Parallel, Error-Tolerant Sibling Reconstruction for Wild Populations Using Microsatellite Markers

BY

ALAN PEREZ-RATHKE
B.S., University of Illinois at Champaign-Urbana, 2005

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2011

Chicago, Illinois

ACKNOWLEDGMENTS

I would like to thank Tanya Berger-Wolf for her invaluable insight, assistance, and for opening my eyes to new computing possibilities. I would also like to thank Ashfaq Khokar for his help in the crafting of this work.

AJPR

TABLE OF CONTENTS

LIST OF TABLES

TABLE **PAGE**

LIST OF FIGURES

LIST OF FIGURES (Continued)

FIGURE **PAGE**

SUMMARY

Wild populations of organisms are often difficult to study in their natural settings. Often, it is possible to infer mating information about these species by genotyping the offspring and using the genetic information to infer sibling and other kinship relationships. While sibling reconstruction has been studied for a long time, none of the existing approaches have targeted scalability. In this thesis, the first parallel, fault-tolerant approach to reconstructing sibling relationships from microsatellite markers is presented. The proposed solution employs domain decomposition along with an efficient culling technique to reduce the number of feasible sibling groups. In order to handle errors in the input data, multiple solutions are computed while minimizing redundant work across solutions. A consensus-based approach is then used to combine the solutions. The proposed solution has been implemented on an 8-core, shared memory platform. Our results show performance scaling proportional to the number of cores being used, particularly for large data sets.

1. INTRODUCTION

Sibling reconstruction is an important problem in understanding fundamental evolutionary and population processes. Advances in genotyping technology have allowed the possibility for more in-depth exploration of biological phenomena such as behavior, mating systems, heritability of adaptive traits, kin selection, and dispersal patterns (1; 2; 3; 4; 5). Correspondingly, there are a number of methods for sibling reconstruction from genetic data (6; 7; 8; 9; 10; 11; 12; 13). Yet, few existing methods allow for inevitable errors in those data and none scale to even moderately large data sets that are becoming common. In (14; 15), a consensus-based approach was introduced to deal with errors in the data. This proposed approach takes a set of reconstructions (possibly from various methods), specifically, solutions based on omission of part of the genetic data, and uses formal consensus techniques (16) to reconcile these solutions into one.

Consensus-based methods are particularly well suited to accommodate for errors in the genotype data since they reconcile the individual biases and errors of each input solution. This is especially important in kinship reconstruction since the likelihood and nature of errors and sampling biases is typically unknown (17). Furthermore, any sibling reconstruction method could theoretically be used as an input for the consensus reconstruction. For populations in which allele frequencies and the population overall are not widely sampled, the 2-allele algorithm (12) serves as an ideal input into the consensus reconstruction as it generates a parsimonious solution using a minimum set cover of the feasible sibling groups. However, performance of the 2-allele method is of concern as the minimum set cover approach has been shown to be NP-hard and inapproximable (18). Not only

is this a theoretical limitation, but, it is of great practical concern since the method, offered to the biological community as an online service (19), takes hours or days to reconstruct siblings in datasets of 500-1000 individuals.

To address these performance issues, a scalable method for 2-allele reconstruction was developed in (20). However, there was no fault-tolerance built into the reconstruction. Furthermore, if we simply utilize the approach in (20) with the fault-tolerant consensus reconstruction methods in (14; 15) to accommodate errors, many redundant, computationally expensive operations would be performed, resulting in little or no computational speedup.

In this thesis, a scalable approach to fault-tolerant sibling reconstruction which minimizes the amount of redundant computations in the 2-allele algorithm is investigated. In addition, we propose further performance improvements to (20) which have resulted in dramatic speedups. The proposed solution has been implemented on an 8-core, shared memory Linux machine. Our results have shown linear speedups over single-core execution, particularly for large data sets.

2. BACKGROUND AND PRIOR WORK

2.1 Microsatellites

Microsatellites (also known as SSRs and STRs), as shown in Figure 1, are sequences of short (one to six base pairs) nucleotide sequence repeats, such as $(CA/GT)_n$ or $(AGC/TCG)_n$ that are strewn thoughout eukaryotic genomes. Though other molecular markers exist, such as AFLPs (amplified fragment length polymorphisms), ISSRs (inter-simple sequence repeats), RAPDs (randomly amplified polymorphic DNA), and SNPs (Single Nucleotide Polymorphisms), microsatellite are the most widely used molecular marker for kinship analysis in wild populations (21).

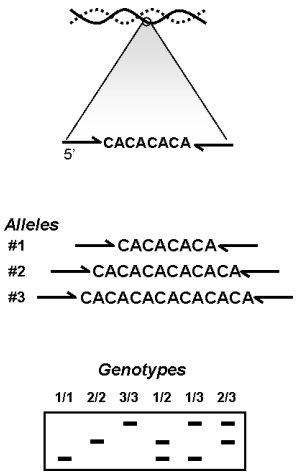

Figure 1. Microsatellite markers.

2.2 Sibling Reconstruction

From (12), the goal of the sibling reconstruction problem is to recover the full sibling relationships (i.e. - groups of individuals with the same parents) from a population. More formally:

Definition 1. Let U be a population of n diploid individuals of the same generation genotyped at l microsatellite loci:

$$U = \{X_1, \ldots, X_n\}, \text{ where } X_i = (\langle a_{i1}, b_{i1}\rangle, \ldots, \langle a_{il}, b_{il}\rangle)$$

and a_{ij} and b_{ij} are the two alleles of the individual i at locus j represented as some identifying string. Assuming no knowledge of the parental information, the formal goal is to find a partition of individuals P_1, ..., P_m such that:

$$\forall 1 \leq k \leq m, \forall X_p, X_q \in P_k : Parents(X_p) = Parents(X_q)$$

2.3 Mendelian Genetics

As stated in (21), Mendelian genetics provides us with a very simple kinship constraint: *an offspring inherits one allele from each of its parents for each locus*. From this (12), relates two necessary (but not sufficient) properties for full sibling groups in the absence of genotyping errors:

4-Allele Property: The total number of distinct alleles occurring at any locus may not exceed 4. Formally, a set of individuals $S \subseteq U$ has the 4-allele property if

$$\forall 1 \leq j \leq l : \left| \bigcup_{i \in s} \{a_{ij}, b_{ij}\} \right| \leq 4$$

.

Because a group of siblings can only inherit from a set of at most 4 parental alleles at any locus, all full sibling groups must necessarily satisfy this constraint.

2-Allele Property: There exists an assignment of individual alleles within a locus to maternal and paternal such that the number of distinct alleles assigned to each parent at this locus does not exceed 2. Formally, a set of individuals $S \subseteq U$ has the 2-allele property if for each individual X_i in each locus there exists an assignment of $a_{ij} = c_{ij}$ or $b_{ij} = c_{ij}$ (and the other allele assigned to c_{ij}) such that

$$\forall 1 \leq j \leq l : \left| \bigcup_{i \in s} \{c_{ij}\} \right| \leq 2 \text{ and } \left| \bigcup_{i \in s} \{\bar{c}_{ij}\} \right| \leq 2$$

In Table I from (12), five individuals have been sampled at two genetic loci - where each allele is represented by a number. Also, numbers within the same locus represent the same alleles. From Table I, we can see the 2-allele property is stricter than the 4-allele property. For instance, the potential sibling group consisting of individuals 1, 3 and 4 satisfies the 4-allele property but fails to satisfy the 2-allele property due to the existence of more than two alleles on the left side of locus l: $\{44, 28, 13\}$. Furthermore, there is no possible swapping of left and right side alleles that will reduce

Individual	Alleles $\langle a, b\rangle$ at locus 1	Alleles $\langle a, b\rangle$ at locus 2
1	44, 44	55, 27
2	12, 56	18, 39
3	28, 44	55, 18
4	13, 13	39, 27
5	28, 51	18, 39

TABLE I

AN EXAMPLE OF INPUT DATA FOR THE SIBLING RECONSTRUCTION PROBLEM.

the count below 2 because individuals 1 and 4, with their alleles of $44/44$ and $13/13$ respectively, already fill the capacity. Therefore, though the potential sibling group satisfies the 4-allele property, it is ruled infeasible due to its violation of the 2-allele property. As noted in (12), it should be observered that any two individuals trivially satisfy the 2-allele and 4-allele properties.

2.4 2-Allele Minimum Set Cover

An algorithm for full sibling reconstruction follows by taking the minimum set cover of all maximal feasible sibling groups that satisfy the 2-allele property (21). Recall a population of n diploid individuals of the same generation genotyped at l microsatellite loci:

$$U = \{X_1, \ldots, X_n\}, \text{ where } X_i = (\langle a_{i1}, b_{i1}\rangle, \ldots, \langle a_{il}, b_{il}\rangle)$$

and a_{ij} and b_{ij} are the two alleles of the individual i at locus j.

From (12), the goal of the Minimum 2-Allele Set Cover problem is to find the smallest number of subsets $S_1, \ldots, S_m$ such that each $S_i \subseteq U$ satisfies the 2-allele constraint and $\bigcup S_i = U$.

The algorithm, hereafter referred to as 2-ALLELE, reconstructs sibling relationships in two major phases: maximal feasible sibling group enumeration and minimum set cover generation. The enumeration of maximal feasible sibling groups begins by generating the initial candidate sets from all $\binom{n}{2}$ pairs of individuals in the population. For each of these unprocessed feasible sibling groups, every individual, if it does not already belong to the group, is added. If adding the individual violates either the 2-allele or 4-allele properties, the resulting group is discarded and processing resumes from the next individual. Another special case occurs if adding the individual causes the set of possible parental genotypes to differ from the original sibling group. If this is the case, the resulting group is copied and added to the set of unprocessed sibling groups for later processing. The original group then resumes processing from the next individual and with the previous individual removed. The enumeration phase of 2-ALLELE is complete when all candidate sibling groups have been processed and no new candidate sibling groups are generated. The final phase, minimum set cover generation, is completed using a commercial set cover solver. A flow chart for 2-ALLELE is shown in Figure 2.

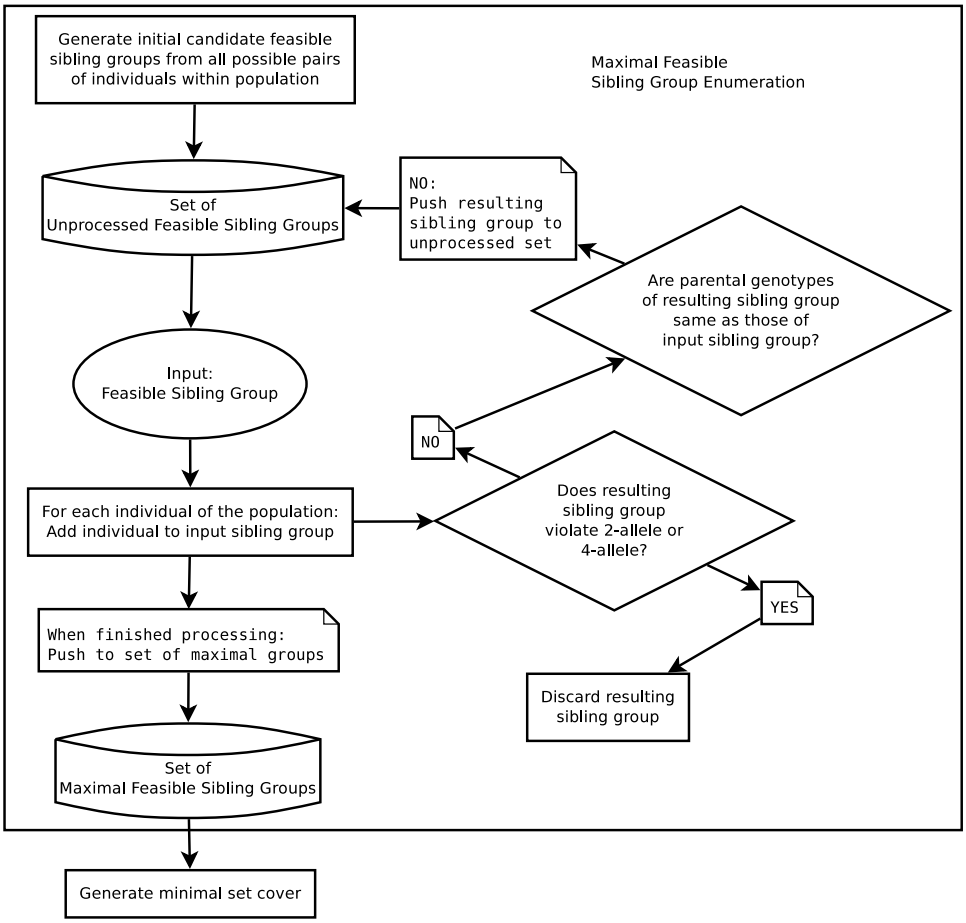

Figure 2. Flowchart for serial 2-Allele Minimum Set Cover Algorithm.

2.5 Parallele 2-Allele Minimum Set Cover

It was noted in (20) that, due to the combinatorial nature of feasible sibling group enumeration, generating maximal feasible groups becomes intractable for large data sets as the number of groups generated is exponential in the number of individuals. To address this performance bottleneck,

a novel loci decomposition was proposed. Unlike the serial, global enumeration in (12), the loci decomposition of (20) enumerates maximal feasible sibling groups at each locus independently. Due to the independent treatment of the loci, enumeration can then be processed in parallel. To ensure that a feasible sibling group satisfies the 2-allele and 4-allele constraints at all loci, a subsequent loci intersection phase is necessary. During the loci intersection phase, a pair of loci are merged by intersecting each of their sets of feasible sibling groups. All loci can eventually be merged in this fashion in $O(l)$ intersection rounds where l is the number of loci. Furthermore, by performing a data decomposition of groups present at each loci, the intersection phase can also be processed in parallel for each feasible sibling group. A flowchart of the parallel 2-allele algorithm is shown in Figure 3.

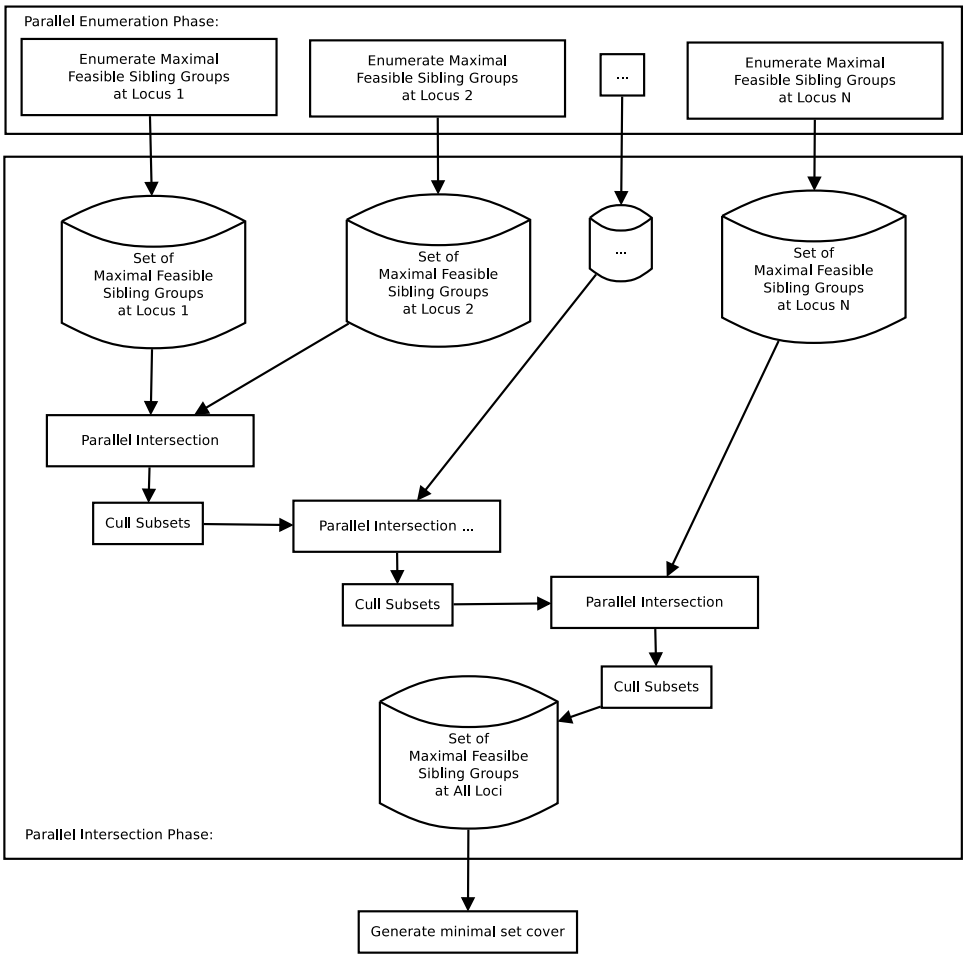

Figure 3. Flowchart for parallel 2-Allele Minimum Set Cover Algorithm.

2.6 Consensus-Based Full Sibling Reconstruction

A common assumption among many kinship reconstruction algorithms, including 2-ALLELE, is that the input genotype data is free of both errors and mutations. However, in practice, these errors cannot be avoided and identifying these errors without prior kinship information is a challenging

task (21). Sheikh *et al.* in (14; 15) proposed a new approach for full sibling reconstruction which tolerates both mutations and genotyping errors in the population data. From (21), the reasoning behind the error-tolerant reconstruction methods begins with considering an individual X_i which has some genotypying errors. Any error that is affecting sibling reconstruction must be preventing X_i's sibling relationship with at least one other individual X_j, who in reality is its sibling. Furthermore, it is unlikely that an error would cause two unrelated individuals to be paired up as siblings, unless all error-free loci do not contain enough information. Thus, a reconstruction method presents itself in which we can discard one locus at a time, under the assumption that it is erroneous, and obtain a full sibling solution based on the remaining loci. If all such solutions place individuals X_i amd X_j in the same sibling group (i.e. - there is a consensus among those solutions), we consider them to be siblings. However, the core of the error-tolerant approach presented in (14; 15) is concerned with classifying those pairs of individuals who do not consistently end up in the same sibling group (i.e. - there is no consensus about their sibling relationship). More formally, from (21), a consensus method can be defined as:

Definition 2. A *consensus* method for the sibling reconstruction problem is a computable function f that takes k solutions $S = \{S_1, \ldots, S_k\}$ as input and computes *one* final solution.

One approach to consensus reconstruction is to use a *strict consensus* which places two individuals into a sibling group only if they are together in all k input solutions (21). Though the method always arrives at a consistent solution, many singleton groups are produced (21). Instead, (14; 15)

proposed a *distance-based consensus* for sibling reconstruction. According to (14; 15), starting with a strict consensus of input solutions, distance-based consensus iteratively merges two sets until the quality of the solution cannot be improved. The quality of solution in (14; 15) is judged by seeking to minimize the number of groups. An interesting characeterstic of this consensus based approach is its modularity - any algorithm may be used to generate the set of k input solutions.

3. IMPROVEMENTS TO THE 2-ALLELE ALGORITHM

3.1 Serial 2-Allele Algorithm Review

Recall the 2-ALLELE algorithm introduced in (12) reconstructs sibling relationships in populations where obtaining parental information is infeasible and allele frequencies, as well as other parameters of the population, are not well sampled. The algorithm first enumerates all feasible sibling groups that satisfy Mendelian genetic constraints of possibly being siblings (the 2-allele constraint). From these enumerated groups, a commercial solver finds the optimal minimum set cover for the population, thereby obtaining the most parsimonious solution.

3.2 Parallel 2-Allele Algorithm Review and Issues

As shown in Figure 4, for even moderately large data sets, computation time for the 2-ALLELE algorithm can be quite costly. For instance, without parallelization, we can extrapolate that data sets in Figure 4 will take days to compute. It was noted in (20) that the enumeration of feasible sibling groups is computationally expensive due to its combinatorial nature. In the serial approach, feasible sibling groups are generated in a fashion consistent across all loci, where a sibling group is consistent at a locus if it does not violate the 2-allele property at that locus. In contrast, the parallel approach in (20) generates feasible sibling groups which are consistent only to a single locus; however, this loci decomposition allows the enumeration to proceed in parallel. In order to extract globally consistent feasible sibling groups, a series of intersection rounds are then performed across all loci to achieve the final solution. An intersection round consists of intersecting every feasible sibling group consistent at a locus with the results from the round prior, beginning with

the intersection of the first and second loci. To enhance performance, a data decomposition of feasible sibling groups from the round prior is performed to allow parallel processing; in other words, the intersections of sibling groups from the round prior with every sibling group from the current locus is done in parallel. Furthermore, sibling groups are internally represented as bit sets, allowing intersections to be performed using bitwise operators.

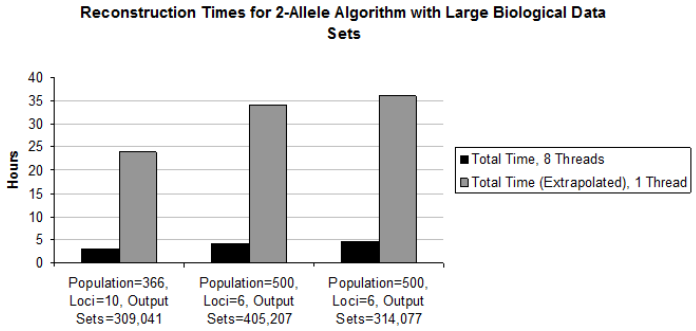

Figure 4. A comparison of actual times for the 2-allele algorithm on 8 cores versus extrapolated times on a single core.

The total number of feasible sibling groups after each intersection round is $O(a \cdot b)$, where a and b are the number of feasible groups from the round prior and the newly intersected locus respectively. In more formal terms:

$$a = \left|\{Sets_{locus_1} \cap Sets_{locus_2} \cap \ldots Sets_{locus_{i-1}}\}\right|$$

$$b = |\{Sets_{locus_i}\}|$$

Therefore, the number of feasible groups can grow exponentially with respect to the number of loci l since there are $l - 1$ intersection rounds. For input data sets with larger populations and/or loci counts, this can easily exceed the amount of available RAM as well as cripple performance; therefore, resulting feasible sibling groups which are subsets of other groups must be culled between each intersection round.

3.3 Parallel Transpose Subset Culling

To determine if a feasible sibling group is a subset of another sibling group, one must compare it against $O(s)$ other groups (where s is equal to the total number of feasible sibling groups resulting from the most recent intersection round). Therefore, as is the case in the original approach of Sheikh *et al.* (20), a subset culling algorithm would make $O(s^2)$ comparisons in order to cull all subsets within a single round. For larger data sets, which generate feasible groups in the tens of millions, our measurements show subset culling rapidly becomes a processing bottleneck, accounting for 80% or more of the total computation time.

To address these performance issues, we have developed a new method for culling subsets, which stems from the following observations:

1. The collection of feasible sibling groups can be represented as a bit matrix.

2. Each feasible sibling group contains, on average, approximately 2% of the population. In other words, the number of feasible groups, s, greatly exceeds the average number of individuals, m, which belong to any sibling group (i.e.; $s \gg m$).

3. In practice, 55% to 85% of feasible siblings groups are not subsets of any other sibling group, and, therefore, will not be culled.

An example bit matrix representative of data typically encountered is shown in Figure 5. Figure 5 shows a collection of feasible sibling groups represented as a 2-d bit matrix M. Each row of M (e.g. the row in the grey box), is a feasible sibling group. Furthermore, $M_{ij} = \{1$ iff individual$_j$ is a member of sibling group$_i$; and 0 otherwise$\}$. From the third observation, we notice that a majority of feasible sibling groups are not subsets of any other group. Therefore, in the original approach of Sheikh *et al.* (20), these groups are the absolute worst case because every single element of the bit matrix must be compared to determine this superset classification. Therefore, a majority of the time, the approach presented in (20) will perform the worst case number of bit comparisons, $s{\cdot}n$, where n is the size of the population, whenever a feasible sibling group is classified as a superset.

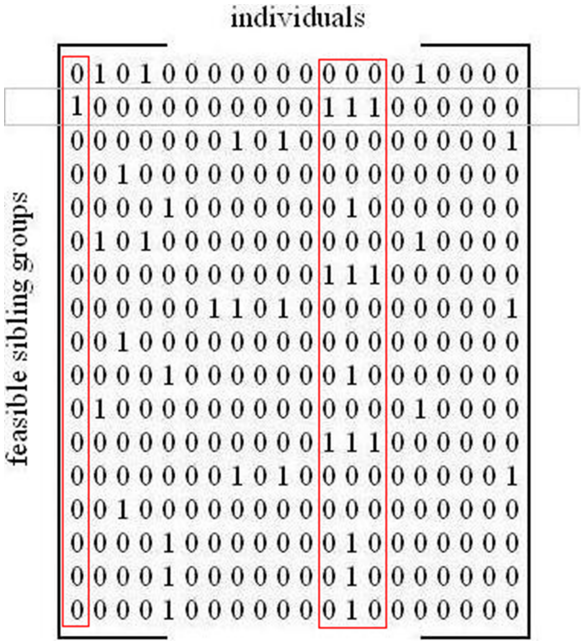

Figure 5. A collection of feasible sibling groups represented as a 2-d bit matrix *M*. A sibling group in the second row, which is not a subset of any other group, can be classified as such using only the bit comparisons bordered in red. Note, however, that an actual bit matrix will have substantially many more rows than columns.

The transpose subset culling algorithm follows from observing that the columns of the bit matrix have meaning too: column$_i$ defines which feasible sibling groups contain individual$_i$ as a member. Furthermore, if we bitwise intersect the columns corresponding to individuals within a sibling group, the resulting bit vector will have a 1 for each sibling group that also contain those individuals. Therefore, if a group is a subset of another group, the resulting bit vector will contain multiple 1s.

In contrast, if a group is not a subset, the resulting bit vector will contain only a single 1. In other words, given a sibling group $S = \{i_1, \ldots, i_k\}$, we intersect columns $i_1, \ldots, i_k$ to find which sibling groups contain all of these individuals. If the resulting intersection contains multiples 1's, then S is a subset of another feasible sibling group (specifically, those groups corresponding to the 1's in the intersection vector). This observation is demonstrated in Figure 6 and Figure 7.

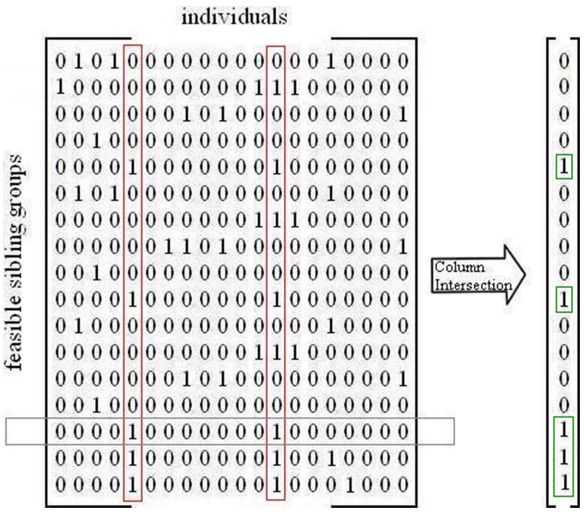

Figure 6. In this case, when intersecting the columns in red for the sibling group in the grey box, the resulting bit vector contains multiple 1s. This means the sibling group in the grey box is a subset of another sibling group

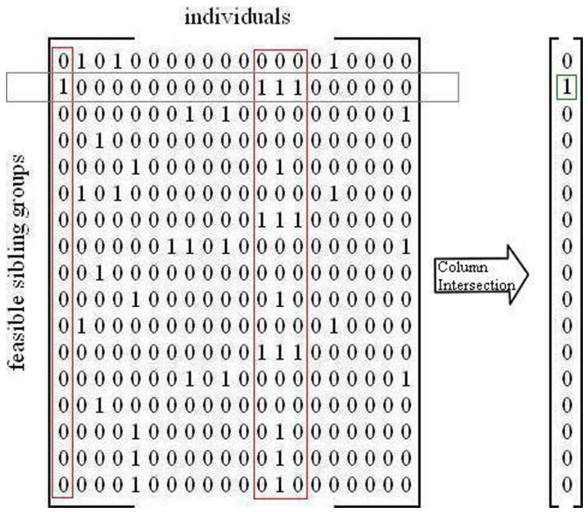

Figure 7. In this case, when intersecting the columns in red for the sibling group in the grey box, the resulting bit vector contains only a single 1. This will occur only for sibling groups which are not subsets of any other group.

3.4 Performance of Parallel Transpose Subset Culling

As shown in Figure 6 and Figure 7, by intersecting the columns containing only the member individuals of a sibling group, the group can be classified as a subset (or not) in $m{\cdot}s$ bit comparisons, where m is the number of individuals belonging to the sibling group and s is the total number of feasible sibling groups. Following from the third observation, since $s \gg n$, where n is the size of the population, this subset culling method of intersecting only the relevant columns of the bit matrix results in substantial performance savings over the approach in (20) (which frequently performs $n{\cdot}s$ bit comparisons in the worst case) as shown in Figure 8.

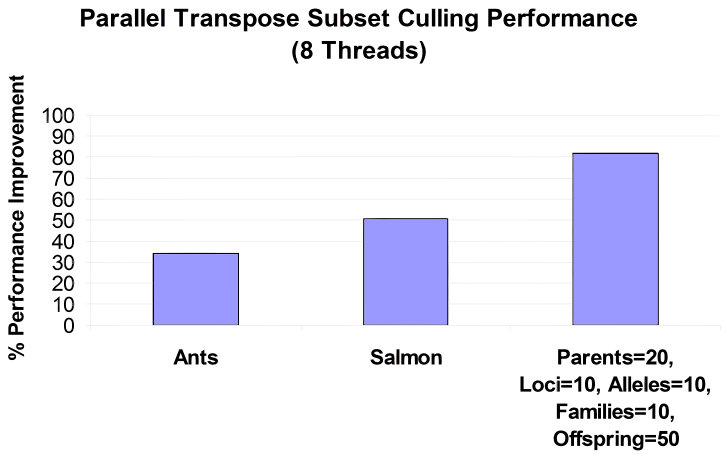

Figure 8. The percentage performance improvement of a parallel transpose subset culling method over the subset culling algorithm in Sheikh *et al.* (20)

The main advantage of the transpose method is that it avoids checking elements of the bit matrix that do not have any association with the sibling group being processed. The first two columns, Ants and Salmon, represent actual biological data sets; the last column is from synthetically generated data. These results were gathered on an 8-core, shared memory Linux machine.

The transpose subset culling approach is also highly data-parallel. Both the transpose matrix conversion and subset determination phases, see Figure 9, can be independently computed for each feasible sibling group. Therefore, this processing bottleneck is now highly scalable on multi-core systems.

Pseudo code of the parallel transpose subset culling algorithm is presented in Figure 9.

Figure 9. Procedure for identifying subsets in parallel.

```
/* M is a bit matrix of feasible sibling groups. */
ParallelTransposeSubsetCulling(bit matrix M)
  /* Take the transpose of M in order to place columns into contiguous
  memory. This allows bitwise operators to be used on columns. */
  T ⇐ parallelComputeTransposeOf(M)
  parallel for each row G_x in M do
    /* Note: each row of M is a sibling group. */
    y ⇐ G_x.getFirstTrueBitPos()
    Result ⇐ T.getRow(y)
    while z ⇐ G_x.getNextTrueBitPos() do
      Result ⇐ Result.bitwiseIntersect(T.getRow(z))
    end while
    if Result.hasMultipleBitsSetToTrue() then
      markAsSubset(Result)
    end if
  end for
```

4. PARALLEL, FAULT-TOLERANT SIBLING RECONSTRUCTION

4.1 Serial Consensus Algorithm Review

Often real world data contains sequencing errors which the 2-allele algorithm alone does not address. These errors are accommodated through a consensus approach (14; 15), which assumes error at a given locus, removes the said locus from computation, and generates a solution using the remaining loci. This locus drop-out procedure is iteratively performed for all loci. The resulting solution sets are initially merged using a strict consensus voting scheme, targeting feasible sibling pairings for which there is no contention among the solution sets. A minimum cost based heuristic is then used to greedily merge the remaining contentious sets until a cost threshold is breached.

4.2 Parallel Consensus Sibling Reconstruction with 2-allele

The consensus algorithm is trivially parallelizable by simply computing each locus drop-out solution independently. However, when using the 2-allele algorithm to compute each solution, a great deal of redundant work is done in the loci intersection phase. This can be seen by comparing the intersections made when computing solutions for $locus_i$ and $locus_{i+1}$:

- Locus drop-out solution for $locus_i$: $locus_1 \cap locus_2 \cap \ldots locus_{i-1} \cap locus_{i+1} \cap \ldots locus_n$
- Locus drop-out solution for $locus_{i+1}$: $locus_1 \cap locus_2 \cap \ldots locus_{i-1} \cap locus_i \cap \ldots locus_n$

Notice the large number of identical loci intersection operations performed for both solutions. In fact, the total amount of loci intersections performed using this straight forward approach is $O(l^2 - 2l)$, where l is the number of loci. The avoidance of redundant work is crucial to the performance of a

consensus sibling reconstruction. For the first data set shown in Figure 4, a consensus reconstruction using the methods in (15) with 10 loci could take up to a week. Furthermore, performance of the loci intersections is highly sensitive to the sequence in which they are intersected. For instance, the following two sequences of intersecting loci arrive at equivalent solutions but can have drastically different performance characteristics:

1. $((locus_1 \cap locus_2) \cap locus_3) \cap locus_4$
2. $(locus_1 \cap locus_2) \cap (locus_3 \cap locus_4)$

In practice, the first sequence has better performance characteristics than the second. Furthermore, the data has shown that intersecting loci with the greatest number of feasible sibling groups (i.e. the highest ordered loci) first also performs best in practice. To explain this, recall that when intersecting the feasible sibling groups between two loci sets, each sibling group in the first loci set is intersected with every group in the second loci set. Therefore, the amount of intermediate sibling groups that must be processed and potentially culled is $O(a \cdot b)$, where a and b are the number of feasible groups from the first and second loci sets respectively. Hence, the intuitive explanation for the performance characteristics of both of these use cases - intersection order and loci order - is that we want to be careful to intersect the smallest loci sets possible, otherwise, we risk paying a substantial processing cost due to variability in culling of feasible sibling groups and the quadratic nature of the loci intersection operation.

Based on these observations, we have designed an algorithm that minimizes redundant loci intersection operations, performs only incremental intersections rather than between large groups of

combined loci, and performs intersections from highest ordered loci to lowest ordered loci. Pseudocode for this algorithm is provided in Figure 10.

Figure 10. Procedure for computing loci drop-out solutions.

```
/* L is a sorted set of loci. C is a cache of loci intersections */
CompLociDropoutSolnsRecur(L, C)
  /* Base case: we only have a single locus to process. */
  if L.SIZE() == 1 then
    OUTPUTSOLUTION(L, C)
    return
  end if

  /* Split set of loci in half. */
  L.SPLIT(l_half, r_half)

  /* Recurse down right child tree. */
  l_cache ⇐ l_half.PARALLELINTERSECTALL(C, intermed_cache)

  COMPLOCIDROPOUTSOLNSRECUR(r_half, l_cache)

  /* Conditionally recurse down left child tree, else perform quadratic procedure. */
  if SHOULDRECURSEDOWNLEFTCHILDTREE(l_half, r_half, thresh)
  then
    r_cache ⇐ r_half.PARALLELINTERSECTALL(C, NULL)
    COMPLOCIDROPOUTSOLNSRECUR(l_half, r_cache)
  else
    COMPLOCIDROPOUTSOLNSQUADRAT(l_half, intermed_cache)
  end if
```

4.3 Performance of Parallel Consensus Sibling Reconstruction with 2-allele

The main part of the algorithm described by Figure 10 is a recursive splitting routine. At each recursion level, the local set of loci is split in half and then halves are processed independently through subsequent recursion calls. For sample loci sets in which the loci orders are relatively similar, the algorithm performs $O(l \cdot log_2(l))$ loci intersection operations, where l is the number of loci. A sample recursion tree for this best case is shown in Figure 11.

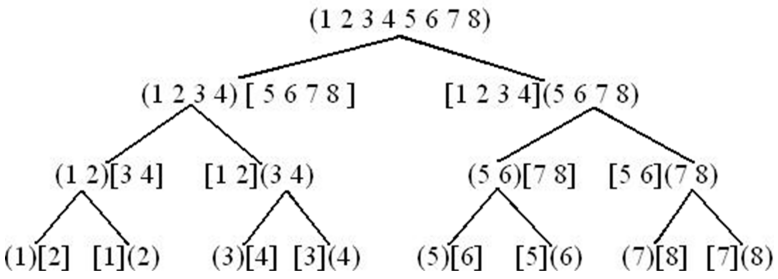

Figure 11. A recursion tree for a well balanced set of loci. Each number i represents the feasible sibling groups at the i-th locus. Loci enclosed in parentheses have not yet been intersected. In contrast, loci enclosed in square brackets have been intersected (including all intersected loci from parent nodes).

However, because there can be a great size disparity between loci in the left and right halves, the left child recursion tree is not always computed. Instead, if the size disparity exceeds a heuristic

threshold, a specialized quadratic method is used for the loci in the left half. This quadratic method is similar to the method described previously except it will re-use cached loci intersections computed for the loci in the right half.

The performance gains from this recursive splitting algorithm are shown in Figure 12. These results were obtained on a Dell PowerEdge 2900 III Server with two Quad Core Processors (Intel Xeon E5430, 2x6MB Cache, 2.66GHz, 1333MHz FSB) with 32GB RAM.

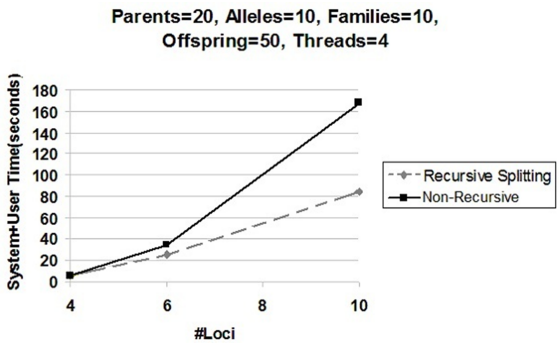

Figure 12. A performance comparison between the non-recursive splitting loci intersection algorithm and a recursive splitting algorithm using the consensus methods of (15).

In the worst case, this algorithm will perform $O(l^2/4 - l)$ loci intersections. However, it is important to note that this algorithm performs relatively well against the non-recursive method because it reduces redundant loci intersections in a fashion that avoids performing intersections

between large groups of combined loci and performs intersections from highest ordered loci to lowest ordered.

Furthermore, using a parallel 2-allele implementation, the consensus algorithm now scales very well to multiple processors as shown in Figure 13. We can also see that the performance benefits from increasing thread count are attributable to the highly parallel loci intersections phase as shown in Figure 14.

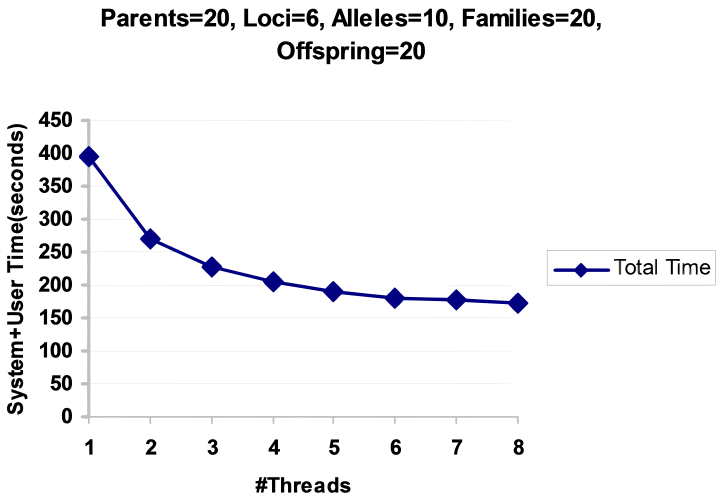

Figure 13. Performance of parallel, fault tolerant sibling reconstruction with increasing thread count.

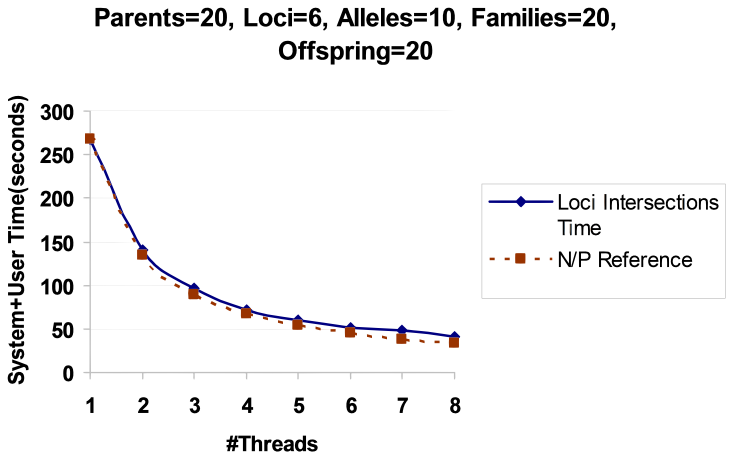

Figure 14. Performance of the computationally expensive loci intersection phase with increasing thread count

5. CONCLUSION

This thesis has presented a technique for parallel, fault-tolerant sibling reconstruction that minimizes the amount of redundant computations when using the 2-allele algorithm. This has resulted in as high as a 50% performance improvement for profiled data sets. Furthermore, it has been shown that this technique is linearly scalable to multiple processors. In addition, this thesis has presented a novel, scalable method for culling unnecessary feasible sibling groups which are subsets of other feasible sibling groups. This subset culling method has resulted in as high as 50% performance improvement over the scalable, 2-allele algorithm presented in (20). The subset culling is a general technique that can be applicable in any context of intersection of a large number of subsets of a set.

CITED LITERATURE

1. da Silva, A. G., Eberhard, J. R., Wright, T. F., Avery, M. L., and Russello, M. A.: Genetic evidence for high propagule pressure and long-distance dispersal in monk parakeet (myiopsitta monachus) invasive populations. Mol Ecol, 19(16):3336–3350, August 2010.

2. Lepais, O., Darvill, B., O'Connor, S., Osborne, J. L., Sanderson, R. A., Cussans, J., Goffe, L., and Goulson, D.: Estimation of bumblebee queen dispersal distances using sibship reconstruction method. Mol Ecol, 19(4):819–831, February 2010.

3. Moritz, R., Haddad, N., Bataieneh, A., Shalmon, B., and Hefetz, A.: Invasion of the dwarf honeybee (apis orea) into the near east. Biological Invasions, 12:1093–1099, 2010.

4. Newton-Fisher, N. E., Thompson, M. E., Reynolds, V., Boesch, C., and Vigilant, L.: Paternity and social rank in wild chimpanzees (pan troglodytes) from the budongo forest, uganda. American Journal of Physical Anthropology, 142(3):417–428, July 2010.

5. Zalewski, A., Michalska-Parda, A., Bartoszewicz, M., Kozakiewicz, M., and Brzezinski, M.: Multiple introductions determine the genetic structure of an invasive species population: American mink neovison vison in poland. Biological Conservation, 143(6):1355–1363, July 2010.

6. Jones, A. G., Small, C. M., Paczolt, K. A., and Ratterman, N. L.: A practical guide to methods of parentage analysis. Molecular Ecology Resources, 10(1):6–30, January 2010.

7. Almudevar, A.: A simulated annealing algorithm for maximum likelihood pedigree reconstruction. Theoretical Population Biology, 63(2):63–75, 2003.

8. Wang, J.: A practical guide to methods of parentage analysis. Genetics, 166:1968–1979, 2004.

9. Beyer, J. and May, B.: A graph-theoretic approach to the partition of individuals into full-sib families. Molecular Ecology, 12:2243–2250, 2003.

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

11. Thomas, S. C. and Hill, W. G.: Sibship reconstruction in hierarchical population structures using markov chain monte carlo techniques. Genet. Res., 79:227–234, 2002.

12. Berger-Wolf, T. Y., DasGupta, B., Chaovalitwongse, W., and Ashley, M. V.: Combinatorial reconstruction of sibling relationships. In Proceedings of 6th International Symposium on Computational Biology and Genome Informatics (CBGI 05), pages 1252–1255, 2005.

13. Berger-Wolf, T. Y., Sheikh, S. I., Dasgupta, B., Ashley, M. V., Caballero, I. C., Chaovalitwongse, W., and Putrevu, S. L.: Reconstructing sibling relationships in wild populations. Bioinformatics, 23(13), 2007.

14. Sheikh, S. I., Berger-Wolf, T. Y., Khokhar, A. A., and DasGupta, B.: Consensus methods for reconstruction of sibling relationships from genetic data. In Proceedings of 4th Workshop on Advances in Preference Handling, 2008.

15. Sheikh, S., Berger-Wolf, T. Y., Ashley, M. V., Caballero, I. C., Chaovalitwongse, W., and DasGupta, B.: Error tolerant sibship reconstruction in wild populations. In Proceedings of 7th International Conference on Computational Systems Bioinformatics (CSB), eds. P. Markstein and Y. Xu, pages 273–284. World Scientific Publishers, August 2008.

16. eds. M. Janowitz, F. Lapointe, F. McMorris, B. Mirkin, and F. Roberts Bioconsensus, volume 61 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 2001.

17. Hoffman, J. I. and Amos, W.: Microsatellite genotyping errors: detection approaches, common sources and consequences for paternal exclusion. Molecular Ecology, 14(2):599–612, 2005.

18. Ashley, M. V., Berger-Wolf, T. Y., Berman, P., Chaovalitwongse, W., DasGupta, B., and Kao, M.: On approximating four covering and packing problems. J Computer and System Science, 2009.

19. Ashley, M. V., Caballero, I. C., Chaovalitwongse, W., DasGupta, B., Govindan, P., Sheikh, S. I., and Berger-Wolf, T. Y.: Kinalyzer, a computer program for reconstructing sibling groups. Molecular Ecology Resources, 9(4):1127–1131, July 2009.

20. Sheikh, S. I., Khokhar, A. A., and Berger-Wolf, T. Y.: Efficient and scalable parallel reconstruction of sibling relationships from genetic data in wild populations. In Proceedings of Tenth IEEE International Workshop on High Performance Computational Biology (HiCOMB 2010) held with IPDPS., 2010.

21. Ashley, M. V., Berger-Wolf, T. Y., Caballero, I. C., Chaovalitwongse, W., DasGupta, B., Sheikh, S. I., et al.: Computational Biology: New Research, pages 231–258. Nova Science Publishers, 2009.

VITA

NAME: Alan J Perez-Rathke

EDUCATION: M.S., Computer Science,
University of Illinois,
Chicago, Illinois, 2011.

B.S., Computer Science,
University of Illinois,
Champaign-Urbana, Illinois 2005.

EXPERIENCE: 2010-2011 Research Assistant,
Computational Population Biology Lab,
Department of Computer Science,
University of Illinois,
Chicago, Illinois.

2005-2009 Software Engineer,
Midway Amusement Games, LLC,
Chicago, Illinois