Using Expert Knowledge to Guide Covering and Mutation in a Michigan Style Learning Classifier System to Detect Epistasis and Heterogeneity

Ryan J. Urbanowicz, Delaney Granizo-Mackenzie, and Jason H. Moore

Computational Genetics Laboratory, Department of Genetics
Dartmouth Medical School, Lebanon, NH, USA
{ryan.j.urbanowicz,jason.h.moore}@dartmouth.edu
http://www.epistasis.org/

Abstract. Learning Classifier Systems (LCSs) are a unique brand of multifaceted evolutionary algorithms well suited to complex or heterogeneous problem domains. One such domain involves data mining within genetic association studies which investigate human disease. Previously we have demonstrated the ability of Michigan-style LCSs to detect genetic associations in the presence of two complicating phenomena: epistasis and genetic heterogeneity. However, LCSs are computationally demanding and problem scaling is a common concern. The goal of this paper was to apply and evaluate expert knowledge-guided covering and mutation operators within an LCS algorithm. Expert knowledge, in the form of Spatially Uniform ReliefF (SURF) scores, was incorporated to guide learning towards regions of the problem domain most likely to be of interest. This study demonstrates that expert knowledge can improve learning efficiency in the context of a Michigan-style LCS.

Keywords: Expert Knowledge, Learning Classifier System, Genetics, Epistasis, Heterogeneity, Evolutionary Algorithm, Mutation, Covering.

1 Introduction

Learning Classifier Systems (LCSs) [1] are a rule-based class of algorithms which combine machine learning with evolutionary computing and other heuristics to produce an adaptive system. We focus on Michigan-style LCSs (M-LCSs) which uniquely make decisions using the entire rule population giving them the ability to perform on-line learning, form niches, and adapt. They have been applied to many problems including behavior modeling, function approximation, classification, and data mining [1].

Scalability and learning speed have been synonymous targets for improvement in the LCS literature [2,3,4,5,6] largely in the context of Pittsburgh-style LCSs (P-LCSs) due to their inherent limitations in this area. However these considerations are just as important for M-LCS algorithms, especially in the context of large-scale, high dimensional problems.

1.1 The Problem Domain: Human Genetics

One domain where these shortcomings clearly impact the utility of LCS is within human genetics. Single nucleotide polymorphisms (SNPs) are single loci in the DNA sequence where alternate nucleotides (i.e. alleles) are observed between members of a species or between paired chromosomes in an individual. In a typical genetic association study, researchers look for differences in SNP allele frequencies between a group of individuals with the disease of interest, and a matched group healthy controls.

Despite the rising quality and abundance of genetic data, epidemiologists continue to struggle with the connection of disease phenotypes to reliable genetic and environmental markers. While strategies seeking single locus associations (i.e. main effects) are often sufficient to address diseases which follow Mendelian patterns of inheritance, their application to diseases characterized as complex has yielded limited success [7,8]. Epistasis and heterogeneity have been recognized as phenomena which complicate the epidemiological mapping of genotype to phenotype [9]. In the present context, epistasis simply refers to attribute interaction. *Heterogeneity*, referring to either *genetic* heterogeneity (locus and allelic) or *environmental* heterogeneity, occurs when individual (or sets of) attributes are independently predictive of the same phenotype (i.e. class).

As proof of principle, M-LCSs were applied to the detection and modeling of simulated epistatic and heterogeneous genetic disease associations [10]. These evaluations identified the strengths and weaknesses of M-LCS on these types of complex, noisy problems. To address the shortcoming of knowledge discovery in M-LCSs we previously introduced an analysis pipeline with statistical and visualization-guided strategies for rule population interpretation [11]. In order to explicitly identify heterogeneity we introduced a strategy to link instances in the dataset to respective heterogeneous subgroups using attribute tracking and feedback [12]. To date, we have a functional LCS algorithm able to concurrently detect patterns of association with epistasis and heterogeneity. In this work, we turn our focus to improving the efficiency and scalability of this strategy.

The present study explores the adaptation of expert knowledge, previously utilized in the context of other evolutionary algorithms, to improve algorithm efficiency [13,14,15,16]. Here, we uniquely introduce expert knowledge to an M-LCS algorithm and develop the strategies to utilize it. To achieve this goal we: (1) derive expert knowledge from Spatially Uniform ReliefF (SURF) [17], (2) adopt a logistic function to reliably transform any set of expert knowledge scores into a set of attribute respective probabilities, (3) utilize these probabilities to intelligently guide covering and mutation operators within M-LCS towards regions of the problem domain most likely to be of interest.

2 Methods

In this section we describe (1) the M-LCS algorithm and run parameters used in this investigation, (2) SURF, the selected source of our EK, (3) logistic transformation of the EK values into probabilities, (4) the incorporation of EK into the covering and mutation mechanisms, and (5) our experimental evaluation of the proposed mechanisms.

2.1 Learning Classifier System: UCS

M-LCSs, often varying widely from version to version, generally possess four basic components; (1) a population of rules or classifiers, (2) a performance component that assesses how well the population of rules collectively explain the data, (3) a reinforcement component that distributes the rewards for correct prediction to each of the rules in the population, and (4) a discovery component that uses different operators to discover new rules and improve existing ones. Learning progresses iteratively, relying on the performance and reinforcement components to drive the discovery of better rules. For a complete LCS introduction and review, see [1].

The sUpervised Classifier System (UCS) [18], is a M-LCS based largely on the very successful XCS algorithm [19], replacing reinforcement learning with supervised learning. UCS was designed specifically to address single-step problem domains such as classification and data mining, displaying particular promise when applied to attribute interaction and heterogeneity in [10].

For evaluation purposes we implement expert knowledge into a Python encoding of the UCS algorithm [10]. We utilize mostly default run parameters with the exception of 200,000 learning iterations, a population size of 1600, tournament selection, uniform crossover, subsumption, attribute mutation probability = 0.04, crossover probability = 0.8, and $\nu = 1$. ν has been described as a "constant set by the user that determines the strength [of] pressure toward accurate classifiers" [20], and is typically set to 10 by default. A low ν was used to place less emphasis on high accuracy in this type of noisy problem domain, where 100% accuracy is only indicative of over-fitting. While we run each algorithm for a maximum of 200,000 iterations, we also stop and evaluate the systems after 10,000, 50,000, and 100,000 iterations. Also, as in [10], we employ a quaternary rule representation, where for each SNP attribute, a rule specifies genotype as (0, 1, or 2), or instead generalizes with "#", a character which implies that the rule doesn't care about the state of that particular attribute. The implementation described above is available on request (*ryanurbanowicz@gmail.com*) and will be posted on the LCS and GBML Central webpage.

2.2 Expert Knowledge from Spatially Uniform ReliefF (SURF)

Expert knowledge (EK) is an external source of information providing, in this context, a measure of attribute quality. In this study, the external measure was statistical, but it could just as easily be biological. The usefulness of EK is entirely dependent on it's quality. There are many statistical and computational methods for determining the quality of attributes. We selected a method that is capable of identifying attributes that predict class primarily through dependencies or interactions with other attributes. To this end we selected Spatially

Uniform ReliefF (SURF) [17] as the source of EK. SURF estimates the quality of attributes through a nearest neighbor algorithm that selects neighbors (case or control instances) within an automatically determined distance threshold. Weights, or quality estimates, for each attribute are estimated based on whether the nearest neighbor (nearest hit) of a randomly selected instance from the same class and the nearest neighbor from the other class (nearest miss) have the same or different values. In addition to SURF, [17] also evaluated Tuned ReliefF (TuRF) [21] which was designed for human genetics application. TuRF systematically removes attributes that have low quality estimates so that the ReliefF values of the remaining attributes can be re-estimated. SURF and TuRF could be combined in future work to derive EK scores, but in the present study we exclusively utilize SURF to generate EK scores for every simulated training dataset described in section 2.5. Note that the SURF run time was negligible compared to that of UCS: requiring a matter of seconds to run.

2.3 Transformation of Expert Knowledge into Probabilities

This section describes our application of the logistic function in order to transform raw EK scores into normalized probability values. We start with a set of raw EK values $K \subset \mathbb{R}$. We know neither the range nor distribution of the EK values. The only requirement is that greater importance should translate to a larger EK score. Let $n = |K|$ and $k_i \in K | 1 \leq i \leq n$ be the ith score.

We use the logistic function to transform the raw values into selection probabilities. Namely,

$$\ell_{\alpha,\beta}(x) = \frac{1}{1 + e^{-(\alpha + \beta x)}}$$

.

Before applying this function we must determine values for constants α and β. First, the user specifies a range constant d. We have chosen $d = 0.4$ which yields an output range of $(0.5 - d = d_l = 0.1, 0.5 + d = d_u = 0.9)$. This range ensures that the attribute with the lowest EK score retains at least a 10% chance to be specified, while the attribute with the highest EK score has no greater than a 90% chance to be specified. Next, the user must specify c, which is the resulting sum of probabilities in the transformed set. This is useful when one wants to guarantee that a selection algorithm, given the set of probabilities, returns a certain number of individuals on average. In this study we want selection to choose 50% of the attributes during covering, therefore we set $c = 10$ since datasets each have 20 attributes. Lastly, the user must specify how many digits of precision (set to 5 here) we use when calculating α in step 5. EK transformation progresses in five steps as follows:

Step 1: Compute range $r = max(K) - min(K)$.

Step 2: Shift scores such that the lowest score is at minimum zero. If $min(K) < 0$ then add $abs(min(K))$ to every EK score.

Step 3: Compute β such that the logistic function's slope 'nicely' occupies the data range, i.e. $\ell_{\alpha,\beta}(-r/2) = d_l$ and $\ell_{\alpha,\beta}(r/2) = d_u$. Since α does not affect slope we set $\alpha = 0$ while calculating β. We solve for β after some simple algebra with the following:

$$\beta = 2\ln(\frac{1-d_l}{d_l})/r$$

Step 4: Compute an initial guess for α such that $\ell_{\alpha,\beta}(min(K)) = d_l$. Since α simply shifts the function left or right we only need to move the curve such that the minimum EK score is transformed to d_l. We solve for this initial α_0 using the following equation derived again with some simple algebra:

$$\alpha_0 = -\beta(min(K) + r/2)$$

Step 5: In order to find α such that the transformed probabilities sum to c we iteratively search for an appropriate value of α using the Newton-Raphson method. For our purposes β is a constant. We applied differential calculus to obtain:

$$\alpha_j = \alpha_{j-1} - \frac{\sum_{i=0}^{n} \ell_{\alpha,\beta}(k_i) - c}{\sum_{i=0}^{n} \ell'_{\alpha,\beta}(k_i)}$$

We iterate this equation until $\alpha_j = \alpha_{j-1}$ with respect to the digits of precision. At this point applying the logistic function to the input scores (K) using the computed parameters, α and β, will produce a set of probabilities that sum to the desired c.

2.4 Expert Knowledge Applied to Covering and Mutation

Once EK-based probabilities has been generated for all attributes, we incorporated these as weights to guide LCS learning. To achieve this, we apply these probabilities to both covering and mutation operators within M-LCS. The covering operator is responsible for population initialization, as well as ensuring that a matching rule exists for a given data instance within each learning iteration. Previously, EK has been successfully applied to population initialization in genetic programming [16]. In the context of LCS, EK probabilities drive the specialization (state is important) or generalization (state is not important) of attributes within rules. Having chosen a c of 10, rule generated via covering will tend to have half of the 20 attributes specified, and half generalized. The standard covering mechanism gives each attribute a 50% chance of being specialized. With the incorporation of EK, attributes with higher EK scores (likely to be useful) will have a higher probability of being specified, and vice-versa.

The mutation mechanism is a discovery component of the M-LCS. When activated, mutation traditionally randomly permutes an element of the rule such that if it had been specified it becomes generalized and vise-versa. Previously, EK has been successfully applied to mutation in genetic programming [15]. Here, we apply EK probabilities to mutation, such that if an attribute is selected for 'possible' mutation, the probability of mutation is equal to the EK probability for that attribute. Specified attributes with high EK-scores will be less likely to be generalized while generalized attribute with high EK-scores will more likely to be flipped to specified. The opposite is true for attribute with low EK-scores.

In this study we evaluate the utilization of EK into UCS over four trials. The trials include the following scenarios: (1) UCS algorithm without EK (UCS), (2) UCS with EK applied to covering only (UCS-EK-Cov), (3) UCS with EK applied to mutation only (UCS-EK-Mut), and (4) UCS with EK applied to both covering and mutation (UCS-EK-Both).

2.5 Data Simulation and Analysis

We evaluate EK using simulated datasets which concurrently model heterogeneity and epistasis as they might appear in a SNP gene association study of common complex disease [10,22]. All data sets were generated using a pair of distinct, two-locus epistatic interaction models, both utilized to generate instances (i.e. case and control individuals) within a respective subset of each final data set. Each two-locus epistatic model was simulated without Mendelian/main effects, as a penetrance table as in [10]. Due to the computational demands of LCSs, this study limited its evaluation to 3 heterogeneity/epistasis model combinations. For simplicity the minor allele frequency of each predictive attribute was set to 0.2, a reasonable assumption for a common complex disease SNP. The three model combinations included a pair of models with a heritability of either (0.1, 0.2, or 0.4). We considered model architectural "difficulties" of both "easy" and "hard" [23]. Balanced datasets simulated from these models were generated as having four different sample sizes (200, 400, 800, or 1600) and a heterogeneous mix ratio of either (50:50 or 75:25) (e.g. 75% of instances were generated from one epistatic model, and 25% were generated from a different one). Twenty replicates of each dataset were analyzed and 10-fold cross validation (CV) was employed to measure average testing accuracy and account for over-fitting. Together, a total of 48 data set configurations (*3 Model Combos* x *4 Sample Sizes* x *2 Ratios* x *2 Difficulties*), and a total of 960 data sets (20 random seeds each) were simulated. With 10-fold CV, 9600 runs of each of the four UCS trials were completed.

For each run we track the following statistics; training accuracy, testing accuracy, generality, macro population size, the power to find both underlying models, the power to find at least one underlying model, the power to correctly rank attribute co-occurrence [11], and run time. Power is a reflection of our ability to reliably mine knowledge from the evolved rule population. Co-occurrence power is a reflection of our ability to distinguish heterogeneous models. Each of these values represent an average over the 10 CV runs.

Statistical comparisons were made using the Wilcoxon signed-rank tests due to a lack of normality in the value distributions. All statistical evaluations were completed using R. Comparisons were considered to be significant at $p \leq 0.05$.

3 Experimental Results and Discussion

This analysis of EK spans over a spectrum of complex datasets, with evaluations taken at four different iteration intervals, and examines a variety of M-LCS statistics in order to compare the performance of UCS, UCS-EK-Cov, UCS-EK-Mut, and UCS-EK-Both. Table 1 summarizes averages and identifies significant differences for all run statistics over all simulated datasets between our four experimental investigations after either 10,000 or 200,000 iterations. Averages after 10,000 iterations reveal the impact of EK very early on in the learning process, while averages after 200,000 give a better sense of it's impact after the learning curve has started to level off.

Table 1. Comparing UCS with EK implementations after either 10,000 or 200,000 learning iterations each averaged over all simulated datasets. Arrows indicate a significant increase or decrease when compared to UCS.

10,000 Iterations							
Statistics	UCS	UCS-EK-Cov	*p*	UCS-EK-Mut	*p*	UCS-EK-Both	*p*
Train Accuracy	0.8268	0.8439	↑ **	0.7918	↓ **	0.8301	↑ **
Test Accuracy	0.5909	0.6112	↑ **	0.6283	↑ **	0.6278	↑ **
Both Power	0.15	0.2677	↑ **	0.2458	↑ **	0.2969	↑ **
Single Power	0.5427	0.7281	↑ **	0.6885	↑ **	0.7302	↑ **
Co-Occur. Power	0.1531	0.1625	-	0.1469	-	0.0396	↓ **
Generality	0.7019	0.6543	↓ **	0.7351	↑ **	0.6580	↓ **
Macro Population	1371.82	1400.71	↑ **	1244.44	↓ **	1276.68	↓ **
Run Time (min)	1.61	1.79	↑ **	1.20	↓ **	1.43	↓ **
200,000 Iterations							
Statistics	UCS	UCS-EK-Cov	*p*	UCS-EK-Mut	*p*	UCS-EK-Both	*p*
Train Accuracy	0.8544	0.8546	-	0.8468	↓ **	0.8467	↓ **
Test Accuracy	0.6134	0.6141	-	0.6283	↑ **	0.6278	↑ **
Both Power	0.3115	0.3083	-	0.3927	↑ **	0.3990	↑ **
Single Power	0.7125	0.7156	-	0.7677	↑ **	0.7625	↑ **
Co-Occur. Power	0.2438	0.25	-	0.2302	-	0.2260	↓ *
Generality	0.7136	0.7138	↑ *	0.7535	↑ **	0.7533	↑ **
Macro Population	1317.09	1316.84	-	1173.03	↓ **	1172.21	↓ **
Run Time (min)	37.48	35.85	↓ **	29.66	↓ **	29.06	↓ **

– Not Sig.
* $p < 0.05$
** $p << 0.001$

Most notable after only 10,000 iterations, all three EK implementations show significant improvement in testing accuracy, and the power to find one or both underlying models supporting the hypothesis that EK can direct LCS towards important regions of the problem space to improve learning efficiency. After 200,000 iterations the advantages of using UCS-EK-Cov become hidden, as most observed statistics are comparable to those seen using UCS. This indicates that

UCS-EK-Cov speeds up learning, but given enough time, UCS is able to achieve similar performance. However, for UCS-EK-Mut, and UCS-EK-Both, we again observe significant improvements in testing accuracy, both, and single power even after 200,000 iterations. Additionally for these two implementations we observe increased rule generality, a smaller macro population size, and a decrease run time, all considered to be indicators of improved learning efficiency in this context. Figure 1A illustrates changing rule generality at different learning intervals for each implementation, while Figure 1B illustrates the same for macro population size. Increasing generality while maintaining or improving accuracy indicates that UCS is doing a better job focusing on attributes that will be valuable for making predictions on subjects it has not yet seen. Decreasing macro population size suggests that a smaller, more reliable, and applicable set of rules have been found by UCS. The only statistic which was not improved via EK incorporation was co-occurrence power. This is a logical finding given that EK operates globally during the learning process. Since co-occurrence power reflects the ability of the system to separate heterogeneous models, it makes sense that a global EK mutation pressure (applied uniformly to all rules) may reduce the systems overall ability to differentiate heterogeneity.

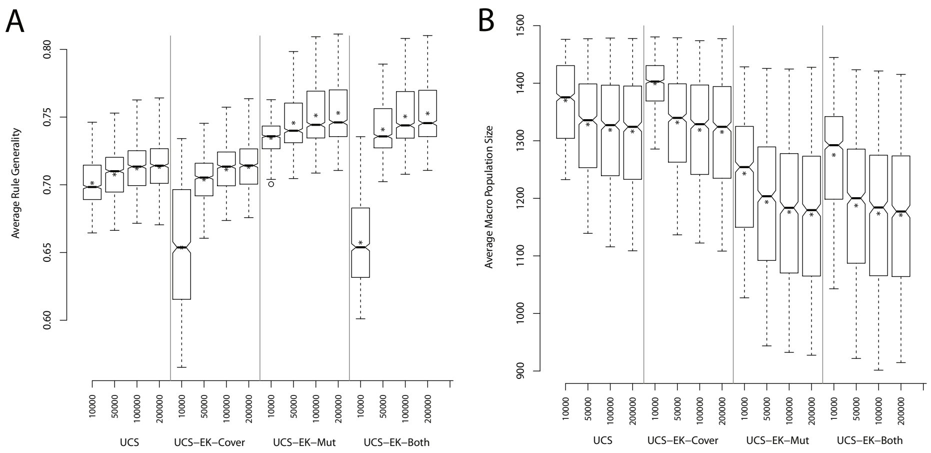

Fig. 1. (A) Comparing average rule population generality and (B) comparing average macro population size between UCS and UCS implementations utilizing EK. In both plots, the values on the x-axis indicate the number of completed learning iterations. Each box includes 960 observations. The star within each box plot indicates the average of those values.

4 Conclusions

The primary conclusion of this work is that EK may be successfully applied to an M-LCS algorithm to improve learning efficiency. We observe better algorithm performance when using EK after as few as 10,000 iterations. We have developed and evaluated a strategy for implementing EK using SURF scores as a source of EK, transforming any EK score source into usable probabilities, and incorporating these probabilities into covering and mutation mechanisms for the M-LCS. Overall, our findings support the inclusion of EK in M-LCS as a strategy for improving learning efficiency. However, in the context of potentially heterogeneous problems it may be better to (1) limit the use of EK to covering alone and (2) guide GA mechanisms such as mutation and crossover with local information as explored in [12] rather than with global information (i.e. EK). In future work we will extend this effort to consider the integration of EK and attribute feedback [12]. We will also extend these analysis to datasets with larger numbers of attributes in order to more directly evaluate improvements in scalability. This work supports the overall conclusion of related studies examining EK in the context of evolutionary algorithms [14,15,16]: that EK can be effective at pointing the algorithm towards attributes of greatest interest therefore facilitating the algorithm's ability to find "the genetic needle in the the genomic haystack".

Acknowledgments. This work was supported by the William H. Neukom 1964 Institute for Computational Science and by Award Number R25CA134286 from the National Cancer Institute. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Cancer Institute or the National Institutes of Health.

References

1. Urbanowicz, R., Moore, J.: LCSs: A Complete Introduction, Review, and Roadmap. Journal of Artificial Evolution and Applications 2009 (2009)
2. Bacardit, J., Goldberg, D.E., Butz, M.V., Llorà, X., Garrell, J.M.: Speeding-Up Pittsburgh Learning Classifier Systems: Modeling Time and Accuracy. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN VIII. LNCS, vol. 3242, pp. 1021–1031. Springer, Heidelberg (2004)
3. Bacardit, J., Stout, M., Hirst, J., Sastry, K., Llorà, X., Krasnogor, N.: Automated alphabet reduction method with evolutionary algorithms for protein structure prediction. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, pp. 346–353. ACM (2007)
4. Llorà, X., Sastry, K.: Fast rule matching for lcss via vector instructions. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 1513–1520. ACM (2006)
5. Bacardit, J., Burke, E., Krasnogor, N.: Improving the scalability of rule-based evolutionary learning. Memetic Computing 1(1), 55–67 (2009)
6. Franco, M., Krasnogor, N., Bacardit, J.: Speeding up the evaluation of evolutionary learning systems using gpgpus. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, pp. 1039–1046. ACM (2010)
7. Shriner, D., Vaughan, L., Padilla, M., et al.: Problems with genome-wide association studies. Science 316(5833) (2007) 1840c
8. Eichler, E., Flint, J., Gibson, G., Kong, A., Leal, S., Moore, J., Nadeau, J.: Missing heritability and strategies for finding the underlying causes of complex disease. Nature Reviews Genetics 11(6), 446–450 (2010)

9. Thornton-Wells, T., Moore, J., Haines, J.: Genetics, statistics and human disease: analytical retooling for complexity. TRENDS in Genetics 20(12), 640–647 (2004)
10. Urbanowicz, R., Moore, J.: The application of michigan-style lcss to address genetic heterogeneity and epistasis in association studies. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, pp. 195–202. ACM (2010)
11. Urbanowicz, R., Granizo-Mackenzie, A., Moore, J.: An Analysis Pipeline with Visualization-Guided Knowledge Discovery for Michigan-Style LCSs. IEEE CIM Special Issue on Computational Intelligence in Bioinformatics (2012)
12. Urbanowicz, R., Granizo-Mackenzie, A., Moore, J.: Instance-Linked Attribute Tracking and Feedback for Michigan-Style Supervised LCSs. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation (2012)
13. Jamshidi, M., et al.: Incorporating a-priori expert knowledge in genetic algorithms. In: Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA 1997, pp. 300–305. IEEE (1997)
14. Moore, J.H., White, B.C.: Exploiting Expert Knowledge in Genetic Programming for Genome-Wide Genetic Analysis. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN IX. LNCS, vol. 4193, pp. 969–977. Springer, Heidelberg (2006)
15. Greene, C.S., White, B.C., Moore, J.H.: An Expert Knowledge-Guided Mutation Operator for Genome-Wide Genetic Analysis Using Genetic Programming. In: Rajapakse, J.C., Schmidt, B., Volkert, L.G. (eds.) PRIB 2007. LNCS (LNBI), vol. 4774, pp. 30–40. Springer, Heidelberg (2007)
16. Greene, C.S., White, B.C., Moore, J.H.: Sensible initialization using expert knowledge for genome-wide analysis of epistasis using genetic programming. In: IEEE Congress on Evolutionary Computation, CEC 2009, pp. 1289–1296. IEEE (2009)
17. Greene, C.S., Penrod, N., Kiralis, J., Moore, J.: Spatially uniform relieff (surf) for computationally-efficient filtering of gene-gene interactions. BioData Mining 2(1), 1–9 (2009)
18. Bernadó-Mansilla, E., Garrell-Guiu, J.: Accuracy-based LCSs: models, analysis and applications to classification tasks. Evolutionary Computation 11(3), 209–238 (2003)
19. Wilson, S.: Classifier fitness based on accuracy. Evolutionary Computation 3(2), 149–175 (1995)
20. Orriols-Puig, A., Bernadó-Mansilla, E.: Revisiting ucs: Description, fitness sharing, and comparison with xcs. Learning Classifier Systems, 96–116 (2008)
21. Moore, J.H., White, B.C.: Tuning ReliefF for Genome-Wide Genetic Analysis. In: Marchiori, E., Moore, J.H., Rajapakse, J.C. (eds.) EvoBIO 2007. LNCS, vol. 4447, pp. 166–175. Springer, Heidelberg (2007)
22. Urbanowicz, R.J., Moore, J.H.: The Application of Pittsburgh-Style Learning Classifier Systems to Address Genetic Heterogeneity and Epistasis in Association Studies. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 404–413. Springer, Heidelberg (2010)
23. Urbanowicz, R., Kiralis, J., Fisher, J., Moore, J.: Predicting Difficulty in Simulated Genetic Models: Metrics for Model Architecture Selection. BMC Bioinformatics (submitted)