A heuristic method for simulating open-data of arbitrary complexity that can be used to compare and evaluate machine learning methods*

Jason H. Moore, Maksim Shestov, Peter Schmitt, Randal S. Olson
Institute for Biomedical Informatics, University of Pennsylvania, D202 Richards Building, 3700 Hamilton Walk, Philadelphia, PA 19104
Email: jhmoore@upenn.edu

A central challenge of developing and evaluating artificial intelligence and machine learning methods for regression and classification is access to data that illuminates the strengths and weaknesses of different methods. Open data plays an important role in this process by making it easy for computational researchers to easily access real data for this purpose. Genomics has in some examples taken a leading role in the open data effort starting with DNA microarrays. While real data from experimental and observational studies is necessary for developing computational methods it is not sufficient. This is because it is not possible to know what the ground truth is in real data. This must be accompanied by simulated data where that balance between signal and noise is known and can be directly evaluated. Unfortunately, there is a lack of methods and software for simulating data with the kind of complexity found in real biological and biomedical systems. We present here the Heuristic Identification of Biological Architectures for simulating Complex Hierarchical Interactions (HIBACHI) method and prototype software for simulating complex biological and biomedical data. Further, we introduce new methods for developing simulation models that generate data that specifically allows discrimination between different machine learning methods.

Keywords: simulation, machine learning, open data.

1. Introduction

Simulation plays an important role in the development of computational and statistical methods because the ground truth is known. This allows the power and false-positive rate of methods to be evaluated and compared. Further, simulation allows features of the data such as size and complexity to be varied to evaluate method robustness. An important criticism of simulation is that the models used may not fully represent the complexity of both the noise structure and signal in the data. This is because the nature of the true signals in data derived from experimental or observational studies of biological or biomedical systems is usually not known. This is particularly true in genetics and genomics where we have barely scratched the surface of measuring all the components that might influence phenotypic variation and, for those measures that exist, have focused our analysis primarily on univariate effects. This sparse slice of possible etiological factors and simplistic analytical approaches have yielded a number of simulation methods that are based parametric statistical methods such as logistic regression or probabilistic methods such as penetrance functions. The goal of the present study is to develop heuristic methods for the discovery of complex biological systems models that can be used to simulate more realistic data. We focus here on the simulation of

* This work is supported by National Institutes of Health grants LM012601, AI116794, and DK112217.

genotypic and phenotypic data in samples derived from human populations that can facilitate the development of methods for genetic association studies and precision medicine.

Methods for simulating genetic data in human population-based studies fall into two general categories. The first set of methods are focused on generating patterns of genetic variation that might be found in human populations. The goal here is to approximate the allele and genotype frequencies that are expected in a human population and the correlation structure of the variants as shaped by selection and recombination. Forward-time simulators of human populations such as simuPOP [1-2], GenomeSIMLA [3-5] and SFS_CODE [6] can be used for this purpose. Genetic data simulated is this way can then be used to simulate phenotypes using a statistical or computational model. Simple additive effects can be simulated using a linear regression model or more complex genetic effects such as gene-gene interactions can be simulated using methods and software such as Epi2Loc [7] or GAMETES [8]. Although useful, these tools don't explicitly build their models using a framework that approximates the hierarchical complexity of biological systems. It is our working hypothesis that the simulation of data using biologically-realistic genotype-phenotype relationships will improve method development by more closely mimicking the hierarchical complexity of human health or model systems.

To address this concern, we previously introduced the Heuristic Identification of Biological Architectures for simulating Complex Hierarchical Interactions (HIBACHI) method and prototype software for simulating complex biological and biomedical data [9]. This approach combines a biological hierarchy, a flexible mathematical framework, a liability threshold model for defining disease endpoints, and a simple stochastic search strategy for identifying high-order gene-gene interaction models of disease susceptibility. HIBACHI allows the explicit definition of a biological framework for the propagation of genetic effects organized in gene regulatory regions, coding regions, noncoding regions, and interacting components such as genes for transcription factors and other regulatory sequences such as microRNAs or long noncoding RNAs. The product at the gene level is a quantitative trait (e.g. protein) that can then be used with a liability model to simulate disease status. Alternatively, multiple traits can be simulated from different sets of genetic variation at the gene level and then combined with additional functions to simulate a higher-order physiologic trait at the level of a pathway or system. Traits from multiple systems could be used to simulate one or more anatomical traits. HIBACHI provides the flexibility to map multiple genotypes to multiple phenotypes through hierarchical biological systems of any complexity. It is our working hypothesis that data simulated in this manner will be more biologically realistic and thus more useful for the evaluation of computational and statistical methods.

The prototype HIBACHI algorithm and software developed by Moore et al. [9] used a fixed biological architecture with mathematical functions that connected genotype with phenotype. The goal of the present study was to extend HIBACHI to include heuristic identification of both the wiring of the biological model and the mathematical functions that create the genotype to phenotype relationship. Because of the extensive size of the search space we chose to use stochastic search for model discovery. Further, due to the modular nature of HIBACHI models we selected genetic programming (GP) as an initial search algorithm because of its flexible representation of models as expression trees and because it inherently explores combinations of model subcomponents through its recombination operator. We describe the stochastic search engine using GP in detail and then

provide an example application that focuses on the discovery of genetic models that produce data that explicitly differentiate the performance of different machine learning algorithms. The results demonstrate the usefulness of HIBACHI with GP as a stochastic search engine for the evaluation of machine learning methods. Methods and software like this will play an important role in the post-genomics data science era focused on understanding the complexity of genotype-phenotype relationships with the end goal of precision medicine.

2. Methods

There are five components to our Heuristic Identification of Biological Architectures for simulating Complex Hierarchical Interactions (HIBACHI) simulation method. The first is the metaphor for the hierarchical biological framework that transmits information from genotype at the DNA sequence level through biomolecular interactions at the gene, cell, and pathway levels to a clinical endpoint. The second is the mathematical framework that generates the genotype to phenotype relationship or pattern. The third is the liability threshold model that is used to define disease status. The fourth is the genetic programming (GP) methods for the discovery of high-order models. The final component is an open-source python-based software package distributed via GitHub for simulating multiple data sets. We describe each of these in turn. The first three components are descriptions included in the work by Moore et al. [9] and repeated with some minor updates here for completeness. The last two describe new components to the method and software.

2.1. *A biology-based framework for simulation of complex biological systems*

The goal of this component is to provide a biological framework or scaffold to serve as a metaphor for genetic variants and their phenotypic relationships propagated through a hierarchical set of mathematical functions. The prototype for HIBACHI developed by Moore et al. [9] used a fixed architecture that was based on genetic effects at the gene level. We describe this framework first and then discuss how this relates to the new approach that uses GP to discover both the wiring diagram and the mathematical functions. The initial HIBACHI framework (see Figure 1) started with protein-coding gene (i.e. mRNA gene) with a single non-synonymous genetic variant that is assumed to change an amino acid. Upstream of the mRNA gene is a promoter with a single regulatory variant and an enhancer with a single regulatory variant. Also included in our initial framework are two genes that code for transcription factors that bind to the regulatory region. We included a protein-coding variant in the gene that codes for each transcription factor. We also included a single variant in a microRNA gene that participates in post-translational regulation. In total, this structure allowed for six genetic variants (coded 0, 1, 2) all influencing a protein product as a quantitative trait. In addition, we included an environmental factor (coded -2, -1, 0, 1, 2) to allow for non-genetic variation in the phenotypic values. It is important to note that this particular biological framework was a preliminary proof of concept. The goal of the present study is to allow this framework to vary as part of the search for models meeting certain objectives using GP. The metaphor still holds but the new GP-based systems allows for much greater flexibility in the size and shape of the models being generated. Other metaphors such as electronic health record (EHR) data could also be used here.

2.2. *A mathematical framework for simulation of complex patterns*

The goal of this component is to provide a flexible mathematical framework for combining features to produce an endpoint. Using the genetics metaphor, genotypic and non-genotypic values to produce phenotypic values. Each biology-based locus feeds into a mathematical function whose result is carried forward to the next function. For example, one transcription factor locus combines with the enhancer locus through a function whose result then combines with the second transcription factor. The result of this operation combines with the locus at the promoter. This result combines with the coding variant in the gene. This result combines with the microRNA locus. This result combines with the environmental factor to produce a protein product. Thus, the protein expression value is dependent on mathematical functions of the six loci and the environmental factor. This produces a distribution with several to millions of possible phenotypic values for most combinations of functions that can then be used with the liability threshold model described below to generate disease status. Cases and controls can then be sampled from this distribution or the continuous output can be used directly as a quantitative trait.

For each run the user can specify a set of mathematical function to use to build the models. Examples of basic functions include addition, subtraction, multiplication, division, modulus, and modulus-2. Logical functions include greater than, less than, AND, OR, and XOR. Bitwise functions include bitwise AND, bitwise OR, and bitwise XOR. Unary functions include absolute value, NOT, factorial, left and right. Large functions include power, log, permute, and choose. Miscellaneous functions include minimum and maximum.

2.3. *A liability threshold model for biology-based simulation*

We use a liability threshold model to simulate disease from the distribution of phenotypic values generated from the genotypic values and mathematical functions as described above. The user can select the liability threshold to achieve a particular disease prevalence. More details about the liability model are provided by Moore et al. [9].

2.4. *Genetic programming for model representation and stochastic search*

We selected genetic programming (GP) as our stochastic search engine for several reasons. First, GP uses binary expression trees that are a convenient data structure for representing HIBACHI models. This makes the models very easy to manipulate and evaluate computationally. Flexible representation is a known advantage of GP [10] Second, GP uses a recombination operator that explicitly swaps subcomponents of expression trees to generate variability in the solutions as part of its iterative search process. This is appealing because HIBACHI models are hierarchical in nature with modular subcomponents representing different biological processes such as transcription factor binding, miRNA regulation of transcription, etc. The ability to mix and match these genomic modules facilitates the development of new models that meet a simulation objective. An introduction to GP is provided by Poli et al. [11] in an open-access book for those seeking additional details of the method.

A key to GP search is the fitness function that specifies the value or quality of a particular set of mathematical functions and their wiring (i.e. the model) represented as an expression tree.

For the evaluation of this approach, we used the performance of pairs of machine learning algorithms as the primary fitness criteria (see below). In addition, we also use as an objective the complexity of the model with the idea that simpler models that meet the data objective are better (i.e. to be minimized). We balance these different objectives using Pareto optimization. We discuss below an application of HIBACHI to the discovery of models that can be used for evaluating machine learning methods.

2.5. *An open-source HIBACHI software package*

The HIBACHI software presented here was programmed entirely in Python. The stochastic search elements used the open-source Distributed Evolutionary Algorithms in Python (DEAP) framework available on GitHub. We used the Python-based scikit-learn machine learning library to carry out all analyses (scikit-learn.org). HIACBHI is available as open-source on GitHub (github.com/epistasislab/hibachi).

2.6. *Discovery and evaluation of HIBACHI models for evaluating machine learning methods*

Our goal is to use HIBACHI to discover models that generate data for which one machine learning algorithms perform better compared to another. We chose to focus on logistic regression, decision trees, and random forests as three commonly used methods. All machine learning was implemented using the open-source scikit-learn library in Python. For each model and dataset generated by HIBACHI we used pairs of machine learning algorithms with default settings to perform the analysis. The balanced accuracy of each analysis was reported and used as multiple objectives through Pareto optimization to maximize the performance of one method while minimizing the performance of the second method. All combinations of methods were evaluated. The difference in performance between the two methods is also combined with a fitness objective that attempts to minimize the complexity of the models in terms of the number of mathematical functions that are used.

3. Results

The HIBACHI method and software allows for simulation of complex datasets with specific properties indicated in the multi-objective fitness function of the genetic programming (GP) algorithm used for model representation and stochastic search. In order to simulate datasets with varying machine learning performance we used a three-objective fitness function, first objective tries to increase the difference in performance of two machine learning methods, second objective optimizes the performance of one of the two methods, and the third objective decreases the complexity of the underlying mathematical function that generates dataset labels.

Performance of machine learning methods was measured by the average area under the curve of the testing data ROC curve of 10 random 80/20 train/test data splits. The simulated datasets consist of 2000 samples with 10 features per sample with an even 50/50 split of cases vs. controls. Each feature is an integer from a set of {0, 1, 2} which corresponds to an encoding of genetic variants. GP algorithm was ran for 1000 generations with 1000 individuals evaluated at each generation.

Increasing generation number would only marginally improve results due to a strong drop off in fitness improvements past 200 generations.

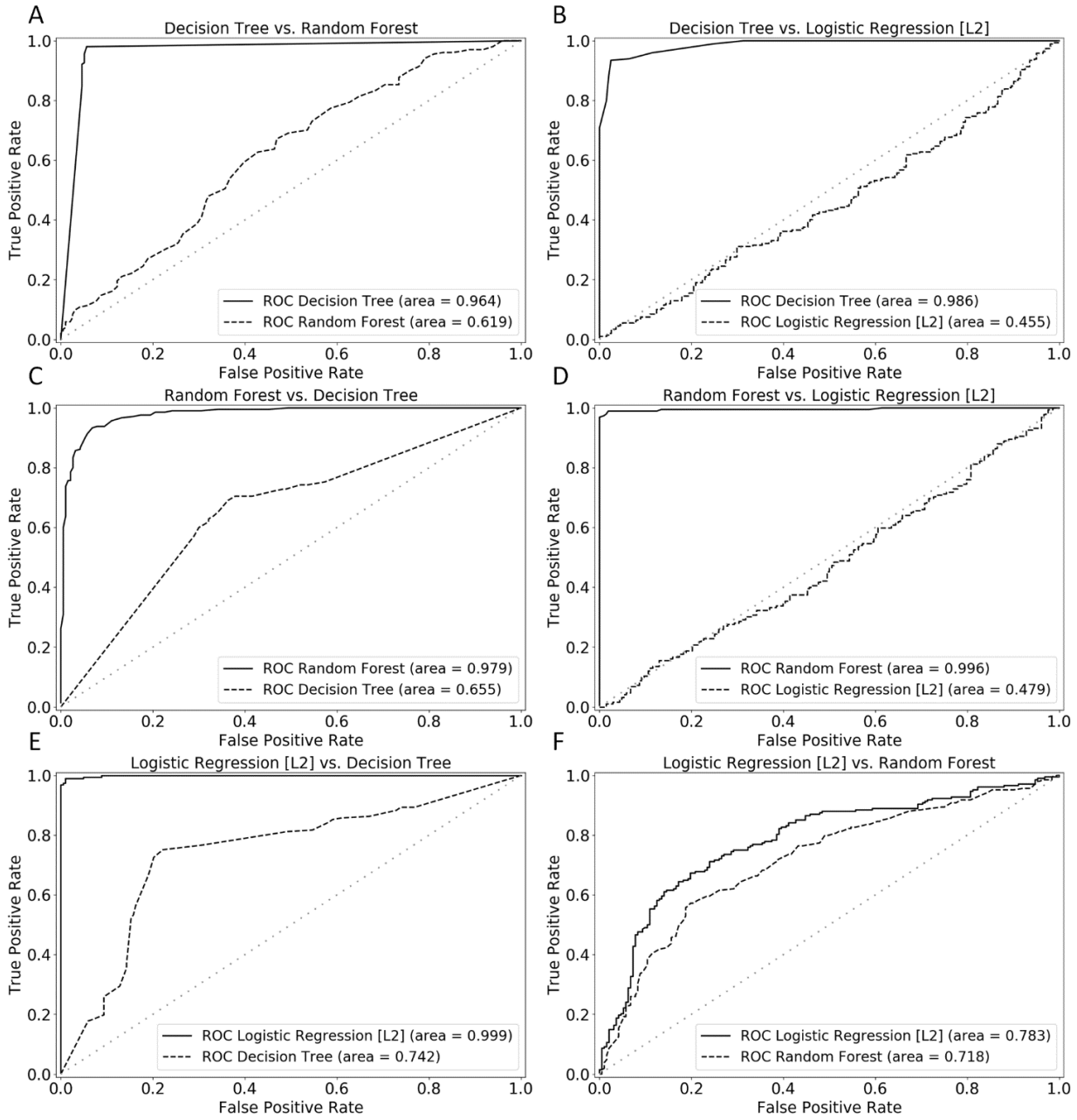

Fig. 1. ROC curves of datasets generated using HIBACHI with a three-objective fitness function. (A) ROC curve Decision Tree vs. Random Forest. (B) ROC curve Decision Tree vs. Logistic Regression. (C) ROC curve Random Forest vs. Decision Tree. (D) ROC curve Random Forest vs Logistic Regression. (E) ROC curve Logistic Regression vs. Decision Tree. (F) ROC curve Logistic Regression vs. Random Forest

We were able to generate synthetic datasets that have tailored performance for any given machine learning methods. Figure 1 shows the ROC curves of datasets generated with HIBACHI using 3 different machine learning methods that include a linear method – L2 penalized logistic regression with default parameters, non-linear method – Decision Tree with default parameters, no

depth limit, and a non-linear ensemble method – Random Forest with default parameters, no depth limit, and 100 estimators. The performance of non-linear vs. linear methods (Figure 1B&D) is as expected, once higher order interactions are introduced via XOR or other non-linear operator in the underlying mathematical function the performance drops to random performance. Figure 2 shows the function tree that gives a perfect performance for the decision tree algorithm and random performance for logistic regression. The reverse where we try to optimize a linear method vs. non-linear (Figure 1E&F), the performance difference is not as significant and the non-linear method never approaching random performance. Most of the operators in the linear vs. non-linear mathematical function are addition and subtraction. Mathematical function of the non-linear vs ensemble methods (Figure 1A) shows a similar performance to linear vs. non-linear method optimizations with no unique operators.

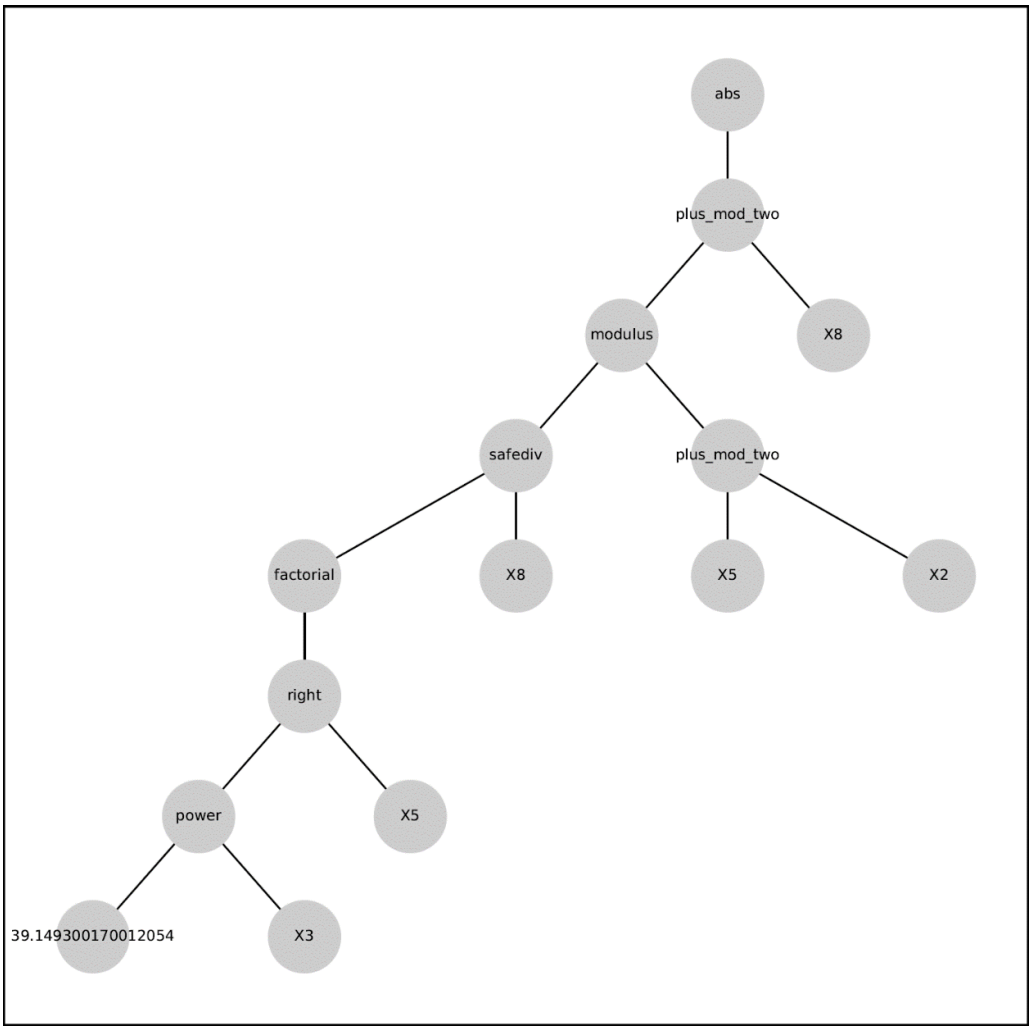

Fig. 2. Mathematical function tree that generates that optimizes decision tree performance (test set ROC AUC of ~1) and minimizes logistic regression performance (test set ROC AUC of ~0.5). X# corresponds to one of the 10 features.

4. Discussion

The generation and open sharing of simulated data is an important part of the artificial intelligence and machine learning development and evaluation process. Unfortunately, there is a lack of methods and software for generating data with the complexity that we often observe in biological and biomedical systems. To address this concern, we previously developed the Heuristic Identification of Biological Architectures for simulating Complex Hierarchical Interactions (HIBACHI) method and prototype software for simulating complex biological and biomedical data [9]. This approach combines a biological hierarchy, a flexible mathematical framework, a liability threshold model for defining disease endpoints, and a simple stochastic search strategy for identifying high-order gene-gene interaction models of disease susceptibility. We present here an extension of HIBACHI that improves its flexibility for generating models of different complexity and for generating models that can be used to evaluate and compare machine learning algorithms.
We also introduce a new Python-based software package as open-source.

The results demonstrate that it is possible to use HIBACHI to discover mathematical models that generate data for which logistic regression, decision trees, and random forests perform differently. As expected, it was quite easy to simulate data for which decision trees and random forests perform dramatically better than logistic regression and difficult to simulate data for which logistic regression does better than random forests. HIBACHI was able to generate data that revealed significant performance differences between other method contrasts. These results demonstrate the value of HIBACHI for using simulation to develop and test machine learning methods.

There are several possible directions for future studies with HIBACHI. First, HIBACHI models can easily be integrated together to build hierarchical models that might more closely reflect the hierarchy of real systems. For example, multiple HIBACHI models could be generated one for each gene in a genetic system with the output representing the continuous distribution of a protein product. Protein products could be combined with additional functions to produce the output of a biochemical system. Biochemical system output could be combined with additional functions to produce the output from a physiological system. These could be combined to produce a liability distribution for health and disease. Second, HIBACHI could be used to generate electronic health record (EHR) data to facilitate the development of methods for the rapidly growing field of clinical informatics. Good method for simulating EHR data are not available. Third, HIBACHI could be used to generate comprehensive sets of open-access data of differing size, shape, and complexity to serve as benchmark suites for method development. Finally, other fitness functions could be added to allow HIBACHI to be used for any number of simulation problems such as simulating data that resembles the patterns a real dataset. This opens the door to using HIBACHI for generating pseudo-simulated data resembling real data thus overcoming privacy and security concerns.

An important limitation of this approach is that it does not explicitly define an effect size for the simulated signal. Although not explicit, this could be overcome by making an effect size estimate part of the fitness function that is used to generate models through stochastic search. For example, you could evaluate models based on how close the subsequent machine learning results are to a target classification or regression error. Smaller errors would generate stronger signals and higher errors would generate weaker signals. An additional limitation is that the method is computationally

complex due the nature of the stochastic search algorithms. Fortunately, this approach can be easily parallelized for cloud and cluster computing. Despite these limitations, HIBACHI provides a vehicle for the flexible simulation of open data for research in the biological and biomedical sciences.

5. Acknowledgments

We would like to thank Mr. Peter Andrews and Dr. Jeff Kiralis for their work on the prototype HIBACHI method and software that led to the new methods presented in this paper. We would also like to thank the Genomics and Computational Biology (GCB) graduate group at the University of Pennsylvania for their generous support of Mr. Maksim Shestov.

References

1. B. Peng and C. I. Amos, *Bioinformatics*. **24(11)**, 1408 (2008)
2. B. Peng and M. Kimmel, *Bioinformatics*. **21(18)**, 3686 (2005)
3. S. Dudek, A. A. Motsinger, D. Velez, S. M. Williams and M. D. Ritchie, *Pacific Symposium Biocomputing*. 499 (2006)
4. T. L. Edwards, W. S. Bush, S. D. Turner, S. M. Dudek, E. S. Torstenson, M. Schmidt, E. Martin and M. D. Ritchie, *Lecture Notes in Computer Science*, **4973**, 24 (2008).
5. M. D. Ritchie and W. S. Bush, *Adv Genet*, **72**, 1 (2010)
6. R. D. Hernandez, *Bioinformatics*. **24(23)**, 2786 (2008)
7. R. K. Walters, C. Laurin and G. H. Lubke, *Twin Res Hum Genet*. **17(4)**, 272 (2014)
8. R. J. Urbanowicz, J Kiralis, N. A. Sinnott-armstrong, T. Heberling, J. M. Fisher and J. H. Moore, *BioData Min*. **5(1)**, 16 (2012)
9. J. H. Moore, R. Amos, J. Kiralis and P. C. Andrews, *Genet Epidemiol*. **39(1)**, 254 (2015)
10. D. Ashlock, Evolutionary Computation for Modeling and Optimization. New York, NY: Springer Science Business Media, Inc. Print (2006)
11. R. Poli, W. B. Langdon, N. F. McPhee and J. R. Koza, A field guide to genetic programming. S. l.: Lulu Press. Print (2008)