BIOINFORMATICS

Vol. 23 ISMB/ECCB 2007, pages i401–i407
doi:10.1093/bioinformatics/btm220

Inferring missing genotypes in large SNP panels using fast nearest-neighbor searches over sliding windows

Adam Roberts[1], Leonard McMillan[1,*], Wei Wang[1], Joel Parker[2], Ivan Rusyn[3] and David Threadgill[4]

[1]Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599, [2]Constella Group, Durham, NC 27713, [3]Department of Environmental Sciences and Engineering, University of North Carolina, Chapel Hill, NC 27599 and [4]Department of Genetics, University of North Carolina, Chapel Hill, NC 27599, USA

ABSTRACT

Motivation: Typical high-throughput genotyping techniques produce numerous missing calls that confound subsequent analyses, such as disease association studies. Common remedies for this problem include removing affected markers and/or samples or, otherwise, imputing the missing data. On small marker sets imputation is frequently based on a vote of the K-nearest-neighbor (KNN) haplotypes, but this technique is neither practical nor justifiable for large datasets.

Results: We describe a data structure that supports efficient KNN queries over arbitrarily sized, sliding haplotype windows, and evaluate its use for genotype imputation. The performance of our method enables exhaustive exploration over all window sizes and known sites in large (150K, 8.3M) SNP panels. We also compare the accuracy and performance of our methods with competing imputation approaches.

Availability: A free open source software package, NPUTE, is available at http://compgen.unc.edu/software, for non-commercial uses.

Contact: mcmillan@cs.unc.edu

1 INTRODUCTION

Panels of single nucleotide polymorphisms (SNPs) are important tools for identifying gene-disease associations. Analyzing complex traits, like most diseases, involves genome-wide mapping of phenotypic traits, which requires many samples along with large and dense marker sets. Such datasets are apt to contain a significant number of missing genotypes. In practice, there are four options for dealing with this problem: (1) repeating the genotyping in missing regions, (2) modifying analysis tools to tolerate missing data, (3) removing SNPs and/or samples with missing data or (4) inferring the missing data. In this report, our focus is on inferring missing genotypes, otherwise known as imputation. Of all these options, imputation attempts to leverage existing tools and data resources while avoiding the labor and cost of additional genotyping, but at the risk of introducing potential biases or errors.

*To whom correspondence should be addressed.

Many genetic linkage and association analysis tools require complete data, yet typical genotyping technologies exhibit missed call rates ranging from 5 to 20% (Huentelman *et al*. 2005). Therefore, methodologies for dealing with unresolved genotypes are an imminent and practical concern. Most tools that handle missing genotypes implicitly perform some sort of *in silico* inference as either a preprocess or a side effect of the analysis. A common method for handling missing genotype calls is to remove either the offending SNP or an entire sample (Kang *et al*. 2004). Such thinning of datasets can lead to significant reductions in the detection power and mapping resolution since it sacrifices good data, which often affects the majority of SNPs.

An alternative to throwing away information is to infer missing genotypes by considering the similarities in the haplotype structures between samples due to sequences at the same genomic locations that are identical by descent. Searching for haplotype similarities among unrelated samples is justifiable on the basis of the coalescent theory. However, the extent of these identity regions is limited due to historical recombinations and mutations. One way of balancing these coalescent and divergent forces is to consider haplotype similarities within variably sized sliding windows.

The problem of inferring missing genetic markers is often confounded by the question of diploid phasing. Phasing alone is a difficult computational challenge, which is further complicated by the introduction of missing genotypes. In fact, most systematic approaches to phasing in the presence of missing markers have proven to be computationally intractable for large datasets. This leads to a class of algorithms that offer only approximate solutions with no guarantees of optimality. Coupling imputation with phasing adds an unnecessary complication to animal studies where inbred (isogenic) lines are commonly used (Threadgill *et al*. 2002). Furthermore, even highly accurate phasing methods, which consider pedigrees (trios), still result in missing calls. For example, a homozygous minority paternal genotype crossed with a missing maternal genotype that yields a heterozygous child implies a transmitted majority allele from the maternal haplotype paired with an unknown (missing value).

A variety of techniques have been applied to the problem of imputing missing genotypes. A common statistical approach is to infer missing genotypes from haplotype frequencies of population samples using either expectation maximization (EM) (Qin *et al.*, 2002) or Bayesian methods (Lin *et al.*, 2002; Niu *et al.*, 2002; Stephens *et al.*, 2001). More recent approaches incorporate models of recombination by partitioning markers into haplotype blocks based on entropy measures (Su *et al.*, 2005) or by inferring a mosaic of haplotype clusters (Scheet and Stephens, 2006). Tree-based imputation methods have also been developed that infer missing genotypes on the basis of perfect phylogeny rather than haplotype structure (Dai *et al.* 2006, Eskin *et al.*, 2003). In comparison to these previous methods, ours is refreshingly simple. It relies on uncomplicated measures of pairwise haplotype dissimilarity. The advantage of its simplicity is that its speed, which is sufficient to enable exhaustive searches over multiple parameters. The result of this combination is a fast imputation approach with competitive accuracy, thus providing a baseline adhering to the age-old principle of Occam's razor—with all else being equal, simpler solutions should be preferred over more complicated alternatives.

2 METHODS

There are three key elements of our imputation approach. The first is a data structure supporting fast K-nearest neighbor (KNN) searches over sliding windows of arbitrary sizes in constant time (independent of window size). Second, the speed afforded by this data structure enables us to perform exhaustive searches over all reasonable window sizes for an optimal size, thus implicitly finding haplotype blocks. Finally, we are able to estimate our imputation accuracy by exhaustively inferring every known SNP value as if it were missing from the dataset. In other words, we do not rely on sampling.

In this section we describe our SNP representation, supporting data structures, and the imputation method itself.

2.1 Data structures and data preparation

2.1.1 SNP vectors Any method involving genetic markers requires a simple and efficient representation of the markers themselves. Assuming biallelic SNPs, we choose to represent the markers as a ternary (three-valued) vector; extensions to handle multi-allelic loci are straightforward. The samples are first sorted alphabetically to provide an ordering to the alleles. We represent the majority allele of the SNP as a '0', the minority allele as a '1' and an unknown value as a '?'. Note that due to the presence of unknowns, the allele represented by the '0' may not in fact be a true majority, but we enforce that the number of '0's in the vector is greater than or equal to the number of '1's. In SNPs with equal allele frequencies the '0' is assigned to the first called allele value in sample order. We store these vectors in an array in chromosome order with separate fields for the chromosome, sequence location, the majority and minority nucleotides and the marker ID. In this way, all significant information is preserved.

2.1.2 Pairwise mismatch vectors The next data structure we generate is the pairwise mismatch vector of each SNP. In this discussion we assume a panel of M markers and S samples. We augment each SNP in the panel with a pairwise mismatch vector, v. This ternary vector has a length of $S(S-1)/2$, with one entry per sample–sample pair. Each entry, v_{ij}, where $j > i$, is set to '1/2' if either s_i or s_j is missing, '0' if $s_i = s_j$, and to '1' if $s_i \neq s_j$ (in practice we scale these values by a factor of 2).

The pairwise mismatch vectors can be computed efficiently by the following method. We first make three ternary vectors of length S.

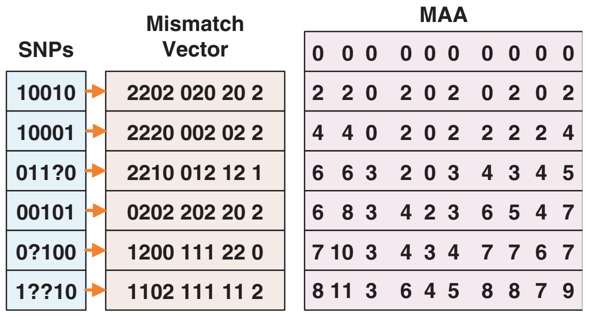

Fig. 1. Our fast imputation method relies on an auxiliary data structure called a mismatch accumulator array (MAA). The number of differences between any two haplotypes within an arbitrary window can be computed by subtracting the MAA entries for the two SNPs bounding the window.

The first vector, z, reflects a mapping of all '1's of the SNP to '2's, all '?'s to '1's, and all '0's to '0's. The second vector, o, maps all SNP '0's to '2's, all '1's to '0's, and all '?'s to '1's. The third vector, q, is all '1's. Next, the pairwise mismatch vector, v, is initialized to an empty vector, and we iterate through the SNP vector (s_i, $i = 0$ to $S-1$) performing the following. If $s_i ==$ '0', we concatenate the vector slice $z[i+1, S-1]$ to v, if $s_i ==$ '1', we concatenate $o[i+1, S-1]$, and if $s_i ==$ '?', we concatenate $q[i+1, S-1]$. When the iteration completes, the pairwise mismatch vector has a length of $S(S-1)/2$ and has been filled as specified.

2.1.3 Mismatch accumulator array Exploring the space of all window sizes surrounding each allele across a large panel of samples appears, at first glance, to be a daunting computational task. Moreover, the task of estimating the imputation accuracy for a given window size requires inferring genotypes that are already known. Ideally, these tasks would be performed exhaustively (every known genotype would be inferred for every window size) to find an optimum setting for the actual unknown genotypes. The immensity of the implied computations would lead most analysts to resort to sampling techniques.

We have developed a data structure, called a mismatch accumulator array (MAA), which tabulates the most time-consuming inner-loop calculations needed to compare genotypes within any window size. Furthermore, it avoids all of the redundant calculations in the naive approach. The cost of constructing an MAA scales linearly with the number of SNPs, and it permits comparisons of sequence similarities between samples within sliding windows of arbitrary sizes in constant time. This allows our method to be both exhaustive and fast. Furthermore, its accuracy is competitive with the most advanced statistically based approaches at a small fraction of their computation time. The MAA can also be easily adapted to perform other tasks such as finding all identical sub-regions among sample pairs.

As each SNP pairwise mismatch vector is computed, the MAA is built as shown in Figure 1. We first initialize a matrix of width $S(S-1)/2$ and height $M+1$, with the first row set to zero. We then loop through the SNPs in their sequence order ($i = 0$ to $M-1$). For each SNP, the mismatch vector is computed as in Section 2.1.2, added the vector to the previous entry (MAA_i) in the MAA, and the sum is placed in the next row (MAA_{i+1}).

For SNPs in the interior of a chromosome, we define a window of size L centered at marker i to extend L markers above and below. For SNPs less than L markers from the beginning or end of a chromosome, the window extends L SNPs in one direction and to the boundary of the chromosome in the other. To find the

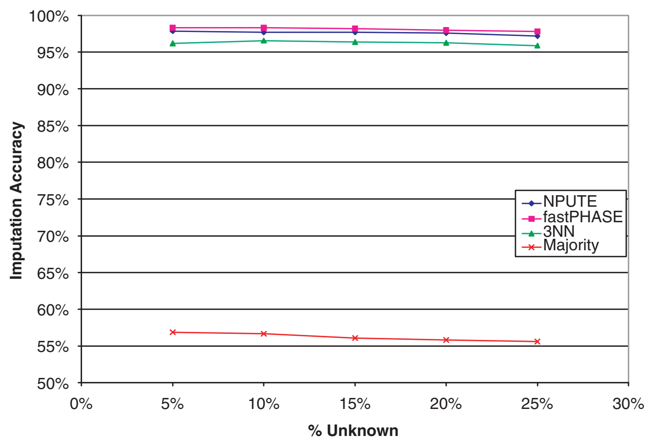

Fig. 2. Imputation accuracy by percent unknown (from 150 k). On a block of 1024 markers, 5, 10, 15, 20 and 25% random missed calls were simulated for five datasets each. The accuracy results after imputation of the same datasets by the four algorithms were then averaged. FastPHASE's accuracy is practically indistinguishable from NPUTE's (0.5% difference on average). Both performed over 1.4% better than 3NN, and outperformed majority allele imputation by over 40%.

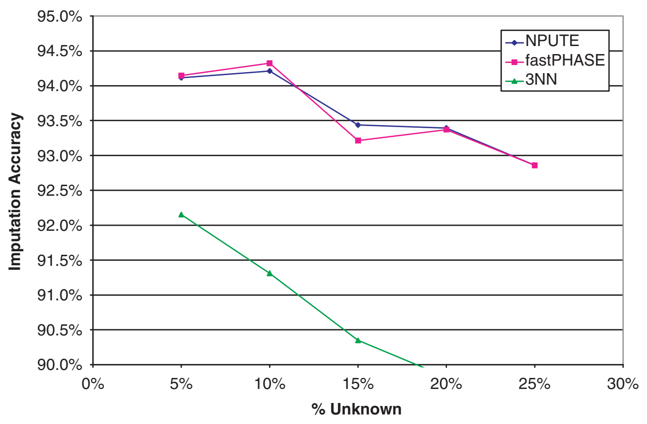

Fig. 3. Imputation accuracy by percent unknown (from Perlegen). The same method was used as in Figure 2 on 1024-SNP blocks from Perlegen. On these datasets, NPUTE slightly underperforms fastPHASE at 2 points, slightly outperforms fastPHASE at 2 other points, and ties at the final point. 3NN does significantly worse than these two methods (average of 3.1%), and majority is not plotted to improve scaling but is again over 40% lower.

mismatch vector of a window thus defined, we subtract the MAA vector indexed by max(i–L,0) from the vector indexed by min($i+L+1$,M) as follows:

$$W(i,L) = MAA_{\min(i+L+1,M)} - MAA_{\max(i-L,0)} \qquad (1)$$

When searching for optimal window sizes, we remove the contributions to the MAA of the SNP being imputed; this is easily accomplished with two more vector operations as follows:

$$W_t(i,L) = (MAA_{\min(i+L+1,M)} - MAA_{\min(i+1,M)}) + (MAA_i - MAA_{\max(i-L,0)}) \qquad (2)$$

This step is unnecessary when imputing actual missing data ('?'s), as they are given a mismatch of equal value (1/2) from all other alleles on the same SNP.

2.1.4 Mismatch hash table and window vector While effective on small datasets, this method requires significant storage on large sets (such as the 8.3 million markers of the Perlegen dataset). We have an alternative method for computing the windows in a constant time without having to store the entire MAA. The normalization of SNPs according to their majority and minority alleles enables us to easily find repeated SNP patterns in the dataset. Our alternative method takes advantage of this and reduces the space requirements to the number of unique SNP vectors, or strain distribution patterns (SDPs).

For this method, we build the pairwise mismatch vectors as before but store them in a hash table indexed by the SDP vector. Thus, all instances of each SDP share a mismatch vector. We also utilize a single window mismatch vector, w, to keep track of the window as we iterate through the SNP vectors ($i=0$ to $M-1$). w is initialized to the sum of the first L SNPs' mismatch vectors. For each window, starting at $i=0$, we add the vector for SNP $i+L+1$ to w and subtract the vector for SNP $max(i-L,0)$. When $i+L+1$ is greater than M, we add nothing to w. In other words, at each step we are adding the next SNP vector to the top of the window and subtracting the SNP vector from the bottom of the window. Thus, at any time, the value of $w=W(i,L)$ as defined in Equation (2) of Section 2.1.3.

When using this method during the search for an optimal L, we remove the center SNP by subtracting its mismatch vector from w.

2.1.4 Finding closest sample(s) The final piece of our imputation strategy is finding the best sample to use for imputing each allele. We do so by choosing the closest haplotype match to that which is being imputed inside the window $W(i,L)$, where i is the index of the SNP being imputed. Recall that $W(i,L)$ is a vector of size $S(S-1)/2$ containing the pairwise mismatches between all haplotypes in the window. When imputing sample h in SNP i, we extract the $S-1$ mismatch values between h and the other haplotypes and organize them in ascending order. The closest haplotype with a known allele is used for the imputation. If multiple haplotypes are tied for the minimum mismatch value, we allow each to vote for the call. A tie in the vote is broken by moving to the next closest mismatch values and so on.

2.2 Our imputation approach

There are two phases to our imputation method. In the first phase, we scan the panel to find an optimal window size for imputation based on our ability to accurately predict the called genotypes. In the second phase, we use this window size to infer the missing (uncalled) genotypes.

2.2.1 Finding the optimal window We estimate the best window size for the imputation by scanning over a large range of windows and estimating our imputation accuracy for known allele values. Due to the speed of our data structures, we are able to estimate based on every known allele in the dataset instead of relying on sampling.

For each window size (L) in our test set, we iterate through the vectors in the MAA ($i=0$ to $M-1$), treating i as the SNP in the center of a window of size L. We find the mismatch vector for the window using (2) in Section 2.1.3. This formula removes the mismatch contribution of SNP i so that its own known value will not bias the imputation. We choose an imputation value for each allele of the SNP using the strategy in Section 2.1.4 and compare it with the actual allele value. The only alleles we do not test in this way are:

(1) Missing genotypes (?'s).
(2) Singleton alleles.

Singleton alleles are ignored because we effectively mask out the allele being imputed. If we mask out a singleton allele, the SNP is no longer biallelic as was assumed.

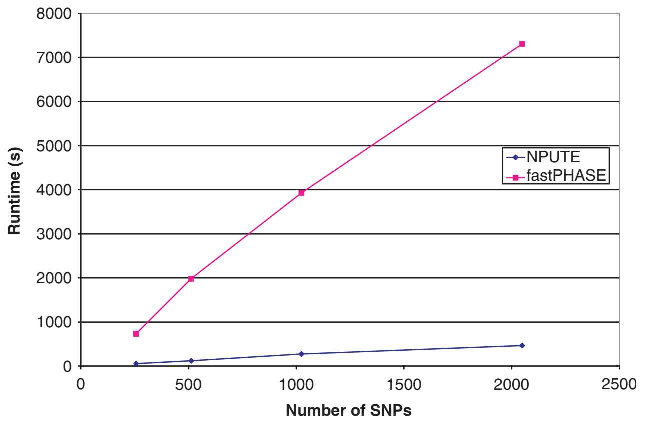

Fig. 4. Imputation runtime by data block size (150 k). Imputation runtime was determined for random 256, 512, 1024 and 2058 marker blocks of the 150 k set for NPUTE and fastPHASE. The runtimes for both methods grow linearly with datasets of increasing size. NPUTE ran faster than fastPHASE for datasets of all sizes.

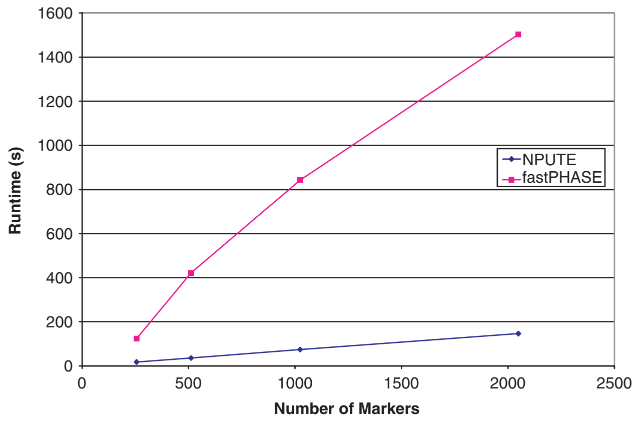

Fig. 5. Imputation runtime by data block size (Perlegen). Imputation runtime was determined in the same manner as in Figure 4 using blocks from the Perlegen set. As on the 150 k set, linear growth is visible, and NPUTE ran faster than fastPHASE for all datasets. The shorter impute times on the Perlegen set are due to the smaller S (16 in Perlegen, 45 in 150 k) with a constant M.

In this manner, we can calculate how well our imputation method would have imputed each known value in the set and determine a good estimate of our performance on the actual missed calls for each window size.

In all datasets that we have tested using our approach, reasonable L's have ranged from 5 to 150. We believe the optimal window size can be found with a search of no more than 100 windows, and thus we have set that as the default number to test. On larger sets (such as the 150 k and Perlegen sets described below) there is a clear decrease in accuracies after the optimal window size has been passed. Such trends permit early termination of the search.

2.2.2 Imputing unknowns After the optimal window size has been estimated, the actual imputation can be carried out. The method is the same as above except only missing calls are imputed and are overwritten with the computed value. Since only missing calls are being imputed, there is no need to subtract out the center mismatch vector as before and Equation (1) is used instead of (2).

3 RESULTS

We have compared our imputation speed and accuracy to several commonly reported imputation methods and to publicly available tools. These range from simple methods, such as assigning all missed calls to the majority allele and K-nearest neighbors voting methods (Troyanskaya *et al*. 2001; Xie *et al*. 2005; Wang and Dudoit, 2004), to the more complex Bayesian (Scheet and Stephens, 2006) approach.

FastPHASE is a publicly available Bayesian algorithm that is widely used for phasing human genotypes. In addition to phasing, it provides an option for imputing missing calls. FastPHASE is a speed improvement over the original PHASE algorithm (Stephens *et al*., 2001), which is widely regarded as the most accurate phasing algorithm (Marchini *et al*., 2006).

All comparative analyses were performed on identical datasets for all methods reported.

3.1 Test data

We compared imputation methods using two test datasets. The first dataset was extracted from a set of combined SNPs from the 140 k Broad/MIT mouse dataset (Wade and Daly, 2005) and the 10 k GNF mouse dataset. This merged dataset has 156 798 SNPs, 46 mouse strains and 4.2% missing genotype calls. We refer it as the 150 k set throughout the rest of this article. The second dataset used was extracted from the Perlegen 8.3 million SNP set, which provides high-resolution map of 16 common mouse strains with 11.1% missing calls. We refer to it as the Perlegen dataset, and it is available at http://mouse.perlegen.com.

To make these datasets a practical size for our competitor's imputation methods, we chose a subset of SNPs from each dataset to run comparisons. We first found the 1024 consecutive markers with the fewest number of missing calls from each dataset (this was easily accomplished using a variant of our MAA data structure). We randomly inserted missed calls to make five simulated sets with each of the following percentages of unknowns: 5, 10, 15, 20 and 25%, and preserved the original values for comparison. Note that the actual missed calls were left in the ground truth datasets and the simulated unknowns were added to reach a desired percentage. These sets were imputed using each method, and the results were compared with the ground truth at points where the simulated missed calls were inserted. We averaged the accuracies for each of the five sets having the same percentage of missed calls.

3.2 Accuracy comparisons

We compared the accuracy of our method, NPUTE, with fastPHASE, majority allele imputation and KNN. The results of the comparison are shown in Figure 2 for a 1024 marker block from the 150 k set. In terms of imputation accuracy, NPUTE is competitive with fastPHASE, and it outperforms both majority allele and KNN imputation methods.

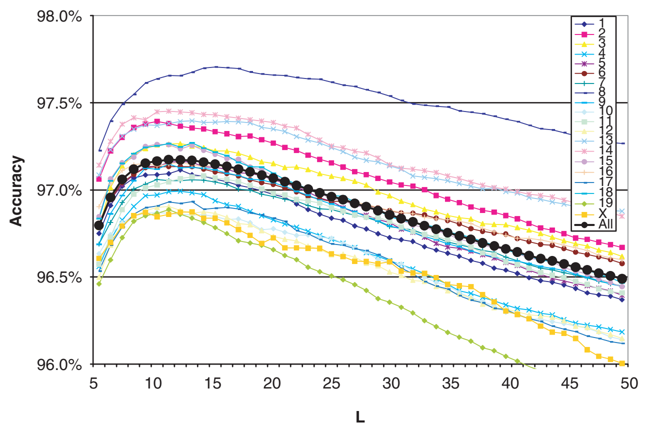

Fig. 6. Estimated accuracy for global and per-chromosome windows (150 k). NPUTE's global performance peaks at $L = 11$ (~350 KB on average), with an estimated accuracy of 97.2%. The peak allows us to prune the search space. The optimal L varies from 10–15 by chromosome in the dataset, and using optimal chromosome Ls yields an improvement of hundreds of calls.

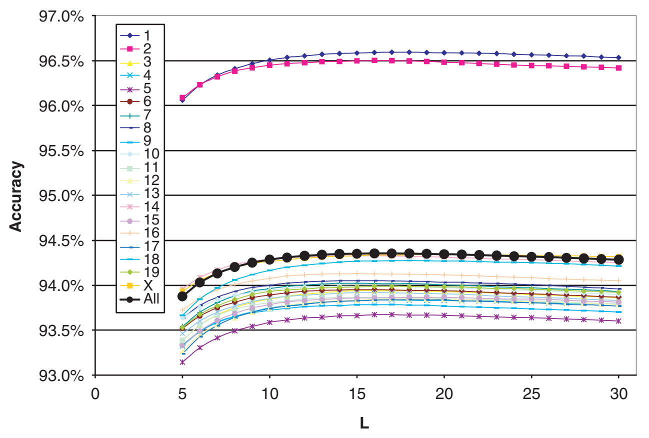

Fig. 7. Estimated Accuracy for global and per-chromosome windows (Perlegen). Global performance peaks at $L = 16$ (~10 KB on avg.), with an estimated accuracy of 94.5%. The search space is pruned earlier due to the higher search costs but does not result in a loss of accuracy. Optimal L varies from 14–20 for different chromosomes. Chromosome windows improve accuracy in over 2 K calls. Smoother curves are due to denser markers.

As expected, the imputation accuracy falls off for all methods as the miss rate increases. For KNN, $k = 3$ is shown because it had the best performance. Our method (NPUTE) and fastPHASE are clear winners, with fastPHASE doing slightly (<0.5%) better. Figure 3 shows the results for just NPUTE and fastPHASE on a 1024 marker block from the Perlegen set. Again, the two methods are essentialy equivalent in their average accuracy.

3.3 Speed comparisons

For speed comparison we ran NPUTE and fastPHASE on 5 random, consecutive blocks of 256, 512, 1024 and 2048 markers taken from the 150 k and Perlegen datasets, giving 40 sets in total. We ran both algorithms on a 3 GHz Pentium 4 CPU with 1 GB of RAM. We did not compare the majority and KNN methods since they were unoptimized and derived from the same code base as NPUTE. The runtime results for the blocks from the 150 k and Perlegen sets are reported in Figures 4 and 5, respectively. NPUTE is typically more than 10 times faster than fastPHASE on these sets with comparable accuracy.

3.4 Full genome imputation

The speed of our method enables us to impute full genome datasets such as the 150 k and 8.3 million SNP (Perlegen) datasets.

3.3.1 Speed and accuracy results for global window Figure 6 shows the estimated accuracy of NPUTE over the entire 150 k dataset for each window size. We are able to prune our optimal window search after the clear descent in the accuracy curve is seen following $L = 11$. Our estimated accuracy at this optimal point is 97.2%. For comparison, KNN with $k = 3$ has an estimated accuracy of 81.5%. The average time to test each window in this large set is 7.5 min and in comparison to the search, actual imputation time is negligible. Extrapolating from fastPHASE's performance on small blocks of the 150 k set, we estimate it would take ~164 h to impute the full set.

We have also estimated our imputation on the Perlegen dataset, the results for which can be seen in Figure 7. Again, we have pruned the window search after the peak to save time. The average time to test each window in this set is 135 min and the full imputation including the window search of 20 sizes runs in ~45 h. The estimated imputation accuracy for the optimal window ($L = 16$) is 94.4%. For comparison, KNN with $k = 3$ imputes is estimated to impute the dataset with 85.6% accuracy. Extrapolating from fastPHASE's performance on smaller blocks from the Perlegen set, we estimate it would take ~88 days to impute the same data.

3.3.2 Speed and accuracy results for chromosome windows We may also choose to search for an optimal window for each chromosome instead of globally. Figures 6 and 7 show the window search results for all chromosomes of 150 k and Perlegen, respectively. While the optimal window size tends to vary only slightly between chromosomes, the number of additional correct calls above the global method is in the thousands for Perlegen. Because the time requirements are the same, imputing the chromosomes separately is recommended.

3.3.2 Confidence score analysis Using the 150 k and Perelegen datasets, we have been able to derive a confidence score that is applicable across datasets. For each imputation call, we take the number of mismatches m in the window between the haplotype being imputed and the haplotype used for the imputation. This is referred to as the minimum mismatch score. We divide m by $2L$ to normalize the score so that it is between 1 and 0. These normalized score for each imputed value can then be reported as our confidence in the

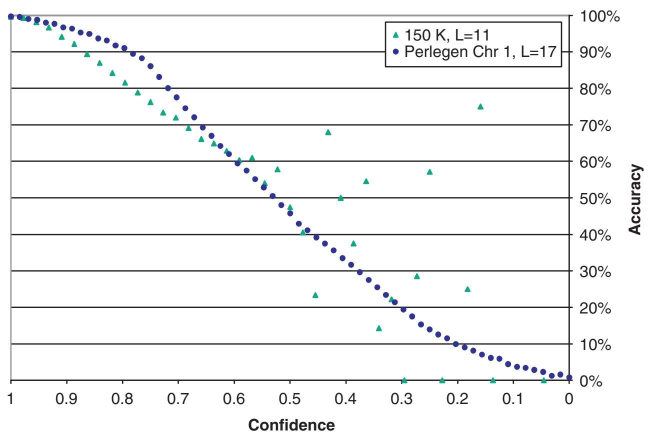

Fig. 8. Estimated accuracy by confidence score on 150 k and Perlegen sets. The normalized minimum mismatch confidence score is a good predictor of imputation accuracy. The random distribution of points for confidence scores less than 0.5 on the 150 k set is due to their sparseness.

calls we have made. Figure 8 shows the accuracy of NPUTE based on this confidence score in both the Perlegen and 150 k datasets. For the 150 k set, there are so few data points for confidence scores less than 0.5 that the accuracies are random on the right half of the graph. Nevertheless, the results for both sets show that this confidence score is a good predictor of imputation accuracy.

4 DISCUSSION

We have presented a conceptually simple, yet extremely effective approach for imputing 'no-call' genotypes. At its core, our method finds the most similar haplotype and uses it to infer the missing data. In regions of high LD, and when given a sufficient sample size, it is not surprising that a method like ours would be effective. A key strength of our approach is its ability to search over sliding widows of arbitrary sizes. This allows us to exhaustively test all window sizes, and to tune the search windows on a per-chromosome, or chromosomal region basis.

A data structure called the MAA enables the efficient window searches required by our approach. The MAA is also extremely versatile. We have made small modifications to it that allow quick scanning for contiguous SNP windows with the fewest missed calls, as well as finding the longest contiguous matching interval for every haplotype pair indexed by SNP position.

Our method is not dependent on any low-level optimizations or hardware-dependent optimizations. In fact, it is entirely implemented using an interpreted language (Python).

One can easily imagine small tweaks to further improve upon our imputation accuracy, such as allowing for non-symmetric windows, or selecting windows on the basis of haplotype blocks, which are found as a preprocess. We can also limit imputation to sites with high confidence, as estimated by the mismatch score, for tools that are able to tolerate missing calls. This also suggests the possibility of an iterative approach that imputes highest confidence no-calls, updates the MAA, and repeats until every site is imputed.

Another application of the confidence score is to use NPUTE for multiple imputation, as proposed by Rubin (1977). A biased coin with probability determined by the confidence score is flipped at each missed genotype to generate multiple imputed datasets with random variation. These sets can then be analyzed and combined by an MI method.

Our algorithm scales in both time and space as O(M S^2), where M is the number of markers and S the number of samples. This can be problematic for datasets with a large S. The dynamic construction of the window vector described in Section 2.1.4 reduces the memory requirement from M to the largest window size, but it is not helpful for datasets with many samples.

For these datasets, we have developed a variant of the MAA. Rather than constructing mismatch vectors based on all pairs, we instead select a subset of k-representative samples (on a chromosome basis) and then find the number of mismatches between each sample and these k-representatives. This results in a mismatch vector of length k S, rather than $S(S-1)/2$. We select these representatives using a k-medoid clustering approach. This reduces the space and time complexity of our imputation method to O(M k S) with a minimal effect on the imputation accuracy. For example, on the 150 k dataset, selecting k to be 8 out of 46, this method only reduces the imputation accuracy from 97.2 to 94.5%.

Another application for our algorithm besides imputation is in genotype quality control. We have noted that some of our highest confidence calls can disagree with the data as given. Such discrepancies might imply genotyping errors rather than imputation errors. Our confidence measure can be used to assign a quality score to all values in a dataset. Low quality calls could then be resequenced in order to improve the integrity of the data. Undoubtedly, even some of our highest confidence calls will be incorrect but our algorithm may be able to point out areas where additional verification will be useful.

ACKNOWLEDGEMENTS

We thank all members of the UNC Computational Genetics working group for discussion of our project and Lynda Yang for help running experiments. We thank the Stephens lab for access to fastPHASE used in the evaluations for this article. Generous grants provided by EPA STAR RD832720, NIH U01CA105417, and NSF IIS 0448392 funded parts of this research.

Conflict of Interest: none declared.

REFERENCES

Dai,J.Y. *et al.* (2006) Imputation methods to improve inference in SNP association studies. *Genet. Epidemiol.*, **30**, 690–702.
Eskin,E. *et al.* (2003) Efficient reconstruction of haplotype structure via perfect phylogeny. *J. Bioinform. Comput. Biol.*, **1**, 1–20.
Huentelman,M. *et al.* (2005) SNiPer: improved SNP genotype calling for Affymetrix 10K GeneChip microarray data. *BMC Genomics*, **6**, 149.

Kang,S.J. *et al*. (2004) Tradeoff between no-call reduction in genotyping error rate and loss of sample size for genetic case/control association studies. *Pac. Symp. Biocomput*., **9**, 116–127.

Lin,S. *et al*. (2002) Haplotype inference in random population samples. *Am. J. Hum. Genet*., **71**, 1129–1137.

Marchini,J. *et al*. (2006) A comparison of phasing algorithms for trios and unrelated individuals. *Am. J. Hum. Genet*., **78**, 437–450.

Niu,T. *et al*. (2002) Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. *Am. J. Hum. Genet*., **70**, 157–169.

Qin,Z.S. *et al*. (2002) Partition-ligation-expectation maximization algorithm for haplotype inference with single nucleotide polymorphisms. *Am. J. Hum. Genet*., **71**, 1242–1247.

Rubin,D.B. (1977) Formalizing subjective notions about the effect of nonrespondents in sample surveys. *J. Am. Stat. Assoc*., **72**, 538–543.

Scheet,P. and Stephens,M. (2006) A fast and flexible statistical model for large-scale population genotype data: applications to inferring missing genotypes and haplotypic phase. *Am. J. Hum. Genet*., **78**, 29–644.

Stephens,M. *et al*. (2001) A new statistical method for haplotype reconstruction from population data. *Am. J. Hum. Genet*., **68**, 978–989.

Su,S. *et al*. (2005) Inference of missing SNPs and information quantity measurements for haplotype blocks. *Bioinformatics*, **21**, 2001–2007.

Threadgill,D.W. *et al*. (2002) Genetic dissection of complex and quantitative traits: from fantasy to reality via a community effort. *Mamm. Genome*, **13**, 175–178.

Troyanskaya,O. *et al*. (2001) Missing value estimation methods for DNA microarrays. *Bioinformatics*, **17**, 520–525.

Wade,C.M. and Daly,M.J. (2005) Genetic variation in laboratory mice. *Nat. Genet*., **37**, 1175–1180.

Wang,Y. and Dudoit,S. (2004) Quantification and visualization of LD patterns and identification of haplotype blocks (2004). *U.C. Berkeley Division of Biostatistics Working Paper Series*, Working Paper 150.

Xie,Q. *et al*. (2005) Decision forest analysis of 61 single nucleotide polymorphisms in a case-control study of esophageal cancer; a novel method. *BMC Bioinformatics*, **6**, S4.