Finding founder sequences from a set of recombinants

Esko Ukkonen *

Department of Computer Science, P.O Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki, Finland
ukkonen@cs.helsinki.fi

Abstract. Inspired by the current interest in the so–called haplotype blocks we consider a related, complementary problem abstracted from the following scenario. We are given the DNA or SNP sequences of a sample of individuals from a (human) population. The population is assumed to have evolved as an isolate, founded some generations ago by a relatively small number of settlers. Then the sequences in our given sample should be a result of recombinations of the corresponding sequences of the founders, possibly corrupted by (rare) point mutations. We are interested in finding plausible reconstructions of the sequences of the founders. Formulated as a combinatorial string problem, one has to find a set of *founder sequences* such that given sequences can be composed from fragments taken from the corresponding locations of the founder sequences. The solution can be optimized for example with respect to the number of founders or the number of crossovers. We give polynomial–time algorithms for some special cases as well as a general solution by dynamic programming.

1 Introduction

Currently, DNA sequences and haplotyped SNP sequences are available in rapidly growing amounts. Therefore it is realistic to consider a situation in which we are given DNA sequences or dense SNP sequences of a sample of individuals of the same species, such as humans. From the sequence sample, one can attempt to uncover the structure of the variation of the sequences. The findings can be, for example, so–called haplotype blocks [1, 9, 2, 6] or phylogenies, e.g. [3].

If our given sequences are from individuals sampled from an isolated population we can hope to see a quite clear structure. Such a population has been evolved as an isolate that was originally founded by a relatively small number of settlers, called the founders, perhaps only a few generations ago. There can even be several such bottleneck periods in the

* Supported by the Academy of Finland under grant 22584.

history of the sampled population. In such a case the sequences in our given sample should be a result of iterated recombinations of the corresponding sequences of the founders. In addition, some corruption by point mutations is possible. This is, however, rare and therefore will be ignored in our present study.

We want to build plausible reconstructions of the sequences of the founders from the given sequences. Seeing potential founder sequences and crossover points associated with them can be useful for example in the analysis of the possible haplotype blocks and in the study of genetic linkage in general. The given sequences are strings of equal length. The task is, according to our simple abstract formulation, to find a set of *founder sequences* such that the given sequences can be composed of fragments taken from the corresponding locations of the founder sequences.

Two optimality criteria for the solution will be studied. It is of interest to estimate the size of the founder population as well as the number of generations after the founders, as reflected by the fragmentation of the founder sequences (see e.g. [7]). Therefore we consider the problems of minimizing the size of the founder set as well as finding founder sets that explain the given sequences in smallest number of crossovers. It is immediate that these are contradictory goals for optimization.

A general solution method based on dynamic programming will be developed. In some special cases the method is shown to run in polynomial time.

In the literature, Hudson and Kaplan [4] give an algorithm for computing a lower bound for the number of recombinations but they do not try to reconstruct the sequences. Kececioglu and Gusfield [5] consider the problem of reconstructing step–by–step history of recombinations and hence the approach has more explicit phylogenetic nature than adopted here. Zhang et al [10] observed that haplotype blocking and related problems can optimally be solved by dynamic programming. Here we will use similar algorithms as building blocks of our methods.

The paper is organized as follows. The sequence reconstruction problem is formalized in Section 2. A convenient equivalent formulation as a coloring problem is also given. In Section 3 we develop a technique for eliminating crossovers in the reconstruction and analyze its behaviour in some special cases. Section 4 gives a general dynamic programming algorithm which aims at finding long recombination fragments and a small number of founder sequences.

2 The founder sequence reconstruction problem

Let the set of *recombinants* $\mathcal{C} = \{C_1, C_2, \ldots, C_m\} \subseteq \Sigma^n$ be a set of sequences of length n over alphabet Σ. Each C_i is written as $C_i = c_{i1}c_{i2}\ldots c_{in}$. In a typical application the recombinants are haplotyped SNP sequences in alphabet $\Sigma = \{0, 1\}$ where the two symbols encode the two most common alleles of each SNP. In the case of DNA, we have $\Sigma = \{A, C, G, T\}$, of course.

A set $\mathcal{F} = \{F_1, \ldots, F_k\} \subseteq (\Sigma \cup \{-\})^n$ is called a *founder set* of $\mathcal{C}$ and each F_i is a *founder sequence* if $\mathcal{C}$ has a *parse* in terms of $\mathcal{F}$. This is the case if each $C_i \in \mathcal{C}$ has a decomposition into non–empty *fragments* f_{ih} such that $C_i = f_{i1}f_{i2}\cdots f_{ip_i}$ and each f_{ih} occurs in some $F \in \mathcal{F}$ exactly in the same location as in C_i. That is, one can write $F = \alpha f_{ih}\beta$ where sequence α is of length $|\alpha| = |f_{i1}\cdots f_{ih-1}|$ and $|\beta| = |f_{ih+1}\cdots f_{ip_i}|$. We always assume that a parse is *reduced* in the sense that two successive fragments f_{ih} and f_{ih+1} are always from different elements of $\mathcal{F}$. The extra symbol '$-$' can occur in the founder sequences but not in the recombinant sequences of $\mathcal{C}$. This symbol indicates positions in the founder sequences that are not reconstructed; hence some founders can be fragmented. Similarly, our given recombinants $\mathcal{C}$ can in practice have missing and uncertain data encoded with specific symbols but we omit these issues here. Also note that an underlying assumption of our developments is the so–called equal crossing over in which the sequence fragments exchanged in a recombination event are of equal length, and hence the fragments retain their location in the sequences.

In a (reduced) parse of $\mathcal{C}$, a recombinant C_i is said to have a *crossover point* at j if the parse of C_i is $C_i = f_1\cdots f_h f_{h+1}\cdots f_{p_i}$ and $|f_1\cdots f_h| = j-1$. Hence C_i has $p_i - 1$ crossover points and $\mathcal{C}$ has $\sum p_i - m$ such points. Other interesting parameters of the parse are the *average fragment length* $\lambda_{ave} = mn/\sum_i p_i$ and the *minimum fragment length* $\lambda_{min} = \min_{ih}\{|f_{ih}|\}$.

Example 1. The recombinant set

$$
\begin{array}{c}
0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,1\,1\\
1\,1\,1\,1\,1\,1\,1\,0\,0\,1\,1\,0\\
0\,0\,1\,0\,1\,1\,1\,1\,0\,1\,1\,0\\
1\,1\,1\,1\,0\,0\,1\,0\,0\,0\,1\,1
\end{array}
$$

has a founder set

$$
\begin{array}{c}
0\,0\,1\,0\,1\,1\,1\,0\,0\,0\,1\,1\\
1\,1\,1\,1\,0\,0\,0\,1\,0\,1\,1\,0
\end{array}
$$

giving for example a parse

$$
\begin{array}{c}
\underline{0\,0\,1\,0}\,0\,0\,0\,1\,\underline{0\,0\,1\,1} \\
1\,1\,1\,1\,\underline{1\,1\,1\,0}\,0\,1\,1\,0 \\
\underline{0\,0\,1\,0\,1\,1\,1}\,1\,0\,1\,1\,0 \\
1\,1\,1\,1\,0\,0\,\underline{1\,0\,0\,0\,1\,1}
\end{array}
$$

where the fragments from the first founder are underlined. This parse has average fragment length $\lambda_{ave} = 4.8$ and minimum fragments length $\lambda_{min} = 4$. Note that the second crossover point of the first and the second sequence which is now at location 9 could be at 10 as well. The resulting parse would still have $\lambda_{ave} = 4.8$ but $\lambda_{min} = 3$. On the other hand, if we want to get a parse with $\lambda_{min} > 4$ a founder set of at least size 4 is needed. Then even the recombinants themselves would do as founders in this case, and the analysis does not uncover any interesting structure.

Moreover,

$$
\begin{array}{c}
1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0
\end{array}
$$

is also a founder set. The resulting parse (there is in fact only one parse in this case) has $\lambda_{ave} = 48/22 \approx 2.2$ and $\lambda_{min} = 1$. □

A founder set $\mathcal{F}$ 'explains' the recombinants $\mathcal{C}$ via a parse. Our problem is to reconstruct from $\mathcal{C}$ an $\mathcal{F}$ that is small in size but also gives a 'good' parse for $\mathcal{C}$. As parses with large λ_{ave} and λ_{min} are considered good (they capture more of the 'linkage' in the recombinants), we want simultaneously to minimize $|\mathcal{F}|$ and to maximize λ_{ave} and λ_{min}. As illustrated by Example 1, these are contradictory goals.

This trade–off of optimization goals leads to the following two versions of our reconstruction problem.

- The *minimum founder set problem* is, given a fragment length bound L, to construct a smallest possible founder set $\mathcal{F}$ such that $\mathcal{C}$ has in terms of $\mathcal{F}$ a parse for which $\lambda_{ave} \geq L$ or $\lambda_{min} \geq L$.
- The *maximum fragment length problem* is, given a bound M for the number of founders, to find largest λ_{ave} and/or λ_{min} that are possible in a parse of $\mathcal{C}$ in terms of a founder set $\mathcal{F}$ of size at most M.

Note that maximizing λ_{ave} is equivalent to minimizing the number of crossover points in the parse.

It is illuminating to formulate the founder reconstruction as a coloring problem. Let assume that each of the k founders has a unique *color* from $\{1, \ldots, k\}$. A parse gives these colors to the symbols of the recombinants:

the symbols in a fragment will get the color of the founder which the fragment belongs to in the parse. This coloring of $\mathcal{C}$ has the property that whenever two symbols c_{ij} and $c_{i'j}$ taken from the same location of different recombinants are different, then their colors must also differ. This is because the parse would map two symbols with the same color on the same symbol of the founder. Colorings with this property are called *consistent.* A coloring has a crossover whenever two adjacent symbols of the same recombinant have a different color. On the other hand, if a consistent coloring of $\mathcal{C}$ is given, it immediately gives a founder set in terms of which the coloring is a parse. A founder sequence with color h is built using the symbols of the recombinants with the color h. The jth location of the founder gets the h colored symbol from the column j of $\mathcal{C}$ if the column has this color, and otherwise it gets '$-$'. The fragments implied by a coloring consist of blocks of equally colored adjacent symbols of a recombinant.

The reconstruction problem is thus equivalent to the problem of finding consistent colorings of the recombinants. The coloring should have some desired property such as a small number of colors or crossover points.

3 Segmentation based algorithms

3.1 Finding the segments

We start with observing that the minimum founder set problem becomes easy if $L = 1$, i.e., if fragments of any length are allowed. Let namely μ be the maximum number of different elements of Σ that can occur in the same position of the sequences in $\mathcal{C}$. That is,

$$\mu = \max_j |\{c_{ij} \mid 1 \leq i \leq m\}|.$$

It follows that an $\mathcal{F}$ for $\mathcal{C}$ must be at least of size μ because $\mathcal{F}$ must offer μ different symbols for a certain location. However, the size μ also suffices if $L = 1$: take μ colors and consistenly color with these colors the symbols on each column of $\mathcal{C}$, and then build $\mathcal{F}$ from this coloring as explained above. The corresponding parse can have one–symbol long fragments. So we have:

Theorem 1. *If $L = 1$, the minimum founder set problem has optimal solution of size $\mu \leq |\Sigma|$.* □

The construction of Theorem 1 can be generalized for blocks of adjacent symbols of recombinants. A *segment* $\mathcal{C}[j, k]$ of $\mathcal{C}$ is a set consisting of the subsequences $c_{ij} \cdots c_{ik}$ of recombinants from location j to location k. The segment has *length* $k - j + 1$; its size, i.e., the number of different sequences in the segment, is denoted $|\mathcal{C}[j,k]|$. A *segmentation* of $\mathcal{C}$ consists of any collection of disjoint segments that cover the whole $\mathcal{C}$.

The current interest in the so–called haplotype blocks aims at finding from dense SNP haplotype data segments and segmentations whose internal variation (the size of the segments) is smaller than expected. Once a scoring function to evaluate each potential segment is fixed and the score is based only on the internal properties of the segment, one can find by dynamic programming a globally optimal segmentation. Recently, Zhang et al [10] gave such an algorithm to find an optimal segmentation that has a minimum total number of SNPs to uniquely distinguish at least α percent of haplotypes within each segment. This method also improved the results of [9] which were obtained using a greedy algorithm.

Our goal is to find a segmentation that would give a parse with $\lambda_{min} \geq L$ for some fixed L. To this end, we construct a segmentation such that each segment has length $\geq L$, and the maximum size of the segments is minimized. Let S_L denote the set of all segmentations of $\mathcal{C}$ such that each segment is of length $\geq L$. We want to compute

$$M(n) = \min_{s \in S_L} \max\{|\mathcal{C}[j,k]| \mid \mathcal{C}[j,k] \in s\}.$$

Then $M(n)$ can be obtained by evaluating $M(L), M(L+1), \ldots, M(n)$ from

$$\begin{cases} M(L) = |\mathcal{C}[1,L]| \\ M(j) \ = \min_{h \leq j-L} \max\{M(h), |\mathcal{C}[h+1,j]|\}, j = L+1, \ldots, n \end{cases} \tag{1}$$

The resulting dynamic programming algorithm needs to evaluate $|\mathcal{C}[j',j]|$ for $j' = j - L, j - L - 1, \ldots$. This can be dome by reading all sequences of $\mathcal{C}$, starting from location j and proceeding to the left and constructing a trie, level by level. The number of leaves at level that corresponds to location j' gives $|\mathcal{C}[j',j]|$. As constructing a new level takes time $O(m)$, we certainly get all necessary size values for a particular $M(j)$ in time $O(nm)$. Repeating for different j gives time bound $O(n^2 m)$.

Once we have $M(n)$, we can construct from the corresponding segmentation (this can be found by the usual trace–back after $M(n)$ has been evaluated) a founder set $\mathcal{F}$ of size $M(n)$. The segmentation gives a parse of $\mathcal{C}$ in terms of $\mathcal{F}$ consisting of fragments of length $\geq L$, hence $\lambda_{min} \geq L$.

Recursions similar to (1) can be used for the other variants of the segmentation problem. For example, to find segmentation such that $\lambda_{ave} \geq L$

and the size of the largest segment (i.e., the size of the founder set) is minimized, first observe that then the number of segments must be $\leq n/L$. Denopting by S all segmentations with at most n/L segments, we want to evaluate

$$M(n, n/L) = \min_{s \in S} \max\{|\mathcal{C}[j,k]| \mid \mathcal{C}[j,k] \in s\}.$$

This leads to recursion

$$\begin{cases} M(j,1) = |\mathcal{C}[1,j]| \\ M(j,r) = \min_{h<j} \max\{M(h,r-1), |\mathcal{C}[h+1,j]|\} \end{cases} \tag{2}$$

where $j = 1, \ldots, n$ and $r = 2, \ldots, n/L$. Evaluation by dynamic programming is possible in time $O(n^3 m)$.

A still simpler greedy algorithm finds a segmentation with smallest number of segments such that each segment has size $\leq M$ where M is a given bound. One only needs first to find the largest k such that $|\mathcal{C}[1,k]| \leq M$, then the largest k' such that $|\mathcal{C}[k+1,k']| \leq M$, and so on until the whole $\mathcal{C}$ has been segmented. The minimality of this segmentation is easy to see. The running time becomes $O(nm)$.

This subsection will be concluded with the following example that shows that founder sets constructed from a segmentation with the above simple method are not necessarily optimal for the given constraint.

Example 2. Let $L = 2$ and consider recombinants

```
1 1 1 0 0 0 0 0 0 0
1 1 1 1 0 0 0 0 0 0
1 1 1 1 1 0 0 0 0 0
0 0 0 1 1 1 0 0 0 0
0 0 0 0 1 1 1 0 0 0
0 0 0 0 0 1 1 1 1 1
0 0 0 0 0 0 1 1 1 1
0 0 0 0 0 0 0 1 1 1
```

From (1) we get an optimal segmentation with segment lengths 4, 2, and 4

```
1 1 1 0|0 0|0 0 0 0
1 1 1 1|0 0|0 0 0 0
1 1 1 1|1 0|0 0 0 0
0 0 0 1|1 1|0 0 0 0
0 0 0 0|1 1|1 0 0 0        (3)
0 0 0 0|0 1|1 1 1 1
0 0 0 0|0 0|1 1 1 1
0 0 0 0|0 0|0 1 1 1
```

The middle segment has the largest size 4. Hence the corresponding founder set is of size 4, and using the segment boundaries as crossover points we get $\lambda_{ave} = 80/24 \approx 3.3$ and $\lambda_{min} = 2$.

On the other hand, sequences

```
1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
```

are also a founder set, giving an obvious parse with 18 fragments, the shortest of them being of length 3. This founder set of size 2 thus has $\lambda_{ave} = 80/18 \approx 4.5$ and $\lambda_{min} = 3$. □

3.2 Colorings with minimum crossovers

The parse suggested by a segmentation can be improved using the idea that we should color the fragments of adjacent segments such that the colors match as well as possible. If the color is the same across a segment boundary on some recombinant, then there is actually no crossover and the two fragments can be joined to make a longer one.

Removing the maximum possible number of potential crossover points from the segmentation (3) of Example 2 will give

```
1 1 1 0|0 0 0 0 0 0
1 1 1 1|0 0 0 0 0 0
1 1 1 1 1 0|0 0 0 0
0 0 0 1 1 1|0 0 0 0
0 0 0 0|1 1 1 0 0 0          (4)
0 0 0 0|0 1 1 1 1 1
0 0 0 0 0 0|1 1 1 1
0 0 0 0 0 0|0 1 1 1
```

From this we get founder sequences

```
1 1 1 0 0 1 1 1 1 1
1 1 1 1 1 0 0 1 1 1
0 0 0 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0
```

The parse indicated by the fragmentation (4) has $\lambda_{ave} = 80/16 = 5$ and $\lambda_{min} = 4$.

We now describe the illustrated technique to eliminate a number of crossover points from a segmentation s. Let P_r denote the equivalence partition of the recombinants that is induced by the equality of the fragments in the rth segment. Hence i and i' belong to the same class of P_r if and only if C_i and $C_{i'}$ are equal in the rth segment. We assume that P_r has always M classes where M is the size of the largest segment (add empty classes to P_r if needed).

An M–coloring (a coloring with M colors) of $\mathcal{C}$ is called *consistent with segmentation* s if all crossovers of the coloring are on the segment boundaries of s and *strictly consistent* with s if it additionally is consistent with the partition P_r of each segment r. We restrict here on strictly consistent M–colorings and minimize the crossovers in this class.

For the minimization, the color of each P_r should be selected properly. Assume that we have fixed the colors of P_r and must next color P_{r+1}. If we give to a class $Y \in P_{r+1}$ the color of $X \in P_r$, we write $\phi(X) = Y$. This way each consistent coloring of P_{r+1} induces a one–to–one *color propagation mapping* ϕ from P_r to P_{r+1}. The best among them can be found with bipartite matching techniques as follows.

If $\phi(X) = Y$ then recombinants $i \in X \cap Y$ do not have a crossover at the boundary between segments r and $r+1$ while all $i \in X - Y$ must have a crossover. Hence selecting $\phi(X) = Y$ eliminates $w(X, Y) = |X \cap Y|$ crossovers.

To find the ϕ that eliminates the largest number of crossovers, denoted $W(P_r, P_{r+1})$, we form a complete weighted bipartite graph with node sets P_r and P_{r+1} and with weights $w(X, Y)$ for each $X \in P_r$, $Y \in Y_{r+1}$. The coloring is propagated optimally by the one–to–one mapping from P_r to P_{r+1} that corresponds to the maximum weight matching in this graph. The matching can be found by standard 'Hungarian method' in time $O(M^3)$; [8]. The weight of this matching equals $W(P_r, P_{r+1})$. This many crossovers are eliminated and hence $U(P_r, P_{r+1}) = m - W(P_r, P_{r+1})$ crossovers are left.

Note that this minimization works for *any* partitions. Hence we summarize it in a general form:

Theorem 2. *A color propagation mapping between two partitions P and P' of the recombinants that gives smallest possible number $U(P, P')$ of crossovers can be constructed in time $O(M^3)$ where M is the size of the partitions.* □

The minimization is repeated on each segment boundary to get a global coloring. The running time (omitting the segmentation time) becomes $O(n(m + M^3))$.

If the size of each segment is M and hence each P_r has M non–empty classes then this coloring has smallest possible number of crossovers among M–color parses whose crossovers are on the predetermined segment boundaries. This is because all such M–colorings must be strictly consistent with the segmentation and the color propagation can be done optimally by the above technique.

The case $M = 2$ is of special interest. We want to construct two founders and minimize the crossovers. Consider segmentation whose every segment has length 1, i.e., every column of $\mathcal{C}$ is its own segment. The size of each such segment must be at most 2, otherwise we cannot succeed. Remove all (uninformative) segments of size 1. All 2–colorings of the remaining segments must be strictly consistent, hence the color propagation can be done optimally with respect to the segmentation. But this segmentation does not exclude any crossover points, so we have the global optimum. Finally let the uninformative segments to inherit the coloring of their closest informative neighbor. This does not increase the number of crossovers.

We have obtained:

Theorem 3. *The maximum average fragment length problem can be solved in time $O(mn)$ for founder sets of size 2.* □

4 General solution

The segmenatation approach of the previous section leads to polynomial time algorithms but the results are suboptimal as we resorted to crossover points given by segmentations and to strictly consistent colorings. For completeness, we give here dynamic programming algorithms for the general case.

Given bound M for the size of $\mathcal{F}$, a parse and the corresponding $\mathcal{F}$ with largest possible λ_{ave} can be constructed as follows. In effect we use again the segmentation in which every column is its own segment, hence no crossover points are excluded. For any column j, the partition P_j that gives the optimum is not known. Therefore we tabulate the minimum number of crossovers in the initial segment of $\mathcal{C}$ up to column j for each possible partition P_j. Then similar values for column $j+1$ can be evaluated using the algorithm of Theorem 2.

More formally, let $\mathcal{P}_j$ denote the set of all consistent M–partitions of the rows of $\mathcal{C}$ at column j. Such partitions have M classes, some of them possibly empty, and they are subpartitions of the partition induced by the equality of the symbols on column j. For any two M–partitions P and P'

of successive columns we get by the method of Theorem 2 the smallest number $U(P,P')$ of crossovers between the columns when partitions P and P' are used.

Denote by $S(j,P)$ the minimum possible number of crossovers in M–colorings of $\mathcal{C}[1,j]$ that have partition $P \in \mathcal{P}_j$ at column j. Then we have $S(1,P) = 0$ for all $P \in \mathcal{P}_1$ and

$$S(j,P) = \min_{P' \in \mathcal{P}_{j-1}} \{S(j-1,P') + U(P',P)\} \tag{5}$$

for all $P \in \mathcal{P}_j$ and $j = 2,\ldots,n$. Then $S(n) = \min_{P \in \mathcal{P}_n} S(n,P)$ is the global minimum and the corresponding parse has $\lambda_{ave} = mn/(S(n,P) + m)$.

Evaluating (5) by dynamic programming gives an algorithm with running time proportional to nN where N denotes the size of the largest $\mathcal{P}_j$. Bound N explodes when m and M grow but the method can possibly be feasible for small parameter values.

The problem of minimizing the number of founders under the requirement $\lambda_{ave} \geq L$ can be solved by reducing to (5). Make a binary search over $1,\ldots,m$ to find the smallest M such that the corresponding λ_{ave}, as evaluated using (5), is at least L. This works correctly as λ_{ave} grows monotonically with M.

5 Conclusion

We described algorithms for the inversion of recombination. The approach was combinatorial ignoring several aspects necessary to consider in practical application of these methods.

An interesting candidate for further developments is to look at recombination history in this framework.

References

1. M. Daly, J. Rioux, S. Schaffner, T. Hudson, and E. Lander. High–resolution haplotype structure in the human genome. *Nature Genetics* 29: 229–232, 2001.
2. S. B. Gabriel, S. F. Schaffner, H. Ngyen, J. M. Moore et al. The structure of haplotype blocks in the human genome. *Science* 296: 2225–2229, 2002.
3. D. Gusfield. Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions. *RECOMB 2002*, 166–175.
4. R. R. Hudson and N. L. Kaplan. Statistical properties of the number of recombination events in the history of a sample of DNA sequences. *Genetics* 111: 147–164, 1985.

5. J. Kececioglu and D. Gusfield. Reconstructing a history of recombinations from a set of sequences. *Discrete Applied Mathematics* 88: 239–260, 1998.
6. H. Mannila, M. Koivisto, M. Perola, L. Peltonen, E. Ukkonen et al. A method to find and compare the strength of haplotype block boundaries. In preparation.
7. M. S. McPeek and A. Strahs. Assessment of linkage disequilibrium by the decay of haplotype sharing, with application to fixed–scale genetic mapping. *Am.J. Hum. Genet.* 65: 858–875, 1999.
8. C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Dover Publications 1998.
9. N. Patil, A. J. Berno, D. A. Hinds, W. A. Barrett et al. Blocks of limited haplotype diversity revealed by high–resolution scanning of human chromosome 21. *Science* 294: 1719–1723, 2001.
10. K. Zhang, M. Deng, T. Chen, M. S. Waterman and F. Sun. A dynamic programming algorithm for haplotype block partition. *PNAS* 99: 7335–7339, 2002.