BitEpi: A Fast and Accurate Exhaustive Higher-Order Epistasis Search

Arash Bayat, Brendan Hosking, Yatish Jain, Cameron Hosking, Natalie Twine and Denis C. Bauer

Health and Biosecurity, CSIRO, Australia

Abstract

Motivation: Higher-order epistatic interactions can be the driver for complex genetic diseases. An exhaustive search is the most accurate method for identifying interactive SNPs. While there is a fast bitwise algorithm for pairwise exhaustive searching (BOOST), higher-order exhaustive searching has yet to be efficiently optimized.

Results: In this paper, we introduce BitEpi, a program to detect and visualize higher-order epistatic interactions using an exhaustive search. BitEpi introduces a novel bitwise algorithm that can perform higher-order analysis more quickly and is the first bitwise algorithm to search for 4-SNP interactions. Furthermore, BitEpi increases detection accuracy by using a novel entropy-based power analysis. BitEpi visualizes significant interactions in a publication-ready interactive graph. BitEpi is 56 times faster than MDR for 4-SNP searching and is up to 1.33 and 2.09 times more accurate than BOOST and MPI3SNP respectively.

Availability: Codes and data are publicly available on GitHub `https://github.com/aehrc/BitEpi`. BitEpi is also available on CodeOcean `https://doi.org/10.24433/CO.3671084.v1`.

Contact: `Arash.Bayat@csiro.au` - `Denis.Bauer@csiro.au`

Supplementary information: Supplementary data are available at *bioRxiv* online.

1 Introduction

Complex diseases often have a multi-genic component where the individual genomic locations can be both independently and interactively contribute to the disease [1]. The interactive effects are referred to as epistatic [2, 3]. Epistatic interactions involving 3 or more SNPs (higher-order) have been suggested to contribute to the 'missing heritability' problem in complex diseases [1, 2]. However, detecting such interactions is computationally challenging due to the exponential complexity of the problem [4, 5, 6, 7, 8]. Given a dataset with n SNPs, the exhaustive epistasis search with the order of m (number of interactive SNPs) requires $\binom{n}{m}$ combinations of SNPs to be tested, resulting in a complexity of $O(n^m)$.

Due to the exponential complexity of higher-order exhaustive search algorithms, it is not practical to apply them to large datasets. However, it is possible to use a filter to reduce the search space to a smaller number of SNPs before a more in-depth analysis [9, 10, 11, 12, 12] (i.e exhaustive search). Random Forest [13] is an efficient method for this filter as it preserves higher-order interactions [14]. Particularly, a new cloud-based implementation of Random Forest called VariantSpark [15] is able to process whole-genome data with 100,000,000 SNPs. It is capable of fitting tens of thousands of trees, which enables the interrogation of the search space more deeply, thereby reducing the chance of missing important interactions.

Irrespective of the applied filtering methodology, the key to discovering and annotating a complete set

of interactions is a fast exhaustive search. There are several algorithms for finding pairwise (2-SNP) interactions between genomic locations using exhaustive search approaches. With execution time a major limitation, algorithmic improvements focus on speedup. For example, TEAM [16] uses a minimum spanning tree algorithm to minimize execution time. More recently, BOOST [17] delivered a 168-fold speed up over TEAM [5] by using bitwise operations for pairwise interactions.

As it is likely that more than two genomic locations interact, efforts have been made to extend the exhaustive search capability to higher-order interactions. For example, CINOEDV [18] offers exhaustive searching for up to 5-SNP epistasis. However, with a focus on the visualization of the interactions, CINODEV was not designed for speed and its non-parallel implementation in `R` is 66.5 times slower than BOOST when processing 100 SNPs [18]. Capable of processing higher-order interactions more efficiently, MDR [19] (Multi-factor Dimensionality Reduction) is an extensive epistasis analysis platform offering parallel exhaustive search functionality. Improving on the algorithmic implementation further, MPI3SNP [20] adapts the bitwise approach used by BOOST. However, with MPI3SNP being limited to 3-SNP searches, the need for a fast higher-order search remains unaddressed.

In this paper, we introduce BitEpi, a fast and accurate exhaustive higher-order epistasis search program able to identify and visualize up to 4-SNP interactions. BitEpi introduces a novel bitwise approach capable of handling higher-order interactions, making it the first bitwise optimization method to be able to search for 4-SNP interactions. Unlike BOOST and MPI3SNP, that code each SNP to 3 bit-vectors, our algorithm uses 1 bit-vector to store each SNP, enabling a more efficient use of modern CPUs. Furthermore, BitEpi uses entropy-based power analysis, which has been demonstrated to better fit sparse contingency tables in epistasis analysis [21, 22, 18].

BitEpi visualizes interactions in a Cytoscape graph. Unlike CINOEDV's static plots, BitEpi's visualization is interactive and dynamic, allowing users to customize the layout (location of the nodes in the graph). This is essential when working with large interaction graphs (e.g. grouping interactive SNPs to better understand underlying biology).

The rest of this paper is organized as follows. Section 2 describes the underlying algorithm used by BitEpi. The performance and accuracy of BitEpi are documented in Section 3.1. Finally, we conclude by outlining future research directions.

2 Method

Processing each combination of SNPs includes two steps: the counting step to find the frequency of genotype combinations and the power analysis to compute the association power and the interaction effect size. The counting step is responsible for most of the execution time. Section 2.1 describes a bitwise process to speed up computing the contingency table for up to four SNPs. The accuracy to identify true epistasis interactions depends on the method used for power analysis. The statistics used to evaluate association power and the interaction effect size from the contingency table are then described in Section 2.2. We elaborate on our experimental setup in Section 2.3.

2.1 Counting

The input to BitEpi is a set of bi-allelic SNPs where there are three possible genotypes (0/0, 0/1 and 1/1). Multi-allelic SNPs should be broken into multiple bi-allelic SNPs before the analysis. Given m is the order of the analysis (number of interactive SNPs), the size of the contingency table is 3^m rows and two columns. Each row represents a different genotype combination for the selected SNPs. Columns represent the case and the control cohorts. Each entry of the table is the number of samples with a specific genotype for the selected SNPs in the case or control cohort. Table 1 illustrates an example contingency table for a pair of SNPs: A and B. The fifth row of the table explains that there are 34 cases and 46 controls with a heterozygous genotype for both A and B.

To speed up the process of counting samples in each cohort with the same genotype, we have implemented a fast bitwise algorithm. Bitwise representation of genotypes allows the genotypes of multiple

Table 1: An example contingency table for 2-SNP interaction of two SNPs: A and B.

A	B	Case	Case
0/0	0/0	23	424
0/0	0/1	263	423
0/0	1/1	534	634
0/1	0/0	87	45
0/1	0/1	34	46
0/1	1/1	56	345
1/1	0/0	345	34
1/1	0/1	56	64
1/1	1/1	456	547

samples to be stored in a machine word (64-bit) and processed in an operation (bit-level parallelization). In our algorithm, a genotype is encoded using two bits (i.e. 00, 01 and 10 for 0/0, 0/1 and 1/1 respectively) and stored in a byte (8-bits). The remaining 6 bits are set to 0. Thus, 8 samples can be stored in a 64-bit machine word (the parallelization factor is 8 samples per operation). Each SNP is stored in 1 bit-vector (**1-Vector** bitwise approach). Our algorithm uses bitwise SHIFT and OR operators to combine genotypes of up to 4 SNPs. In the resulting vector, each byte represents the genotype of all m SNPs for a sample. Thus, the counting process loops through the resulting vector and counts the frequency of each byte.

Figure 1 is an example that shows the binary representation of genotypes of four different SNPs: A, B, C and D across 8 samples (4 cases and 4 controls). The second, third and fourth SNPs are then shifted to the left by 2, 4 and 6 bits respectively. Next, all four SNPs are combined using bit-wise OR operations. These two steps are also shown in Figure 1. In the resulting array, each byte represents a genotype combination for a sample (a row in the contingency table). For example, 00010010 (for sample S4) represents the row in which D and B have the 0/0 genotype, C has the 0/1 genotype and A has the 1/1 genotype. To form the contingency table, BitEpi loops through the OR vector and counts the occurrences of each byte.

BitEpi eliminates the shift operations at each test by pre-computing 2, 4 and 6 bit shifted versions of the entire dataset (producing 3 extra copies) and storing them in memory before the analysis. Since the number of SNPs for an exhaustive epistasis analysis is limited, the redundancy in memory usage and the time to pre-compute shifted datasets are negligible.

Our 1-Vector bitwise approach is different from the 3-Vector bitwise approach used in BOOST and MPI3SNP. Algorithm 1 and Algorithm 2 illustrate the 3-Vector and 1-Vector bitwise approaches to compute one column of the contingency table in a m-SNPs interaction (i.e case column or control column). Both cohorts can be processed using the same algorithm.

Here, C represent a column of the contingency table where $C[i]$ is the number of samples in the ith row of the table (i starts from 0). $\{P[1] \dots P[m]\}$ represent m SNPs and R is a temporary variable (a 64-bit machine word).

In Algorithm 1, each SNP is encoded into 3 bit-vectors, $v[1]$, $v[2]$ and $v[3]$. Each bit-vector corresponds to a genotype (0/0, 0/1 and 1/1 respectively). For $P[i]$, if the jth sample has the 0/1 genotype, then the jth bit in $P[i].v[2]$ is set to 1. Each bit-vector is stored in an array of 64-bit machine words where each word contains the information for 64 samples (1 bit per sample). Thus the parallelization factor is 64 samples per operation. $P[i].v[j][k]$ represents the kth word of the jth vector of the ith SNP. There are $\lceil \frac{s}{64} \rceil$ words in each vector where s is the number of samples in the cohort (i.e. cases or controls). The core operation of Algorithm 1 includes m bit-wise AND operations, a $BitCount$ operation to count number of set bits (1's) in the result of AND operations (R) as well as an ADD operation. In this program there are m nested loops each iterating from 1 to 3. x_i is the iterator for the ith loop. These loops result in the complexity of $3^m s$ to perform each test.

On the other hand, our proposed 1-Vector bitwise method shown in Algorithm 2 does not have the 3^m exponential complexity. The down side is the lower parallelization factor (8 compared to 64). In Algorithm 2, $P^k[i].v[j]$ represents the jth word in the bit-vector of ith SNP shifted k bits to the left. The core operation of the algorithm is m bitwise OR operation and 8 increment operations. $R.byte[b]$ represents b^{th}

SNPs	Samples							
	Controls				Cases			
	S1	S2	S3	S4	S5	S6	S7	S8
A	0/1	1/1	0/1	1/1	0/1	0/1	1/1	0/1
B	0/0	1/1	0/0	0/0	1/1	0/1	0/1	0/0
C	0/1	0/0	1/1	0/1	1/1	0/1	0/0	0/0
D	0/1	0/0	0/1	0/0	1/1	1/1	0/1	0/1
	Binary Genotypes							
A	00000001	00000010	00000001	00000010	00000001	00000001	00000010	00000001
B	00000000	00000010	00000000	00000000	00000010	00000001	00000001	00000000
C	00000001	00000000	00000010	00000001	00000010	00000001	00000000	00000000
D	00000001	00000000	00000001	00000000	00000010	00000010	00000001	00000001
	Shifted Binary Genotypes							
A	00000001	00000010	00000001	00000010	00000001	00000001	00000010	00000001
B << 2	00000000	00001000	00000000	00000000	00001000	00000100	00000100	00000000
C << 4	00010000	00000000	00100000	00010000	00100000	00010000	00000000	00000000
D << 6	01000000	00000000	01000000	00000000	10000000	10000000	01000000	01000000
	Bitwise OR of Shifted Binary Genotypes							
OR	01010001	00001010	01100001	00010010	10101001	10010101	01000110	01000001

Figure 1: The bitwise representation of 4 example SNPs (A, B, C, and D) and the shifted bit-vectors as well as combined bit-vector.

```
for x_1 ← 1 to 3 do
    ⋱
    for x_m ← 1 to 3 do
        for w ← 1 to ⌈s/64⌉ do
            R ← P[1].v[x_1][w];
            for j ← 2 to m do
                R ← R ∧ P[j].v[x_j][w];
            end
            row ← 0;
            for j ← 1 to m do
                row ← (row × 3) + (x_j − 1);
            end
            C[row] ← C[row] + BitCount(R);
        end
    end
end
```

Algorithm 1: 3-Vector bitwise algorithm used in BOOST and MPI3SNP

byte in R (R consist of 8 bytes).

```
for w ← 1 to ⌈s/8⌉ do
    R ← P^0[1].v[w];
    for x ← 2 to m do
        R ← R ∨ P^{2(x−1)}[x].v[w];
    end
    for b ← 1 to 8 do
        C[R.byte[b]] + +;
    end
end
```

Algorithm 2: 1-Vector bitwise algorithm used in BitEpi

In the final executable all loops that are not a function of s are unfolded and flattened. Nevertheless, this implementation optimization does not eliminate the fact that in the 3-Vector method the time to test combinations of m SNPs exponentially increases with m (3^m). The results in Section 3.1 show this dependency.

2.2 Power Analysis

BitEpi computes two metrics for each combination of SNPs: the combined association power (β) and the interaction effect size (α). In a m-SNP analysis, β represents the combined association power of all m SNPs together. α represents the gain in the association power that only exists when considering all m SNPs together. If one of the SNPs is excluded from the set, the association power drops by at least α.

α and β are entropy metrics designed based on the concept of set purity in Gini-Index. The purity of a set p is computed using Equation 1 where x and y represent the number of case and control samples in the set. Each row of the contingency table represents a set of samples. The weighted average purity of these sets represents the combined association power of the given contingency table (β). The weight for each set is the ratio of the number of samples in the set to the total number of samples. Assuming x_i and y_i represent the number of case and control samples in the ith row of the contingency table, the combined association power is computed using Equation 2 where $\frac{x_i+y_i}{n}$ and $\frac{{x_i}^2+{y_i}^2}{(x_i+y_i)^2}$ are weight and purity of ith set (row) respectively.

$$p = \frac{x^2 + y^2}{(x+y)^2} \tag{1}$$

$$\beta = \sum_{i=1}^{m^3} (\frac{x_i + y_i}{n})(\frac{{x_i}^2 + {y_i}^2}{(x_i + y_i)^2}) \tag{2}$$

Assume G^m is a set of m SNPs $(a_1, a_2, ..., a_m)$ and G_i^{m-1} is G^m excluding a_i (i.e. G_i^{m-1} is a subset of G^m). Combined association power of G^m (β_{G^m}) is always greater than or equal to the combined association power of G_i^{m-1} ($\beta_{G_i^{m-1}}$). We call $\beta_{G_i^{m-1}}$ the lower-order β for β_{G^m}.

A high value of β_{G^m} does not necessarily indicate a strong interaction between SNPs. For example $\beta_{(A,B,C)}$ could be significant only because $\beta_{(A,C)}$ is significant. In this case, B does not play a role in the interaction (i.e. including B has a negligible effect on combined association power).

We are interested in the set of SNPs where the association power is driven by the interaction between

all SNPs in the set, as opposed to an individual SNP or through additive effects of SNP subsets. Thus to compute α_{G^m}, we subtract maximum lower-order β from β_{G^m} (see Equation 3). BitEpi computes α and β for individual SNPs too (normal GWAS). To compute α for an individual SNP β_{G^0} is computed as the purity of the set that includes all samples.

$$\alpha_{G^m} = \beta_{G^m} - \max_{i=1}^{m} \beta_{G_i^{m-1}} \tag{3}$$

In order to compute α_{G^m}, the program needs to compute $\beta_{G_i^{m-1}}$. Since there could be common SNPs between two sets of m SNPs, the same $\beta_{G_i^{m-1}}$ should be recomputed multiple times. For example, to compute $\alpha_{(A,B,C,x)}$ where x could be any SNP in the dataset other than A, B and C, $\beta_{(A,B,C)}$ should be recomputed. This results in a huge computational redundancy. To avoid this redundancy, prior to computing α_{G^m}, BitEpi computes all lower-order βs ($\beta_{G^{m-1}}$) and stores them in a multi-dimensional array. Using a multi-dimensional array to store β for all possible $(m-1)$-SNP combinations results in memory redundancy (memory is allocated but not used). However, lower order β values are accessed frequently and a multi-dimensional array allows for the fastest retrieval.

BitEpi can perform any combination of m-SNP α and β test in the same analysis where m could be 1, 2, 3 or 4. There is a special mode of operation called `best`. For each SNP, the `best` mode lists the 2-SNP, 3-SNP and 4-SNP interaction with the highest α.

BitEpi is implemented in `C++` with support for multi-threading. It includes a python wrapper so that it can be installed using `pip` and used in a python program. An `R` script is provided to turn BitEpi `best` output to a static `igraph` graph and a dynamic Cytoscape graph.

2.3 Experimental setup

Several synthetic datasets are used to evaluate the performance and accuracy of BitEpi and compare it with BOOST, MPI3SNP, and MDR.

To test the accuracy (detection power), we use GAMETES [23] to generate ground truth datasets (where the interactive SNPs are known). We create 10 simulated 2-SNP epistasis models with different heritability and minor allele frequency (MAF=0.01 and 0.5) of the interactive SNPs (PM1∼PM10), see *Supplementary Data* Table 4. Each model includes one 2-SNP interaction. We also create 9 epistasis models (TM1∼TM9) each of which includes one 3-SNP interactions (see *Supplementary Data* Table 5). For each model, 100 datasets are generated each with 100 SNPs and 2,000 samples (1,000 cases and 1,000 controls). To compute the detection power of algorithm A for model M, we process all 100 datasets generated from model M using algorithm A and count how many times the known interactive SNPs are ranked first. Model files with detailed model parameters are available in *Supplementary Data*.

To test execution time, we create much larger datasets by randomly assigning genotypes and phenotypes to samples. Each dataset consists of a different number of SNPs and samples (see *Supplementary Data* Table 1, *Supplementary Data* Table 2 and *Supplementary Data* Table 3 as well as Table 2).

To benchmark the performance of BitEpi against existing tools and test a wider range of epistatic models, we also compare on previously published synthetic datasets [24]. These datasets include 12 Marginal Effect (ME1∼ME12) and 40 No Marginal Effect (NME1∼NME40) epistasis models where each model includes one 2-SNP interaction. For each epistasis model, 100 datasets each with 100 SNPs and 1,600 samples (800 case and 800 controls) are simulated.

In our analysis, we use a multi-threaded implementation of BOOST available in Plink v1.9 (`https://www.cog-genomics.org/Plink/1.9/epistasis#fast`).

3 Result

3.1 BitEpi is faster for higher-order interactions

Table 2 compares the execution time of BitEpi's α test with BOOST, MPI3SNP and MDR, all exhaustive search algorithms finding up to 2-SNP, 3-SNP, and 4-SNP interactions, respectively. Both BitEpi

and BOOST can be applied to the largest dataset (50,000 SNPs), where BOOST's targeted 2-SNP algorithm is up to 4 times faster than BitEpi. For higher-order interactions, BitEpi performs the fastest out of all surveyed methods. Specifically, BitEpi performs up to 1.7 times faster than MPI3SNP for 3-SNP searches (2,000 SNPs dataset) and up to 65, 76 and 56 times faster than MDR for 2-SNP, 3-SNP and 4-SNP searches (20,000 SNPs, 1,000 SNPs, and 200 SNPs datasets) respectively. Note that we report the largest dataset the algorithms were capable of processing within the given compute resources and time-cutoff (1 hour).

BitEpi's observed speedups over MPI3SNP are likely due to the 1-Vector algorithm in BitEpi being independent of the order of the epistasis interaction. This allows BitEpi to perform the individual interaction tests at the same speed irrespective of whether a 2-SNP, 3-SNP or 4-SNP interaction is tested. To quantify the improvement, we compute the test time for each order. As the order of epistasis also increases the number of tests that need to be performed, we normalize execution time by the number of tests performed, to be able to directly compare the individual 2-SNP, 3-SNP and 4-SNP tests between 3-Vector and 1-Vector bitwise algorithms. We hence compute the average test time as $(\frac{ExecutionTime \times NumberOfThreads}{\binom{n}{m}})$. Where $\binom{n}{m}$ is the number of m-SNP tests in a dataset with n SNPs for the datasets highlighted in Table 2.

Figure 2 shows that the exponential factor (3^m) in the 3-Vector bitwise procedure causes the execution time to increase from the 2-SNP test with BOOST taking $0.7\mu s$ on average, to $4.5\mu s$ on average for the 3-SNP test with MPI3SNP (6.3 times slower). As there are no 4-SNP bitwise methods published to date, we need to extrapolate from the 3-SNP searches, which results in execution time of $4.5\mu s \times 3 = 13.5\mu s$ for a 4-SNP search. Note that the complexity of the 3-Vector bitwise method grow exponentially with the number of interactive SNPs (3^m).

This stands in stark contrast to the 1-Vector approach (BitEpi) where the execution time remains constant ($2.7 \sim 2.9\mu s$) for all tested orders, resulting in 1.7 and 4.7 fold faster tests for 3-SNP and 4-SNP search, respectively. Note that the non-bitwise 4-SNP method tested, MRD, is significantly slower with an average test time of $210\mu s$, $204\mu s$ and $206\mu s$ for 2-SNP, 3-SNP and 4-SNP search, respectively.

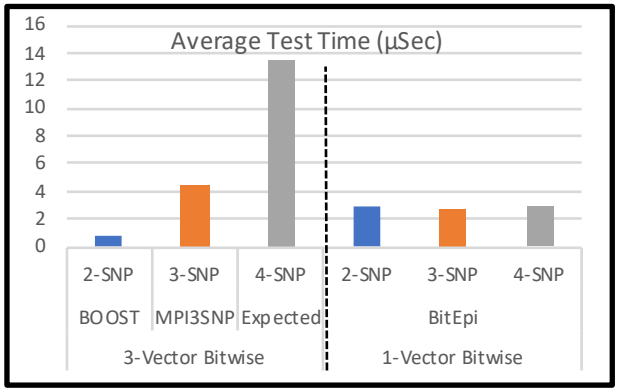


Figure 2: The average test time using BitEpi's 1-Vector bitwise approach compared to the existing 3-Vector bitwise approach. The expected 4-SNP average test time with the 3-Vector bitwise approach is computed as $MPI3SNP \times 3$. The average test time is computed based on the highlighted execution time in Table 2

BitEpi scales linearly with the number of samples, as shown in *Supplementary Data* Table 2 (total execution time with increase samples) and *Supplementary Data* Figure 1.b (normalized by samples). BitEpi scales exponentially with the number of SNPs, v, resulting in $O(v^2)$, $O(v^3)$ and $O(v^4)$ for 2-SNP, 3-SNP and 4-SNP, respectively, show in *Supplementary Data* Table 1 (total execution time) and *Supplementary Data* Figure 1.a (normalized execution time per SNP). Both execution times can be curbed by parallelization, as shown in *Supplementary Data* Table 3 and *Supplementary Data* Figure 1.c, in which using 2,4,8 and 16 CPUs results in a non-saturated, near-linear speed-up.

3.2 BitEpi is more accurate in detecting interactions

To compare the accuracy of BitEpi with BOOST and MPI3SNP, we compute the detection power for all

Table 2: The execution time (in seconds) of epistasis algorithms for 2,000 samples and different numbers of SNPs. The process is killed if it takes more than an hour to complete and the execution time is not measured. If the execution time is less than a second it is reported as 1 in this table. All programs are executed with 16 parallel threads. Highlighted execution times are used to compute the average test time (see Figure 2).

Order of Epistasis	Algorithm	Number of SNPs								
		100	200	500	1,000	2,000	5,000	10,000	20,000	50,000
2-SNP	BitEpi	1	1	1	1	1	4	11	40	**227**
	BOOST	1	1	1	1	1	1	4	15	**56**
	MDR	1	1	2	6	29	148	390	**2619**	-
3-SNP	BitEpi	1	1	4	28	**221**	-	-	-	-
	MPI3SNP	1	1	6	47	**375**	-	-	-	-
	MDR	3	14	212	**2121**	-	-	-	-	-
4-SNP	BitEpi	1	15	**460**	-	-	-	-	-	-
	MDR	51	**834**	-	-	-	-	-	-	-

models simulated by GAMETES including models simulated by others [24]. Figure 3a shows the 2-SNP detection power of BitEpi and BOOST for the models we simulated. Except for the PM1 model where both methods result in poor detection power, BitEpi performs better than BOOST (i.e. between 1.22 and 1.33 times more accurate) and reaches 100% detection power for PM7∼PM10 Models.

Figure 3b shows the 3-SNP detection power of BitEpi and MPI3SNP. BitEpi performs better than MPI3SNP for TM2∼TM6 Models (i.e. between 1.56 and 2.09 times more accurate), and equivalent for the rest. Numerical comparisons are available in *Supplementary Data* Table 4 (2-SNP) and *Supplementary Data* Table 5 (3-SNP).

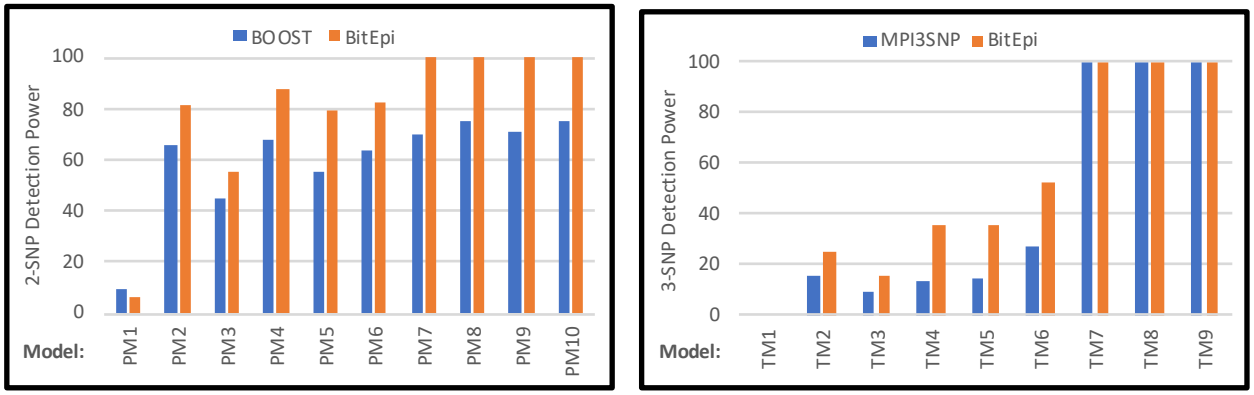
We also compute the 2-SNP detection power of BitEpi and BOOST for 12 ME (Marginal Effect) and 40 NME (No Marginal Effect) epistasis models simulated in [24]. *Supplementary Data* Figure 2 illustrates the comparison result. Numerical comparisons are available in *Supplementary Data* Table 6 and *Supplementary Data* Table 7. BitEpi's detection power for ME models is 44% higher than BOOST's on average. For NME models, BitEpi's average detection power is the same as BOOST's.

Out of the 71 epistasis models we have evaluated, BitEpi performs better than other methods in 24 cases, similar to other methods in 39 cases and is less accurate than other methods in 8 cases. Due to the accurate isolation of interaction effect sizes, BitEpi eliminates false positives more effectively.

3.3 BitEpi visualizes interactions

BitEpi visualizes interactions in an interactive Cytoscape graph. In the graph, SNPs and interactions are shown as nodes. Edges connect interaction nodes to the corresponding interactive SNPs. For example in Figure 4, the 2-SNP interaction node BC is connected to SNP nodes B and C. The example plot shows the top 3 significant SNPs as well as the top 3 significant 2-SNP, 3-SNP, and 4-SNP interactions highlighted by colors (red, blue, orange and green respectively). Gray nodes represent SNPs that are themselves not listed as the top 3 significant SNPs but are part of interactions that are in the top 3. For the colored nodes, the size of a node represents the combined association power (β). For the gray node, the size is set to the minimum node size. Node sizes are scaled such that all nodes in the graph are visible.

The top significant SNPs and interactions are chosen by α but the node size in the graph represents β.



(a) 2-SNP detection power of BitEpi and BOOST.

(b) 3-SNP detection power of BitEpi and MPI3SNP.

Figure 3: Compare detection power of BitEpi with BOOST and MPI3SNP for 2-SNP and 3-SNP analysis.

This is because we are looking for the most significant interactions, and once we find them we are interested in their combined association power. Note that node sizes of different graphs can not be compared.

The dataset used to generate the graph in Figure 4 is created using GAMETES with 3 models: a single SNP (A), a 2-SNP interaction (BC) and a 4-SNP interaction (DEFG). Models and the dataset are distributed with BitEpi. As shown in the graph, BitEpi correctly identifies the underlying ground truth signal and the user can get detailed information about each node by selecting a node in the Cytoscape environment.

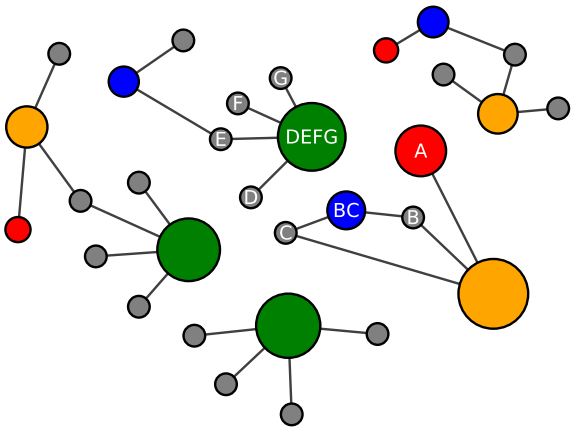


Figure 4: BitEpi output graph (exported from Cytoscape) with labeled ground truth signal in an example dataset.

Conclusion

We demonstrated that the current best practice for exhaustive epistasis search tools (BOOST, MPI3SNP) can be improved upon in both speed and accuracy. While heuristics such as Random Forest remain necessary to reduce the initial search space, BitEpi is then capable of detecting higher-order interactions of up to 4-SNP exhaustively, resulting in an up to 1.7 and 56 fold faster execution time than other surveyed methods for 3-SNP and 4-SNP searches, respectively.

BitEpi uses a novel 1-Vector bitwise approach that is designed for higher-order analysis and allows mod-

ern 64-bit machines to be used more effectively than the previous 3-Vector bitwise approaches. It also isolates the interaction effect size using an entropy-based metric to eliminate false positives. BitEpi visualizes the results in an interactive graph that can be dynamically scaled and rearranged, streamlining interpretation and publication.

Future improvements will cover more advanced visualization approaches using either d3 or Cytoscape JavaScript library for dynamic web-based visualization and the end-to-end integration with cloud-based Random Forest implementation VariantSpark [15], to enable epistasis search within the ultra-high dimensional data of whole-genome sequencing cohorts.

Acknowledgements

AB, BH and YJ implemented the algorithm and perform experiments. AB, NT, and DB conceived the work and written most of the paper. CH refined the methodology section. All authors have read and approved the manuscript.

Funding

CSIRO and the eHealth reserach centre has funded this work.

References

[1] Wen-Hua Wei, Gibran Hemani, and Chris S Haley. Detecting epistasis in human complex traits. *Nature Reviews Genetics*, 15(11):722, 2014.

[2] Daniel M Weinreich, Yinghong Lan, C Scott Wylie, and Robert B Heckendorn. Should evolutionary geneticists worry about higher-order epistasis? *Current opinion in genetics & development*, 23(6):700–707, 2013.

[3] Matthew B Taylor and Ian M Ehrenreich. Higher-order genetic interactions and their contribution to complex traits. *Trends in genetics*, 31(1):34–40, 2015.

[4] Clément Niel, Christine Sinoquet, Christian Dina, and Ghislain Rocheleau. A survey about methods dedicated to epistasis detection. *Frontiers in genetics*, 6:285, 2015.

[5] Junliang Shang, Junying Zhang, Yan Sun, Dan Liu, Daojun Ye, and Yaling Yin. Performance analysis of novel methods for detecting epistasis. *BMC bioinformatics*, 12:475, dec 2011.

[6] Li Chen, Guoqiang Yu, David J. Miller, Lei Song, Carl Langefeld, David Herrington, Yongmei Liu, and Yue Wang. A ground truth based comparative study on detecting epistatic SNPs. In *2009 IEEE International Conference on Bioinformatics and Biomedicine Workshop*, pages 26–31. IEEE, nov 2009.

[7] Mathieu Emily. A survey of statistical methods for gene-gene interaction in case-control genome-wide association studies. *Journal de la Societe Française de Statistique*, 159(1):27–67, 2018.

[8] Heather J Cordell. Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans. *Human molecular genetics*, 11(20):2463–2468, 2002.

[9] Margaret J Eppstein and Paul Haake. Very large scale relieff for genome-wide association analysis. In *2008 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 112–119. IEEE, 2008.

[10] Makiko Yoshida and Asako Koike. Snpinterforest: a new method for detecting epistatic interactions. *BMC bioinformatics*, 12(1):469, 2011.

[11] Xia Cao, Guoxian Yu, Jie Liu, Lianyin Jia, and Jun Wang. Clustermi: Detecting high-order snp interactions based on clustering and mutual information. *International journal of molecular sciences*, 19(8):2267, 2018.

[12] Yan Meng, Qiong Yang, Karen T Cuenco, L Adrienne Cupples, Anita L DeStefano, and Kathryn L Lunetta. Two-stage approach for identifying single-nucleotide polymorphisms associated with rheumatoid arthritis using random

forests and bayesian networks. In *BMC proceedings*, volume 1, page S56. BioMed Central, 2007.

[13] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[14] Rui Jiang, Wanwan Tang, Xuebing Wu, and Wenhui Fu. A random forest approach to the detection of epistatic interactions in case-control studies. *BMC bioinformatics*, 10(1):S65, 2009.

[15] Arash Bayat, Piotr Szul, Aidan R O'Brien, Robert Dunne, Oscar J Luo, Yatish Jain, Brendan Hosking, and Denis C Bauer. Variantspark, a random forest machine learning implementation for ultra high dimensional data. *bioRxiv*, page 702902, 2019.

[16] Xiang Zhang, Shunping Huang, Fei Zou, and Wei Wang. Team: efficient two-locus epistasis tests in human genome-wide association study. *Bioinformatics*, 26(12):i217–i227, 2010.

[17] Xiang Wan, Can Yang, Qiang Yang, Hong Xue, Xiaodan Fan, Nelson LS Tang, and Weichuan Yu. Boost: A fast approach to detecting gene-gene interactions in genome-wide case-control studies. *The American Journal of Human Genetics*, 87(3):325–340, 2010.

[18] Junliang Shang, Yingxia Sun, Jin-Xing Liu, Junfeng Xia, Junying Zhang, and Chun-Hou Zheng. Cinoedv: a co-information based method for detecting and visualizing n-order epistatic interactions. *BMC bioinformatics*, 17(1):214, 2016.

[19] Jason H Moore and Peter C Andrews. Epistasis analysis using multifactor dimensionality reduction. In *Epistasis*, pages 301–314. Springer, 2015.

[20] Christian Ponte-Fernández, Jorge González-Domínguez, and María J Martín. Fast search of third-order epistatic interactions on cpu and gpu clusters. *The International Journal of High Performance Computing Applications*, page 1094342019852128, 2019.

[21] Ting Hu, Yuanzhu Chen, Jeff W Kiralis, Ryan L Collins, Christian Wejse, Giorgio Sirugo, Scott M Williams, and Jason H Moore. An information-gain approach to detecting three-way epistatic interactions in genetic association studies. *Journal of the American Medical Informatics Association*, 20(4):630–636, 2013.

[22] Sangseob Leem, Hyun-hwan Jeong, Jungseob Lee, Kyubum Wee, and Kyung-Ah Sohn. Fast detection of high-order epistatic interactions in genome-wide association studies using information theoretic measure. *Computational biology and chemistry*, 50:19–28, 2014.

[23] Ryan J Urbanowicz, Jeff Kiralis, Nicholas A Sinnott-Armstrong, Tamra Heberling, Jonathan M Fisher, and Jason H Moore. Gametes: a fast, direct algorithm for generating pure, strict, epistatic models with random architectures. *BioData mining*, 5(1):16, 2012.

[24] Peng-Jie Jing and Hong-Bin Shen. Macoed: a multi-objective ant colony optimization algorithm for snp epistasis detection in genome-wide association studies. *Bioinformatics*, 31(5):634–641, 2014.