Rapid Haplotype Inference for Nuclear Families

Amy L. Williams [1,2] David E. Housman [2] Martin C. Rinard [1] David K. Gifford [1]

[1] Computer Science and Artificial Intelligence Laboratory and
[2] Center for Cancer Research,
Massachusetts Institute of Technology, Cambridge, MA, USA

Address for correspondence and reprints:
Amy Williams
MIT CSAIL
32 Vassar St, 32-G536
Cambridge, MA 02139, USA

Running head: Hapi: Rapid Haplotyping for Families

Keywords: Haplotypes, Dynamic Programming, Genotypes, SNPs

Abstract

We have developed Hapi, an efficient algorithm for haplotyping nuclear familes that dramatically outperforms existing haplotyping algorithms. Other family-based haplotyping approaches have hard limits on the size and number of families they can analyze because of their exponential complexity. Effective algorithms for family-based haplotyping are essential because haplotypes currently produced by algorithms for unrelated individuals suffer from poor accuracy beyond very short genomic distances. Hapi's efficiency comes from applying newly-discovered insights into the haplotyping problem that enable the elimination of large numbers of unnecessary states. Our approach is very effective: when applied to a dataset containing 103 families, Hapi performs between 5.5–517 times faster than a state-of-the-art algorithm. Because of its efficiency, Hapi is capable of addressing large family-based datasets which can now be produced by increasingly sophisticated genotyping and DNA sequencing methodologies and should have broad applicability.

Introduction

The emergence of high throughput genotyping technologies has enabled rapid, low-cost assays of single nucleotide polymorphisms (SNPs) in large datasets of human subjects. These genotype data provide two unordered allele values at each queried genomic position, with each allele derived from the two homologous chromosomes in a diploid cell. However, genotype data do not identify which variant is present on each homologous chromosome. A haplotype, or the assignment of these phase relationships among genotyped markers, is an important element in studies directed towards gene discovery.

Genome scale haplotypes cannot be discovered using direct molecular means at present, so computational methods must be used to infer them. For datasets of unrelated individuals, these computational techniques leverage probabilistic constraints governed by mathematical models of population dynamics. For related individuals, constraints are induced by the combination of the observed genotypes, the Mendelian laws of inheritance implied by the relationships between individuals. Algorithms for haplotyping unrelated individuals[1,2] include PHASE,[3] HAPLOTYPER,[4] and HAP2.[5,6] The models these algorithms approximate are often insufficient to prevent switch errors (i.e., positions with incorrectly assigned haplotypes relative to the previous heterozygous locus[6,3]) except across short genomic distances. Data from unrelated individuals have been used in many studies, notably in the International HapMap project,[7] but such studies can only discover information about the results of meiosis (including the location of hotspots) averaged across thousands of generations and both genders. On the other hand, algorithms that apply to individuals with known relationships[8–14] exploit constraints based on the expectations of Mendelian inheritance and genetic linkage, inferring haplotypes in a more accurate and direct manner. Additionally, these datasets and algorithms enable the identification of the probable sites of *de novo* meiotic recombinations and gene conversions (which appear as short double crossovers), and have been used to build genetic maps of recombination rates[15] and identify hotspots.[16] They can also be used to perform linkage analysis to study the genetic basis of disease within families.

Hapi is a new dynamic programming algorithm that infers both minimum-recombinant and maximum likelihood haplotypes, and is substantially faster than all other haplotyping algorithms for the nuclear family problem. Nuclear family derived genotypic data identifies parents and their children, but provides no information about relationships within a larger pedigree. Minimum-recombinant haplotypes assign family members' genotypes to homologs such that the number of recombinations that occur in the homologs the parents transmitted to the children is minimized. Maximum-likelihood haplotypes utilize user-provided recombination frequencies between successive loci to calculate the most likely haplotypes. Maximum likelihood haplotypes are often substantially similar or identical to minimum-recombinant haplotypes. Both approaches to haplotype estimation have strengths and weaknesses. Minimum-recombinant haplotyping may yield suboptimal results when the recombination frequencies between loci in some region varies widely. For example, because recombination frequencies are correlated with genomic distance, this can occur if the distance between some loci is much larger or smaller than others. Maximum-likelihood haplotyping reports only the most likely haplotype, a feature that can be misleading to a user when the difference in probability to alternate haplotype is small. Typically this occurs when the number of recombinations across the alternate haplotypes are the same, and in such a case, minimum-recombinant haplotyping reports the ambiguities. It is worth noting that geneticists manually perform minimum-recombinant haplotype assignment when analyzing small datasets. Hapi enables this approach to be applied to the very large datasets currently produced by high-throughput SNP genotyping.

Several existing programs for haplotyping related individuals are based on the Lander-Green algorithm,[8] and include Merlin,[9] GENEHUNTER,[10,11] and Allegro.[12] These algorithms use hidden Markov models (HMMs) to obtain a probability distribution of haplotype assignments for individuals in a pedigree; typically they are used to find the maximum likelihood haplotype estimate. The state space for these HMMs is composed of inheritance vectors at each locus that are bit strings encoding which chromosome homolog a parent transmitted for each child in the pedigree at that locus. This state space is inherently exponential, with 2^{2n} possible values, where n is the number of non-founders or individuals with at least one parent in the pedigree. Although Merlin, GENEHUNTER, and Allegro contain optimizations that greatly reduce the size of the state space they examine, all are relatively inefficient; in general, each requires exponential time in the number of non-founders in the pedigree.

Superlink[17] is another maximum likelihood haplotyping algorithm that uses Bayesian networks. While Superlink employs several optimizations to improve its efficiency, it performed slower than Merlin in our experiments.

To compare Hapi's runtime performance with existing work, we ran Merlin, Superlink, and Hapi on a dataset containing 103 nuclear families. (We excluded GENEHUNTER and Allegro from these comparisons since Merlin has been shown to be faster than each of these.[9]) Hapi performed 5.5–517 times faster than Merlin and 22–1091 times faster than Superlink (see Results). We also compared Hapi to PedPhase 2.0,[14] a system for computing minimum-recombinant haplotypes. Whereas Hapi finished in 4.732 seconds, PedPhase did not complete analyzing even one chromosome in over six hours time.

Methods

Hapi is similar to existing haplotyping approaches that use HMMs and dynamic programming, but it employs a novel set of optimizations that dramatically improve its efficiency. These optimizations are based on properties we discovered about the haplotyping problem that facilitate the elimination of a large number of states from the analysis. There are four primary optimizations. (1) For each locus, Hapi only builds states that are consistent with Mendel's laws given the genotypes of the individuals. (This optimization will give results consistent with other haplotyping strategies. Details about this issue are not readily apparent in all related papers, but the paper describing GENEHUNTER shows that it assigns a 0 probability to states that are inconsistent with individuals' genotypes.[11]) (2) When a parent is homozygous, Hapi only builds states in which the homolog that parent transmitted does not exhibit recombination. This optimization is natural in the case of minimum-recombinant haplotyping, but requires special handling in the case of maximum-likelihood haplotyping as we discuss later. (3) We discovered that some states are equivalent in the recombinations they will exhibit downstream of a given locus. Hapi detects these kinds of states and retains only the one with minimum recombinations or maximum likelihood. (4) Finally, at loci where Mendelian inheritance cannot unambiguously infer for a set of children which parent transmitted each allele, we use a novel, concise representation of the ambiguities instead of forming an exponential number of states for all the possibilities across all the children. More details about each of these optimizations will be given later in this section.

We begin by describing our minimum-recombinant haplotyping algorithm. Later we will describe how to use the same basic strategy to calculate maximum likelihood haplotypes.

Hapi uses inheritance vectors, represented using bit strings, to encode which chromosome homolog each parent transmitted to each child at a locus. These bit strings are composed of $2c$ bits, where c is the number of children in the nuclear family.

A dynamic programming equation for calculating minimum-recombinant haplotypes is given below. The function $R(l, \vec{v})$ calculates the minimal number of recombinations necessary to reach inheritance vector $\vec{v}$ at locus l.

$$R(l, \vec{v}) = \min_{\vec{w}}\{R(l-1, \vec{w}) + d(\vec{w}, \vec{v})\} \tag{1}$$

Here, $R(l-1, \vec{w})$ is the minimum number of recombinations necessary to reach an inheritance vector $\vec{w}$ at the previous locus $l-1$. $d(\vec{w}, \vec{v})$ is the number of recombinations between vectors $\vec{w}$ and $\vec{v}$, which can easily be implemented as the number of bits that differ between them, i.e., the hamming distance. The initial recombination value at locus $l = 0$ is defined natural as $R(l = 0, \vec{v}) = 0$.

A naive implementation of the above dynamic programming recurrence would initialize all 2^{2c} possible inheritance vectors at locus $l = 0$ and would model most or all of these vectors at successive loci. Hapi functions differently: the initial locus has only one inheritance vector, and successive loci model a very small number of inheritance vectors.

Hapi uses what we term a locus *state* to store the information computed in the above dynamic programming equation. A locus state stores: (1) an inheritance vector; (2) the assignment of the heterozygous parent(s) genotype alleles to homologs that is consistent with this inheritance vector; (3) the minimal number of recombinations necessary to reach this state/inheritance vector value; (4) a pointer to the state or states at the previous locus that yields this minimal number of recombinations; and (5) a bit string encoding which children have ambiguous inheritance values (necessary for some kinds of loci as we describe later). Because the parents' allele to homolog assignments imply part or all of the inheritance vector values, there is only one consistent parent assignment for each inheritance vector. Note that it is possible for multiple states at the previous locus to yield the minimum number of recombinations at the current locus, i.e., there are sometimes ambiguities. When Hapi determines the overall minimum-recombinant solution, it marks ambiguities for the applicable loci and individuals and outputs this information to the user.

After evaluating equation (1) by building the necessary states for all loci, it is straight forward to deduce haplotypes. Hapi does this by performing the assignments of alleles to homologs as dictated by the minimum-recombinant state at the final locus and then back tracing to states at previous loci. Performing these assignments is simple for the parents since the state contains the assignment of parent alleles to homologs. For the children, the inheritance vectors encode a bit for each parent–child relationship which designates which homolog the parent transmitted to the child at that locus. Using this information, the system locates the alleles that each parent transmitted and assigns each to the homolog designated to have been transmitted by the given parent. After performing these assignments, Hapi back traces by following the pointer from the final state to the state at the previous locus and repeating the process again. Rather than

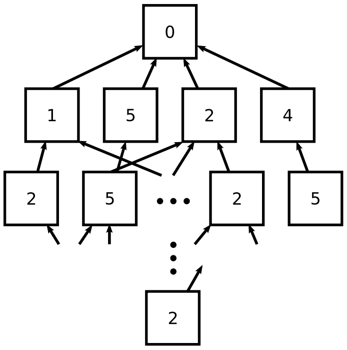


Figure 1: **Example Graph of States Across Several Loci** A pictorial representation showing the relationship between states at different loci. Each row of boxes correspond to a locus; boxes represent a state and indicate the numbers of recombinations the state incurs; arrows point to previous state(s). Once the system deduces a single state at some locus—shown here as the bottom box—it back traces by traversing the pointers and assigns the haplotype values based on the states it encounters. The numbers are not from real data.

Locus Type	Parent p	Parent q
Fully Informative for Both Parents	a/b	a/c or c/d
Fully Informative for One Parent	a/b	a/a or c/c
Partly Informative	a/b	a/b
Uninformative	a/a	a/a or b/b

Figure 2: **The Four Locus Types Hapi Distinguishes** The four types of loci our algorithm handles separately with the names we use to refer to them. Either parent may have the genotypes listed for parents p and q.

waiting until the final locus to make these assignments, Hapi does this work whenever a locus yields only one state (which happens frequently). That one state and those leading to it are guaranteed to have minimum recombinations. Performing this process before the final locus allows the system to reclaim the memory used to store states.

We give an illustrative example of what the graph of states generated by our algorithm may look like in Figure 1. In this graph, boxes represent states, and each row of boxes corresponds to the states for a single locus. The number in the box represents the minimal number of recombinations necessary to reach that state. The first locus (top-most box) has only one box/state with an initial value of 0 recombinations. At the second locus, there are four states that have between 1 and 5 recombinations. Note that at the third locus, the second state has pointers to two different states at the previous locus. The final locus has only one state. Once the system determines this final state, it performs back tracing along pointers to previous states, and uses the haplotype values stored in the encountered states to make the allele assignments as we just described.

We now describe how we produce states based on four locus types. The manner in which Hapi processes a locus depends on the genotypes of the parents at that locus. Figure 2 lists the locus types Hapi distinguishes and gives the names we use to refer to them.

Fully Informative for Both Parents Loci Two of the primary optimizations Hapi employs are extremely effective in the case of loci that are fully informative for both parents. The first optimization is to only build states that are consistent with observed genotypes. Both parents are heterozygous but have differing genotypes at this type of locus. In this case, it is simple to deduce which allele each parent transmitted to each child; thus, for a particular assignment of the parents' alleles to homologs, there is only one possible inheritance vector. With two possible allele assignments for each of the two parents, there are exactly four possible inheritance vectors/states at this type of locus.

The inheritance vectors across these four states differ in a specific pattern that enables us to apply Hapi's second optimization. Consider two states that have opposite assignments of one parent's alleles to homologs and the same allele assignment for the other parent. The inheritance vector values corresponding to the fixed parent are the same across these states, and the inheritance vector values corresponding to the differing parent are exactly opposite each other. This is the case because the parent with opposite allele assignments transmits the same allele to each child regardless of which homolog the alleles are assigned to: opposite allele assignments yield opposite inheritance bit values. Figure 3 shows an example of two states with opposite assignments of alleles to homologs for one parent, and the resulting inheritance vectors have opposite values for this parent. Although this figure shows a fully informative for one parent locus, the inheritance vector values shown apply to the current scenario where one parent has fixed allele assignments.

These two inheritance vectors with opposite values for one parent are equivalent to each other in terms of the number and location of recombinations induced at downstream loci. Hapi uses inheritance vectors to detect recombinations. Recombinations are exhibited by a difference in the homolog transmitted to one or more of the children between two loci. Because the inheritance values in these states are exactly opposite each other, each of these inheritance vectors encodes the same set of children as receiving a given homolog. The two states merely use opposite labels for the homologs as implied by the opposite assignments of the parent's alleles to homologs. Choosing one of the states instead of the other results in the downstream loci having opposite assignments of the parent's alleles homologs, consistent with the assignment made in the chosen state. The resulting downstream recombinations are the same regardless of which state the system chooses because the sets of children that share a common homolog (receive the same homolog "label") are the same between states.

This property of equivalent labelings applies across all four possible states at a fully informative for both parent locus. The states differ from each other in the assignments of one or both of the parents and the inheritance vectors corresponding to each of the parents are either identical or are exactly opposite each other. Because of this equivalent labeling among states, it suffices to eliminate all but one of them. Hapi therefore retains only the state that has the minimum number of recombinations.

A locus that is fully informative for both parents allows the ideal application of the optimizations in Hapi. No matter how many states are stored at the previous locus, only one state results at this type of locus.

Fully Informative for One Parent Loci At loci that are fully informative for one parent, Hapi applies three of its four optimizations, including the two optimizations just described. This type of locus has one heterozygous parent, and we can easily deduce half of the inheritance vector values—those corresponding to the heterozygous parent—for the two possible assignments of that parent's alleles to homologs. For the other half of the inheritance vector values—those corresponding to the homozygous parent—the data give no information about which homolog the parent transmitted. Rather than building states that contain all 2^c possible inheritance vector values for the transmissions of the homozygous parent, we simply propagate the inheritance values from each of the states at the previous locus. This approach assumes a lack of recombination when the results of meiosis cannot be observed and will always yield minimal recombinations. The next locus that is heterozygous for the parent in question will indicate when recombination occurs. The exact location of that recombination is unknown (the maximum likelihood approach makes use of recombination frequencies to estimate the location) but must be between that locus and the most recent upstream locus heterozygous for the parent under consideration. Propagating inheritance values from previous states constitutes another of Hapi's optimizations.

After building all states using the inheritance values from states at the previous locus, we can apply the optimization outlined above that removes equivalently labeled states. Each previous state maps to two states at the current locus that have inheritance vector values that are equivalent: the inheritance values corresponding to the homozygous parent are the identical, and those corresponding to the heterozygous parent are oppositely and therefore equivalently labeled. Because of this fact, the system can eliminate half the states that get built, retaining those with fewer recombinants.

In general, if there are n states at the previous locus, the system builds at most $2n$ states at the current locus, and it retains half of these, or at most n states. The system can retain fewer than n states if multiple states at the previous locus have the same inheritance vector values for the parent that is homozygous at the current locus. This commonly occurs when two loci that are fully informative for alternate parents appear in succession. In this case, the first locus can have at most two inheritance vector values corresponding to its heterozygous parent. At the second locus, there are again two inheritance vector values corresponding to the heterozygous parent, and, with two values for the homozygous parent at the previous locus, a total of four states result. These two successive loci are analogous to a single locus that is fully informative for both parents, and Hapi retains only one state at the second of these two loci. (Note: we discuss later that in the presence of ambiguous inheritance values—introduced at partly informative loci—the resulting number of states at this fully informative for one parent locus can be larger than n, the number of states at the previous locus.)

Figure 3 gives an example from real data of a fully informative locus with one state at the previous locus. To make the inheritance vectors more readable, we display them using ordered pairs for each child. It can be seen in this example that the system copies the inheritance vector values for homozygous parent from the previous locus state. As

		Parents		Children					# Rec
		p_0	p_1	c_0	c_1	c_2	c_3	c_4	
Prev	$\vec{v}$			$\langle 0,1\rangle$	$\langle 1,1\rangle$	$\langle 1,1\rangle$	$\langle 0,0\rangle$	$\langle 1,1\rangle$	0
State	hap	⟨a, g⟩	⟨a, a⟩	⟨g, a⟩	⟨a, a⟩	⟨a, a⟩	⟨a, a⟩	⟨a, a⟩	4
a	$\vec{v}$			$\langle \mathbf{1},1\rangle$	$\langle \mathbf{0},1\rangle$	$\langle \mathbf{0},1\rangle$	$\langle 0,0\rangle$	$\langle \mathbf{0},1\rangle$	
State	hap	⟨g, a⟩	⟨a, a⟩	⟨g, a⟩	⟨a, a⟩	⟨a, a⟩	⟨a, a⟩	⟨a, a⟩	1
b	$\vec{v}$			$\langle 0,1\rangle$	$\langle 1,1\rangle$	$\langle 1,1\rangle$	$\langle \mathbf{1},0\rangle$	$\langle 1,1\rangle$	

Figure 3: **Two States at a Fully Informative for One Parent Locus Built From the Previous State Shown** An example of a fully informative for one parent locus showing one state at the previous locus and the two states Hapi builds based on this previous state. This example is from the real dataset discussed in Results. The rows labeled $\vec{v}$ show the states' inheritance vectors and the rows labeled hap give haplotype assignments of the alleles. Hapi copies the inheritance vector values corresponding to the homozygous parent from the previous state to states a and b. Recombinations result from differing inheritance vector values from the previous state; these differences appear in bold and the states' total number of recombinations appear in the right-most column. Note that the heterozygous parent's inheritance vector values in the two states are exactly opposite each other and are therefore equivalently labeled.

well, the inheritance vector values corresponding to the heterozygous parent are opposite each other in the two states a and b. These two states therefore have equivalent inheritance vectors and, because state b has fewer recombinations, it is retained and state a is eliminated.

Partly Informative Loci Partly informative loci (and children with missing data; see below) are the primary reason the haplotyping problem for families is challenging. Both parents are heterozygous at this locus type, but they have the same genotype. As a result, heterozygous children have the same genotype as their parents and are *a priori* ambiguous as to which parent transmitted each allele: either parent could transmit either allele. For a given assignment of the parents' alleles to homologs, each heterozygous child has two possible inheritance vector values. For h heterozygous children, there are 2^h possible inheritance vectors for each of the four possible assignments of parents' alleles to homologs, and therefore $4 \cdot 2^h$ inheritance vectors/states. Hapi uses a technique, the last of its four optimizations, to avoid this exponential blowup in states.

Hapi builds states at partly informative loci using the states at the previous locus, mapping each previous state to four states corresponding to each assignment of parents' alleles to homologs. Note that multiple previous states can map to the same state, so the number of states usually does not quadruple. Homozygous children have only four possible inheritance vector values corresponding to the four assignments of parents' alleles, so they do not increase the number of necessary states.

Hapi assigns heterozygous children's inheritance vector values based on the inheritance values in the previous state. These children have two possible inheritance vector values for a given assignment of parents' alleles to homologs, and the two values are opposite each other. If the inheritance value in the previous state is equivalent to one of these two values, Hapi uses the value equivalent to the previous state in the state being built. The other inheritance value results in two crossovers for the child, one from each parent. Such an event is extremely unlikely, yet if it were to take place, downstream loci that are fully informative would reveal its occurrence. In that rare case, Hapi marks the partly informative locus as ambiguous during back tracing, since it is impossible to know whether these two recombinations took place at the earlier partly informative locus or at the later locus. (Maximum likelihood haplotyping determines the location of the recombinations based on recombination frequencies.)

In the case that the inheritance value in the previous state is not equal to one of the two ambiguous inheritance values, the previous inheritance value must differ from these two values in exactly one bit. For example, if the previous value is $\langle 0, 0\rangle$ and is not equal to either of the values at the current locus, they must be $\langle 0, 1\rangle$ and $\langle 1, 0\rangle$. (Otherwise one of the values would be $\langle 1, 1\rangle$, and the opposite value of $\langle 0, 0\rangle$ is equal to the previous value.) The differences across the values represent a recombination in one or the other parent. Which parent recombined is ambiguous at this locus and can only be determined at later loci.

Rather than creating separate states for these two inheritance values—which would yield an exponential number of states across multiple children—Hapi instead marks the child as having ambiguous inheritance. A child's inheritance being marked as ambiguous means that its inheritance vector value can be inverted without inducing additional recombinations—both possibilities result in the exactly one recombination. The choice of which of the two inheritance values to store in the state is arbitrary, and Hapi indicates that a child is ambiguous using another bit vector. For our

		p_0	p_1	c_0	c_1	c_2	c_3	c_4	# Rec
Prev	$\vec{v}$			$\langle 0, 1\rangle$	$\langle 1, 1\rangle?$	$\langle 1, 1\rangle$	$\langle 0, 0\rangle?$	$\langle 1, 1\rangle$	0
State a	hap $\vec{v}$	⟨a, g⟩	⟨a, a⟩	⟨g, a⟩ $\langle \mathbf{1}, 1\rangle$	⟨a, a⟩ $\langle \mathbf{0}, \underline{0}\rangle$	⟨a, a⟩ $\langle \mathbf{0}, 1\rangle$	⟨a, a⟩ $\langle 0, \underline{0}\rangle$	⟨a, a⟩ $\langle \mathbf{0}, 1\rangle$	4
State b	hap $\vec{v}$	⟨g, a⟩	⟨a, a⟩	⟨g, a⟩ $\langle 0, 1\rangle$	⟨a, a⟩ $\langle 1, \underline{1}\rangle$	⟨a, a⟩ $\langle 1, 1\rangle$	⟨a, a⟩ $\langle \mathbf{1}, \underline{1}\rangle$	⟨a, a⟩ $\langle 1, 1\rangle$	1

Figure 4: **Two States at a Fully Informative for One Parent Locus Built From the Previous State with Ambiguous Inheritance Shown** An example, modified from Figure 3 and not from real data, showing a state with ambiguous inheritance values (marked by ?) at the previous locus, and the two states Hapi builds based on it. For unambiguous children's inheritance vector values, the system copies the bits corresponding to the homozygous parent from the previous state. For ambiguous children, two opposite inheritance values are valid for the previous state, and the system uses the homozygous parent bit from the inheritance value that matches the heterozygous parent's bit in the state being built. Both of the two inheritance values are are necessarily represented, one in each of the resulting states. As the underlined values show, the inheritance values for the homozygous parent differ across the two outputs. As such, the states are not equivalent, and Hapi cannot eliminate either. Bold values indicate recombinations.

explanation here, we designate ambiguous values with the ? symbol. One can view an ambiguous inheritance value as a set of values, so $\langle 0, 0\rangle? = \langle 1, 1\rangle? = \{\langle 0, 0\rangle, \langle 1, 1\rangle\}$. For the earlier example with a previous inheritance value of $\langle 0, 0\rangle$, the resulting inheritance value would be $\langle 0, 1\rangle?$. The use of these ambiguous values effectively merges the exponential number of states that would otherwise result. Merging the states in this way suffices because (1) Hapi can later resolve which of the unambiguous inheritance vectors is optimal, and (2) the number of recombinations remains the same regardless of which fixed inheritance vector ultimately results.

If the previous inheritance value is itself ambiguous, the resulting value must also be ambiguous. In the case of recombination, the resulting value is unequal to the previous value, such as with $\langle 0, 0\rangle?$ and $\langle 0, 1\rangle?$.

Hapi resolves ambiguous inheritance values for a state during the back tracing process discussed earlier. During back tracing, when the system encounters a state that has one or more ambiguous inheritance values, it compares these values to the corresponding values at the next (already resolved) locus. If the unambiguous form of this value (i.e., that without the ? symbol) or its opposite is equal to the inheritance value at the next locus, the system assigns the equivalent value at the current locus. If neither is equal, recombinations occur on either side of this locus and the inheritance value is truly ambiguous. In this rare case, Hapi's output reports the child's haplotype at this locus as ambiguous.

Ambiguous Inheritance Values and Fully Informative for One Parent Loci Earlier we noted that the introduction of ambiguous inheritance values affects the way we handle fully informative for one parent loci. The issue surrounds the need at that locus type to propagate inheritance vector values for the homozygous parent from the previous locus. When states have unambiguous inheritance values, Hapi simply copies these values, and the result is that a previous state maps to two states with equivalent inheritance vectors. In this case, Hapi can eliminate of one of the states.

The situation is different when a previous state has children with ambiguous inheritance values. In this case, the corresponding two inheritance vectors that Hapi builds at this locus are not equivalent because the homozygous parent's inheritance values are opposite one another rather than equivalent.

Consider the example in Figure 4, which is modified from the example in Figure 3 to include ambiguous inheritance values. As usual, the inheritance vector values for the heterozygous parent are opposite in the two states. However, the ambiguous inheritance values correspond to two entirely opposite values, so the two resulting states do not have identical inheritance vector values for the homozygous parent (we underline these differing values in the figure). The consequence of this is that the two inheritance vectors are not equivalent, and the algorithm cannot eliminate one of the two states. Even so, because the heterozygous parent's inheritance values are still exactly opposite, if the next locus is fully informative for the other parent, Hapi can produce one state at that locus. Note that although these two inheritance vectors are not equivalent, other previous states may map to states that are equivalent, thereby enabling the elimination of some states. In other words, the number of states does not always double when there are ambiguous inheritance values.

Uninformative Loci Uninformative loci are homozygous for both parents and therefore give no information about recombinations. As such, Hapi does not build states at these loci, and when the system analyzes the succeeding locus, it uses the states from the most recent informative locus. The system does resolve the children's phase at these loci, deducing which is consistent with the parents' genotypes and assigning the children's alleles to homologs as needed.

Initial State To build the initial state from which to haplotype a given chromosome, Hapi uses either a fully informative for both parent locus or two loci that are fully informative for opposite parents. Beginning at the first locus on the chromosome, Hapi looks for those types of loci to define the initial state. It skips any partly informative loci it encounters before the initial state is defined. Later, after defining an initial state and haplotyping the remainder of the chromosome, Hapi resolves haplotypes at these early partly informative loci by performing reverse haplotyping starting from the locus that established the initial state. Uninformative loci do not depend on nor produce states, so the system phases these loci whenever it encounters them.

It is straight forward to define an initial state using a locus that is fully informative for both parents. This locus type has exactly four possible inheritance vectors, and deducing these vectors does not require any previous state. The choice of which of the four inheritance vectors to use in the initial state is arbitrary, since, as discussed earlier, the inheritance vectors are equivalent. Different choices result in the same haplotypes assigned to different homologs in the parents. That is, the parents' haplotypes are merely labeled differently, but otherwise the results are identical.

A fully informative for one parent locus defines half of an inheritance vector, giving information only for the bits that correspond to the heterozygous parent. This locus type has two possible values for that portion of the inheritance vector, and just as before, Hapi chooses between the two arbitrarily. The choice affects only the homolog labels for the heterozygous parent's haplotypes. The initial state thus becomes partially defined with values for the heterozygous parent. Later, when the system encounters a locus that is fully informative for the undefined parent (or a locus fully informative for both parents), it fills in the inheritance vector values for the undefined parent, and haplotyping proceeds forward normally from this point. The system handles any intervening loci that are fully informative for the already-defined parent in the normal way, while still leaving the homozygous parents' inheritance vector bits undefined.

Figure 5 (described in more detail below) gives an example of an initial state defined from two fully informative loci (numbered 8 and 12).

Missing Data Missing genotype data can result either because of quality control mechanisms associated with genotyping technologies or because of non-Mendelian errors (which can be removed using various software packages[18,19]). It is straight forward to handle loci that have children with missing data. Hapi simply copies the inheritance vector values corresponding to the child with missing data from the state(s) at the previous locus to the newly built states at the current locus. This approach assumes a lack of recombination for that child, which suffices for the same reason that assuming no recombination at loci where a parent is homozygous suffices. Because the inheritance vector values for that child will no longer be opposite each other between states, but will instead be identical, Hapi cannot eliminate states in the way it does when no data is missing. However, it is possible to eliminate states in most cases.

Consider a set of states that have equivalent inheritance vectors when the missing data children are ignored and with identical inheritance values for those missing data children (i.e., states built based on the same previous state). Let x be the number of children with missing data, and let r be the value of the smallest number of recombinations among this set of states. The states in this set are x or $2x$ recombinations away from having equivalent inheritance vectors, depending on whether the inheritance values are opposite each other for transmissions from one or both of the parents. (Viewed another way, if two states have the same assignment of alleles to homologs for one parent and opposite assignments for the other, the inheritance vectors are x recombinations away from being equivalent. If both parents have opposite allele assignments, the inheritance vectors must be entirely opposite each other and therefore $2x$ recombinations separate them: the missing data children's inheritance values are identical, not opposite.) Considering states that are separated by x recombinations, a state that has more than $r + x$ recombinations will always be less optimal than the minimal state and can therefore be removed. Even if all the missing data children later recombine relative to the state with r recombinations, which would produce an inheritance vector equivalent to the larger state, that minimal state would yield $r + x$ recombinations—i.e., fewer than that for the larger state.

Although this technique will not always eliminate the same number of states as if full data were available, it is quite effective. Our experimental results demonstrate this as Hapi very efficiently analyzes a real dataset that includes missing data (see Results). Often one state at a locus will have zero or one recombinations compared to another state that has all or all but one child recombining. In such a case, the technique just described can eliminate the state with more recombinations.

Hapi does not yet handle loci that are missing data for one or both parents. It can be modified to do so by building states corresponding to all possible parent genotypes consistent with the children's genotypes.

Locus		p_0	p_1	c_0	c_1	c_2	c_3	c_4	# Rec
8	*hap* $\vec{v}$	⟨a, a⟩	⟨a, c⟩	⟨a, a⟩ $\langle -, 0\rangle$	⟨a, c⟩ $\langle -, 1\rangle$	⟨a, c⟩ $\langle -, 1\rangle$	⟨a, c⟩ $\langle -, 1\rangle$	⟨a, a⟩ $\langle -, 0\rangle$	0
12	*hap* $\vec{v}$	⟨g, t⟩	⟨t, t⟩	⟨t, t⟩ $\langle 1, 0\rangle$	⟨t, t⟩ $\langle 1, 1\rangle$	⟨g, t⟩ $\langle 0, 1\rangle$	⟨t, t⟩ $\langle 1, 1\rangle$	⟨t, t⟩ $\langle 1, 0\rangle$	0

14	*hap* $\vec{v}$	⟨c, a⟩	⟨c, a⟩	⟨a, c⟩ $\langle 1, 0\rangle$	⟨a, a⟩ $\langle 1, 1\rangle$	⟨c, c⟩ $\langle 0, \mathbf{0}\rangle$	⟨a, c⟩ $\langle 1, \mathbf{0}\rangle$?	⟨a, c⟩ $\langle 1, 0\rangle$	2
16	*hap* $\vec{v}$	⟨a, a⟩	⟨g, a⟩	⟨a, g⟩ $\langle 1, 0\rangle$	⟨a, a⟩ $\langle 1, 1\rangle$	⟨a, g⟩ $\langle 0, 0\rangle$	⟨a, g⟩ $\langle 1, 0\rangle$	⟨a, a⟩ $\langle 1, \mathbf{1}\rangle$	3
17	*hap* $\vec{v}$	⟨t, c⟩	⟨c, c⟩	⟨c, c⟩ $\langle 1, 0\rangle$	⟨c, c⟩ $\langle 1, 1\rangle$	⟨t, c⟩ $\langle 0, 0\rangle$	⟨c, c⟩ $\langle 1, 0\rangle$	⟨t, c⟩ $\langle \mathbf{0}, 1\rangle$	4

14	*hap* $\vec{v}$	⟨c, a⟩	⟨a, c⟩	⟨a, c⟩ $\langle 1, \mathbf{1}\rangle$?	⟨a, a⟩ $\langle 1, \mathbf{0}\rangle$	⟨c, c⟩ $\langle 0, 1\rangle$	⟨a, c⟩ $\langle 1, 1\rangle$	⟨c, a⟩ $\langle \mathbf{0}, 0\rangle$?	3
16	*hap* $\vec{v}$	⟨a, a⟩	⟨a, g⟩	⟨a, g⟩ $\langle 1, 1\rangle$	⟨a, a⟩ $\langle 1, 0\rangle$	⟨a, g⟩ $\langle 0, 1\rangle$	⟨a, g⟩ $\langle 1, 1\rangle$	⟨a, a⟩ $\langle 0, 0\rangle$	3
17	*hap* $\vec{v}$	⟨t, c⟩	⟨c, c⟩	⟨c, c⟩ $\langle 1, 1\rangle$	⟨c, c⟩ $\langle 1, 0\rangle$	⟨t, c⟩ $\langle 0, 1\rangle$	⟨c, c⟩ $\langle 1, 1\rangle$	⟨t, c⟩ $\langle 0, 0\rangle$	3

Figure 5: **Example Establishing an Initial State and Showing Two State Paths** An example from the real dataset described in Results. The loci are from chromosome 3 and we number them sequentially in the order they occur physically. For simplicity and conciseness, we omit uninformative loci and one non-recombinant fully informative locus between 8 and 12. Bold inheritance vector values designate recombinations. Each state lists its total number of recombinations. Note that the state at locus 14 with minimum recombinations is ultimately not minimum-recombinant globally. See the example subsection for a detailed description of this figure.

Example We give a brief example illustrating some aspects of our algorithm in Figure 5. This example is from real data for one of the families in the Huntington's Disease Venezuela Collaborative Study[20] dataset discussed in Results. The initial locus 8 defines inheritance vector values for parent 1, the heterozygous parent, but leaves the values for parent 0 undefined (designated by $-$). When analyzing this example, Hapi produces a complete initial state at locus 12, where it deduces inheritance vector values for parent 0 and copies those for parent 1 from locus 8. (Note: this figure omits uninformative loci.)

Locus 14 is partly informative, and with only one state at the previous locus, it has only four states corresponding to the four possible assignments of the parents' alleles to homologs. The figure shows two of these four states, one on the left and one on the right. The two omitted states have four and five recombinations at locus 14 and still more at locus 16 and 17.

The left side state at locus 14 has two recombinations. It maps to two states at locus 16, one with a total of three recombinations and one with five; the figure shows the state with fewer recombinations. The two states at locus 16 each map to the same two states at locus 17, and we include the one with fewer recombinations in the figure.

The right side state for locus 14 has three recombinations. Although this is greater than the two local recombinations shown for the left side state, this state actually yields fewer recombinations globally. It maps to two states at locus 16, one of which produces no additional recombinations, and likewise that non-recombinant state produces zero recombinations at locus 17. This path of states therefore has only three recombinations, which is minimal across these loci.

Although the above analysis considered the downstream effects of each state at locus 14 separately, Hapi considers all states at successive loci at the same time and does not revisit each locus. The four states at locus 14 each map to two non-equivalent (because of ambiguous inheritance values) states at locus 16, for a total of eight states. Because locus 16 is fully informative for parent 1, the inheritance vector values for that parent are equivalently labeled in these states. Locus 17 is heterozygous for parent 0 and produces exactly four equivalently labeled states, and the state with the fewest recombinations must be globally minimal. This globally minimal state is on the right side of the figure.

Maximum Likelihood Haplotyping

We now formulate the problem of maximum likelihood haplotyping and show how it can be solved using the same techniques as those we employ for minimum-recombinant haplotyping.

Given a family with c children, genotyped loci numbered $0 \ldots L$ for each individual, and assigned inheritance vectors $\vec{v}_l$ for each locus l, let θ_l be the user-provided recombination frequency (i.e., probability of recombination) between locus l and $l-1$. Also let $r(l) = d(\vec{v}_{l-1}, \vec{v}_l)$ or the number of recombinations (hamming distance) between the inheritance vectors at loci $l-1$ and l. Then the following equation gives the probability of the given haplotype assignment:

$$\mathcal{P} = \prod_{l=1}^{L} \theta_l^{r(l)} \cdot (1 - \theta_l)^{(2c - r(l))} \tag{2}$$

Using log likelihoods, this can be written as:

$$\mathcal{L} = \sum_{l=1}^{L} \ln(\theta_l) \cdot r(l) + \ln(1 - \theta_l) \cdot [2c - r(l)] \tag{3}$$

This formulation of the maximum likelihood problem shows clearly the relationship of the maximum likelihood problem to the minimum-recombinant one. If all loci have the same recombination frequency $\theta < 0.5$, then maximum likelihood solution is the same as the minimum-recombinant one since $\ln(\theta) < \ln(1 - \theta)$ across all loci, so decreased $r(l)$ values increase the overall likelihood. However, when the recombination frequencies differ across loci, this may not always be the case: more recombinations at one locus may be more likely than fewer recombinations at another.

A dynamic programming equation computing maximum likelihood haplotypes can be written as the following, where l is a locus, and $\vec{v}, \vec{w}$ are inheritance vectors:

$$P(l, \vec{v}) = \max_{\vec{w}}\{P(l-1, \vec{w}) \cdot \theta_l^{d(\vec{w},\vec{v})} \cdot (1 - \theta_l)^{(2c - d(\vec{w},\vec{v}))}\} \tag{4}$$

Using log likelihoods, the dynamic programming formulation becomes:

$$L(l, \vec{v}) = \max_{\vec{w}}\{L(l-1, \vec{w}) + \ln(\theta_l) \cdot d(\vec{w}, \vec{v}) + \ln(1 - \theta_l) \cdot [2c - d(\vec{w}, \vec{v})]\} \tag{5}$$

Immediate application of the above formula is problematic because we cannot completely ignore uninformative loci: they have non-zero recombination frequencies that affect the overall probability of a solution. Without some novel insight, it is necessary to model most or all of the $2^c/4$ non-equivalent inheritance vectors at uninformative loci.

We resolve this issue by computing modified recombination frequencies for informative loci that include the recombination frequencies for upstream uninformative loci. Specifically, the modified recombination frequency includes the recombination frequencies for all uninformative loci that occur between a given informative locus and nearest upstream informative locus. Note that loci that are informative for only one parent p are uninformative for the other parent q, and such a locus is therefore one of the uninformative loci accounted for at the next locus informative for parent q. (This applies to fully informative for one parent loci; partly informative loci are informative for both parents since we observe the outcome of meiosis for both parents.) It is therefore necessary to compute separate recombination frequencies for each parent, and we denote the modified recombination frequency for a parent p at locus l as ϕ_{lp} and the frequency of non-recombination (expressed above as $(1 - \theta_l)$) as ψ_{lp}.

To calculate ϕ_{lp} and ψ_{lp} for a locus l informative for parent p, let l_0 be the nearest upstream locus informative for parent p and let $l_1, \ldots, l_{n-1}$ be the loci uninformative for parent p that appear between l_0 and l. Let $l_n = l$ and let l^* be the locus with the highest recombination frequency, i.e., find $l^* \in \{l_1, \ldots, l_n\}$ such that $\theta_{l^*} = max_{i=1}^{n} \theta_{l_i}$. Then:

$$\phi_{lp} = \theta_{l^*} \cdot \prod_{i=1, l_i \neq l^*}^{n} (1 - \theta_{l_i}) \tag{6}$$

Thus, the probability of recombination between locus l_0 and $l_n = l$ is equal to the maximum recombination frequency θ_{l^*} multiplied by the probably of not recombining anywhere else. Note that θ_{l^*} is the probability of recombining between locus l^* and $l^* - 1$, and either or both of these loci can be uninformative. Hapi stores the location of l^* so that the final haplotype solution includes any recombinations in the most likely position.

A consequence of this formula is that at most one recombination can occur between any two informative loci on a given homolog. Thus, within a region of uninformative loci, we do not model the possibility of intervening gene conversions or double recombinations. Not modeling such events make sense because it is impossible to observe or verify them. Furthermore, haplotypes that include additional recombinations or gene conversions not directly implied by the data are less likely than those without these events since $\theta_l < 0.5$ means recombination is less likely than non-recombination. Therefore, even if we were to model such events, they would not ultimately appear in the haplotype solution, so we lose nothing by not modeling them.

The probability of not recombining between locus l_0 and $l_n = l$ is the product of non-recombination across each of the locus intervals:

$$\psi_{lp} = \prod_{i=1}^{n}(1 - \theta_{l_i}) \tag{7}$$

To increase numerical stability and efficiency, Hapi uses the log likelihood formulation of this dynamic programming problem. This formula substitutes multiplication for exponentiation and uses the values $\ln(\phi_{lp})$ and $\ln(\psi_{lp})$ which requires summation instead of multiplication to calculate. The dynamic programming equation for calculating maximum log likelihood haplotypes at a locus l informative for parent p is then given by the following:

$$L(l, \vec{v}) = \max_{\vec{w}}\{L(l-1, \vec{w}) + \ln(\phi_{lp}) \cdot d(\vec{w}, \vec{v}) + \ln(\psi_{lp}) \cdot [c - d(\vec{w}, \vec{v})]\} \tag{8}$$

For a locus that is informative for both parents, including partly informative loci, the system separates the recombinations for each parent in $d(\vec{w}, \vec{v})$, and includes terms for both ϕ_{lp_0} and ϕ_{lp_1} as well as the corresponding ψ terms.

This formulation solves the problem of needing to track states at uninformative loci, but leads to another issue. Our minimum-recombinant formulation of ambiguous inheritance values exploits the fact that the number of recombinations was the same regardless of which of the two ambiguous inheritance values gets used, i.e., regardless of which parent recombines. In this maximum likelihood framework, the equivalent property would be that the recombination frequency is the same in both parents. This is the case for the original θ_l recombination frequencies, which do not vary across genders (though it could be fruitful to use differing recombination frequencies since recombination frequencies and locations do vary quite significantly across genders). However, the modified recombination values formulated above will typically span different sets of loci for each parent and will therefore differ between the parents.

To handle this issue, we temporarily store the probabilities for the ambiguous children in the state rather than including those values in the overall state probability. In order to eliminate very unlikely states and paths, we do track the total minimum and maximum ambiguous probabilities across all the ambiguous children. When multiple previous states map to the same state, Hapi does not retain any paths from previous states whose maximum probability, including probabilities for ambiguous children, is less than the minimum probability for another state. Eventually the system performs back tracing and resolves the ambiguous children's probability values. Assigning the final haplotypes requires both back tracing to compute the probabilities along each of the paths, and then, once the system determines the path with the maximum likelihood, it traces forward and assigns the haplotype values at each locus.

Although it is unlikely to occur in practice, there are scenarios in which recombining on both homologs upstream or downstream of some ambiguous recombination can have increased likelihood. That is, three recombinations can be more likely than one recombination since recombining on both homologs changes which parent transmits the otherwise ambiguous recombination. If the recombination probability for one parent is significantly lower than the other parent, additional recombinations that change which parent recombines can increase the overall likelihood. Hapi checks whether such additional recombinations increase the likelihood of a path in order to find the assignment with maximum likelihood.

Our formulation above has the same recombination frequency for all children at a given locus. In our implementation, we track recombination frequencies for each child individually. This is necessary because of missing data: if a child is missing data at a given locus, that locus is effectively uninformative for that child even though it may be informative for the other children. As such, which locus is informative will differ across children, and it is necessary track per-child recombination frequencies.

Results

We have evaluated Hapi's performance compared to Merlin[9] and Superlink,[17] programs in current use for family-based haplotype assignment. Like most algorithms for computing maximum likelihood haplotypes, both Merlin and Superlink have exponential complexity in general. However, each contain many optimizations and are the most suitable programs for comparison to Hapi. Merlin is presently the state-of-the-art haplotyping algorithm, outperforming both GENEHUNTER and Allegro.[9] Because of Merlin's superior performance, we omitted GENEHUNTER and Allegro from our comparison.

We ran Hapi, Merlin, and Superlink on a dataset of nuclear families derived from a pedigree dataset from the Huntington's Disease Venezuela Collaborative Study.[20] This Venezuelan pedigree has 757 individuals and 458 families. Neither Merlin nor Superlink can successfully haplotype such a large pedigree. Hapi can currently only analyze nuclear families where both parents have marker data, so the pedigree was broken up into such families. Though Superlink is applicable to general pedigrees, we found it was unable to haplotype nuclear families with more than eight

Venezuelan Families	2.80 GHz Pentium 4					1.40 GHz Pentium M	
	Hapi	Merlin	Superlink	Hapi Speedup vs.		Hapi	PedPhase 2.0
				Merlin	Superlink		
All	4.588s	2371s	5005s*	517x	1091x	4.732s	$>$ 21600s (6h)†
$\leq$ 3 Children	3.214s	17.75s	70.74s	5.52x	22.0x	3.451s	$>$ 21600s (6h)†

Figure 6: **Timing Results for Hapi and Other Programs for Haplotyping Families** Runtimes for maximum likelihood haplotyping in Hapi, Merlin, and Superlink on nuclear families from the Huntington's Disease Venezuela Collaborative Study.[20] Times are listed for haplotyping all nuclear families and for those with three or fewer children. *Superlink failed to haplotype the family with 11 children; we therefore used only 8 of the children from the 11 child family to time it. Times are averages from running Hapi eight times and Merlin and Superlink three times each. Runtimes also on a different machine for minimum-recombinant haplotyping in Hapi (averaged from eight runs) and PedPhase †for chromosome 1 only.

children (see below). The choice to break up such a large pedigree into smaller sets of related individuals appears to be necessary regardless of which haplotyping tool is used.

The nuclear family dataset contains 103 nuclear families where both parents have data. These families have a total of 438 individuals. Note that because we analyzed the families separately, we double counted individuals that appear in more than one family (e.g., as a parent in one and a child in another, or as a parent in more than one family).

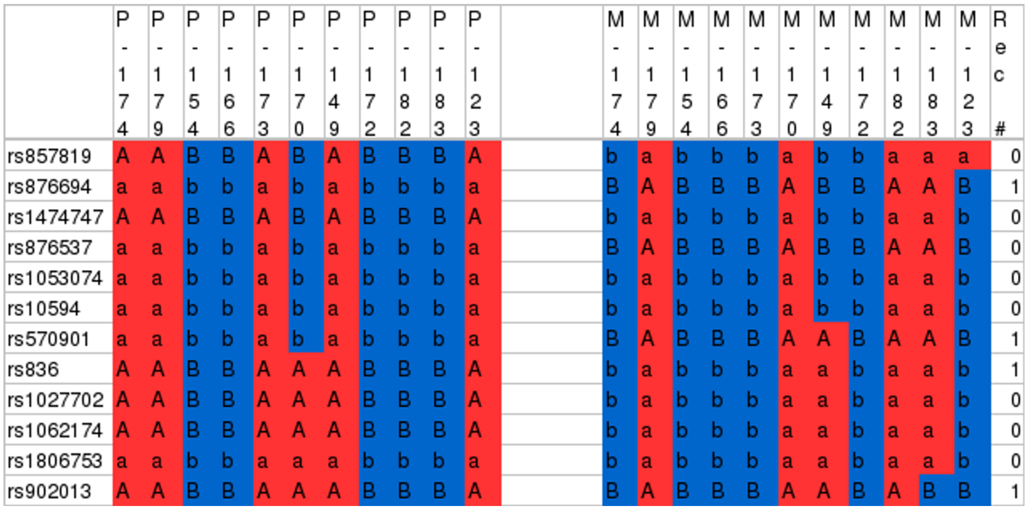
These families range in size from 1 to 11 children, with an average of 2.23 children per family. There are 86 families with 3 or fewer children (308 total individuals), with an average of 1.56 children for that subset of families. Using the Illumina linkage IV_v3 SNP panel, genotypes at 5,456 SNPs covering the whole genome were obtained for each individual in the dataset.[20] The numbers of SNPs per chromosome are roughly proportional to the chromosome's size and range from 102 on chromosome 21 to 468 on chromosome 2. Prior to analysis, the PEDSTATS[18] and PedCheck[19] programs were used to remove genotypes exhibiting non-Mendelian errors.

Figure 6 shows timing results from our experiments. The times on the left side of the figure are for performing maximum likelihood haplotyping in Hapi, Merlin, and Superlink on a 2.80GHz Pentium 4 machine with 3GB of RAM. We used Hapi to infer maximum likelihood rather than minimum-recombinant haplotypes in this set of experiments because Merlin and Superlink handle that problem, and because that form of haplotyping is slower in Hapi. Neither Hapi or Merlin exceeded the available memory in these experiments. When run on the chromosome 1 for all families in the dataset, Superlink ran for over 6 hours and reported that 0% of the haplotyping was complete at that time. We empirically discovered that Superlink would either run without reporting any progress or would crash when we used it to haplotype a family with more than eight children. The times for Superlink therefore reflect its haplotyping a modified set of families, with three of the children removed from the original 11 child family. Superlink did not exceed available memory when analyzing this modified dataset. The figure shows times for haplotyping all families in the dataset (modified for Superlink), as well as the subset of families with three or fewer children. Because of the fixed overhead involved in printing the haplotypes in Hapi and Merlin ($>$ 2 seconds in each program), we report the times only for reading in the dataset and performing the haplotyping, but not printing the results. Source code is not publicly available for Superlink, so we could not modify it to avoid printing haplotypes, but such a change is unlikely to dramatically affect its runtime.

As the figure shows, Hapi is substantially faster than Merlin, running 517 times faster for the entire dataset and 5.52 times faster for the subset of families with three or fewer children. Hapi compares even more favorably against Superlink's haplotyping of the modified dataset, running 1091 times faster for the large dataset and 22.0 times faster for the three children subset.

Hapi's speedup for the entire dataset demonstrates experimentally the vast difference between the theoretical complexity of these algorithms. At the same time, the more modest gains for the families with three or fewer children is unsurprising. The other algorithms scale exponentially in the number of non-founders or, in the case of nuclear families, in the number of children in the family being analyzed. When that number is very small, an exponential algorithm will not differ as significantly from one that has polynomial runtime in practice (see Appendix for complexity analysis). Our algorithm is still significantly faster than both Merlin and Superlink even in this case that is less taxing to an exponential algorithm.

Besides these maximum likelihood systems, we compared Hapi's minimum-recombinant haplotyping to PedPhase 2.0, which uses an Integer Linear Programming algorithm to calculate minimum-recombinant haplotypes for pedigrees.[14] PedPhase 2.0 runs only in Windows, and we used a 1.40GHz Pentium M laptop with 1.24 GB of RAM to



	P-174	P-179	P-154	P-166	P-173	P-170	P-149	P-172	P-182	P-183	P-123		M-174	M-179	M-154	M-166	M-173	M-170	M-149	M-172	M-182	M-183	M-123	Rec#
rs857819	A	A	B	B	A	B	A	B	B	B	A		b	a	b	b	b	a	b	b	a	a	a	0
rs876694	a	a	b	b	a	b	a	b	b	b	a		B	A	B	B	B	A	B	B	A	A	B	1
rs1474747	A	A	B	B	A	B	A	B	B	B	A		b	a	b	b	b	a	b	b	a	a	b	0
rs876537	a	a	b	b	a	b	a	b	b	b	a		B	A	B	B	B	A	B	B	A	A	B	0
rs1053074	a	a	b	b	a	b	a	b	b	b	a		b	a	b	b	b	a	b	b	a	a	b	0
rs10594	a	a	b	b	a	b	a	b	b	b	a		b	a	b	b	b	a	b	b	a	a	b	0
rs570901	a	a	b	b	a	b	a	b	b	b	a		B	A	B	B	B	A	A	B	A	A	B	1
rs836	A	A	B	B	A	A	A	B	B	B	A		b	a	b	b	b	a	a	b	a	a	b	1
rs1027702	A	A	B	B	A	A	A	B	B	B	A		b	a	b	b	b	a	a	b	a	a	b	0
rs1062174	A	A	B	B	A	A	A	B	B	B	A		b	a	b	b	b	a	a	b	a	a	b	0
rs1806753	a	a	b	b	a	a	a	b	b	b	a		b	a	b	b	b	a	a	b	a	a	b	0
rs902013	A	A	B	B	A	A	A	B	B	B	A		B	A	B	B	B	A	A	B	A	B	B	1

Figure 7: **Inheritance Vector Output from Hapi Imported into a Spreadsheet for Visualization** Output from Hapi showing the inherited homologs on chromosome 1 for a family with 11 children from the Huntington's Disease Venezuela Collaborative Study.[20] Hapi produces CSV format output, which we imported into a spreadsheet. To color the cells, we used conditional formatting based on the homolog value transmitted. The output of inheritance vector values uses letters A and B. Lower-case letters indicate the transmitting parent is homozygous and the presence of recombination unknown. Each column is labeled with the child's numerical id with either a 'P' or an 'M' preceding it to indicate either paternal or maternal-derived homologs. The left most column gives the SNP rs numbers, and the right most column lists the number of recombinations across all children at the given locus.

compare runtimes of these two systems. The right side of Figure 6 gives timing results on this machine for Hapi and PedPhase. We ran PedPhase on the entire dataset and on the families with three or fewer children. In both cases, PedPhase did not exceed available memory, and ran for over 6 hours without haplotyping even chromosome 1. Because 464 of the 5,456 total SNPs reside on chromosome 1, we estimate that the total runtime for PedPhase on this dataset would be at least 70 hours. In contrast, Hapi completes haplotyping the entire dataset in 4.732 seconds (in Linux) on this machine.

Hapi produces output in text or CSV format, suitable for import into a spreadsheet. It can output either the actual haplotypes with allele values or the children's inheritance vector values. The latter is useful for inspecting the results of meioses, including recombination patterns. Figure 7 shows example inheritance vector output from Hapi for a family with 11 children, imported into a spreadsheet. This output uses letter symbols rather than bit values with lower case letters indicating that a locus is uninformative. To help identify recombinations sites, we use the spreadsheet program's conditional formatting feature to color the cells based on which homolog the child received. The ability to import data into spreadsheets makes Hapi a very effective tool for geneticists. While Merlin and Superlink can output similar information, each uses its own text format that is less intuitive and less convenient for most geneticists.

Discussion

Assignment of haplotypes is an important element in a number of significant areas of genetic analysis, including locating genes involved in human disease, analyzing the products of meiosis to locate recombination hotspots and gene conversions, and studying population dynamics and history for humans and other species. Because of their importance, researchers have developed computational algorithms for inferring haplotypes from genotypes. The most effective approach to this problem is to use data for individuals whose family relationships are known. Inferring minimum-recombinant haplotypes for the individuals in a pedigree is known to be NP-hard in general.[21,13] Problems classified as NP-hard are not known to have a polynomial time (i.e., efficient) solution, and are therefore thought to be computationally intractable. Existing algorithms computing either maximum likelihood (based on recombination rates) or minimum-recombinant solutions for pedigrees consequently have exponential complexity.

Hapi is an efficient algorithm for inferring both minimum-recombinant and maximum likelihood haplotypes for nuclear families. Hapi runs in polynomial time in practice (see Appendix for algorithm complexity details), and our experimental data demonstrate the effectiveness of our approach. When haplotyping a large dataset of nuclear families, Hapi outperforms the state-of-the-art system Merlin with a speedup of between 5.5–517 times and is between 22–1091 times faster than Superlink.

Four key insights (implemented as optimizations) underly Hapi's efficiency. The first is that it is only necessary to build states that are consistent with the Mendelian laws of inheritance applied to the individuals' observed genotypes. Second, when a parent is homozygous, it is unnecessary to build states that represent recombinations, and our maximum likelihood approach uses a novel calculation of recombination rates to make this possible. Third, many of the possible states at a locus have equivalent inheritance vectors, and among these, only the state with the fewest recombinations or the maximum likelihood needs to be retained while the others can be eliminated. Fourth, we have formulated a novel representation of inheritance vectors that includes ambiguous inheritance values. This representation reduces the need to represent an exponential number of states at loci of the type we call partly informative.

As time passes and technology improves, genotype datasets will continue to grow in size, both numbers of individuals and numbers of loci assayed. As such, faster tools for haplotype analysis will be essential. Existing algorithms for haplotyping related individuals have hard limits on the size of families they can analyze because of their exponential complexity. These algorithms are consequently ineffective for datasets with thousands of families or for families with large numbers of children. Hapi provides a solution that is able to meet many of these future challenges.

Acknowledgements

We thank Angelina Lee, Adam Kieżun, and Bonnie Kirkpatrick for helpful discussions about our algorithm, and Junne Kamihara, Karen Zee, Shaun Mahony, and Sylvia Pearce for comments on this paper. We also thank Josh Taylor and Bill Thies for pointers to information helpful in writing this paper. This work is supported by NIH grants 5-T90-DK070069 and 5-P01-NS055923, and an NSF Graduate Research Fellowship.

Appendix

As we discuss in Results, Hapi is extremely efficient, running orders of magnitude faster than existing algorithms. These experimental results demonstrate that Hapi is very efficient in practice. If L is the number of marker loci, s is the maximum number of states produced at any locus, and c is the number of children, Hapi's complexity is $O(m \cdot s \cdot c)$. This is so because the algorithm visits each of the L loci at most two or, for maximum likelihood haplotyping, three times (additional visits during back tracing/forward tracing). At each locus, Hapi builds $O(s)$ states for each, and each state has size $O(c)$. While Hapi is efficient in practice, there are two rare corner cases where $s = O(2^c)$. That is, the number of states can be exponential in the number of children in the family being analyzed.

Because of the nature of the requirements for these corner cases to occur, they are extremely unlikely to happen in practice. As well, if a series of loci do exhibit one of these cases, the number of states become exponential only transiently because a later locus that is fully informative for both parents and with data for all the children (or two successive loci fully informative for opposite parents) will produce a single state. One necessary condition for this state blowup is therefore that the series of loci not contain such a locus (or loci) that produce a single state.

The first exponential case arises because of missing data. Consider a fully informative for one parent locus where the previous locus has n states. If one or more children have missing data at this locus, the number of states for this locus, which would otherwise be n (absent ambiguous inheritance values), can instead become $2n$. In order for this increase in states to occur, a large proportion of the children must either recombine or have missing data; otherwise the optimization described in the main text would apply. Such a scenario is unlikely in practice since recombination is rare, but one can construct such a pathological input. Note that if a child that was missing data at some previous locus has data at the current locus, states that were added at a previous locus can effectively merge.

The above properties indicate the possibility, however remote, for Hapi to produce exponential states from loci with missing data. Consider a series of loci where each successive locus has missing data for the same set of children at the previous locus as well as missing data for one more child. (The first locus with data for all children, the second missing data for child c_0, the third missing data for child c_0 and c_1, etc.) This scenario could lead to an exponential number of states, but only if the previously mentioned optimization does not apply. A more complicated scenario in which at least half of the children have missing data at the starting locus would defeat the optimization (since the value of x mentioned the text would be large), but would still be very unlikely.

The second way in which an exponential number of states may occur comes through partly informative loci. Ignoring additional states that may arise because of missing data, the first partly informative locus will have 4 states. Without any recombination, the maximum number of states that may arise at a partly informative locus is 12. This maximum of 12 states occurs because lack of recombination constrains children's inheritance values to be fixed relative to each other, and for a particular assignment of parent's alleles to homologs, there are three possible inheritance values for heterozygous children (two non-ambiguous and one ambiguous.)

When recombination occurs in a child or children between two partly informative loci, the number of states at the second locus increases *additively*, with two additional states for each assignment of parent's alleles, (for a total of eight added states). The recombination changes the constraints on the possible children's inheritance values and effectively breaks the children into two classes. The children in one class exhibits recombination relative to the other, but until the remainder of the algorithm completes, which of these two classes has fewer recombinations is unknown, so the system must track additional states (or both classes). In general, this can lead to at most $4(2c+1)$ states, a polynomial number, so the number of states cannot become exponential across a series of partly informative loci.

The number of states can grow to be exponential through interactions between partly informative and fully informative for one parent loci. Whenever a locus has states with ambiguous inheritance values and occurs immediately before a fully informative for one parent locus, the number of states may double, as described in the text. The scenario in which an exponential number of states can occur is when there are a series of alternating partly informative and fully informative for one parent loci. At each partly informative locus, one of the children must recombine and become heterozygous, yielding an ambiguous inheritance value, and that child must remain heterozygous at all successive partly informative loci. The recombination results in an addition $4 \cdot 2$ states at each partly informative locus, followed by a doubling in the number of states at the fully informative for one parent locus. The number of states doubles c times for an exponential blowup.

The above scenario is extremely unlikely. It requires recombinations to occur in all the children, all at partly informative loci, and without encountering a locus or loci that produce a single output state. Moreover, after recombining, the children must be heterozygous across all partly informative loci. This is unlikely because homozygous and heterozygous genotypes are equally probable for children at such loci. Although this scenario could occur for a small number of children (say two or three), the likelihood decreases as the number of children increases. Thus, when an exponential blowup would be most problematic (for large c), it becomes even less likely.

Web Resources

Hapi will be made freely available for non-profit use at the time of publication at http://hapi.csail.mit.edu/.

References

1. Niu, T. (2004). Algorithms for inferring haplotypes. Genet. Epidemiol. *27*, 334–347.

2. Marchini, J., Cutler, D., Patterson, N., Stephens, M., Eskin, E., Halperin, E., Lin, S., Qin, Z. S., Munro, H. M., Abecasis, G. R., et al. (2006). A comparison of phasing algorithms for trios and unrelated individuals. Am. J. Hum. Genet. *78*, 437–450.

3. Stephens, M. and Donnelly, P. (2003). A comparison of bayesian methods for haplotype reconstruction from population genotype data. Am. J. Hum. Genet. *73*, 1162–1169.

4. Niu, T., Qin, Z. S., Xu, X., and Liu, J. S. (2002). Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. Am. J. Hum. Genet. *70*, 157–169.

5. Lin, S., Chakravarti, A., and Cutler, D. J. (2004). Haplotype and missing data inference in nuclear families. Genome Res. *14*, 1624–1632.

6. Lin, S., Cutler, D. J., Zwick, M. E., and Chakravarti, A. (2002). Haplotype inference in random population samples. Am. J. Hum. Genet. *71*, 1129–1137.

7. The International HapMap Consortium. (2007). A second generation human haplotype map of over 3.1 million snps. Nature *449*, 851–861.

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

9. Abecasis, G. R., Cherny, S. S., Cookson, W. O., and Cardon, L. R. (2002). Merlin—rapid analysis of dense genetic maps using sparse gene flow trees. Nat. Genet. *30*, 97–101.

10. Markianos, K., Daly, M. J., and Kruglyak, L. (2001). Efficient multipoint linkage analysis through reduction of inheritance space. Am. J. Hum. Genet. *68*, 963–977.

11. Kruglyak, L., Daly, M. J., Reeve-Daly, M. P., and Lander, E. S. (1996). Parametric and nonparametric linkage analysis: A unified multipoint approach. Am. J. Hum. Genet. *58*, 1347–1363.

12. Gudbjartsson, D. F., Jonasson, K., Frigge, M. L., and Kong, A. (2000). Allegro, a new computer program for multipoint linkage analysis. Nat. Genet. *25*, 12–13.

13. Li, J. and Jiang, T. (2003). Efficient rule-based haplotyping algorithm for pedigree data. In RECOMB pp. 197–206.

14. Li, J. and Jiang, T. (2004). An exact solution for finding minimum recombinant haplotype configurations on pedigrees with missing data by integer linear programming. In RECOMB pp. 101–110.

15. Kong, A., Gudbjartsson, D. F., Sainz, J., Jonsdottir, G. M., Gudjonsson, S. A., Richardsson, B., Sigurdardottir, S., Barnard, J., Hallbeck, B., Masson, G., et al. (2002). A high-resolution recombination map of the human genome. Nat. Genet. *31*, 241–247.

16. Coop, G., Wen, X., Ober, C., Pritchard, J. K., and Przeworski, M. (2008). High-resolution mapping of crossovers reveals extensive variation in fine-scale recombination patterns among humans. Science *319*, 1395–1398.

17. Fishelson, M., Dovgolevsky, N., and Geiger, D. (2005). Maximum likelihood haplotyping for general pedigrees. Hum. Hered. *59*, 41–60.

18. Wigginton, J. E. and Abecasis, G. R. (2005). Pedstats: descriptive statistics, graphics and quality assessment for gene mapping data. Bioinformatics *21*, 3445–3447.

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

20. Gayán, J., Brocklebank, D., Andresen, J. M., Alkorta-Aranburu, G., Group, T. U.-V. C. R., Cader, M. Z., Roberts, S. A., Cherny, S. S., Wexler, N. S., Cardon, L. R., et al. (2008). Genomewide linkage scan reveals novel loci modifying age of onset of Huntington's disease in the Venezuelan HD kindreds. Genet. Epidemiol. *32*, 445–453.

21. Doi, K., Li, J., and Jiang, T. (2003). Minimum recombinant haplotype configuration on tree pedigrees. In WABI pp. 339–353.