BIOINFORMATICS

Vol. 00 no. 00 2005
Pages 1–9

Tag SNP Selection in Genotype Data for Maximizing SNP Prediction Accuracy

Eran Halperin*[a], Gad Kimmel* [b], Ron Shamir[b]

[a]International Computer Science Institute, Berkeley, Ca 94704, USA.
[b]School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel.
*These authors contributed equally to this paper.

ABSTRACT
Motivation: The search for genetic regions associated with complex disease, such as cancer or Alzheimer's disease, is an important challenge that may lead to better diagnosis and treatment. The existence of millions of DNA variations, primarily single nucleotide polymorphisms (SNPs), may allow the fine dissection of such associations. However, studies seeking disease association are limited by the cost of genotyping SNPs. Therefore, it is essential to find a small subset of informative SNPs (tag SNPs) that may be used as good representatives of the rest of the SNPs.
Results: We define a new natural measure for evaluating the prediction accuracy of a set of tag SNPs, and use it to develop a new method for tag SNPs selection. Our method is based on a novel algorithm that predicts the values of the rest of the SNPs given the tag SNPs. In contrast to most previous methods, our prediction algorithm uses the genotype information and not the haplotype information of the tag SNPs. Our method is very efficient, and it does not rely on having a block partition of the genomic region.

We compared our method to two state of the art tag SNP selection algorithms on 58 different genotype data sets from four different sources. Our method consistently found tag SNPs with considerably better prediction ability than the other methods.
Availability: The software is available from the authors upon request.
Contact: kgad@tau.ac.il

1 INTRODUCTION

Most of the genetic variation among different people can be characterized by single nucleotide polymorphisms (SNPs), which are mutations at single nucleotide positions that occurred during human history and were passed on through heredity. Most of these SNPs are bi-allelic, that is, only two bases (*alleles*) are observed across the population at such sites. It has been estimated that there are about seven million common SNPs (i.e., SNPs with minor allele frequency of at least 5%) in the human genome [11, 3]. Alleles of SNPs in close physical proximity to each other are often correlated, and the variation of the sequence of alleles in contiguous SNP sites along a chromosomal region (*haplotype*) is known to be of limited diversity. The identification and analysis of haplotypes, currently a major effort of the international community [17], is expected to play a key role in trait and disease association studies [12, 13].

The objective of disease association studies is to find genetic factors correlated with complex disease. In these studies, the DNA of individuals from two populations (healthy individuals and carriers of the disease) is sampled. Then, discrepancies in the haplotype structure of the two populations are revealed by various statistical tests. These discrepancies serve as evidence for the correlation of the genomic region studied with the disease.

Clearly, the statistical significance of the study is directly affected by the number of individuals typed. On the other hand, the total cost of the study is also affected by the number of SNPs typed. Therefore, to save resources, one wishes to reduce the number of SNPs typed per individual. This is usually done by choosing an appropriate small subset of the SNPs, called *tag SNPs*, that could predict the rest of the SNPs with a small error. Thus, when preforming a disease association study, the geneticist would experimentally test for association by only considering the tag SNPs, thereby considerably saving resources (or, alternatively, increasing the power of the statistical tests by increasing the number of individuals). Hence, a key problem is to find a set of tag SNPs of minimum size that would have a very good prediction ability. In this paper we propose a new method called STAMPA (Selection of Tag SNPs to Maximize Prediction Accuracy) that finds a set of tag SNPs given a genotype sample taken from a set of unrelated individuals.

Finding a high-quality set of tag SNPs is a challenging task for several reasons. One of the main challenges is that the haplotype information is usually not given, and instead we get the genotypes. As opposed to haplotypes, the *genotypes* give the bases at each SNP in both copies of the chromosome, but lack the *phase*, i.e., information as to the chromosome on which each base appears. Due to technology constraints, most sequencing techniques provide the genotypes and not the haplotypes. There are however, computational tools that use the correlations between neighboring SNPs in order to predict the phase information. Their accuracy depend on the proximity and correlation of the SNPs tagged. When a set of tag SNPs is chosen and then tagged, the rest of the SNPs are not measured and instead must be predicted from this information. The accuracy of such prediction is limited, since the correlation between the tag SNPs is not necessarily as strong as the correlation between SNPs that are in close proximity to each other. One of the advantages of our tag SNPs predictor is that it only uses the genotype information and does not require knowledge of the haplotypes of the tag SNPs. We use the phase information in a reference training set to select the tag SNPs, and subsequently predict the other SNP values in a test individual on the genotype of that individual for the tag SNPs only. To the best of our knowledge, all extant programs that aim to explicitly predict individual SNPs use the haplotypes of the tag SNPs.

Another issue that is crucial in the search for tag SNPs is the definition of an adequate measure of the prediction quality. Many of the current tag SNP selection methods partition the region into blocks of limited diversity (e.g., [18, 20, 19]), and find a set of tag SNPs that aims to predict the common haplotypes of each block. There are various disadvantages to such methods, most apparent is the lack of cross-block information and the dependency of the tag SNPs choice on the block definition. We propose here a new natural measure, *prediction accuracy*, which directly evaluates the average SNP prediction quality.

There is a large body of research on finding a highly predictive set of tag SNPs [18, 1, 2, 4, 14]. In contrast to most previous methods, our method uses the genotype information for the tag SNP selection. Zhang et al. [19] have also used genotypes information for tag SNP selection. However, their study selects the SNPs so as to maximize haplotype diversity, and given the genotypes of the tag SNPs in a tested individual it infers blocks and common haplotypes, but does not predict the individual SNPs. Another key difference between our method and previous ones is that we do not rely on any block partition.

We performed extensive tests of STAMPA on genotypes from a variety of sources. Our tests covered 58 datasets from four sources: HapMap project [17], ENCODE project [17], Daly et al. [5], and Gabriel et al. [7]. We show that using STAMPA, very accurate results are achieved. For example, only 17 tag SNPs out of 103 SNPs (16.5%) suffice to attain prediction accuracy of 95% in the data of Daly et al.. Our method is also very efficient: Runs on a regular PC required seconds to several minutes on all datasets.

We compared our algorithm to two state of the art tag SNP selection algorithms: ldSelect [4] and HapBlock [19]. Our experiments show that STAMPA consistently outperforms both of these methods. On the average ldSelect uses ten times more tag SNPs than STAMPA in order to achieve prediction accuracy of 90%. Our algorithm was also more accurate than HapBlock on each of the 58 datasets, sometimes by more than 15%. Moreover, the running time of STAMPA was much smaller than HapBlock. For example, on chromosome 5q31 data set, STAMPA was faster by a factor of 97. Such advantage will be more prominent on future larger data sets.

2 PROBLEM FORMULATION

In order to present our method, we first formalize the problem of tag SNPs prediction. We first need to introduce some notations and definitions. Since we are only interested in bi-allelic SNPs, we assume that each haplotype is represented by a binary string. Thus, a haplotype of length m is a sequence over $\{0,1\}^m$. A genotype of length m is represented by a $\{0,1,2\}$ sequence, where 0 and 1 stand for the homozygous types {0,0} and {1,1}, respectively, and 2 stands for a heterozygous type. We are given a set of n genotypes $g_1, \ldots, g_n$ of length m each. We use $g_{i,j}$ to denote the j-th component (0,1, or 2) of the vector g_i. A *phasing* of a genotype g_i is a pair of haplotypes, $h_i^1, h_i^2 \in \{0,1\}^m$, such that $h_{i,k}^1 \neq h_{i,k}^2$ if $g_{i,k} = 2$ and $h_{i,k}^1 = h_{i,k}^2 = g_{i,k}$ if $g_{i,k} \in \{0,1\}$. We also use the notation $g(j)$ to denote the j-th SNP in genotype g.

Consider a genomic region that spans a set of m SNPs. The frequencies of the genotypes in that region across the entire populations are given by some unknown distribution function $\Pr(g_i \in \mathcal{G})$, where $\mathcal{G}$ is the sample space of all genotypes in the population. A prediction algorithm is a function $f : \{0,1,2\}^t \rightarrow \{0,1,2\}^m$. Informally, the prediction algorithm uses the genotype values of the tag SNPs in order to predict the values of the rest of the SNPs. For a given vector $q \in \{0,1,2\}^t$ of tag SNPs values, let $f_j(q)$ denote the j-th component of that vector. Note that f_j refers to the components of the predicted vector of all m SNPs, given the tag genotypes q. Finally, let $z_T : \{0,1,2\}^m \rightarrow \{0,1,2\}^t$ be the restriction of the genotype to the tag SNPs position. For instance, for a set of tag SNPs $T = \{1,3,5,6\}$, the restriction of the genotype $g_i = 0122010$ is $z_T(g_i) = 0201$.

Our goal is to find a minimum size set of tag SNPs and a prediction algorithm, such that the prediction error is minimized. Formally, for a given t, our objective is to find a set of tag SNPs T of size t, and a prediction function f, such that the following expression is minimized.

$$\eta = \sum_{j=1}^{m} \Pr[f_j(z_T(g)) \neq g(j)], \tag{1}$$

where the probability is over the sample space given by $\Pr(g \in \mathcal{G})$. In other words, for a randomly picked individual from the population, we want to minimize the expected number of prediction errors.

The main problem in achieving this goal is that the frequencies of the genotypes in the population are unknown. Therefore, we use a training dataset of genotypes, $g_1, \ldots, g_n$

in order to learn the distribution of genotypes in the data. For a given prediction algorithm $f : \{0,1,2\}^t \to \{0,1,2\}^m$, we are interested in finding a set of tag SNPs T of size t, such that expression (1) is minimized when the genotype is randomly picked from the training set. Formally, if $X_T = |\{(i,j) \mid g_{i,j} \neq f_j(z_T(g_i))\}|$, where $g_{i,j}$ is the j-th SNP of g_i, then we are looking for a set T of SNPs of size t such that X_T is minimized. The resulting prediction rate of the tag SNPs depends both on the prediction function f and on the choice of the tag SNPs.

3 THE PREDICTION ALGORITHM

In this section we present our prediction algorithm. The algorithm is based on the observation made by several biological studies, that the correlation between SNPs tends to decay as the physical distance increases (see, e.g., [7, 2, 5, 9]). We assume that given the genotypes values of two SNPs, the probabilities of the values at any intermediate SNPs do not change by knowing the values of additional distal ones. Formally, this assumption can be stated as:

$$\forall s : a < s < b,\ \forall q : q < a \text{ or } q > b,\ \forall v \in \{0,1,2\},\ \forall i : \Pr[g_{i,s} = v | g_{i,a}, g_{i,b}] \approx \Pr[g_{i,s} = v | g_{i,a}, g_{i,b}, g_{i,q}]. \qquad (2)$$

Thus, our prediction function predicts a SNP value using only the values of the two closest tag SNPs to this SNP. Precisely, $f_i(z_T(g)) = f(g_{j_1}, g_j, g_{j_2})$ where j_1 and j_2 are the closest tag SNPs to j, on both sides, if possible. Although many biological studies support this assumption, it clearly does not hold for all SNPs, or in all data sets. However, the assumption is a rather faithful approximation of the reality in most cases. As we shall show in Section 5, using this assumption we achieve very high prediction rates.

Given a set of tag SNPs $T = (s_1, \ldots, s_t)$, we use the procedure $\mathrm{Predict}$ given in Figure 1 to predict the value of SNP i given the value of the tag SNPs. We assume that we are given the training set of genotypes $g_1, \ldots, g_n$ together with their corresponding haplotypes $h_1^1, h_1^2, h_2^1, \ldots, h_n^2$, where $h_i^j = (h_{i1}^j, \ldots, h_{im}^j) \in \{1,2\}^m$ for $j = 1, 2$. The haplotypes can be computed from the genotypes using a variety of available algorithms (e.g., [10, 6, 15, 8]).

Let j_1 and j_2, $j_1 < i < j_2$ be the positions of the tag SNPs closest to position i on both sides. If there is no tag SNP in position $j_2 > j$, then j_1 and j_2 are the last two SNPs, and if there is no tag SNPs in position $j_1 < j$ then j_1 and j_2 are the first two SNPs. The procedure $\mathrm{Predict}(i, j_1, j_2, a_1, a_2)$ uses a majority vote in order to determine which value is more likely to appear in position i given that positions j_1 and j_2 have the values $a_1 \in \{0,1,2\}$ and $a_2 \in \{0,1,2\}$ respectively. In order to use the phased information given by the model, we use two majority votes to determine the two different alleles. For instance, if $a_1 = 0$ and $a_2 = 2$, we find the most likely allele given that the alleles in positions j_1 and j_2 are both 0, and another allele given that the alleles in positions j_1 and j_2 are 0 and 1 respectively. Further details are given in Figure 1. Note that predicting SNP i using the procedure $\mathrm{Predict}$ makes no use of most of the tag SNPs - we simply ignore all the tag SNPs except for the ones closest to i.

4 ALGORITHMS FOR TAG SNP SELECTION

Recall that our goal is to find a set of tag SNPs T of size t, such that X_T is minimized, where $X_T = |\{(i,j) \mid g_{i,j} \neq \mathrm{Predict}(j, j_1, j_2, g_{i,j_1}, g_{i,j_2}))\}|$. We give two algorithms for selecting the tag SNPs. Both algorithms use the prediction algorithm as a subroutine. The first is a polynomial algorithm that guarantees an optimal solution. The second is a simpler and faster random sampling algorithm. We shall discuss their performance in Section 5.

4.1 An Exact Algorithm

We now describe an algorithm that solves this problem to optimality. The algorithm, called STAMPA (Selection of Tag SNPs for Maximizing Prediction Accuracy), uses dynamic programming.

Let $X_T^{i,j} = 1$ if $g_{i,j} \neq \mathrm{Predict}(j, j_1, j_2, g_{i,j_1}, g_{i,j_2})$ and let $X_T^{i,j} = 0$ otherwise. Clearly, $X_T = \sum_{i,j} X_T^{i,j}$. For every pair of SNPs $m_1 < m_2$ we next define three auxiliary score functions, score(m_1, m_2), score$_1(m_1, m_2)$ and score$_2(m_1, m_2)$, which will be used in the dynamic program recursion. These score functions evaluate the expected number of errors in a subregion (a contiguous set of SNPs), given a partial set of the tag SNPs. We assume that $m_1, m_2 \in T$ and that for each $m_1 \leq j \leq m_2$, $j \notin T$. Then, we define

$$\text{score}(m_1, m_2) = \sum_{i=1}^{n} \sum_{j=m_1}^{m_2-1} X_T^{i,j}.$$

Thus, score(m_1, m_2) is the total number of prediction errors in SNPs $m_1, \ldots, m_2 - 1$, given that m_1 and m_2 are tag SNPs, and that there are no tag SNPs between m_1 and m_2. Since the procedure $\mathrm{Predict}$ infers a SNP value by considering only its neighboring tag SNPs, we can readily compute the score, while disregarding the information on all the other tag SNPs.

For score$_1(m_1, m_2)$, we assume that m_1 and m_2 are the last two tag SNPs. Then, the score is defined as

$$\text{score}_1(m_1, m_2) = \sum_{i=1}^{n} \sum_{j=m_1}^{m} X_T^{i,j}.$$

Thus, score$_1(m_1, m_2)$ is the the total number of prediction errors in SNPs $m_1, \ldots, m$ when the last two SNPs are in positions m_1, m_2. Again, since $\mathrm{Predict}$ only uses the closest tag SNPs in order to compute the SNP values, we can compute score$_1$ independently of the locations of the rest of the SNPs.

Similarly, for score$_2(m_1, m_2)$ we assume that m_1 and m_2 are the first two tag SNPs, and define

$$\text{score}_2(m_1, m_2) = \sum_{i=1}^{n} \sum_{j=1}^{m_2-1} X_T^{i,j}.$$

```
ALGORITHM Predict($i$,$j_1$,$j_2$,$a_1$,$a_2$)
Input: $i, j_1, j_2 \in \{1, \dots, m\}$, and $a_1, a_2 \in \{0, 1, 2\}$.
Output: An integer $v \in \{0, 1, 2\}$ which is a predicted value of a SNP in position $i$, given that in position $j_1$ and $j_2$
the values are $a_1$ and $a_2$ respectively.
  1. For every $(x, y, z) \in \{0, 1\}^3$ we let $C(x, y, z) = \{(j, p) \mid h^p_{jj_1} = x, h^p_{jj_2} = y, h^p_{ji} = z\}$ be the set of haplotypes
     having the values $x, y, z$ in positions $j_1, i$ and $j_2$ respectively.
  2. Let $A(x, y) = z \in \{0, 1\}$, where $|C(x, y, z)| \geq |C(x, y, 1 - z)|$ breaking ties arbitrarily.
  3. Let $c(x, y) = |C(x, y, 0)| + |C(x, y, 1)|$.
  4. We compute the values of two variables $x, y$ using the following case analysis.
      • If $a_1 < 2$ and $a_2 < 2$, then we set $x = y = A(a_1, a_2)$.
      • If $a_1 = 2$, $a_2 = 2$ and $c(0, 0) \cdot c(1, 1) \geq c(0, 1) \cdot c(1, 0)$, then $x = A(0, 0)$ and $y = A(1, 1)$.
      • If $a_1 = 2$, $a_2 = 2$ and $c(0, 0) \cdot c(1, 1) < c(0, 1) \cdot c(1, 0)$, then $x = A(0, 1)$ and $y = A(1, 0)$.
      • If $a_1 = 1$, $a_2 = 2$ ($a_2 = 1$, $a_1 = 2$), then we set $x = A(1, 1)$ and $y = A(1, 0)$ ($y = A(0, 1)$).
      • If $a_1 = 0$, $a_2 = 2$ ($a_2 = 0$, $a_1 = 2$), then we set $x = A(0, 0)$ and $y = A(0, 1)$ ($y = A(1, 0)$).
  5. If $x \neq y$ output 2, else output $x$.
```

Fig. 1. The procedure Predict. We implicitly assume that the training set and its phase are given. The variables x and y computed by the case analysis represent the majority votes for the two haplotypes induced by the values a_1 and a_2. Note that the output value is determined by simply counting the frequencies of different partial haplotypes in the training set that match a_1 and a_2, and taking the majority vote.

In this case, score$_2(m_1, m_2)$ is the total number of prediction errors in SNPs $1, \dots, m_2 - 1$ when the first two SNPs are in positions m_1, m_2.

We next define the function f that will be used in the dynamic programming recursion. $f(m^*, l)$ is defined for $l \geq 2$ and $1 \leq m^* \leq m$. For $l < t$, the function $f(m^*, l)$ represents the minimum number of prediction errors in SNPs $1, 2, \dots, m^*$, given that the l-th tag SNP is in position m^*. For $l = t$, the function $f(m^*, t)$ represents the minimum number of prediction errors in all SNPs $1, 2, \dots, m$ given that the last tag SNP is in position m^*. Formally, we define $f(m^*, l)$ in the following way:

- For $l = t$, $f(m^*, t) = \sum_{i=1}^{n} \sum_{j=1}^{m} X_T^{i,j}$ when the last tag SNP is in position m^*.
- For $t > l \geq 2$, $f(m^*, l) = \sum_{i=1}^{n} \sum_{j=1}^{m^*-1} X_T^{i,j}$ when the l-th tag SNP is in position m^*.

It is easy to verify by the definitions of f and of score, score$_1$ and score$_2$, that the following recurrence relation holds:

$$f(m^*, l) = \begin{cases} \min_{1 \leq m' < m^*} \text{score}_2(m', m^*) & l = 2 \\ \min_{l-1 \leq m' < m^*} \{f(m', l-1) + \text{score}(m', m^*)\} & 2 < l < t \\ \min_{t-1 \leq m' < m^*} \{f(m', t-1) + \text{score}_1(m', m^*)\} & l = t \end{cases} \quad (3)$$

We now apply dynamic programming in order to find the value of $f(m^*, t)$ for every $t \leq m^* \leq m$, using the above recurrence relation. Since $f(m^*, t)$ is the total number of prediction errors given that the last tag SNP is in position m^*, it is clear that the minimum value of X_T over all possible sets of tag SNPs of size t is $\min_{\{m^* | t \leq m^* \leq m\}} f(m^*, t)$. Using back pointers in the process, one can also find a set of tag SNPs minimizing X_T.

Complexity analysis: We first compute the three scores for all $\binom{m}{2}$ possible pairs of SNPs. For every pair the running time is $O(mn)$. Hence, the total running time for this stage is $O(m^3 n)$. We keep the scores in a matrix, and we use that matrix in order to compute f. Given the computed scores, for every $m^* \leq m$, computing $f(m^*, 2)$ takes $O(m^*)$, so doing this for all m^* takes $O(m^2)$. Similarly, computing $f(m^*, i)$ for every $i < t, m^* < m$ takes $O(m^2 t)$. Finally, computing $f(m^*, t)$ for every $m^* \leq m$ takes $O(m^2)$. Since $t \leq m$ the total running time is $O(m^3 n)$.

If the number of SNPs is large (even in the hundreds), a running time of $O(m^3 n)$ is very expensive. On the other hand, in practice, the correlation between SNPs is decaying when the physical distance between the SNPs increases. Put differently, tag SNPs tend to predict well other SNPs in the same or neighboring block, but not further away. Thus, having a very large distance between neighboring tag SNPs yields poor prediction power. Hence, in most practical cases one can use a bound c on the maximal distance in SNPs between neighboring tag SNPs. c will depend on the SNP typing density and will typically not exceed 20 or 30. In such a case, computing score(m_1, m_2) takes $O(mc^2 n)$ and computing score$_1$ and score$_2$ takes $O(c^3 n)$. Computing $f(m^*, i)$ for each i takes $O(mtc)$. Thus, the total running time is $O(mtc + mc^2 n) = O(mc(cn + t))$.

4.2 Random Sampling

In some cases we are interested in finding quickly a very small number of tag SNPs that roughly predict the rest of the SNPs. That is, we are willing to give up some of the prediction power if we can get a very small number of tag SNPs. In those cases, the assumption that the tag SNPs are close to each other cannot be made, c is very large, and the exact algorithm may be too slow. We therefore suggest a very simple and much more efficient algorithm that does not guarantee optimal results.

The algorithm is as follows: We generate 100 sets of tag SNPs, $T_1, T_2, \ldots, T_{100}$, each generated by randomly picking t positions out of the m possible positions. We then compute X_{T_i} for $i = 1, 2, \ldots, 100$, and we choose the set of tag SNPs T_i that minimizes X_{T_i}. This algorithm is very naive, but we show that it gives reasonable results in practice.

5 RESULTS ON BIOLOGICAL DATASETS

5.1 Description of the Datasets

We used four datasets encompassing 58 different genomic regions.

- A dataset due to Daly et al. [5]. In this study, genotypes for 103 SNPs, from a 500 KB region of chromosome 5q31, were collected from 129 mother, father and child trios from European derived population in an attempt to identify a genetic risk for Crohn's disease. We only used the children population in this data set.
- Population D from the study of Gabriel et al. [7]. The data consist of 51 sets of genotypes from various genomic regions, with number of SNPs per region ranging from 13 to 114. The sets contained 30 mother, father, child trios that were taken from a Yoruba's population, from which we only used the 60 genotypes of the parents.
- Regions ENm013, ENr112 and ENr113 of the ENCODE project [17]. These are 500 KB regions of chromosomes $7q21.13, 2p16.3$ and $4q26$ respectively, which were collected from 30 trios. The number of SNPs genotyped in each region is $361, 412$ and 515 respectively (thus, the density of the sample is 3-5 times greater than the density of [5]). We used the 60 genotypes corresponding to the parents from each dataset.
- Genotypes from the HapMap project [17]. We used three sets of SNPs spanning the three genes PP2R4, STEAP, and TRPM8. For each of these genes we took the HapMap SNPs that are spanned by the gene plus 10 KB upstream and downstream. The resulting sets contain $39, 23$ and 102 SNPs. In this data set we used the parents genotypes.

5.2 Implementation

STAMPA was implemented in C. All reported runs used a Linux operating system on a 4Ghz PC using 500M cache. Running times are discussed below (Figure 3 and Table 2).

The $\mathrm{Predict}$ procedure requires a phased training set. To obtain that solution when applying STAMPA, we used the GERBIL algorithm [10]. Running times for phasing using GERBIL were almost always below one minute. The Daly et al. data set required the most time, about 2 minutes. These times are not included in the reporting below.

5.3 Exact Solution vs. Random Sampling Algorithm

We first measured the prediction accuracy of the two algorithms in Section 4. For STAMPA, we used $c = 30$ as the upper bound of distance between tag SNPs. The experiments were performed in a leave one out manner: We repeatedly removed one of the genotypes from the set, used the remaining genotypes as the training set in order to find a set of tag SNPs, and used these tag SNPs in order to predict the other SNPs in the removed genotype.

The results show that that STAMPA uses very few tag SNPs in order to predict the other SNPs with high confidence. For example, in chromosome 5q31 data set [5], typing 2 SNPs suffices to predict all of the 103 SNPs with 80% accuracy, 6 SNPs are needed to achieve 90%, and only 17 SNPs need to be typed for 95%.

The results of the comparison of the two algorithms are summarized in Figure 2. As expected, in most cases, STAMPA was more accurate than the random sampling algorithm. However, when the number of tag SNPs is small, there is a clear advantage for the random sampling algorithm. For example, in Encode region ENr113, when less than 15 tag SNPs are required, the prediction accuracy of the random sampling algorithm was higher. This gap can be explained by the fact that when the number of tag SNPs is small, the upper bound for the distance between tag SNPs is too restrictive for STAMPA. It is important to emphasize, that each of the two algorithms has a parameter, that can be increased to obtain more accurate results, but at the expense of larger running times. Such is the parameter c in STAMPA, and the number of samples in the random sampling algorithm. Although in our experiments we saw a clear advantage to STAMPA, in some situations we expect the opposite to be true, e.g., when SNPs are genotyped with high density in a very long region, and the number of tag SNPs is required to be very small.

5.4 Comparisons to Extant Methods

We chose to compare our algorithm to two recent algorithms for tag SNP selection that are widely used: ldSelect, due to Carlson et al. [4], which uses a greedy approach, and HapBlock due to Zhang et al. [19], which uses dynamic programming and a partition-ligation EM subroutine to phase subintervals in the recursion. Two additional tag SNP selection algorithms that were reported in the literature [2, 14] could not be included in the comparisons since their implementations were not available.

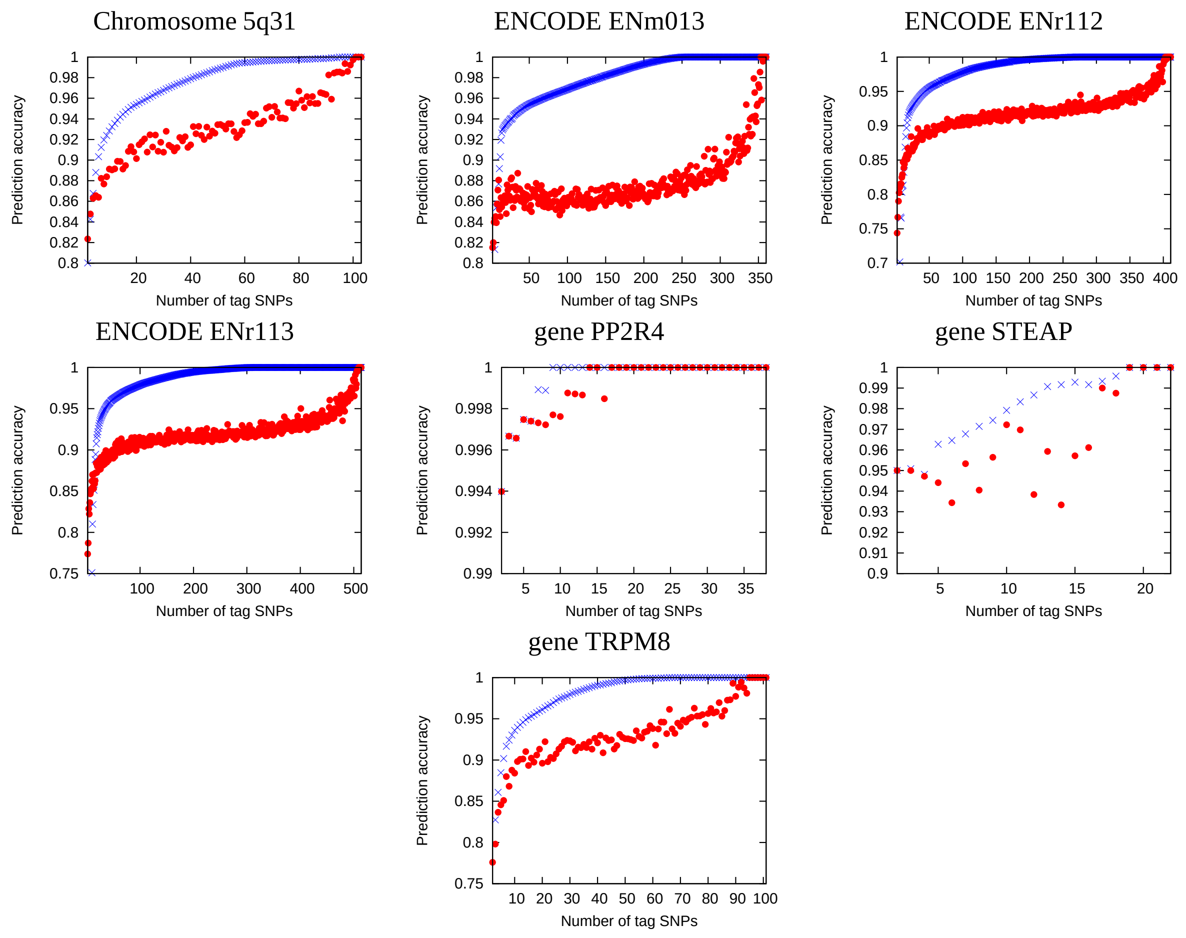

Fig. 2. Prediction accuracy as a function of the number of tag SNPs used in the two selection algorithms. Blue X - STAMPA, red circles - random sampling algorithm.

In order to evaluate the prediction accuracy of a tag SNP selection algorithm, one has to provide a prediction algorithm such as Predict. Unfortunately, ldSelect and HapBlock do not provide a prediction algorithm. Hence, in order to evaluate the prediction accuracy of these algorithms, we had to choose a prediction algorithm for each of them.

ldSelect requires as input phased genotypes. We used PHASE [15] to obtain the phasing solution, since it is a widely used and highly accurate phasing program (see, e.g., [10]). The output of the program is sets of SNPs and for each one a subset of its tag SNPs. SNPs in a set are not necessarily contiguous. We used a majority vote of the tag SNPs inside each set as the prediction method of SNPs in this set. (This rule is equivalent to that of Predict in the case of two tag SNPs, with the key difference that Predict assumes a specific order of the two tag SNPs and the predicted one.)

HapBlock gets as input a genotype matrix and outputs the tag SNPs. There are several input parameters for this software, such as the algorithm for block partitioning and the method of tag SNP selection. Additional numeric parameters are required, e.g., a threshold for common haplotypes. We used the default values presented in the software manual [16]. Since the input to this program is unphased genotypes, and no prediction algorithm was suggested, we used our own prediction algorithm (Section 3) to measure the accuracy of tag SNPs chosen by the algorithm.

In Table 1 and in Figure 3 we give a summary of the comparison of STAMPA to ldSelect. In each of the methods, we searched for the minimal number of tag SNPs needed in order to reach accuracy of at least 80% and 90%, respectively. Since the input format of ldSelect does not allow specifying the number of tag SNPs, but rather the Pearson correlation value between the tag SNPs and the predicted SNPs, we searched for the minimal Pearson correlation value needed in order to reach 80% (or 90%) accuracy. Reducing the value of the Pearson correlation results in a smaller number of tag SNPs. Our experiments show that STAMPA consistently outperforms ldSelect.

Data set	80% accuracy		90% accuracy		Total number of SNPs
	STAMPA	ldSelect	STAMPA	ldSelect	
5q31	2	64	6	91	103
Gabriel et al.	3.4 (1.8)	41.6 (14.8)	12.1 (6.3)	51 (17.8)	55.6 (20.2)
ENm013	5	84	12	189	360
ENr112	9	97	17	169	411
ENr113	11	83	18	325	514
PP2R4	2	6	2	6	38
STEAP	2	20	2	22	22
TRPM8	3	38	6	53	101

Table 1. Performance of STAMPA and ldSelect. The number of tag SNPs needed in order to reach accuracy of 80% and 90% by each algorithm is listed. For the data of Gabriel et al., the first number is the average over all 51 datasets, followed by the standard deviation in parentheses. See Figure 3 for more detailed results on these datasets.

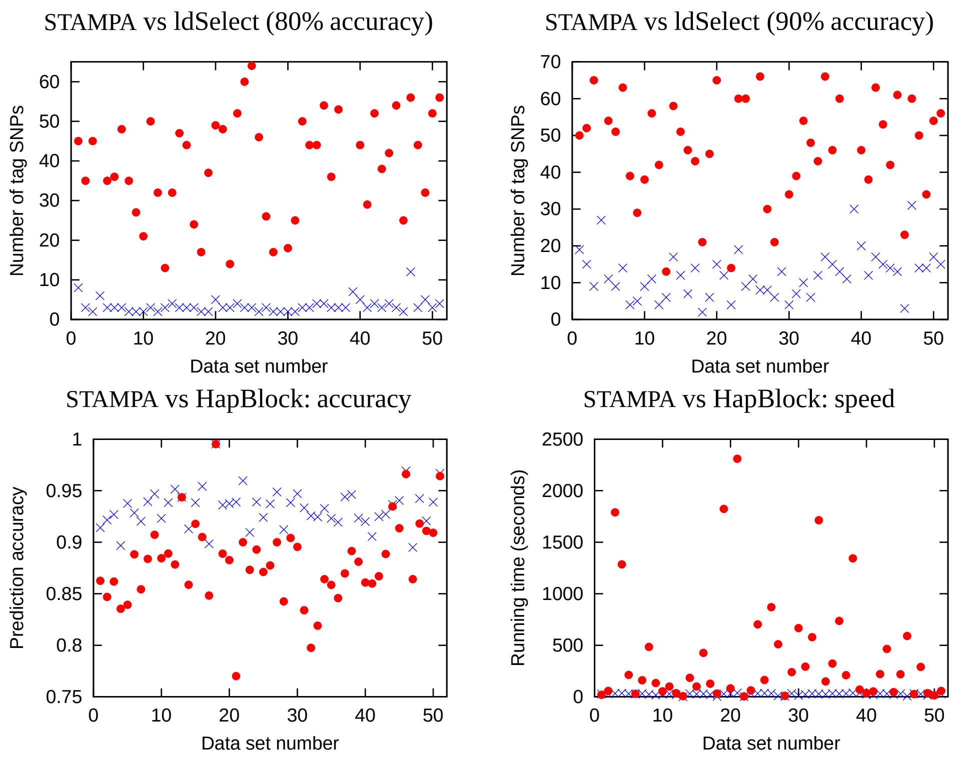

Fig. 3. Performance of STAMPA, ldSelect and HapBlock on each of the 51 genotyped regions in Gabriel et al. [7]. The x-axis is the 51 datasets in arbitrary order; blue X - STAMPA, red circles - the other algorithm. Top: comparison to ldSelect: the number of tag SNPs found by the algorithm to reach accuracy of 80% (left) and 90% (right). Bottom: comparison to HapBlock: prediction accuracy (left) and running times (right) of the algorithms on each data set.

On the average ldSelect uses ten times more tag SNPs than STAMPA in order to reach accuracy of 90%.

In Table 2 and in Figure 3 we give a summary of the comparison of STAMPA and HapBlock . We used the same number of tag SNPs generated by HapBlock to select tag SNPs with STAMPA. In all of the 58 data sets STAMPA was more accurate. Moreover, the running time of STAMPA was much smaller than HapBlock. For example, on chromosome 5q31 data set, STAMPA was faster by a factor of 97. Such advantage will be more prominent on future larger data sets.

Data set	Number of tag SNPs	Prediction accuracy		Running times (seconds)	
		STAMPA	HapBlock	STAMPA	HapBlock
5q31	17	0.949	0.889	179	17,311
ENm013	15	0.929	0.759	78	8,710
ENr112	33	0.939	0.822	87	3,810
STEAP	3	0.951	0.763	3	5
TRPM8	12	0.942	0.811	34	140
Gabriel et al.	16.9 (6.5)	0.932 (0.019)	0.88 (0.04)	1,282	20,131

Table 2. Prediction accuracy and running times of STAMPA and HapBlock. The number of tag SNPs is determined according the output of HapBlock software, using its default parameters. No comparison could be performed on ENr113 since HapBlock gave no solution due to memory overload. The gene PP2R4 was dropped since HapBlock outputs only one tag SNP for that gene, so comparison was meaningless. For the data of Gabriel et al., the fi rst number is the average over all 51 datasets, followed by the standard deviation in parentheses; running times are totals over all 51 datasets. See Figure 3 for more detailed results on these datasets.

6 DISCUSSION

In this paper we have defined a novel measure for evaluating the quality of tag SNP selection. The measure we use, *prediction accuracy*, has a very simple and intuitive meaning: It aims to maximize the expected accuracy of predicting untyped SNPs, given the unphased (genotype) information of the tag SNPs. The prediction itself is done using a simple majority vote. By making an additional natural approximate assumption that SNP values can be determined best based on the values of their nearest tag SNPs on each side, the prediction becomes quite simple, and the optimal selection of tag SNPs can be done in polynomial time.

We presented a method for tag SNPs selection and for SNP prediction based on the genotype values of the tag SNPs. Our selection method, called STAMPA, is unique in its treatment of the prediction part. Most extant methods for tag SNP selection (e.g., [18, 1, 2, 4, 14]) rely on haplotype information that is often not readily available in real life scenarios. One exception is the HapBlock algorithm [19], which selects the tag SNPs based on the genotypes and not on the haplotypes. However, HapBlock selects the tag SNPs in order to maximize diversity of the common haplotypes in blocks, and it is not clear whether that method could be easily extended to a SNP prediction algorithm using genotype data for the tag SNPs.

Another difference between STAMPA and HapBlock is in the use of phasing: Although both methods employ the dynamic programming approach, HapBlock solves many phasing subproblems in the dynamic programming recursion, determines the blocks and selects the tag SNPs in each block. In contrast, STAMPA uses phased data for the training set and then employs only the much simpler and faster prediction algorithm in the recursion. This is the reason the latter algorithm is much faster.

We presented two tag SNP selection algorithms, one based on dynamic programming and the other based on random sampling. The dynamic programming algorithm guarantees an optimal solution in polynomial time, but may be prohibitively slow in practice when the number of tag SNPs is large. A practical compromise that we used is to limit the distance between neighboring tag SNPs. Under this restriction optimality is not guaranteed anymore, but our results using over 50 different genotype sets show that accuracy is very good in most cases even with a modest distance bound ($c = 30$). The distance-bounded dynamic programming approach usually provides better results than the random sampling approach. These findings are consistent with the report in [19], where a different criterion (power) was used to evaluate random sampling and HapBlock performance on simulated data. On the other hand, the random sampling algorithm is very efficient, and therefore we believe that it may be useful in some specific situations, e.g., on large data sets where a very sparse set of tag SNPs is sought.

In comparison to another tag SNP selection algorithm, ldSelect [4], STAMPA consistently obtained higher accuracy. This is not surprising, since ldSelect uses a simple greedy approach. Interestingly, even the random sampling approach outperformed ldSelect (results not shown). ldSelect has the added flexibility to select tag SNPs for non-contiguous sets of SNPs, and thus may have an advantage over STAMPA in the cases where the LD does not decay with distance.

What is the best measure for selecting tag SNPs? The answer is not clear yet, and also depends on the context. We propose here the expected prediction accuracy, and show that under reasonable assumptions it yields an efficient and accurate method for selection. Still, other criteria have been proposed. If the ultimate goal is to detect disease association, the power of a selection method may be evaluated using this criterion. We intend to explore the power of STAMPA in disease association in the future. Another objective may be to maximize the distinction between common haplotypes in blocks. STAMPA does not provide common haplotypes and does not assume any block structure, which simplifies the algorithmics but may be viewed as a disadvantage. Our work shows that if the expected number of errors is of interest, then our algorithms provide more accurate prediction compared to existing algorithms.

7 ACKNOWLEDGMENT

RS holds the Raymond and Beverly Sackler Chair in Bioinformatics at Tel Aviv University, and was supported also in part by the Israeli Science Foundation (grant 309/02).

REFERENCES

[1]H. I. Avi-Itzhak, X. Su, and F. M. De La Vega. Selection of minimum subsets of single nucleotide polymorphisms to capture haplotype block diversity. In *Proceedings of Pacific Symposium on Biocomputing (PSB 03)*, volume 8, pages 466–477, 2003.

[2]V. Bafna, B. V. Halldorsson, R. Schwartz, A. Clark, and S. Istrail. Haplotypes and informative SNP selection algorithms: Don't block out information. In *Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology (RECOMB 03)*, pages 19–27. The Association for Computing Machinery, 2003.

[3]D. Botstein and N. Risch. Discovering genotypes underlying human phenotypes: past successes for Mendelian disease, future approaches for complex disease. *Nature Genetics*, 33:228–237, 2003.

[4]C. S. Carlson, M. A. Eberle, M.J. Rieder, Q. Yi, L. Kruglyak, and D. A. Nickerson. Selecting a maximally informative set of single-nucleotide polymorphisms for association analysis using linkage disequilibrium. *American Journal of Human Genetics*, 74(1):106–120, 2004.

[5]M. J. Daly, J. D. Rioux, S. F. Schaffner, T. J. Hudson, and E. S. Lander. High-resolution haplotype structure in the human genome. *Nature Genetics*, 29(2):229–232, 2001.

[6]E. Eskin, E. Halperin, and R. M. Karp. Large scale reconstruction of haplotypes from genotype data. In *Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology (RECOMB 03)*, pages 104–113. The Association for Computing Machinery, 2003.

[7]S. B. Gabriel, S.F. Schaffner, H. Nguyen, J. M. Moore, J. Roy, B. Blumenstiel, J. Higgins, M. DeFelice, A. Lochner, M. Faggart, S. N. Liu-Cordero, C. Rotimi, A. Adeyemo, R. Cooper, R. Ward, E. S. Lander, M. J. Daly, and D. Altshuler. The structure of haplotype blocks in the human genome. *Science*, 296:2225–2229, 2002.

[8]G. Greenspan and D. Geiger. Model-based inference of haplotype block variation. In *Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology (RECOMB 03)*, pages 131–137. The Association for Computing Machinery, 2003.

[9]G. Kimmel and R. Shamir. Maximum likelihood resolution of multi-block genotypes. In *Proceedings of the Eighth Annual International Conference on Research in Computational Molecular Biology (RECOMB 04)*, pages 2–9. The Association for Computing Machinery, 2004.

[10]G. Kimmel and R. Shamir. GERBIL: Genotype resolution and block identifi cation using likelihood. *Proceedings of the National Academy of Sciences of the United States of Ameirca (PNAS)*, 102:158–162, 2005.

[11]L. Kruglyak and D. A. Nickerson. Variation is the spice of life. *Nature Genetics*, 27:234–236, 2001.

[12]E. R. Martin, E. H. Lai, J. R. Gilbert, A. R. Rogala, A. J. Afshari, J. Riley, K. L. Finch, J. F. Stevens, K. J. Livak, B. D. Slotterbeck, S. H. Slifer, L. L. Warren, P. M. Conneally, D. E. Schmechel, I. Purvis, M. A. Pericak-Vance, A. D. Roses, and J. M. Vance. SNPing away at complex diseases: analysis of single-nucleotide polymorphisms around APOE in Alzheimer's disease. *American Journal of Human Genetics*, 67:383–394, 2000.

[13]R. W. Morris and N. L. Kaplan. On the advantage of haplotype analysis in the presence of multiple disease susceptibility alleles. *Genetic Epidemiology*, 23:221–233, 2002.

[14]I. Pe'er, P. Bakker, J. C. Barret, D. Altshuler, and M. J. Daly. A branch and bound algorithm for the chromosome tagging problem. In *Proceedings of the Second RECOMB Satellite Workshop on Computational Methods for SNPs and Haplotypes*, pages 89–98, 2004.

[15]M. Stephens and P. Donnelly. A comparison of Bayesian methods for haplotype reconstruction from population genotype data. *American Journal of Human Genetics*, 73(6):1162–1169, 2003.

[16]`http://www.cmb.usc.edu/˜msms/HapBlock`.

[17]`http://www.hapmap.org/`.

[18]K. Zhang, M. Deng, T. Chen, M.S. Waterman, and F. Sun. A dynamic programming algorithm for haplotype block partitioning. *Proc. Natl. Acad. Sci. USA*, 99(11):7335–9, 2002.

[19]K. Zhang, Z. Qin, J. Liu, T. Chen, M. S. Waterman, and F. Sun. Haplotype block partitioning and tag SNP selection using genotype data and their applications to association studies. *Genome Research*, 14(5):908–916, 2004.

[20]K. Zhang, F. Sun, M. S. Waterman, and T. Chen. Dynamic programming algorithms for haplotype block partitioning: applications to human chromosome 21 haplotype data. In *Proceedings of The Seventh Annual International Conference on Research in Computational Molecular Biology (RECOMB 03)*, pages 332–340. The Association for Computing Machinery, 2003.