Hybrid CPU/GPU Acceleration of Detection of 2-SNP Epistatic Interactions in GWAS

Jorge González-Domínguez[1], Bertil Schmidt[1],
Jan Christian Kässens[2], and Lars Wienbrandt[2]

[1] Parallel and Distributed Architectures Group,
Johannes Gutenberg University - Mainz, Germany
Email: {*j.gonzalez, bertil.schmidt*}*@uni-mainz.de*
[2] Department of Computer Science, Christian-Albrechts-University of Kiel
Email: {*jka, lwi*}*@informatik.uni-kiel.de*

Abstract. High-throughput genotyping technologies allow the collection of up to a few million genetic markers (such as SNPs) of an individual within a few minutes of time. Detecting epistasis, such as 2-SNP interactions, in Genome-Wide Association Studies is an important but time consuming operation since statistical computations have to be performed for each pair of measured markers. In this work we present EpistSearch, a parallelized tool that, following the log-linear model approach, uses a novel filter to determine the interactions between all SNP-pairs. Our tool is parallelized using a hybrid combination of Pthreads and CUDA in order to take advantage of CPU/GPU architectures. Experimental results with simulated and real datasets show that EpistSearch outperforms previous approaches, either using GPUs or only CPU cores. For instance, an exhaustive analysis of a real-world dataset with 500,000 SNPs and 5,000 individuals requires less than 42 minutes on a machine with 6 CPU cores and a GTX Titan GPU.
Keywords: Bioinformatics, GWAS, Epistasis, Pthreads, CUDA

1 Introduction

High-throughput genotyping technologies allow the collection of hundreds of thousands to a few million genetic markers, such as Single Nucleotide Polymorphisms (SNPs) of an individual within a few minutes of time. In Genome-Wide Association Studies (GWAS) these genotypes are typically measured for several thousand individuals and then linked to a given phenotype of each individual, such as the presence (case) or absence (control) of an associated disease. In classical single-locus GWAS each genetic marker is then analyzed individually in order to identify markers that are significantly different for cases and controls. Unfortunately, this approach is generally not powerful enough to model complex traits for which the detection of joint genetic effects (epistasis) needs to be considered [1,2]. In 2-way statistical epistasis each pair of measured markers is therefore tested in order to discover interactions between SNP-pairs that explain the given phenotype.

Consequently, a number of algorithms and tools have been developed to address the problem of detecting epistasis in recent years using one or several statistical tests over all SNP-pairs [3]. The main goal of these approaches is to find SNP-pairs whose joint values show a statistically significant difference compared to the individual SNP values. One of the most popular approaches uses statistical regression methods [4,5]. These tests are very precise but the pair-wise analyses are very computationally-expensive. As an example, it is necessary to apply the statistical tests to 125 billion pairs when analyzing a moderately-sized dataset consisting of half million SNPs [6,7].

Many recent approaches are filtration-based; i.e. they firstly apply a computationally faster filter and subsequently perform the full statistical analysis only to the SNP-pairs not discarded by the preliminary filter. SNPHarvester [8] uses path algorithms to identify several groups of SNPs associated to the same disease. Then, it applies the statistical method only to the pairs generated within each group. SNPRuler [9] narrows the search space through a learning approach based on predictive rule learning. BOOST [10] introduces the Kirkwood Superposition Approximation (KSA) as preliminary filter. This last tool was taken as basis for our work because it is currently widely used by biologists (see e.g. [11,12,13]). Furthermore, it is faster than previous approaches not only for CPU but also for GPU computation (GBOOST [14]).

As the development of epistasis tools has attracted extensive research interests, even more recent work that try to improve precision using different statistical methods has arisen. iLOCi [15] uses a statistical method based on the difference of the dependency of controls and cases, but our preliminary benchmarking demonstrated that it is much slower than BOOST. GWIS [16], which presents a GPU implementation of a method based on ROC-curves, could not be tested since merely a web interface is publicly available. Thus, these tools and their statistical methods are not as commonly used by biologists as BOOST and GBOOST. In this paper we present EpistSearch. In order to further improve the speed of this approach our tool introduces a novel preliminary filter and takes advantage of heterogeneous CPU/GPU architectures through inter-task hybrid parallelism to perform fast epistasis search in GWAS datasets.

The rest of the paper is organized as follows. Section 2 describes the BOOST method that is adapted by our tool. Our novel preliminary KSASA filter is presented in Section 3. Section 4 describes our hybrid parallelization approach. Runtime performance is evaluated and compared in Section 5 using both simulated and real-world datasets. Section 6 concludes the paper.

2 Background

2.1 Contingency Tables

We work with datasets of biallelic genetic markers where major alleles are denoted with capital letters and minor alleles with lowercase letters. Therefore, for each SNP there are three genotypes $\{AA,Aa,aa\}$, which are numerically represented as $\{0,1,2\}$. The number of SNPs and individuals are denoted as M

Table 1. Example of contingency table

Cases	SNP2=0	SNP2=1	SNP2=2	Controls	SNP2=0	SNP2=1	SNP2=2
SNP1=0	n_{000}	n_{010}	n_{020}	SNP1=0	n_{001}	n_{011}	n_{021}
SNP1=1	n_{100}	n_{110}	n_{120}	SNP1=1	n_{101}	n_{111}	n_{121}
SNP1=2	n_{200}	n_{210}	n_{220}	SNP1=2	n_{201}	n_{211}	n_{221}

and N, respectively. The individuals are categorized as cases (value 0) and controls (value 1). The filters that select the SNP-pairs that present interaction use a 3x3x2 contingency table per pair. As seen in the example of Table 1, each cell ijk stores the count of individuals categorized as k (case or control) with the value of the first SNP as i, and the second SNP as j. We can also fill the contingency table with probabilities: $\pi_{ijk} = n_{ijk}/N$.

2.2 Log-Linear Models and the KSA Filter

The purpose of a 2-SNP statistical epistasis tool is to identify SNP-pairs whose joint values are significantly different from the joint values expected from the individual SNP values. In [10] Wan et al. prove that the search for interaction with regression models can be simplified using log-linear models. They define interaction from the perspective of the log-linear models as the information contained in the joint distribution but not in its lower-order factorization. This definition led to measure interaction as $\hat{L}_S - \hat{L}_H$, where $\hat{L}_S$ and $\hat{L}_H$ represent the maximum log-likelihood of the saturated and the homogeneous association models, respectively. It can be calculated from the values of the contingency table as:

$$N \sum_{ijk} \left[\hat{\pi}_{ijk} \log \left(\frac{\hat{\pi}_{ijk}}{\hat{p}_{ijk}} \right) \right]$$

where $\hat{\pi}_{ijk}$ is the joint distribution obtained under the saturated model and $\hat{p}_{ijk}$ the distribution obtained under the homogeneous association model. They establish that all pairs with log-linear measure higher than certain threshold T present epistasis. Although this log-linear model is affordable, it still requires a lot of computation as $\hat{p}_{ijk}$ has to be computed through iterative methods. This is the reason why BOOST applies a simpler filter based on the Kirkwood Superposition Approximation (KSA). The authors proved the following upper bound:

$$\begin{aligned}
\hat{L}_{\mathrm{S}} - \hat{L}_{\mathrm{H}} &\leq \hat{L}_{\mathrm{S}} - \hat{L}_{\mathrm{KSA}} \\
\hat{L}_{\mathrm{S}} - \hat{L}_{\mathrm{KSA}} &= N \sum_{ijk} \left[\hat{\pi}_{ijk} \log \left(\frac{\hat{\pi}_{ijk}}{\hat{p}^k_{ijk}} \right) \right] \\
\hat{p}^k_{ijk} &= \frac{1}{\eta} \frac{\pi_{ij.} \pi_{i.k} \pi_{.jk}}{\pi_{i..} \pi_{.j.} \pi_{..k}} \\
\eta &= \sum_{ijk} \frac{\pi_{ij.} \pi_{i.k} \pi_{.jk}}{\pi_{i..} \pi_{.j.} \pi_{..k}}
\end{aligned}$$

The equations above show that the KSA value can be directly calculated from the cells of the contingency table without iterative methods. Therefore, BOOST and GBOOST accelerate their analyses using the KSA filter ($\hat{L}_S - \hat{L}_{KSA}$). From now, we call the value of $\hat{L}_S - \hat{L}_{KSA}$ for a specific SNP-pair its "KSA value". As the KSA value is an upper bound of the log-linear measure, these tools calculate it for all SNP-pairs and discard those with a KSA value lower than T. Finally, they only apply the log-linear filter to the remaining pairs. For simplicity, we refer to [10] to find the proofs and further explanation of the KSA and log-linear filters.

3 KSA's Superposition Approximation (KSASA)

Although the KSA filter does not need iterative methods, a relatively large amount of numerical computations still have to be performed on each pair. Thus, we have designed a novel simpler filter called KSA Superposition Approximation (KSASA). EpistSearch applies the KSASA filter (upper bound for KSA) to all SNP-pairs, discarding all that have a value below the threshold, and only calculating the KSA and log-linear values for the other. The pseudo-code of EpistSearch is summarized in Algorithm 1.

```
foreach SNP-pair P do
    v = KSASA_Value(P)
    if v > T then
        v = KSA_Value(P)
        if v > T then
            v = LogLinear_Value(P)
            if v > T then
                Print P in the output file as pair with epistasis
            end
        end
    end
end
```

Algorithm 1: Pseudo-code of EpistSearch

In order to prove that KSASA is an upper bound for KSA, let E and O denote the counts of expected (control) and observed (case) studies, then the total variation distance and the total spread are:

$$\delta(E,O) = \frac{1}{2}\sum_x |E_x - O_x| = \frac{1}{2}\sum_{ij} |\pi_{ij1} - \pi_{ij0}|$$

$$D_{\text{spread}}(E,O) = \sum_x (E_x - O_x)^2 = \sum_{ij} (\pi_{ij1} - \pi_{ij0})^2$$

Following a similar approach as for the design of the log-linear and KSA filters, we use the discrete Kullback-Leibler divergence as measure:

$$D_{\text{KL}}(E,O) = \sum_x E_x \log\left(\frac{E_x}{O_x}\right) = \sum_{ij} \pi_{ij1} \log\left(\frac{\pi_{ij1}}{\pi_{ij0}}\right)$$

This Kullback-Leibler divergence between the empirical distributions of the input classes is much faster to calculate than the KSA value. However, we need to prove that it is an upper bound of the KSA value in order to be used as prefilter:

$$\hat{L}_{\mathrm{S}} - \hat{L}_{\mathrm{H}} \leq \hat{L}_{\mathrm{S}} - \hat{L}_{\mathrm{KSA}} \leq N * D_{\mathrm{KL}}(E, O)$$

This inequality reduces to prove that the Kullback-Leibler divergence between the maximum likelihood estimate of the joint distribution obtained from a homogeneous association model and the maximum likelihood estimate of the joint distribution obtained from the Kirkwood superposition approximation is bounded by the Kullback-Leibler divergence of the empirical distributions of the input classes:

$$\sum_{ijk} \left[\hat{\pi}_{ijk} \log \left(\frac{\hat{\pi}_{ijk}}{\hat{p}^{k}_{ijk}} \right) \right] \leq c \sum_{ij} \pi_{ij1} \log \left(\frac{\pi_{ij1}}{\pi_{ij0}} \right)$$

Reducing the inequality further we obtain:

$$\sum_{ijk} \frac{n_i jk}{N} \left[\log \left(\frac{n_i jk}{N} \right) + \log(\eta) - \log \left(\frac{n_{ij.} n_{i.k} n_{.jk}}{n_{i..} n_{.j.} n_{..k}} \right) \right] \leq$$
$$c \sum_{ij} \left[\frac{n_{ij1}}{n_{..1}} \log \left(\frac{n_{ij1}}{n_{..1}} \right) - \frac{n_{ij0}}{n_{..0}} \log \left(\frac{n_{ij0}}{n_{..0}} \right) \right]$$

Since it can be shown that the last inequality holds[a], $N \cdot D_{KL}(E, O)$ is a valid upper bound that can be used as our KSASA prefilter.

4 Parallelization Approach

4.1 Optimization of the Calculation of Contingency Tables

A boolean representation of genotype data is employed in BOOST in order to calculate the values of the 18 cells of the contingency tables in a fast manner. EpistSearch optimizes this approach further by reducing the number of explicitly calculated cells to only 8 (shown without "-" in the Table 2). When loading the datasets, the sums of the AA and aa biallelic values are calculated per SNP. This information is also provided to the filters and can then be used to calculate the remaining cells of the table if necessary. As the sums are only calculated once per SNP, this approach is faster than calculating the values of 10 additional cells per SNP-pair.

4.2 Inter-Task Hybrid CPU-GPU Parallelism

Although a heterogeneous CPU-GPU architecture is the common platform for GPU-based applications, the CPU usually performs tasks that are inherently sequential or have a low computational intensity. Therefore, GPU applications

[a] Because of the page limitation the detailed proof is omitted in this paper

Table 2. Values of the contingency table explicitly calculated by EpistSearch

Cases	SNP2=0	SNP2=1	SNP2=2	Controls	SNP2=0	SNP2=1	SNP2=2
SNP1=0	n_{000}	-	n_{020}	SNP1=0	n_{001}	-	n_{021}
SNP1=1	-	-	-	SNP1=1	-	-	-
SNP1=2	n_{200}	-	n_{220}	SNP1=2	n_{201}	-	n_{221}

usually waste most of the computational power of CPU multicores. For instance, GBOOST [14] applies intra-task parallelism where the GPU computes the KSA filter for all pairs and the CPU computes only the log-linear filter of the pairs that were not discarded. As the percentage of pairs that pass the KSA filter is usually very low, the CPU is often idle. On the contrary, EpistSearch applies inter-task parallelism so that the CPU and GPU threads perform the whole computation but for different SNP-pairs. This hybrid parallelism has already been shown to be effective in biological sequence database search [17] and next generation sequencing read alignment [18]. Furthermore, the CPU computation is also parallelized with the POSIX Threads Programming technology (Pthreads) [19] to take advantage of CPU multicore platforms.

4.3 CUDA Implementation

We use the CUDA programming model [20] for the GPU implementation of EpistSearch. A single kernel that performs the whole analysis of a set of SNP-pairs is developed. The overall approach works as follows:

1. The whole information of the SNPs is transferred to the device memory through pinned copies at the beginning of the execution.
2. The CUDA kernel that analyzes the interaction of a subset of pairs is launched several times. In the kernel each thread creates the contingency table of a number of SNP-pairs independently and performs the necessary filters.

The execution finishes when all pairs have been processed. When assigning the GPU resources to the different parts of the code, we gave the highest priority to the KSASA filter, as it is executed for all SNP-pairs. Therefore, this filter is implemented using registers and it does not directly accesses the device memory.

The current implementation of EpistSearch can only work with datasets that fit into the device memory. The largest currently available WTCCC dataset contains about 500,000 SNPs from 5,000 individuals. This can be stored in around 600MB of memory, which is available in almost any modern GPU. For example, a Tesla K40 GPU, with 12GB of memory, would be able to analyze datasets with more than 5 million SNPs from 25,000 individuals. This should be sufficient to analyze most large-scale datasets in the near future.

Depending on the results of the KSASA filter, GPU threads that test pairs discarded by this preliminary filter would be idle while other threads are performing the KSA and Log-Linear filters. For instance, in a scenario where the probability of a SNP-pair passing the KSASA filter is 0.01, 99% of threads would finish

their computation in the kernel after the KSASA filter, but they would have to wait for the remaining 1%. As mentioned in Section 4.2, GBOOST addresses this thread divergence problem by performing the calculation of the KSA and log-linear values on the CPU. Although this approach eliminates CUDA thread divergence, it significantly decreases performance if many SNP-pairs pass the first filter. An alternative solution would be the division of the computation in two different kernels: the first one for the generation of the contingency tables and the KSASA filter (performed for all SNP-pairs), and the second kernel for the KSA and log-linear filters. However, the overhead of copying the contingency tables of pairs that pass the KSASA filter between kernels would cause a significant performance overhead. Therefore, EpistSearch maintains only one kernel with all the computation but, in order to reduce thread divergence, each thread evaluates 64 SNP-pairs every time the kernel is launched.

5 Performance Evaluation

The performance of EpistSearch has been evaluated by looking for interactions between SNP-pairs in several simulated datasets and one real dataset. All the experiments have been conducted on a system with a hex-core Intel Core i7 Sandy Bridge 3.20GHz CPU with 12MB cache, and two different NVIDIA Kepler GPUs, whose specifications are shown in Table 3. The runtime of EpistSearch is compared to BOOST and GBOOST using the same dataset and threshold. Note that EpistSearch and (G)BOOST produce the same output for all the experiments. Thus, the accuracy of EpistSearch and (G)BOOST is identical, and therefore we just compare the runtime performance.

Table 3. Specifications of the two GPUs used for the experimental evaluation

Name	Number of SMs	Number of cores	Core frequency	Memory size
GTX 650Ti	4	768	980MHz	2GB
GTX Titan	14	2688	875.5MHz	6GB

Our first evaluation uses only CPU cores and 6 different simulated datasets generated with the genomeSIMLA tool [21]. All the datasets are based on the same penetrance table (epi1 model in the supplementary material of [10]), but vary in terms of the number of SNPs and individuals.

Table 4 shows the percentage of pairs that pass the KSASA and log-linear filters for each dataset explored on the CPU. The percentage for the KSA filter is not included because it is always very similar to the reported log-linear percentage. It can be seen that the KSASA filter discards much less SNP-pairs than the KSA and log-linear filters. The percentages vary from 8.88% to 25.46% (almost 3x) and from 0.0006% to 0.017% (more than 28x) in the KSASA and log-linear filters, respectively. Thus, we can assert that the evaluation is performed in very different scenarios. Figure 1 shows the execution times of BOOST and

Table 4. Percentage of pairs that pass the KSASA and log-linear filters in the CPU experiments.

Num. Inds. →	**800**		**1600**		**3200**	
Num. SNPs →	**10K**	**40K**	**10K**	**40K**	**10K**	**40K**
KSASA	18.84	15.95	12.17	8.88	25.46	14.27
log-linear	11×10^{-4}	6×10^{-4}	27×10^{-4}	8×10^{-4}	170×10^{-4}	19×10^{-4}

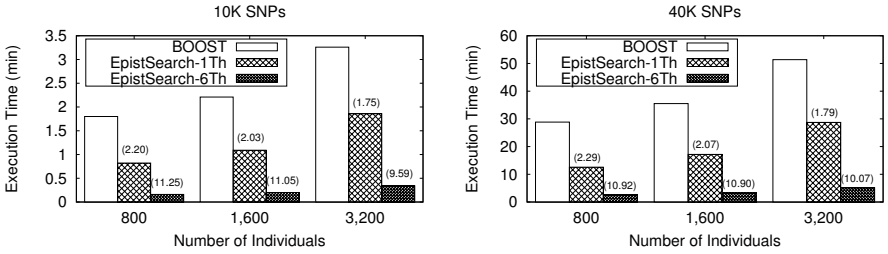

Fig. 1. Execution times of BOOST and EpistSearch (with 1 and 6 threads) in the CPU. Speedups compared to BOOST are shown in brackets.

EpistSearch when running only on the multicore CPU part of the test platform. Additionally, it shows the speedups for each EpistSearch execution (in parenthesis) compared to BOOST. As BOOST does not have support for parallelism it can only exploit one of the cores. For EpistSearch we present results using only one core and the whole hex-core processor. EpistSearch significantly outperforms BOOST even when using only one core: it is more than 2x faster for all experiments with 800 and 1600 individuals and more than 1.7x faster in any case. Moreover, the Pthreads implementation achieves a speedup of around 5x for all experiments when using the 6 cores ($\simeq$ 85% of parallel efficiency). Therefore, EpistSearch finishes the analyses of the datasets between 9.5x and 11.3x faster than BOOST on the studied hex-core machine.

The characteristics of the datasets used for the evaluation of the GPU-based code are shown in Table 5. Due to the power of the GPUs, we use larger datasets. Furthermore, the variability of the percentage of SNP-pairs that pass each filter is even higher than in the CPU experiments: from 6.13% to 52.02% (8.5x) for the KSASA filter and from 0.0006% to 0.4% (667x) for the log-linear filter. Figures 2 and 3 compare the performance of GBOOST and EpistSearch working with the GTX 650Ti and GTX Titan GPU, respectively. Regarding EpistSearch, we provide the runtimes for GPU-only as well as for hybrid CPU/GPU execution. The results indicate the following trends:

- EpistSearch is always faster than GBOOST, either using the 6 CPU cores or not, and independently of the characteristics of the dataset and the GPU.
- In cases where a high percentage of pairs present interaction the improvement of performance achieved by EpistSearch is the most significant. For instance, in the experiment with 40K SNPs and 25,600 individuals EpistSearch is more

Table 5. Percentage of pairs that pass the KSASA and log-linear filters in the GPU experiments.

Num. Inds. →	**6400**		**12800**		**25600**	
Num. SNPs →	**40K**	**160K**	**40K**	**160K**	**40K**	**160K**
KSASA	20.27	6.13	35.49	7.03	52.02	9.35
log-linear	110×10^{-4}	6×10^{-4}	800×10^{-4}	7×10^{-4}	4000×10^{-4}	12×10^{-4}

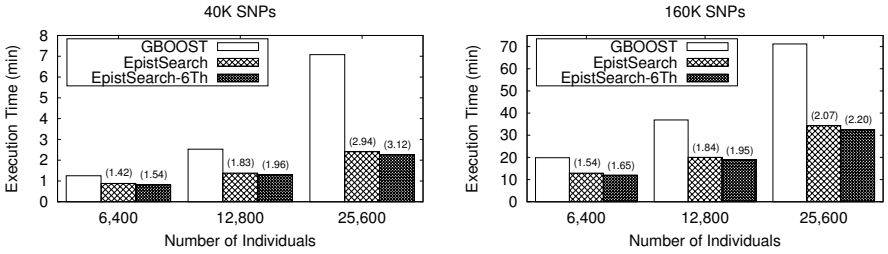

Fig. 2. Execution times of GBOOST and EpistSearch (with and without 6 additional CPU threads) on the GTX 650 Ti GPU. Speedups compared to GBOOST are shown in brackets.

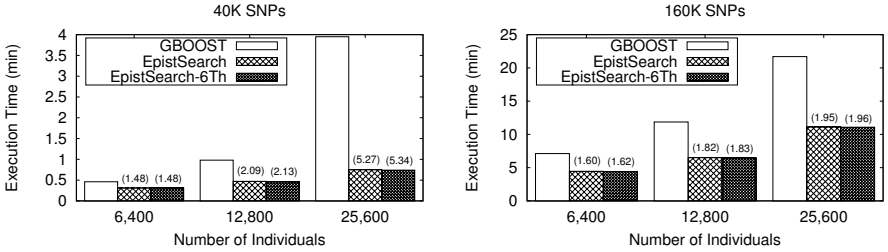

Fig. 3. Execution times of GBOOST and EpistSearch (with and without 6 additional CPU threads) on the GTX Titan GPU. Speedups compared to GBOOST are shown in brackets.

than 3x and 5x faster than GBOOST on the GTX 650 Ti and the GTX Titan GPUs, respectively.

- The speedup obtained by our tool compared to GBOOST in the other cases is always between 1.4x and 2.2x on both GPUs.
- EpistSearch runs between 2.8x and 3.2x faster on the GTX Titan than on the GTX 650 Ti GPU, while the improvement of GBOOST is only between 1.7x and 3.0x. It means that EpistSearch shows better scalability on larger number of SMs.
- The hybrid GPU/CPU combination consistently improves performance compared to the GPU-only execution. However, the improvement is relatively small (around 1.1x faster for the GTX 650 Ti GPU and 1.03x faster for the

GTX Titan GPUs) since the speedups of the GPU compared to the CPU are relatively high (40x for GTX 650 Ti and 122x for GTX Titan).

Finally, we have also applied EpistSearch and GBOOST to analyze a real-world dataset obtained from the WTCCC project. This datasets contains valuable information with cases of seven common human diseases: bipolar disorder, coronary artery disease, "Crohn disease", hypertension, rheumatoid arthritis, type 1 diabetes and type 2 diabetes. The project provides information about 3,000 shared controls and 2,000 cases per disease. Table 6 compares the runtime of EpistSearch, GBOOST and other two GPU-based tools (EpiGPU [22] and SHEsisEPI [23]) when analyzing the bipolar disorder disease (500,000 SNPs and 5,000 individuals) on different GPUs. Some results are obtained from the publications of the corresponding authors. Besides the execution time, we also show performance in terms of millions of evaluated SNP-pairs per second. Again, EpistSearch is faster than GBOOST on all GPUs. Although results for EpiGPU and SHEsisEPI must be treated carefully since the comparison is done over different architectures, we can infer that they are significantly slower than EpistSearch (as mentioned in Section 1, even slower than GBOOST).

Table 6. Performance comparison of different tools when looking for epistasis in a dataset with 500,000 SNPs and 5,000 samples. Results obtained from the publications of the corresponding authors are marked with (*).

Tool	Architecture	Time	Speed (10^6 tests per second)
EpistSearch	GTX Titan + 6 Intel Core i7	42 m	49.81
EpistSearch	GTX Titan	43 m	49.04
GBOOST	GTX Titan	1 h 01 m	34.23
EpistSearch	GTX 650Ti + 6 Intel Core i7	1 h 48 m	19.29
EpistSearch	GTX 650Ti	1 h 57 m	17.81
GBOOST	GTX 650Ti	2 h 41 m	12.97
GBOOST*	GTX 285	2 h 43 m	12.81
EpiGPU*	GTX 580	2 h 55 m	11.90
SHEsisEPI*	GTX 285	27 h	1.29

6 Conclusions

We have presented EpistSearch, a tool to search for epistasis between SNP-pairs in a fast manner taking advantage of CPU and GPU parallelism. The results produced by this tool can help to find genetic expressions for multiple common human diseases. Similar to BOOST and its GPU variant (GBOOST), which are currently two of the fastest and most popular available tools, EpistSearch is based on a definition of interaction via logistic regression models. Although our tool outputs the same list of pairs with epistasis than BOOST for all the experiments included in this paper (thus, providing the same accuracy), EpistSearch has been optimized by calculating less elements of the contingency tables and by applying

a novel preliminary filter. Therefore, EpistSearch uses a three-stage approach where only the simplest (but less precise) filter is applied to all the SNP-pairs. The most precise and most computationally expensive filters are only applied to the pairs that were not discarded by the preliminary test. In addition, an inter-task hybrid CPU-GPU parallelism has been implemented using Pthreads and CUDA in order to concurrently work on both multicore CPUs and GPUs.

We have also compared the performance of EpistSearch to BOOST and GBOOST on a hex-core modern machine with two available GPUs using simulated and real datasets. This experimental evaluation shows that EpistSearch is consistently faster in all the experiments, even though the characteristics of the input datasets are very different. For CPU computation, our tool obtains a speedup higher than 2x compared to BOOST using the same resources (only one CPU core) and it is able to accelerate the computation up to 11.3x by exploiting the 6 cores of the machine. Moreover, depending on the characteristics of the dataset of SNPs, EpistSearch obtains a speedup of more than 3x and 5x on a GTX 650 Ti and a GTX Titan GPU, respectively.

As future work, we will extend EpistSearch so it can work with a larger number of SNPs, even if they do not fit in the GPU memory. Furthermore, we will develop a multiGPU version.

References

1. Maher, B.: Personal Genomes: the Case of the Missing Heritability. Nature **456**(7218) (2008) 18–21
2. Moore, J.H., Asselbergs, F.W., Williams, S.M.: Bioinformatics Challenges for Genome-Wide Association Studies. Bioinformatics **26**(4) (2010) 445–455
3. Cordell, H.J.: Detecting Gene-Gene Interactions that Underlie Human Diseases. Nature Reviews Genetics **10**(6) (2009) 392–404
4. Zhao, J., Jin, L.: Test for Interaction Between Two Unlinked Loci. The American Journal of Human Genetics **78**(1) (2006) 15–27
5. Purcell, S., et al: PLINK: a Tool Set for Whole-Genome Association and Population-Based Linkage Anlyses. The American Journal of Human Genetics **81**(3) (2007) 559–575
6. Wellcome Trust Case Control Consortium. `http://www.wtccc.org.uk/` (Last visit: January 2014)
7. Consortium, T.W.T.C.C.: Genome-Wide Association Study of 14,000 Cases of Seven Common Diseases and 3,000 Shared Controls. Nature **447**(7145) (2007) 661–678
8. Yang, C., He, Z., Wan, X., Yang, Q., Xue, H., Yu, W.: SNPHarvester: a Filtering-Based Approach for Detecting Epistatic Interaction in Genome-Wide Association Studies. Bioinformatics **25**(4) (2009) 504–511
9. Wan, X., Yang, C., Yang, Q., Xue, H., Tang, N.L., Yu, W.: Predictive Rule Inference for epistatic Interaction Detection in Genome-Wide Association Studies. Bioinformatics **26**(1) (2010) 30–37
10. Wan, X., Yang, C., Yang, Q., Xue, H., Tang, N.L., Yu, W.: BOOST: A Fast Approach to Detecting Gene-Gene Interactions in Genome-Wide Case-Control Studies. The American Journal of Human Genetics **87**(3) (2010) 325–340

11. Bi, J., Gelernter, J., Sun, J., Kranzler, H.R.: Comparing the Utility of Homogeneous Subtypes of Cocaine Use and Related Behaviors with DSM-IV Cocaine Dependence as Traits for Genetic Association Analysis. American Journal of Medical Genetics **165**(2) (2014) 148–156
12. Chu, M., et al: A Genome-Wide Gene-Gene Interaction Analysis Identifies an Epistatic Gene Pair for Lung Cancer Susceptibility in Han Chinese. Cancinogenesis **32**(3) (2014) 572–577
13. Milne, R.L., et al: A Large-Scale Assessment of Two-Way SNP Interactions in Breast Cancer Susceptibility Using 46,450 Cases and 42,461 Controls from the Breast Cancer Association Consortium. Human Molecular Genetics **23**(7) (2014) 1934–1946
14. Yung, L.S., Yang, C., Wan, X., Yu, W.: GBOOST: A GPU-Based Tool for Detecting Gene-Gene Interactions in Genome-Wide Case Control Studies. Bioinformatics **27**(9) (2011) 1309–1310
15. Piriyapongsa, J., Ngamphiw, C., Intarapanich, A., Kulawonganunchai, S., Assawamakin, A., Bootchai, C., Shaw, P.J., Tongsima, S.: iLOCi: a SNP Interaction Priorization Technique for Detecting Epistasis in Genome-Wide Association Studies. BMC Genomics **13**(Supl 7) (2012)
16. Goudey, B., Rawlinson, D., Wang, Q., Shi, F., Ferra, H., Campbell, R.M., Stern, L., Inouye, M.T., Ong, C.S., Kowalczyk, A.: GWIS - Model-Free, Fast and Exhaustive Search for Epistatic Interactions in Case-Control GWAS. BMC Genomics **14**(Supl 3) (2012)
17. Liu, Y., Wirawan, A., Schmidt, B.: CUDASW++ 3.0: Accelerating Smith-Waterman Protein Database Search by Coupling CPU and GPU SIMD Instructions. BMC Bioinformatics **14**(177) (2013)
18. Liu, Y., Schmidt, B.: CUSHAW2-GPU: Empowering Faster Gapped Short-Read Alignment Using GPU Computing. IEEE Design & Test of Computers (in press)
19. POSIX Threads Programming. `https://computing.llnl.gov/tutorials/pthreads/` (Last visit: January 2014)
20. NVIDIA Developer CUDA Zone. `https://developer.nvidia.com/category/zone/cuda-zone` (Last visit: January 2014)
21. genomeSIMLA Webpage. `http://chgr.mc.vanderbilt.edu/genomeSIMLA/genomeSIMLA/Introduction.html` (Last visit: January 2014)
22. Hemani, G., Theocharidis, A., Wei, W., Haley, C.: EpiGPU: Exhaustive Pairwise Epistasis Scans Parallelized on Customer Level Graphic Cards. Bioinformatics **27**(11) (2011) 1462–1465
23. Hu, X., Liu, Q., Zhang, Z., Li, Z., Wang, S., He, L., Shi, Y.: SHEsisEpi, a GPU-Enhanced Genome-Wide SNP-SNP Interaction Scanning Algorithm, Efficiently Reveals the Risk Genetic Epistasis in Bipolar Disorder. Cell Research **20**(7) (2010) 854–857