BIOINFORMATICS ORIGINAL PAPER

Vol. 24 no. 1 2008, pages 86–93
doi:10.1093/bioinformatics/btm552

Genetics and population analysis

Identification of linked regions using high-density SNP genotype data in linkage analysis

Guohui Lin[1,†], Zhanyong Wang[2,†], Lusheng Wang[2], Yu-Lung Lau[3] and Wanling Yang[3,*]

[1]Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada, [2]Department of Computer Science, City University of Hong Kong, Kowloon, and [3]Department of Paediatrics & Adolescent Medicine, LKS Faculty of Medicine, The University of Hong Kong, 21 Sassoon Road, Hong Kong

Received on August 10, 2007; revised on October 13, 2007; accepted on October 30, 2007

Advance Access publication November 17, 2007

Associate Editor: Keith Crandall

ABSTRACT

Motivation: With the knowledge of large number of SNPs in human genome and the fast development in high-throughput genotyping technologies, identification of linked regions in linkage analysis through allele sharing status determination will play an ever important role, while consideration of recombination fractions becomes unnecessary.

Results: In this study, we have developed a rule-based program that identifies linked regions for underlined diseases using allele sharing information among family members. Our program uses high-density SNP genotype data and works in the face of genotyping errors. It works on nuclear family structures with two or more siblings. The program graphically displays allele sharing status for all members in a pedigree and identifies regions that are potentially linked to the underlined diseases according to user-specified inheritance mode and penetrance. Extensive simulations based on the χ^2 model for recombination show that our program identifies linked regions with high sensitivity and accuracy. Graphical display of allele sharing status helps to detect misspecification of inheritance mode and penetrance, as well as mislabeling or misdiagnosis. Allele sharing determination may represent the future direction of linkage analysis due to its better adaptation to high-density SNP genotyping data.

Availability: http://paed.hku.hk/uploadarea/yangwl/html/index.html

Contact: yangwl@hkucc.hku.hk

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 INTRODUCTION

The fundamental problem in linkage analysis is to identify regions whose allele is shared by all or most affected members in a family but by none or few unaffected family members. In dominant inheritance situation sharing of one mutation allele can cause disease phenotype while in recessive cases, sharing of two disease alleles in that region is necessary for affected status. Traditional linkage analyses were usually based on sparse microsatellite markers when recombination fraction between markers has to be considered, and analysis tools designed for analyzing microsatellite genotype data may not work optimally with high-density SNP genotype data. With the rapid advances in high-throughput genotyping technologies, programs for accurate determination of allele sharing in families will play an ever important role for linkage analysis. This should allow extraction of full inheritance information and accurate determination of the shared regions among family members. Consideration of recombination fraction is no longer necessary for this type of genotyping data.

Existing software tools for linkage analysis are all probabilistic, where recombinant rates are estimated in a way to maximize the likelihood of the observed data (Abecasis *et al*., 2002; Gudbjartsson *et al*., 2000; Kruglyak *et al*., 1995; Lander and Green, 1987). The well-known software tools in this regard include GeneHunter (Kruglyak *et al*., 1995), LINKAGE (Lathrop *et al*., 1984), Allegro (Gudbjartsson *et al*., 2000) and Merlin (Abecasis *et al*., 2002), etc. with different performance and efficiencies. The traditional linkage analysis method is limited either by their ability dealing with large amount of markers such as the ones using Elston–Steward algorithm (Elston and Stewart, 1971) or by the size of the families, such as the ones using the Lander–Green algorithm (Lander and Green, 1987), although tremendous improvement has been applied to them through modifications such as in Merlin (Abecasis *et al*., 2002) and Allegro (Gudbjartsson *et al*., 2000).

Leykin *et al*. (2005) and Sellick *et al*. (2004) demonstrated that high-density SNP genotype data, such as those from microarrays, can be used for large-scale and cost-effective linkage analysis. The underlying principle is that there will be sufficient number of informative markers between any two recombination points, and thus the allele sharing status among the family members can be unambiguously determined. This renders consideration of recombination fraction between markers unnecessary and demands highly efficient programs for allele sharing determination that work on a large number of markers. Accurate determination of allele sharing among family members is not restrained by inheritance mode assumption and it makes further applications much easier.

To approach this goal, we have developed an efficient, rule-based program that can accurately determine the haplotype

*To whom correspondence should be addressed.

†The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint First Authors.

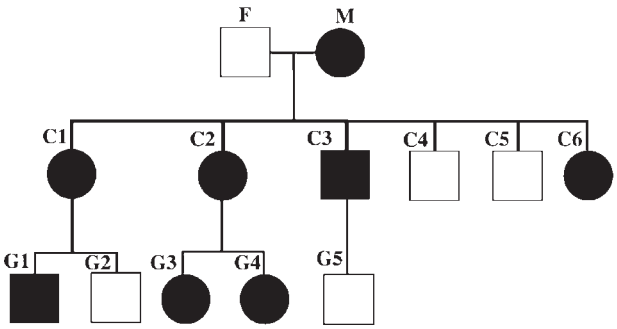

Fig. 1. The structure of the test pedigree. The designated affected status was shown by filled circles/boxes for affected and open circles/boxes for unaffected.

allele sharing status for individuals in a family and thus identify the potential mutation regions. Our rule-based algorithm greatly reduces computational intensity, even in the face of large dataset such as high-density SNP genotyping. It graphically displays the allele sharing status among family members, and provides a way for other downstream applications. The key component of the program is a rule-based haplotype inference algorithm that assigns haplotypes to the smallest nuclear family in a top-down fashion, which could either be a trio or a parent and a child, and determines haplotypes of other family members in a sequential order. The program uses minimum number of breakpoints to explain the genotype data. Parental haplotype phases are revised as more people are added as long as the revision reduces the total number of breakpoints and still explains the genotype data. An error correction step was added after all the revision. It is worth noting that many haplotype inference algorithms and programs have been developed (Chung and Gusfield, 2003; Gusfield, 2001; Halperin and Eskin, 2004; Qian and Beckmann, 2002; Zhang *et al.*, 2005). However, our program is the first to emphasize on correctly inferring allele sharing status (rather than haplotype phase) and on identifying linked regions for affected families.

Extensive simulations on Affymetrix Human Mapping 50K/250K GeneChips show that our program can always correctly detect allele sharing status and mutation regions with accuracy, based on user-specified inheritance mode and parameters on penetrance and phenocopy.

2 METHODS

2.1 The dynamic programming algorithm

Our program first finds the nuclear family on top of the family tree, which is either a trio consisting of parents and a child or a parent and a child, and assigns haplotype phase to each individual without generating any breakpoints. So the program works with missing parental data. In the test pedigree as shown in Figure 1, the first smallest nuclear family consists of F, M, and one of their children, C1. The program goes through the smallest nuclear families in a breadth-first-search (BFS) fashion. For each such smallest nuclear family, our program employs a dynamic programming algorithm to assign the haplotypes, under the partial assignment if any, via using a minimum number of breakpoints.

Note that after assigning haplotypes to the parents and the first child, some (informative) sites in F, M and C1 become unswappable (when not all three of them are heterozygous), while some others remain to be swappable (when all three are heterozygous). The program proceeds to consider the next smallest nuclear family, which in the test pedigree consists of F, M and child C2, so on and so forth until all family members are gone through.

For the nuclear family of F, M and C2, the algorithm allocates 4 one-dimensional tables DP^{00}, DP^{01}, DP^{10} and DP^{11}, where $DP^{pq}[i]$ records the minimum number of breakpoints necessary from site i on to the end of the chromosome, when child C2 inherits the p haplotype of F at site i and the q haplotype of M at site i for $p=0$, 1 and $q=0$, 1. In the case that such an inheritance is impossible, $DP^{pq}[i]$ is set to a large value 'MAX'. Therefore, the least value among $DP^{00}[0]$, $DP^{01}[0]$, $DP^{10}[0]$ and $DP^{11}[0]$ is the minimum number of breakpoints needed to explain the genotype data for this nuclear family, given that F and M have partially assigned haplotypes (due to child C1).

The recurrences for table entry $DP^{00}[i]$ calculation have to go through a number of cases. For example, if it happens that

(1) $F[i]^0 = C2[i]^0$ and $M[i]^0 = C2[i]^1$,

then $DP^{00}[i]$ can be set to $DP^{00}[i+1]$ and if at this site at least one of them is unswappable then all three of them become unswappable. If

(2) $F[i]^1 = C2[i]^0$, $M[i]^0 = C2[i]^1$, and at this site F is swappable,

then $DP^{00}[i]$ can still be set to $DP^{00}[i+1]$ with $F[i]^0$ and $F[i]^1$ swapping their values and, similarly, if at this site one of M and C2 is unswappable then all three of them become unswappable. Other cases in which $DP^{00}[i]$ can be set to $DP^{00}[i+1]$ include

(3) $F[i]^0 = C2[i]^0$, $M[i]^1 = C2[i]^1$, and at this site M is swappable,

(4) $F[i]^0 = C2[i]^1$ and $M[i]^0 = C2[i]^0$, and at this site C2 is swappable,

(5) $F[i]^1 = C2[i]^0$, $M[i]^1 = C2[i]^1$, and at this site both F and M are swappable,

(6) $F[i]^1 = C2[i]^1$, $M[i]^0 = C2[i]^0$, and at this site both F and C2 are swappable,

(7) $F[i]^0 = C2[i]^1$, $M[i]^1 = C2[i]^0$, and at this site both M and C2 are swappable, and

(8) All three of F, M and C2 are swappable at this site.

Note that under no genotyping error assumption and following Mendelian inheritance rules, it could be possible that none of the above eight cases happens and therefore $DP^{00}[i]$ gets the largest value 'MAX'. But whenever $DP^{00}[i]$ gets a proper value less than 'MAX', it is compared with $DP^{01}[i+1]+1$, $DP^{10}[i+1]+1$, and $DP^{11}[i+1]+2$, and takes the minimum value among them:

- $DP^{00}[i]$ is not greater than any of $DP^{01}[i+1]+1$, $DP^{10}[i+1]+1$ and $DP^{11}[i+1]+2$. In this case, there is no breakpoint in between sites i and $i+1$.
- $DP^{00}[i]$ is set to $DP^{01}[i+1]+1$. In this case, one maternal breakpoint is created in between sites i and $i+1$.
- $DP^{00}[i]$ is set to $DP^{10}[i+1]+1$. In this case, one paternal breakpoint is created in between sites i and $i+1$.
- $DP^{00}[i]$ is set to $DP^{11}[i+1]+2$. In this case, one maternal breakpoint and one paternal breakpoint are created in between sites i and $i+1$.

The entries in the other three tables DP^{01}, DP^{10} and DP^{11} are similarly calculated.

After all the four tables have been filled out, the minimum of $DP^{00}[0]$, $DP^{01}[0]$, $DP^{10}[0]$ and $DP^{11}[0]$ records the least number of breakpoints

necessary to explain the genotype data for F, M and C2, under the partial haplotype assignment. A standard tracing will recover all those paternal and maternal breakpoints associated with this least number, as well as the haplotypes assigned to all three members. And there are previously swappable markers become fixed through the process.

The program continues to consider the next smallest nuclear family. When a smallest nuclear family consists of only one parent, the dynamic programming algorithm still applies but the number of cases to be considered is much smaller (about half of the trio case). At the end, if there are still swappable SNP sites, randomly fixing them, will produce complete haplotypes for all the family members.

2.2 Breakpoint pushback

Depending on the order of consideration, a breakpoint assigned to one child could have been assigned to other siblings. When multiple siblings showed recombination at the same position, it is an indication that parental phase was not assigned correctly. Revision of the parental phase will convert the recombination to one child and reduce the overall number of recombinations needed to explain the data. Our program performs a process called *breakpoint pushback* to examine for every breakpoint location that whether revising the parental haplotypes can reduce the total number of breakpoints. For example, when there is a site at which there are more than half of the second generation members having the same (either paternal or maternal) breakpoints at the same location, our program will push this particular breakpoint back to their parent through revising his/her haplotypes. When two breakpoints are extremely close to each other in one individual, the program also tries to push one of them to other siblings by revising the parental phase. In reality, this usually only works when only two siblings are available. Since when there are more than two siblings, this usually generates more breakpoints and would therefore violate our rule of breakpoint minimization. This is important as we judge whether some breakpoints are generated due to genotype errors, as discussed below.

2.3 Genotype data error correction

For large-scale SNP genotyping, certain number of experimental errors is unavoidable. In our real case test, we have noticed that there are breakpoint pairs spanning a very short distance in one individual, separated by one or two informative marker sites. Therefore, we have adopted a rule to correct errors like these instead of calling for two consecutive breakpoints to explain the data. In the simulation and the real case study described below, we have adopted a criterion that, when two adjacent breakpoints (or 1 breakpoint when at the end of the chromosome) are <1 Mb apart and in between there are less than three informative SNP sites to support break calling, the two breakpoints are interpreted as caused by genotyping errors. The optimal criterion for error correction is upon the user to decide and should be based on the quality of the genotyping data, and the density of the SNP markers used.

2.4 Identifying mutation regions

After error correction, we go through every SNP site on the chromosome and find the regions shared by all or most of the affected family members (considering phenocopy) but none or few of the unaffected family members (considering penetrance). Those regions are reported as suspected mutation regions. Since multiple solutions may exist for certain regions and our program gives only one haplotype solution, it is possible that the reported regions do not completely overlap with the mutation region. Since our program pushes the breakpoints towards the downstream of the chromosome, we extend the found regions to the left by looking at the SNP sites one by one from right to left and add the consecutive sites where we can revise the haplotype allele such that the allele is shared by diseased family members, but not the normal family members. The program in its current form works on families in a one by one fashion. The users are asked to input the maximum number of unaffected individuals to be allowed to share the mutation allele (allowing for penetrance) and the maximum number of affected individuals in the family to be allowed not to share the potential mutation region (allowing for phenocopy). The program then chooses the potential linked region(s) based on the allele sharing status determined by the program and the parameters entered by the user.

2.5 Graphical display of allele inheritance

After haplotype assignment, breakpoint pushback and error correction, the allele inheritance status for any founder (F or M in the family structure of Fig. 1) can be displayed for users to visualize the allele sharing status. Allele sharing visualization provides a way of easily spotting the regions shared by affected but not by unaffected in families, or to be used to evaluate the possibility of incomplete penetrance, phenocopy or potential errors such as misdiagnoses or mislabeling.

3 COMPUTATIONAL RESULTS

3.1 Generating haplotype data

To test our program, we need to generate haplotype datasets and see if our program can infer these haplotypes from the corresponding genotype data. We have used real genotype data from a three generation family (Fig. 1), as well as simulation data with various pedigree structures.

Children haplotypes were generated through random inheritance of parental alleles after simulating recombination events according to the χ^2 model for recombination with m equals 4 (Broman, 2000; Zhao *et al*., 1995) and according to male/female averaged genetic map for chromosome 1 downloaded from HapMap (http://hapmap.org). The χ^2 model assumes that crossover intermediates (C events) are distributed along the four-strand sister chromatid bundle based on a Poisson distribution with a rate of $2(m+1)$ C events per Morgan, and every C event will either resolve in a crossover (Cx) or not (Co). When a C event resolves in a Cx, the next m C's must resolve as Co events, and after m Co's the next C must resolve as a Cx, i.e. the C's resolve in a sequence of $\ldots\mathrm{Cx(Co)}^m\mathrm{Cx(Co)}^m\ldots$ The leftmost C has an equal chance to be one of $\mathrm{Cx(Co)}^m$ (Zhao *et al*., 1995). The simulation process assumes no chromatid interference. The siblings were simulated to randomly inherit one strand of the four-strand chromatid bundle from each parent. However, in the simulation that considers disease status, a mutation was randomly assigned to be close to a SNP site, and the affected offsprings were forced to inherit the mutation strand and the unaffected were forced not to inherit the mutation strand. This rule loosens when considering incomplete penetrance or phenocopy in the simulation. After all second generation individuals have their simulated haplotypes, we simulated the haplotypes for the third generation members using their parent's haplotypes with recombination, together with a pair of randomly generated haplotypes for the other missing parent. We have simulated 95 families this way, and each were repeated the inheritance process 20 times with the assumption of no genotype error, 0.1% of genotype error,

Table 1. Recovery of breakpoints and linked regions using 50K or (250K) data

	Breakpoint Recovery Precision (250K)	Breakpoint Recall (250K)	Shared mutation region recovery (250K)	False positive linked region (250K)
No error	99.29% (99.67%)	97.51% (99.37%)	100% (100%)	0% (0.37%)
0.1% error	94.43% (91.72%)	97.47% (99.17%)	100% (100%)	0.32% (0.74%)
0.5% error	78.05% (66.7%)	97.37% (98.89%)	99.95% (100%)	2.94% (4.06%)

or 0.5% genotype error, respectively. For the error simulation, each site was simulated to have 0.1% or 0.5% chance to be misgenotyped, to any of the other two genotypes with equal chance (such as AA to AB or to BB). Simulations of other family structures were done in a similar manner.

3.2 Breakpoint Recovery Accuracy

In the breakpoint generation process, a simulated (real) breakpoint could locate at a homozygous SNP site and is not possible to be recovered precisely by any computational haplotyping algorithms. Also, our program pushes the breakpoints towards the downstream of the chromosome, i.e. creating a breakpoint only if necessary. A referred breakpoint is said to be *recovered* if (1) it is identical to the real breakpoint or (2) the inferred breakpoint is on the left of the real breakpoint and they are within 5 SNP's or (3) the inferred breakpoint x is on the right of the real breakpoint y and we can re-construct the haplotype SNP's of the segment between x and y for the individual while the haplotype data of other individuals on this segment remain unchanged. Case (3) is reasonable since there could be multiple solutions and our program gives only one solution.

The breakpoint recovery *precision* is defined as the number of simulated breakpoints being recovered (excluding false positive) divided by the number of breakpoints generated by the program (including false positives). The breakpoint recovery *recall* is defined as the number of simulated breakpoints being recovered (excluding false positives) divided by the number of simulated (real) breakpoints.

The average precision and recall over 1900 simulated datasets are 99.29% and 97.51%, respectively (Table 1). On this particular pedigree structure, the average number of simulated breakpoints over 1900 datasets for chromosome 1 is 45.36; the average number of breakpoints created by our program is 44.91, among which 44.47 are true positives. All the numbers are improved when using 250 K chips without error. (See the first row in Table 1.) When genotype errors were simulated, the breakpoint precision was reduced while recall remains steady (see the second and third rows in Table 1).

When we ran our program on these datasets by simulating 0.5% and 0.1% genotyping errors without turning on error correction feature, our program generated many breakpoints on these error sites and the achieved breakpoint recovery precision was only 22.3% and 58.5%, respectively, though recall remains high at 97.37% and 98.1%, respectively. With the genotype error correction option turned on, the achieved breakpoint recovery precision was improved to 78.05%, and 94.43%, respectively. These results demonstrate the strong immunity of our program to genotyping errors.

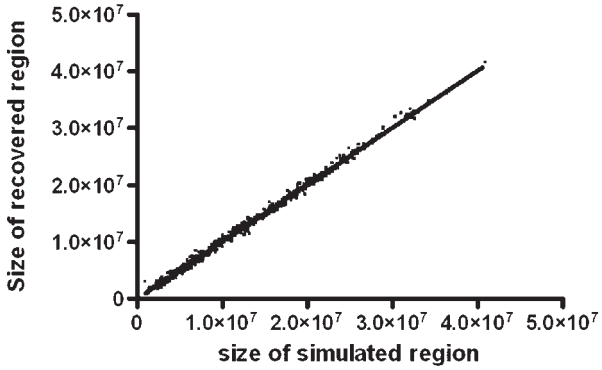

Fig. 2. The correlation between the sizes (bp) of the simulated mutation regions by all the affected and the sizes of the recovered regions by our program (bp).

3.3 Identification of linked regions for affected families

Assuming an autosomal dominant model, we simulated the disease mutation for this pedigree structure and tested whether our program can detect the disease regions with accuracy and specificity. In the simulations assuming no error, and 0.1% error, we have recovered all the simulated mutation regions. In 1900 simulations with error rate of 0.5%, the program successfully identified the mutation regions 1899 times and failed only once. Examination of the failed case revealed that the simulated region is relatively small (1.7×10^6 bp), and happened to occur in a region that is sparsely covered by the 50 K Xba chip, and there is no informative sites between two simulated breakpoints. This may raise an issue that even for linkage analysis using high-density SNP markers, there could be some information gain using even denser marker genotyping, such as Affymetrix 250K chips, in rare situations when the shared regions are small. Figure 2 showed the correlation of the sizes of the simulated mutation regions and those of the recovered regions. This demonstrated that, using the 50 K genechip data, we can recover nearly all the simulated mutation regions with good accuracy (the correlation coefficient $r^2 = 0.9949$, and the P-value $P < 0.0001$).

When generating the 1900 haplotype datasets for testing, there are 40 datasets that contains two regions that are shared by all affected family members but none of the unaffected. For each of the 40 cases, one of the regions is the real mutation region and the other region is generated by chance based on the χ^2 recombination model. In these cases, the real mutation regions are longer than the regions generated by chance. Out of the 40 cases, our program successfully detected both regions for 37 cases and missed the short regions (generated by chance)

3 times. The three missed regions are very small in size, and were located in regions with sparse marker coverage.

A *false positive* region is a region reported by the program as potential mutation region but is actually not a region shared by all the affected family members in the simulation based on the χ^2 recombination model. For the simulations without error, we did not find any false positive region. However, the number of false positive regions increased to 0.32% and 2.94% when 0.1% and 0.5% of errors were simulated. The sizes of those false positive regions are usually very small, and usually reflect cases when an error was simulated close to a breakpoint and when surrounding markers are usually non-informative. With our error correction feature turned off, the number of false positive regions increases dramatically. Again, this shows that our program has strong immunity to data errors.

We also performed the simulation process according to incomplete penetrance or phenocopy scenarios, and in all cases similar results to assuming full penetrance and no phenocopy were produced.

3.4 Applicability to different family structures

We tested different pedigree structures for the applicability of the program on different families. Apparently families with only one child are not informative. When there are two siblings with both parental data available, the program produced poor precision and recall in terms of recombination sites recovery but accurately recovered allele sharing status and detected all the simulated mutation regions. The poor recovery of the recombination sites is understandable since no pushback process can be initiated for families having less than three siblings. This is not a problem since the purpose of the program is to determine allele sharing status and to identify linkage regions. For families with three siblings and parental data available, the program performs well in both crossover point recovery and allele sharing status determination, with precision of 98.3%, recall of 96.2% and 100% recovery of simulated mutation regions. So the minimum requirement for the pedigree structure is two siblings with parental data available, and the program performs better with increased number of siblings. In the situation when one parental data is missing, the program performs well on certain situations but not on others (see Supplementary table) in terms of recovery rate of mutation regions and accuracy of the recovered regions. The program will not work when parental data is totally missing. Further development of the program in these situations is needed in order to make it work on various family structures. With the ever increasing knowledge of SNP and haplotype allele frequencies for a given population, missing data could be imputed and to help determine allele sharing status when parental data is not available, but is beyond the scope of this deterministic algorithm discussed in this study.

3.5 A real case study

We also used real data from a family consisting of 13 members with the structure as shown in Figure 1 with designated affected status in order to compare our program with other linkage analysis methods. Here we have shown the result for chromosome 17, to demonstrate the output of our program.

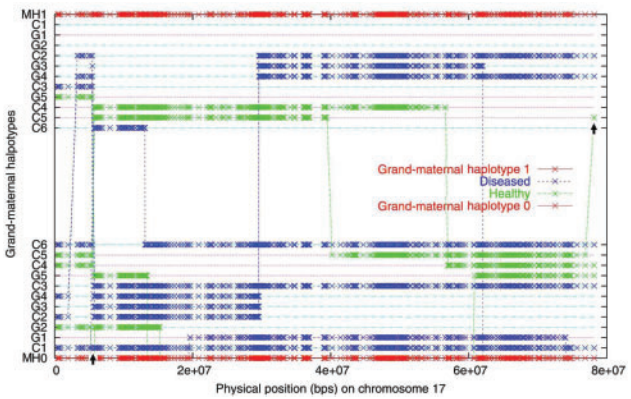

Fig. 3. The computed grand-maternal haplotype allele sharing status among the 11 members, excluding grandparents, assuming no genotyping error.

In Figure 3, we have showed the allele inheritance status from the affected maternal grand parent, M, with the disease status labeled in different colors. After assigning these haplotypes to all individuals in the family, the program started re-haplotyping the whole family to see whether the number of breakpoints could be further reduced. Note that, as pointed by the left arrow at the bottom, a breakpoint was called on C2, C3, C4, C5 and C6, while revision of maternal phase here and calling the breakpoint on C1 instead would effectively abolish the breakpoints called on all other siblings. Also note that in Figure 3, as pointed by the arrow on the right in the middle of the figure, a single SNP genotype requires assignment of a breakpoint, and the program corrected it according to the error correction rule (between this site and the end of the chromosome is below 1 Mb). However, it should be noted that the region is poorly covered, and more genotyping is needed to determine the inherited allele in this telomere region for individual C5. Our program outputs warnings on error correction for users to make judgment or to validate experimentally on those regions.

The final grand-maternal haplotype allele sharing status among the 11 members (excluding the grandparents) is depicted in Figure 4A. Here we also compared our result for this family for chromosome 17 with result from Merlin. As shown in Figure 4A, region 2 is clearly a region shared by all the designated affected individuals in this family and the result is consistent with Merlin output (Fig. 4B) reporting a lod score of 3 for this region. However, region 1 is clearly shared by G1 and G2 and is inherited from unaffected F individual in the family as shown in Figure 5, pointed as region 1 by double arrow, and should not be a linked region.

4 DISCUSSION

4.1 Allele sharing determination

The general problem of computing the minimum number of crossover breakpoints to explain a genotype dataset is an NP-hard problem (Doi *et al.*, 2003; Gusfield, 2001). On the other hand, the minimum number of breaks may not necessarily conform to the true situations. In the simulations,

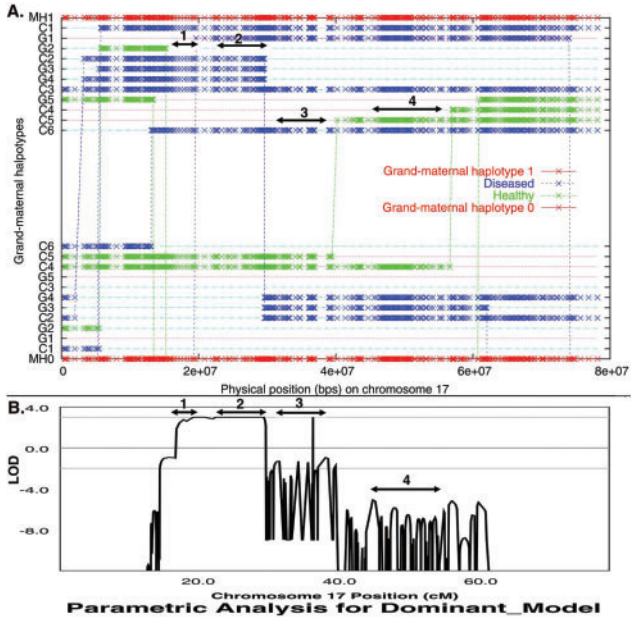

Fig. 4. The grand-maternal haplotype allele sharing for chromosome 17 among the 11 members, excluding grandparents, after breakpoint pushback process and error correction, and lod scores from Merlin with parametric testing of dominant model. (**A**) Output from our program with allele sharing status for founder M. (**B**) Output from Merlin with LOD scores. Regions for comparison were labeled with double arrows.

we have observed that the number of breakpoints generated by our program is usually less than that of the simulated data. In most cases, the parental haplotypes can be partitioned into segments at the crossover breakpoints. If we wish to construct the haplotypes, we would face an exponential number of possible combinations of the haplotype segments (Hodge *et al.*, 1999). However, the desired allele sharing status among the family members does not require the completely identified phases. Figures 3 and 4 clearly show that the input genotype data can be explained using different numbers of breakpoints (with or without breakpoint pushback process), but the derived grand parental allele sharing status is identical to each other. This suggests that the allele sharing status determined by our program is reliable and can be used for identifying linked regions for affected families in linkage analysis.

4.2 Linked region identification

With the allele sharing status determined accurately by our program, one immediate application of this sharing map is to determine the chromosomal region(s) responsible for certain genetic diseases without considering recombination fraction. We compare our program with the widely used linkage analysis software Merlin. As shown in Figure 4, both Merlin and our program detected a roughly 10 Mb region in chromosome 17 that is shared by all the diseased members but none of the normal family numbers. Merlin reported a lod score of 3.0 on this region, while our program showed a complete co-segregation of this region with the disease, and should warrant a lod score of 3.3 as calculated by $Z(\Theta) = n\log(2)$ for

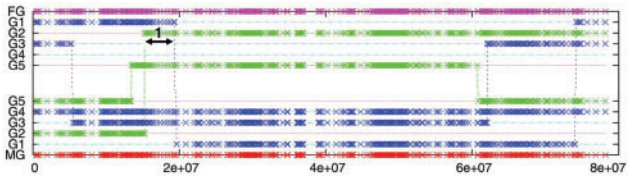

Fig. 5. The grand-parental haplotype allele sharing status among the 5 third generation members for chromosome 17. Region 1 is labeled with double arrow and shown to be shared by G1 and G2 and inherited from the unaffected grandfather allele (FG). The grand paternal allele was shown on top of the figure, and the grand maternal allele was shown on the bottom (MG). The alleles for affected third generation individuals, G1, G3, G4, were labeled in blue; the alleles for the unaffected G2 and G5, were labeled in green.

full penetrance cases (Ott, 1999), where n stands for the count of non-recombinants and is 11 here. A gain in lod score is an advantage of the deterministic method comparing to classical maximum-likelihood methods.

We have compared in some detail on the output from Merlin and our program for chromosome 17. It turned out that although both programs detected the same region, our program does show better accuracy on the left edge of the shared region (indicated as region 2 in Fig. 4). Region 1 was clearly shown to be shared by diseased member G1 and unaffected member G2, and was derived from the normal grandfather as shown in Figure 5. This indicates that based on dense SNP genotype data, deterministic allele sharing detection may perform better in refining linked regions than programs based on likelihood and consideration of recombination fractions, though detailed comparison is still needed in order to evaluate their performances on dense SNP data.

We have checked the raw data corresponding to region 1 in Figure 4, and confirmed that both G1 and G2 inherited from the normal grandfather (F), supported by 10 informative SNP sites. For regions that do not co-segregate with the disease, Merlin showed irregular pattern of lod score calculation (regions 3 and 4 in Fig. 4B). This is probably due to the fact that, all these widely used linkage analysis software were originally designed based on traditional genotype method, usually from sparse microsatellite markers. When applying these methods on high-density SNP genotyping data, with marker spacing changed from hundreds of thousands kilobase to as small as a few kilobase, the calculation of recombination fraction between marker pairs become unnecessary. Merlin works better on SNP data due to the use of sparse trees (Abecasis *et al.*, 2002), but still considers recombination probability between adjacent markers. These programs work well on true linked regions due to multipoint analysis, but not so well on regions with low lod score, such as regions 3 and 4 as shown in Figure 4. For these two regions, our program clearly showed that there should be no recombination within each region.

With the availability of SNP markers at >1 per kb in most regions in our genome, and the high-throughput genotype method such as high density GeneChips, it is increasingly possible to locate the linked disease regions with high accuracy. For genetic diseases of Mendelian inheritance, the allele sharing

status produced by our program could unambiguously point out the mutation region(s). Even for complex diseases, which may display genetic heterogeneity, it is expected that certain mutations may have near dominant effect in multiplex families of early onset or particular phenotypes. The clear demonstration of allele sharing status among this kind of families could help locate the regions shared by affected members more often than can be explained by random chances. The allele sharing determination method does not have to deal with multiple comparison issues and is model-free. The method also generates a lod score 0.3 higher than likelihood methods considering recombination fractions, such as for region 2 shown in Figure 4. The graphical display presents a way for users to check for marker density, edges of the linked regions and for potential errors in diagnoses and sample labeling.

4.3 The uncertainty of breakpoint locations

In our program, breakpoints are always pushed towards the downstream of the chromosome as far as possible. That is, a breakpoint is not designated until the genotype cannot be explained otherwise. In this way, non-informative sites are always passed until an informative site is encountered and at that particular site, a crossover is called.

In the 1900 simulation datasets for mutation region detection, there are 24 cases that the simulated mutations locate to the right of nearby breakpoints, and between them there are no informative sites, and by default of the program the mutation will be included in the region on the left of the region shared by all diseased members but none of normal members. Thus, it is important to extend the suspected regions to the left as described in Section 2.4, to the nearest informative marker sites.

4.4 Computational efficiency

For linkage analysis, there are two basic algorithms for cosegregation detection for a marker or multiple markers. For the Elston–Stewart algorithm (Elston and Stewart, 1971), the complexity is linear in terms of the size of the pedigree but exponential in terms of the number of markers. The other algorithm is the Lander–Green algorithm (Lander and Green, 1987), whose complexity is exponential in terms of the size of the pedigree and linear in terms of the number of markers. The complexity of our program is linear in terms of both number of markers and sizes of the pedigrees. It is not restricted by memory resources and takes a few seconds to run in practice.

4.5 The rule-base algorithm

The line of work on rule-based methods can probably date back to Wijsman (1987). Several methods using the parsimony rule or similar mechanisms for haplotype reconstruction have been proposed (Chung and Gusfield, 2003; Halperin and Eskin, 2004; Li and Jiang, 2003; Qian and Beckmann, 2002; Zhang *et al*., 2005), yet the applicability of these programs to real high-density SNP genotyping data for linkage analysis in finding disease regions has never been tested.

Probably the closely related work to ours is the one done by Wirtenberger *et al*. (2005), where the authors used the genotype data by the GeneChip 10K arrays for two three-generation pedigrees connected via two siblings to infer haplotype blocks shared by the individuals and thus to deduce the genome-wide recombination distribution pattern. It appears that their approach decomposes the pedigrees into trios and then applies Mendelian inheritance rules on each trio to determine the haplotypes. This certainly does not take full advantage of the pedigree structure, but their results seemingly demonstrate that high-density SNP markers can be used for whole genome crossover inference. Unlike in Wirtenberger *et al*. (2005), we take full advantage of the pedigree structure, rather than decomposing it into trios.

Our program works on large datasets and accurately determines allele sharing status in the face of phase uncertainty at certain sites. This is important because phase uncertainty always exists, especially when the size of the pedigree is small (Hodge *et al*., 1999). Nevertheless, uncertainty on exact details of haplotype will not affect the accurate determination of allele sharing status when using high-density SNP markers. This is because the informative markers between crossover sites are usually sufficient to determine allele sharing status among family members. The importance of designing downstream algorithms that are not based on exact haplotype details are also pointed out in Stephens and Scheet (2005).

Our simulations also show that for rare cases when the shared mutation regions are very small, increasing marker density may be helpful for the detection of such regions. The allele sharing component of the program is model-free and provides flexibility for downstream applications in identifying causal mutations or susceptibility genes.

It needs to point out that the rule-based algorithm has its limitations. While it works well on nuclear family structures with two or more siblings, its performance is suboptimal on situations when one parental data is missing. It does not work on cases when parental data is totally missing. Families with missing parental data may be analyzed by considering information on population allele frequencies and haplotypes, and we are currently developing algorithms that can work on those situations. Nevertheless, our program is the first attempt to use deterministic method for linkage analysis and serves as a proof of principle that rule-based deterministic methods may be better adapted to high-density SNP genotyping data. We believe explorations in this area will eventually lead to programs that work on all cases in a way that suits best to the development in genotyping technologies.

5 CONCLUSIONS

We have developed a program for inference and visualization of haplotype allele sharing status among the members of a pedigree. It showed robustness and accuracy in detecting linked regions and in refining the edge of those regions. The program is highly efficient and is not restricted by the number of markers or the size of the families. The allele sharing status visualization feature also provides a means for post-computational inspection and should be extremely useful in situations such as misspecification of penetrance, misdiagnosis or mislabeling.

ACKNOWLEDGEMENTS

W.Y. is grateful to support from URC of the University of Hong Kong (200611159004). W.Y. and Y.L.L. thank the partial research support from The Shun Tak District Min Yuen Tong of Hong Kong. G.L. is grateful to the research support from CFI and NSERC. L.W. is supported by RGC of HKSAR, China [Project No. CityU 1196/03E].

Conflict of Interest: none declared.

REFERENCES

Abecasis,G. *et al*. (2002) Merlin–rapid analysis of dense genetic maps using sparse gene flow trees. *Nat. Genet*., **30**, 97–101.

Broman,K.W. and Weber,J. (2000) Characterization of human crossover interference. *Am. J. Hum. Genet*., **66**, 1911–1926.

Chung,R. and Gusfield,D. (2003) Perfect phylogeny haplotyper: haplotype inferral using a tree model. *Bioinformatics*, **19**, 780–781.

Doi,K. *et al*. (2003) Minimum recombinant haplotype configuration on tree pedigrees. In *Proceedings of the WABI'03*. pp. 339–353.

Elston,R.C. and Stewart,J. (1971) A general model for the analysis of pedigree data. *Hum. Hered*., **21**, 523–542.

Gudbjartsson,D.F. *et al*. (2000) Allegro, a new computer program for multipoint linkage analysis. *Nat. Genet*., **25**, 12–13.

Gusfield,D. (2001) Inference of haplotypes from samples of diploid populations: complexity and algorithms. *J. Comput. Biol*., **8**, 305–323.

Halperin,E. and Eskin,E. (2004) Haplotype reconstruction from genotype data using imperfect phylogeny. *Bioinformatics*, **20**, 1842–1849.

Hodge,S. *et al*. (1999) Loss of information due to ambiguous haplotyping of snps. *Nat. Genet*., **21**, 360–361.

Kruglyak,L. *et al*. (1995) Parametric and nonparametric linkage analysis: a unified multipoint approach. *Am. J. Hum. Genet*., **58**, 1347–1363.

Lander,E. and Green,P. (1987) Construction of multilocus genetic linkage maps in human. *Proc. Natl Acad. Sci. USA*, **84**, 2363–2367.

Lathrop,G. M. *et al*. (1984) Strategies for multilocus linkage analysis in humans. *Proc. Natl Acad. Sci. USA*, **81**, 3443–3446.

Leykin,I. *et al*. (2005) Comparative linkage analysis and visualization of high-density oligonucleotide snp array data. *BMC Genet*., **6**, 7.

Li,J. and Jiang,T. (2003) Efficient inference of haplotypes from genotypes on a pedigree. *J. Bioinform. Comput. Bio*., **1**, 41–69.

Ott,J. (1999) Aspects of statistical inference. In *Analysis of Human Genetic Linkage*. 3rd edn. Johns Hopkins University Press, Baltimore.

Qian,D. and Beckmann,L. (2002) Minimum recombinant haplotyping in pedigrees. *Am. J. Hum. Genet*., **70**, 1434–1445.

Sellick,G. *et al*. (2004) Genomewide linkage searches for mendelian disease loci can be efficiently conducted using high-density snp genotyping arrays. *Nucleic Acids Res*., **32**, e164.

Stephens,M. and Scheet,P. (2005) Accounting for decay of linkage disequilibrium in haplotype inference and missing-data imputation. *Am. J. Hum. Genet*., **76**, 449–462.

Wijsman,E. M. (1987) A deductive method of haplotype analysis in pedigrees. *Am. J. Hum. Genet*., **41**, 356–373.

Wirtenberger,M. *et al*. (2005) SNP microarray analysis for genome-wide detection of crossover regions. *Hum. Genet*., **117**, 389–397.

Zhang,K. *et al*. (2005) Haplore: a program for haplotype reconstruction in general pedigrees without recombination. *Bioinformatics*, **21**, 90–103.

Zhao,H. *et al*. (1995) Statistical analysis of crossover interference using the chi-square model. *Genetics*, **139**, 1045–1056.