S. Marc, G. Bernardi, M. Lathrop, J. Weissenbach, The 1993–94 Genethon Human Genetic Linkage Map, Nature Genetics 7(2) (1994) 246–339.

[7] J.M. Hall, C.A. LeDuc, A.R. Watson, A.H. Roter, An approach to high-throughput genotyping, Genome Res. 6(9) (1996) 781–790.

[8] X.Y. Hauge, M. Litt, A study of the origin of 'shadow bands' seen when typing dinucleotide repeat polymorphisms by the PCR, Hum. Mol. Genetics 2(4) (1993) 411–415.

[9] A.J. Jeffreys, J.F.Y. Brookfield, R. Semeonoff, Positive identification of an immigration test-case using human DNA fingerprints, Nature 317 (1985) 818–819.

[10] R.M. Karp, Mapping the Genome: Some Combinatorial Problems Arising in Molecular Biology, Proceedings 25th ACM STOC, 1993, pp. 278–285.

[11] E.S. Lander, N.J. Schork, Genetic dissection of complex traits, Science 265 (1994) 2037–2048.

[12] G.M. Lathrop, J.M. Lalouel, Efficient computations in multilocus linkage analysis, Am. J. Hum. Genetics 42 (1988) 498–505.

[13] T.C. Matise, M.W. Perlin, A. Chakravarti, Automated construction of genetic linkage maps using an expert system (MultiMap): application to 1268 human microsatellite markers, Nature Genetics 6(4) (1994) 384–390.

[14] K.B. Mullis, F.A. Faloona, S.J. Scharf, R.K. Saiki, G.T. Horn, H.A. Erlich, Specific enzymatic amplification of DNA in vitro: the polymerase chain reaction, Cold Spring Harbor Symp. Quant. Biol. 51 (1986) 263–273.

[15] J. Ott, Analysis of Human Genetic Linkage, Revised ed., The Johns Hopkins University Press, Baltimore, MA, 1991.

[16] C.H. Papadimitriou, K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall Inc., Englewood Cliffs, NJ, 1982.

[17] M.W. Perlin, M.B. Burks, R.C. Hoop, E.P. Hoffman, Toward fully automated genotyping: allele assignment, pedigree construction, phase determination, and recombination detection in duchenne muscular dystrophy, Am. J. Hum. Genetics 55 (1994) 777–787.

[18] M.W. Perlin, G. Lancia, S.K. Ng, Toward fully automated genotyping: genotyping microsatellite markers by deconvolution, Am. J. Hum. Genetics 57 (1995) 1199–1210.

[19] P.W. Reed, J.L. Davies, J.B. Copeman, S.T. Bennett, S.M. Palmer, L.E. Pritchard, S.C.L. Gough, Y. Kawaguchi, H.J. Cordell, K.M. Balfour, S.C. Jenkins, E.E. Powell, A. Vignal, J.A. Todd, Chromosome-specific microsatellite sets for fluorescence-based, semi-automated genome mapping, Nature Genetics 7(3) (1994) 390–395.

[20] N. Risch, Linkage strategies for genetically complex traits, Am. J. Hum. Genetics 46 (1990) 222–253.

[21] D.A. Schwengel, A.E. Jedicka, E.J. Nanthakumara, J.L. Weber, R.C. Levitt, Comparison of fluorescence-based semi-automated genotyping of multiple microsatellite loci with autoradiographic techniques, Genomics 22 (1994) 46–54.

[22] L.S. Schwartz, J. Tarleton, B. Popovich, W.K. Seltzer, E.P. Hoffman, Fluorescent multiplex linkage analysis and carrier detection for Duchenne/Becker muscular dystrophy, Am. J. Hum. Genetics 51 (1992) 721–729.

[23] R.N. Smith, Accurate size comparison of short tandem repeat alleles amplified by PCR, BioTechniques 18(1) (1995) 122–128.

[24] J.D. Watson, M. Gilman, J. Witkowski, M. Zoller, Recombinant DNA Scientific American Books, W.H. Freeman and Co., San Fransisco, CA, 1992.

[25] J. Weber, P. May, Abundant class of human DNA polymorphisms which can be typed using the polymerase chain reaction, Am. J. Hum. Genetics 44 (1989) 388–396.

[26] D.E. Weeks, K. Lange, The affected pedigree member method of linkage analysis, Am. J. Hum. Genetics 42 (1988) 315–326.

[27] K.J. Wu, A. Stedding, C.H. Becker, Matrix-assisted laser desorption time-of-flight mass spectrometry of oligonucleotides using 3-hydroxypicolinic acid as an ultraviolet-sensitive matrix, Rapid Commun. Mass Spectrom. 7 (1993) 142–146.

[28] J.S. Ziegle, Y. Su, K.P. Corcoran, L. Nie, P.E. Mayrand, L.B. Hoff, L.J. McBride, M.N. Kronick, S.R. Diehl, Application of automated DNA sizing technology for genotyping microsatellite loci, Genomics 14 (1992) 1026–1031.