PedPhase Version 2.0 User Manual

Jing Li
Department of Computer Science
University of California-Riverside, CA 92521
and Department of Electrical Engineering and Computer Science
Case Western Reserve University
jili@cs.ucr.edu.

July 22, 2004

PedPhase Version: 2.0. Release time: July 2004. Author: Jing Li Copyright (c): Jing Li. All rights reserved. Collaborators: Tao Jiang and Koichiro Doi. The PedPhase software and this document can be obtained at `http://www.cs.ucr.edu/~jili/haplotyping.html`. This software is provided "AS IS" and is free for noncommercial use. The developers will not be responsible for any damages resulting from its use. Please report bugs to the author at jili@cs.ucr.edu.

References:

(i) Jing Li and Tao Jiang. An exact solution for finding minimum recombinant haplotype configurations on pedigrees with missing data by integer linear programming. *In Proc. RECOMB'04*, pp. 101–110, 2004.

(ii) Jing Li and Tao Jiang. Efficient rule-based haplotyping algorithms for pedigree data. *In Proc. RECOMB'03*, pages 197–206, 2003. Full version in *J. Bioinfo. and Comp. Biol.* 1(1):41-69, 2003.

(iii) Koichiro Doi, Jing Li, and Tao Jiang. Minimum recombinant haplotype configuration on pedigrees without mating loops. *Workshop on Algorithms in Bioinformatics (WABI)*, September 15-20, 2003, Budapest, Hungary;

Contents

1 Introduction **5**

1.1 Overview 5

1.2 What's new in version 2.0 5

1.3 Algorithms 6

1.4 Platforms 8

2 Running the software **9**

2.1 Obtaining and installing the software 9

2.2 File list 9

2.3 Running the software 9

2.4 Input files and data preparation 10

2.5 Output files 11

2.6 Conventions and restrictions 11

Chapter 1

Introduction

1.1 Overview

The program PedPhase contains functions for inferring haplotypes from genotypes for members in a general pedigree. Specifically, we are interested in solving the *minimum recombinant haplotype configuration* (MRHC) problem, *i.e.* given a pedigree and the genotype information of each member in the pedigree, we are interested in finding the haplotype configuration for each member such that the total number of or recombination events in the whole pedigree is minimized. Figure 1.1 illustrates an example input and the expected output of the MRHC problem. The diagram on the left shows the pedigree structure and genotype information. The diagram on the right shows the paternal/maternal origin of each allele in a child (*i.e.* the inferred haplotype/phase information). Notice that, there is a recombinant between loci 6 and 7 of the maternal haplotype in member 3-4.

1.2 What's new in version 2.0

In addition to the four algorithms in version 1.0, PedPhase V2.0 implements an integer linear programming (ILP) algorithm as described in [5]. The ILP algorithm is an exact algorithm for MRHC that integrates missing data imputation and haplotype inference in one framework. It also incorporates marker interval distance into the formulation thus overcomes the inadequacy of many other rule-based algorithms that ignore the important information. The program also outputs the likelihood of a haplotype solution based on the distance information and the standard assumptions for pedigree analysis. Experiment results show that the algorithm is effective for practical size human pedigrees. Users should use this algorithm other than the algorithms in version 1.0 whenever possible.

Figure 1.1: An illustration of the input and expected output of the MRHC problem. In the pedigree, a square represents a male and a circle represents a female. The children (*e.g.* 3-3 and 3-4) are placed under their parents (*e.g.* 3-1 and 3-2). The member ID and genotype data are placed under each member. On the left, the blank between two alleles at a locus indicates that the locus is phase-unknown. On the right, a | separates the paternal allele from the maternal allele. For each phase-resolved locus, we further use two numbers in parentheses to indicate that the alleles come from the parents' paternal (represented as 0) and maternal (represented as 1) alleles. The number of recombinants can be easily derived from this information.

1.3 Algorithms

We briefly describe the idea of the algorithms in PedPhase. For details of each algorithm, please refer to the corresponding paper.

- **ILP algorithm** The ILP algorithm formulates the MRHC problem with missing data using the integer linear programming technique and employs a branch-and-bound strategy that utilizes a partial order relationship and some other special relationships among variables to decide the branching order. The partial order relationship is discovered in the preprocessing of constraints by considering unique properties in the formulation. A directed graph is built based on the variables and their partial order relationship. By identifying and collapsing the strongly connected components in the graph, we may greatly reduce the size of an ILP instance. Non-trivial lower and upper bounds on the optimal number of recombinants are introduced at each branching node to effectively prune the search tree. When multiple solutions exist, a best haplotype configuration is selected based on a maximum likelihood approach. The paper also shows for the first time how to incorporate marker interval distance into a rule-based haplotyping algorithm.

PedPhase V1.0 implements four algorithms, namely, the block-extension algorithm, constraint-finding algorithm, locus-based DP (dynamic programming) algorithm and member-based DP algorithm. The algorithms are described in detail in [3, 4, 2] and will be shortly discussed later. The program is written in C++ and has been tested on simulated data and real data. The test results show that the programs perform

Algorithm	Time Complexity	Optimal	# of Solutions	Target Input Data
Block-extension	$O(dmn)$	No	1	large dataset with few recombinants
Constraint-finding	$O(m^3n^3)$	Yes	all	dataset with 0 recombinant
Locus-based DP	$O(nm_0 2^{2m_0})$	Yes	all	loopless pedigree, small number of markers
Member-based DP	$O(mn2^{4n})$	Yes	all	small pedigree

Table 1.1: Comparison of the four algorithms. In the above time complexity formulas, n is the size of the pedigree, m the number of loci, m_0 the maximum number of heterozygous loci in any member, and d the largest number of children in a nuclear family.

better than previous known algorithm and are thus useful for haplotyping pedigree data.All the algorithms are rule-based and thus model-free. The block-extension algorithm is an efficient heuristic algorithm for MRHC that performs very well when the input data requires few recombinants. The other three are exact algorithms for MRHC under various restrictions. A summary of major differences of the four algorithms can be found in Table 1.1. All the algorithms take pedigree structure and genotype data in one file as input and output haplotypes for each member of the pedigree. Please refer to 2.4 for the format of input file and to 2.5 for the output files.

- **Block-extension algorithm** The block-extension algorithm described in [3, 4] first attempts to resolve the haplotype configuration of all unambiguous loci using the *Mendelian law* of inheritance. Some sensible greedy strategies (such as avoiding double recombinants within a small region of loci) are then used to resolve loci that are adjacent to the previously resolved loci, resulting in *blocks* of consecutive resolved loci. The algorithm then uses the longest block in the pedigree to resolve more unresolved loci under the minimum recombination principle. This may extend some blocks into longer blocks. The process is repeated until no blocks can be extended. The algorithm then fills the remaining gaps between blocks in each member by considering the haplotype information at the other members of the same nuclear family. The time complexity of the above algorithm is $O(dmn)$, where n is the size of the pedigree, m the number of loci, and d the largest number of children in a nuclear family.

- **Constraint-finding algorithm for 0-recombinant data** The constraint-finding algorithm [3, 4] first determines if an MRHC instance has 0-recombinant haplotype configurations and identify all such configurations if it does. More precisely, the algorithm first identifies all necessary (and sufficient) constraints on the haplotype configurations derived from the Mendelian law and the zero-recombinant assumption, represented as a system of linear equations on binary variables over the cyclic group Z_2 (*i.e.* integer addition `mod` 2), and then solves the equations to obtain all consistent haplotype configurations satisfying the constraints, using a simple method based on Gaussian elimination. These consistent haplotype configurations are shown to be feasible 0-recombinant solutions. The running time for representing and solving the equations is $O(m^3n^3)$ and the time for enumerating all config-

urations is proportional to the number of feasible 0-recombinant solutions.

- **Locus-based dynamic programming algorithm** This algorithm assumes that the number of marker loci is bounded by a small constant and performs dynamic programming on the members of the input pedigree. It works only for pedigrees without mating loops. The algorithm takes advantage of the tree structure of the input pedigree and has a running time linear in the size of the pedigree. It could be very useful for solving MRHC in practice because most real pedigrees are loopless and involve a small number of marker loci (in each haplotype block).

- **Member-based dynamic programming algorithm** This algorithm assumes that the input pedigree is small and performs dynamic programming on the marker loci in each member of the pedigree simultaneously. The algorithm works for any input pedigree (without or with mating loops) and has a running time linear in the number of marker loci. It could be useful as a subroutine for solving MRHC on small pedigrees, *e.g.* nuclear families from a large input pedigree or independent nuclear families from a (semi-)population data.

1.4 Platforms

PedPhase V2.0 only has executable code for Windows currently due to the using of IBM OSL library. We will use the source code from the Computational Infrastructure for Operations Research (COIN-OR) project in our future version so that we can compile PedPhase on different platforms. PedPhase V1.0 has executable code for Windows and Linux.

Chapter 2

Running the software

2.1 Obtaining and installing the software

The PedPhase software and this document can be obtained at http://www.cs.ucr.edu/˜jili/haplotyping.html. Once downloading the software, the user could create a folder and uncompress the files under the newly created folder. The user could use WinZip to unzip the file on Windows and use the command "`tar -xzvf PedPhase.tar.z`" to unzip the file on Linux.

2.2 File list

In addition to the executable file PedPhase.exe (PedPhase on Linux) and the user's guide PedPhaseReadMe.pdf, there are four more example files: blockext.txt, dp.txt, 29-26-2.txt and markerdist.txt. The files contain simulated data. In blockext.txt there are 100 pedigrees and each pedigree consists of 15 members. The number of markers is 10. The block-extension, constraint-finding and locus-based DP algorithms could easily handle the data in minutes on a regular PC. A smaller pedigree with 8 members in dp.txt can be used to test the member-based DP algorithm. The file 29-26-2.txt contains two pedigrees and each pedigree contains 29 members and 26 markers. This data set can be used to test the ILP algorithm. The markers are unevenly spaced and the recombination fractions of marker intervals are in the file markerdist.txt. The file oslmd6030.dll is the OSL library file.

2.3 Running the software

One can launch PedPhase on a Command Prompt (DOS) window on Windows by typing the following command:

```
PedPhase.exe -option infile.
```

Similarly, one can launch PedPhase by typing the following command on Linux terminal:

`PedPhase -option infile`.

The above assumes that PedPhase is in the current directory. Otherwise, we need include the correct path information in front of the executable file.

There are 5 options in the current version. Different options will invoke different algorithms as follows.

```
-b:  block-extension
-c:  constraint-finding
-l:  locus-based DP
-m:  member-based DP
-i:  ILP
```

2.4 Input files and data preparation

The input file consists of the pedigree information and the marker information for each member. The structure of the input file is very simple. Each line represents one individual. Different fields are separated using Tab. Each line consists of the following fields: `FamilyID IndividualID FatherID MotherID Gender AffectionStatus LiabilityClass Allele11 Allele12 Allele21 Allele22 ....`

`FamilyID` and `IndividualID` are natural numbers. `FatherID` and `MotherID` are the current individual's parents' `IndividualID` and are set to 0 if the current individual does not have parent information available in the pedigree. For `Gender`, 1 stands for male and 2 stands for female. `AffectionStatus` and `LiabilityClass` are reserved fields and are not used in the current version. Allele numbers (`Allele11 Allele12 Allele21 Allele22 ...`) are non-negative integers where 0 stands for missing value.

The input file can be prepared by using any file editors such as Notepad on Windows or Emacs or vi on Linux. It can also be exported from a database system. There can be multiple pedigrees in an input file, but the marker information for all individuals must be consistent (*i.e.* all the individuals have the same marker loci). At the end of the input file, there is an empty line. The user is encouraged to check Mendelian errors using other programs like PedCheck [6] before feeding the data to PedPhase, although we have implemented a simple algorithm for checking Mendelian inconsistency in nuclear families for the block-extension algorithm. All other algorithms will report no solutions if there are errors in the data.

When users invoke the ILP algorithm, PedPhase will search for the marker distance file with the fixed name "markerdist.txt" under the current directory. The file contains the recombination fractions of each marker interval, one at each line. For example, the default file has 25 numbers corresponding to the 29-26-2.txt data that has 26 markers and in the same order. The users should make sure the correctness of the marker distance file. If the file exsits, PedPhase will use it as described in [5]. Otherwise, PedPhase will assume that the markers are evenly spaces and no likelihood will be reported.

2.5 Output files

There are three output files for each input file. If the name of the input is `a.txt`, the names of the output files would be `a.out`, `a.dat` and `a.res` respectively. The file `a.out` contains the haplotyped data (as well as pedigree information that is the same as the input data). For each locus, the two alleles are separated by a '—' to indicate that the first allele is paternal and the second maternal. At the end of each line, two binary strings follow to further indicate the grand-parental source of each allele (*i.e.* 0 means that the allele is from the parent's paternal haplotype and 1 means that the allele is from the parent's maternal haplotype). For founders of the pedigree, we arbitrarily fix the haplotype at a heterozygous locus and set the grand-parental source of the alleles at this locus to 0s. The file `a.dat` has exactly the same information but has been formatted so that the program WPEDRAW [1] could read it directly. The user can use WPEDRAW to draw the traditional pedigree graphs from such a file. But WPEDRAW can only draw data with 10 markers according to our experience. Some auxiliary information like the time used by PedPhase, the number of recombinants in each pedigree and the likelihood of a haplotype solution (only if using ILP with marker distances) can be found in the file `a.res`.

2.6 Conventions and restrictions

There are some conventions and restrictions in the current version of PedPhase:

- The PedPhase V2.0 can only run on Windows and current implementation can only output one optimal solutions. We are trying to use a new linear programming library, namely the COIN-OR, in our future version. Hopefully we can provide executable code on different platforms and will output all optimal solutions.
- An individual has either both parents or no parents at all in the pedigree. In the latter case, the individual is called a founder of the pedigree. Since there is no way to tell the paternal haplotype from the maternal haplotype in a founder, we arbitrarily fix the haplotype of a heterozygous locus in each founder.
- We have implemented some algorithms in block-extension to detect Mendelian inconsistency only in nuclear families, and recommend the user to use some other programs like PedCheck to check Mendelian errors first.

Bibliography

[1] D. Curtis A program to draw pedigrees using linkage or linksys data files. *An. of Hum. Genet.*, 54:365–367, 1990.

[2] K. Doi, J. Li and T. Jiang. Minimum Recombinant Haplotype Configuration on Pedigrees without Mating Loops. In Proc. WABI2003 pages 339-353.

[3] J. Li and T. Jiang, Efficient rule-based haplotyping algorithms for pedigree data. *Proc. RECOMB'03*, pages 197–206, 2003.

[4] J. Li and T. Jiang, Efficient inference of haplotypes from genotypes on a pedigree. *J. Bioinfo. and Comp. Biol.* 1(1):41-69, 2003.

[5] J. Li and T. Jiang. An exact solution for finding minimum recombinant haplotype configurations on pedigrees with missing data by integer linear programming. *In Proc. RECOMB'04*, pp. 101–110, 2004.

[6] J. R. O'Connell and D. E. Weeks. Pedcheck: a program for identification of genotype incompatibilities in linkage analysis. *Am J Hum Genet*, 63(1):259–266, 1998.