Using Evolutionary Computing on Consumer Graphics Hardware for Epistasis Analysis in Human Genetics

Nicholas A. Sinnott-Armstrong
Dartmouth Medical School
One Medical Center Drive
Lebanon, NH, USA 03756

Casey S. Greene
Dartmouth Medical School
One Medical Center Drive
Lebanon, NH, USA 03756

Jason H. Moore
Dartmouth Medical School
One Medical Center Drive
Lebanon, NH, USA 03756
Jason.H.Moore@dartmouth.edu

ABSTRACT

Biological systems are both complex and robust. Because of this epistasis, or gene-gene interactions, are thought to be a ubiquitous component of common human diseases. Unfortunately, due to the non-linear nature of these interactions, detecting and characterizing epistasis requires algorithms which are combinatorial in complexity. One such algorithm is Multifactor Dimensionality Reduction (MDR). Expert knowledge guided evolutionary computing wrappers around MDR have previously been shown to be a powerful way to efficiently analyze datasets for interactions. Evolutionary computing can effectively address some of the challenges these datasets present. Unfortunately examining the statistical significance of results requires permutation testing, which increases the computation requirements by a factor of 1000. Here we implement an expert knowledge guided ant system on graphics processing units (GPUs) and show that the GPU implementation makes the rigorous statistical analysis of large datasets practical.

Categories and Subject Descriptors

J.3 [**Life and Medical Sciences**]: biology and genetics, health

General Terms

Algorithms, Performance

Keywords

Evolutionary Computing, GPU, Genetics

1. INTRODUCTION

New technologies in human genetics facilitate the measurement of more than 10^6 DNA sequence variations across the genome. The challenge in human genetics has shifted from the measurement of these variations to the development of computational and statistical tools capable of identifying genetic variations that play a role in the initiation, progression and severity of common human diseases such as breast cancer. Unfortunately this endeavor will be difficult due to epistasis; that is, non-additive gene-gene interactions. Epistasis is believed to be a ubiquitous component of the genetic architecture of common human diseases.

Two approaches have previously been taken with multifactor dimensionality reduction (MDR) to address this problem. The first involves selecting a subset of the space to search. Previously Greene et al. [1] have shown that an expert knowledge guided ant colony optimization (ACO) approach can successfully discover good solutions despite examining only a small subset of the search space. The second, used by Sinnott-Armstrong et al. [2], involves accelerating the MDR analysis itself using high performance non-traditional computing platforms such as graphics processing units (GPUs).

Both approaches alone provide substantial decreases in analysis time but moving from developing good models to assessing statistical significance requires permutation testing. Permutation testing, the method used to assess significance, necessitates an additional 1000 fold increase in computation time. To address this issue we develop an expert knowledge guided ACO implementation for GPUs to further improve performance and make the statistical analysis feasible.

2. ALGORITHMS

2.1 Novelty

While both the expert knowledge guided ACO approach and an MDR implementation for GPUs have already been developed, the combination of both necessitates changes in the MDR implementation. For the previous GPU implementation of MDR, the architecture relied on using a grid structure for executing across all combinations of attributes and the indices were stored in registers. For the ant colony approach, a different thread access structure was needed. Here the grid indices are offsets into an array of random numbers instead of actual attribute indices.

2.2 Efficiency

The algorithm is divided into a few parts, each of which has a different efficiency. The weight summing, in which all of the pseudo-probabilities resulting from offsetting the scores of previous generations are added together to scale, is very efficient and is based on the fast parallel reduction whitepaper released by NVIDIA. It exploits coherent reads, half-warp synchronization, unrolled loops, and many other structures which lead to very high efficiency. The same is true of most of the reduction steps within the MDR algorithm itself. Roulette wheel selection is run on a single

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.

thread and distributed through a shared variable and the advantages are substantial over a CPU selection process.

2.3 GPU-side

Both ACO and the MDR components are implemented on the GPU. Only random number generation, reading of the dataset, and the collation of results from multiple GPUs are run on the CPU. The authors evaluated the NVIDIA Mersenne Twister random number generator but found that the Python implementation was sufficiently fast and reduced code complexity thus improving maintainability.

2.4 Elegance

Conceptually, we implement the previously described ACO approach for MDR which uses a pheromone to select attributes based on a combination of expert knowledge and results from previous iterations [1]. The implementation of the ACO portions of the code are designed to be modular and use the map-reduce methodology almost exclusively.

2.5 Portability

The implementation uses PyCUDA which is supported on Windows, GNU/Linux, and Mac OS X machines with any 8800 series or higher NVIDIA GPU. Early GPUs which do not support atomic operations require a flag to ensure compatibility.

2.6 Suitability

On the GPU, decisions about shared memory coherency and constant memory are required. In these cases, the code has been rigorously evaluated in multiple different scenarios and the most efficient has been selected. Furthermore, asynchronous execution support is levied and thus minimal CPU overhead is required while executing.

3. SPEED

3.1 Speedup

Three different execution patterns were tested, which utilize one, three, or six GPUs. The results are shown below in Table 1. The highest speedup was with the largest number of ants, and it increased performance by a factor of twenty over the standard CPU solution.

3.2 Resources

The bulk of the code runs in a single kernel `Bucketd` which has an occupancy of 50% according to the CUDA Occupancy Calculator. The main limiting factor in the occupancy of the main kernel is the shared memory, since the shared memory usage increases linearly with the number of threads.

3.3 Scalability

The solution scales very well. It runs on multiple GPUs on a single workstation (tested up to 6), and should allow for seamless network execution although this feature is still under development. Furthermore this implementation should work on GPUs with any number of multiprocessors.

4. EVOLUTIONARY COMPUTATION

4.1 Utility

These results show that GPUs and evolutionary computation (EC) can be combined to make feasible the analysis of large biomedical datasets for tasks that were previously considered computationally prohibitive. EC, particularly in the area of human genetics, has shown promise for solving hard problems important for human health. The combination of EC and GPUs can greatly improve our understanding of common human diseases.

4.2 Practicality

The previously described ACO approach reduces computation time requirements by examining smaller portions of the dataset effectively but this presents a trade off between computation time and the portion of the search space that can be examined. By using GPUs, we can increase the portion of the search space that can be examined while reducing the amount of time necessary for the analysis. On CPUs the discovery of a good solution is not computationally prohibitive. Researchers, however, often want to know not only what the best solution was but if it was statistically significant. Assessing significance with permutation testing requires randomizing the data 1000 times and the re-evaluating these new datasets. On the CPUs this would require 7277 minutes – an entire work week! On six GPUs only 385 minutes would be required.

4.3 Science

In human genetics the challenge is both to develop predictive genetic models and to show rigorously that these models are able to significantly explain an individual's disease risk. For MDR, this requires permutation testing, which is computationally expensive. By increasing the range of datasets to which MDR can be effectively applied we improve our ability to examine and understand epistasis and its role in common human diseases.

5. BENCHMARK RESULTS

Config	Updates	Time 1 (s)	Time 2 (s)	Time 3 (s)
6 GPU	3600	23.004	21.503	24.740
3 GPU	1800	16.285	16.705	16.039
1 GPU	600	8.974	9.031	8.965
8 CPU	3600	442.000	419.516	448.396
8 CPU	1800	208.087	227.936	226.286
8 CPU	1200	144.662	143.523	139.819
8 CPU	600	54.297	75.257	47.998

6. DISCUSSION AND CONCLUSIONS

We implemented ACO for epistasis analysis in human genetics on GPUs. We found that this approach greatly improved the feasibility of permutation testing and thus the assessment of statistical significance.

7. ACKNOWLEDGMENTS

The authors would like to thank Fabio Cancare and Peter Andrews for code review.

8. REFERENCES

[1] C. S. Greene, B. C. White, and J. H. Moore. Ant colony optimization for genome-wide genetic analysis. *Lecture Notes in Computer Science*, 5217:37–47, 2008.

[2] N. A. Sinnott-Armstrong, C. S. Greene, F. Cancare, and J. H. Moore. Accelerating epistasis analysis in human genetics with consumer graphics hardware. *BMC Research Notes*, in press.