Journal of Bioinformatics and Computational Biology
Vol. 11, No. 2 (2013) 1350002 (19 pages)

DOI: 10.1142/S0219720013500029

Imperial College Press
www.icpress.co.uk

WHOLE GENOME IDENTITY-BY-DESCENT DETERMINATION

HADI SABAA[*,‡], ZHIPENG CAI[*,§,‡‡], YINING WANG[*,¶],
RANDY GOEBEL[*,‖], STEPHEN MOORE[†,**,§§] and GUOHUI LIN[*,††]

[*]*Department of Computing Science, University of Alberta*
Edmonton, Alberta T6G 2E8, Canada

[†]*Department of Agriculture, Food, and Nutritional Science*
University of Alberta, Edmonton, Alberta T6G 2P5, Canada

[‡]*sabaa@ualberta.ca*
[§]*zcai@ualberta.ca*
[¶]*yining@ualberta.ca*
[‖]*rgoebel@ualberta.ca*
[**]*stephen.moore@ualberta.ca*
[††]*guohui@ualberta.ca*

Received 13 February 2012
Revised 14 September 2012
Accepted 24 November 2012
Published 14 January 2013

High-throughput single nucleotide polymorphism genotyping assays conveniently produce genotype data for genome-wide genetic linkage and association studies. For pedigree datasets, the unphased genotype data is used to infer the haplotypes for individuals, according to Mendelian inheritance rules. Linkage studies can then locate putative chromosomal regions based on the haplotype allele sharing among the pedigree members and their disease status. Most existing haplotyping programs require rather strict pedigree structures and return a single inferred solution for downstream analysis. In this research, we relax the pedigree structure to contain ungenotyped founders and present a cubic time whole genome haplotyping algorithm to minimize the number of zero-recombination haplotype blocks. With or without explicitly enumerating all the haplotyping solutions, the algorithm determines all distinct haplotype allele identity-by-descent (IBD) sharings among the pedigree members, in linear time in the total number of haplotyping solutions. Our algorithm is implemented as a computer program *i*BDD. Extensive simulation experiments using 2 sets of 16 pedigree structures from previous studies showed that, in general, there are trillions of haplotyping solutions, but only up to a few

[††]Corresponding author.

[‡‡]Current address: Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA.

[§§]Current address: Centre for Animal Science, Queensland Alliance for Agriculture & Food Innovation, University of Queensland. Brisbane St Lucia, QLD 4072, Australia.

thousand distinct haplotype allele IBD sharings. *i*BDD is able to return all these sharings for downstream genome-wide linkage and association studies.

Keywords: Haplotyping; identical by descent; haplotype allele sharing; linkage analysis.

1. Background

A *single nucleotide polymorphism* (SNP) is a DNA sequence variation occurring when a single nucleotide in the genome differs between individuals, or between paired homologous chromosomes in an individual of a diploid species (e.g. humans). SNPs have been used as genetic markers in linkage analysis and association studies, where the sharing status of the alleles among members is used to draw inferences on the inheritable properties. In this research, we have developed a program called *i*BDD, which determines all distinct haplotype allele identity-by-descent (IBD) sharings *in one whole genome scan*, for the most complex pedigree genotype datasets.

SNPs are believed to contribute to the most genetic variations in human populations.[1] The rapid development of genotyping technology has led to the identification of thousands to millions of SNPs for various species; for humans, these *common variants* provide the foundation for *genome-wide association studies* (GWAS) under the *common disease−common variant* (CDCV) hypothesis. Recently, GWAS have achieved a great deal of success,[2] but genetic fine-mapping for complex diseases such as cancer and mental illness is still a great challenge.

Nevertheless, the unphased genotype data is recognized as a fundamental bottleneck in general genetic linkage and association studies, particularly for rare diseases. For diploid species, including humans, at each biallelic SNP locus, the unphased genotype data contains two alleles (nucleotides), without specifying their parental origins. A haplotype is a *phased* genotype which, at each SNP locus, contains the allele of the same parental origin.

In the mapping of disease-susceptible genes in genetic linkage and association studies, one important assumption is that such disease-susceptible genes are in linkage disequilibrium to certain SNPs, so that these SNP markers can be the anchors of disease-susceptible genes. Given its biallelic nature, genetic linkage and association studies based on SNP genotype in general requires a large number of samples, positive samples in particular, so that the association study results are statistically significant.[2] On the other hand, if haplotypes for a large nonrecombinant chromosomal region can be determined for all the samples, then the disease-susceptible haplotype alleles may be easily identified, since the haplotype allele sharing can be better determined. This approach has been particularly successful for simple Mendelian disease genetic linkage on pedigree data. For general population-based GWAS, haplotypes clearly contain more inheritance information than unphased genotype, and the ideal case is to build the dense SNP haplotype map: this provides more detailed and deterministic haplotype allelic information for the mapping of disease-susceptible genes. It should be emphasized that, as SNPs act as

anchors, it is the allele sharing that is used for linkage inference, and thousands to millions of haplotype allele configurations provide the identical sharing. Our *i*BDD program from this research is designed to determine all distinct haplotype allele sharings.

Haplotypes can be experimentally determined, but it is very expensive to do so.[3] In practice, a less costly alternative is to collect genotype data. Therefore, efficient and accurate computational methods for the inference of haplotypes from genotype data are of considerable value. There is a rich and growing literature on haplotype inference from genotype data, also commonly referred to as *phasing* or *haplotyping*. Research that focuses on unrelated individuals — population data — is reviewed in Refs. 4 and 5, with its recent representative fastPHASE by Scheet and Stephens[6]; Research on related individuals — pedigree data — is reviewed in Ref. 7, including (exact and approximate) likelihood-based methods[8–16] and genetic rule-based strategies,[17–22] with its representative PedPhase by Li and Jiang.[17] The likelihood-based methods usually work for low-density SNP data but not high-density data; neither can they handle large (in many cases, even moderately large) datasets because of the extensive computations required. Additional information and assumptions, such as Hardy−Weinberg equilibrium and marker recombination rates, are generally required to calculate the likelihoods.

Rule-based methods for haplotyping exploit the Mendelian laws of inheritance to minimize the total number of *crossover events* (also called *recombination events* or *breakpoints*) in all pedigree members[7] needed for explaining the observed genotype data. They generally run faster than likelihood-based methods. Nevertheless, this computational minimization problem is NP-hard[17] in general, indicating that it is unlikely that there is a fast algorithm that reconstructs such optimal haplotyping solutions (also called *configurations*). When no recombination events are allowed and the pedigree structure is full (i.e. every nonfounder pedigree member has both parents genotyped), the problem is then to infer zero-recombination haplotypes from the given pedigree genotypes, which is called the *zero-recombination haplotype configuration* (ZRHC) problem. ZRHC turns out to be polynomial time solvable. Li and Jiang[17] presented an $O(m^3n^3)$ time algorithm and a computer program PedPhase which solves ZRHC by reducing the problem to solving a system of linear equations over the cyclic group Z_2, where m is the number of loci and n is the number of members in the pedigree. Doan and Evans[23] present another $O(2^{m^2}m^3n^2)$ time algorithm for ZRHC, a consequence of a fixed-parameter tractable algorithm for the general minimization problem.

It should be noted that the PedPhase and related rule-based methods, as well as those based on likelihood (e.g. PhyloPed[16]), require (either specifically or *conventionally*) the full pedigree structure — any nonfounder pedigree member must have both parents genotyped. As an exception, *i*Linker is rule-based and does not require a full pedigree, but the pedigree can have only a couple or a single founder.[20] In summary, most existing rule-based haplotyping methods have rather strict pedigree structure requirements, while a practical pedigree often contains multiple founders,

some of them may have passed away and their genotype data can no longer be collected. For instance, none of the pedigrees used in the seven previous case studies[20,24–29] is full.

In our research, we relax the past pedigree structure requirement in our haplotyping algorithm to allow for ungenotyped founders, as long as the pedigree stays connected and every ungenotyped member appears in exactly one nuclear family. We note that, while we deal with ungenotyped pedigree members, every genotyped member must have complete genotype data. There is existing work that deals with missing genotype data in genotyped members.[30] (When there is a large portion of missing data in a population dataset, there is another line of work that does haplotyping and imputation using the inferred haplotypes[6,31–33]). Under the zero-recombination assumption, our novel rule-based haplotyping algorithm has thus the largest pedigree coverage; it can produce all haplotyping solutions in $O(m^3n^3)$ time, where m is the number of loci and n is the number of (genotyped) members in the pedigree; and it is extended into a whole genome haplotyping algorithm to minimize the number of zero-recombination chromosomal regions (i.e. haplotype blocks). Moreover, note that genetic linkage and association studies use allele sharing information amongst the individuals to make inferences, but not the detailed haplotype alleles of each individual.[20,34–38] We therefore take advantage of all the haplotyping solutions (produced implicitly or explicitly) to determine all distinct genome-wide haplotype allele identity-by-descent (IBD) and *identity-by-state* (IBS)[34] sharings among all pedigree members, together with their associated numbers of haplotyping solutions. It is important to point out that the use of sharing properties of haplotypes adequately addresses the issue of haplotype ambiguity (particularly, the founder haplotype ambiguity) in linkage analysis and haplotype-based association studies. All these functions are coded in a Perl program *i*BDD, which is available upon request.

2. Materials and Methods

A *pedigree* describes the parent−offspring relationship among individuals. In our discussion of a pedigree, members are present only if they are genotyped. Those members who have no parents are the *founders* of the pedigree. A *nuclear family* in a pedigree consists of the parent(s) and all the children. In this paper, connected complex pedigrees are considered, in which a nonpresent parent of a nuclear family is an ungenotyped founder. Our pedigrees are *connected*, such that each nuclear family has at least one parent and an ungenotyped founder appears in exactly one nuclear family. The genotype at a (biallelic) SNP locus is *homozygous* if the two alleles are the same (i.e. AA or BB), or is *heterozygous* if the two alleles are different (i.e. AB).

Our haplotyping algorithm is based on the Mendelian laws of inheritance, which states that, at each locus of a pair of homologous autosomes of a child, one allele is inherited from her father (the *paternal* allele) and the other from mother (the

maternal allele). A child does not inherit a complete autosome from each parent, since *crossover* (also called *recombination*) events occur. That is, during the meiosis process, the two homologous autosomes of a parent may be shuffled and four chromatids are generated, each of which is a shuffled copy of the two homologous autosomes of the parent. One of these chromatids is passed on to the child. Between any two consecutive SNP loci along an autosome of the parent, if recombination occurs, then there is a *breakpoint site* between these two loci; this creates a *breakpoint* between the two loci on the corresponding chromatid passed on to the child.

2.1. *Zero-recombination haplotyping*

Given a pedigree and the unphased genotype data of its members, Li and Jiang[17] presented a constraint-based haplotyping algorithm, PedPhase, under the recombination-free assumption. PedPhase defines a binary parental source (PS) variable z_i for each child z at every SNP locus i, to record the parental origin of the two alleles: $z_i = 1$ if and only if SNP i is heterozygous and child z inherits allele A from her mother. Adopting the cyclic group rule "$1 + 1 = 0$," PedPhase produces a system of linear equations as haplotyping constraints, where subsets of equations are collected trio by trio. For example, let x denote the mother, y denote the father, and z be their child; let i and j denote two consecutive SNP loci at which x is heterozygous (i.e. x is homozygous at loci $i + 1, i + 2, \ldots, j - 1$); assume z is heterozygous as well at these two loci (but z is not necessarily heterozygous at any loci of $i + 1, i + 2, \ldots, j - 1$). Depending on the genotype of y at these two loci, PedPhase writes down the following set of constraints:

$$\begin{cases} \text{either } x_i = x_j & \text{or} \quad x_i + x_j = 1, \\ \text{either } y_i = y_j & \text{or} \quad y_i + y_j = 1, \\ x_i + x_j = z_i + z_j & \text{and/or} \quad y_i + y_j = z_i + z_j. \end{cases}$$

It has been proven that all the linear equations are satisfied if and only if the extracted haplotyping solution conforms with the input genotype data. Because the equations are written down for trios, PedPhase requires a full pedigree structure i.e. every nonfounder member must have both parents genotyped.

We rewrite these haplotyping constraints on a trio (x, y, z) to separate them into two independent subsets of constraints, for the two parent−child pairs (x, z) and (y, z) respectively. For example, for the pair (x, z), only one linear constraint is written down, disregarding what the genotype y has at loci i and j:

$$x_i + x_j + z_i + z_j = 0.$$

In this way, we relax the pedigree structure to allow for one ungenotyped parent per nuclear family — this factoring provides the advantage of the constraint rewriting scheme. That is, in the above case, parent y can be ungenotyped. Furthermore, if indeed y is ungenotyped, then besides the equations for all (x, child) pairs, additional equations can be expressed to constrain the number of haplotypes for y to be at most

two. It should be noted that in the case where the pedigree structure is full, our rewriting scheme becomes exactly the same as the PedPhase original scheme. Similarly, it can be shown that these constraints in the rewriting scheme are all satisfied if and only if the corresponding haplotyping solution is feasible. For the detailed rewriting scheme that handles dozens of distinct scenarios and its mathematical proof of correctness, interested readers may refer to Li and Jiang[17] and Cheng *et al.*[21] Moreover, the total number of linear equations in our new system is proven to be no more than 7 mn, resulting in an $O(m^3n^3)$-time haplotyping algorithm, where m is the total number of SNPs along the chromosome and n is the size of the pedigree.

2.2. *Parsimonious whole genome haplotyping*

For small chromosomal regions, the zero-recombination assumption can be reasonable and the genetic linkage and association studies based on zero-recombination haplotyping solutions are meaningful. For genotype datasets on whole chromosomes, we need to relax the zero-recombination assumption but take advantage of the zero-recombination haplotyping algorithm to greedily and optimally determine the maximal zero-recombination haplotype blocks, and their associated haplotyping solutions.

The following outlines at a high level on how this is accomplished in *i*BDD. Starting with $i = 2$, the zero-recombination haplotyping algorithm is run to check whether solutions exist for chromosomal region $[1, i]$, which contains loci $1, 2, \ldots, i$. If affirmative, i is incremented by 1; and the checking process is repeated until at some point, there is no solution to the linear system written for chromosomal region $[1, i]$. This gives a maximal zero-recombination chromosomal region $[1, i-1]$, and all haplotyping solutions are recovered for the region (either implicitly by storing those determined PS variables or explicitly by listing all haplotype configurations). Next, chromosomal region $[1, i-1]$ is chopped, and the haplotyping process moves on to repeatedly determine the other maximal zero-recombination chromosomal regions starting SNP locus i.

Note that in this incremental haplotyping process, haplotyping solutions for chromosomal region $[1, j]$ are fully used in the zero-recombination haplotyping algorithm to compute the haplotyping solutions for chromosomal region $[1, j+1]$. Our implementation ensures that the total number of linear equations is confined in $O(mn)$, to guarantee the $O(m^3n^3)$ running time for the whole genome scan.

2.3. *The identity-by-descent determination*

Given an explicit haplotyping solution for a zero-recombination haplotype block, or given an implicit haplotyping solution represented as a solution to the linear system (i.e. the PS values), we can use genotype data to trace the inheritance of each haplotype allele of a genotyped founder member within the entire block. Essentially, we identify for each child's two haplotype alleles their parental source. After this is

done, all the pedigree descendants who share the same founder haplotype allele form a cluster. The haplotype allele identical-by-descent (IBD) sharing is characterized as a collection of such clusters labeled with the founder identity. Note that this process does not require (and thus allows the user to choose to not create) explicit haplotyping solutions. The process runs in $O(kn)$ time, where k is the total number of maximal zero-recombination haplotype blocks.

3. Results

For our experiments, we use a real dataset from Wirtenberger *et al.*[39] to generate simulation datasets. This real dataset contains 877 SNPs on chromosome 1 for independent individuals genotyped by GeneChip Human Mapping 10 K Xba array. To demonstrate the performance of our *i*BDD, six real pedigree structures are employed, which have been used in previous genetic linkage and association studies (Fig. 1 in Lin *et al.*,[20] Fig. 1 in Martin *et al.*,[25] Fig. 2 in Sinsheimer *et al.*,[29] Fig. 2 in Lin *et al.*,[24] Fig. 1 in Hauser *et al.*,[27] and Fig. 1 in Howell *et al.*,[28] respectively). These pedigrees vary wildly in size, the number of nuclear families, the number of founders, and the number of ungenotyped founders. A summary of their structural characteristics is included in Table 1. We followed a trio simulation process which generates two whole chromosomal haplotypes for a child from her parents' haplotypes according to the $\chi^2(m)$-model for crossover events with $m = 4$.[40–42] Using this trio generation process, the pedigree genotype datasets are simulated, 100 datasets for each pedigree.

3.1. *Breakpoint recovery*

One important performance measure of a haplotyping algorithm is whether a true/simulated breakpoint can be correctly recovered. In the pedigree genotype dataset simulation process, a simulated parental breakpoint site could arise in between two consecutive homozygous SNP loci, thus it cannot be precisely recovered by any computational approach. We adopted the criteria mentioned in previous work[16,20,42] to do the mapping between the simulated breakpoint sites and the breakpoint sites computed by *i*BDD: a simulated breakpoint site is classified as *recovered* if there is a computed breakpoint site on the same parent such that the SNPs in between these two locations, if any, are all homozygous; otherwise, the breakpoint is classified as *missed*.

Table 1. Properties of the six pedigrees we used in the simulation study.

Pedigree No.	1	2	3	4	5	6
#Members	16	19	17	24	10	20
#Generations	3	3	3	5	3	3
#Nuclear families	4	3	4	12	2	4
#Founders	5	4	5	9	3	5
#Ungenotyped	3	2	2	3	1	4

Table 2. The mean breakpoint recovery precision and recall by *i*BDD on each of the six pedigrees, over the averages of 100 simulated datasets, and their standard deviations.

Pedigree No.	Precision	Recall
1	0.893 ± 0.053	0.748 ± 0.066
2	0.684 ± 0.064	0.879 ± 0.061
3	0.862 ± 0.057	0.793 ± 0.060
4	0.915 ± 0.043	0.683 ± 0.053
5	0.729 ± 0.082	0.886 ± 0.065
6	0.678 ± 0.075	0.751 ± 0.074
Average	0.793	0.790

The breakpoint recovery *precision* is defined as the ratio of the number of simulated breakpoint sites that are recovered (true positives) over the number of breakpoint sites computed by *i*BDD (true and false positives). The breakpoint recovery *recall* is defined as the ratio of the number of true positives over the total number of simulated breakpoint sites. We calculated the precision and recall on each simulated dataset as the averages over all the haplotyping solutions. The mean precision and recall values over all 100 datasets associated with the pedigree are reported in Table 2. The general conclusion on breakpoint recovery is that *i*BDD generates slightly fewer breakpoints than the simulated truth, achieving average precision of 79.3%, to recover most of the true breakpoints (average recall 79.0%). Figure 1 plots the mean recall (y-axis) versus precision (x-axis) values on the 100 datasets simulated for pedigree No. 1. One can see from this plot and Table 2 that the breakpoint recovery is quite stable across the 100 datasets for each pedigree, suggesting its variation depends mostly on the pedigree structure.

3.2. *Haplotype allele sharing recovery*

We use the F-score from information retrieval to measure the accuracy of haplotype allele sharing recovery, compared with the simulated sharing. More specifically, we determine all the zero-recombination chromosomal regions for the simulated haplotype configuration. These simulated zero-recombination chromosomal regions intersect with the *i*BDD zero-recombination chromosomal regions to produce a set of *common* zero-recombination chromosomal regions to the simulated sharing and the *i*BDD sharing. Note that on each of these zero-recombination chromosomal regions, the simulated sharing is also a collection of pedigree member clusters labeled with the founder identity. **We first arbitrarily name the two clusters associated with the same founder *paternal* and *maternal***, respectively. The two clusters associated with the same founder in the *i*BDD sharing are accordingly named, *paternal* and *maternal*, respectively, such that the F-score of the mapping between the simulated and the *i*BDD sharings is maximized (over two possible choices). Upon completing all the founders, the (weighted) F-score for this chromosomal region is

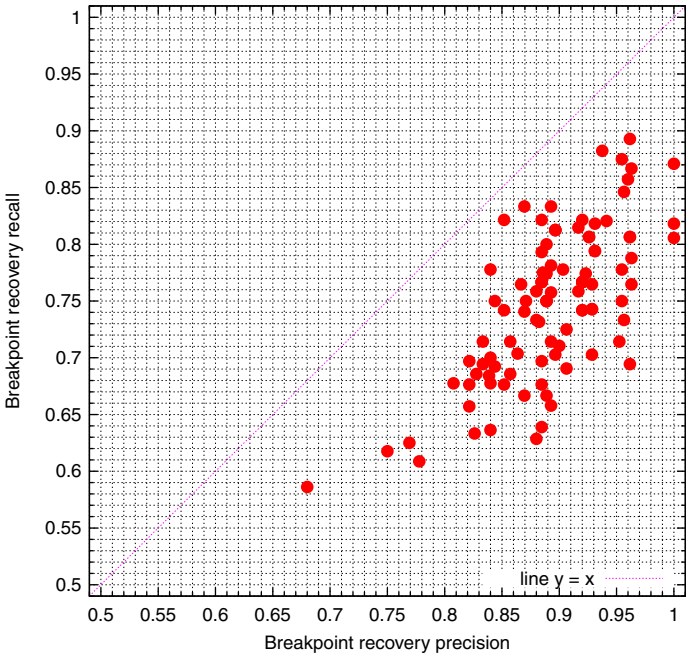


Fig. 1. The mean recall versus precision values on the 100 datasets simulated for pedigree No. 1.

computed between the two collections of labeled clusters. The chromosomal F-score between the simulated and the *i*BDD haplotype allele IBD sharings is taken as the weighted average F-score over all maximal zero-recombination chromosomal regions, where the weight of a region is taken as the number of SNPs in that region.

In addition to the haplotype allele IBD sharing, we have also collected all distinct haplotype allele IBS sharings produced by *i*BDD, and calculated the F-score between the simulated haplotype allele IBS sharing and each *i*BDD haplotype allele IBS sharing. That is, on each simulated dataset, we calculated the F-score of each IBD (IBS, respectively) sharing against the simulated IBD (IBS, respectively) sharing to demonstrate the performance of haplotype allele sharing recovery by *i*BDD. The average F-score over all distinct IBD (IBS, respectively) sharings by *i*BDD against the simulated IBD (IBS, respectively) sharing, across all 100 datasets simulated for each of the six pedigrees, is reported in Table 3. Note that given a haplotyping solution, the associated haplotype allele IBD sharing is a refinement of the associated haplotype allele IBS sharing. The number of distinct haplotype allele IBS sharings produced by *i*BDD is about four times the number of distinct haplotype allele IBD sharings, for all the 600 datasets in our simulation studies. Figure 2 plots the average recovery F-scores (i.e. against simulation) on the 100 datasets simulated for pedigree No. 1. One can see from Fig. 2 and Table 3 that most of the haplotype allele sharings produced by *i*BDD are close to the simulated truth, yet some could be a bit far away.

Table 3. The mean haplotype allele sharing F-scores by *i*BDD against the simulated ones on each of the six pedigrees, over the averages of 100 simulated datasets, and their standard deviations.

	F-score	
Pedigree No.	IBD	IBS
1	0.978 ± 0.005	0.996 ± 0.002
2	0.968 ± 0.007	0.998 ± 0.001
3	0.980 ± 0.003	0.996 ± 0.002
4	0.984 ± 0.003	0.999 ± 0.001
5	0.986 ± 0.003	0.998 ± 0.002
6	0.975 ± 0.004	0.996 ± 0.002
Average	0.978	0.997

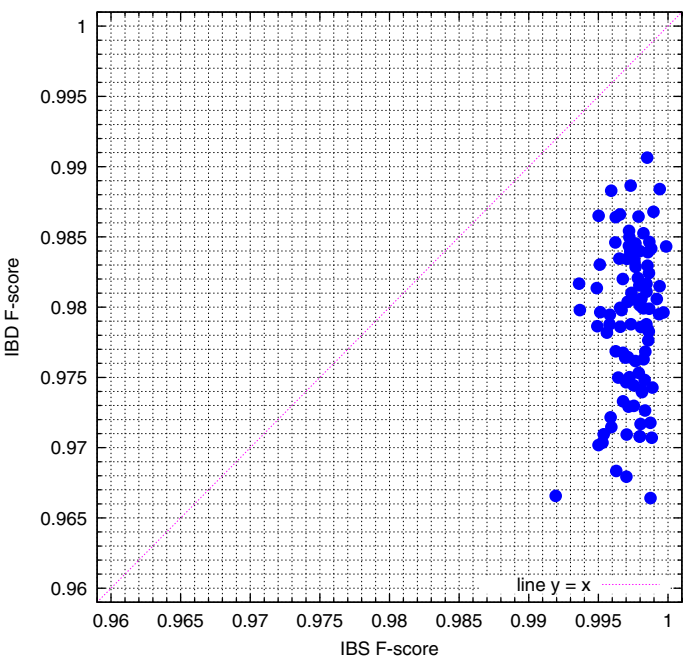


Fig. 2. The mean F-scores of the haplotype allele IBD and IBS sharings compared with the simulated sharings, respectively, on the 100 datasets simulated for pedigree No. 1.

This suggests that using only one sharing in linkage and association studies might not work well, as the sharing could significantly deviate from the truth.

4. Discussion

Guaranteed by its worst-case cubic running time, *i*BDD runs fast to produce all haplotyping solutions, from which it takes only a fraction of time to determine all the

distinct haplotype allele IBD and IBS sharings together with their associated numbers of haplotyping solutions. Among the six pedigrees used in our study, pedigree No. 1 is moderately complex (Table 1). *i*BDD was able to terminate in about two minutes on average, using an Intel E6850 3.0 GHz processor with 4 GB RAM.

4.1. *The huge number of haplotyping solutions versus a handful distinct haplotype allele sharings*

Previously, the uncertainty in the inferred multiple haplotyping solutions was conjectured to be a potential problem for haplotype-based association studies. By using the haplotype allele sharing status, we believe that haplotype ambiguities in multiple haplotyping solutions can be eliminated.

From the simulation study, we observed that for each simulated dataset, the number of haplotyping solutions by *i*BDD is always large, in the trillions. Indeed, for a pedigree of n nonfounders and a total number of H heterozygous loci across all genomes of all individuals, there could be 2^{H-n} haplotyping solutions. Nevertheless, there are only a few to dozens of, or up to a few thousand distinct haplotype allele IBD sharings (i.e. up to 2^n), each associated with thousands to millions of haplotyping solutions. For example, across the 100 simulated datasets for pedigree No. 1 (plotted in Fig. 3) the average number of distinct haplotype allele IBS sharings is

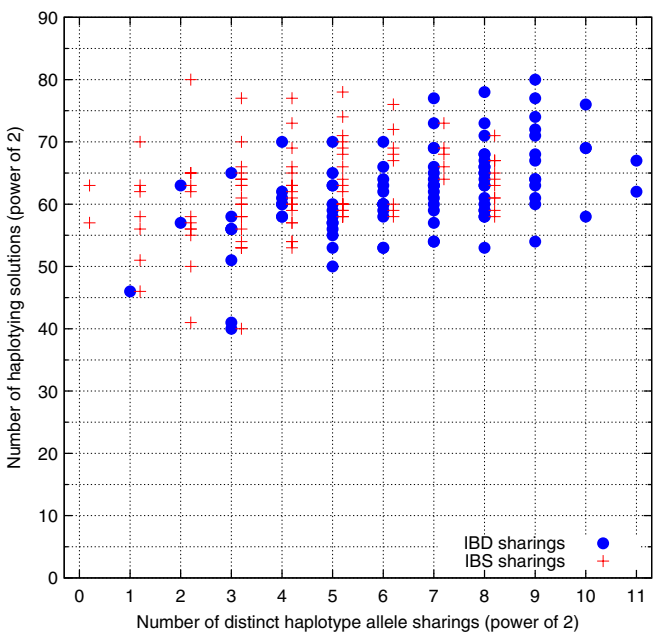


Fig. 3. The numbers of haplotyping solutions, distinct haplotype allele IBD and IBS sharings on the 100 datasets simulated for pedigree No. 1. It is interesting to note that 2 datasets have a unique IBS sharing — 2 crosses in column 0.

48.24, the average number of distinct haplotype allele IBD sharings is 235.78, and their associated numbers of haplotyping solutions are in the range of 2^{60}'s.

The observation suggests that using only one or a few inferred haplotyping solutions in genetic linkage and association studies[16,20,26] is not appropriate. Given that trying all haplotyping solutions is computationally prohibitive, our *i*BDD becomes helpful in feasibly enumerating all distinct haplotype allele IBD (or IBS) sharings.

4.2. *Comparison to existing haplotyping methods*

Our *i*BDD is derived from PedPhase,[17] and therefore it is reasonable to make comparison to PedPhase. Unfortunately, since PedPhase requires full pedigree structure (i.e. every nonfounder member must have both parents genotyped), it does not run on any of the six pedigrees we used earlier. Moreover, PedPhase is a zero-recombination haplotyping algorithm only, which has to be extended into a general haplotyping algorithm for comparison purposes. We thus adopted the parsimony rule in the same way as in *i*BDD to extend it to xPedPhase, which determines all maximal zero-recombination chromosomal regions together with all haplotyping solutions. We also made a comparison to *i*Linker,[20] another rule-based haplotyping algorithm which returns only one haplotyping solution. For fair comparison to *i*Linker, we adopted the scheme to use the first haplotyping solutions by *i*BDD and xPedPhase, respectively, and made comparison among these three using breakpoint recovery and haplotype allele IBD sharing recovery.

Note that *i*Linker accepts only pedigrees with a couple founders or a single founder, while xPedPhase accepts only full pedigrees. We therefore used another set of 10 pedigrees in this comparative study, summarized in Table 4. This, however, makes the direct comparison between *i*Linker and xPedPhase impossible; so the only comparison is via *i*BDD. That is, these 10 pedigrees all have a couple founders, and thus we can run both *i*Linker and *i*BDD. For each of them, missing founders are added to make the pedigree full, and subsequently, simulation datasets are generated for running xPedPhase and *i*BDD. Again, 100 full datasets are simulated for each full pedigree, from which the genotype data for the missing founders are dropped to form the nonfull datasets for the corresponding nonfull pedigree. The average breakpoint precision and recall and haplotype allele IBD sharing F-score are collected, for

Table 4. Properties of pedigrees used to compare the performance of xPedPhase, *i*Linker, and *i*BDD.

Pedigree No.	1	2	3	4	5	6	7	8	9	10
#Members	4	5	7	9	10	13	11	13	15	16
#Generations	2	2	3	3	3	3	3	3	3	3
#Nuclear families	1	1	2	3	3	4	3	3	4	4
#Founders	2	2	3	4	4	5	4	4	5	5
#Ungenotyped	0	0	1	2	2	3	2	2	3	3

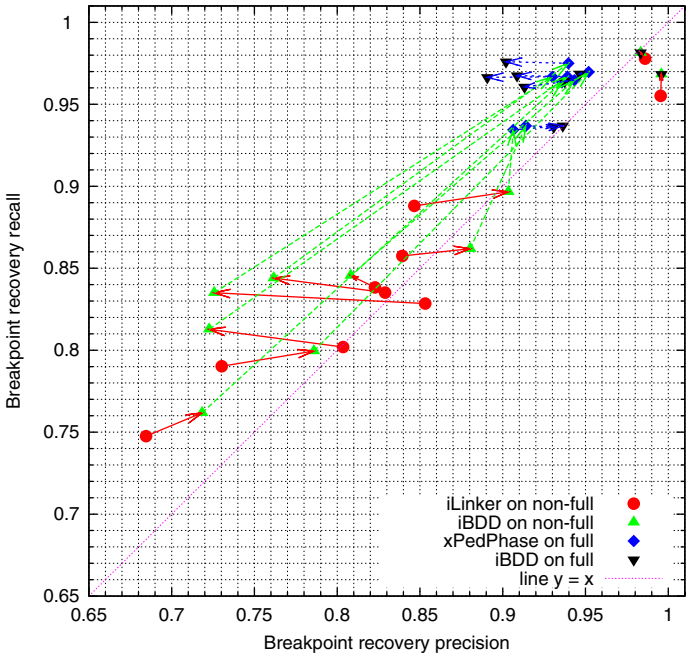


Fig. 4. The mean recall versus precision values of breakpoint recovery for *i*Linker (red dots) and *i*BDD (greed triangles) on 10 non-full pedigrees, and xPedPhase (blue triangles) and *i*BDD (black diamonds) on the corresponding full pedigrees, respectively; each pedigree is associated with 100 simulated genotype datasets.

xPedPhase on the full datasets only, for *i*BDD on both the full datasets and the nonfull datasets, and for *i*Linker on the nonfull datasets only.

Figure 4 plots the average recall against average precision on the 10 pedigrees. In the plot, there are four dots associated with each pedigree, except for the first two pedigrees on which xPedPhase regularly failed. These four dots correspond to the four pairs of (precision, recall) by *i*Linker on the nonfull datasets, by *i*BDD on the nonfull datasets, by xPedPhase on the full datasets, and by *i*BDD on the full datasets. They are connected sequentially using red, blue, and green arrows. From these arrows, one can see the general tendency of improved performance. In terms of breakpoint recovery, *i*BDD performance was not significantly different from xPedPhase on the full datasets — slightly better recall versus slightly worse precision, *i*BDD performed better than *i*Linker on the nonfull datasets, and nonsurprisingly, breakpoint recoveries on the full datasets are better than on the corresponding nonfull datasets. Note that on the full datasets, *i*BDD and xPedPhase are expected to perform no differently, since they essentially use the same haplotyping algorithm; yet the slight difference we have seen is perhaps due to different implementation.

Besides rule-based haplotyping methods, we have also tried to make comparisons with likelihood-based haplotyping algorithms, including Haploview,[43] fastPHASE,[6]

and SuperLink.[15] Unfortunately, for the most recent versions of all three programs, Haploview and SuperLink did not produce the complete haplotype for all pedigree members, while fastPHASE did not seem to follow the pedigree structure, resulting in haplotyping solutions that break the Mendelian rules of inheritance. Consequently we do not have results on this comparison to present here.

4.3. *Possible reasons for low breakpoint recovery*

Section 3 shows that the *i*BDD haplotype allele sharing recovery is almost perfect, but the breakpoint recovery is not. Figure 4 shows that for each pair of a full and the corresponding nonfull pedigrees, *i*BDD performed much better on the datasets associated with the full pedigree; the difference on average is 0.1324 (or 16.0%) in precision and 0.1274 (or 14.8%) in recall. This suggests that the nonfull pedigree structure seems a major reason for low breakpoint recovery. Theoretically, in the face of nonfull pedigree structure, where some founder members are not genotyped, the constraints derived from them are only a small subset (less than 50%) of the constraints derived from that member if the member is genotyped. Though there are no parental source (PS) variables defined for ungenotyped founders, this could lead to a much larger solution space compared to the solution space associated with the corresponding full pedigree; furthermore, many of these additional solutions may be farther away from the true haplotyping solution.

4.4. *High haplotype allele sharing recovery*

While pedigree structure has a major impact on breakpoint recovery, fortunately, as we showed in Sec. 3, it has relatively minor effects to the haplotype allele sharing. Our comparative study shows that the IBD/IBS sharing recovery difference is only 0.0069/0.0020, or 0.7%/0.2%. One possible reason is that the haplotype allele sharing is the same for many distinct haplotyping solutions, and consequently more robust against the uncertainties arising from phase inference. This phenomenon also suggests that the use of this more robust haplotype allele sharing directly in genetic linkage and association studies may resolve the issue of phase ambiguities, which has been previously observed.[8,10,16,35,44,45]

4.5. *Use of haplotype allele sharing in genetic linkage*

Our simulation study shows that for each simulated dataset, trillions of haplotyping solutions give rise to only a few close, though distinct, haplotype allele IBD (and IBS) sharings, most of which are very close to the simulation sharing.

We have access to the real genotype data associated with pedigree No. 1, which was used for the linkage analysis of a family disease[20] (data not to be released). We ran *i*BDD on this dataset for all haplotyping solutions and for all distinct haplotype allele IBD sharings. Since member M is the diseased founder, we are interested in the two clusters of pedigree members who share a haplotype allele with M. There are

4096 distinct haplotype allele IBD sharings found (for 2^{58} haplotyping solutions). These sharings differ slightly toward the tail of the chromosome, and they all point to two chromosomal regions — 19.62−20.87 Mbps (7 SNPs) and 22.48−29.57 Mbps (68 SNPs) — that are shared exclusively by *all nine diseased members*, including the diseased founder M. Surprisingly, the LOD score approach[13] does not detect significant linkage. In this sense, the beauty of our *i*BDD program — its ability to determine the explicit sharing of each haplotype allele — is confirmed, while the traditional LOD score approaches for linkage analysis is not really effective, in particular when the pedigree is small.

5. Conclusions

We presented an efficient $O(m^3n^3)$-time whole genome haplotyping algorithm for a pedigree genotype dataset to minimize the number of zero-recombination haplotype blocks, where n is the size of the pedigree and m is the number of SNPs, and the pedigree can contain ungenotyped founders. With or without explicitly enumerating all the haplotyping solutions, our *i*BDD program determines all distinct haplotype allele IBD sharings among the pedigree members. Extensive simulation experiments supported that *i*BDD is able to return all these sharings for downstream genome-wide linkage and association studies. It would be interesting to develop a similar score as the LOD score in Merlin, such that the score incorporates all distinct haplotype allele sharings and their associated number of haplotype configurations. The new score landscape will be more useful in linkage analysis.

Acknowledgments

This research is supported in part by AARI, AICML, ALIDF, iCORE, and NSERC.

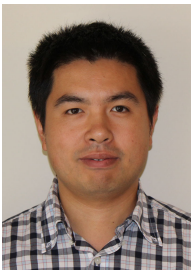
References

1. International Human Genome Sequencing Consortium, Initial sequencing and analysis of the human genome, *Nature* **409**:860−921, 2001.
2. Altshuler D, Daly MJ, Lander ES, Genetic mapping in human disease, *Science* **322**:881−888, 2008.
3. Lin S, Chakravarti A, Cutler DJ, Haplotype and missing data inference in nuclear families, *Genome Res* **14**:1624−1632, 2004.
4. Niu T, Algorithms for inferring haplotypes, *Genet Epidemiol* **27**:334−347, 2004.
5. Salem RM, Wessel J, Schork NJ, A comprehensive literature review of haplotyping software and methods for use with unrelated individuals, *Human Genom* **2**:39−66, 2005.
6. Scheet P, Stephens M, A fast and flexible statistical model for large-scale population genotype data: Applications to inferring missing genotypes and haplotypic phase, *Am J Human Genet* **78**:629−644, 2006.
7. Gao G, Allison DB, Hoeschele I, Haplotyping methods for pedigrees, *Human Hered* **67**:248−266, 2009.
8. Lander ES, Green P, Construction of multilocus genetic linkage maps in human, *Proc Nat Acad Sci USA* **84**:2363−2367, 1987.

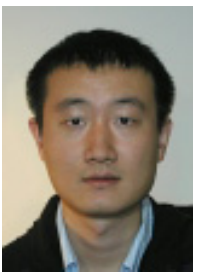
9. Sobel E, Lange K, O'Connell JR, Weeks DE, Haplotype algorithms, in Speed TP, Waterman MS (eds.), *Genetic Mapping and DNA Sequencing*, *IMA Volumes in Mathematics and Its Applications*, Springer, New York, pp. 89−110, 1995.
10. Kruglyak L, Daly MJ, Reeve-Daly MP, Lander ES, Parametric and nonparametric linkage analysis: A unified multipoint approach, *Am J Human Genet* **58**:1347−1363, 1996.
11. Lin S, Speed TP, An algorithm for haplotype analysis, *J Comput Biol* **4**:535−546, 1997.
12. Gudbjartsson DF, Jonasson K, Frigge ML, Kong A, Allegro, a new computer program for multipoint linkage analysis, *Nature Genet* **25**:12−13, 2000.
13. Abecasis GR, Cherny SS, Cookson WO, Cardon LR, Merlin — rapid analysis of dense genetic maps using sparse gene flow trees, *Nature Genet* **30**:97−101, 2002.
14. Gao G, Hoeschele I, Sorensen P, Du FX, Conditional probability methods for haplotyping in pedigrees, *Genetics* **167**:2055−2065, 2004.
15. Fishelson M, Dovgolevsky N, Geiger D, Maximum likelihood haplotyping for general pedigrees, *Human Hered* **59**:41−60, 2005.
16. Kirkpatrick B, Halperin E, Karp RM, Haplotype inference in complex pedigrees, *J Comput Biol* **17**:269−280, 2010.
17. Li J, Jiang T, Efficient rule-based haplotyping algorithms for pedigree data, *Proc 7th Ann Conf Res Comput Mol Biol (RECOMB'03)*, pp. 197−206, 2003.
18. Zhang K, Sun F, Zhao, H, Haplore: A program for haplotype reconstruction in general pedigrees without recombination, *Bioinformatics* **21**:90−103, 2005.
19. Xiao J, Liu L, Xia L, Jiang T, Fast elimination of redundant linear equations and reconstruction of recombination-free Mendelian inheritance on a pedigree, *Proc 18th Ann ACM-SIAM Symp Discrete Algorithms (SODA'07)*, pp. 655−664, 2007.
20. Lin G, Wang Z, Wang L, Lau YL, Yang W, Identification of linked regions using high-density SNP genotype data for linkage analyses, *Bioinformatics* **24**:86−93, 2008.
21. Cheng Y, Sabaa H, Cai Z, Goebel R, Lin G, Efficient haplotype inference algorithms in one whole genome scan for pedigree data with non-genotyped founders, *Acta Mathematicae Applicatae Sinica (English Series)* **25**:477−488, 2009.
22. Liu L, Jiang T, A linear-time algorithm for reconstructing zero-recombinant haplotype configuration on pedigrees without mating loops, *J. Combin Optim* **19**:217−240, 2010.
23. Doan DD, Evans PA, An FPT haplotyping algorithm on pedigrees with a small number of sites, *Algorith Mol Biol* **6**:8, 2011.
24. Lin S, Thompson E, Wijsman E, Achieving irreducibility of the Markov chain Monte Carlo method applied to pedigree data, *Math Med Biol* **10**:1−17, 1993.
25. Martin ER, Monks SA, Warren LL, Kaplan NL, A test for linkage and association in general pedigrees: The pedigree disequilibrium test, *Am J Human Genet* **67**:146−154, 2000.
26. Tapadar P, Ghosh S, Majumder PM, Haplotyping in pedigrees via a genetic algorithm, *Human Heredity*, pp. 43−56, 2000.
27. Hauser MA, Conde CB, Kowaljow V, Zeppa G, Taratuto AL, Torian UM, Vance J, Pericak-Vance MA, Speer MC, Rosa AL, Myotilin mutation found in second pedigree with LGMD1A, *Am J Human Genet* **71**:1428−1432, 2002.
28. Howell N, Smejkal CB, Mackey DA, Chinnery PF, Turnbull DM, Herrnstadt C, The pedigree rate of sequence divergence in the human mitochondrial genome: There is a difference between phylogenetic and pedigree rates, *Am J Human Genet* **72**:659−670, 2003.
29. Sinsheimer JS, Plaisier CL, Huertas-Vazquez A, Aguilar-Salinas C, Tusie-Luna T, Pajukanta P, Lange K, Estimating ethnic admixture from pedigree data, *Am J Human Genetics* **82**:748−755, 2008.

30. Li X, Li J, Efficient haplotype inference from pedigrees with missing data using linear systems with disjoint-set data structures, *Proc Seventh Comput Syst Bioinform Conf*, pp. 297−308, 2008.
31. Browning SR, Browning BL, Rapid and accurate haplotype phasing and missing-data inference for whole-genome association studies by use of localized haplotype clustering, *Am J Human Genet* **81**:1084−1097, 2007.
32. Marchini J, Howie B, Myers S, McVean G, Donnelly P, A new multipoint method for genome-wide association studies by imputation of genotypes, *Nature Genet* **39**:906−913, 2007.
33. Li Y, Willer CJ, Ding J, Scheet P, Abecasis GR, MaCH: Using sequence and genotype data to estimate haplotypes and unobserved genotypes, *Genet Epidemiol* **34**:816−834, 2010.
34. Cotterman CW, A calculus for statistical genetics, PhD thesis, Ohio State University, Columbus, 1940.
35. Guo SW, Proportion of genome shared identical by descent by relatives: Concept, computation, and applications, *Am J Human Genet* **56**:1468−1476, 1995.
36. Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA, Score tests for association between traits and haplotypes when linkage phase is ambiguous, *Am J Human Genet* **70**:425−434, 2002.
37. Tzeng JY, Devlin B, Wasserman L, Roeder K, On the identification of disease mutations by the analysis of haplotype similarity and goodness of fit, *Am J Human Genet* **72**:891−902, 2003.
38. Lin DY, Zeng D, Likelihood-based inference on haplotype effects in genetic association studies, *Am J Human Genet* **101**:89−104, 2006.
39. Wirtenberger M, Hemminki K, Chen B, Burwinkel B, SNP microarray analysis for genome-wide detection of crossover regions, *Human Genet* **117**:389−397, 2005.
40. Zhao H, Speed TP, McPeek MS, Statistical analysis of crossover interference using the chi-square model, *Genetics* **139**:1045−1056, 1995.
41. Broman KW, Weber JL, Characterization of human crossover interference, *Am J Human Genet* **66**:1911−1926, 2000.
42. Cai Z, Sabaa H, Wang Y, Goebel R, Wang Z, Xu J, Stothard P, Lin G, Most parsimonious haplotype allele sharing determination, *BMC Bioinform* **10**:115, 2009.
43. Barrett JC, Fry B, Maller J, Daly MJ, Haploview: Analysis and visualization of LD and haplotype maps, *Bioinformatics* **21**:263−265, 2005.
44. Hawley ME, Kidd KK, HAPLO: A program using the EM algorithm to estimate the frequencies of multi-site haplotypes, *J Hered* **86**:409−411, 1995.
45. Weeks DE, Sobel E, O'Connell JR, Lange K, Computer programs for multilocus haplotyping of general pedigrees, *Am J Human Genet* **56**:1506−1507, 1995.

Hadi Sabaa received his PhD degree from the Department of Computing Science at the University of Alberta. His research interests lie in bioinformatics and genomics, typically in SNP haplotyping and GWAS.



Zhipeng Cai received his BS degree in Computer Science from Beijing Institute of Technology, China, and his MS and PhD degrees in Computing Science from University of Alberta, Canada. He is currently an Assistant Professor in the Department of Computer Science at the Georgia State University, USA. His research interests include bioinformatics, wireless networks, and optimization theory. Dr. Cai received a number of awards and honors, including the NSERC Postdoc Fellowship and the PhD Outstanding Research Achievement Award. He also served as a guest editor for *Algorithmica* and *Journal of Combinatorial Optimization* and as an Editorial Board Member for *International Journal of Sensor Networks*.



Yining Wang is a PhD student in the Department of Computing Science at the University of Alberta. His research interests lie in bioinformatics and applications of machine learning to genomics.

Randy Goebel is Vice President of the innovates Centre of Research Excellence (iCORE) at Alberta Innovates Technology Futures (www.albertainnovates.ca), Chair of the Alberta Innovates Academy, Professor of Computing Science at the University of Alberta (www.cs.ualberta.ca), and Principal Investigator in the Alberta Innovates Centre for Machine Learning (www.aicml.ca). Dr. Goebel has research interests that include applications of machine learning to systems biology, visualization, and web mining, as well as work on natural language processing, web semantics, and belief revision.

Stephen Moore is a highly regarded research scientist who, prior to his appointment with QAAFI, worked for the cattle industry in Alberta, Canada. He has more than 20 years' experience in bovine genomics, including his role as Chair in Bovine Genomics at the University of Alberta since 1999. In his role as CEO, Livestock Gentec, University of Alberta, Professor Moore led many successful projects to identify genes that underlie production and quality traits in cattle. The expertise he brings to QAAFI's Centre for Animal Science reflects the centre's capacity to embark on research to help Australian and international livestock industries flourish now and into the future. Professor Moore took up his appointment as Director of the Centre For Animal Science at QAAFI in September 2011.

Guohui Lin is an Associate Professor of Computing Science at the University of Alberta. He received his PhD in Theoretical Computer Science from the Chinese Academy of Sciences in 1998. His research interests include Bioinformatics and Computational Biology, and the recent work focuses on whole genome phylogenetic analysis for viruses, cancer bioinformatics, and bovine genomics. He is a member of ACM and IEEE Computer Society.