BIOINFORMATICS

Vol. 00 no. 00 2005
Pages 1–5

Supplementary Material for "Exact algorithms for haplotype assembly from whole-genome sequence data"

Zhi-Zhong Chen [1,*], Fei Deng [2] and Lusheng Wang [2*]

[1]Division of Information System Design, Tokyo Denki University, Hatoyama, Saitama 350-0394, Japan.
[2]Department of Computer Science, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong.

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

1 EXAMPLE

Table 1 shows an example matrix X and a solution of X. Reads r_1 through r_8 are gapless, r_9 is a multi-gapped read, r_{10} is a paired-end read, and the lengths of th reads are 2, 2, 3, 2, 2, 2, 2, 2, 8, and 7, respectively. Moreover, the MEC score of the solution (h, h') for X is 2. Furthermore, the monotone columns are the 1st, 6th, 7th, 9th, and 10th columns.

Table 1. An example read matrix X consisting of 10 reads spanning 12 positions, and a solution (h, h') for X.

reads	1	2	3	4	5	6	7	8	9	10	11	12
r_1	1	1	–	–	–	–	–	–	–	–	–	–
r_2	–	0	0	–	–	–	–	–	–	–	–	–
r_3	–	1	0	0	–	–	–	–	–	–	–	–
r_4	–	1	1	–	–	–	–	–	–	–	–	–
r_5	–	–	–	0	1	–	–	–	–	–	–	–
r_6	–	–	–	1	0	–	–	–	–	–	–	–
r_7	–	–	–	–	–	–	–	–	–	–	0	0
r_8	–	–	–	–	–	–	–	–	–	–	1	1
r_9	–	–	–	–	0	–	–	1	0	1	–	1
r_{10}	–	–	–	–	–	1	0	0	0	–	1	1
h	1	0	0	0	1	1	0	0	0	1	1	1
h'	1	1	1	1	0	1	0	1	0	1	0	1

In order for the matrix X in Table 1 to satisfy Assumptions A2 and A3 in the main text, we can flip 1's and 0's in the 2nd, 11th, and 12th columns and further delete the 1st, 6th, 7th, 9th, and 10th columns of X. The resulting matrix is shown in Table 2.

For the example matrix in Table 2, the 1st column contributes 1 to the MEC score of (h, h') and so does the 7th column, while the other columns contribute 0. Note that the MEC score of (h, h') is the total contribution of the columns of X.

The matrix X in Table 2 is a block itself and its splittable columns are the 1st, 3rd, and 4th columns. Moreover, the 1st row of X is singular. Furthermore, X has three reduced blocks.

*to whom correspondence should be addressed

Table 2. The resulting matrix obtained by modifying the matrix X in Table 1 so that it satisfies Assumptions A2 and A3 in the main text, where the three submatrices consisting of consecutive bold letters are the reduced blocks and are also unsplittable blocks.

reads	1	2	3	4	5	6	7
r_1	0	–	–	–	–	–	–
r_2	**1**	**0**	**–**	–	–	–	–
r_3	**0**	**0**	**0**	–	–	–	–
r_4	**0**	**1**	**–**	–	–	–	–
r_5	–	–	**0**	**1**	–	–	–
r_6	–	–	**1**	**0**	–	–	–
r_7	–	–	–	**–**	**–**	**1**	**1**
r_8	–	–	–	**–**	**–**	**0**	**0**
r_9	–	–	–	**0**	**1**	**–**	**0**
r_{10}	–	–	–	**–**	**0**	**0**	**0**
h	1	0	0	1	0	0	0
h'	0	1	1	0	1	1	0

For the matrix X in Table 2, $\eta_{1,0} = \eta_{7,0} = \bar{s}_{7,0} = 3$, $\eta_{2,0} = \eta_{3,0} = \eta_{4,0} = \eta_{6,0} = \bar{s}_{1,0} = \bar{s}_{2,0} = \bar{s}_{3,0} = \bar{s}_{4,0} = \bar{s}_{6,0} = 2$, and $\eta_{5,0} = \eta_{1,1} = \eta_{2,1} = \eta_{3,1} = \eta_{4,1} = \eta_{5,1} = \eta_{6,1} = \eta_{7,1} = \bar{s}_{5,0} = \bar{s}_{1,1} = \bar{s}_{2,1} = \bar{s}_{3,1} = \bar{s}_{4,1} = \bar{s}_{5,1} = \bar{s}_{6,1} = \bar{s}_{7,1} = 1$. So, Lemma 1 in the main text ensures that the first 6 columns of X are intrinsically heterozygous.

2 FALSE-HETEROZYGOUS SITES AND THE GENERAL CASE MODEL

First, let us clarify the terminologies. Recall that a diploid organism has two alleles at each position. A SNP site is *homozygous* if the two alleles at this site are identical; otherwise, it is *heterozygous*. Here, the terms "homozygous" and "heterozygous" are used when two copies of a chromosome are considered. When considering the input read matrix X, we say that a column is *monotone* if 0 or 1 does not appear in the column; otherwise, it is *non-monotone*.

Next, let us recall how to obtain the input matrix X. Given a set of reads sequenced from the two copies of a chromosome of an

© Oxford University Press 2005.

individual, we can obtain a matrix, say M, by aligning all the reads to the reference sequence.

Due to possible errors in reads, some monotone columns in the error-free case may become non-monotone in practice. Therefore, people use various SNP/variant calling methods (Li *et al.*, 2008; DePristo *et al.*, 2011; Li *et al.*, 2009) to remove all identified homozygous sites to obtain the resulting (filtered) matrix X. As the result, X only contains those (putative) heterozygous sites found by the SNP/variant calling methods and all the columns in X are non-monotone. This filtered matrix X will be used as the input matrix to the haplotype assembly problem. Most of the columns in the filtered input matrix X correspond to true-heterozygous SNP sites. However, it is still possible that a small number of columns in X should actually correspond to homozygous sites. Such a column in X is referred to as *false-heterozygous*.

Now, let us look at an example. Table 3 gives an example read matrix consisting of 6 reads and 8 SNP sites. The two real haplotypes h and h' are also given in the same table, where we can see that sites 1 and 5 are homozygous and the others are heterozygous. The first three reads are from haplotype h, while the remaining three are from haplotype h'. Since there is no error in the reads, we can see that columns 1 and 5 are monotone and can be ignored in the filtered input matrix X. However, when errors occur in reads, it is possible that all the eight columns in the matrix are non-monotone. Table 4 gives a new read matrix obtained from the one shown in Table 3 by inserting two errors (marked in red). All the eight columns are now non-monotone and would be included in the filtered input matrix X, where columns 1 and 5 should actually correspond to homozygous sites. In this case, if the all-heterozygous model is used, one can get an erroneous solution $(h, h') = (10001000, 01110111)$, just as shown in Table 5. As the result, the MEC score under the all-heterozygous model becomes 3, where the first entry in both reads r_1 and r_3 and the fifth entry in read r_6 should be corrected. In contrast, if we use the general-case model, we can still obtain the two real haplotypes (in Tables 3 and 4) as a solution to the same input as shown in Table 5, where the associated MEC score is 2. This is the reason why the general case can be more useful in practice.

Table 3. A read matrix containing no errors, and the two real haplotypes h and h'.

reads	1	2	3	4	5	6	7	8
r_1	0	0	0	0	1	0	0	0
r_2	0	0	–	–	1	0	0	0
r_3	0	–	0	0	–	–	–	–
r_4	0	1	1	1	–	–	–	–
r_5	0	–	–	1	1	1	1	1
r_6	0	1	–	–	1	1	1	1
h	0	0	0	0	1	0	0	0
h'	0	1	1	1	1	1	1	1

Table 4. A read matrix containing 2 errors (marked in red), and the two real haplotypes h and h'.

reads	1	2	3	4	5	6	7	8
r_1	0	0	0	0	1	0	0	0
r_2	1	0	–	–	1	0	0	0
r_3	0	–	0	0	–	–	–	–
r_4	0	1	1	1	–	–	–	–
r_5	0	–	–	1	0	1	1	1
r_6	0	1	–	–	1	1	1	1
h	0	0	0	0	1	0	0	0
h'	0	1	1	1	1	1	1	1

Table 5. A read matrix containing 2 errors (marked in red), and the two constructed haplotypes h and h' under the all-heterozygous model.

reads	1	2	3	4	5	6	7	8
r_1	0	0	0	0	1	0	0	0
r_2	1	0	–	–	1	0	0	0
r_3	0	–	0	0	–	–	–	–
r_4	0	1	1	1	–	–	–	–
r_5	0	–	–	1	0	1	1	1
r_6	0	1	–	–	1	1	1	1
h	1	0	0	0	1	0	0	0
h'	0	1	1	1	0	1	1	1

3 ILP FORMULATION FOR THE ALL-HETEROZYGOUS CASE

$$\begin{aligned}
\text{Minimize} \quad & \sum_{i=1}^{p} w_i \sum_{j \in J_{i,0}} c_j \left(1 - x_j - y_i + 2t_{i,j}\right) \\
& + \sum_{i=1}^{p} w_i \sum_{j \in J_{i,1}} c_j \left(y_i + x_j - 2t_{i,j}\right) \\
\text{Subject to} \quad & \forall_{1 \le i \le p} \ y_i \in \{0,1\} \\
& \forall_{1 \le j \le q} \ x_j \in \{0,1\} \\
& \forall_{1 \le i \le p} \forall_{1 \le j \le q} \ t_{i,j} \in \{0,1\} \\
& t_{i,j} \le y_i \\
& t_{i,j} \le x_j \\
& t_{i,j} \ge y_i + x_j - 1
\end{aligned}$$

4 ILP FORMULATION FOR THE GENERAL CASE

Suppose that we want to compute an optimal pair (h, h') of haplotypes for D. Let p (respectively, q) be the number of rows (respectively, columns) of D. For each integer j with $1 \le j \le q$, let c_j be the multiplicity of $D[j]$. Similarly, for each integer i with $1 \le i \le p$, let w_i be the multiplicity of the ith row of D, $J_{i,0}$ (respectively, $J_{i,1}$) be the set of integers $j \in \{1, 2, \ldots, q\}$ such that $D[j]$ is intrinsically heterozygous and the ith entry in $D[j]$ is a 0 (respectively, 1), and $\overline{J_{i,0}}$ (respectively, $\overline{J_{i,1}}$) be the set of integers

$j \in \{1, 2, \ldots, q\}$ such that $D[j]$ is not known to be intrinsically heterozygous and the ith entry in $D[j]$ is a 0 (respectively, 1).

As in the all-heterozygous case, we introduce a binary variable y_i for the ith row of D whose value is supposed to be 1 if and only if the read corresponding to r_i is aligned to h, where r_i is the ith row of D. Again as in the all-heterozygous case, for each intrinsically heterozygous column $D[j]$ of D, we introduce a binary variable x_j for $D[j]$ whose value is supposed to be 1 if and only if the jth bit of h is a 1 (and hence the jth bit of h' is a 0). However, for each column $D[j]$ of D that is not known to be intrinsically heterozygous, we need to introduce two binary variables x_j and z_j for $D[j]$ such that the value of x_j (respectively, z_j) is supposed to be 1 if and only if the jth bit of h (respectively, h') is a 1. Then, the problem of finding an optimal pair (h, h') of haplotypes for D becomes the following integer programming problem:

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{i=1}^{p} w_i \sum_{j \in J_{i,0}} c_j \left(x_j y_i + (1 - x_j)(1 - y_i)\right) \\
+ \quad & \sum_{i=1}^{p} w_i \sum_{j \in J_{i,1}} c_j \left((1 - x_j) y_i + x_j (1 - y_i)\right) \\
+ \quad & \sum_{i=1}^{p} w_i \cdot \sum_{j \in \overline{J_{i,0}}} c_j \left(x_j y_i + z_j (1 - y_i)\right) \\
+ \quad & \sum_{i=1}^{p} w_i \cdot \sum_{j \in \overline{J_{i,1}}} c_j \left((1 - x_j) y_i + (1 - z_j)(1 - y_i)\right) \\
\text{Subject to} \quad & \forall_{1 \le i \le p} \ y_i \in \{0, 1\} \\
& \forall_{1 \le j \le q} \ x_j \in \{0, 1\} \\
& \forall_{j \in \overline{J_{i,0}} \cup \overline{J_{i,1}}} \ z_j \in \{0, 1\}
\end{aligned}
$$

The above integer programming problem is not linear because it contains quadratic terms such as $y_i x_j$ and $y_i z_j$. So, for each pair (i, j) with $1 \le i \le p$ and $1 \le j \le q$, we introduce a binary variable $t_{i,j}$ (for replacing $y_i x_j$) and add the following three constraints to ensure that $t_{i,j} = y_i x_j$:

$$t_{i,j} \le y_i \qquad t_{i,j} \le x_j \qquad t_{i,j} \ge y_i + x_j - 1$$

Moreover, for each pair (i, j) with $1 \le i \le p$ and $j \in \overline{J_{i,0}} \cup \overline{J_{i,1}}$, we introduce a new binary variable $u_{i,j}$ (for replacing $y_i z_j$) and add the following three constraints to ensure that $u_{i,j} = y_i z_j$

$$u_{i,j} \le y_i \qquad u_{i,j} \le z_j \qquad u_{i,j} \ge y_i + z_j - 1$$

The resulting ILP formulation is as follows:

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{i=1}^{p} w_i \sum_{j \in J_{i,0}} c_j \left(1 - x_j - y_i + 2t_{i,j}\right) \\
+ \quad & \sum_{i=1}^{p} w_i \sum_{j \in J_{i,1}} c_j \left(y_i + x_j - 2t_{i,j}\right) \\
+ \quad & \sum_{i=1}^{p} w_i \cdot \sum_{j \in \overline{J_{i,0}}} c_j \left(z_j + t_{i,j} - u_{i,j}\right) \\
+ \quad & \sum_{i=1}^{p} w_i \cdot \sum_{j \in \overline{J_{i,1}}} c_j \left(1 - z_j - t_{i,j} + u_{i,j}\right) \\
\text{Subject to} \quad & \forall_{1 \le i \le p} \ y_i \in \{0, 1\} \\
& \forall_{1 \le j \le q} \ x_j \in \{0, 1\} \\
& \forall_{1 \le i \le p} \forall_{1 \le j \le q} \ t_{i,j} \in \{0, 1\} \\
& \qquad t_{i,j} \le y_i \\
& \qquad t_{i,j} \le x_j \\
& \qquad t_{i,j} \ge y_i + x_j - 1 \\
& \forall_{1 \le i \le p} \forall_{j \in \overline{J_{i,0}} \cup \overline{J_{i,1}}} \ u_{i,j} \le y_i \\
& \qquad u_{i,j} \le z_j \\
& \qquad u_{i,j} \ge y_i + z_j - 1
\end{aligned}
$$

By Assumption A3, we can further add the following constraints:

$$\forall_{j \in \overline{J_{i,0}} \cup \overline{J_{i,1}}} \ x_j + z_j \le 1 \tag{1}$$

By experiments, we have found that Constraints 1 may slow down CPLEX when solving the problem for easy blocks, but can speed up CPLEX when solving the problem for hard blocks.

5 LEMMA 1 AND ITS PROOF

Lemma 1: Suppose that $j \in \{1, 2, \ldots, n\}$ satisfies $\min\{\eta_{j,0}, \eta_{j,1}\} \ge \lfloor \frac{\bar{s}_{j,0} + \bar{s}_{j,1}}{2} \rfloor$. Then, $X[j]$ is intrinsically heterozygous.

Proof: Suppose that (h, h') is an optimal solution of X such that the jth bit of h is the same as that of h'. Then, since (h, h') is optimal, the contribution of $X[j]$ to the MEC score of (h, h') is $\min\{\eta_{j,0}, \eta_{j,1}\}$ and in turn is $\eta_{j,1}$ by Assumption A3 in the main text. Thus, we may assume that the jth bits of h and h' are both a 0, because otherwise we have $\eta_{j,1} = \eta_{j,0}$ and we may exchange 0 and 1 in the remainder of this proof.

Let $n_{h,0}$ (respectively, $n_{h,1}$) be the number of nonsingular rows r_i in X such that $d(r_i, h) \le d(r_i, h')$ and the jth entry in r_i is a 0 (respectively, 1). Similarly, let $n_{h',0}$ (respectively, $n_{h',1}$) be the number of nonsingular rows r_i in X such that $d(r_i, h') < d(r_i, h)$ and the jth entry in r_i is a 0 (respectively, 1). If $n_{h,0} + n_{h',1} \le n_{h,1} + n_{h',0}$, then we flip the jth bit of h and in turn the contribution of $X[j]$ to the MEC score of (h, h') becomes at most $n_{h,0} + n_{h',1}$ because singular rows contributes nothing to heterozygous columns. On the other hand, if $n_{h,0} + n_{h',1} > n_{h,1} + n_{h',0}$, then we flip the jth bit of h' and in turn the contribution of $X[j]$ to the MEC score of (h, h') becomes $n_{h,1} + n_{h',0}$. So, in either case, the contribution

of $X[j]$ to the MEC score of (h, h') becomes $\min\{n_{h,0} + n_{h',1}, n_{h,1} + n_{h',0}\}$. Note that $\min\{n_{h,0} + n_{h',1}, n_{h,1} + n_{h',0}\} \leq \frac{n_{h,0} + n_{h',1} + n_{h,1} + n_{h',0}}{2} = \frac{\bar{s}_{j,0} + \bar{s}_{j,1}}{2}$, and in turn $\min\{n_{h,0} + n_{h',1}, n_{h,1} + n_{h',0}\} \leq \lfloor \frac{\bar{s}_{j,0} + \bar{s}_{j,1}}{2} \rfloor$ because $\min\{n_{h,0} + n_{h',1}, n_{h,1} + n_{h',0}\}$ is integral. Now, since $\min\{\eta_{j,0}, \eta_{j,1}\} \geq \lfloor \frac{\bar{s}_{j,0} + \bar{s}_{j,1}}{2} \rfloor$, we know that in either case, the flipping does not increase the contribution of $X[j]$ to the MEC score of (h, h') and hence (h, h') remains to be optimal. This finishes the proof. □

Using Lemma 1, we can find the intrinsically heterozygous columns in X in time linear in the total length of reads in X.

6 SPECIFICS OF USING CPLEX

To solve the all-heterozygous (respectively, general) case of the problem for a chromosome, we first cut the chromosome into as many reduced-blocks as possible (cf. Section 2.2 (respectively, Section 2.3) of the main text), next use CPLEX to solve the reduced-blocks on the PC, and finally assemble the solutions of the blocks into a solution of the whole chromosome.

When we use CPLEX to find optimal solutions for hard reduced-blocks, we first use our heuristics to find heuristic solutions and then use them as the starting solutions to search for optimal solutions. Since CPLEX uses branch-and-cut search and our heuristic solutions are often close to optimal, the starting solutions enable CPLEX to cut off many branches of the search tree. Consequently, CPLEX can find optimal solutions for the blocks much faster than searching from scratch.

When we use CPLEX to find optimal solutions for very hard blocks, we need to set the parameter CPX_PARAM_MIPEMPHASIS to be CPX_MIPEMPHASIS_BESTBOUND. Our experiments show that this use of CPX_PARAM_MIPEMPHASIS enables CPLEX to find an optimal solution for a very hard block of the HuRef dataset without using much memory (namely, within 6.6GiB).

7 EXPERIMENTAL RESULTS FOR THE HUREF DATASET

7.1 The Number of Nonsingular Blocks

For the all-heterozygous (respectively, general) case, Table 6 (respectively, Table 7) summarizes the numbers of nonsingular blocks of the 22 chromosomes in the HuRef dataset obtained by block reduction only and by both block reduction and block decomposition, respectively. As can be seen from the tables, block decomposition enables us to cut the nonsingular blocks (obtained by block reduction only) of a chromosome into many more smaller blocks.

7.2 Heuristic Results for The Hard Blocks

The 3 (respectively, 7) hard blocks for the all-heterozygous (respectively, general) case are shown in the first (respectively, second) part of Table 8. Since they are hard, we use the (first) heuristic detailed in Section 2.2 of the main text to find heuristic solutions for them. The results are summarized in the same table, where the optimal MEC scores for the hard blocks are found by CPLEX.

Table 6. The numbers of nonsingular blocks of the 22 chromosomes of the HuRef dataset in the all-heterozygous case, where column "chr" shows the index number of a chromosome, column "#r_blocks" shows the number of nonsingular blocks obtained by block reduction only, and column "#rd_blocks" shows the number of nonsingular blocks obtained by both block reduction and block decomposition.

chr	#r_blocks	#rd_blocks	chr	#r_blocks	#rd_blocks
1	1524	1981	12	911	1156
2	1735	2327	13	578	743
3	1262	1615	14	609	770
4	1239	1607	15	540	700
5	1206	1518	16	654	815
6	1088	1417	17	551	666
7	1103	1395	18	495	623
8	910	1157	19	392	520
9	848	1070	20	433	534
10	803	992	21	232	298
11	790	1022	22	246	321

Table 7. The numbers of blocks of the 22 chromosomes of the HuRef dataset in the general case, where the columns mean the same as in Table 6.

chr	#r_blocks	#rd_blocks	chr	#r_blocks	#rd_blocks
1	1524	1956	12	911	1144
2	1735	2184	13	578	735
3	1262	1602	14	609	764
4	1239	1582	15	540	687
5	1206	1503	16	654	811
6	1088	1400	17	551	659
7	1103	1375	18	495	616
8	910	1146	19	392	510
9	848	1053	20	433	530
10	803	985	21	232	296
11	790	1012	22	246	319

8 SWITCH ERROR RATES

In this section, we examine the switch error rates of our program for a portion of the simulated datasets of Geraci (2010). Since we also consider the general case, we need to first generalize the definition of switch error rate. For this purpose, let $H = (h_1, h_2)$ be a pair of real haplotypes and $H' = (h'_1, h2')$ be a pair of haplotypes computed by a program, where h_1, h_2, h'_1, and h'_2 have the same length n.

For convenience, we use $h[i]$ to denote the ith bit of a haplotype h. For each $1 \leq i \leq n$ with $h_1[i] \neq h_2[i]$ and $h'_1[i] \neq h'_2[i]$, we define a bijection m_i from $\{1, 2\}$ to itself as follows. If $h_1[i] = h'_1[i]$, then $m_i[1] = 1$ and $m_i[2] = 2$; otherwise, $m_i[1] = 2$ and $m_i[2] = 1$.

We next define the *switch error* $e_i \in \{0, 1\}$ at each position $i \in \{1, 2, \ldots, n\}$ as follows.

Table 8. Heuristic results for the hard reduced-blocks of the HuRef dataset in the all-heterozygous and the general cases, where column "chr" shows the index number of a chromosome, column "pos" shows the starting position of a reduced block in the chromosome, column "#SNPs" shows the number of SNP sites in the block, column "score" shows the MEC score of an optimal or heuristic solution for the block, column "lb" shows the lower bound on the MEC score of an optimal solution for the block found by our heuristic, and column "time" shows the running time (in minutes) for the block.

chr	pos	#SNPs	optimal		heuristic		
			score	time	score	lb	time
The all-heterozygous case							
1	77502	1015	1963	52	1963	1961	7
15	783	1165	2339	1566	2340	2312	254
22	1	398	1518	68	1519	1514	16
The general case							
1	11500	461	789	2011	790	763	6
1	77502	1015	1642	829	1642	1639	260
8	14931	1073	570	49	570	570	49
15	1	779	684	658	684	682	56
15	782	1166	?	?	2036	2015	2121
21	76	354	536	35	536	525	2
22	1	398	1310	11942	1310	1307	920

- If exactly one of $h_1[i] = h_2[i]$ and $h'_1[i] = h'_2[i]$ holds, then $e_i = 1$.
- If both $h_1[i] = h_2[i]$ and $h'_1[i] = h'_2[i]$ hold but $h_1[i] \neq h'_1[i]$, then $e_i = 1$.
- If both $h_1[i] \neq h_2[i]$ and $h'_1[i] \neq h'_2[i]$ hold and the largest integer $j \in \{1, 2, \ldots, i-1\}$ with $h_1[j] \neq h_2[j]$ and $h'_1[j] \neq h_2[j]$ satisfies that $m_i[1] \neq m_j[1]$ (and hence $m_i[2] \neq m_j[2]$), then $e_i = 1$.
- If none of the above three cases occurs, $e_i = 0$.

The *switch error rate* between H and H' is $\sum_{i=1}^{n} e_i/n$.

Table 9 shows an example pair of H and H', where $e_1 = e_2 = e_4 = e_6 = e_7 = 1$ and $e_3 = e_5 = e_8 = 0$. So, the switch error rate between the example H and H' is 0.625.

Table 9. An example pair of $H = (h_1, h_2)$ and $H' = (h'_1, h'_2)$ with 5 switch errors occurring at positions 1, 2, 4, 6, and 7.

haplotype	1	2	3	4	5	6	7	8
h_1	0	0	0	0	0	0	0	0
h_2	0	1	1	1	1	0	1	1
h'_1	0	1	0	1	1	0	0	0
h'_2	1	1	1	0	0	1	1	1

The average switch error rates for a portion of the simulated datasets of Geraci (2010) are shown in Table 10. From the table, we can see that the average switch error rates in the general case are significantly smaller than those in the all-heterozygous case.

Table 10. Average switch error rates of our exact program for those simulated datasets of Geraci (2010) corresponding to those combinations (ℓ, c, e) with $\ell \in \{100, 350\}$, $c \in \{3, 5, 8, 10\}$, and $e = 10\%$.

c	$l = 100$		$l = 350$	
	general case	all-hete. case	general case	all-hete. case
3	0.0735	0.1395	0.0684	0.137
5	0.0203	0.159	0.0235	0.1716
8	0.0059	0.1987	0.0057	0.2202
10	0.0015	0.2156	0.0022	

9 EXPERIMENTAL RESULTS FOR THE NA12878 DATASET

Our experimental results for the NA12878 dataset of Duitama *et al.* (2012) are summarized in Table 11 from which one can see that our approach is much faster than those of He *et al.* (2010) and Bansal & Bafna (2008). Since the MaxSAT solver (namely, Clone) used by He *et al.* (2010) can only run on a 32-bit Linux machine, the running times of our program and Clone in Table 11 are measured on a 32-bit Linux desktop PC with i7-975 CPU and 5.9GiB RAM. On the other hand, since HapCUT of Bansal & Bafna (2008) cannot run on a 32-bit Linux machine, the running times of HapCUT in Table 11 are measured on a (more powerful) 64-bit Linux desktop PC with i7-3960X CPU and 31.4GiB RAM. For each chromosome in the NA12878 dataset, we ran HapCUT 10 times, where each run was given the default maximum number (namely, 100) of iterations. Consequently, for each chromosome in the NA12878 dataset, the running time of HapCUT in Table 11 is the total time of the 10 runs while the score of HapCUT in the table is the best score among the 10 runs.

REFERENCES

DePristo, M.A., Banks, E., Poplin, R., *et al.* (2011) A framework for variation discovery and genotyping using nextgeneration DNA sequencing data. *Nature Genetics*, **43**, 491-498.

Li, H., Ruan, J. and Durbin, R. (2008) Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res.,*, **18**, 1851.

Li, H. *et al.* (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**, 2078-2079.

Table 11. Comparing our approach with the MaxSAT approach of He *et al.* (2010) and HapCUT of Bansal and Bafna (2008) using the NA12878 dataset, where the columns mean the same as in Table 8.

	all-heterozygous case					general case	
	optimal			heuristic		optimal	
chr	score	our time	MaxSAT time	score	HapCUT time	score	time
1	7120	2.0	15.7	7124	250	6889	2.7
2	6926	2.2	16.5	6936	281	6700	2.8
3	5295	1.9	24.1	5299	213	5122	2.4
4	4184	1.5	12.6	4186	220	4072	1.9
5	4736	1.7	12.6	4743	217	4637	2.0
6	5448	2.0	13.0	5464	425	5248	2.5
7	4311	1.6	11	4311	214	4174	2.0
8	4434	1.8	10.8	4439	237	4301	2.3
9	4102	2.6	8.7	4104	194	3974	1.7
10	4653	1.6	10.3	4658	213	4508	2.0
11	4013	1.5	9.9	4018	192	3903	1.9
12	4041	1.6	10.1	4042	145	3907	2.0
13	2772	1.1	6.6	2778	113	2669	1.4
14	2908	1.3	6.5	2909	119	2814	1.8
15	3000	1.1	5.9	3001	119	2903	1.4
16	3976	1.9	9.0	3983	246	3844	2.5
17	3606	1.5	8.3	3612	115	3448	1.9
18	2414	1.2	6.1	2417	100	2337	1.2
19	2838	1.0	4.5	2840	94	2707	1.3
20	2888	1.1	5.3	2892	115	2783	1.5
21	1416	0.5	2.8	1416	58	1367	0.6
22	2549	1.4	8.7	2550	111	2422	2.2