Constructing the Minimum Founder Set from a Set of Recombinant Sequences

Jingli Wu
College of Computer Science and Information Technology, Guangxi Normal University, Guilin, 541004, China
Email: wjlhappy@mailbox.gxnu.edu.cn

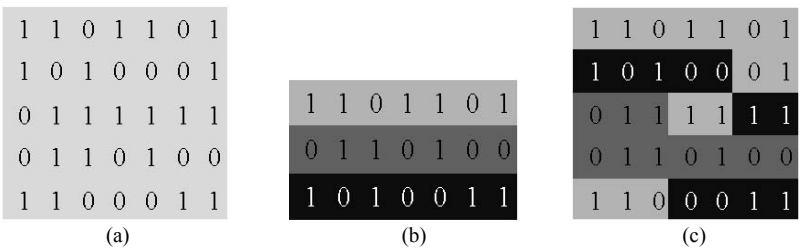
Zhaocan Wang
College of Computer Science and Information Technology, Guangxi Normal University, Guilin, 541004, China
Email: 648340393@qq.com

***Abstract*—Genetic recombination plays an important role in shaping the within-species genetic variations. It has been generally accepted that current-day population evolved from a small number of specific sequences called founders, and the genomic sequences (called recombinants) of individuals within the population are composed of segments from the founders due to recombination. A related computational problem is the so-called Minimum Founder Set problem: construct the minimum founder set from a set of given recombinant sequences so that the minimum fragment length is not less than the given fragment length bound. Ukkonen presented a segmentation based dynamic programming (SDYN) algorithm to solve it. However, the number of reconstructed founders obtained by the SDYN algorithm is restricted by the given fragment length bound in a certain situation. In this paper, a practical constructive heuristic algorithm HMFS is proposed for solving the problem. The HMFS algorithm partitions the sites of founders into three parts and reconstructs them respectively. Experimental results show that the HMFS algorithm can reconstruct a founder set very quickly, and the solutions obtained by the HMFS algorithm are close to or better than those obtained by the SDYN algorithm. In addition, the HMFS algorithm runs much faster than the SDYN algorithm, it is still efficient for solving large size problems.**

***Index Terms*—bioinformatics, minimum founder set problem, recombinant, founder, reconstruction, heuristic, algorithm**

I. INTRODUCTION

In the post-genome era, with the rapid development of DNA sequencing, abundant DNA sequences and haplotyped SNP sequences are available. The investigation on genealogy inference has become one of the main topics in genomics, which is a precondition for dissecting the relationship between genotypes and phenotypes, and discovering the genetic basis of complex diseases. Finding meiotic recombination events is a major difficulty in inferring genealogy, which has become a key problem in bioinformatics [1]–[5]. In most combinatorial models, given a set of sequences from individuals of a population (for example, humans), the population is assumed to descend from a small number of specific sequences called *founders*. Due to recombination, a genomic sample sequence, called a *recombinant*, is thus a concatenation of fragments from the founders. When all the recombinants are arranged in aligned rows, they look like a mosaic of fragments from the founder sequences [5], as shown in Fig. 1. Some biology study findings validate this model. For example, it is indicated in [6] that "The Ferroplasma type II genome seems to be a composite from three ancestral strains that have undergone homologous recombination to form a large population of mosaic genomes".



(a) (b) (c)

Figure 1. An example for illustrating the *Minimum Founder Set* problem on binary sequences. (a) shows a set of five recombinants; (b) shows a set of three founders; (c) shows a kind of mosaic structure for the recombinants in (a).

In Recent years, in order to infer the evolutionary history of the recombinants, researchers try to construct more recombinants from relatively fewer founders. However, the information about the founder sequences is unknown. The founder sequence reconstruction problem, i.e., reconstructing the founder set or recombinant mosaic

doi:10.4304/jcp.8.11.2942-2950

pattern from a group of recombinant samples, has received attention in bioinformatics.

Since there are a large number of possible mosaic patterns or founder sets for a set of sample sequences, biologically meaningful models need to be given for inferring breakpoints and founders. In 2002, Ukkonen [3] first proposed an optimization form of the problem based on the mosaic model and parsimony. Two optimization versions were considered: 1. the number of founders (called the Minimum Founder Set problem); 2. the number of recombinations (called the Maximum Fragment Length problem, or the Minimum Mosaic problem).

The recent research works are focusing mainly on minimizing the number of recombinations, i.e., the second optimization problem, which needs to give a bound for the number of founders before inferring the founders. Ukkonen [3] presented a dynamic programming based algorithm for this problem. The algorithm does not scale well when the number of founders or the number/length of the recombinants grows. Wu et al. [5] developed an exact algorithm RECBLOCK based on tree search. RECBLOCK can give a very good solution when the number of founders is small, and does not scale well either with the increase of founder number. In 2009, Roli et al. [7] presented a heuristic algorithm to solve the Maximum Fragment Length problem, and a tabu search algorithm to further optimize the solution obtained by the heuristic one. The tabu search algorithm can get rather good solutions efficiently and scales well while solving problems of large size, and it outperforms the heuristic version of RECBLOCK on large size instances. In 2010, Wu [8] created a new lower bound (called C bound) for the minimum mosaic problem by using integer linear programming, which is more accurate in practice than an existing bound CH [5]. In 2011, Blin et al. [9] put forward an accurate exponential algorithm. When the number of founder sequences increases, the performance of the algorithm still deteriorates.

However, the number of founders is unknown before hand in realistic application, it maybe more important to study the first optimization version, i.e., the Minimum Founder Set problem. To the best of our knowledge, only Ukkonen [3] proposed a segmentation based dynamic programming of time $O(n^3m)$ for solving the model (it is named as SDYN in this paper), where m is the number of recombinants, n is the number of SNP sites. The SDYN algorithm derives the number of reconstructed founders and the corresponding segmentation from a set of recombinants. We observe that the number of reconstructed founders obtained by the SDYN algorithm is restricted by the given fragment length bound, i.e., when the number of SNP sites n can be dividable by the given fragment length bound L, the number of reconstructed founders obtained by the SDYN algorithm must not be more than 2^L. In this paper, a constructed heuristic algorithm HMFS is proposed for solving the problem. The HMFS algorithm partitions the sites of founders into three parts and reconstructs them respectively. The solutions obtained by the HMFS algorithm are close to or better than those obtained by the SDYN algorithm, which is established by a number of experiments. Furthermore, the HMFS algorithm runs much faster than the SDYN algorithm, it is efficient even for solving large size problem.

The rest of this paper is organized as follows. In Section 2, The Minimum Founder Set problem is formalized. In Section 3, the HMFS algorithm is described. In Section 4, the comparisons between algorithms HMFS and SDYN are given through experimental results. Finally, some conclusions are drawn in Section 5.

II. The Minimum Founder Set Problem

Suppose that we have a set of m recombinants $C=\{C_1,\ldots,C_m\}$, where $C_i=c_{i1}c_{i2}\ldots c_{in}$ ($i=1,\ldots,m$) denotes a sequence of length n over a given alphabet $\sum$, i.e., $c_{ij}\in\sum$ ($j=1,\ldots,n$). In this work, a typical biological application is concerned where the recombinants are haplotyped SNP sequences and $\sum=\{0,1\}$. The two symbols 0 and 1 encode the two most common alleles of each SNP site. As shown in Fig. 1(a), each row denotes a recombinant and each column denotes a SNP site. Let $F=\{F_1,\ldots,F_k\}$ denote another set of sequences of length n over the alphabet $\sum$, where $F_t=f_{t1}f_{t2}\ldots f_{tn}$ ($f_{tj}\in\sum$, $t=1,\ldots,k$; $j=1,\ldots,n$). As shown in Fig. 1(b), each row denotes a founder and each column denotes a SNP site.

If the set of recombinants C can be reconstructed from F, F is called a *founder set* of C. This is the case if each $C_i\in C$ can be decomposed into $p_i(1\le p_i\le n)$ nonempty strings (fragments) Fd_{i1}, $Fd_{i2},\ldots, Fd_{ip_i}$, such that each Fd_{ij} ($j=1,2,\ldots, p_i$) occurs in at least one sequence of F exactly at the same position as in C_i. Take the second recombinant in Fig. 1(a) for example, it can be partitioned into two fragments 10100 and 01 by one breakpoint, such that the two fragments respectively occur in the third and the first founders at the same position as in the second recombinant. In this case, we say C has a *parse* in terms of F. Given all of the fragments Fd_{ij} ($i=1,2,\ldots,m, j=1,2,\ldots, p_i$), the minimum fragment length $\lambda_{\min}$ is defined as Formula (1),

$$\lambda_{\min}=\min(|Fd_{ij}|), i=1,...,m, j=1,...,p_i, \quad (1)$$

where $|Fd_{ij}|$ denotes the length of string Fd_{ij}.

Based on the above concepts, an important optimization model was proposed by Ukkonen [3] for the sequence reconstruction problem, as follows:

The Minimum Founder Set problem (*MFS*): Given a set of recombinants C and a fragment length bound $L(1\le L\le n)$, to construct a smallest possible founder set F such that C has in terms of F a parse for which the minimum fragment length $\lambda_{\min}\ge L$.

III. ALGORITHM HMFS

In this section, a constructive heuristic algorithm HMFS is described. In the following some notations and definitions used in HMFS are given first, and the HMFS algorithm is described afterwards.

A. Notations and Definitions

Define an $m\times n$ recombinant matrix. For convenience of description, the recombinant matrix is still denoted by C, where each row denotes a recombinant, and each column denotes a SNP site. Similarly, define a $k\times n$ founder matrix F, where each row denotes a founder, and each column also denotes a SNP site. The HMFS algorithm constructs founder set F from the first column to the last one. The columns currently being solved are called *current column*, which is denoted as (f_col,…, $f\_col+l-1$), where f_col (resp. $f\_col+l-1$) indicates the first (resp. the last) column of *current column*, and l denotes the number of columns of *current column*. For each row $C[i,-](i=1,\ldots,m)$, let $cb(i)$ store the position of the last breakpoint, $cf(i)$ store the index of the founder that represents row i after the last breakpoint $cb(i)$, and $cl(i)$ store the fragment length between the last breakpoint $cb(i)$ and the last column of *current column* $f\_col+l-1$. Take the fifth recombinant in Fig. 1(a) for example, assume that *current column* is the 6th column, i.e., $f\_col=6$, $l=1$, then $cb(5)=3$, $cf(5)=3$ (3 represents the third founder), and $cl(5)=3$. For each row $F[t, -]$ ($t=1,\ldots,k$), let set $CS(t)$ include the recombinants whose $cf(\cdot)$ values are equal to t, and $|CS(t)|$ denote the number of recombinants in set $CS(t)$.

Assume that $S=(s_{i,j})_{d\times n}$ is a set of d sequences of length n over $\sum$, and r is a l-length ($l=1, L$) binary string. Define $n_{l,r}(S, j)$ ($j=L+1,\ldots,n-L+1$) as the number of rows i' in S with $s_{i',j}\ldots s_{i',j+l-1}=r$. Let $p_{l,r}(S, j)$ ($j=L+1,\ldots,n-L+1$) be the proportion of r strings in the non null strings of columns ($j,\ldots, j+l-1$) in matrix S, as shown in Formula (2):

$$p_{l,r}(S,j)=\begin{cases}\dfrac{n_{l,r}(S,j)}{\sum_{r=B_0^l}^{B_{2^l-1}^l} n_{l,r}(S,j)}, & if \sum_{r=B_0^l}^{B_{2^l-1}^l} n_{l,r}(S,j)\neq 0,\\ 0, & \text{otherwise,}\end{cases} \tag{2}$$

where B_x^l denotes the l-length binary string of the decimal integer x. Given matrixes C and F, $w_{l,r}(j)$ ($j=L+1,\ldots,n-L+1$) is defined as the ratio between $p_{l,r}(F, j)$ and $p_{l,r}(C, j)$:

$$w_{l,r}(j)=\frac{p_{l,r}(F,j)}{p_{l,r}(C,j)},\quad j=L+1,\ldots,n-L+1. \tag{3}$$

Take the recombinant and founder matrices in Fig. 1(a) and (b) for example (here we also call them as C and F), assume that $L=2$ and $r=11$, then $n_{2,11}(C,6)=2$, $n_{2,11}(F,6)=1$, $p_{2,11}(C,6)=0.4$, $p_{2,11}(F,6)=0.3$, $w_{2,11}(6)=0.75$.

B. Heuristic Algorithm HMFS

HMFS, a constructive heuristic algorithm, is presented here. The input is an $m\times n$ recombinant matrix C and a fragment length bound $L(1\le L\le n)$. The output is a founder matrix F. The HMFS algorithm is based on the intuition that there can not be any breakpoints among the first (resp. the last) L columns under the constraint of $\lambda_{\min}\ge L$. It partitions the n sites of founders into three parts and reconstructs them respectively. In the following, some key steps in designing the HMFS algorithm are described.

1) Creating the Initial Founder Matrix: The HMFS algorithm starts with creating the initial founder matrix F, which is depicted as follows.

Let $k=1$, and set $F[k, j]=C[1, j]$ ($j=1,\ldots,L$). For every row $C[i,-](i=2,\ldots,m)$ of matrix C, if there does not exist such an $F[t,-](t=1,\ldots,k)$ that $C[i, j]=F[t,j](j=1,\ldots,L)$, insert a new row $F[k+1,-]$ into matrix F and set $F[k+1,j]=C[i, j](j=1,\ldots,L)$, $k=k+1$. After creating the initial founder matrix F, the first L columns of row $F[t,-](t=1,\ldots,k)$ are assigned values, and the variables $cb(i)$, $cf(i)$ and $cl(i)$ ($i=1,\ldots,m$) are initialized. In particular, $cb(i)=0$, $cl(i)=L$ ($i=1,\ldots,m$).

2) Computing the Middle Columns: The middle $n-2L$ columns of matrix F, i.e., columns $F[-,L+1],\ldots,F[-,n-L]$, are solved one by one. Column $F[-, j]$ ($j=L+1,\ldots,n-L$) is constructed as follows.

Firstly, the rows of matrix F are sorted according to their $|CS(\cdot)|$ values. A row t_1 ($t_1=1,\ldots,k$) whose $|CS(t_1)|$ value is odd is sorted before the row t_2 ($t_2=1,\ldots,k$) whose $|CS(t_2)|$ value is even, for $F[t_1, j]$ with odd $|CS(t_1)|$ value is easier to be ascertained than $F[t_2, j]$ with an even $|CS(t_2)|$. IF $|CS(t_1)|$ has the same parity as $|CS(t_2)|$ (i.e., are both even or both odd), the two rows t_1 and t_2 are sorted by their $|CS(\cdot)|$ values in ascending order. Let $r_1,\ldots,r_k$ denote the row order so obtained.

Secondly, the rows of matrix F are processed in the sorted order, i.e., $F[r_t,j](t=1,\ldots,k)$ is computed using the following two steps:

(a) If $n_{1,0}(CS(r_t),j)$ is greater (resp. less) than $n_{1,1}(CS(r_t),j)$, $F[r_t,j]$ is set to 0 (resp. 1), otherwise go to Step (b).

(b) If $w_{10}(j)$ is less (resp. greater) than $w_{11}(j)$, $F[r_t,j]$ is set to 0 (resp. 1), otherwise it is set to $C[h,j]$, here recombinant h has the maximum $cl(\cdot)$ among the recombinants of set $CS(r_t)$ (i.e., $h,g\in CS(r_t)$, $cl(h)\ge cl(g)$, $h\ne g$).

After column $F[-,j](j=L+1,\ldots,n-L)$ is constructed, some adjustments need to be made to let the proportion of 0 (resp. 1) entries of column j in matrix F approximate what it is in matrix C. That is to say if $p_{1,0}(F, j)$ is less than $p_{1,0}(C, j)$ (resp. $p_{1,1}(F, j)$ is less than $p_{1,1}(C, j)$), we should try to find such a row $F[t,-](t=1,\ldots, k)$ of F with $F[t, j]=1$ (resp. 0) and a row $C[i,-](i=1,\ldots,m)$ of C with $C[i, j]=0$ (resp. 1) that $cf(i)=t$, then set $F[t, j]=0$ (resp. 1) to improve the proportion of 0 (resp. 1) entries of column j in matrix F.

3) Computing the Last L Columns: The last L columns are constructed together.

Firstly, the founders of matrix F are partitioned into two sets G_1 and G_2. Given a founder $F[t,-](t=1,\ldots,k)$, if there exists at least one recombinant i of $CS(t)$ such that $cl(i)$ is less than L, $F[t,-]$ is classed into G_1, otherwise it

is classed into G_2.

Secondly, the founders of set $G_d(d=1,2)$ are sorted by their $|CS(\cdot)|$ values in ascending mode, let $r_1,..., r_{|G_d|}$ denote the founders ordering so obtained.

Finally, the two sets G_1 and G_2 are processed in turn. The founders $F[t,-]$ ($t=1,..., |G_d|$, $d=1, 2$) of set G_d are computed in the sorted order with the following three steps:

(a) If $n_{L,r}(CS(r_t),n-L+1)$ is greater than $n_{L,r}(CS(r_t),n-L+1)(r',r\in\{c_{in-L+1}...c_{in}|c_{i-}\in CS(r_t)\}$, $r'\neq r)$, $(F[r_t,n-L+1]...F[r_t, n])$ is set to r', otherwise go to Step (b).

(b) Let set S_r store the strings r' with $n_{L,r}(CS(r_t),n-L+1)\geq n_{L,r}(CS(r_t),n-L+1)(r',r\in\{c_{in-L+1}...c_{in}|c_{i-}\in CS(r_t)\}$, $r'\neq r))$. If $w_{Lr'}(n-L+1)$ is less than $w_{Lr}(n-L+1)(r', r\in S_r, r'\neq r)$, $(F[r_t,n-L+1]...F[r_t, n])$ is set to r', otherwise go to Step (c).

(c) Let set SS_r store the strings r' with $w_{Lr'}(n-L+1)\leq w_{Lr}(n-L+1)$ $(r', r\in S_r, r'\neq r)$. $(F[r_t,n-L+1]...F[r_t, n])$ is set to $(C[h, n-L+1]...C[h, n])$, where recombinant h has the maximum $cl(\cdot)$ among the recombinants of set $\{c_i|c_i\in CS(r_t), c_{in-L+1}...c_{in}\in SS_r\}$ (i.e., h, $g\in\{c_i|c_i\in CS(r_t), c_{in-L+1}...c_{in}\in SS_r\}$, $cl(h)\geq cl(g)$, $h\neq g$). In addition, if $F[r_t,-]\in G_1$, the HMFS algorithm still requires that $cl(h)$ and $cl(g)$ are all less than L.

After the last L columns of matrix F are constructed, some adjustments are also made to let the proportion of r ($r\in\{C[i, n-L+1]...C[i, n]|i=1,...,m\}$) strings of columns $(n-L+1,...,n)$ in matrix F approximate what it is in matrix C. For a given r value, if $p_{L,r}(F,n-L+1)$ is less than $p_{L,r}(C,n-L+1)$, we should try to find such a founder $F[t,-]$ of F with $(F[t,n-L+1]...F[t,n])\neq r$ and a recombinant $C[i,-]$ of C with $(C[i, n-L+1]...C[i, n])=r$ that $cf(i)=t$, then set $(F[t,n-L+1]...F[t,n])=r$.

4) Reassigning Founders: After computing *current column* $(f\_col,\ldots,f\_col+l-1)$ $(f\_col=L+1,\ldots,n-L+1)$, some recombinants $C[i,-](i=1,\ldots,m)$, which can not be represented anymore by their current representatives (founders) $F[cf(i),-]$ (i.e., $C[i, f\_col]\ldots C[i, f\_col+l-1] \neq F[cf(i), f\_col]\ldots F[cf(i), f\_col+l-1](l=1, L)$), need to be reassigned representatives as follows:

(a) Search for a founder $F[t,-](t=1,...,k)$ that can equally represent $C[i,-]$ starting from breakpoint $cb(i)$, i.e., $C[i, j]=F[t, j](j=cb(i),..., f\_col+l-1)$. If such a founder can be found, set $cf(i)=t$, $cl(i)=cl(i)+l$, otherwise go to Step (b).

(b) If $cl(i)\geq L$ and there exists a founder $F[t,-](t=1,...,k)$ that has the same value as $C[i,-]$ on *current column*, i.e., $C[i,j]=F[t,j](j=f\_col,..., f\_col+l-1)$, a new breakpoint is added to recombinant $C[i,-]$, and set $cb(i)= f\_col-1$, $cf(i)=t$, $cl(i)=l$, otherwise go to Step (c).

(c) Create a new founder $F[k+1,j]=C[i,j](j=1,..., f\_col+l-1)$, and set $cb(i)=0$, $cf(i)=k+1$, $cl(i)= f\_col+l-1$, $k=k+1$.

Based on the above mentioned steps, the HMFS algorithm for reconstructing the minimum founder set is summarized in Fig. 2.

Algorithm 1: HMFS algorithm

Input: an $m\times n$ recombinant matrix C, a fragment length bound L $(1\leq L\leq n)$
Output: a founder matrix F
1. $k=1$, $F[1, j]=C[1, j]$ $(j=1,...,L)$
2. **for** $i=2$ **to** m
3. **if** there is not an $F[t,-](t=1,...,k)$ to meet $C[i, j]=F[t,j](j=1,...,L)$ **then**
4. $F[k+1,j]=C[i, j](j=1,...,L)$, $k=k+1$
5. initialize $cb(i)$, $cf(i)$, $cl(i)$ $(i=1,...,m)$
6. **for** $j=L+1$ **to** $n-L$
7. sort the rows in F according to $|CS(\cdot)|$, and $r_1,...,r_k$ denote the sorted order
8. **for** $t=1$ **to** k
9. **if** $n_{1,0}(CS(r_t),j)> n_{1,1}(CS(r_t),j)$ **then** $F[r_t,j]=0$
10. **else if** $n_{1,0}(CS(r_t),j)< n_{1,1}(CS(r_t),j)$ **then** $F[r_t,j]=1$
11. **else if** $w_{10}(j)< w_{11}(j)$ **then** $F[r_t,j]=0$
12. **else if** $w_{10}(j)> w_{11}(j)$ **then** $F[r_t,j]=1$
13. **else** $F[r_t,j]=C[h,j]$ ($h,g\in CS(r_t)$, $cl(h)\geq cl(g)$, $h\neq g$)
14. adjust the proportion of 0 (resp. 1) entries of column j in matrix F
15. reassigning founders
15. partition the rows of F into two sets G_1 and G_2
16. sort the founders in set $G_d(d=1,2)$ according to $|CS(\cdot)|$, and $r_1,..., r_{|G_d|}$ denote the sorted order
17. **for** $d=1$ **to** 2
18. **for** $t=1$ **to** $|G_d|$
19. **if** $n_{L,r}(CS(r_t),n-L+1)> n_{L,r}(CS(r_t),n-L+1)$ $(r',r\in\{c_{in-L+1}...c_{in}|c_{i-}\in CS(r_t)\}, r'\neq r)$
20. **then** $(F[r_t,n-L+1]...F[r_t, n])=r'$
21. **else if** $w_{Lr'}(n-L+1) < w_{Lr}(n-L+1)(r', r\in S_r, r'\neq r)$ **then** $(F[r_t,n-L+1]...F[r_t, n])=r'$
22. **else** $F[r_t,n-L+1]...F[r_t,n])=(C[h,n-L+1]...C[h,n]$ $(h,g\in\{c_i|c_i\in CS(r_t), c_{in-L+1}...c_{in}\in SS_r\}$, $cl(h)\geq cl(g)$, $h\neq g)$
23. adjust r $(r\in\{C[i, n-L+1]...C[i, n]|i=1,...,m\})$ strings of columns $(n-L+1,...,n)$ in matrix F
24. reassigning founders

Figure 2. HMFS Algorithm.

Algorithm HMFS tries to delegate more recombinants with the existing founders during the process of reconstruction. Under the constraint of $\lambda_{\min} \geq L$, there can not be any breakpoints among the first L columns, therefore they are firstly created together in Step 1. Next, the middle ($n-2L$) columns are computed one by one. It is regarded that the entry value (0 or 1) with larger proportion in column j of sequence set $CS(r_t)$ should be assigned to $F[r_t,j]$, and the proportion of 0 (resp. 1) entries of column j in matrix F should be close to what it is in matrix C. Finally, the last L columns are also computed together under the constraint of $\lambda_{\min} \geq L$. The string r ($r \in \{ B_0^L,...,B_{2^L}^L \}$) with largest proportion in columns ($n-L+1,...,n$) of sequence set $CS(r_t)$ should be assigned to $F[r_t, n-L+1]...F[r_t, n]$, and the proportion of string r of columns ($n-L+1,...,n$) in matrix F should also be close to what it is in matrix C.

IV. Experimental Results

In this section, both real biological data and simulation data were used to compare the performance of algorithm HMFS with algorithm SDYN designed by Ukkonen [3]. The experiments were implemented on an Intel Pentium IV 2.0GHz with a 1GB of RAM. The operating system was Windows XP Professional and the compiler was Microsoft Visual C++ 6.0. In our experiments, we used the following measurements to evaluate the two algorithms:

(1) reconstruction rate RR, the accuracy of the reconstructed founder set.

Assume that $Fo=\{Fo_1,\ldots,Fo_{|Fo|}\}$ is the original founder set with $|Fo|$ n-length founders, and $Fr=\{Fr_1,\ldots,Fr_{|Fr|}\}$ is the reconstructed founder set with $|Fr|$ n-length founders. The proportion RR of nucleotides that are reconstructed correctly is defined as follows:

$$RR = (1 - \frac{\sum_{1 \leq i \leq |Fo|} \min\{r_{ij} \mid j = 1,...,|Fr|\}}{|Fo| \times n}) \times 100, \quad (4)$$

$$r_{ij} = \sum_{1 \leq c \leq n} (Fo_{ic} \oplus Fr_{jc}), i = 1,...,|Fo|, j = 1,...,|Fr|, \quad (5)$$

where $\oplus$ denotes the logical XOR operator.

(2) $|Fr|$, the number of founders in the reconstructed founder set Fr.

(3) $|Fr|-|Fo|$, the difference in the number of founders between sets Fr and Fo.

(4) $\#BP$, the sum of the number of breakpoints across all recombinant sequences, as defined in Formula (6):

$$\#BP = \sum_{1 \leq i \leq m} p_i - m \quad (6)$$

(5) f_{r_avg}, the average fragment length corresponding to the reconstructed founder set Fr, as defined in Formula (7):

$$f_{r\_avg} = \frac{\sum_{1 \leq i \leq m} \sum_{1 \leq j \leq p_i} |Fd_{ij}|}{m \times \sum_{1 \leq i \leq m} p_i} \quad (7)$$

(6) the running time (Time).

Since the original founder set Fo is unknown, measurements (2), (4), (5) and (6) are adopted in the experiments performed on real biological data, while measurements (1), (3) and (6) are used in the experiments performed on simulation data.

A. Experiments on Kreitman'ADH Data

The alcohol dehydrogenase (ADH) data of Kreitman [11] consist of 11 haplotypes over 43 polymorphic sties of the ADH locus of fruit fly, Drosophila melanogaster. Among these 11 haplotypes, there are three identical ones. After removing the duplicate sequences, there are 9 haplotypes left.

Table I shows the experimental results on ADH data, where ten sets of parameters were set in dealing with the minimum fragment length bound L. From this table, we can see that the number of reconstructed founders $|Fr|$ of the two algorithms are both increased with the augmentation of L, and $|Fr|$ obtained by the HMFS algorithm is close to that obtained by the SDYN algorithm. However, the HMFS algorithm can generate much fewer breakpoints than the SDYN algorithm, that is to say that the average fragment length f_{r_avg} obtained by the HMFS algorithm is much longer than that obtained by the SDYN algorithm. The running speed of the HMFS algorithm is much faster than the SDYN algorithm.

TABLE I.
Results for Kreitman's ADH data.

L	HMFS				SDYN			
	$\|Fr\|$	$\#BP$	f_{r_avg}	Time(s)	$\|Fr\|$	$\#BP$	f_{r_avg}	Time(s)
1	2	122	3.2	0.0002	2	270	1.4	0.0018
2	5	26	14.9	0.0001	4	90	4.3	0.0017
3	4	34	11.4	0.0001	4	72	5.4	0.0016
4	6	13	29.8	0.0001	5	54	7.2	0.0014
5	6	13	29.8	0.0001	5	45	8.6	0.0012
6	7	5	77.4	0.0001	5	45	8.6	0.0011
7	7	5	77.4	0.0001	6	27	14.3	0.0010
8	7	5	77.4	0.0001	6	27	14.3	0.0009
9	7	2	193.5	0.0001	7	18	21.5	0.0008
10	7	2	193.5	0.0001	7	18	21.5	0.0007

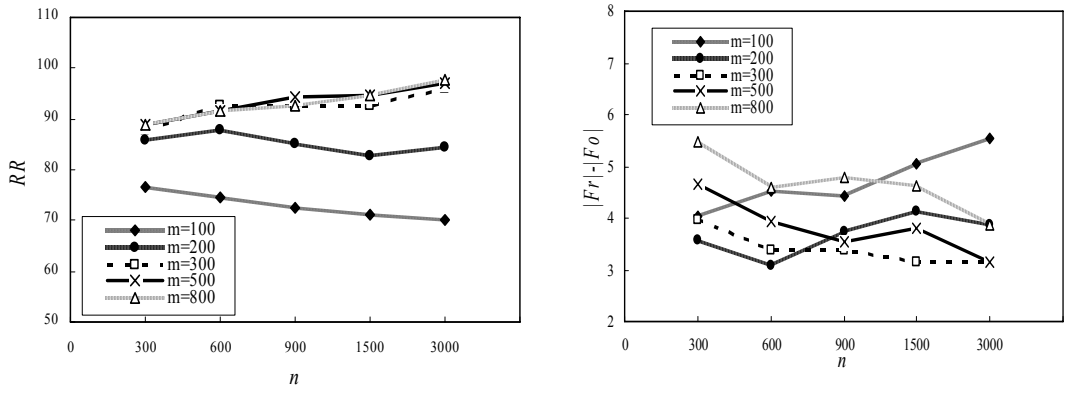
B. Experiments on Simulation Data

In this subsection, simulation data were used to evaluate algorithms HMFS and SDYN. Firstly, the original founder set *Fo* was generated randomly. Secondly, *m* recombinants were generated by randomly patching up *n_seg* random segments from the original founders. We evaluated the performance of the two algorithms by varying the number of the original founders |*Fo*|, the fragment length bound *L*, the number of recombinants *m*, the number of sites *n*, and the original average fragment length f_{o_avg}, which is equal to $\frac{n}{n\_seg}$. 100 data sets were generated for each parameter setting. The average over 100 runs at each parameter setting was calculated and presented.

Fig. 3 shows *RR* and |*Fr*|−|*Fo*| obtained by the HMFS algorithm with different *m* and *n*, where *L*=2, |*Fo*|=10 and *n_seg*=10. In Fig. 3(a), when *m*=300, 500 and 800, the HMFS algorithm gets relatively high reconstruction rates, and *RR* varies between a very narrow range for each *n* setting. In addition, as shown in Fig. 3(b), the HMFS algorithm also gets relatively small |*Fr*|−|*Fo*| when *m*=300. Therefore, *m* was set to 300 in the following experiments for the trade-off among *RR*, |*Fr*|−|*Fo*| and running time.

In Table II, nine sets of parameters were set in dealing with the number of original founders |*Fo*|, and *L*=2, *m*=300, *n*=600, *n_seg*=4.

With the increase of |*Fo*|, |*Fr*|−|*Fo*| obtained by the HMFS algorithm varies from 1.6 to 3.5, while |*Fr*|−|*Fo*| obtained by the SDYN algorithm varies from −6 to 2. It is apparent that when |*Fr*|−|*Fo*| is a negative, there must exist certain founders of set *Fo* not being reconstructed in set *Fr*, hence the HMFS algorithm is superior to the SDYN algorithm in reconstructing founder set.



(a) (b)

Figure 3. *RR* and |*Fr*|-|*Fo*| of different *m* and *n* with *L*=2, |*Fo*|=10, *n_seg*=10. (a) *RR*, (b)|*Fr*|-|*Fo*|.

TABLE II.
PERFORMANCE OF DIFFERENT |*Fo*| WITH *L*=2, *m*=300,*n*=600, *n_seg*=4

|*Fo*|	HMFS			SDYN	
	RR	|*Fr*|−|*Fo*|	Time(s)	|*Fr*|−|*Fo*|	Time(s)
2	99.87	1.6	0.0148	2	1701.8402
3	99.44	2.4	0.0171	1	1271.8567
4	98.98	2.9	0.0186	0	1028.1378
5	98.87	3.0	0.0196	-1	888.9338
6	98.44	3.4	0.0202	-2	769.9364
7	97.59	3.5	0.0205	-3	683.0628
8	96.96	2.8	0.0211	-4	635.9056
9	96.45	2.6	0.0226	-5	598.5243
10	95.05	2.6	0.0230	-6	550.7405

The running time of the HMFS algorithm is much shorter than that of the SDYN algorithm. When |*Fo*| is 10, the running time of algorithm SDYN is 550.7405 seconds, while the HMFS algorithm only uses 0.0230 seconds. Therefore, algorithm HMFS is still efficient for solving the instances with large *m* and *n*.

The SDYN algorithm obtains the number of reconstructed founders and the corresponding segmentation from a set of recombinants [3]. Since it is possible to construct many founder sets from a kind of segmentation, and there is not a concrete construction method described in reference [3], we only computed *RR* for the HMFS algorithm in our experiments. From Table II, we can see that algorithm HMFS gets relatively high reconstruction rates in every |*Fo*| setting, and *RR* decreases with the increase of |*Fo*|.

Given a fixed number of recombinants *m*, the increase of |*Fo*| decreases the recombinant coverage, which is the number of recombinants covering a SNP site of the original founders. As shown in Table II, when |*Fo*| changes from 2 to 10, the recombinant coverage decreases from 150 to 30. This is disadvantage for

TABLE III.
PERFORMANCE OF DIFFERENT L WITH $|Fo|$=6, m=300, n=600, n_seg=4

L	HMFS			SDYN	
	RR	$\|Fr\|-\|Fo\|$	Time(s)	$\|Fr\|-\|Fo\|$	Time(s)
2	98.44	3.4	0.0202	-2	709.9364
3	98.18	7.1	0.0205	2.8	750.4958
4	96.99	11.2	0.0204	9.3	727.6994
5	96.25	14.7	0.0208	10.5	742.7943
6	96.14	17.0	0.0202	18.3	776.4091
7	95.71	22.1	0.0206	26.4	732.3704
8	95.32	22.9	0.0207	33.6	733.5970
9	94.58	25.0	0.0209	36.5	718.7402
10	94.52	28.3	0.0218	37.2	745.2394

founder reconstruction.

In Table III, nine sets of parameters were set in dealing with the fragment length bound L, and $|Fo|$=6, m=300, n=600, n_seg=4.

From this table we can see that $|Fr|-|Fo|$ obtained by the two algorithms are all increased with the augment of L, and the increment speed of the HMFS algorithm is slower than that of the SDYN algorithm. When parameter L increases, the probability of a recombinant event occurring decreases, which is disadvantageous for constructing the minimum founder set, hence $|Fr|-|Fo|$ increases accordingly.

The running speed of the HMFS algorithm is more than 30000 times faster than that of the SDYN algorithm, the running time of the HMFS algorithm changes from 0.0202 to 0.0218 seconds, while the running time of the SDYN algorithm changes from 709.9364 to 745.2394 seconds. In every L setting, algorithm HMFS is able to get relatively high reconstruction rates, and RR decreases slightly with the increase of L.

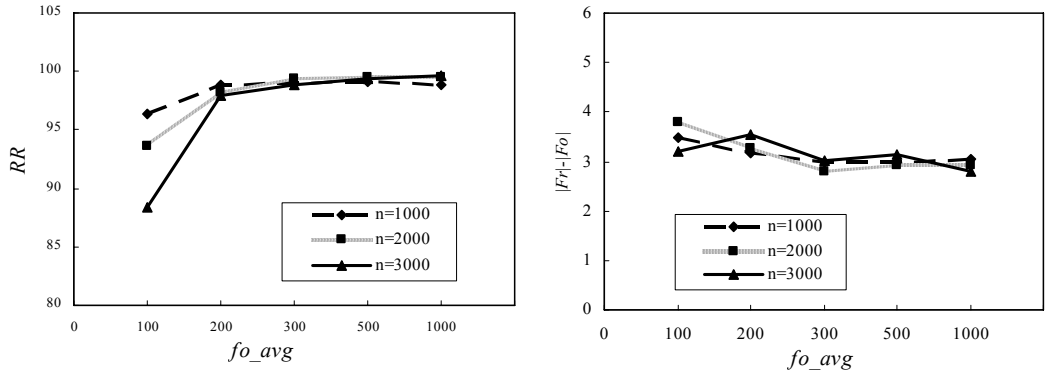
Table IV shows the experimental results for different f_{o_avg} settings. For each f_{o_avg}, three sets of parameters were set in dealing with n and n_seg, and L=2, $|Fo|$=6, m=300.

In our experiments, it has been noticed that when the number of SNP sites n can be dividable by the given fragment length bound L, the obtained number of reconstructed founders by the SDYN algorithm must not be more than 2^L. Since in the following experiments with L=2, $|Fr|-|Fo|$ obtained by the SDYN algorithm must be between −5 and −2, we only show the experimental results obtained by the HMFS algorithm. It can be seen from Table IV that with the increase of parameter f_{o_avg}, the performance of the HMFS algorithm improves gradually. Take n=3000 for example, when f_{o_avg} changes from 15 to 150, RR increases from 62.621 to 95.879, $|Fr|-|Fo|$ decreases from 29.4 to 2.2, and the running time decreases from 0.2749 to 0.0987 seconds. The HMFS algorithm is still efficient for reconstructing long founders.

TABLE IV.
PERFORMANCE OF DIFFERENT f_{o_avg} AND n WITH L=2, $|Fo|$=6, m=300

f_{o_avg}	n	n_seg	RR	$\|Fr\|-\|Fo\|$	Time(s)
15	300	20	79.32	5.1	0.0167
	1500	100	64.905	17.8	0.1334
	3000	200	62.621	29.4	0.2749
20	300	15	85.53	3.6	0.0144
	1500	75	67.227	11.6	0.1272
	3000	150	64.015	20.9	0.2129
25	300	12	90.98	2.9	0.0140
	1500	60	70.732	8.4	0.1141
	3000	120	65.456	14.9	0.1842
30	300	10	94.01	2.3	0.0139
	1500	50	75.130	6.5	0.1023
	3000	100	66.871	11.9	0.1692
50	300	6	95.74	1.9	0.0137
	1500	30	87.523	3.3	0.0956
	3000	60	76.625	5.9	0.1426
75	300	4	96.31	1.9	0.0136
	1500	20	89.648	2.8	0.0825
	3000	40	84.049	4.1	0.1235
100	300	3	96.92	1.9	0.0134
	1500	15	94.234	2.2	0.0666
	3000	30	88.414	3.2	0.1137
150	300	2	97.52	1.8	0.0133
	1500	10	97.945	2.1	0.0482
	3000	20	95.879	2.2	0.0987

It has been reported that one centimorgan, which is a unit of measure of recombination frequency, is equivalent to one million base pairs on average in human beings [10]. Since there is about one SNP per 1000 bases, the number of SNP sites covered by one centimorgan is about 1000. As shown in Fig. 4, further experiments were conducted to test the performance of the HMFS algorithm in dealing with parameters f_{o_avg} and n. When f_{o_avg} ranges between 200 and 1000, HMFS can get very high reconstruction rate, and the number of reconstructed founders $|Fr|$ is very close to that of the original founders $|Fo|$.



(a) (b)

Figure 4. *RR* and $|Fr|$-$|Fo|$ of different f_{o_avg} and n with L=2, $|Fo|$=6, m=300. (a) *RR*; (b)$|Fr|$-$|Fo|$.

V. CONCLUSIONS

During meiosis, genetic recombination generates a mosaic structure of the genome, and each resulting haplotype consists of fragments from different ancestral sequences. In this paper, we study the *Minimum Founder Set* problem, which is an important computational model for inferring breakpoints and founders.

A novel constructive heuristic algorithm HMFS is presented for finding the minimum founder set. The HMFS algorithm partitions the sites of founders into three parts and reconstructs them respectively. Comparing with the SDYN algorithm, algorithm HMFS can get close or better solution with much shorter running time, which was tested by a number of experiments. Furthermore, the HMFS algorithm has high efficiency and is intended to reconstruct long founders. In conclusion, algorithm HMFS is a practical method for solving the *Minimum Founder Set* problem.

ACKNOWLEDGMENT

The authors are grateful to Professor Yufeng Wu for his kindly providing the ADH sample data, and Professor Qi Zhang for bringing the minimum mosaic problem to our attention. This research was supported in part by Guangxi Natural Science Foundation under Grant No. 2011GXNSFB018068, No. 2011GXNSFB018070 and No. 2012GXNSFAA053219, the National Natural Science Foundation of China under Grant No. 61165009, and "Bagui Scholar" Project Special Funds.

REFERENCES

[1] J. Kececioglu and D. Gusfield, "Reconstructing a history of recombinations from a set of sequences," *Discrete Applied Mathematics*, vol. 88, no. 1-3, pp. 239–260, 1998.

[2] R. Schwartz, A. Clark, and S. Istrail, "Inferring piecewise ancestral history from haploid sequences," in *Proceedings of Computational Methods for SNPs and Haplotype Inference, LNCS 2983*, 2002, pp. 62–73.

[3] E. Ukkonen, "Finding founder sequences from a set of recombinants," in *Proceedings of the 2nd Workshop on Algorithms in BioInformatics, LNCS 2452*, 2002, pp. 277–286.

[4] P. Rastas and E. Ukkonen, "Haplotype inference via hierarchical genotype parsing," in *Proceedings of the 7th Workshop on Algorithms in BioInformatics, LNCS 4645*, 2007, pp. 85–97.

[5] Y. Wu and D. Gusfield, "Improved algorithms for inferring the minimum mosaic of a set of recombinants," in *Proceedings of Combinatorial Pattern Matching, LNCS 4580*, 2007, p. 150C161.

[6] G. Tyson, J. Chapman, P. Hugenholtz, E. Allen, R. Ram, P. Richardson, V. Solovyev, E. Rubin, D. Rokhsar, and J. Banfield, "Community structure and metabolism through reconstruction of microbial genomes from the environment," *Nature*, vol. 428, pp. 37–43, 2004.

[7] A. Roli and C. Blum, "Tabu search for the founder sequence reconstruction problem: a preliminary study," in *Proceedings of IWANN 2009, Part II, LNCS 5518*, 2009, pp. 1035–1042.

[8] Y. Wu, "Bounds on the Minimum Mosaic of Population Sequences under Recombination," in *Proceedings of Combinatorial Pattern Matching, LNCS 6129*, 2010, p. 152-163.

[9] G. Blin, R. Rizzi, F. Sikora, and S. Vialette, "Minimum mosaic inference of a set of recombinants," in

Proceedings of the 17th Computing: the Australasian Theory Symposium, 2011, pp. 23–30.
[10] M. Scott, P. Matsudaira, H. Lodish, J. Darnell, L. Zipursky, C. Kaiser, A. Berk, and M. Krieger, in *Molecular cell biology, 5th edition*, 2004.
[11] M. Kreitman, "Nucleotide polymorphism at the alcohol dehydrogenase locus of drosophila melanogaster," *Nature*, vol. 304, pp. 412–417, 1983.
[12] T.I.H. Consortium, "The international hapmap project," *Nature*, vol. 426, no.6968, pp. 789–796, 2003.

Jingli Wu, born in 1978, an associate professor of the College of Computer Science and Information Technology, Guangxi Normal University, Guilin, P.R. China. She received her Ph.D. degree in Computer Science from Central South University of P.R. China in 2008. Her current research interests include biocomputing, algorithm analysis and optimization.

Zhaocan Wang, born in 1987, he is currently working towards his M.S. degree in College of Computer Science and Information Technology, Guangxi Normal University. His research interests are in the areas of bio-computing and evolutionary algorithm