# Reconstruction of Half-Sibling Population Structures

by

Daniel Dexter

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2012

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

## Abstract

Half-sibling reconstruction is the task of determining maternal and paternal sibling relationships from observed genotypes of same-generation individuals in a population. Knowledge of how populations are structured allows biologists to understand mating habits of different species, how threatened a population is, and how best to protect threatened or endangered species.

This thesis examines the problem of half-sibling reconstruction and explains an accurate and fast heurstic for reconstructing half-siblings. The heuristic reconstructs half-sibling relationships with high accuracy on large biological populations where existing algorithms fail due to running time constraints. In addition to identifying and discussing some of the major problems with half-sibling reconstruction, we also prove that even the task of determining whether a half-sibling reconstruction obeys genetic inheritance laws is $NP$-complete. Some solutions for overcoming the inherent difficulty of half-sibling reconstruction are also proposed.

**Acknowledgements**

I would like to thank my supervisor Dr. Dan Brown for all of his support, infinite patience, and kindness. He has helped me mature both professionally and academically as a computer scientist by encouraging and furthering my mathematical knowledge. Thank you to Dr. Debra Goldberg who piqued my interest and encouraged my involvement in bioinformatics. I would also like to thank Dr. Ming Li and Dr. Bin Ma for reviewing my thesis.

My fellow graduate students also deserve many thanks. Special thanks to Rita Ackerman and David Loker for increasing my understanding of clustering and to Jakub Truszkowski for all of his support. Thank you to my lab mates who have been a source of encouragement during my time in Waterloo.

Although all of my friends have my gratitude for everything they have done, I would like to specifically acknowledge Ben Joeris for introducing me to the mathematical side of computer science and for all of the advice and explanations he has given me over the years. I would also like to thank William Van Treuren for his support as a friend and fellow scientist.

This thesis would not have been possible without the immense support I have received from my family. My mother, Maureen Tanaka, and father, Kevin Dexter, have all of my thanks and love. Thank you for always believing in me academically and for constantly supporting and guiding me in all areas of my life.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Half-sibling reconstruction is the task of determining maternal and paternal sibling relationships from observed genotypes of same-generation individuals in a population. Reconstructions can take place with or without information about the genotypes of the parents, but are often more accurate when the parents' genotypes have been identified. In cases where the parental genotypes are unknown, a secondary objective may be to infer a parent's genotype based on the genotype of the offspring.

## 1.1 Motivation for Half-Sibling Reconstruction

Conservation biologists and molecular ecologists use pedigree analysis to gain insight into the mating habits and practices of populations. For example, knowing the reproduction mechanics of a population helps biologists make important ecological decisions about a region *e.g.* [16, 28]. The information may also be used to assist in reproduction and conservation of endangered or threatened species [10, 15]. A sub-field of pedigree analysis seeks

answers to the question of how same-generation individuals are related. Identifying related full sibling individuals, or individuals who share both a common mother and common father, is well-studied and many algorithms exist for modeling such populations. A similar, but much more difficult task involves discovering individuals who are related by a single parent, called half-siblings. Half-sibling relationships can always be used to reproduce full-sibling relationships; however, the converse is not necessarily true. Additionally, correct half-sibling reconstruction also allows biologists to measure the degree of polygamy within a species, which is not possible with full-sibling reconstruction alone.

Knowledge about half-sibling relationships has important real-world applications and answers questions that full sibling reconstruction cannot. For example, knowing which individuals share a single common parent allows biologists to measure the degree of polygamy within a population [27], since the mates of each parent can easily be computed from half-sibling partitionings. Half-sibling reconstruction also provides specific insight about pollination patterns of plant populations. In plant populations, mothers are pollinated by potentially distant fathers. The diversity of pollinating fathers can be used to measure the degree of isolation, due to deforestation, which threatens many forests [16].

## 1.2 Biological Background and Notation

### 1.2.1 Notation

Information about individuals' genotypes are collected and expressed through the measurement of *microsatellites*, sequences of repeating DNA base pairs, such as ATATATAT, at a specific site on a single chromosome. The number of repeats gives an integer value denoting the *allele* for an individual. Microsatellites are collected from homologous chromosome

pairs. It is impossible to distinguish the two chromosomes with inexpensive technology. Each measurement site is called a microsatellite *locus*. In practice, scientists identify and report alleles at multiple loci in a population and these loci are assumed to be independent from one another, as they are found on different chromosomes.

We will assume that each individual is diploid, meaning that population members possess two of each type of chromosome: in particular, this excludes loci on the sex chromosomes. Exactly one chromosome is inherited from each of the individual's parents; therefore, each locus will have a maternal and paternal allele. Let $m$ be the number of measured loci for a population. Each locus in the population will have a variable number of alleles, $k$, which we represent as $A_l = \{a_0, a_1, \ldots, a_{k-1}\}$. In practice, some alleles are more common than others.

During mating, a single maternal and paternal allele at each locus combine to give an individual's genotype, which is unordered: $(a_i, a_j)$ is equivalent to $(a_j, a_i)$. Unfortunately, it is not always possible to reconstruct an individual's alleles at a given locus. *Allelic dropout* is a common error in genotyping, where information about a locus cannot be confidently determined and is omitted. We express sites with allelic dropouts as $(*, *)$.

The half-sibling problem is: given a population of $n$ offspring and their genotypes at $m$ loci, reconstruct a maternal and paternal partitioning, $\mathcal{M}$ and $\mathcal{P}$ respectively, which obey the Mendelian laws for half-siblings. Each partition corresponds to a half-sibling family, called a half-sibship. For each pair of $M \in \mathcal{M}$ and $P \in \mathcal{P}$, the individuals in $F := M \cap P$ are full-siblings, since they are offspring of a common mother and father.

### 1.2.2 Mendelian Compatibility

Sibling reconstruction finds a population clustering which obeys Mendelian genetics. In the full-sibling clustering $\mathcal{F}$, each individual appears only once. For half-siblings, an algorithm must construct $\mathcal{M}$ and $\mathcal{P}$ when both sexes are polygamous or only one of the two partitions when one sex is monogamous.

The following rules assist the sibling reconstruction process by determining whether or not groups of individuals can be biologically related. Berger-Wolf *et al.* [4] give two Mendelian properties of diploid full-siblings. In any full-sibgroup, at each locus, at most four alleles appear, since there are two parents, each with at most two alleles. This is the *4-allele property*. The *2-allele property* enforces the rule that for each full-sibling group, there is a partitioning of the alleles at each locus into a maternal and paternal group, such that each individual obtains exactly one allele from the maternal set and one from the paternal set. Sheikh *et al.* [29] extend these rules to half-siblings. The *half-sibship property* states that for each locus in a half-sibling family, there exist two alleles $\{a_i, a_j\}$, which are the alleles of the shared parent, each individual possesses either $a_i$ or $a_j$ at that locus.

## 1.3 Contributions

Half-sibship reconstruction is still relatively unexplored. This thesis combines and expands on current research related to pedigree reconstruction. In particular, we examine whether similarity measures and techniques used in full sibling reconstruction algorithms can be applied to the half-sibling problem. We also provide a fast heuristic-based algorithm for reconstructing half-sibling populations and compare it to current algorithms. Although a handful of algorithms exist for the half-sibling reconstruction problem, they are all too

slow and fail to find correct partitionings when the population is large. Our heuristic algorithm, SibJoin [9], is thousands of times faster than existing methods and can find reasonably accurate partitionings for large populations where current algorithms fail. Despite its heuristic basis, simulated and real populations show that SibJoin has competitive accuracy to the best existing methods.

Although half-sibling reconstruction appears similar to the full-sibling reconstruction problem, there are several differences that make the half-sibling version a more difficult problem. We discuss barriers to accurate half-sibling family reconstruction and present solutions to some of these problems. We also prove that even deciding whether half-sibling partitionings are valid under Mendelian inheritence rules is *NP*-complete and formulate an integer program which solves the related optimization objective of minimizing the number of individuals which need to be removed to make the proposed reconstruction valid. The *NP*-hardness result has important negative implications for existing half-sibling reconstruction algorithms, discussed in Sections 2.2, which attempt to reconstruct half-sibships from full-sibships.

# Chapter 2

# Related Work

## 2.1 Full Siblings

Full sibling pedigree reconstruction is well-studied. Most existing sibship discovery algorithms use statistical models of populations and maximum likelihood, combinatorial minimization of some objective, or heuristics. A majority of existing algorithms are likelihood-based which makes them unsuitably slow for large populations. Combinatorial methods using integer programming (IP) are parallelizable, but are still too slow when populations reach the low hundreds of individuals. A detailed survey of existing full-sibling reconstruction algorithms is given by Jones *et al.* [20], where the benefits and costs of relevant reconstruction algorithms are discussed in detail.

### 2.1.1 Likelihood

Most full sibling pedigree reconstruction algorithms use likelihood approaches to model populations [1,21,30,33,34]. Likelihood methods estimate the probability of the data under different partitionings of a population. An optimal solution maximizes this probability. Most likelihood methods use Markov Chain Monte Carlo, simulated annealing, or other search strategies to find their proposed solutions. These strategies are often very slow, making them ill-suited for sibling reconstruction on large data sets. On the other hand, because this class of algorithm establishes a probabilistic model, it is often possible to directly incorporate error handling and prior assumptions about the population structure, to increase accuracy. Of the likelihood-based approaches, COLONY [33], COLONY 2 [34], and PRT 2 [1] are specifically related to the results of this thesis.

Likelihood algorithms for reconstructing sibships are either pairwise methods, which investigate the relationship between pairs of individuals, or group methods. COLONY is a group likelihood method that models entire families. In the COLONY algorithms phenotype is defined as the observed genotype of an individual and the genotype to be the true genotype without error. The likelihood model allows COLONY to account for two types of errors which the authors call class I and class II errors. Class I errors occur when one of the alleles fails to be amplified by the polymerase chain reaction and may only occur for heterozygotic loci. All other errors, such as misidentification or mutation, are class II errors. COLONY uses the class I and II errors to calculate the probability of the phenotype given the underlying genotype. The full likelihood equation calculates the likelihood of each individual's phenotype under different parent phenotypes. The original COLONY program only allows one sex to be polygamous, but polygamy of both sexes was introduced in COLONY 2. Both maximum likelihood algorithms use simulated annealing to find the most likely population structure and avoid getting trapped in local maximums;

however, there is still no guarantee that the best global solution will be found. Although COLONY and COLONY 2 offer robust error modeling, results by Sheikh *et al.* [29], as well as our own results, show that COLONY and COLONY 2 become prohibitively slow for even medium-sized populations. Additionally, as demonstrated in Almudevar and Anderson [1], COLONY 2 often splits true sibgroups into smaller groups, resulting in an incomplete reconstruction.

PRT 2 [1] is a likelihood method that can be significantly faster than either of the COLONY algorithms because it only considers maximal sibgroup families. A maximal sibgroup is a group which is compatible under rules of Mendelian genetics, but for which adding any other individual will make the group infeasible. PRT 2 allows the user to select from three different algorithms for enumerating MSGs. The fastest method is a heuristic which enumerates full siblings based on compatible triples of individuals and takes seconds to run for populations of hundreds of individuals, but only generate a partial set of MSG's. The slower options are graph-based and can take hours to run for large populations, but evenutally enumerate the entire set of MSG's. Choosing which of the slower algorithms to use depends on the estimated size of populations. The first method is a top down algorithm that starts with large sibgroups and splits them until compatible groups are found. The other is a bottom up algorithm that is more suitable when sibgroups are small. One drawback of providing different options is that a user may pick the wrong one if little is known about family sizes of the population. Choosing wrong MSG constructor results in poor performance since the algorithm begins searching for maximal groups of inappropriate size. Because PRT 2 only calculates maximum likelihood from MSGs, it is faster than COLONY, but does not account for errors and will perform poorely if true sibgroups are small, but the population is highly compatible.

### 2.1.2 Combinatorial Optimization

Combinatorial optimization approaches seek to provide a sibship partitioning which minimizes or maximizes some objective function, such as the number of families, matings, or parents. As with likelihood methods, finding global optima for large populations can be computationally demanding. However, many optimization techniques are easily parallelizable.

KINALYZER [3] seeks a minimum set cover by using an integer programming (IP) formulation where each set is subject to restrictions of Mendelian compatibility for full-siblings. That is, it seeks to minimize the number of matings. KINALYZER yields decent results [11]; however, like the COLONY programs, it does not scale well with population size. The minimum set cover objective used by KINALYZER is *NP*-hard to optimize [11] and, so KINALYZER cannot handle populations with more than a few hundred individuals. Additionally, KINALYZER is very sensitive to errors since there is no built-in tolerance for mislabeled or mutated alleles. Another problem is that KINALYZER can find multiple optimal solutions, since its objective is just the number of matings, and it provides no way to choose which solution is best.

Brown and Berger-Wolf provide a different IP formulation which still minimizes the number of sibgroups, but only requires a polynomial amount of variables and constraints [8]. The approach uses the property that an incompatible full sibling family will contain an incompatible triplet of individuals under Mendelian genetics. We discuss such incompatibilities in more detail in Chapter 3. IP constraints are generated for each incompatible triplet in the full-sibling population, to require that they do not all fall in the same sibgroup. Unlike KINALYZER, the incompatible triplets IP does not keep track of sibgroups directly. Instead, a constraint for each distinct triplets of individuals enforces the condition that if individual $i$ is related to $j$ and $j$ is related to $k$, then $i$ must also be related to $k$. In

total, the IP has $O(n^2)$ variables and $O(n^3)$ constraints, but generally takes more time than KINALYZER for reconstructing sibships, even though its IP is exponentially smaller [8].

### 2.1.3 Fast Heuristics

Heuristics have been applied to the sibgroup reconstruction problem so that researchers may obtain putitive sibgroups for large populations. By making use of simplifying observations, heuristics can produce reasonably accurate results hundreds to thousands of times faster than pure likelihood or combinatorial methods.

Brown and Berger-Wolf propose a clustering algorithm which joins two individuals based on the number of genetically compatible third partners [8]. Brown and Berger-Wolf use probabilistic arguments to justify the assumption that if two individuals form a large number of compatible full-sibling triplets, then they are likely to be full-siblings. The clustering algorithm they use represents each individual as a vertex in a graph with weighted edges that denote the number of third individuals for which the pair are compatible. Once triplet similarities for each pair of individuals has been calculated, the algorithm sets a threshold. Any edge with a lower weight than the threshold is removed and the remaining connected individuals are tested for full sibling compatibility. Once all compatible families are joined together, the threshold is raised by one and the process is repeated until all groups are valid full sibships: true full siblings are likely to have a higher edge weight than unrelated individuals. The result is that the largest compatible families are joined first, but the algorithm becomes increasingly selective until no incompatibilities remain. For a population of $n$ individuals with $m$ loci, this algorithm has an $O(n^3m)$ runtime. On simulated and real population data, the heuristic is as accurate or more accurate than KINALYZER and hundreds of times faster. The algorithm is important because it demonstrates that

simple heuristics can quickly produce accurate results.

## 2.2 Half-Siblings

Far fewer algorithms exist for reconstructing half sibling families than for full siblings. Weaker Mendelian constraints for half-siblings create a larger space of feasible solutions and make it more difficult to identify unrelated individuals. As a result, many techniques used by full sibling algorithms perform too slowly to be adapted to the half sibling problem. Current half sibling algorithms include COLONY 2, PRT 2, and an IP based on the minimum set cover formulation of KINALYZER.

COLONY 2 expands the half-sibling compatibility of COLONY from polygamy in one sex to allowing it in both sexes. Like the full sibling case, COLONY 2 accounts for allelic errors and produces accurate results. However, the ability to handle polygamy in both sexes make COLONY 2 even slower and unsuitable for reconstructing large populations.

PRT 2 also claims to support half-siblings, but half sibling groups are never directly computed. Instead, the outputed solution presents full sibling groups and a list of which pairs of groups can form valid half-sibling families. This is problematic in instances where both sexes are highly polygamous because there will be many pairs of half sibling compatible full sibling families, however, PRT 2 does not indicate which compatibilities are true half sibling groups nor which are maternal and which are paternal. Additionally, in Section 6.1, we will show that reconstructing half sibling populations is *NP*-Hard. This makes PRT 2 an ineffective half sibling tool for all but the most simple instances.

Recent work has proposed half-sibling IP strategies which are similar to the full-sibling strategies in KINALYZER, though they are unsuccessful at reconstructing large populations [29]. The most viable of these is the half-sibling minimum set cover (HS-MSC) IP.

| $M_0:(4,8)$ | $M_1:(1,4)$ | $M_2:(1,6)$ | $P_0:(9,10)$ | $P_1:(2,3)$ | $P_2:(5,7)$ |
|---|---|---|---|---|---|
| (4,9) | (1,2) | (1,5) | (4,9) | (1,2) | (1,5) |
| (4,10) | (1,3) | (1,7) | (4,10) | (1,3) | (1,7) |
| (8,9) | (2,4) | (5,6) | (8,9) | (2,4) | (5,6) |
| (8,10) | (3,4) | (6,7) | (8,10) | (3,4) | (6,7) |

(a) The true family structures of a population

| $C_0$ | $C_1$ |
|---|---|
| (2,4) | (1,2) |
| (3,4) | (1,3) |
| (4,9) | (1,5) |
| (4,10) | (1,7) |
| (8,9) | (5,6) |
| (8,10) | (6,7) |

(b) The HS-MSC solution

| $M_0:(4,8)$ | $M_1:(1,6)$ | $P_0:(9,10)$ | $P_1:(2,3)$ | $P_2:(5,7)$ |
|---|---|---|---|---|
| (2,4) | (1,2) | (4,9) | (2,4) | (1,5) |
| (3,4) | (1,3) | (4,10) | (3,4) | (1,7) |
| (4,9) | (1,5) | (8,9) | (1,2) | (5,6) |
| (4,10) | (1,7) | (8,10) | (1,3) | (6,7) |
| (8,9) | (5,6) | | | |
| (8,10) | (6,7) | | | |

(c) Full family model which minimizes the total number of clusters

Figure 2.1: An example population which is correctly reconstruced by minimizing the total number of matings, but incorrectly reconstructed by minimizing the total number of families

However, the HS-MSC failes to estimate half-sibling groups for large populations due to slow runtimes and is still unavailable for public use. Additionally, there is no evidence that minimizing the number of sibgroups is the right thing to do in all instances [1]. The KINALYZER algorithm has a natural parsimonious explanation: it minimizes the number of matings which must occur to produce a population. For half-siblings, minimizing the number of clusters is not equivalent to minimizing the number of matings.

Figure 2.1 demonstrates an example where minimizing the total number of clusters leads to an incorrect sibling reconstruction even though the matings result in the same population. Figure 2.1a shows a true population structure, which requires three matings, $M_0$ with $P_0$, $M_1$ with $P_1$, and $M_2$ with $P_2$, and six clusters, which is also the structure obtained by minimizing the total number of matings. Figure 2.1b shows the result of min-

imizing the total number of clusters with the HS-MSC which requires four matings and five clusters to explain. The HS-MSC algorithm does not produce separate maternal and paternal partitionings. Instead, it attempts to find a single minimum set cover which obeys Mendelian inheritance for half-siblings. In this example, the HS-MSC algorithm produces a solution with two clusters: $C_0$ has a common parent of $(4, 8)$ and $C_1$ has a common parent of $(1, 6)$. One of the fundamental problems with the HS-MSC objective is that it fails to distinguish between maternal and paternal families. Moreover, it excludes real half-sibling families since it does not produce maternal and paternal partitionings. Notice that the minimum set cover $C_0$ and $C_1$ produced by the HS-MSC in Figure 2.1b are the same as the maternal clusters $M_0$ and $M_1$ in Figure 2.1c. However, the HS-MSC algorithm does not reconstruct any of the paternal half-sibling families given by $P_0$, $P_1$, and $P_2$ in Figure 2.1c.

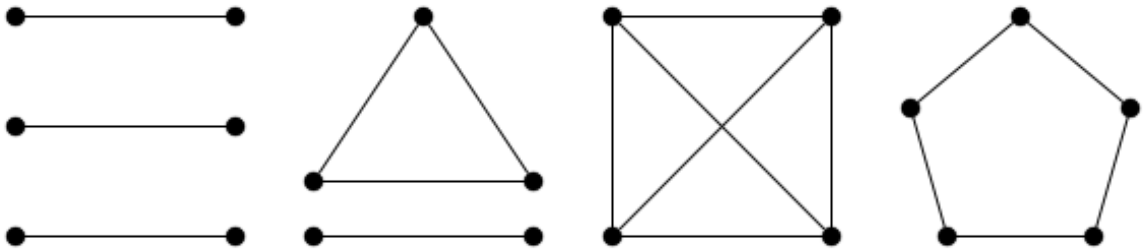
To illustrate why minimizing the number of clusters may lead to a poor solution, assume that the two half-sibling clusters in Figure 2.1b are correct. Figure 2.1c shows the full maternal and paternal partitionings that are generated by minimizing the total number of clusters. In this example, the population is generated by $M_0$ mating with $P_0$ and $P_1$ while $M_1$ mates with $P_1$ and $P_2$ for a total of four matings. Therefore, minimizing the number of clusters may require a more complicated mating structure to reproduce a population. To summarize, there are two problems with the HS-MSC approach: it fails to produce half-sibling clusters and it can result in an unparsimonious mating structure.

We have introduced the existing algorithms for reconstructing half-sibships; however, the PRT 2 and HS-MSC approaches do not reconstruct complete half-sibling partitions. Additionally, the COLONY and HS-MSC algorithms are too slow to be used with populations in the hundreds of individuals. As DNA sequencing becomes less expensive and biologists are able to produce larger population samples, alternative approaches must be developed in order to process the samples in a reasonable amount of time.

# Chapter 3

# Quantifying Pairwise Similarity

Sibship reconstruction is, in effect, a clustering problem: individuals are clustered into families based on some measure of relatedness. The relatedness measure is meant to encourage joining members of the same family. The ability of clustering algorithms to accurately reconstruct groups of related individuals depends on finding an appropriate measure of similarity between individuals. Brown and Berger-Wolf proposed a successful similarity measure for full siblings based on Mendelian compatible triples of individuals [8]. We compare their triplet similarity to a simpler method which measures similarity based on shared alleles between pairs of individuals and determine which is more suitable for half-siblings. Our results indicate that triplet compatibility is acceptable in most cases and preferable when the number of distinct alleles at each locus is large, but that allele similarity is a better choice for most reasonable populations.



(a) (b) (c) (d)

Figure 3.1: Minimal allele adjacency graphs of incompatible half-sibling families

## 3.1 Triplet Similarity

For full siblings, one method of defining similarity between individuals $i$ and $j$ is to count the number of third individuals $k$ that can be full siblings with $i$ and $j$ simultaneously. Building a similarity matrix for all pairs of individuals in a population requires enumerating all triplets in a population and testing their compatibility, which takes $O(n^3m)$ time by brute force. Triplet similarity is an effective measure for full siblings, in part because any incompatible candidate family contains an incompatible triplet [8]. Therefore, it is unlikely that pairs of unrelated individuals will obtain a high similarity.

Unfortunately, an incompatible half-sibling family is not guaranteed to have an incompatible half-sibling triplet. In fact, it can require as many as six individuals to identify a half-sibling incompatibility.

**Definition 1.** *For an allele pool $A_l$ at locus $l$, an allele adjacency graph is a graph $G(V, E)$ with a vertex for each distinct allele $a_i \in A_l$ and edges between all pairs of allele vertices that co-occur at locus $l$ of an individual.*

In an allele adjacency graph, edges are equivalent to individuals and each pair of connected vertices represents the two alleles found at the fixed locus of an individual. For

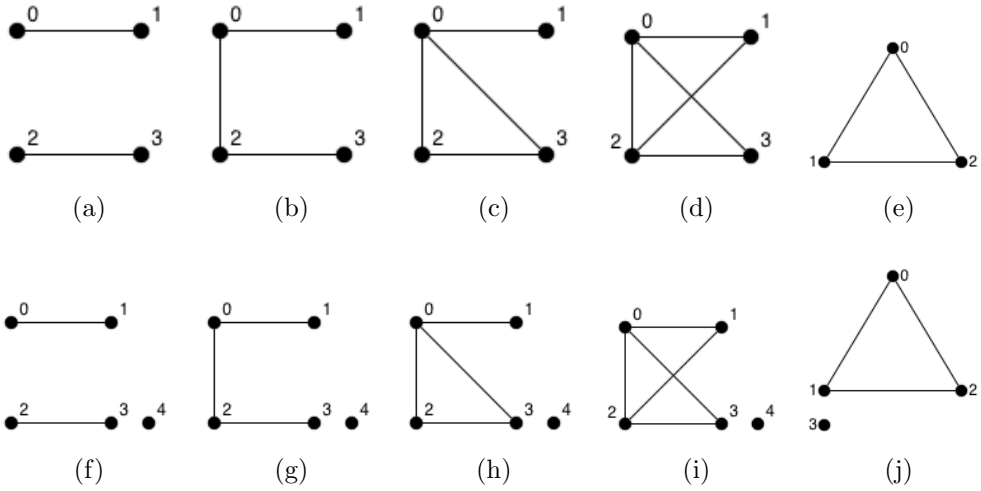



Figure 3.2: Candidate compatible allele adjacency graphs for $H(V, E)$

example, a family of six individuals such as $\{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$ would form an allele adjacency graph which is isomorphic to the graph in Figure 3.1c.

Given an allele adjacency graph for a fixed locus in a population, determining whether the individuals are compatible half-siblings at the locus is equivalent to determining if the graph has a minimum vertex cover of at most two vertices. The two vertices correspond to the alleles of the common parent.

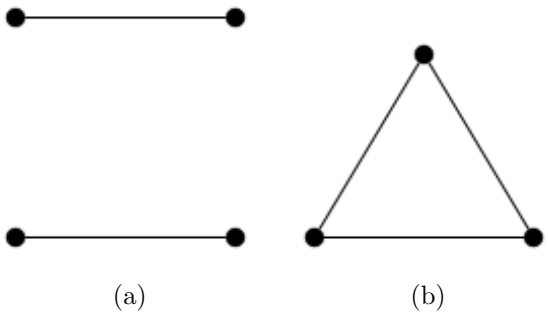



Figure 3.3: Minimal graphs with a minimum vertex cover number of two

**Theorem 1.** *Minimal incompatible half-sibling subgroups contain up to six individuals.*

*Proof.* Figure 3.1 demonstrates all the minimal allele adjacency graphs for incompatible half-sibling families. We will argue that any incompatible half-sibling allele adjacency graph must, up to an isomorphism, contain one of the four graphs in Figure 3.1 as a subgraph.

Without loss of generality, we modify the structure slightly by replacing any self-edge $(v_i, v_i)$, corresponding to a homozygote, with a new vertex $v_i'$, an edge $(v_i, v_i')$, and the requirement that only $v_i$ may share an edge with $v_i'$. For the vertex cover problem, this modified graph preserves the original solution since the new edge can only be covered by $v_i$ or $v_i'$, while the self-edge of the original graph can only be covered by choosing $v_i$. Therefore, if $v_i'$ appears in the vertex cover set, then it may be replaced with $v_i$ since the only edge touching $v_i'$ is $(v_i, v_i')$. This modification allows us to generalize homozygotic structure in the proof and ignore cases with self-edges.

Assume for contradiction that an incompatible allele adjacency graph $G(V, E)$ exists which does not contain any graph in Figure 3.1 as a subgraph, but still requires a common parent with three alleles; that is, has a vertex cover number of three. Furthermore, assume that this graph is minimal in the sense that removing any edge will reduce the minimum vertex cover size from three vertices to two. Remove the third vertex $v^*$ in the vertex cover and all adjacent edges. The resulting subgraph, call it $H(V, E)$, must have a two vertex cover, say $\{v_i, v_j\}$.

First, we observe that there are only two minimal graphs with a minimum vertex cover of two, which are shown in Figure 3.3, since any connected graph with more than four vertices must either be a star, which has minimum vertex cover number of one, or have two non-adjacent edges, which is isomorphic to Figure 3.3a. We enumerate all the possible graphs for $H(V, E)$ by adding edges that preserve the two vertex cover to the minimal

graphs in Figure 3.3.

It is also necessary to consider cases where $G(V, E)$ has more than one connected component. If $G(V, E)$ has multiple components which don't share an edge and $v^*$ connects to a separate component, then it is forced to increase the minimum vertex cover number by one, regardless of the structure of the second component. Therefore, we may represent the second connected component with a single vertex that shares an edge with $v^*$. We will try to reconstruct $G(V, E)$, examining six cases.

1. Adding $v^*$ to the graph in Figure 3.2a will result in a minimal three vertex cover graph with edges $\{(v^*, v_0), (v^*, v_1)\}$ or $\{(v^*, v_2), (v^*, v_3)\}$, but these graphs are isomorphic to Figure 3.1b. Therefore, $deg(v^*) > 2$ will violate the assumption of minimality for $G(V, E)$.

2. Adding $v^*$ to Figure 3.2b with edges $\{(v^*, v_1), (v^*, v_3)\}$ will result in a minimal graph, but this graph is isomorphic to Figure 3.1d. Adding $\{(v^*, v_0), (v^*, v_1)\}$ or $\{(v^*, v_2), (v^*, v_3)\}$ also create incompatible half-sibling graphs, but these graphs cannot be $G(V, E)$ since they are not minimal.

3. Adding edge $(v^*, v_1)$ to Figure 3.2c produces a graph with a three vertex cover, but it violates the minimality assumption. Any other three vertex cover graph contains the graphs in case b as a subgraph.

4. Any three vertex cover created by adding $v^*$ to Figure 3.2d contains a three vertex cover graph from case c as a subgraph.

5. Adding $v^*$ with an edge to each of $v_0, v_1, v_2$ in Figure 3.2e produces a minimal three vertex cover graph, but it cannot be $G(V, E)$ since it is isomorphic to Figure 3.1c.

6. Lastly, we consider the case where $v^*$ contains an edge to a node that is not connected to the rest of the graph. However for each of Figure 3.2f to Figure 3.2j, an edge between $v^*$ and the unconnected vertex will result in a graph which contains either Figure 3.1a or Figure 3.1b as a subgraph. If there is no edge between $v^*$ and the unconnected vertex, then cases 1 through 5 all deny the existence of $G(V, E)$.

□

Assuming that loci are independent, each locus must be checked for incompatibilities which adds a factor $O(m)$ to testing group incompatibility. Therefore, capturing all of the information given by forbidden half-sibling substructures would require enumerating all $O(n^6)$ sets of six individuals and would take $O(n^6 m)$ time by brute force, which is infeasible for large populations.

We have proved that minimal half-sibling incompatibilities with four, five, and six individuals exist, but how commonly do they occur and how often do they matter? To determine

| Variable parameter | Parameter settings | Frequency of incompat. $x$-lets $x=3$ | $x=4$ | $x=5$ | $x=6$ |
|---|---|---|---|---|---|
| $k$: number of alleles | 5 | 0.765 | 0.249 | 0.027 | 0.001 |
| | 6 | 0.839 | 0.205 | 0.016 | 0.000 |
| | 7 | 0.876 | 0.159 | 0.019 | 0.000 |
| | 8 | 0.922 | 0.126 | 0.012 | 0.000 |
| | 9 | 0.951 | 0.089 | 0.005 | 0.000 |
| | 10 | 0.949 | 0.081 | 0.006 | 0.000 |
| $f$: family size | 5 | 0.617 | 0.088 | 0.004 | 0.000 |
| | 10 | 0.994 | 0.772 | 0.499 | 0.021 |
| | 15 | 1.000 | 0.993 | 0.968 | 0.270 |
| | 20 | 1.000 | 0.999 | 0.998 | 0.774 |

Table 3.1: Frequency of incompatible $x$-lets for $x \in \{4, 5, 6\}$ in 1000 randomly chosen families with fixed alleles or fixed family size.

how {4, 5, 6}-let incompatibilities are affected by the number of alleles at a locus, we enumerate populations of all $\binom{k}{2} + k$ individuals for $k = \{5, 6, \ldots 10\}$ alleles. From these populations, we sample 1000 sets of size 6 uniformly at random without replacement and count how many sets contain each type of minimal incompatibility. A similar test is conducted on 1000 sets of sizes {5, 10, 15, 20} with $k$ fixed at 7 alleles. The results are given in Table 3.1. The number of incompatible {4, 5}-lets tend to decrease as the number of alleles grows. However, for realistic allele pool sizes, the percentage of incompatible families due to quadruplets is high, at around 20%. Furthermore, when alleles are held constant, the percentage of incompatible {4, 5, 6}-lets increases substantially with sib-group size.

The results of a similar experiment answer the second question of how often {4, 5, 6}-let incompatibilities matter. In this experiment, incompatible $x$-lets for $x \in \{4, 5, 6\}$ are only counted if a family contains no incompatible $y$-lets for $y < x$. The result of the experiment, given in Table 3.2, indicate that enumerating only triplets will miss several incompatibilities, particularly in the range of unique alleles which would occur in real populations. Unsurprisingly, incompatible triplets become more common as the number of individuals in a set increases. Therefore, triplets should not exclusively be used to find incompatible families unless the families are very large, but may provide an adequate similarity score approximation.

## 3.2 Allele Similarity

As a simple alternative to a triplet-based similarity measure, we may use a pairwise measure based on counting the number of shared alleles at each locus of a pair of individuals. Given two individuals, each with $m$ loci, this similarity function mathces alleles of a pair of individuals. The similarity is the number of matches across all loci of the two individuals.

| Variable parameter | Parameter settings | Frequency of incompat. $x$-lets | | | |
|---|---|---|---|---|---|
| | | $x = 3$ | $x = 4$ | $x = 5$ | $x = 6$ |
| $k$: number of alleles | 5 | 0.776 | 0.05 | 0.000 | 0.003 |
| | 6 | 0.856 | 0.029 | 0.000 | 0.000 |
| | 7 | 0.894 | 0.019 | 0.000 | 0.000 |
| | 8 | 0.919 | 0.009 | 0.000 | 0.000 |
| | 9 | 0.950 | 0.007 | 0.000 | 0.000 |
| | 10 | 0.955 | 0.009 | 0.000 | 0.000 |
| $f$: family size | 5 | 0.632 | 0.043 | 0.002 | 0.000 |
| | 10 | 0.995 | 0.000 | 0.000 | 0.000 |
| | 15 | 1.000 | 0.000 | 0.000 | 0.000 |
| | 20 | 1.000 | 0.000 | 0.000 | 0.000 |

Table 3.2: Frequency of incompatible $x$-lets for $x \in \{4, 5, 6\}$ in 1000 randomly chosen families with fixed alleles or fixed family size. Incompatible $x$-lets are only counted if the family does not contain any incompatible $y$-lets for $y < x$.

For example, the pair of individuals $x = [(1, 2), (2, 2), (1, 3)]$ and $y = [(1, 1), (2, 2), (2, 3)]$ has a similarity of $s_{xy} = 4$, since for $(1, 2)$ and $(1, 1)$, we do not double count the similarity between the 1 alleles.

To see why this approach is useful, let $X$ be the random variable that represents the number of shared alleles between two individuals at a single locus. We can calculate the expected number of alleles for each relationship type assuming an even allele distribution.

$$\mathbf{E}[X|\text{full siblings}] = \frac{8k^2 + 4k + 1}{8k^2} \tag{3.1}$$

$$\mathbf{E}[X|\text{half-siblings}] = \frac{2k^3 + 5k^2 + k + 1}{4k^3} \tag{3.2}$$

$$\mathbf{E}[X|\text{unrelated}] = \frac{3k^2 - k - 1}{k^3} \tag{3.3}$$

Eliminating allele double-counting, we end up with Eq. 3.1, 3.2, and 3.3. For full-siblings, the expected number of shared alleles approaches 1 as the number of alleles grows,

for half-siblings, the expectation approaches $\frac{1}{2}$, and the expectation approaches 0 for unrelated individuals. For a population with $m$ loci, the expected number of shared alleles for any two individuals is $m \cdot \mathbf{E}[X]$ and strongly concentrated around that value.

**Theorem 2.** *The probability that a pairwise allele similarity deviates far from its mean decreases exponentially as the number of loci increases.*

*Proof.* Let $X$ be a random variable as described above. For independent loci, the allele similarity $X_i$ is the allele similarity of the $i$'th locus with $0 \leq X_i \leq 2$ for $1 \leq i \leq m$. By application of Hoeffding's inequality to the mean allele similarity $\overline{X} = \sum_{i=0}^{m} X_i/m$,

$$\Pr(|\overline{X} - \mathbf{E}[\overline{X}]| \geq t) \leq 2 \cdot \exp\left(-\frac{t^2 m}{2}\right)$$

□

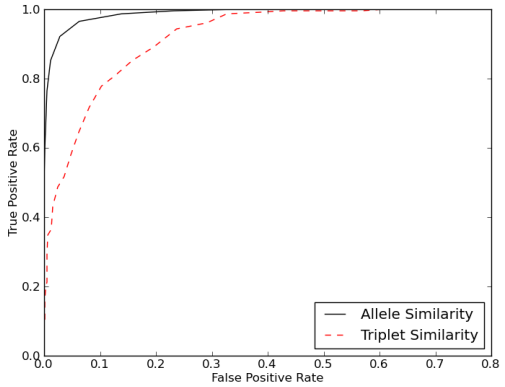
The comparatively high expected value for full-siblings also makes this similarity measure an acceptable method for calculating full-sibling similarity as well, though we focus on half-siblings here. Additionally, computing allele similarity takes $O(n^2 m)$ time, compared to the $O(n^3 m)$ time to enumerate and compare triplets.

## 3.3 Experimental Results

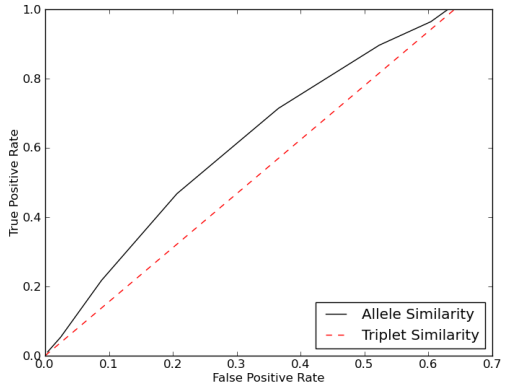
We examine how well triplet and allele similarities perform using several test sets that are designed to enable independent tests of changes in alleles, loci, population size, and family size. Ideal similarity measurements maximize the score between pairs of true full or half-siblings and suppress the scores of unrelated pairs of individuals. We assess the

effectiveness of each similarity measure by counting the number of true and false positives at each similarity threshold. For each threshold of similarity, the pairs of individuals with similarity greater than or equal to the threshold are classified as positive identifications and the rest are negative identifications. Pairs of individuals which are true half-siblings and above the threshold are true positives, while incorrect pairs of individuals above the threshold are counted as false positives. Plotting the true positive rate versus false positive rate at each threshold produces a receiver operating characteristic (ROC) curve. The ROC curve exposes the tradeoff between true positive and false positives at each threshold. Ideally, clustering algorithms prefer thresholds with high true positive rates and comparitively low false positive rates. Unfortunately, these values cannot be computed when the true population is unknown.

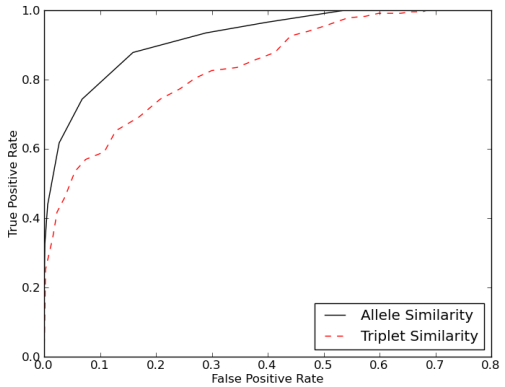
We plotted allele similarity and triplet similarity ROC curves against each other to determine which measure was better in different situations. Figure 3.4 shows outcomes where similarity measurements did well and where they did poorly as well as instances where allele similarity outperformed triplet similarity and *vice versa*. Both similarity measurements perform well when the number of loci or discrete alleles is high. The ideal ROC curve is one where the true positive rate reaches 1.0 before any false positives are introduced, but this rarely occurs for either similarity measurements. Figure 3.4a shows a test with a large number of loci which results in a nearly ideal allele similarity. Figure 3.4b shows the opposite case with two alleles where the true and false positive rates are growing by about the same amount at each threshold. When the number of distinct alleles is below three, the entire population can form a valid half sibling family. In general, poor performance was observed when allele or locus counts were low. Figure 3.4c, which represents a test case with 40 individuals, 6 alleles, and 10 loci, is more representative of a population that may actually occur in the wild. In these average cases, the allele similarity measure



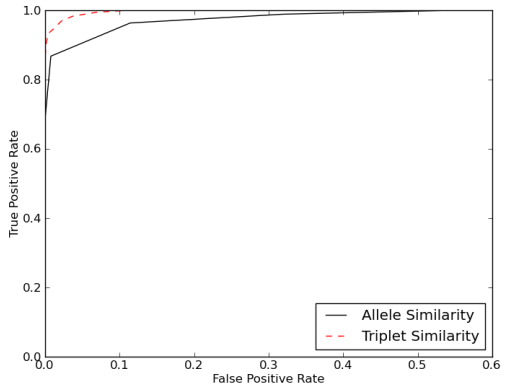

(a) 40 individuals, 6 alleles, 20 loci: The allele similarity ROC curve is very close to optimal




(b) 40 individuals, 2 alleles, 6 loci: Both similarity measures do poorly since there are only two distinct alleles per locus




(c) 40 individuals, 6 alleles, 10 loci: The allele similarity outperforms triplet similarity in average instances




(d) 40 individuals, 20 alleles, 6 loci: The triplet similarity outperforms allele similarity for very high distinct allele counts

Figure 3.4: ROC curves illustrating cases where each similarity measure does well or poorly

almost always outperformed triplet similarity for half-siblings. Lastly, Figure 3.4d shows an instance with 20 alleles where triplet similarity dominates allele similarity at all thresholds. For large numbers of distinct alleles, triplet similarity bests allele similarity; however, the allele range where this is true is unrealistically large for real populations. In almost all

cases allele similarity performed better than triplet similarity for high loci counts.

As microsatellite technology gets cheaper, one can expect the number of measured loci to grow and the number of distinct alleles to stay small. We have derived the expected number of shared alleles between full siblings, half-siblings, and unrelated individuals. Moreover, Theorem 2 proves that allele similarity will grow in accuracy as the number of independent loci increases. We have also shown experimentally that, when the number of distinct alleles is small, allele similarity can provide a more accurate similarity score than triplet similarity. When comparing half-siblings, allele similarity provides a more accurate similarity measure than the triplet similarity and should be prefered in most cases.

# Chapter 4

# Metrics for Comparing Sibship Partitionings

In order to analyze an algorithm's ability to reconstruct sibships, we must have a good metric for quantifying the difference between two population assignments. Such a metric is important for determining how close an algorithm's solution is to simulated or known sibship structures, but it can also be used to reduce error by comparing multiple candidate partitionings and establishing a confidence rating for individuals of a population when the true population structure is unknown [12]. One method for finding the difference between two partitionings was proposed by Painter as the number of individuals that must be reassigned in either partitioning until the two partitionings are identical [27]. A later method proposed by Almudevar *et al.* defines a distance metric between two populations as the minimum number of individuals which must be removed from both populations in order for the two to be identical [2]. These two formulations are identical [1] and the value of this measure can be computed by computing a maximum matching between the

two partitionings. Although this maximum matching method is widely used in sibship reconstruction literature and in bioinformatics in general [14, 24], it offers only a coarse estimate of the difference between two partitionings. Therefore, we advocate a method from information theory, called variation of information, which yields a better understanding of the difference between two population partitionings.

## 4.1 Maximum Matching

As previously stated, Painter [27] proposed the partition distance based on maximum matching; however, a polynomial time algorithm for computing the maximum matching was not given until five years later by Gusfield who stated but did not prove that the best algorithm at the time, due to Almudevar *et al.* [2], had an exponential worst-case runtime. Gusfield reduced the problem to an instance of the assignment problem [14], which has a known polynomial worst-case time complexity. Konolovalov *et al.* presented an algorithm, which was also based on reduction to the assignment problem, with an $O(n^3)$ time bound.

The classical assignment problem is: given an $m \times n$ matrix $M$, select cells of $M$ such that the sum of the cells are maximal, and no row or column in $M$ has more than one selected cell. For two partitionings of the data, $\mathcal{P}$ and $\mathcal{P}'$, create an instance of the assignment problem by forming $M$ with $|\mathcal{P}|$ rows and $|\mathcal{P}'|$ columns corresponding to the clusters in $\mathcal{P}$ and $\mathcal{P}'$ respectively. For each $P_i \in \mathcal{P}$ and $P'_j \in \mathcal{P}'$, $M_{i,j} := |P_i \bigcap P'_j|$. Gusfield proved that the individuals which must be removed are exactly the individuals in the symmetric difference of $P_i$ and $P'_j$ for all selected cells $(i, j)$.

When comparing an algorithm's solution to a known population structure, the partition distance takes on a very literal interpretation as the number of correctly placed individuals. However, the partition distance due to maximum matching does not convey

| $\mathcal{P}$ | $\mathcal{P}'_1$ | $\mathcal{P}'_2$ |
|---|---|---|
| 1, 2, 3, 4, 5 | 1, 2 | 1, 2 |
| 6, 7, 8, 9, 10 | 6, 7, 8, 9, 10 | **3**, 6, 7, 8, 9, 10 |
| 11, 12, 13, 14, 15 | 11, 12, 13, 14, 15 | **4**, 11, 12, 13, 14, 15 |
| 16, 17, 18, 19, 20 | 16, 17, 18, 19, 20 | **5**, 16, 17, 18, 19, 20 |
| | **3, 4, 5** | |

Table 4.1: True partitioning $\mathcal{P}$ with candidate sibling partitionings $\mathcal{P}'_1$ and $\mathcal{P}'_2$. $\mathcal{P}'_1$ and $\mathcal{P}'_2$ both receive the same partition distance score from maximum matching, even though $\mathcal{P}'_1$ is preferable. The individuals which ought to be removed due to maximum matching are bolded.

the extent to which misplaced individuals are incorrect. An example of this lack of resolution is given in Table 4.1. In the example, $\mathcal{P}'_1$ and $\mathcal{P}'_2$ both receive a maximum matching score of $\frac{17}{20}$, or 0.85. The best solution for both partitionings would be to remove individuals $\{3, 4, 5\}$ from the population, making them equivalent to $\mathcal{P}$. However, even though both partitionings have the same distance, $\mathcal{P}'_1$ is peferable to $\mathcal{P}'_2$ because its incorrectness is due only to the failure to join clusters $\{1, 2\}$ and $\{3, 4, 5\}$. The algorithm which produces solution $\mathcal{P}'_2$ not only fails to join $\{1, 2\}$ with $\{3, 4, 5\}$, it incorrectly joins each individual $\{3, 4, 5\}$ into the wrong family. Moreover, these splits are common in some algorithms, such as in COLONY and COLONY 2.

Partition distance by maximum matching is a good starting point since its solution produces a list of incorrect individuals. However, it is unable to measure the degree to which misplaced individuals are incorrect. In pedigree reconstructions, a split is preferable to an incorrect join since the split does not claim that unrelated individuals are related, yet both receive the same score under maximum matching. We have given a simple example of when the maximum matching partition distance fails to pick the best partitioning. Meila [25] gives another example. Therefore, a preferable metric would take the structure of the solution into account.

## 4.2 Variation of Information

We advocate an alternative information theoretic metric called variation of information, which was first proposed by Meila [25], to overcome the lack of resolution in the partition distance's score. Variation of information measures the entropy, or degree of disorder, within two partitionings and the mutual information, or how much both partitionings have in common, to produce its score. VI measures how much knowing the partition element an individual belongs to in one partition explains where it is in the other partition, and *vice-versa*. Unlike maximum matching, in which higher scores correspond to closer results, variation of information measures the degree to which two partitionings vary. Therefore, lower scores are preferable.

In order to determine the variation of information between two population partitionings, the amount of disorder within each cluster must be quantified. The amount of disorder is known as the *entropy* of a partitioning. If a random individual is chosen from the population, the entropy tells us the amount of uncertainty we have about which cluster that individual will be chosen from. For instance, if a partitioning were to contain exactly one cluster which assigned all population members as siblings, then there would be no uncertainty: the randomly chosen individual must have been chosen from the single cluster, so the entropy is 0. However, if an algorithm were to return a partitioning where each cluster held exactly one individual, then the uncertainty about which cluster the individual was chosen from would be much higher.

Define the true partitioning of a population into families as $\mathcal{P}$ and an algorithm's partitioning as $\mathcal{P}'$, with $p := |\mathcal{P}|$ and $p' := |\mathcal{P}'|$. Compute the probability of an individual selected uniformly at randomly from our population of size $n$ as $P(i) = |P_i|/n$ for each $P_i \in \mathcal{P}$ and similarly for $\mathcal{P}'$. Using the two random variables, we can compute the entropy,

which is always non-negative, of $\mathcal{P}$ and $\mathcal{P}'$, denoted $H(\mathcal{P})$, and $H(\mathcal{P}')$.

$$H(\mathcal{P}) = -\sum_{i=1}^{p} P(i) \log P(i) \tag{4.1}$$

*Mutual information*, denoted $I(\mathcal{P}, \mathcal{P}')$, measures how much information is shared between two partitionings. Intuitively, mutual information is a measure of the amount of information one partitioning would give about the structure of the other. If the amount of uncertainty in two partitionings is high, but the partitions are very similar, then knowing one partition gives significant insight into the structure of another. However, if the amount of uncertainty within each partitioning is low, then the amount of information gained will be less. Mutual information is, therefore, dependent on the joint distribution of the random variables for $\mathcal{P}$ and $\mathcal{P}'$, given by $P(i, i') = (|P_i \bigcap P'_i|)/n$.

$$I(\mathcal{P}, \mathcal{P}') = \sum_{i=1}^{p} \sum_{i'=1}^{p'} P(i, i') \log \frac{P(i, i')}{P(i)P'(i')} \tag{4.2}$$

The variation of information between partitionings $\mathcal{P}$ and $\mathcal{P}'$, which we will call $VI(\mathcal{P}, \mathcal{P}')$, may now be computed in $O(n + p \cdot p')$ time.

$$VI(\mathcal{P}, \mathcal{P}') = (H(\mathcal{P}) - I(\mathcal{P}, \mathcal{P}')) + (H(\mathcal{P}') - I(\mathcal{P}, \mathcal{P}')) \tag{4.3}$$

The VI between two partitionings is 0 if and only if the two partitionings are identical and smaller VI corresponds to more similar partitionings. Like entropy, the VI is always non-negative. VI has a tight upper bound of $\log n$ [25]; therefore, we will normalize VI to a value in $[0, 1]$ before reporting the score for each of our trials in later sections.

Returning to the toy example with $\mathcal{P}$, $\mathcal{P}'_1$, and $\mathcal{P}'_2$ from Table 4.1, the normalized variation of information, calculated using $\log_2$, between the real partitioning and each of

| | $\mathcal{P}$ | $\mathcal{P}'_1$ | $\mathcal{P}'_2$ |
|---|---|---|---|
| Entropy ($H$) | 2.0 | 2.2427 | 1.8955 |
| $I(\mathcal{P}, \mathcal{P}')$ | | 2.0 | 1.4150 |
| Normalized $VI(\mathcal{P}, \mathcal{P}')$ | | 0.0562 | 0.2465 |

Table 4.2: Variation of information calculation for example from Figure 4.1 using base 2 logarithms. VI correctly identifies $\mathcal{P}'_1$ as the better of the two candidate partitions.

the candidate partitionings is given in Table 4.2. As expected, $\mathcal{P}'_1$ has a higher entropy than $\mathcal{P}'_2$, which has less groups and more members in most groups. However, the mutual information for $\mathcal{P}'_1$ is much higher, due to the fact that $\mathcal{P}$ and $\mathcal{P}'_1$ are identical except for a split, whereas the algorithm for $\mathcal{P}'_2$ mixes the three individuals across incorrect clusters. Finally, after normalization, $VI(\mathcal{P}, \mathcal{P}'_1) = 0.0562$ while $VI(\mathcal{P}, \mathcal{P}'_2) = 0.2465$.

VI is a powerful metric because it is able to discern when individuals are misplaced and, as demonstrated in our toy example, is a more appropriate measurement of the difference between two partitionings, particularly in our context. The VI metric can be computed quickly and gives better information than the current maximum matching solution. Therefore, results in later sections will be reported in terms of their VI score.

## 4.3 Partition Distance for Half-Siblings

Both maximum matching and variation of information are easy to calculate when there are only two partitionings, which is the case for full-sibling reconstruction. Unfortunately, half-sibling partitioning comparison requires a total of four partitionings: two maternal and two paternal. Furthermore, microsatellite DNA samples do not provide information about the sex of the parent that each allele was inherited from. In half-sibling solutions where the sex of each cluster is known, the overall VI is the average VI between the maternal

partitionings $\mathcal{M}$ and $\mathcal{M}'$ and paternal partitionings $\mathcal{P}$ and $\mathcal{P}'$ given in Eq. 4.4.

$$VI_{HS} = \frac{VI(\mathcal{M}, \mathcal{M}') + VI(\mathcal{P}, \mathcal{P}')}{2} \Big/ \log n \tag{4.4}$$

In most cases, however, we do not know which partitioning each cluster belongs to. Assuming that an algorithm enforces the Mendelian requirement that an individual must have one genetic father and mother, it is possible to reconstruct feasible partitionings where no individual appears in the same partitioning more than once: there is a partitioning of the individuals by mothers and one by fathers. This will be discussed in detail later. However, once the requirement is enforced, each cluster must be labeled either maternal or paternal so that the resulting partitionings can be compared.

Each cluster in an algorithm's solution will force a set of clusters to the opposite sex. For an algorithm where each individual appears in exactly two clusters, the rule is simple: for each individual $i$ which appears in clusters $C_j$ and $C_k$, $C_j$ and $C_k$ cannot have the same sex. Enforcing this rule across all individuals results in two partitionings that are equivalent to a bipartite graph with vertices representing clusters and edges representing clusters which cannot have a parent with the same sex. One side of the bipartite graph is roughly maternal, while the other side is roughly paternal. A major concern is that this graph may not be fully connected: it may have multiple unattached components. A fully connected graph can be expected when matings are highly polygamous, which due to mating transitivity, will force many of the clusters into one connected component. A fully connected graph is good because it reduces the opportunity for error when deciding the sex of the parent for each cluster. If the graph is fully connected, then either the left side is maternal and the right side is paternal or *vice versa*. On the other hand, many connected components forces us to make many decisions about which clusters are labeled maternal or paternal. There are two cases to consider: one where one half-sibling reconstruction has

**Algorithm 1** Variation of information for half-siblings

```
1: function HALF-SIB VI(B, M, P)
2:     for H ∈ B do                                  ▷ H is a connected component in B
3:         H+, H− ← (female, male)                   ▷ H+ is left side of H, H− is the right side.
4:     end for
5:     for H ∈ B do
6:         vi0 ← VI(B, M, P)
7:         H+, H− ← (male, female)
8:         vi1 ← VI(B, M, P)                         ▷ Compute VI with switched sexes
9:         if vi0 < vi1 then
10:            H+, H− ← (female, male)
11:        end if
12:    end for
13: end function
```

known sexes for each cluster and one where neither reconstruction has known sexes.

### 4.3.1 Comparing to a Reference Sibship

The first case arises when comparing an algorithm's solution to known reference partitionings. In the reference partitionings it is assumed that the sex corresponding to each partition is known. If this is the case, then it is easy to find the two partitionings from the clusters produced by an algorithm which minimize the total variation of information.

The determination of each cluster's sex can be done greedily by the algorithm described in Algorithm 1. Each connected componentof the bipartite half-sibling graph must be assigned a sex. For each connected component, the greedy heuristic calculates the VI with the left partitioning $H^+$ as maternal and again with $H^+$ paternal. The parental sex assignment with the lowest VI is chosen for each connected component. Minimizing the overall VI is a natural objective since it is assumed that the algorithm is trying to reconstruct half-sibling clusters which are correct. Additionally, the greedy algorithm will

produce the minimal VI.

**Theorem 3.** *Algorithm 1 produces a minimal variation of information given the constructed clusters.*

*Proof.* By Eq. 4.3 and Eq. 4.4

$$VI_{HS} \propto H(\mathcal{M}) + H(\mathcal{M}') + H(\mathcal{P}) + H(\mathcal{P}') + I(\mathcal{M}, \mathcal{M}') + I(\mathcal{P}, \mathcal{P}') \tag{4.5}$$

$H(\mathcal{M}')$ and $H(\mathcal{P}')$ change when the sexes of the connected component are swapped, but entropy is the sum of terms which depend on exactly one cluster. Therefore, the total entropy $H(\mathcal{M}) + H(\mathcal{M}') + H(\mathcal{P}) + H(\mathcal{P}')$ is preserved regardless of which clusters are labeled maternal and which are paternal.
Mutual information also depends on the sex assigned to each cluster, but by Eq. 4.2, the choice of sex for each connected component will not affect the other connected components as long as the sexes of the reference partitionings stay fixed. □

The ability to minimize VI when comparing against a reference solution is important since most algorithms guage their effectiveness by comparing solutions to simulated or real-world known populations. Unfortunately, the greedy assumption made in Algorithm 1 depends on knowing the structure of the two partitionings from the reference solution. The case is not as clear when comparing two candidate solutions.

### 4.3.2 Comparing Two Candidate Sibships

When candidate half-sibling partitionings are compared against a reference solution, it is easy to determine the sex of the parent of each candidate cluster with a greedy algorithm

because the female and male partitionings of the reference clusterings are fixed. However, some applications rely on the comparison of candidate partitionings to reconstruct more accurate partitionings [12]. Unlike full-sibships, in which there is only one partitioning for each solution, half-sibships require a maternal and paternal partitioning with each individual appearing once in each.

When neither solution has defined sexes for its clusters, the mating transitivity described earlier creates dependencies where one connected component's sex choice can affect the VI calculation at other connected components. More concretely, suppose that the assignments $H_i^+ :=$ maternal and $H_j^+ :=$ maternal minimize the VI for the $i^{th}$ and $j^{th}$ connected component respectively. When neither solution has sexes assigned *ab initio*, it is possible for the assignment $H_i^+ :=$ maternal to preclude the next assignment $H_j^+ :=$ maternal. As a result, any VI calculation algorithm needs to make optimization decisions about which to assign as maternal and which to assign as paternal. These decisions may be very difficult to make, especially in highly polygamous populations, since any decision about the sex of one parent would affect the decision about sex of many others. At the present time, it is unknown whether assigning sexes to clusters when neither of the solutions have sexes defined is *NP*-hard. Therefore, extending classical metrics for comparing full-sibling solutions works well as a guage of the accuracy of algorithms when compared to some known population structure, but not for comparing two candidate results where the partitionings are unknown. When none of the partitionings have assigned sexes and the polygamy rate is high, there is a strong chance that the connected components will exhibit mating transitivity that makes it difficult to assign a VI score.

# Chapter 5

# Half-Sibling Reconstruction Algorithms

## 5.1 Integer Programming Formulation

In previous Section 2.2 we discussed the HS-MSC IP, which constructs half sibling partitionings by minimizing the total number of family clusters and gave an example where the IP produces incorrect clusters. In the example reconstruction given in Section 2.2, the HS-MSC algorithm produced paternal clusters. In this section, we propose an IP which enforces Mendelian genetic laws and gives a full solution where each individual belongs to a maternal and paternal clustering. Unlike HS-MSC, the new IP minimizes the total number of clusterings across both the maternal and paternal partitioning. A major drawback of the HS-MSC is that complete maternal and paternal partitionings must be inferred from the minimum set cover solution. The authors do not provide a method of doing this. Furthermore, because the HS-MSC does not produce separate partitionings for mothers and

$$\textbf{minimize} \quad \sum_{s\in\{0,1\}} \sum_{j\in J^{(s)}} \left( z_j^{(s)} + \sum_{i<n} x_{i,j}^{(s)} \varepsilon_{i,j} \right)$$

**subject to**

$$\sum_{s\in\{0,1\}} y_{i,k,l}^{(s)} - a_{i,k,l} = 0, \qquad 0 \le i < n, 0 \le l < m, k \in K$$

**for each sex** $s$

$$\begin{aligned}
z_j^{(s)} - x_{i,j}^{(s)} &\ge 0, & 0 \le i < n, j \in J^{(s)} \\
\sum_{j\in J^{(s)}} x_{i,j}^{(s)} &= 1, & 0 \le i < n \\
\sum_{k\in K} y_{i,k,l}^{(s)} &= 1, & 0 \le i < n, 0 \le l < m \\
\sum_{k\in K} p_{j,k,l}^{(s)} &\le 2, & 0 \le i < n, j \in J^{(s)} \\
p_{j,k,l}^{(s)} - x_{i,j}^{(s)} - y_{i,k,l}^{(s)} &\ge -1, & 0 \le i < n, j \in J^{(s)}, k \in K, 0 \le l < m \\
z_{j+1}^{(s)} - z_j^{(s)} &\le 0, & j \in J^{(s)}
\end{aligned}$$

Figure 5.1: 0/1 integer program to find minimum number of parents needed to explain a population.

fathers, it is possible for the minimum set cover to include some clusters from both sexes, which makes it difficult to say anything useful about mating patterns. Additionally, since there is only one partitioning of the individuals, many true half-sibling families will fail to be clustered by the HS-MSC algorithm. On the other hand, the new IP always produces a valid maternal and paternal partitioning and the IP makes a clear distinction about which is maternal and which is paternal.

An IP formulation with the new objective is given in Figure 5.1 and is suitable for small populations. Separate variables are kept for the maternal and paternal partitionings. These variables are indexed by a parameter $(s)$, which corresponds to the sex of the

partitioning. The objective minimizes the total number of clusterings needed to explain a population. The index $i$ denotes the individual, $j$ the cluster, $k$ the allele, and $l$ the locus, for all variables. The $a_{i,k,l}$ constants represent how many of each allele $k$ individual $i$ contains at locus $l$: for homozygotes, this constant is 2. The $y^{(s)}_{i,k,l}$ variables track the actual alleles for each individual at each locus. There are up to four $y$ variables for each locus of each individual. The $x$ variables map an individual $i$ to a cluster $j$. Finally, the $p$ variables store the alleles that each parent must have at each locus.

The first constraint forces individuals to join maternal and paternal clusters that satisfy the individual's allele requirements. There are two instances of each of the remaining constraints: one for each sex's partitioning. The first of the "per-sex" constraints sets the $z$ variable representing cluster $j$ to one if cluster $j$ is non-empty. Minimizing $z$ in the objective forces individuals to form as few families as possible: in particular, to minimize the number of parents. The next constraint guarantees that each individual will belong to exactly one maternal and paternal cluster. The fourth constraint guarantees that the parent of each cluster has at most two alleles and the fifth constraint ensures that each individual receives one allele from each parent.

Like the HS-MSC, the new IP requires a guess about the number of families $|J^{(s)}|$ in each partition. The upper bound for each $j^{(s)}$ is $n$ since each individual has exactly one mother and father. However, large values of $j(s)$ greatly increase the size of the solution space and result in many optimal solutions since a family can be placed in any empty cluster, which results in a combinatorial explosion of equivalent optimal solutions. Smaller guesses about cluster size greatly reduce the number of variables and constraints, leading to a faster solution. However, if the guess is too small, the IP may never find a valid solution. Adding the sixth constraint forces the chosen clusters, $z_i$ variables, to be contiguous so that all of the empty clusters are forced together. Including this constraint reduces the

number of equivalent solutions to $O(|Z|!)$, where $|Z|$ is the number of non-empty clusters in the optimal solution. We use random perturbations to force a single optimal solution. We generate a random perturbation matrix $\mathcal{E} \in [1 \times 10^{-10}, 1 \times 10^{-9}]^{i \times j}$ and add the product $x_{i,j}^{(s)} \varepsilon_{i,j}$ to the objective equation for all $i$, $j$, and $s$. These small perturbations do not affect the optimal number of clusters since they are several orders of magnitude smaller than the $z_j$ variables; however, they break ties between equivalent solutions and force a single optimal solution.

## 5.2 The SibJoin Heuristic

SibJoin, which uses hierarchical clustering to reconstruct half-sibling families, is an alternative to IP and likelihood methods. Instead of searching through large sections of the feasible solution space, SibJoin uses heuristics to determine individuals or families to join. As a result, SibJoin is thousands of times faster than likelihood and IP solutions and can be used to reconstruct populations which have previously been unsolvable due to the population size. We describe the SibJoin algorithm, and test it against COLONY 2 on simulated and real population data sets and against the HS-MSC integer program on real biological populations.

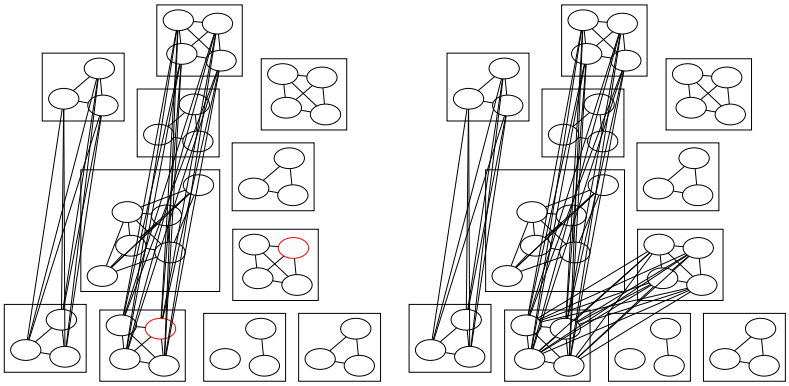
Some clustering algorithms rely on measurements of similarity between individuals. We denote the similarity between individuals $x$ and $y$ as $s_{xy}$ and the similarity between clusters $C_i$ and $C_j$ as $sim(C_i, C_j)$. The terms partitioning and clustering may be used interchangeably.endfigure

| Variable parameter | Parameter settings | Average # bad joins | Error after % of total joins 20% | 40% | 60% | 80% |
|---|---|---|---|---|---|---|
| $k$: number of alleles | 2 | 33.8 | 0.112 | 0.201 | 0.393 | 0.632 |
| | 5 | 7.8 | 0.056 | 0.090 | 0.146 | 0.345 |
| | 10 | 0.1 | 0.000 | 0.000 | 0.000 | 0.000 |
| | 15 | 0.0 | 0.000 | 0.000 | 0.000 | 0.000 |
| | 20 | 0.0 | 0.000 | 0.000 | 0.000 | 0.000 |
| $m$: number of loci | 2 | 27.4 | 0.267 | 0.462 | 0.589 | 0.718 |
| | 5 | 5.3 | 0.248 | 0.299 | 0.430 | 0.613 |
| | 10 | 0.6 | 0.000 | 0.000 | 0.000 | 1.000 |
| | 15 | 0.1 | 0.000 | 0.000 | 0.000 | 0.000 |
| | 20 | 0.0 | 0.000 | 0.000 | 0.000 | 0.000 |
| $n$: population size | 10 | 0.1 | 0.000 | 0.000 | 0.000 | 0.000 |
| | 50 | 5.4 | 0.058 | 0.070 | 0.122 | 0.280 |
| | 100 | 19.9 | 0.026 | 0.045 | 0.087 | 0.298 |
| | 200 | 54.3 | 0.026 | 0.063 | 0.177 | 0.366 |
| $f$: family size | 1 | 63.1 | 0.082 | 0.257 | 0.426 | 0.716 |
| | 5 | 13.8 | 0.014 | 0.055 | 0.156 | 0.291 |
| | 10 | 2.2 | 0.088 | 0.175 | 0.260 | 0.429 |
| | 20 | 2.2 | 0.551 | 0.631 | 0.774 | 0.774 |

Table 5.1: Fraction of total bad joins accumulated after SibJoin is 20%, 40%,..., 80% complete.

### 5.2.1 Joining Families

SibJoin begins with $2n$ clusters, each of which contains a single individual. Every individual appears in exactly two clusters, representing the maternal and paternal half-sib groups. A variation of single linkage clustering is used to determine which clusters to join. Single linkage clustering is a form of agglomerative clustering that determines the similarity of two clusters $C_i$ and $C_j$ by computing $sim(C_i, C_j) = \max_{x \in C_i, y \in C_j} s_{xy}$, and then joining groups with high similarity. A sample join is demonstrated in Figure 5.2. Ties in similarity are broken by joining the groups with the highest combined number of members first since large compatible half-sibling groups are more likely to be related than small groups.



(a) Identify half-sibs (in red). (b) Merge their half-sib clusters

Figure 5.2: Demonstration of a successful iteration of SibJoin. Nodes represent individuals, edges represent a half or full-sibling relationship constructed by the algorithm, and nodes which share a box represent true full-siblings.

Traditional clustering techniques mandate that only one copy of each individual is allowed. SibJoin implements a modified form of single linkage clustering which places restrictions on which clusters may be joined according to Mendelian compatibility constraints and handles the multiple copies of individuals necessary to reconstruct maternal and paternal half-sibling structures.

Single linkage clustering is chosen because of the assumption that individuals with high allele similarity are very likely to be half or full siblings. The heuristic does well in practice. For each simulated test case, we analyze the number of incorrect family merges that SibJoin makes during the first 20, 40, 60, 80, 100% of its joins and report what fraction of the total error was accumulated by each threshold. In the experiment, populations which contained no errors were excluded from the average to avoid biasing the error downward. The average number of incorrect joins is, however, averaged over all trials. The results shown in Table 5.1 verify that most of the incorrect joins happen toward the end of the clustering process when joins are selected between individuals with low allele similarity. In

most test cases, SibJoin accumulates less than 10% of its total error by the time it is 25% complete and less than 45% of its total error once it is 80% complete. There are instances, such as family size 20, where SibJoin appears to perform poorly during early joins; however, in these cases, the total number of bad joins is low. As a result, one incorrect join accounts for a large percentage of the total error. Unsurprisingly, when the total number of errors is very large, SibJoin also makes early mistakes. Both the total errors and the early errors result from SibJoin not having enough information to make informed early decisions: for example, when the number of distinct alleles or total loci is very small.

SibJoin's success comes from two observations. First, in order for bad joins to occur between any pair of individuals $i$ and $j$, the similarity between $i$ and $j$ would need to be larger than the similarity between $i$ and each of $i$'s real half-siblings, and likewise for $j$. Secondly, as clusters grow, the odds that two unrelated clusters form a compatible half-sibship rapidly diminishes, even if there are surprisingly similar members of these clusters.

Joining must only occur if two clusters form a valid half-sibship. At the initialization of the algorithm, each individual is assigned a feasible parent set with size at most $O(k)$ per locus. Each join results in a parent set which is the intersection of the parent set from the two joined clusters. If the intersection produces the null set, then there is no parent which can explain the new cluster and the join is rejected. Therefore, testing whether or not a join is valid takes $O(km)$ time. When a site experiences allelic dropout, SibJoin makes no assumptions about its parental restrictions; however, sites with genotype $(*, *)$ are never counted toward allele similarity between individuals.

Unlike crisp clustering methods which mandate that each individual appear in exactly one cluster, a half-sibling solution contains both a maternal and paternal group for each individual. We enforce the restriction that any set of individuals sharing both a maternal and paternal cluster must be compatible full-siblings under the 4-allele and 2-allele prop-

erties by maintaining a clustering of full-siblings. Because incompatible full-sibling groups are less likely than incompatible half-sibling groups of the same size, at each similarity step SibJoin joins clusters which form valid full-sibships first. The complete algorithm for deciding the order of joins is given in Algorithm 2 where $simHash$ is a hash $H(key, value)$ such that $simHash(k)$ is a list of pairs of individuals with allele similarity $k$ and $\lambda$ and $\Lambda$ are the sets of half-sibling and full-sibling clusters respectively.

Microsatellites give no information about which alleles are maternal and which are paternal. Since SibJoin constructs families in an iterative manner, part of a maternal family could be reconstructed on the maternal side, while the other part is constructed on the paternal side. If we are too strict about which sets we call maternal and paternal, then the two halves will never be joined and the half on the paternal side will likely force

**Algorithm 2** SibJoin join selection

```
1: function SELECT JOINS(simHash, λ, Λ)
2:     for t = numLoci * 2 → 0 do
3:         C ← Sort(simHash(t))                ▷ Largest average half-sib cluster size first
4:         for (i_x, i_y) ∈ C do
5:             if Λ(i_x), Λ(i_y) are compatible full-sibs then
6:                 C ← C/(i_x, i_y)
7:                 Join(Λ(i_x), Λ(i_y))
8:                                             ▷ Each full-sib join requires two half-sib joins
9:                 Join(λ(i_x), λ(i_y))        ▷ Join largest compatible HS families
10:                Join(λ(i_x), λ(i_y))        ▷ And the remaining cluster for each family
11:            end if
12:        end for
13:        for (i_x, i_y) ∈ C do
14:            if λ(i_x), λ(i_y) are compatible half-sibs then
15:                Join(λ(i_x), λ(i_y))        ▷ Join largest compatible HS families
16:            end if
17:        end for
18:    end for
19: end function
```

incorrect future joins. The solution is to implement an instance of the bipartite graph $G = (V, E)$ discussed in Section 4.3, where each cluster is a vertex and edges exist between clusters which share an individual. Let a join between clusters $C_i$ and $C_j$ be an event which combines $C_j$ into $C_i$ and let $E(v)$ be the set of edges that touch $v$. In our graph, $join(C_i, C_j)$ results in $E(v_i) := E(v_i) \bigcup E(v_j)$ followed by the removal of $v_j$ and all edges in $E(v_j)$. Enforcing bipartiteness as a postcondition of the join operation allows flexibility while ensuring that the solution results in each individual having one parent of each sex.

### 5.2.2 Allowing Candidate Parents

Identifying candidate parents can drastically increase the correctness of sibship reconstructions. SibJoin allows for the inclusion of candidate parents for either or both sexes. If candidate parents are given, a first round of clustering will attempt to join individuals using parent sets which contain only candidate parents. Once no more joins can be made with the restricted parent set, SibJoin will then continue to join clusters as described in the general case. The second round of joins ensures that unobserved parents will not prevent the algorithm from correctly reconstructing half-sibling families.

## 5.3 Experimental Results

### 5.3.1 Simulated Data Set Results

Simulation sets were constructed to test various parameters. Our model generates individuals from an equal number of mothers and fathers. For each mating, parents are chosen randomly, and children are generated from mother-father pairs according to an even allele

distribution. Simulated data had default parameter values of 6 alleles per locus, 6 loci, half-sibling family sizes of 5 individuals, and a population size of 40 individuals. The results are an average of ten trials per parameter value. Trials which failed to complete in 1 day are reported as '-'. The population size was increased to 80 individuals for family size trials so that the partitionings did not become trivial. The locus count was increased to 10 and family size to 20 when testing population sizes above 200 individuals. A summary of our parameter tests and their results may be found in Table 5.2. Testing occurred on a 2.66 GHz machine, containing 8 GB of RAM, and running Python 2.7.

In most cases, the reported VI score approximates the ratio of the partition distance to population size. Overall, COLONY 2 was more accurate, but took thousands of times longer, often with only small gains in accuracy. SibJoin does much worse than COLONY 2 on the 10 allele per site test set, but the discrepancy is due to a single trial for which SibJoin produces a solution with a VI of 0.084 while COLONY 2 produces a perfect reconstruction. For the 10 locus test set, SibJoin's VI is again higher, but in practice the false positive difference between it and COLONY 2 is about one individual per trial.

SibJoin does worst when the population size is large and the family size is small. For instance, when tested with a 100-individual population and families of 5 individuals, SibJoin rendered a VI of 0.201 compared to COLONY 2's VI of 0.086. When family sizes are small and population sizes are large, it is much more likely for two unrelated individuals to be mistakenly labeled as half-siblings. However, SibJoin's accuracy rapidly improves with modest increases in family size. In fact, SibJoin is more accurate than COLONY 2 in trials with families containing 20 individuals. Unsurprisingly, both methods poorly reconstruct populations where only two alleles are present. With only two alleles, all individuals can be full or half-siblings.

We may also use SibJoin to explore populations with extreme numbers of individuals.

| Variable parameter | Parameter settings | SibJoin | | COLONY 2 | |
|---|---|---|---|---|---|
| | | Runtime | VI (normalized) | Runtime | VI (normalized) |
| $k$: number of alleles | 2 | 2.8 ms | 0.396 | 48.9 min | 0.553 |
| | 5 | 13.2 ms | 0.222 | 19.7 min | 0.110 |
| | 10 | 6.7 ms | 0.014 | 12.8 min | 0.004 |
| | 15 | 5.1 ms | 0.014 | 10.2 min | 0.006 |
| | 20 | 5.7 ms | 0.003 | 10.0 min | 0.000 |
| $m$: number of loci | 2 | 8.7 ms | 0.469 | 10.7 min | 0.524 |
| | 5 | 10.1 ms | 0.156 | 17.2 min | 0.130 |
| | 10 | 11.1 ms | 0.035 | 14.2 min | 0.001 |
| | 15 | 12.7 ms | 0.002 | 20.4 min | 0.000 |
| | 20 | 12.1 ms | 0.000 | 21.3 min | 0.000 |
| $n$: population size | 10 | 0.4 ms | 0.042 | 2.29 min | 0.343 |
| | 50 | 16.8 ms | 0.104 | 17.1 min | 0.078 |
| | 100 | 82.5 ms | 0.201 | 73.5 min | 0.086 |
| | 200 | 3.31 sec | 0.230 | - | - |
| | 500 | 34.68 sec | 0.013 | - | - |
| | 1000 | 2.84 min | 0.015 | - | - |
| | 2000 | 12.43 min | 0.018 | - | - |
| $f$: family size | 1 | 51.9 ms | 0.546 | - | - |
| | 5 | 51.1 ms | 0.183 | 29.6 min | 0.051 |
| | 10 | 46.2 ms | 0.040 | 19.6 min | 0.017 |
| | 20 | 58.4 ms | 0.009 | 21.7 min | 0.042 |

Table 5.2: Simulated test results for SibJoin and COLONY 2 averaged over 10 trials. Trials which did not complete in 24 hours are marked '-'.

SibJoin was able to reconstruct sibgroup assignments for populations of 500, 1000, and 2000 individuals in under 10 minutes, yet problems of this magnitude are intractable for the HS-MSC and both of the COLONY programs. Furthermore, despite being thousands to tens of thousands of times faster than COLONY 2, SibJoin still rivals the maximum likelihood algorithm in overall accuracy.

| Data Set | Algorithm | Runtime | VI (normalized) | False Positives |
|---|---|---|---|---|
| 112 crickets | COLONY 2 | 35.7 min | 0.000 | 0 |
| | HS-MSC | - | n/a (see caption) | 2 |
| | SibJoin | 19.3 ms | 0.014 | 1 |
| 288 kelp rockfish | COLONY 2 | 624.5 min | 0.000 | 0 |
| | HS-MSC | - | n/a (see caption) | 0 |
| | SibJoin | 87.5 ms | 0.000 | 0 |
| 672 kelp rockfish | COLONY 2 | - | - | - |
| | HS-MSC | - | - | - |
| | SibJoin | 5.02 sec | 0.108 | 78 |

Table 5.3: Tests for biological data. A '-' indicates that an algorithm did not complete after 24 hours. SibJoin was the only algorithm able to construct a solution for a 672 individual population of rockfish. The variation of information is not computed for the HS-MSC since it allows instances of the same individual, which causes ill-defined VI scores.

### 5.3.2 Biological Data Set Results

SibJoin was tested on two biological data sets. The first data set is a population of 112 field crickets with 7 mothers and 6 sampled loci [7]. The second data set is a population of 672 kelp rockfish with 7 mothers and 7 sampled loci [31]. Neither COLONY 2 nor the HS-MSC produced a solution for the 672 rockfish population, so samples from three of the parents were taken to reduce the population size to 288 individuals. In both populations, only maternal parentage was available. For all trials, SibJoin was run in a configuration that only attempts to reconstruct the maternal sex.

Our results are compared to the HS-MSC results in [29] and to our own benchmarks on COLONY 2. Because the HS-MSC is not yet publicly available, we could not assess runtime information for the program. However, the authors do note that the HS-MSC IP finished in under one day. The difference between the two runtimes is not explained merely by CPU speed increases across a small number of years. Additionally, neither COLONY 2 nor the HS-MSC's half-sibling minimum set cover approach constructed a feasible answer

for the 672 rockfish data set: COLONY 2 was stopped after running for three days. SibJoin constructs an accurate solution in under 10 seconds.

The HS-MSC ILP does not enforce that individuals must have one parent of each sex and both partition distance and variation of information are ill-defined when the result is not a true partitioning. In the population of 112 crickets, the HS-MSC had two false positives and was otherwise correct. In the test set containing 288 rockfish, HS-MSC had 4 false positives and was otherwise correct. COLONY 2 was correct in all instances. SibJoin correctly reconstructed the half-sibship for the 288 rockfish and only misplaced one individual in the cricket test. SibJoin was the only algorithm to complete for the population of 672 rockfish. Overall, SibJoin is as accurate as the HS-MSC and nearly as accurate as COLONY 2, but is much faster than either: SibJoin solves the small rockfish instance over 42,000 times faster than COLONY 2.

### 5.3.3 Integer Programming Performance

The integer program proposed in Section 5.1 keeps track of all alleles in each family at each locus and for every individual. As a result, the IP can have as many as $O(kmn^2)$ constraints, which causes the IP to fail for all but the smallest instances. Table 5.4 compares the accuracy of the IP to the SibJoin algorithm for varying population sizes. Each population size, from 10 to 30 individuals, was tested over ten trials and the averages are reported in Table 5.4. The IP failed to produce partitionings which minimized the number of clusters for 2 trials when the population size was 20 individuals, it failed 6 times when the population contained 25 individuals, and it failed to reconstruct families for any population with 30 individuals. Additionally, the IP only marginally out performed SibJoin in accuracy for the 25 individual population test. In each other instance, SibJoin was

| n | $VI_{SJ}$ | $VI_{IP}$ | IP Runtime (seconds) | IP Failed Reconstructions |
|---|---|---|---|---|
| 10 | 0.000 | 0.039 | 0.546 | 0 |
| 15 | 0.046 | 0.119 | 106.065 | 0 |
| 20 | 0.078 | 0.080 | 2905.782 | 2 |
| 25 | 0.089 | 0.087 | 12324.613 | 6 |

Table 5.4: Performance of half-sibling IP, measured by $VI_{IP}$ versus SibJoin, $VI_{SJ}$ for varying population sizes. Ten trials were conducted for each population size and the averaged score is reported.

more accurate. Unsurprisingly, the running time of the IP appears to grow exponentially with population size. For 25 individuals, the IP took an average of 3 hours and 25 minutes to solve in the instances where an optimal solution was found. SibJoin took less than a second for each trial. These results indicate that SibJoin is a better alternative to the IP in almost all instances. Since the IP we gave is an expanded version of the HS-MSC algorithm, the results indicate that SibJoin should also produce more accurate reconstructions than the HS-MSC formulation.

Even when the IP manages to finish, there is no guarantee that the minimum set covers are correct. In Chapter 3, we argued that part of what made the half-sibling reconstruction problem so difficult, was identifying incorrectly placed individuals. Although the IP presented in this chapter achieves reasonable results for small families, both it and the HS-MSC formulation suffer from multiple optimal solutions where parentage for a few individuals is incorrectly assigned with no impact to the objective. At the same time, we have shown that a simple and fast heuristic performs more accurately and thousands of times faster than either of the IPs and rivals the accuracy of full likelihood methods such as COLONY 2. We demonstrated that the heuristic's speed allows us to reconstruct family relationships for populations that are too large for existing methods. Additionally, since the heuristic joins the most similar individuals and families first, it is more accurate

than existing methods when the allele pool or locus count is very small. Lastly, SibJoin is deterministic and does not suffer from multiple optimal solutions like both of the set cover IP's. These traits make SibJoin an important alternative to existing algorithms when sample populations are large or computing power is limited.

# Chapter 6

# Forced Allele Incompatibilities

## 6.1 Complexity of the Valid Half-Sibling Partitioning decision problem

In half-sibling problems, a complication arises from the requirement that each half-sibling must be contained in a maternal and paternal partitioning. To respect Mendelian genetics, half-sibling partitionings must be created so that each child receives exactly one allele from each parent at each locus. However, choosing the alleles that were inherited from each parent for each individual in a polygamous population is a non-trivial task and gives rise to a new decision problem. A forced allele for one individual influences the choice of opposite sex parent. In the worst case, choosing a parent can influence the choice of parent for every other maternal and paternal family in the population due to the fact that each parent could have mated with multiple other individuals.

The sub-population in Table 6.1 demonstrates how forced allele choices for mothers $M_0$ and $M_1$ can make their common mate $P_0$ incompatible with his candidate offspring. In

this example, the three half-sibships can be explained by $\{M_0 = (0, 2), P_0 = (0, *), M_1 = (0, 1)\}$. However, notice that $M_0$ forces offspring $(0, 1)$ to inherit allele 0 and $M_1$ forces offspring $(0, 3)$ to inherit allele 0. As a result, $(0, 1)$ must inherit allele 1 from $P_0$ and $(0, 3)$ must inherit allele 3 from $P_0$. At the same time, $P_0$ must also pass allele 0 to offspring $(0, 0)$. Therefore, there is an incompatibility. Even though each half-sibling family is valid when viewed independently, the choice of alleles for two different mates leads to an incompatibility for $P_0$. Although this is a small example, highly polygamous populations can have even further-reaching effects. Suppose that a candidate solution has mother $M_0$ mate with $P_0$ who also mates with $M_1$. Now suppose that $M_1$ also mates with $P_1$ who additionally mates with $M_2$. $M_0$ directly influences which alleles must be inherited from $P_0$ in our model. $P_0$ also influences which alleles our model can choose for $M_1$. Therefore, the mating relationship is transitive: in our example, $M_0$ now influences which alleles the model chooses for parent $M_1$. These forced allele choices are propogated so that eventually $M_0$ influences the model's allele choices for $M_2$. In highly polygamous populations, it is likely that the allele choice for each parent will be influenced by many other parents in the proposed population structure, which can make it difficult to decide allele assignments for each parent. In fact, we will shortly show that deciding whether a valid allele assignment exists for each parent in a proposed population structure, and thus if a population obeys the laws of Mendelian inheritance, is an *NP*-complete problem.

Given maternal and paternal half sibling partitionings, with each individual belonging to exactly one maternal and one paternal partition, is it possible to assign genotypes to the parents of each half-sibling family in a way that respects the property that every individual must inherit one of exactly two alleles from each parent? We will call this problem HALF-SIB PARENT COVER.

**Theorem 4.** *HALF-SIB PARENT COVER is NP-complete.*

| $M_0 : (0, 2)$ | $P_0 : (?, ?)$ | $M_1 : (0, 1)$ |
|---|---|---|
| (**0**,1) | (0,**1**) | (**0**,3) |
| (**0**,0) | (0,**0**) | (**0**,4) |
| (**2**,2) | (0,**3**) | (**1**,1) |

Figure 6.1: A sub-population which illustrates how the candidate offspring of $M_0$ and $M_1$ can force an incompatible half-sibling reconstruction through their matings with $P_0$. Forced alleles are bolded. Notice that the forced alleles for offspring of $M_0$ and offspring of $M_1$ force a situation where $P_0$ must have 3 alleles in order to satisfy Mendelian inheritance for his offspring.

| literals | family | possible shared parent |
|---|---|---|
| $x_i$ | $(y_j, y_k)_0$ | $(y_j, y_k)$ |
| | $(y_j, y_k)_1$ | |
| $x_j$ | $(y_i, y_k)_0$ | $(y_i, y_k)$ |
| | $(y_i, y_k)_1$ | |
| $x_k$ | $(y_i, y_j)_0$ | $(y_i, y_j)$ |
| | $(y_i, y_j)_1$ | |

Figure 6.2: Demonstration of the selection gadget transformation for the $p^{th}$ clause $(x_i, x_j, x_k)$. Subscripts are used to differentiate individuals with the same alleles

*Proof.* We first show that HALF-SIB PARENT COVER $\in$ *NP*. Given an instance of the problem and a certificate which assigns a genotype to the parent of each half sibling family, we can verify in polynomial time that the solution is valid by determining which allele each parent contributes for every individual and checking that there are no instances where the same allele of a heterozygotic individual is assigned by both the mother and the father. If a parent does not force an allele, *e.g.* the parent is $(a, b)$ and the child is also $(a, b)$, then the decision of which allele to cover is deferred to the parent of opposite sex. If both assignments are ambiguous, then the choice of which allele is maternal and which is paternal is arbitrary. If a child contains a locus which is homozygotic, then each parent must force the same allele.

Next, we give a polynomial-time reduction from the *NP*-complete MONOTONE ONE-

| $y_i$ fam. | $y_i$ fam. | $y_j$ fam. | $y_j$ fam. | $y_k$ fam. | $y_k$ fam. |
|---|---|---|---|---|---|
| $(y_i, y_j)_0$ | $(y_i, y_k)_0$ | $(y_i, y_j)_1$ | $(y_j, y_k)_0$ | $(y_i, y_k)_1$ | $(y_j, y_k)_1$ |
| $(s_p, y_i)_0$ | $(s_p, y_i)_1$ | $(s_p, y_j)_0$ | $(s_p, y_j)_1$ | $(s_p, y_k)_0$ | $(s_p, y_k)_1$ |
| $(s_p, z)_0$ | $(s_p, z)_1$ | $(s_p, z)_2$ | $(s_p, z)_3$ | $(s_p, z)_4$ | $(s_p, z)_5$ |
| $(y_j, z)_0$ | $(y_k, z)_0$ | $(y_i, z)_0$ | $(y_k, z)_1$ | $(y_i, z)_1$ | $(y_j, z)_1$ |

Figure 6.3: Demonstration of the mapping gadget which maps a parent choice in each of the first gadget's maternal families to an allele. Allele $y_i$ is forced if and only if $x_i$ is true in the MONOTONE ONE-IN-THREE-SAT solution.

| $c_p/c_q$ | $c_p/c_r$ | $c_q/c_r$ |
|---|---|---|
| $(s_p, y_i)$ | $(s_p, y_i)$ | $(s_q, y_i)$ |
| $(s_p, y_i)$ | $(s_p, y_i)$ | $(s_q, y_i)$ |
| $(s_q, y_i)$ | $(s_r, y_i)$ | $(s_r, y_i)$ |
| $(s_q, y_i)$ | $(s_r, y_i)$ | $(s_r, y_i)$ |
| $(s_p, z)$ | $(s_p, z)$ | $(s_q, z)$ |
| $(s_q, z)$ | $(s_q, z)$ | $(s_r, z)$ |

Figure 6.4: Construction of the enforcement gadget for literal $x_i$ appearing in clauses $c_p$, $c_q$, and $c_r$

IN-THREE SAT problem to HALF-SIB PARENT COVER. The ONE-IN-THREE SAT problem is, given a set of boolean clauses, each containing three literals, determine whether a configuration of literals exists such that exactly one literal in each clause is set true. The MONOTONE ONE-IN-THREE SAT problem is the ONE-IN-THREE SAT problem with the constraint that no literals may be negated. This is also called EXACT-COVER-BY-3-SETS (X3C), which was used in the first proof of the NP-hardness of parsimony phylogeny [13]. The reduction requires three gadgets that translate literals and clauses in MONOTONE ONE-IN-THREE-SAT into alleles and families in HALF-SIB PARENT COVER respectively.

The first gadget translates picking a literal in a clause to picking a parent for a family. The second gadget defines paternal families that map the choice of parent to alleles which

correspond to literal choices. From the MONOTONE ONE-IN-THREE SAT perspective, the third gadget enforces the rule that if a literal is chosen to be set true in one clause, it must be chosen to be true in all of the clauses it belongs to. Define a one-to-one function $f : x \to y$ which assigns each SAT literal to a unique integer allele value.

1. The *selection gadget* provides a mechanism which is analagous to choosing the true literal in each clause of the SAT instance. First, we create a maternal family from the mapping of literals to alleles for each clause individually. Each family is constructed so that there are exactly three valid parents for a family of 6 individuals. For a clause with literals $(x_i \vee x_j \vee x_k)$, the corresponding $y_i$, $y_j$, and $y_k$ will be the alleles present in the created family. Three children are created by taking each pairwise grouping of the $y$ values. A copy of each child is made so that there are a total of 6 children per selection gadget family. There are three possible mothers for each selection gadget family. Figure 6.2 demonstrates this portion of the gadget for clause $(x_i, x_j, x_k)$. Each mother possesses two of the three alleles in the family. Choosing mother $(y_j, y_k)$ corresponds to setting literals $x_j$ and $x_k$ to false and setting literal $x_i$ to true in the MONOTONE ONE-IN-THREE SAT formulation. By definition, only one parent may be chosen for each of these families which satisfies the one-in-three SAT requirement of the MONOTONE ONE-IN-THREE SAT problem.

2. The *mapping gadget* creates two paternal families for each potential mother, producing a total of six paternal families per maternal family created by the selection gadget. The paternal families map the choice of mother onto a single allele with the property that allele $y_i$ is forced in the paternal families if and only if parent $(y_j, y_k)$ was chosen as the mother. A father with genotype $(y_i, z)$ sets $x_i$ literal to true in the MONOTONE ONE-IN-THREE SAT instance.

Let the number of clauses equal $m$. In order to construct these families, we introduce alleles $s_0 \dots s_{m-1}$, one for each clause, and another distinct allele $z$. The $z$ allele is only used to ensure that the correct relationships are enforced. The $s$ alleles are used in the third gadget to enforce consistent state assignments across all clauses for each literal. Figure 6.3 illustrates how to construct the paternal families. Either the $s_p$ allele or the $y$ allele is inherited from the father in each paternal family, but it is impossible for the father to have both $s_p$ and the $y$ alleles as alleles. Multiple copies of the $(s_p, y_i)$ child may be needed for the enforcement gadget. Let $k_i$ be the number of clauses that contain $x_i$. Each paternal family corresponding to $y_i$ must have $k_i - 1$ such children.

3. Lastly, we construct a gadget that forces the property that if an allele is picked in one selection gadget family, it must be picked in every selection gadget family that contains the allele. Analogously, a true literal must be true in every clause and a false literal must be false in every clause. The *enforcement gadget* forces this requirement by constructing a constraining family for each pair of clauses in which a literal occurs. If a literal $x_i$ appears in clauses $c_p$ and $c_q$, then a family will be constructed so that either $y_i$ is forced or $s_p$ and $s_q$ are forced. The gadget makes use of the $(s_p, y_i)$ and $(s_q, y_i)$ individuals created by the mapping gadget. Figure 6.4 demonstrates how these families are constructed.

   Each enforcement gadget family for $y_i$ has two copies of $(s_p, y_i)$ which are the two children from the mapping gadgets containing $(y_i, y_j)$ and $(y_i, y_k)$. The redundant $(s, y)$ children prevent one mapping gadget from lying about its assigned allele. If a child has the same genotype as its parent, then which allele was received from that parent is ambiguous. For example, consider allele $y_i$ with parent $(y_i, y_j)$ chosen. All paternal mapping gadgets which contain child $(y_i, y_j)$ will possess an ambiguity as

to whether $y_i$ or $y_j$ comes from the mother. As a result, we could pick $y_i$ to be true in the corresponding mapping gadget family, even though it ought to be false. However, even if $(y_i, y_j)$ is ambiguous, $(y_i, y_k)$ is not and, due to the redundance of $(s, y_i)$ children in the enforcement gadget, the $(y_i, y_j)$ mapping gadget would not be able to lie without creating an infeasible instance of the problem. Having one $(s, y)$ child from each mapping gadget family solves this ambiguity, because it is possible for one mapping gadget to falsely report the forced allele, but never both. If a literal is in a single clause, then it will not have an enforcement gadget family and the selection gadget family corresponding to the clause will contain an individual where it is impossible to determine which allele was inherited from the mother and the father. However, the ambiguity does not affect the feasibility of the solution: either allele may be chosen in the selection gadget without consequence since the allele choice is not propagated to other families and does not influence the choice of alleles for any other individuals in the family.

If $s_p$ is forced, then $s_q$ must also be forced to avoid an incompatibility. As a result, $y_i$ is forced in both paternal mapping gadget families. However, if $y_i$ is forced, then $s_p$ and $s_q$ are forced to be true in the paternal mapping gadget families. Therefore, if $y_i$ is selected in one family, it must be selected in all families that contain it. Conversely, If $y_i$ is not selected in a family, then it is not selected in any family that contains it.

In the MONOTONE ONE-IN-THREE SAT problem, a literal from each clause must be set to true. The selection gadget translates the task of choosing an allele to picking the parents of maternal families. Each selection gadget family contains three distinct alleles $\{y_i, y_j, y_k)$. Choosing maternal parent $(y_i, y_j)$ is equivalent to setting $x_k$ true and the $x_i$ and $x_j$ literals to false. Since each literal may appear more than once in a MONOTONE ONE-IN-THREE SAT instance, the equivalent relationship is that any selected maternal

genotype in the selection gadget must be selected in each maternal family for which the genotype is a candidate parent. The enforecement gadget ensures that the proper maternal parent selections occur. However, the enforcement gadget cannot directly enforce the requirement on the maternal families due to restrictions from Mendelian inheritance. Therefore, the mapping gadget uses mating transitivity to act as a bridge between the selection gadget and the enforcement gadget by introducing new alleles and individuals that allow the enforcement gadget to influence which parents are selected in the selection gadget. Finally, let $n$ be the number of literals and $m$ be the number of clauses. Constructing the HALF-SIB PARENT COVER instance requires $O(m)$ children for the first gadget, $O(m^2 \cdot n)$ additional children for the second gadget, and $O(1)$ additional children for the third gadget, so the resulting transformation is polynomial in size. □

The reduction builds an instance of HSPC with one locus for each individual. The decision instance of this problem is not substantially more difficult for children with multiple loci due to the independence assumptions about individual loci. Determining if a population with multiple loci fulfills the rule of one allele from each parent at each locus requires solving an instance of HALF-SIB PARENT COVER at each locus independently. If there are $l$ loci, then this adds a factor $O(l)$ to the problem.

Figure 6.5 demonstrates a reduction from a MONOTONE ONE-IN-THREE SAT instance with two clauses to the HALF-SIB PARENT COVER problem. There are several feasible solutions to the M-1-3-SAT instance, but the example illustrates the case where literals $x_2$ and $x_4$ are set true in the M-1-3-SAT instance. The inherited allele for each individual in each family is bolded to represent the corresponding HSPC solution where mothers $(1, 3)$ and $(1, 5)$ are chosen in the selection gadget.

$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_4 \vee x_5)$

(a) An M-1-3-SAT instance with two clauses $C_0$ and $C_1$

| $M_0$ | $M_1$ |
|---|---|
| $(\mathbf{1}, 2)_0$ | $(\mathbf{1}, 4)_0$ |
| $(\mathbf{1}, 2)_1$ | $(\mathbf{1}, 4)_1$ |
| $(\mathbf{1}, 3)_0$ | $(\mathbf{1}, 5)_0$ |
| $(1, 3)_1$ | $(1, 5)_1$ |
| $(2, \mathbf{3})_0$ | $(4, \mathbf{5})_0$ |
| $(2, \mathbf{3})_1$ | $(4, \mathbf{5})_1$ |

(b) Maternal selection gadget families

| $M_3$ |
|---|
| $(s_0, \mathbf{1})_0$ |
| $(s_0, \mathbf{1})_1$ |
| $(s_1, \mathbf{1})_0$ |
| $(s_1, \mathbf{1})_1$ |
| $(s_0, \mathbf{z})_6$ |
| $(s_1, \mathbf{z})_6$ |

(c) Maternal enforcement gadget family for $x_1$

| $P_0$ | $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ |
|---|---|---|---|---|---|
| $(1, \mathbf{2})_0$ | $(1, \mathbf{3})_0$ | $(\mathbf{2}, 3)_0$ | $(1, \mathbf{2})_1$ | $(1, 3)_1$ | $(\mathbf{2}, 3)_1$ |
| $(\mathbf{s_0}, 1)_0$ | $(\mathbf{s_0}, 1)_1$ | $(s_0, \mathbf{2})_0$ | $(s_0, \mathbf{2})_1$ | $(s_0, 3)_0$ | $(\mathbf{s_0}, 3)_1$ |
| $(\mathbf{s_0}, z)_0$ | $(\mathbf{s_0}, z)_1$ | $(s_0, \mathbf{z})_2$ | $(s_0, \mathbf{z})_3$ | $(s_0, z)_4$ | $(\mathbf{s_0}, z)_5$ |
| $(\mathbf{2}, z)_0$ | $(\mathbf{3}, z)_0$ | $(3, \mathbf{z})_1$ | $(1, \mathbf{z})_0$ | $(1, z)_1$ | $(\mathbf{2}, z)_1$ |

| $P_6$ | $P_7$ | $P_8$ | $P_9$ | $P_{10}$ | $P_{11}$ |
|---|---|---|---|---|---|
| $(1, \mathbf{4})_0$ | $(1, \mathbf{5})_0$ | $(\mathbf{4}, 5)_0$ | $(1, \mathbf{4})_1$ | $(1, 5)_1$ | $(\mathbf{4}, 5)_1$ |
| $(\mathbf{s_1}, 1)_0$ | $(\mathbf{s_1}, 1)_1$ | $(s_1, \mathbf{4})_0$ | $(s_1, \mathbf{4})_1$ | $(s_1, 5)_1$ | $(\mathbf{s_1}, 5)_1$ |
| $(\mathbf{s_1}, z)_0$ | $(\mathbf{s_1}, z)_1$ | $(s_1, \mathbf{z})_2$ | $(s_1, \mathbf{z})_3$ | $(s_1, z)_4$ | $(\mathbf{s_1}, z)_5$ |
| $(\mathbf{4}, z)_0$ | $(\mathbf{5}, z)_0$ | $(5, \mathbf{z})_1$ | $(1, \mathbf{z})_2$ | $(1, z)_3$ | $(\mathbf{4}, z)_1$ |

(d) Paternal mapping gadget families

| $P_{12}$ | $M_4$ | $M_5$ | $M_6$ | $M_7$ | $M_8$ | $M_9$ | $M_{10}$ |
|---|---|---|---|---|---|---|---|
| $(\mathbf{s_0}, z)_6$ | $(s_0, \mathbf{z})_0$ | $(s_1, \mathbf{z})_0$ | $(\mathbf{1}, z)_0$ | $(2, \mathbf{z})_0$ | $(3, \mathbf{z})_0$ | $(4, \mathbf{z})_0$ | $(5, \mathbf{z})_0$ |
| $(\mathbf{s_1}, z)_6$ | $(s_0, \mathbf{z})_1$ | $(s_1, \mathbf{z})_1$ | $(1, z)_1$ | $(2, \mathbf{z})_1$ | $(\mathbf{3}, z)_1$ | $(\mathbf{4}, z)_1$ | $(\mathbf{5}, z)_1$ |
| | $(\mathbf{s_0}, z)_2$ | $(\mathbf{s_1}, z)_2$ | $(\mathbf{1}, z)_2$ | | | | |
| | $(\mathbf{s_0}, z)_3$ | $(\mathbf{s_1}, z)_3$ | $(1, z)_3$ | | | | |
| | $(s_0, z)_4$ | $(s_1, z)_4$ | | | | | |
| | $(\mathbf{s_0}, z)_5$ | $(s_1, \mathbf{z})_5$ | | | | | |

(e) Extra families for completeness

Figure 6.5: Changing a two clause M-1-3-SAT instance to an HSPC instance

## 6.2 Correcting Allele Incompatibilities

Unfortunately, the *NP*-completeness of the decision version of the HSPC problem makes it very unlikely that a polynomial time algorithm exists for identifying incompatibilities. As a result, it is impractical to verify that joining two clusters results in partitionings with valid parent assignments. However, it is still desirable to correct these errors in order to obtain a more accurate sibship reconstruction. In this section, we will discuss a simplification of the HALF-SIB PARENT COVER which increases the average accuracy of SibJoin and may be solved in polynomial time. We also present a 0-1 integer program for solving an optimization version of the problem that asks for the minimum number of individuals which need to be removed from the population in order to eliminate forced allele incompatibilities.

### 6.2.1 Shallow Incompatibility Detection

One way of reducing the complexity HALF-SIB PARENT COVER problem is to limit the scope of the search for incompatibility. Instead of investigating transitive relationships of parents, which is required in HSPC instances, and which likely requires super-polynomial time, we may instead ask for individuals which have an incompatible set of candidate parents. Specifically, if all the maternal and paternal candidate parents force the same allele at a heterozygotic locus of an individual, then that individual is incompatible with one or both of its assigned families; since both alleles need to arise somehow. Because we are not concerned with the restrictions from forced alleles of other individuals, this subproblem may be completed in $O(l \cdot m^2 \cdot n)$ time.

This type of shallow detection was incorporated into our fast heuristic, SibJoin. It decreased the average error for each test. In our experiments, we found that encorporating

this shallow detection into the initial joining phase tends to magnify the impact of misplaced individuals by preventing otherwise correct joins. Therefore, it is preferable to let SibJoin run to completion and fix forced allele errors by removing incompatible individuals from the population and re-running the algorithm starting from the remaining partitionings.

### 6.2.2 Complete Forced Allele Incompatibility Detection

Although a shallow search for forced allele incompatibilities improves results, it does not completely resolve all forced allele incompatibilities: again, the problem is *NP*-Hard. This is due to the chaining nature of individuals in half-sibling families. Under the assumption that maternal and paternal partitionings are mostly correct, which is the case for SibJoin, a natural question is to ask the minimum number of individuals which must be removed in order to resolve the forced allele incompatibility, and leave an instance where valid parents may be assigned to every cluster.

Figure 6.6 gives a 0/1 integer program which minimizes the set of individuals which must be removed in order to resolve forced allele incompatibility in the population. For a population with $n$ individuals, each possessing $m$ measured loci, let $x_i = 1$ denote the decision to remove individual $i$ from the population. The variable $y^l_{j,k}$ represents the $k^{th}$ allele at the $l^{th}$ locus of family $j$. Denote the multi-set which contains the maternal and paternal families as $\mathcal{C}$. $\pi_0$ and $\pi_1$ are functions which map an individual to its maternal and paternal index in $\mathcal{C}$ respectively. $\lambda_0$ and $\lambda_1$ map the first and second allele of an individual to an index in $K$, the set of all alleles.

The first constraint enforces that no parent can have more than two alleles. The second and third constraints enforce Mendelian mating requirements on individuals: an individual is invalid if it does not receive one allele from its mother and its other allele from its father.

$$\textbf{minimize} \quad \sum_i x_i$$

$$\textbf{subject to}$$

$$2 - \sum_{k \in K} y^l_{j,k} \geq 0, \qquad 0 \leq j < |\mathcal{C}|, 0 \leq l < m$$

$$x^l_i + \frac{1}{2}(y^l_{\pi_0(i),\lambda_0(i)} + y^l_{\pi_1(i),\lambda_1(i)}) \geq 1, \qquad 0 \leq i < n, 0 \leq l < m$$

$$x^l_i + \frac{1}{2}(y^l_{\pi_0(i),\lambda_1(i)} + y^l_{\pi_1(i),\lambda_0(i)}) \geq 1, \qquad 0 \leq i < n, 0 \leq l < m$$

$$x_i - x^l_i \geq 0, \qquad 0 \leq i < n, 0 \leq l < m$$

Figure 6.6: 0/1 integer program to find minimum number of individuals which, when removed, creates a valid instance of the HALF-SIB PARENT COVER.

There are two possible ways to satisfy this constraint. Either the child received its first listed allele from its mother and the second allele from its father or vice versa. The two constraints are the logical or of these two possibilities. Finally, the last constraint enforces the requirement that $x_i$ must be 1, corresponding to individual $i$ being selected for removal, if the individual has any incompatible loci. The minimization objective forces $x_i$ and $x^l_i$ to be 0 as often as possible. Since there can never be more families than individuals, $|\mathcal{C}| < n$, the integer program has a total of $O(m \cdot n)$ constraints.

## 6.3 Experimental Results

Using the IP outlined in Figure 6.6, it is possible to identify the minimum set of individuals which must be removed in order to make a solution from SibJoin feasible. The IP acts on the assumption that a relatively small set of incorrectly placed individuals causes bad joins, so finding the minimum individuals to remove should capture many incorrect individuals.

| Fixed parameter | Parameter | Norm. VI | FP | Recall | Precision | Timeout rate |
|---|---|---|---|---|---|---|
| $k$: number of alleles | 2 | 0.396 | 25.9 | 0.000 | 0.000 | 0.0 |
| | 5 | 0.225 | 12.3 | 0.300 | 0.694 | 0.0 |
| | 10 | 0.013 | 0.2 | 0.000 | 0.000 | 0.0 |
| | 15 | 0.014 | 0.0 | - | - | 0.0 |
| | 20 | 0.003 | 0.0 | - | - | 0.0 |
| $m$: number of loci | 2 | 0.491 | 23.7 | 0.109 | 0.563 | 0.0 |
| | 5 | 0.150 | 6.6 | 0.355 | 0.537 | 0.0 |
| | 10 | 0.032 | 1.2 | 0.62 | 0.650 | 0.0 |
| | 15 | 0.002 | 0.0 | - | - | 0.0 |
| | 20 | 0.000 | 0.0 | - | - | 0.0 |
| $n$: population size | 10 | 0.042 | 0.5 | 0.2 | 1 | 0.0 |
| | 50 | 0.098 | 10.2 | 0.340 | 0.679 | 0.0 |
| | 100 | 0.201 | 41.0 | 0.400 | 0.765 | 0.1 |
| | 200 | 0.220 | 88.9 | 0.408 | 0.780 | 1.0 |
| $f$: family size | 1 | 0.527 | 58.5 | 0.317 | 0.778 | 0.7 |
| | 5 | 0.181 | 22.8 | 0.439 | 0.756 | 0.0 |
| | 10 | 0.038 | 3.6 | 0.313 | 0.477 | 0.0 |
| | 20 | 0.009 | 1.4 | 0.000 | 0.000 | 0.0 |

Table 6.1: SibJoin trials with forbidden allele detection. A '-' occurs when there are no false positives

Although the IP generally solves quickly, it struggles to find a global optimum for populations of hundreds of individuals. In these cases the IP gets very close, often within three percent, to integrality, but never reaches an optimal integer solution since the IP runs out of memory. To hedge against this, we enforce a 5 minute time limit on the IP. We report the percent of trials that failed to reach integrality in Table 6.1. An approximate solution is acceptable as long as there is a reliable way to correctly re-add identified individuals into the population.

One possibility for reducing error is to detect and disallow shallow allele incompatibilities during clustering. Strict enforcement was tested with the SibJoin algorithm. Unfortunately, enforcing this requirement during the original joining phase raises the error

rate substantially: early incorrect joins have a much larger impact on future incompatibilities due to the new rule.

Instead, we report the effects of adding an extra step to the SibJoin algorith that, after the original algorithm completes, identifies individuals to remove with the IP given in Figure 6.6. These individuals are removed from existing families and placed in their own individual clusters. SibJoin is then allowed to run again with the added post-condition that a join will never create a shallow incompatibility.

Table 6.1 reports the recall and precision of the IP: the percentage of all incorrect individuals that are identified by the IP and the percentage of individuals that are actually incorrect among the individuals identified by the IP. We find that the integer program can have a poor recall, finding only 30% of the false positives in some situations; however, the precision is relatively high. For individuals in the minimum removal set, the number of incorrectly placed individuals is consistently above 50%. The precision is significant since SibJoin's total error rate is often far below 50%: randomly choosing a subpopulation where more than 50% of the individuals are misplaced is unlikely. Although the IP does not identify all false positive individuals, the individuals are often false positives. If there is a way to correctly reintroduce the set of individuals identified by the IP, then the error rate will decrease significantly.

Unfortunately, an algorithm which correctly re-adds the false positives has proved elusive. Several methods were tried and often lead to slightly worse scoring candidate reconstructions. The best method, which re-added individuals as their own clusters and forced shallow allow compatibility, only decreased the total error by four percent in some tests and marginally increased it in others.

The IP does worst when there are only two alleles or two loci. This is unsurprising since there will be no incompatibilities when each locus contains less than three alleles and

low loci counts mean that there is a smaller risk of forbidden forbidden allele structures with bad joins. However, both recall and precision tend to increase with population size as demonstrated by the 100 and 200 population size test cases. For populations with 200 individuals, the IP did not reach integrality within 5 minutes, but still produced high recall and precision relative to the other tests, which indicates that the IP is still useful for large populations and that approximation algorithms may do a decent job at identifying false positives.

# Chapter 7

# Conclusion

## 7.1 Reconstructing Half-Sibships

Many of the difficulties that arise during half-sibship reconstruction occur because each individual must be assigned to a cluster in each of two different partitionings. We have discussed methods for reconstructing these families so as to preserve Mendelian inheritence laws throughout reconstruction. We have also compared the triplet similarity measure used in some full sibling reconstruction algorithms to a simpler allele-based similarity measure and demonstrated that for half-siblings, the allele similarity measure more accurately identifies related individuals. Additionally, the pairwise similarity matrix of a population can be computed an order of magnitude faster with allele similarity than with triplet similarity. The speedup is significant since populations can have hundreds of individuals with identified loci in the double digits.

We have also demonstrated an application of allele similarity with our fast SibJoin heuristic. SibJoin is a bottom-up algorithm based on single linkage clustering. Our ex-

periments show that despite being a heuristic, the algorithm competes in accuracy with existing likelihood-based algorithms, but is thousands of times faster in practice. The speed of our algorithm is important since existing algorithms fail to reconstruct half-sibling families when the population size is above a few hundred individuals. SibJoin can reconstruct these populations in seconds. We have demonstrated that SibJoin is able to reconstruct real biological populations that existing algorithms fail to reconstruct, and it does so with high accuracy.

## 7.2 Determining Reconstruction Validity

Being able to assess the validity of half-sibling reconstruction is important for both determining how well an algorithm reconstructs known partitionings as well as for recognizing when an algorithm makes mistakes in the absence of the true population structure. To solve the first problem, we have employed information theoretic techniques for measuring the quality of an algorithm's reconstruction. Existing methods such as maximum matching, only offer a very basic understanding of how well an algorithm reconstructs a population. Instead, we have modified an information theoretic metric, called variation of information, so that it may be used with half-sibling partitionings.

It is also important to determine whether or not a proposed population structure is valid under Mendelian inheritance assumptions. For half-siblings, we have proved that even determining if such a structure obeys Mendelian laws is *NP*-complete. This realization has important implications for half-sibling algorithms in general since most existing algorithms do not specifically enforce which allele is inherited from the mother and which is inherited from the father. We have also provided an integer program that solves an optimization variant of the problem: what is the minimum number of individuals that must

be removed from a population in order for the population structure to be valid. The IP was run against SibJoin's population reconstructions. Although the IP only had a recall of 30 to 40 percent when run against SibJoin's population reconstructions, the precision was high: 55 to 78 percent of the individuals identified for removal were actually incorrect.

## 7.3 Future Work

Although we are often able to identify many of the incorrectly assigned individuals in a population, we have yet to find a good way of moving the individuals to their correct families. Unidentified misplaced individuals often prevent individuals from being compatible with their true families. Correctly replacing misplaced individuals depends on being able to identify other misplaced individuals. Techniques like boostrapping for phylogenies may provide a good way to both quantify confidence in a given model and increase reconstruction accuracy.

Another open question is whether or not there exists a polynomial time algorithm to compute the variation of information between two solutions when neither solutions' parental sexes are known. Having a good metric for measuring the difference between half-sibling partitionings is important and may also lead to algorithms that increase reconstruction accuracy.

Finally, current algorithms focus on reconstructing sibship relationships for a single generation. However, it would be useful to generalize the full and half-sibling techniques to reconstruct families for multiple generations of individuals.

# References


[1] A Almudevar and E Anderson. A New Version of PRT Software for Sibling Groups Reconstruction with Comments Regarding Several Issues in the Sibling Reconstruction Problem. *Mol. Ecol. Resour.*, 12(1):164–178, 2011.

[2] A Almudevar and C Field. Estimation of Single-Generation Sibling Relationships Based on DNA Markers. *J. Agric. Biol. Envir. S.*, 4(2):136–165, 1999.

[3] M Ashley, I Caballero, W Chaovalitwongse, B Dasgupta, P Govindan, S I Sheikh, and T Y Berger-Wolf. KINALYZER, a computer program for reconstructing sibling groups. *Mol. Ecol. Resour.*, 9(4):1127–1131, 2009.

[4] T Y Berger-Wolf and B DasGupta. Combinatorial reconstruction of sibling relationships. *Proc. of CGBI*, pages 3–6, 2005.

[5] T Y Berger-Wolf, S I Sheikh, B DasGupta, M V Ashley, I C Caballero, Wanpracha Chaovalitwongse, and S Lahari Putrevu. Reconstructing sibling relationships in wild populations. *Bioinform.*, 23(13):i49–56, July 2007.

[6] M S Blouin, M Parsons, V Lacaille, and S Lotz. Use of microsatellite loci to classify individuals by relatedness. *Mol. Ecol.*, 5(3):393–401, 1996.

[7] A Bretman and T Tregenza. Measuring polyandry in wild populations: a case study using promiscuous crickets. *Mol. Ecol.*, 14(7):2169–2179, 2005.

[8] D Brown and T Y Berger-Wolf. Discovering Kinship through Small Subsets. *Proc. 10th WABI-10*, 6293 in LNCS:111–123, 2010.

[9] D Brown and D Dexter. SibJoin: A Fast Heuristic for Half-Sibling Reconstruction. *Proc. 12th WABI-12*, 7534 in LNCS:44–56, 2012.

[10] G Caughley. Directions in Conservation Biology. *J. of Anim. Ecol.*, 63(2):215–244, 1994.

[11] W A Chaovalitwongse, C A Chou, T Y Berger-Wolf, B DasGupta, S Sheikh, M V Ashley, and I C Caballero. New Optimization Model and Algorithm for Sibling Reconstruction from Genetic Markers. *INFORMS J. Comp.*, 22(2):180–194, 2009.

[12] J A Coombs, B H Letcher, and K H Nislow. PedAgree: software to quantify error and assess accuracy and congruence for genetically reconstructed pedigree relationships. *Conserv. Genet. Resour.*, 2(1):147–150, 2010.

[13] L Foulds and Graham R. The Steiner Problem in Phylogeny is *NP*-Complete. *Adv. Appl. Math.*, 3:43–49, 1982.

[14] D Gusfield. Partition-distance : A problem and class of perfect graphs arising in clustering. *Info. Proc. Lett.*, 82(3):159–164, 2002.

[15] P Hedrick, R Lacy, F Allendorf, and M Soule. Directions in Conservation Biology: Comments on Caughley. *Conserv. Biol.*, 10(5):1312–1320, 1996.

[16] Y Isagi, T Kanazashi, W Suzuki, H Tanaka, and T Abe. Highly variable pollination patterns in *Magnolia obovata* revealed by microsatellite paternity analysis. *Int. J. Plant Sci.*, 165(6):1047–1053, 2004.

[17] M D Jennions and M Petrie. Why do females mate multiply? A review of the genetic benefits. *Biol. Rev. Camb. Philos. Soc.*, 75(1):21–64, 2000.

[18] A Jones, C Small, K Paczolt, and N Ratterman. A practical guide to methods of parentage analysis. *Molecular ecology resources*, 10(1):6–30, January 2010.

[19] B Jones, G D Grossman, D C I Walsh, B A Porter, J C Avise, and A C Fiumera. Estimating Differential Reproductive Success From Nests of Related Individuals, With Application to a Study of the Mottled Sculpin, *Cottus bairdi*. *Genet.*, 176(4):2427–2439, 2007.

[20] O R Jones and J Wang. Molecular marker-based pedigrees for animal conservation biologists. *Anim. Cons.*, 13(1):26–34, January 2010.

[21] Steven T Kalinowski, Aaron P Wagner, and Mark L Taper. Ml-Relate: a Computer Program for Maximum Likelihood Estimation of Relatedness and Relationship. *Molecular Ecology Notes*, 6(2):576–579, 2006.

[22] M Kimura and T Ohta. Population Stepwise mutation model and distribution of allelic frequencies in a finite population. *Proc. Natl. Acad. Sci. USA*, 75(6):2868–2872, 1978.

[23] D Konovalov, B Litow, and N Bajema. Partition-distance via the assignment problem. *Bioinform.*, 21(10):2463–2468, May 2005.

[24] Y Lin, V Rajan, and B Moret. A Metric for Phylogenetic Trees Based on Matching. *IEEE/ACM Trans. Comp. Bio and Bioinf.*, 9(4):1014–1022, December 2011.

[25] M Meila. Comparing clusterings by the variation of information. *Proceedings of COLT*, 2777:173–187, 2003.

[26] B D Neff and M R Gross. Microsatellite evolution in vertebrates: inference from AC dinucleotide repeats. *Evolution Int. J. Org. Evolution*, 55(9):1717–1733, 2001.

[27] I Painter. Sibship reconstruction without parental information. *J. Agric. Biol. Envir. S.*, 2(2):212–229, 1997.

[28] U Sezen, R Chazdon, and K Holsinger. Genetic consequences of tropical second-growth forest regeneration. *Science*, 307(5711):891, 2005.

[29] S I Sheikh, T Y Berger-Wolf, A A Khokhar, I C Caballero, M V Ashley, W Chaovalitwongse, C Chou, and B Dasgupta. Combinatorial reconstruction of half-sibling groups from microsatellite data. *J. of Bioinf. Comput. Biol.*, 8(2):337–356, 2010.

[30] B R Smith, C M Herbinger, and H R Merry. Accurate partition of individuals into full-sib families from genetic data without parental information. *Genet.*, 158(3):1329–1338, 2001.

[31] S M Sogard, E Gilbert-Horvath, E C Anderson, R Fisher, S A Berkeley, and J C Garza. Multiple paternity in viviparous kelp rockfish, *Sebastes atrovirens*. *Environ. Biol. Fishes*, 81(1):7–13, 2008.

[32] S Wagner and D Wagner. Comparing Clusterings - An Overview. Technical report, Faculty of Informatics, Universit t Karlsruhe, 2006.

[33] J Wang. Sibship reconstruction from genetic data with typing errors. *Genet.*, 166(4):1963–1979, 2004.

[34] J Wang and A W Santure. Parentage and sibship inference from multilocus genotype data under polygamy. *Genet.*, 181(4):1579–1594, 2009.