Two-Locus Association Mapping in Subquadratic Time

Panagiotis Achlioptas
Department of Computer Science
University of Crete, Greece
achliopt@csd.uoc.gr

Bernhard Schölkopf
Department of Empirical Inference
MPI for Intelligent Systems
Tübingen, Germany
bs@tuebingen.mpg.de

Karsten M. Borgwardt
Machine Learning and Computational Biology Research Group
MPIs Tübingen, Germany
karsten@tuebingen.mpg.de

ABSTRACT

Genome-wide association studies (GWAS) have not been able to discover strong associations between many complex human diseases and single genetic loci. Mapping these phenotypes to pairs of genetic loci is hindered by the huge number of candidates leading to enormous computational and statistical problems. In GWAS on single nucleotide polymorphisms (SNPs), one has to consider in the order of 10^{10} to 10^{14} pairs, which is infeasible in practice. In this article, we give the first algorithm for 2-locus genome-wide association studies that is subquadratic in the number, n, of SNPs. The running time of our algorithm is data-dependent, but large experiments over real genomic data suggest that it scales empirically as $n^{3/2}$. As a result, our algorithm can easily cope with $n \sim 10^7$, i.e., it can efficiently search all pairs of SNPs in the human genome.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining

General Terms

Algorithms, Experimentation, Measurement

Keywords

Epistasis detection, two-locus association mapping, lightbulb algorithm

1. INTRODUCTION

A central question in genetics is whether natural variation in phenotypes can be explained by differences between individuals on the genomic level. Advances in sequencing technology allow us now to study these genomic differences at an unprecedented level of detail. Particularly in humans, millions of single nucleotide polymorphisms (SNPs) have been determined [5], that is positions in the genome where one nucleotide varies between different individuals.

KDD'11, August 21–24, 2011, San Diego, California, USA.

Current genome-wide association studies (GWAS) tend to focus on single loci or single SNPs that are associated with a phenotype. GWAS that do consider groups of loci usually assume an additive effect of these loci. Interactive effects, even between pairs of loci, are typically not searched for. Such an *epistatic* interaction effect occurs if the average phenotypic value of the two-locus genotypes deviates from that expected by summing the allelic effects of the two loci [6]. Such gene $\times$ gene interactions are widely believed and acknowledged to be a major factor in many complex traits such as many humans diseases [12]. For instance, the presence of gene $\times$ gene interactions has been demonstrated in type 1 diabetes [7], type 2 diabetes [8], inflammatory bowel diseases [4] and prostate and breast cancer [18, 1]. Why does one then not routinely scan for pairs of interacting SNPs? The first, computational reason is that the number of pairs of SNPs is extremely large, in the order of 10^{10} to 10^{14}, and is hence not amenable to a straightforward solution by exhaustive enumeration. Testing each of these pairs for association with the phenotype leads directly to the second, the statistical reason. When performing billions of tests, one has to correct for multiple hypothesis testing, in order to reduce the number of false positives, which in turn may lead to a loss in power.

Our focus in this article is on the first of these two reasons, the computational burden caused by the gigantic number of candidate pairs in two-locus mapping. We present an algorithm for efficiently retrieving the top h scoring pairs among all pairs of SNPs, assuming binary phenotypes and the difference-in-correlation [9] as the association criterion. Unlike earlier approaches to this problem, the runtime of our algorithm is *subquadratic in the number of SNPs*.

There are various ways of defining measures of statistical dependence between a pair of SNPs and a binary phenotype. One common strategy is to use a logistic regression model between a SNP pair and the phenotype and to check whether the interaction term coefficient differs significantly from zero. Another measure is the difference in correlation, that is we search for the SNPs whose correlation maximally differs between cases and controls [9]. The latter strategy is the one for which we develop a subquadratic algorithm here.

Previous work on two-locus association mapping can be broadly divided into the following four categories.

First, there are approaches that exhaustively consider all SNP pairs, for instance in linear or logistic regression, or multifactorial dimension reduction [15]. Due to the large number of candidate pairs, these studies usually only consider tens to hundreds of SNPs.

Second, one may proceed in a two-step procedure, in which one first performs a one-locus association study, and then performs an exhaustive pairwise search between the most significant SNPs from the one-locus mapping [11, 21].

Third, there are approaches, such as genetic algorithms [13, 17], which efficiently return local optima of the function that scores the importance of SNP pairs, but they are incomplete in the sense that they do not necessarily find the best pairs.

Fourth, there are recently published data mining methods such as FastAnova [20] and TEAM [19], which are able to prune a fraction of the computations, but their running time still scales as $\Omega(n^2 m)$, where n is the number of SNPs and m is the number of individuals in the sample. To the best of our knowledge, our method is the first one which overcomes the runtime bottleneck of 2-Locus GWAS of being quadratic in the number of SNPs.

The remainder of this article is organized as follows. In Section 2, we formalize our optimization problem and give an overview of our strategy. In Section 3, we describe the algorithm of [14] for finding the most correlated pair in a set of binary variables. We then present a similar algorithm for the least correlated pair, and, based on that, an algorithm for the pair of variables with maximum difference in correlation between two different sets of measurements (see Section 5). Next, we describe how to extend this last algorithm to real-valued variables and to Pearson's correlation coefficient measure (Section 6), and discuss strategies on how to set the parameters of this approach (Sections 7.1 and 7.2). Lastly, we evaluate performance in a series of experiments on synthetic and real-world data (Section 8).

2. PROBLEM STATEMENT

The goal of 2-locus genome-wide association studies (2L-GWAS) is to find *pairs* of Single Nucleotide Polymorphisms (SNPs) that jointly determine a phenotypical trait. As common in clinical genetics, we assume that traits are binary.

We assume that we are given an $(m_1 + m_2) \times n$ matrix D of $m_1 + m_2$ patients with n SNPs, where each SNP takes values in $\{0, 1, 2\}$. This matrix consists of two submatrices A and B, of size $m_1 \times n$ and $m_2 \times n$, respectively. A is the set of m_1 cases with phenotype 1, B the set of m_2 controls with phenotype 0. Each SNP is a $(m_1 + m_2) \times 1$ column vector of D. We seek pairs of SNPs whose statistical dependence with the phenotype is maximally different between cases and controls.

Let $P : \{0, 1, 2\}^m \times \{0, 1, 2\}^m \to \mathbb{R}$ be a measure of correlation between two vectors (SNPs). Given a matrix C, let $P_C(i, j)$ denote the value of P with input the columns i, j. Our optimization problem is then to find the pair

$$\arg\max_{i,j} |P_A(i, j) - P_B(i, j)|. \tag{1}$$

Assuming that P can be computed in linear time, as is always the case in practice, the problem can be trivially solved in time $O(n^2(m_1 + m_2))$. Without any assumptions on the function P it is not clear that one can do much better. Unfortunately, such a running time greatly limits the size of SNP collections that can be studied in practice. For example, we had to use a cluster of one thousand CPUs for several days in order to solve the problem for 850k SNPs to achieve a ground truth for our experiments.

Our starting point is the seminal work of Paturi et al. for the "Light Bulb Problem" [14], showing that for binary matrices, i.e., with elements in $\{0, 1\}$, one can find the nearest pair of columns in L_1 distance in time $O(n^{1+\gamma})$, where $\gamma < 1$ depends on the difference in distance between the closest and second closest pairs. We first show that in this same setting it is possible to find the pair described in equation (1), with guarantees on time and recall similar to those of [14]. To do this, we exploit heavily both the fact that the data is binary and the linearity of the L_1-distance. Both of these assumptions, though, are not ideal for SNP data, where data is typically ternary, i.e., the entries of D are in $\{0, 1, 2\}$, and a commonly used P is Pearson's correlation coefficient [9]. To address these issues we employ Locality Sensitive Hashing (LSH) to transform the original SNP data into binary data which preserve the original angles. While this transformation, in the worst case, can somewhat distort the Pearson correlation of two SNPs, we show that a modification of our algorithm completely eliminates this problem, without incurring a significant computational penalty.

3. THE MOST CORRELATED PAIR

We first discuss how to solve the problem captured by (1) when the matrices of interest are binary and the correlation P of two columns in D (that is, of two SNPs) is defined as the fraction of rows in which they take the same value, i.e. it is equal to 1 minus their L_1 distance divided by their dimension. The binary assumption is valid for SNPs in inbred species, as all their SNPs are homozygous and can be represented by a two-state variable, but this definition of correlation, as far as we know, has not been used in biological studies yet.

Given a binary matrix D, let $P_D(i, j)$ denote the correlation of column pair $\{i, j\}$ in D, i.e., the fraction of rows in which columns i and j take the same value. Let p_1 and p_2 denote the highest (greatest) and second highest correlations present across all pairs. In a breakthrough 1989 paper, Paturi, Rajasekaran, and Reif [14] gave a subquadratic algorithm for finding

$$\arg\max_{i,j} P_D(i, j).$$

Specifically, the running time of their algorithm is $\tilde{O}(n^{1+\gamma})$, where $\gamma = \log p_1 / \log p_2 < 1$ and the $\tilde{O}(\cdot)$ notation suppresses $(\log n)^q$ factors for any finite $q > 0$. Since the results of [14] are the starting ground for our work, we begin with a few words about how and why the algorithm in [14] works. To simplify the exposition we assume, for now, that p_1, p_2 are given to us as part of the input. We will later see how this assumption can be removed. The algorithm uses a sample-size parameter $k = k(n, \gamma) = O(\log n)$. The choice of k is discussed later.

Algorithm CO

1. Appropriately set the sample-size parameter k.
2. Let $t = n^\gamma \log n$, where $\gamma = \log p_1 / \log p_2$.
3. To succeed with probability $1 - 1/\text{poly}(n)$, repeat t times:
   (a) Select k distinct rows of D uniformly at random, to form a $k \times n$ matrix R.
   (b) Partition the columns of R into (no more than 2^k) equivalence classes $E_1, E_2, \ldots, E_\ell$.

(c) For every pair of columns $\{i, j\}$ that end up in the same equivalence class, increment (after perhaps creating) the corresponding counter C_{ij} by 1.

4. Report the pair of columns with maximum counter.

Intuitively, the parameter k is chosen so that we expect to increase a linear number of counters in each round, i.e.,

$$\mathbb{E}\left[\sum_{i=1}^{\ell}\binom{|E_i|}{2}\right] = \Theta(n) \ .$$

The parameter t is chosen so as to ensure that with probability $1 - 1/\text{poly}(n)$ after t iterations the most correlated pair has increased its associated counter more times than the second most correlated pair.

4. THE LEAST CORRELATED PAIR

Our first step is to give a subquadratic algorithm for finding

$$\arg\min_{i,j} P_D(i, j). \qquad (2)$$

We will need the following definition.

Definition 1. *Given a matrix D, let $\overline{D}$ denote the matrix that results by complementing its elements.*

Given a matrix D with m rows and n columns let p_1 and p_2 denote the lowest (smallest) and second lowest correlations present in D. The running time of our algorithm is $\tilde{O}(n^{1+\gamma})$, where $\gamma = \log(1-p_1)/\log(1-p_2)$. Our algorithm for finding its least correlated pair of columns is

Algorithm AC

1. Let U be the $m \times 2n$ matrix that results by concatenating D and $\overline{D}$ row-wise.

2. Find the most correlated pair of columns in U among those pairs of columns in which exactly one of the two columns comes from each of $D, \overline{D}$.

The correctness of algorithm AC comes from observing that for every pair of columns i, j in U in which one column comes from D and one from $\overline{D}$ we have $P_U(i, j) = 1 - P_D(i, j)$.

The crucial observation that makes the above interesting is the following.

Observation 1. *Step 2 of AC can be carried out using a small modification of CO. Namely, each time we are about to increment a counter C_{ij} in CO we check whether exactly one of each of i, j come from matrices $D, \overline{D}$.*

5. THE MOST DIFFERENT PAIR

The ideas underlying our algorithm for finding the least correlated pair can be adapted to solve the problem of finding

$$\arg\max_{i,j} |P_A(i, j) - P_B(i, j)| \qquad (3)$$

for any two $m \times n$ binary matrices A, B. Note that now the two matrices are required to not only have the same number of columns, but also the same number of rows ($m_1 = m_2 = m$). We later discuss how to avoid this requirement by employing randomization. Our main tool is the following.

Lemma 1. *For two $m \times n$ matrices A, B let*

$$\{x, y\} = \arg\max_{i,j} P_A(i, j) - P_B(i, j) \ .$$

Let S be an arbitrary subset of $\{1, 2, \ldots, n\}$ and let B' be the matrix that results by complementing all columns of B that belong in S. Let C be the $2m \times n$ matrix that results by concatenating A and B', column-wise. Let $\mathcal{P} = \mathcal{P}(S)$ consist of those pairs of columns of C that have exactly one element from S. If $\{x, y\} \in \mathcal{P}$, then

$$\{x, y\} = \arg\max_{i,j \in \mathcal{P}} P_C(i, j) \ .$$

Proof of Lemma 1. We will first do some calculations which allow for matrices A and B to have m_1 and m_2 rows, respectively, where m_1 and m_2 can be different. Consider any two pairs of columns $S, T \in \mathcal{P}$. Assume that the columns in pair S agree on s_1 rows of A and s_2 rows of B. Similarly, assume that the columns in pair T agree on t_1 rows of A and t_2 rows of B.

In C, the correlation of the columns in S is

$$\frac{s_1 + m_2 - s_2}{m_1 + m_2} = \frac{s_1 - s_2}{m_1 + m_2} + \frac{m_2}{m_1 + m_2}$$

while the correlation of the columns in T is

$$\frac{t_1 + m_2 - t_2}{m_1 + m_2} = \frac{t_1 - t_2}{m_1 + m_2} + \frac{m_2}{m_1 + m_2} \ .$$

If $m_1 = m_2 = m$, we see that ordering the elements of $\mathcal{P}$ by their correlation in matrix C is the same as ordering them by their correlation gap in matrices A, B. In fact, the two are related by an affine transformation. $\square$

In order to ensure that when applying Lemma 1 the sought-after pair of columns $\{x, y\}$ is considered, we could try to take different random samples of columns S from C. Alternatively, we can simply consider a twice as large matrix

$$D = \begin{pmatrix} A & A \\ B & \overline{B} \end{pmatrix}$$

and let $\mathcal{P}$ consist of those pairs of columns in D that have exactly one column from amongst the first n. Then

$$\{x, y\} = \arg\max_{i,j \in \mathcal{P}} P_D(i, j) \ .$$

Thus, by applying algorithm CO to D we will find the pair maximizing $P_A(i, j) - P_B(i, j)$. Reversing the roles of A, B, i.e., applying CO to the matrix

$$E = \begin{pmatrix} A & \overline{A} \\ B & B \end{pmatrix}$$

we find the pair maximizing $P_B(i, j) - P_A(i, j)$. A single comparison then gives us the maximizer of $|P_A(i, j) - P_B(i, j)|$.

5.1 Unequal size matrices

When the two matrices A, B have $m_1 \neq m_2$ rows, respectively, the proof of Lemma 1 fails. In particular, if $m_1 \gg m_2$ then, complemented or not, the elements of B have small influence in the correlation of columns in C.

To overcome this problem we "enrich" the smaller of the two matrices by repeating some of its rows. Specifically, if $m_1/m_2 > 1$, we append $l = \lfloor m_1/m_2 \rfloor - 1$ identical copies of B and a random sample of $m_1 - (l+1)m_2$ rows of B. If we are lucky and m_1/m_2 is integer, then it is clear that every column pair has exactly the same correlation in the resulting matrix as it did in B. If m_1/m_2 is not integer, correlations are only preserved in expectation. Nevertheless, the associated variance is reasonable as the fraction of random rows in the resulting matrix is always less than $1/2$. Finally, in the more interesting biological setting where we use LSH functions, we show how to completely overcome this issue.

6. NON-BINARY DATA

So far, our algorithm applies to binary data and a correlation measure which is the probability of two binary variables being identical. Next, we will show how these ideas can be expanded to handle arbitrary real-valued vectors and Pearson's correlation coefficient. For this, we need to introduce the Locality Sensitive Hash functions (LSH) of [3].

To hash a collection of vectors in $\mathbb{R}^m$, we choose a spherically random vector $\vec{r} \in \mathbb{R}^m$ (by taking its coordinates to be m independent $N(0,1)$ random variables) and define a hash function $h_{\vec{r}}$ as follows:

$$h_{\vec{r}}(\vec{u}) = \begin{cases} 1 \text{ if } \vec{r} \cdot \vec{u} \geq 0 \\ 0 \text{ if } \vec{r} \cdot \vec{u} < 0 \end{cases} \tag{4}$$

FACT 1. *For any pair of vectors $\vec{v}, \vec{u}$, $Pr[h_r(\vec{u}) = h_r(\vec{v})] = 1 - \theta(\vec{u}, \vec{v})/\pi$, where θ is the angle between the two vectors.*

Fact 1 states that the probability that a uniformly random hyperplane separates two vectors is proportional to their angle. Thus, the smaller the angle, the larger the probability that two vectors will hash to the same value. If we hash our input vectors with s independent random hyperplanes, each input vector becomes a (hash) vector in $\{0,1\}^s$, and the expected L1 distance of two hash vectors is proportional to the angle between the corresponding input vectors. It is clear that as s grows, this (random) L1 distance becomes a better and better estimator of the angle between the original vectors. Therefore, the algorithms developed in the previous sections applied on the hash vectors, are randomized algorithms for finding the pairs with largest, smallest and "most different" angles.

To handle Pearson's correlation coefficient we exploit its close relation with the cosine function and, therefore, with the angle between vector pairs.

6.1 Pearson's correlation coefficient

Pearson's correlation coefficient of two random variables u, v is equal to

$$\rho = \frac{cov(v,u)}{\sigma_v \sigma_u} \ , \tag{5}$$

i.e., their covariance divided by the product of their standard deviations. Given d samples of each random variable, organized as d-dimensional vectors $\vec{u}, \vec{v}$, an equivalent definition (see [16]) is:

$$\rho = \cos(\vec{v} - \bar{v}, \vec{u} - \bar{u}), \tag{6}$$

where $\bar{u} = \mathbf{1}\mu_u$ and $\bar{v} = \mathbf{1}\mu_v$, where μ_u and μ_v are the mean values of $\vec{u}$ and $\vec{v}$, respectively.

In view of (6), we see that when seeking the pair of SNPs that are "most different" in terms of Pearson's correlation coefficient between cases and controls, we are simply seeking the pair of SNP vectors whose cosine is most different. We approximate this cosine by the angle of the vectors $\vec{v} - \bar{v}$ and $\vec{u} - \bar{u}$, i.e., we apply the methods of the previous section to them. The reason this works is as follows.

When a pair of vectors has different cosine between the two data sets it must also have some difference in angle between them. Luckily, the absolute value of the derivative of the arccos function is lower-bounded by 1 and stays close to 1 in most of $[-1, +1]$ (of course, if it equaled 1 throughout, the replacement of cosine by angle would be exact.) In particular, in $[-0.85, 0.85]$ this derivative stays bounded by 2, meaning that for pairs whose cosine in both data sets is in this range, their probability of being placed in different buckets is distorted by no more than a factor of 2.

Of course, a factor of 2 would still be a huge systematic error in our context, but there is a simple remedy. Namely, we change the role of the CO algorithm and its variants: instead of using them as methods to find directly the top-scoring pairs by keeping track of the (biased) associated counters we use them as filtering algorithms that simply propose "potentially interesting" pairs to examine. Specifically, for every pair co-occurrence, instead of incrementing a counter and ultimately evaluate pairs by the value of their counters, we measure and evaluate directly the exact Pearson's correlation coefficient "on the fly". (To speed things up one can use a bit-vector to make sure they only measure each pair at most once. As we will see, this is hardly necessary as the majority of pairs that ever co-occur, do so exactly once.)

Finally, the a priori valid concern regarding pairs of SNPs whose cosine might be very close to ± 1 (and for which, thus, the distortion resulting from replacing cosines by angles could be arbitrary) is not borne out by the data. The reason is that the "most different" pairs end up having a difference in cosine of at least 0.7 which puts their angle difference very well within the near-linear regime of the arccos function.

6.2 Number of hyperplanes used

In order to binarize the SNP data via LSH, we map both the case vectors and the control vectors into $\{0,1\}^s$ using s independent random vectors for each. Clearly, the more hyperplanes we use, the better the approximation of the correlation of the SNPs in our dataset. While, theoretically, $s = \Theta(\log n)$ suffices, in practice the correct number of hyperplanes to use is domain-dependent and has to do with parameters like the initial dimension of the vectors and their distribution. As a rule of thumb in our experiments, since the number of individuals was rather small, we took $s = \max\{m_1, m_2\}$. Note now that since the binarized data for cases and controls are of the same dimension, we do not run into the class-size imbalance problem mentioned in Section 5.1.

7. PRACTICAL CONSIDERATIONS

7.1 Running CO in practice

Since p_1 and p_2 are not known a priori, in order to run the algorithm one must replace γ by an estimate $\hat{\gamma}$. In that case, CO is guaranteed to return a pair with correlation strictly larger than $\kappa = p_1(1 - 1/\hat{\gamma})$. (So, it will return the most correlated pair, when $p_2 \leq \kappa$.) In practice, one can run the algorithm multiple times with $\hat{\gamma}$ and check whether the returned pair changes. If that happens, then the estimate for γ needs to be increased. Note that since the exponent will increase, the running time expended prior to the increase is completely dominated (and can therefore be absorbed) in the new exponent. As we will see in Section 7.2, though, in practice there seem to be much more effective ways to address this issue that avoid estimating γ altogether.

7.2 Running AC in practice

As can be easily seen, the modified version of the CO algorithm from Section 3 is at the heart of our approach. To implement it efficiently, in each iteration, after we partition the columns into equivalence classes (buckets), we further partition the columns in each bucket into two equivalence classes, depending on if they were complemented or not. We then increment the counters of all pairs going across the two partitions. In this way, the additional running time after we partition the columns into buckets E_ℓ is proportional only to the number of counters we actually increment, not $\sum E_\ell^2$.

We now discuss how we choose our row-sample size k and how we choose the number of iterations to run. In short: we don't. That is, while one can use the (analogues of the) theoretical estimates from [14] for setting k and t these turn out to be too conservative in practice. Moreover, to overcome the need to know p_1, p_2 in order to set t, entails an additional burden to the running time as we run multiple iterations of the algorithm, effectively searching for p_1, p_2. Instead, we tune both parameters "on the fly", i.e., in a data-dependent way, using the same underlying logic as [14] but applied in a much more fine-grained manner.

7.2.1 Row-sample size

We introduce an adaptive strategy that adjusts k at the end of each iteration i of the algorithm based on the number, C_i, of SNP pairs that were examined during iteration i. In particular, we introduce a parameter called Target=$\Theta(n)$ and we adjust k with the aim that Target$/2 \leq C_i \leq 2$Target. We initialize Target to a sufficiently small value, so as not too risk too many co-occurrences in the first iteration. From then on, if $C_i <$ Target$/2$ we increment k, whereas if $C_i >$ $2 \cdot$ Target we decrement k.

As for the exact value of Target itself, while in [14], the sample size k corresponds to Target $= n$, in practice we found that setting Target to about one order of magnitude less, achieved the same quality of solutions with significantly fewer computations. We discuss this further in Section 7.3.

7.2.2 Returning top h pairs

In running CO as a subroutine for AC we have it report not just the pair of columns with maximum correlation but in fact the top-h pairs. We then measure the actual Pearson's correlation boost for these pairs and consider whether we should report them. For this, we maintain a heap of size h which we update whenever an unmeasured pair with higher correlation than the current heap minimum is discovered. In our experiments h ranges from 10 to a few thousands, making its maintenance a minor fraction of the overall running time.

7.2.3 Number of iterations

When we are interested in reporting the "top-h" pairs we keep track of the top-$2h$ pairs discovered at the end of each iteration of the algorithm. We stop at iteration t if in the last $t/2$ iterations, there was no change in the set of top-$2h$ pairs.

7.3 Runtime complexity

After centering the data (an operation linear in the size of the data) we compute for each (centered) column its inner product with $s = O(\log n)$ random vectors. At this point, we have a new binary dataset and the real work of our algorithm begins, proceeding in iterations. The work in each iteration can be partitioned into two parts. In the first part we take a random sample of $k = O(\log n)$ rows (individuals) and place them into buckets depending on their k-bit vector. In the second part, pairs that lie in the same bucket are examined. Therefore, if we examine $t \sim n^{1+\gamma}$ pairs in total during the execution of the algorithm, where $\gamma > 0$, the total number of operations by the algorithm is bounded by $O(n(s^2+n^\gamma))$. Since, typically, $\gamma \geq 1/2$, we see that the total number of examined pairs is a good proxy for the algorithm's running time. Indeed, in practice, the relationship between the number of discovered pairs and the actual running time is essentially the same across a large span of data sets, data sizes, and recall levels.

8. EXPERIMENTAL RESULTS

We have run experiments on four different data sets. These include both binary and ternary SNPs and both real and synthetic data. All experiments were performed either on a single MacPro OS X 10.6 with 16 GB RAM when this was feasible, or using a cluster of 1,000 CPUs, when we needed to scale appropriately.

8.1 Experimental setting

To avoid unfairness in comparisons that result from implementation, hardware, and programming issues, we have focused in our experiments on the key quantity that determines the asymptotic behavior of all relevant algorithms: the number of pairs whose joint statistical behavior is ever examined. So, for example, for brute force this is $\binom{n}{2}$. In our algorithm we will see that the relevant exponent is often significantly lower and gets to about $n^{3/2}$ for real-life massive data sets.

It is important to point out that the running time of our algorithm is linear in this quantity while, at the same time, as is clear by the algorithm's description, all other sampling, binning, and embedding operations are dwarfed by the step of actually measuring the joint behavior of pairs that fall in the same bucket.

Comparison partners: To quantify the quality of our results both in terms of runtime and accuracy, we compare it to i) a brute-force enumeration of all pairs and ii) a two-stage filtering procedure, in which we employ exhaustive enumeration on the subset of top-scoring SNPs from a one-locus association mapping. In order to be able to compute the brute-force approach, we ran experiments on a cluster of 1,000 CPUs, some of which took several days. This brute-force served as a reference and ground truth for evaluating the quality of the results of our algorithm.

We also evaluate the performance of our algorithm by comparing it to the common *two-stage filtering procedure*, that is applying brute force on a carefully selected subset of the SNPs. To select the SNPs we ranked them based on their correlation with class membership and retain only the top-ranking SNPs, ranging from 10% to 50% (0.1 to 0.5) of all SNPs. Then we run an exhaustive search on all pairs of SNPs from this subset.

Evaluation We compared our novel algorithm to its comparison partners based on the following criteria:

- the **speed up factor**, that is, the total speed up compared to brute force in terms of CPU time.
- the **exponent** $(1+\gamma)$ of the number of pairs our algorithm examined. Our method considers $(n^{(1+\gamma)})$ pairs compared to the n^2 pairs computed in brute-force.
- the **recall** among the top-h pairs (where $h \in \{10, 100, 500, 1k, 10k\}$) returned by our algorithm. This is the percentage of the top pairs found by brute force that were also detected by our algorithm.

8.2 Datasets

In our experiments, we used the following datasets:

Full-genome scale real human data: For 850k SNPs from real human data set (dbGap [10]) and 100 individuals in cases and controls (respectively), we used a cluster of 1000 machines for more than 3 days in order to discover by brute force the top 10k pairs as a reference.

Synthetic human data: This is a human data set (ternary SNPs) generated by the simulator Hapsample (Wright et al., 2007) which is publicly accessible from the website http://www.hapsample.org. It consists of 95k SNPs, with 200 cases and 200 controls.

Besides the original dataset we also created smaller datasets by keeping a random sample of 30k and 50k SNPs, while always maintaining all 200 cases and 200 controls for each SNP. On all these datasets we ran brute force enumeration, and the two-stage filtering approach as well.

Real binary data from USC We do as above, for 50k, 100k and 210k SNPs of the genome of a real inbred organism [2]. While our ground truth is the top 1k pairs for the first two smaller sizes of SNP collections, for the largest one is the top 10k. Here the initial number of individuals in cases and controls is 80 and 86 and we transform them (via hashing with appropriate number of random hyper planes) into matrices with 100 rows each, before we run our method.

Synthetic binary data: We also generated synthetic binary data with 200 cases and 200 controls. We generated one dataset with 10k, 50k and 100k SNPs each. The SNPs were drawn randomly from a uniform Binomial distribution. We refer to these datasets as NOISE datasets.

8.3 Results

Key to experiments: For each input, we run our algorithm with different "allowances" on the total number of pairs to measure. We report this allowance as "Pairs" in the table below. Since pairs can be suggested for measurement multiple times, we also report the total number of such "Measurements" proposed. As one can see, these two numbers are quite close, suggesting that it is probably not worthwhile to try to keep track of which pairs have already been measured. To highlight the *asymptotic* nature of our improvements, we write this number of measurements as n^{β} and report the value of β for each experiment. Finally, the remaining columns are our recall fraction relative to the ground truth as computed by Brute Force.

Comparison to brute force We ran our method on all of these four datasets and report results in the following Tables 1, 2, 4, 5, and 6. Except for the NOISE dataset, our method achieves high recall rates on all three datasets, up to 100%, and a speed up factor of 10 to 1000 compared to brute force. The runtime exponent varies for each dataset and SNP number, but on average it is approximately 3/2, compared to 2 for the brute force approach. Only on the NOISE dataset, our recall rates are low, owing to the fact that due to its random character, the correlation differences are here rather uniform across all pairs. As the performance of our algorithm depends on the discrepancy in score between the top pairs and the other pairs, it performs considerably worse on NOISE.

In making comparisons in this context, it is important to keep in mind that for any given value of h, the task of finding the top-h pairs becomes *relatively* easier as the size of the dataset, n, increases. This is because with h fixed and n increasing, the h-th highest ranked pair is more and more "special" relative to the remaining $\binom{n}{2}$ pairs as n is increased, i.e., it has higher contrast. As a result, the *relative* effort of finding it, expressed as a fraction of $\binom{n}{2}$, goes down. This is very well reflected in our experiments, where for a given recall rate at a given h, the exponent of the number of measured pairs keeps dropping as the size of the dataset increases.

Comparison to two-stage approach When we compare our approach to the common two-stage approach [11] on the hap-sample dataset for 30k, 50k and 95k SNPS (see Table 3), we observe that the recall rates of this filtering approach are lower than ours, while we still achieve a significant speed-up over its runtime.

9. DISCUSSION AND CONCLUSIONS

In this article, we have presented a first subquadratic-runtime algorithm for SNP-SNP interaction detection. It renders epistasis detection even on the whole human genome with order 10^7 SNPs [5] computationally feasible on a single PC.

In the current presentation, we have focused on reducing the computational burden caused by the quadratic number of candidate pairs. The other burden is that of determining statistical significance in multiple hypothesis testing. We can compute p-values for our difference-in-correlation test statistic based on the null distribution described in [9], and correct for multiple hypothesis testing, for instance, by Bonferroni correction. Alternatively, we can perform permutations of the phenotypes and approximate the null distribution empirically. In future work, we will examine how to perform this permutation testing more efficiently than just iteratively applying our current algorithm.

Our plan for future work on the biological side is to apply the here described algorithm to the Wellcome Trust's GWA datasets for various human diseases and to plant phenotypes from rice, maize and *Arabidopsis*. On the engineering side, we will study how to implement the here described algorithm on graphical processing units, to yield an extremely fast implementation. On the algorithmic side, we will study the exciting extension to multi-SNP-interaction detection.

# SNPs	Measurements	Exponent	Speedup	Top 10	Top 100	Top 500	Top 1k	Top 10k
845,814	660,788,168	1.48	542	1.0	0.95	0.93	0.91	0.90
845,814	919,392,776	1.51	390	1.0	0.96	0.93	0.93	0.92
845,814	1,194,631,961	1.53	300	0.9	0.94	0.89	0.88	0.88
845,814	2,052,068,774	1.57	175	1.0	0.97	0.97	0.98	0.97

Table 1: Real human dataset

# SNPs	Measurements	Pairs	Exponent	Speedup	Top 10	Top 100	Top 500	Top 1k
30,000	8,699,003	8,220,128	1.55	51	1.00	1.00	0.99	0.93
30,000	14,560,722	13,982,726	1.60	31	1.00	1.00	1.00	0.97
50,000	25,404,287	25,086,963	1.58	50	1.00	1.00	0.99	0.96
50,000	29,721,852	29,301,074	1.59	43	1.00	1.00	1.00	0.98
50,000	55,311,371	54,005,100	1.65	23	1.00	1.00	1.00	0.98
50,000	67,141,119	65,376,171	1.67	19	1.00	1.00	1.00	0.98
95,173	6,725,792	6,711,455	1.37	674	0.80	0.79	0.87	0.86
95,173	13,567,508	13,526,763	1.43	334	0.90	0.91	0.93	0.90
95,173	31,249,917	31,083,315	1.51	145	1.00	1.00	0.96	0.96
95,173	40,225,617	39,965,897	1.53	113	1.00	0.98	0.96	0.98

Table 2: Synthetic human dataset

# SNPs	Fraction kept	Exponent	Speedup	Top 10	Top 100	Top 500	Top 1k
30,000	0.10	1.49	101	0.80	0.50	0.10	0.05
30,000	0.20	1.62	26	0.80	0.61	0.27	0.14
30,000	0.30	1.70	12	0.90	0.68	0.50	0.31
50,000	0.10	1.52	101	0.90	0.46	0.11	0.06
50,000	0.20	1.64	26	0.90	0.67	0.28	0.15
50,000	0.30	1.72	12	0.90	0.70	0.55	0.36
50,000	0.40	1.77	7	0.90	0.71	0.55	0.36
50,000	0.50	1.81	5	0.90	0.71	0.57	0.48
95,173	0.10	1.54	101	0.40	0.39	0.32	0.16
95,173	0.20	1.66	26	0.40	0.40	0.52	0.38
95,173	0.30	1.73	12	0.40	0.44	0.56	0.52

Table 3: Performance of the two-stage filtering approach on the synthetic human dataset. The second column is the fraction of SNPs retained after filtering. The remaining columns are as in the experiments with our algorithm.

# SNPs	Measurements	Pairs	Exponent	Speedup	Top 10	Top 100	Top 500	Top 1k
10,000	3,102,004	2,944,313	1.62	17	1.00	0.99	0.80	0.60
10,000	3,750,058	3,496,129	1.64	14	1.00	1.00	0.82	0.67
10,000	6,236,283	5,649,158	1.70	9	1.00	1.00	0.90	0.75
50,000	33,507,669	32,325,103	1.60	38	1.00	1.00	1.00	0.99
50,000	22,940,479	22,295,748	1.57	55	1.00	0.99	0.97	0.97
100,000	8,255,645	8,186,657	1.38	606	1.00	0.86	0.82	0.80
100,000	16,556,322	16,372,904	1.44	302	0.90	0.92	0.91	0.90
100,000	39,731,329	39,039,859	1.52	126	1.00	1.00	0.98	0.98
100,000	52,762,001	51,732,700	1.54	95	1.00	1.00	0.99	0.98

Table 4: Random SNP samples from USC dataset

# SNPs	Measurements	Pairs	Exponent	Speedup	Top 10	Top 100	Top 500	Top 1k	Top 10k
210,000	18,202,620	18,062,488	1.36	1212	0.90	0.80	0.71	0.71	0.68
210,000	48,844,947	48,384,057	1.44	452	1.00	0.86	0.87	0.86	0.82
210,000	53,260,691	52,724,216	1.45	414	0.80	0.81	0.89	0.88	0.86
210,000	74,054,023	73,044,942	1.48	298	0.90	0.90	0.93	0.93	0.91
210,000	95,844,600	94,598,677	1.50	231	1.00	0.99	0.95	0.94	0.92

Table 5: Entire USC dataset

# SNPs	Measurements	Pairs	Exponent	Speedup	Top 10	Top 100	Top 500	Top 1k
10,000	3,790,269	3,651,534	1.64	14	0.30	0.26	0.20	0.19
10,000	7,619,321	7,082,458	1.72	7	0.60	0.50	0.38	0.38
50,000	15,079,241	14,960,262	1.53	83	0.10	0.12	0.07	0.07
50,000	55,200,578	53,907,259	1.65	23	0.20	0.19	0.18	0.18
100,000	27,253,865	27,143,127	1.49	184	0.00	0.07	0.05	0.04
100,000	39,759,698	39,540,714	1.52	126	0.00	0.07	0.07	0.06

Table 6: NOISE dataset (each entry is 0 or 1 independently, with equal probability).

10. ACKNOWLEDGMENTS

P.A. was visiting the Max Planck Institutes in Tübingen while this research work was done. The authors would like to thank Oliver Stegle and Nino Shervashidze for their comments on the final text. P.A. would like to thank Dimitris Achlioptas for numerous inspiring conversations.

Funding support for the Genome-Wide Association of Schizophrenia Study was provided by the National Institute of Mental Health (R01 MH67257, R01 MH59588, R01 MH59571, R01 MH59565, R01 MH59587, R01 MH60870, R01 MH59566, R01 MH59586, R01 MH61675, R01 MH60879, R01 MH81800, U01 MH46276, U01 MH46289 U01 MH46318, U01 MH79469, and U01 MH79470) and the genotyping of samples was provided through the Genetic Association Information Network (GAIN). The datasets used for the analyses described in this manuscript were obtained from the database of Genotypes and Phenotypes (dbGaP) found at http://www.ncbi.nlm.nih.gov/gap through dbGaP accession number phs000021.v3.p2. Samples and associated phenotype data for the Genome-Wide Association of Schizophrenia Study were provided by the Molecular Genetics of Schizophrenia Collaboration (PI: Pablo V. Gejman, Evanston Northwestern Healthcare (ENH) and Northwestern University, Evanston, IL, USA.

11. REFERENCES

[1] C. E. Aston, D. A. Ralph, D. P. Lalo, S. Manjeshwar, B. A. Gramling, D. C. DeFreese, A. D. West, D. E. Branam, L. F. Thompson, M. A. Craft, D. S. Mitchell, C. D. Shimasaki, J. J. Mulvihill, and E. R. Jupe. Oligogenic combinations associated with breast cancer risk in women under 53 years of age. *Human Genetics*, 116(3):208–221, Feb. 2005.

[2] S. Atwell, Y. S. Huang, B. J. VilhjÃąlmsson, G. Willems, M. Horton, Y. Li, D. Meng, A. Platt, A. M. Tarone, T. T. Hu, R. Jiang, N. W. Muliyati, X. Zhang, M. A. Amer, I. Baxter, B. Brachi, J. Chory, C. Dean, M. Debieu, J. de Meaux, J. R. Ecker, N. Faure, J. M. Kniskern, J. D. G. Jones, T. Michael, A. Nemri, F. Roux, D. E. Salt, C. Tang, M. Todesco, M. B. Traw, D. Weigel, P. Marjoram, J. O. Borevitz, J. Bergelson, and M. Nordborg. Genome-wide association study of 107 phenotypes in *Arabidopsis thaliana* inbred lines. *Nature*, 465(7298):627–631, Jun 2010.

[3] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, 2002.

[4] J. H. Cho, D. L. Nicolae, L. H. Gold, C. T. Fields, M. C. LaBuda, P. M. Rohal, M. R. Pickles, L. Qin, Y. Fu, J. S. Mann, B. S. Kirschner, E. W. Jabs, J. Weber, S. B. Hanauer, T. M. Bayless, and S. R. Brant. Identification of novel susceptibility loci for inflammatory bowel disease on chromosomes 1p, 3q, and 4q: evidence for epistasis between 1p and IBD1. *Proceedings of the National Academy of Sciences of the United States of America*, 95(13):7502–7507, June 1998.

[5] T. H. Consortium. A second generation human haplotype map of over 3.1 million SNPs. *Nature*, 449(7164):851–61, Oct. 2007.

[6] H. J. Cordell. Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans. *Hum. Mol. Genet.*, 11(20):2463–2468, Oct. 2002.

[7] H. J. Cordell. Detecting gene-gene interactions that underlie human diseases. *Nat Rev Genet*, 10(6):392–404, June 2009.

[8] N. J. Cox, M. Frigge, D. L. Nicolae, P. Concannon, C. L. Hanis, G. I. Bell, and A. Kong. Loci on chromosomes 2 (NIDDM1) and 15 interact to increase susceptibility to diabetes in mexican americans. *Nature Genetics*, 21(2):213–215, Feb. 1999.

[9] T. Kam-Thong, D. Czamara, K. Tsuda, K. Borgwardt, C. M. Lewis, A. Erhardt-Lehmann, B. Hemmer, P. Rieckmann, M. Daake, F. Weber, C. Wolf, A. Ziegler, B. Putz, F. Holsboer, B. Scholkopf, and B. Muller-Myhsok. EPIBLASTER-fast exhaustive two-locus epistasis detection strategy using graphical processing units. *Eur J Hum Genet*, Dec. 2010.

[10] M. Mailman, M. Feolo, Y. Jin, M. Kimura, K. Tryka, R. Bagoutdinov, L. Hao, A. Kiang, J. Paschall, L. Phan, N. Popova, S. Pretel, L. Ziyabari, M. Lee, Y. Shao, Z. Wang, K. Sirotkin, M. Ward, M. Kholodov, K. Zbicz, J. Beck, M. Kimelman, S. Shevelev, D. Preuss, E. Yaschenko, A. Graeff, J. Ostell, and S. Sherry. The NCBI dbGaP Database of Genotypes and Phenotypes. *Nature Genetics*, 39(10):1181–6, 2007.

[11] J. Marchini, P. Donnelly, and L. R. Cardon. Genome-wide strategies for detecting multiple loci that influence complex diseases. *Nat Genet*, 37(4):413–417, Apr. 2005.

[12] S. K. Musani, D. Shriner, N. Liu, R. Feng, C. S. Coffey, N. Yi, H. K. Tiwari, and D. B. Allison. Detection of gene x gene interactions in genome-wide association studies of human population data. *Human Heredity*, 63(2):67–84, 2007.

[13] R. Nakamichi, Y. Ukai, and H. Kishino. Detection of closely linked multiple quantitative trait loci using a genetic algorithm. *Genetics*, 158(1):463–475, May 2001.

[14] R. Paturi, S. Rajasekaran, and J. Reif. The light bulb problem. In *Proc. 2nd Annu. Workshop on Comput. Learning Theory*, pages 261 – 268, San Mateo, CA, 1989. Morgan Kaufmann.

[15] M. D. Ritchie, L. W. Hahn, and J. H. Moore. Power of multifactor dimensionality reduction for detecting gene-gene interactions in the presence of genotyping error, missing data, phenocopy, and genetic heterogeneity. *Genetic Epidemiology*, 24(2):150–157, Feb. 2003.

[16] J. L. Rodgers and W. A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42, 1988.

[17] Y. Wang, X. Liu, K. Robbins, and R. Rekaya. AntEpiSeeker: detecting epistatic interactions for case-control studies using a two-stage ant colony optimization algorithm. *BMC Bioinformatics*, 3:117–117, 2010.

[18] J. Xu, C. D. Langefeld, S. L. Zheng, E. M. Gillanders, B. Chang, S. D. Isaacs, A. H. Williams, K. E. Wiley, L. Dimitrov, D. A. Meyers, P. C. Walsh, J. M. Trent, and W. B. Isaacs. Interaction effect of PTEN and CDKN1B chromosomal regions on prostate cancer linkage. *Human Genetics*, 115(3):255–262, Aug. 2004.

[19] X. Zhang, S. Huang, F. Zou, and W. Wang. TEAM: efficient two-locus epistasis tests in human genome-wide association study. *Bioinformatics (Oxford, England)*, 26(12):i217–227, June 2010.

[20] X. Zhang, F. Zou, and W. Wang. Fastanova: an efficient algorithm for genome-wide association study. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 821–829, Las Vegas, Nevada, USA, 2008. ACM.

[21] Y. Zhang and J. S. Liu. Bayesian inference of epistatic interactions in case-control studies. *Nature Genetics*, 39(9):1167–1173, September 2007.