The χ-ary Extended Compact Classifier System: Linkage Learning in Pittsburgh LCS

Xavier Llorà[†], Kumara Sastry[‡], David E. Goldberg[‡], Luis delaOssa[§]

[†]National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, 1205 W. Clark Street, Urbana, IL 61801, USA
xllora@illigal.ge.uiuc.edu

[‡]Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 104 S. Mathews Avenue, Urbana, IL 61801, USA
{kumara,deg}@illigal.ge.uiuc.edu

[§]Departamento de Sistemas Informáticos, Universidad de Castilla-La Mancha, Campus Universitario s/n, 02071 Albacete, Spain
ldelaossa@dsi.uclm.es

ABSTRACT

This paper proposes a *competent* Pittsburgh LCS that automatically *mines* important substructures of the underlying problems and takes problems that were *intractable* with first-generation Pittsburgh LCS and renders them *tractable*. Specifically, we propose a χ-ary extended compact classifier system (χeCCS) which uses (1) a competent genetic algorithm (GA) in the form of χ-ary extended compact genetic algorithm, and (2) a niching method in the form restricted tournament replacement, to evolve a set of maximally accurate and maximally general rules. The results clearly show that linkage exists in the multiplexer problem which needs to be accurately discovered and efficiently processed in order to solve the problem in *tractable* time. The results also show that in accordance with the facetwise models from GA theory, the number of function evaluations required by χeCCs to successfully evolve an optimal rule set scales exponentially with the number of address bits (building block size) and quadratically with the problem size.

Categories & Subject Descriptors

I.2.6 [Artificial Intelligence]: Learning–Concept Learning.

General Terms

Algorithms, Design, Theory.

Keywords

Learning Classifier Systems, Competent Genetic Algorithms, Extended Compact Classifier System

1. INTRODUCTION

One of the daunting challenges in genetics based machine learning (GBML) is the principled integration of *competent* genetic algorithms (GAs) [6]—GAs that solve boundedly difficult problems quickly, reliably, and accurately—for evolving maximally general, maximally accurate rules. Despite their demonstrated scalability—on both problems that are difficult on a single as well as hierarchical level—limited studies have used competent GAs in GBML [3, 4]. Butz *et al* [3] studied techniques to identify problem structures in XCS. In a subsequent study, Butz *et al* [4] investigated methods to identify and effectively process *building blocks* in XCS. Specifically, they used a two-level composed problem (n-parity m-multiplexer), and used competent GAs to identify global and local structure. Due to the binary encoding used, the low-level structure (parity) was successfully identified and processed, but left the high-level structure (multiplexer) unidentified.

Modeling on the evolution of estimation of distribution algorithms (EDAs) [15], we approached the integration of competent GAs into the Pittsburgh LCS in a principled manner by first considering a simplest EDA—compact genetic algorithm [9]—and developed the compact classifier system (CCS) [13]. Our initial analysis with CCS showed that it is able to evolve maximally general and maximally accurate rule sets while guaranteeing a compact representation of the rule set space. However, the scalability analysis of CCS revealed that it requires exponential number of function evaluations to fully solve the multiplexer problem. The exponential scalability is due two factors: (1) *Attribute independence assumption* that leads CCS to require an exponentially large population size to evolve some of the rules belonging to the optimal rule set, which strongly hints to the presence of interactions among the decision variables, and (2) *rule set formation by multiple cGA runs*, that is, CCS requires multiple cGA runs to assemble a rule set that solves the problem. Assuming that the probability of drawing a rule in a given cGA run is $1/\left|[O]\right|$ ($\left|[O]\right|$ the optimal rule set [12]), the number of runs required to ensemble an rule set increases exponentially with the number of rules in the optimal set [O].

In this paper, we remedy both drawbacks of CCS and propose a method that not only discovers necessary substructures, but also evolves a set of maximally general and maximally accurate rules simultaneously within the framework of Pittsburgh LCS. Specifically, we propose a χ-ary extended compact classifier system (χeCCS) which uses (1) a linkage-learning GA in the form of χ-ary extended compact genetic algorithm (χeCGA) [5, 16], and (2) a niching method in the form of restricted tournament replacement [10] to evolve a set of maximally general and maximally accurate rule set. Confirming CCS results, χeCCS results show that linkage does exist in the multiplexer problem, which needs to be discovered in order to evolve a rule set in *tractable* time. We show that in accordance with existing population-sizing models for EDAs, the population size required by χeCCS scales exponentially with the number of address bits (building block size) and linearly with the problem size (number of building blocks). Additionally, the number of function evaluations required by χeCCs to successfully evolve an optimal rule set scales exponentially with the number of address bits (building block size) and quadratically with the problem size, despite the exponential growth in the size of the optimal rule set.

The rest of the paper is structured as follows. We introduce χeCCS in section 2. In section 3, we summarize the initial analysis and results obtained using χeCCS. Finally, we summarize the work done and present key conclusions in section 4.

2. THE χ-ary EXTENDED COMPACT CLASSIFIER SYSTEM

The χ-ary extended compact classifier system (χeCCS) relies on a χ-ary extended compact genetic algorithm (χeCGA) [5, 16] to identify *building blocks* among the rules. As in CCS, χeCCS uses a default rule for close-world assumption, but represents the rules using a ternary encoding instead of the binary one used in CCS. The use of a χ-ary approach is to focus the linkage learning between the conditions of the rules. Whereas, a binary version would be misled and only group bits of a single condition together (low-level *building blocks*) [4]. Another key element to the evolution of a set of rules is the ability to provide proper niching capabilities—as already pointed out elsewhere by Bernadó-Mansilla et al. [1, 2].

The χ-ary extended compact genetic algorithm (χeCGA) [5, 16], is an extension of Harik's binary eCGA [11]. Unlike the original eCGA, χeCGA can handle fixed-length chromosomes composed of genes with arbitrary cardinalities (denoted by χ). As in the original eCGA, χeCGA is based on a key idea that the choice of a good probability distribution is equivalent to linkage learning. The measure of a good distribution is quantified based on minimum description length(MDL) models. The key concept behind MDL models is that given all things are equal, simpler distributions are better than the complex ones. The MDL restriction penalizes both inaccurate and complex models, thereby leading to an optimal probability distribution. The probability distribution used in eCGA is a class of probability models known as marginal product models (MPMs). MPMs are formed as a product of marginal distributions on a partition of the genes. MPMs also facilitate a direct linkage map with each partition separating tightly linked genes.

The χeCGA can be algorithmically outlined as follows:

1. Initialize the population with random individuals.
2. Evaluate the fitness value of the individuals
3. Select good solutions by using s-wise tournament selection without replacement [8].
4. Build the probabilistic model: In χeCGA, both the structure of the model as well as the parameters of the models are searched. A greedy search is used to search for the model of the selected individuals in the population.
5. Create new individuals by sampling the probabilistic model.
6. Evaluate the fitness value of all offspring
7. Replace the parental population (before selection) with the offspring population using restricted tournament replacement (RTR) [10]. We use RTR in order to maintaining multiple maximally general and maximally accurate rules as niches in the population.
8. Repeat steps 3–6 until some convergence criteria are met.

Three things need further explanation: (1) the fitness measure, (2) the identification of MPM using MDL, and (3) the creation of a new population based on MPM.

In order to promote maximally general and maximally accurate rules à la XCS [18], χeCCS compute the *accuracy* (α) and the *error* (ε) of an individual [14]. In a Pittsburgh-style classifier, the accuracy may be computed as the proportion of overall examples correctly classified, and the error is the proportion of incorrect classifications issued. Let n_{t+} be the number of positive examples correctly classified, n_{t-} the number of negative examples correctly classified, n_m the number of times that a rule has been matched, and n_t the number of examples available. Using these values, the *accuracy* and *error* of a rule r can be computed as:

$$\alpha(r) = \frac{n_{t+}(r) + n_{t-}(r)}{n_t} \qquad (1)$$

$$\varepsilon(r) = \frac{n_{t+}}{n_m} \qquad (2)$$

We note that the error (equation 2) only takes into account the number of correct positive examples classified. This is due to the close-world assumption of the knowledge representation which follows from using a default rule. Once the *accuracy* and *error* of a rule are known, the fitness can be computed as follows.

$$f(r) = \alpha(r) \cdot \varepsilon(r) \qquad (3)$$

The above fitness measure favors rules with a good classification accuracy and a low error, or maximally general and maximally accurate rules.

The identification of MPM in every generation is formulated as a constrained optimization problem,

$$\text{Minimize} \qquad C_m + C_p \qquad (4)$$

$$\text{Subject to}$$

$$\chi^{k_i} \leq n \quad \forall i \in [1, m] \qquad (5)$$

where C_m is the model complexity which represents the cost of a complex model and is given by

$$C_m = \log_\chi(n+1) \sum_{i=1}^{m} \left(\chi^{k_i} - 1\right) \quad (6)$$

and C_p is the compressed population complexity which represents the cost of using a simple model as against a complex one and is evaluated as

$$C_p = \sum_{i=1}^{m} \sum_{j=1}^{\chi^{k_i}} N_{ij} \log_\chi \left(\frac{n}{N_{ij}}\right) \quad (7)$$

where χ is the alphabet cardinality, m in the equations represent the number of BBs, k_i is the length of BB $i \in [1, m]$, and N_{ij} is the number of chromosomes in the current population possessing bit-sequence $j \in [1, \chi^{k_i}]$[1] for BB i. The constraint (Equation 5) arises due to finite population size.

The greedy search heuristic used in χ-eCGA starts with a simplest model assuming all the variables to be independent and sequentially merges subsets until the MDL metric no longer improves. Once the model is built and the marginal probabilities are computed, a new population is generated based on the optimal MPM as follows, population of size $n(1 - p_c)$ where p_c is the crossover probability, is filled by the best individuals in the current population. The rest $n{\cdot}p_c$ individuals are generated by randomly choosing subsets from the current individuals according to the probabilities of the subsets as calculated in the model.

One of the critical parameters that determines the success of χeCGA is the population size. Analytical models have been developed for predicting the population-sizing and the scalability of eCGA [17]. The models predict that the population size required to solve a problem with m building blocks of size k with a failure rate of $\alpha = 1/m$ is given by

$$n \propto \chi^k \left(\frac{\sigma_{BB}^2}{d^2}\right) m \log m, \quad (8)$$

where n is the population size, χ is the alphabet cardinality (here, $\chi = 3$), k is the building block size, $\frac{\sigma_{BB}^2}{d^2}$ is the noise-to-signal ratio [7], and m is the number of building blocks. For the experiments presented in this paper we used $k = |a|+1$ (where $|a|$ is the number of address inputs), $\frac{\sigma_{BB}^2}{d^2}$=1.5, and $m = \frac{\ell}{|I|}$ (where ℓ is the rule size).

3. RESULTS

We conducted a set of initial tests of the χeCCS to evaluated whether it is capable of: (1) identifing and exploit problem structure, and (2) co-evolving a set of rules in a single run. Specifically, we used the multiplexer problem [18] with 3, 6, 11, 20, and 37 inputs.

3.1 Substructure in the multiplexer

We begin by investigating whether χeCCS is able *mine* the substructures of the multiplexer problem. The results show that χeCGA does indeed discover important substructures, and the models for each of the multiplexer problems are shown in table 1. From the models shown in table 1, we can clearly see that the *building-block* size grows linearly with the number of address bits ($|a|$). Since the maximally accurate and maximally general rules specify $|a|$+1 positions in the rule, we can intuitively expect the building block size to grow with the problem size.

[1]Note that a BB of length k has χ^k possible sequences where the first sequence denotes be $00\cdots0$ and the last sequence $(\chi-1)(\chi-1)\cdots(\chi-1)$

Table 1: Illustrative models evolved by eCCS for different sizes of the multiplexer problem. The number in parenthesis shows the average number of maximally accurate and maximally general rules evolved after 10 independent runs.

3-input multiplexer (3)
`[0 2] [1] (3)`
6-input multiplexer (7)
`[0 3][1 4][2 5]`
11-input multiplexer (14)
`[0 4 5][1 6 10][2 7 8][3 9]`
20-input multiplexer (31)
`[0 1 3 4][2 10 11 15][5 13][6][7][8] [9 17][12 18][14][16][19]`
37-input multiplexer (38)
`[0 1 3 9 15][2 7 11 29 33][4 16 34] [5 20 21][6 8 12 14][10 25 26] [13 18 23 30][17 19 24 31][22 32] [27 35][28][36]`

3.2 Getting a set of rules

We now investigate the niching capability of RTR that permits the evolution of an optimal rule set. Before we present the number of rules evolved in χeCCS, we first calculate the total number of maximally accurate and maximally general rules that exist for a given multiplexer, given a default rule. That is, we can compute the number of rules in [O] [12] and the number of overlapping rules given the number of address inputs $|a|$ as follows:

$$size(|a|) = |[O]| + |[OV]| = 2^{|a|} + |a|\,2^{|a|-1} = (2+|a|)2^{|a|-1} \quad (9)$$

For the the 3-input, 6-input, 11-input, 20-input, and 37-input multiplexer, the total number of maximally accurate and maximally general rules is 3, 8, 20 ,48, and 118 respectively.

However, not all these rules are needed to assemble a rule-set that describes the target concept. For instance, a minimal ensemble is the one provided by $[O]$. The number of maximally accurate and maximally general rules evolved on an average in the ten independent are shown in table 1. The results clearly show that RTR does indeed facilitate the simultaenous evolution of the optimal rule set. We also investigated the scalability of the number of function evalutions required by χeCCS to evolve at least $|[O]|$ rules during a single run. We note that the size of $[O]$ grows exponentially with respect of the number of address inputs. The population size used was the one introduced in the previous section. Figure 1 shows the number of iterations required by χeCCS only grows linearly. Therefore, the number of function evaluations scale exponentially with the number of address bits (building-block size) and quadratically with the problem size (number of building blocks). This is despite the exponential growth in the size of the optimal rule set. Moreover, the scalability results are as predicted by the facetwise models developed for competent GAs.

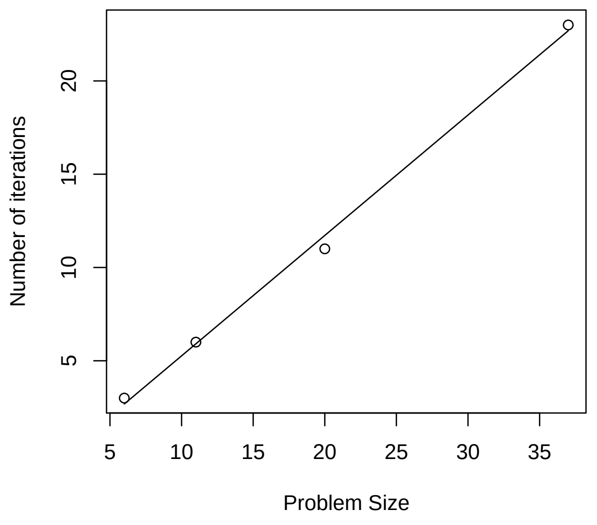

Figure 1: Number of iterations required to obtain at least $|[O]| = 2^{|a|}$ maximally accurate and maximally general rules. Results are the average of ten independent runs.

4. CONCLUSIONS

We have presented how linkage can be successfully identified and exploited to evolve a set of maximally general and maximally accurate rules using Pittsburgh-style learning classifier systems. We introduced the χ-ary extended compact classifier system (χeCCS) which uses (1) a χ-ary extended compact genetic algorithm (χeCGA), and (2) restricted tournament replacement to evolve a set of maximally accurate and maximally general rule set. The results show that linkage exists in the multiplexer problem—confirming CCS results—and also show that in accordance with the facetwise models from GA theory, the number of function evaluations required by χeCCs to successfully evolve an optimal rule set scales exponentially with the number of address bits (building block size) and quadratically with the problem size.

5. ACKNOWLEDGMENTS

This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant FA9550-06-1-0096, and the National Science Foundation under grant IIS-02-09199. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

6. REFERENCES

[1] E. Bernadó-Mansilla and J. M. Garrell-Guiu. MOLeCS: A MultiObjective Learning Classifier System. *Proceedings of the 2000 Conference on Genetic and Evolutionary Computation*, 1:390, 2000.

[2] E. Bernadó-Mansilla, X. Llorà, and I. Traus. *MultiObjective Machine Learning*, chapter MultiObjective Learning Classifier System, pages 261–288. Springer, 2005.

[3] M. V. Butz, P. L. Lanzi, X. Llorà, and D. E. Goldberg. Knowledge extraction and problem structure identification in XCS. *Parallel Problem Solving from Nature - PPSN VIII*, 3242:1051–1060, 2004.

[4] M. V. Butz, M. Pelikan, X. Llorà, and D. E. Goldberg. Extracted global structure makes local building block processing effective in XCS. *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, 1:655–662, 2005.

[5] L. de la Ossa, K. Sastry, and F. G. Lobo. Extended compact genetic algorithm in C++: Version 1.1. IlliGAL Report No. 2006013, University of Illinois at Urbana-Champaign, Urbana, IL, March 2006.

[6] D. E. Goldberg. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Norwell, MA, 2002.

[7] D. E. Goldberg, K. Deb, and J. H. Clark. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6:333–362, 1992. (Also IlliGAL Report No. 91010).

[8] D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5):493–530, 1989.

[9] G. Harik, F. Lobo, and D. E. Goldberg. The compact genetic algorithm. *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 523–528, 1998. (Also IlliGAL Report No. 97006).

[10] G. R. Harik. Finding multimodal solutions using restricted tournament selection. *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 24–31, 1995. (Also IlliGAL Report No. 94002).

[11] G. R. Harik, F. G. Lobo, and K. Sastry. Linkage learning via probabilistic modeling in the ECGA. In M. Pelikan, K. Sastry, and E. Cantú-Paz, editors, *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, chapter 3. Springer, Berlin, in press. (Also IlliGAL Report No. 99010).

[12] T. Kovacs. *Strength or Accuracy: Credit Assignment in Learning Classifier Systems*. Springer, 2003.

[13] X. Llorà, K. Sastry, and D. E. Goldberg. The Compact Classifier System: Motivation, analysis, and first results. *Proceedings of the Congress on Evolutionary Computation*, 1:596–603, 2005.

[14] X. Llorà, K. Sastry, D. E. Goldberg, A. Gupta, and L. Lakshmi. Combating user fatigue in iGAs: Partial ordering, support vector machines, and synthetic fitness. In *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, volume 2, pages 1363–1370, Washington DC, USA, 25-29 June 2005. ACM Press.

[15] M. Pelikan, F. Lobo, and D. E. Goldberg. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21:5–20, 2002. (Also IlliGAL Report No. 99018).

[16] K. Sastry and D. E. Goldberg. Probabilistic model building and competent genetic programming. In R. L. Riolo and B. Worzel, editors, *Genetic Programming Theory and Practise*, chapter 13, pages 205–220. Kluwer, 2003.

[17] K. Sastry and D. E. Goldberg. Designing competent mutation operators via probabilistic model building of neighborhoods. *Proceedings of the Genetic and Evolutionary Computation Conference*, 2:114–125, 2004. Also IlliGAL Report No. 2004006.

[18] S. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.