Do not Match, Inherit: Fitness Surrogates for Genetics-Based Machine Learning Techniques

Xavier Llorà, Kumara Sastry,
Tian-Li Yu, and David E. Goldberg
IlliGAL Report No. 2007011
March, 2007

Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 S. Mathews Avenue
Urbana, IL 61801
Office: (217) 333-2346
Fax: (217) 244-5705

Do not Match, Inherit: Fitness Surrogates for Genetics-Based Machine Learning Techniques

Xavier Llorà[1], **Kumara Sastry**[2], **Tian-Li Yu**[2], and **David E. Goldberg**[2]

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

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

March 27, 2007

Abstract

One benefit of using probabilistic model-building genetic algorithms is the possibility of creating cheap and accurate surrogate models. Learning classifier systems—and genetics-based machine learning in general—can greatly benefit from such surrogates which can replace the costly matching procedure of a rule against large data sets. In this paper we investigate the accuracy of such surrogate fitness function when coupled with the probabilistic models evolved by the χ-ary extended compact classifier system (χeCCS). We present results showing how functional alignment between the probabilistic model of χeCCS and the surrogate fitness is required. We also present a transformation of populations of rules based on the dependency structure matrix genetic algorithm (DSMGA) that allows building accurate models of overlapping *building blocks*—a necessary condition to accurately estimate the fitness of the evolved rules.

1 Introduction

A daunting challenge for learning classifier systems (LCS)—and genetics-based machine learning (GBML) in general—is the amount of time spent in the rule matching procedure required to estimate the fitness of a rule. Recently, Llorà and Sastry (2006) propose an efficient implementation of rule matching strategies that uses hardware accelerated vector operations. However, such efficient implementations do not get rid of the need to match candidate rules against the instance of a given data set. In recent years there has also been a renaissance of techniques for fitness inheritance in the genetic algorithms community, particularly in the estimation of distribution algorithms (EDAs). The goal is to build cheap surrogate fitness functions accurate enough to replace the calculation of expensive fitness functions. Sastry, Lima, and Goldberg (2006) proposed a new surrogate fitness based on substructural information and linear estimation.

The introduction of the χ-ary extended compact classifier system (χeCCS) by Llorà, Sastry, Goldberg, and de la Ossa (2006) showed that GBML approaches could greatly benefit from the use of scalable genetic algorithms (Goldberg, 2002). In this paper, we explore how surrogate fitness functions can be created to estimate the fitness of a rule to avoid matching the rule against the instances contained in a given data set. The fitness surrogate proposed by Sastry, Lima, and Goldberg (2006) uses the substructural information obtaining from the probabilistic model-building done by eCGA (Harik, Lobo, & Sastry, 2006), hence, it should be possible to build such surrogates based on the models evolved by χeCCS.

In this paper we present how surrogate fitness functions can be built using substructural information evolved by χeCCS. We explore how such surrogates behave on two widely used test problems in the GBML literature: the hidden `XOR` and the multiplexer problem. Both problems have different underlying properties. Actually, using both problems we are able to identify the need for a functional alignment between the model containing the substructural information and the functional form of the surrogate fitness function proposed. In another words, the substructural model evolved by χeCCS rely on non-overlapping *building blocks* (BBs)—accurate enough to solve the class of Boolean functions quickly, reliably, and accurately (Llorà, Sastry, Goldberg, & de la Ossa, 2006). However in a certain class of problems, such models may not align with the functional function of the surrogate proposed by Sastry, Lima, and Goldberg (2006)—which require an overlapping model to solve, for instance, the multiplexer problem. In this paper we also present how model-building techniques that evolve overlapping BBs—we used the dependency structure matrix genetic algorithm (DSMGA) by Yu, Goldberg, Yassine, and Chen (2003)—can be adapted to effectively build accurate models for rule sets and, hence, effectively use fitness inheritance schemes to GBML techniques.

The rest of this paper is structured as follows. Section 2 presents a general overview of the χeCCS. Then, section 3 reviews some related work on fitness inheritance for EDAs, and section 4 describes the surrogate fitness used in this paper. We conduct initial experiments—as reported in section 5—that show the need, in certain class of problems, to discover models able to express overlapping BBs. We review one of such method (DSMGA) in section 6, and what transformations may be required to apply such schemes to model rule set—section 7. Finally, we present the conclusions and further work for this paper in section 8.

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) (de la Ossa, Sastry, & Lobo, 2006; Sastry & Goldberg, 2003) 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*) (Butz, Pelikan, Llorà, & Goldberg, 2005). 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 and Garrell-Guiu (2000) and Bernadó-Mansilla, Llorà, and Traus (2005).

The χ-ary extended compact genetic algorithm (χeCGA) (de la Ossa, Sastry, & Lobo, 2006; Sastry & Goldberg, 2003), is an extension of Harik's binary eCGA (Harik, Lobo, & Sastry, 2006). 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 (Goldberg, Korb, & Deb, 1989).
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) (Harik, 1995). 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 (Wilson, 1995), χeCCS compute the *accuracy* (α) and the *error* (ε) of an individual (Llorà, Sastry, Goldberg, Gupta, & Lakshmi, 2005). 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:

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

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

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.

$$f(r) = \alpha(r) \cdot \varepsilon(r) \tag{3}$$

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. The goal is to minimize the *model complexity* and the *compress population complexity*. Further details may be found elsewhere (Harik, Lobo, & Sastry, 2006; Llorà, Sastry, Goldberg, & de la Ossa, 2006). 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 (Sastry & Goldberg, 2004). 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

$$n \propto \chi^k \left(\frac{\sigma_{BB}^2}{d^2}\right) m \log m, \tag{4}$$

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 (Goldberg, Deb, & Clark, 1992), 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).

As mentioned earlier, to assemble a rule set that describes the concept we need to maintain multiple maximally accurate and maximally general rules. Since we would like to maintain multiple maximally accurate, maximally general rules, we need an efficient niching method, that does not adversely affect the quality of the probabilistic models. Therefore, following previous studies in EDAs (Pelikan, 2005), we use restricted tournament replacement (RTR) (Harik, 1995). We note that a sub-structural niching method might be better than RTR in stably maintaining multiple niches (Sastry, Abbass, Goldberg, & Johnson, 2005), and it can be readily incorporated into the proposed algorithm. In RTR, each new offspring solution $\mathbf{x}$ is incorporated into the original population using the following three steps: (1) select a random subset $\mathbf{W}$ of size w (called window size) from the original population (before selection), (2) find the solution $\mathbf{y}$ in $\mathbf{W}$ that is most similar to $\mathbf{x}$ (in terms of Euclidean distance), and (3) make a tournament between $\mathbf{x}$ and $\mathbf{y}$ where $\mathbf{x}$ replaces $\mathbf{y}$ if it is better than $\mathbf{y}$. The parameter w is called window size, and a good rule of thumb for setting this parameter is $w = \min\{\ell\}$, where ℓ is the problem size (Pelikan, 2005). We note that the window size w affects the number of niches that can be maintained by RTR. That is increasing the window size can potentially increase the number of niches that can be maintained in the population and also increases the probability of maintaining the niches (Harik, 1995; Pelikan, 2005).

We note that the population size n, affects the success probability of maintaining *all* maximally general, maximally accurate rules, γ. In essence, RTR requires larger population sizes to maintain the global optima for longer time. This is a well understood phenomena of niching methods and has been analyzed by Mahfoud for fitness sharing (Mahfoud, 1994) and is applicable to RTR as well (Sastry, Abbass, Goldberg, & Johnson, 2005). The minimum population size required by RTR for maintaining at least one copy of all but one maximally general maximally accurate rules in the population is given by (Mahfoud, 1994; Sastry, Abbass, Goldberg, & Johnson, 2005)

$$n \propto \frac{\log\left[\left(1-\gamma^{1/t}\right)/n_{opt}\right]}{\log\left[\left(n_{opt}-1\right)/n_{opt}\right]} \tag{5}$$

where t is the number of generations we need to maintain all the niches, n_{opt} is the total number of maximally general maximally accurate rules.

3 Evaluation Relaxation in EDAs

In *evaluation relaxation*, an accurate, but computationally expensive fitness function—such as the matching procedure in the χeCCS—is replaced by a less accurate, but inexpensive surrogate function, and thereby the total number of costly fitness evaluations are reduced (Barthelemy & Haftka, 1993; Grefenstette & Fitzpatrick, 1985; Jin, 2005; Pelikan & Sastry, 2004; Sastry, 2001; Sastry, Pelikan, & Goldberg, 2004; Smith, Dike, & Stegmann, 1995). The low-cost, less-accurate fitness estimate can either be (1) *exogenous*, as in the case of approximate fitness functions (Barthelemy & Haftka, 1993; Jin, 2005; Llorà, Sastry, Goldberg, Gupta, & Lakshmi, 2005), where, external means are used to develop the fitness estimate, or (2) *endogenous*, as in the case of *fitness inheritance* (Smith, Dike, & Stegmann, 1995) where, some of the offspring fitnesses are estimated based on fitness of parental solutions.

Sastry, Pelikan, and Goldberg (2004) proposed a fitness inheritance method for EDAs, specifically for eCGA—a similar method was proposer for the Bayesian optimization algorithm (BOA) (Pelikan, 2005) by Pelikan and Sastry (2004). Similar to earlier fitness inheritance study (Smith, Dike, & Stegmann, 1995), all the individuals in the initial population were evaluated using the expensive fitness function. Thereafter, an offspring was evaluated either using a surrogate with a user-specified inheritance probability p_i, or using the expensive fitness function with a probability $1 - p_i$. However, the proposed method used the probabilistic models of eCGA to determine the structural form of the surrogate. That is, the MPM model used in eCGA, which partitions the variables of the underlying search problem into linkage groups, were used to determine the variable interactions used in the surrogate. Therefore, the process of learning a surrogate model was sub-divided into estimating the fitness contributions of all possible subsolutions in every partition according to the linkage map that is automatically and adaptively identified by the probabilistic model of eCGA. The authors used all evaluated parents and offspring in estimating the partial contributions of the subsolutions (or schemata) to the overall fitness of a candidate solution (Sastry, Pelikan, & Goldberg, 2004).

Specifically, they used schema theory basis for determining the relative and partial contribution of a schema to the overall fitness. That is, they defined fitness of a schema h as the difference between the average fitness of individuals that contain the schema and the average fitness of the population (Sastry, Pelikan, & Goldberg, 2004):

$$\hat{f}_s(h) = \frac{1}{n_h} \sum_{\{i|x^{(i)} \supset h\}} f(x^{(i)}) - \frac{1}{M} \sum_{i=1}^{M} f(x^{(i)}), \tag{6}$$

where n_h is the total number of individuals that contain the schema h, $x^{(i)}$ is the i^{th} evaluated individual and $f(x^{(i)})$ its fitness, and M is the total number of individuals that were evaluated. If a particular schema in not present in the evaluated population, its fitness is arbitrarily set to zero.

While the automatic and adaptive incorporation of problem knowledge in terms of the probabilistic models built by EDAs and other linkage learners is very powerful, the estimation of the partial fitness contributions of solutions is somewhat ad hoc. Moreover the schema-theory basis for estimating these partial fitness contributions of schemata works well only on certain class of search problems. For example, the fitness inheritance method of fails to provide significant speedup on noisy problems (Sastry, Pelikan, & Goldberg, 2004), and the method of fails on hierarchically-decomposable problems Pelikan and Sastry (2004).

4 Fitness Inheritance in EDAs using Least Squares Fitting

To address the issues presented in the previous section, Sastry, Lima, and Goldberg (2006) proposed a new surrogate fitness based on substructural information—as shown before—and linear estimation. We review in this section the main elements of such an approach since it is the basis of the fitness inheritance scheme used by χeCCS. Similar to Sastry, Pelikan, and Goldberg (2004), individuals with exact fitness are used to estimate the sub-structural fitnesses of the remaining individuals. These sub-structures that are defined by the probabilistic model can be viewed and directly mapped into schemata. The fitness associated with the different schemas that match an individual is then combined to estimate his fitness. In this study, schema or building-block fitness is defined as the relative (to the average fitness of the population) fitness contribution to the overall fitness of an individual.

After the model is built the linkage groups are treated as building-blocks partitions, thus all possible schemata under this structure are considered. Considering a MPM example for a 4-bit problem, whose model is `[1,3][2][4]`, the schemata for which the fitness is predicted are {0*0*, 0*1*, 1*0*, 1*1*, *0**, *1**, ***0, ***1}. The total number of schemas is given by

$$N = \sum_{i=1}^{m} 2^{k_i}, \tag{7}$$

where m is the number of BBs and k_i is the size of the i^{th} BB (number of variables belonging to the BB).

The fitness values of the schemata are estimated as follows. Each individual used for learning is mapped into a binary vector of size N, where each variable of the vector uniquely identifies a given schema. That is, the vector is instantiated by the following delta function

$$\delta(x, h_j) = \begin{cases} 1, & \text{if } x \supset h_j \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

where x is the individual to be converted and h_j is the j^{th} schema. Basically, the vector will have value "1" for the schemas that contain individual x and "0" otherwise. After mapping M evaluated individuals using the above function, the following matrix with dimension $(M \times N)$ is obtained:

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M,1} & a_{M,2} & \cdots & a_{M,N} \end{pmatrix}, \tag{9}$$

where $a_{i,j} = \delta(x^{(i)}, h_j)$. We note that $x^{(i)}$ denotes the i^{th} individual used for learning the surrogate fitness model. We note that the rank of matrix $\mathbf{A}$ is $N - m + 1$.

Also, the relative (to the average) fitness of each evaluated individual is kept in a vector with dimension $(M \times 1)$ as

$$\mathbf{f} = \begin{pmatrix} f(x^{(1)}) - \bar{f} \\ f(x^{(2)}) - \bar{f} \\ \vdots \\ f(x^{(M)}) - \bar{f} \end{pmatrix}, \tag{10}$$

where $f(x^{(i)})$ is the evaluated fitness of the i^{th} individual used for learning and $\bar{f}$ is the average fitness of all M evaluated individuals (both from parent and offspring population). The average fitness it then given by

$$\bar{f} = \frac{1}{M} \sum_{i=1}^{M} f(x^{(i)}). \tag{11}$$

Given that there are N different schema fitnesses to estimate, the fitness coefficients associated with the N binary variables can be displayed as vector of dimension $(N \times 1)$

$$\hat{\mathbf{f}}_{\mathbf{s}} = \begin{pmatrix} \hat{f}_s(h_1) \\ \hat{f}_s(h_2) \\ \vdots \\ \hat{f}_s(h_N) \end{pmatrix}, \tag{12}$$

where $\hat{f}_s(h_j)$ is the fitness of schema h_j.

The task of estimating the relative fitness of each schema can be stated as finding a vector $\hat{\mathbf{f}}_{\mathbf{s}}$ that satisfies the equality:

$$\mathbf{A}\hat{\mathbf{f}}_{\mathbf{s}} = \mathbf{f}. \tag{13}$$

In practice, this equality might not be entirely satisfied and one must instead seek for minimizing the difference between left and right terms of Equation 13. For that, it is used a multi-dimensional least squares fitting approach. Thus, under the least squares fitting principle the problem of estimating the fitness of schemata can now be reformulated as finding the appropriate values for vector $\hat{\mathbf{f}}_{\mathbf{s}}$ such that the following squared error function χ^2 is minimized:

$$\chi^2 = \left(\mathbf{A}\hat{\mathbf{f}}_{\mathbf{s}} - \mathbf{f}\right)^T \left(\mathbf{A}\hat{\mathbf{f}}_{\mathbf{s}} - \mathbf{f}\right). \tag{14}$$

The solution to the above problem is a well-known result from literature, therefore details on the resolution are not provided and the interested reader should refer elsewhere (Björk, 1996; Draper & Smith, 1966; Haykin, 1996; Kailath, Sayed, & Hassibi, 2000; Rao & Toutenburg, 1999). The method used in this paper to perform multi-dimensional least squares fitting was provided by the R project for statistical computing[1].

After obtaining the estimates for schema fitnesses, the estimation of an individual's fitness is a straightforward process that consists in summing the average fitness of the population to the fitness of each schema that contains the individual being considered. The estimated fitness of an individual x is then given by

$$f_{inh}(x) = \bar{f} + \sum_{j=1}^{N} \delta(x, h_j)\hat{f}_s(h_j), \tag{15}$$

where $\hat{f}_s(h_j)$ is given by the j^{th} element of vector $\hat{\mathbf{f}}_{\mathbf{s}}$.

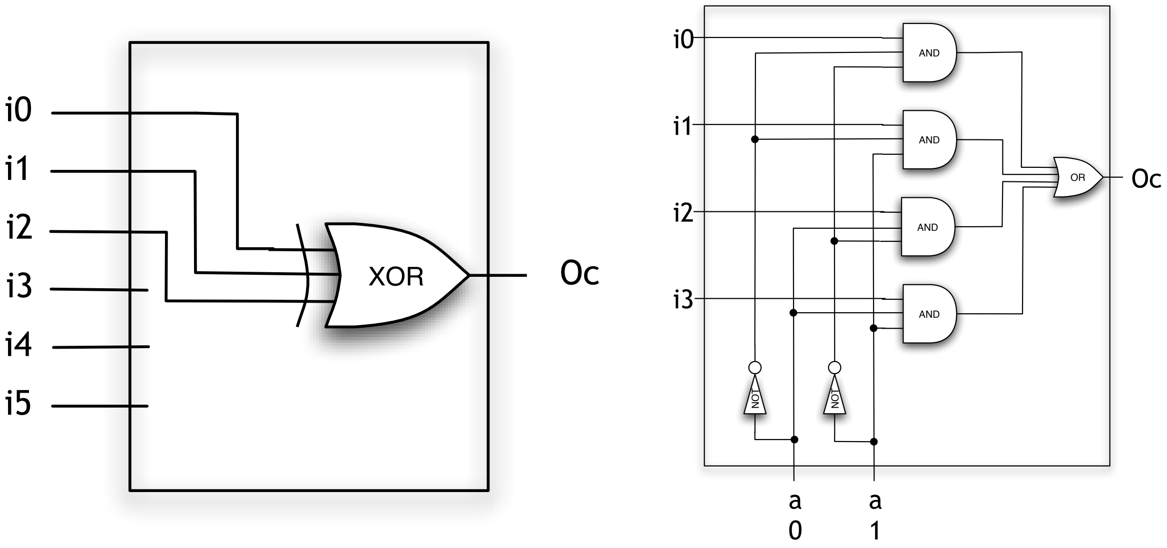
It can be easily seen that the surrogate obtained by using a structure inferred from a perfect model and the coefficients via least squares yields that is identical to Walsh transform (Goldberg, 1989) of the accurate fitness function. This clearly suggests that given an accurate probabilistic model, we can obtain a surrogate that accurately estimates the fitness of untested solutions.

5 Fitness Inheritance in χeCCS: Functional Misalignment

The surrogate fitness model proposed in the previous section has been successfully used in fitness inheritance schemes in the optimization domain (Sastry, Lima, & Goldberg, 2006). To successfully

---

[1] http://www.r-project.org/



(a) Hidden `XOR` (b) 6-input multiplexer

Figure 1: The two problems presented in this figure present different properties. `XOR` is a problem with no overlapping in the variable space. The 6-input multiplexer, on the other hand requires exhaustive reuse of varible to provide the proper output.

apply such fitness surrogate in the learning domain of the χeCCS (Llorà, Sastry, Goldberg, & de la Ossa, 2006) and other genetics-based machine learning, we conducted preliminary analysis. We used two well-know problems to the genetics-based machine learning community: (1) the hidden `XOR` problem, and (2) the 6-input multiplexer. We summarize in this section the insights obtain from this simple experimentation—see Figure 1.

The hidden `XOR` problem requires discovering a `XOR` function embedded in a set of inputs—most of them irrelevant to the `XOR` classification task. Our initial test used a simple 3-bit `XOR` embedded in a 6 binary input space. Hence, we have a binary input space and a binary classification task. χeCCS easily solves the problem providing the following output rules—with their associate fitness—and the final model based on MPMs:

Rules:	`111###`	→ 1	$f(r) = 0.625$
	`001###`	→ 1	$f(r) = 0.625$
	`010###`	→ 1	$f(r) = 0.625$
	`100###`	→ 1	$f(r) = 0.625$
	default	→ 0	
Model:	`[i0 i1 i2] [i3][i4][i5]`		

The rules provides by χeCCS correctly describes the hidden `XOR` problem. Moreover, the model evolved by χeCCS clearly groups the two variable `[i0 i1 i2]` involved in the hidden `XOR`. With this information we should be able to create the surrogate fitness following the steps described in the previous section. Once we obtain the surrogate, we generated all the possible rules (3^6) and computed the average error of the surrogate fitness as

$$\epsilon\left(f_{inh}(x)\right) = \sqrt{\frac{\left(f(x) - f_{inh}(x)\right)^2}{pop\_size}} \tag{16}$$

For illustrative purposes we present the average error of the surrogate for the evolved χeCCS model and two other suboptimal models.

Model	$\epsilon\left(f_{inh}(x)\right)$
`[i0][i1][i2][i3][i4][i5]`	14.976%
`[i0 i1][i2][i3][i4][i5]`	14.952%
`[i0 i1 i2][i3][i4][i5]`	**0.890%**

The results presented above show how when the correct model is provided, the surrogate model performs remarkably well, being smaller than 0.9%. Encouraged by these results, we repeated the experiments for the 6-input multiplexer problem. The following tables presents the error of the fitness surrogate for a totally independent model and by the model provided by χeCCS.

Model	$\epsilon\left(f_{inh}(x)\right)$
`[i0][i1][i2][i3][i4][i5]`	18.703%
`[i0 i2][i1 i5][i3 i4]`	**18.665%**

However, the results were not as compelling as the ones obtained on the hidden `XOR` problem. As stated by Sastry, Lima, and Goldberg (2006) the proposed surrogate fitness based on substructural information and linear estimation should hold for any given accurate model. The reason for the poor performance of the surrogate fitness was not the method, but the model used to create such a surrogate. The model evolved by χeCCS—the same as eCGA—is based on MPMs and, hence, by definition non-overlapping. This means that χeCCS is able to solve the multiplexer problem using a non-overlapping and approximate model. This model does not prevent the χeCCS to solve the multiplexer problem quickly, reliably, and accurately (Llorà, Sastry, Goldberg, & de la Ossa, 2006) pushing the boundaries to embrace the largest multiplexer problems solve till them.

However, this approximated non-overlapping model do not produce and accurate surrogate fitness function. This is the result of functional misalignment between the functional form required for the surrogate and the approximated non-overlapping model. In another words, to successfully use such surrogate for the multiplexer problem, we need to induce overlapping models. A validation of this intuition is presented below.

Model	$\epsilon\left(f_{inh}(x)\right)$
`[i0][i1][i2][i3][i4][i5]`	18.703%
`[i0 i2][i1 i5][i3 i4]`	18.665%
`[i0 i1 i2][i0 i1 i3][i0 i1 i4][i0 i1 i5]`	**0.733%**

The last model presented in the previous table shows how an overlapping model leads to a surrogate fitness for the multiplexer problem with the same degree of accuracy than the one achieve in the hidden `XOR` problem one. This model easily follows the intuitive model for the multiplexer underlying functionality—as presented in Figure 1(b).Thus, if we want to use fitness inheritance we need to use a model builder that is able to produce such overlapping models. The following section introduces a competent GA that is able to evolve such models as the one required to solve the multiplexer problem.

6 Identifying Overlapping Building Blocks using DSMGA

This section gives a brief introduction to the model-building process used in the dependency structure matrix genetic algorithm (DSMGA), which is later used in this paper as a basis of the proposed method. A detailed descriptions of DSMGA is beyond the scope of this paper and can be found elsewhere (Yu, Goldberg, Yassine, & Chen, 2003). DSMGA utilizes the dependency structure matrix (DSM) clustering techniques to extract the information of building blocks and uses the information to accomplish BB-wise crossover. In this section we introduce the concept of DSM and the DSM clustering problem. Then, we describe the metric to cluster DSMs and the algorithm used.

A dependency structure matrix is essentially an adjacency matrix representation of a graph where each entry d_{ij} represents the dependency between node i and node j (Steward, 1981; Yassine, Falkenburg, & Chelst, 1999). Entries d_{ij} can be real numbers or integers. The larger the d_{ij} is, the higher the interaction is between node i and node j. If we focus on the $[0,1]$ domain, then $d_{ij} = 0$ means that node i and node j do not interact, and $d_{ij} = 1$ means that node i and node j interact with each other. The diagonal entries (d_{ii}) have no significance and are usually set to zero or blacked-out. For elaborate exposition of DSM, please see MIT DSM web site: `http://www.dsmweb.org/`.

The goal of DSM clustering is to find subsets of DSM elements (*i.e.*, clusters) so that nodes within a cluster are maximally interacting, and clusters are minimally interacting. In a typical DSM clustering problem, overlapping clusters (clusters that share same nodes) are permissible. The DSM model of linkage allows overlapping of variables, as opposed as the non-overlapping MPM proposed by eCGA (Harik, Lobo, & Sastry, 2006). However, rearranging a DSM to obtain the proper clusters requires a metric to compute the usefulness of the proposed clustering rearrangement (Fernandez, 1998; Sharman, Yassine, & Carlile, 2002; Yu, Goldberg, Yassine, & Chen, 2003).

DSMGA relies on a DSM clustering metric based on the minimal description length principle (MDL) (Rissinen, 1978). Suppose that we have a model which describes a given data set, $DSM = [d_{ij}]$. Here, the model means a description that specifies which node belongs to which cluster. Usually, the model does not completely describe the given data; otherwise, the model would be too complex to use. Therefore, the description length that the model needs to describe the given data consists of two parts: the model description and the mismatched data description.

The minimum description length principle (MDL) (Rissinen, 1978) satisfies the needs for dealing with the above trade-off. The MDL can be interpreted as follows: among all possible models, choose the model that uses the minimal length for describing a given data set (that is, model description length plus mismatched data description length). There are two key points that should be noted when MDL is used: (1) the encoding should be uniquely decodable, and (2) the length of encoding should reflect the complexity.

Model Encoding. The description of each cluster starts with a number which is sequentially assigned to each cluster, and then this is followed by a sequence of nodes in the cluster. It is easily seen that the length of this model description is as follows:

$$\Sigma_{i=1}^{n_c} \left(\log_2 n_n + cl_i \cdot \log_2 n_n \right), \tag{17}$$

where n_c is the number of clusters in the model, n_n is the number of nodes, cl_i is the number of nodes in the i-th cluster. If n_n and n_c are known, the above model description is uniquely decodable. When n_n is given, and by assuming $n_c \leq n_n$, then $\log n_n$ bits are needed to describe n_c. It is a constant for all models, and therefore they are omitted without loss of accuracy.

Mismatched Data Description. Based on the model, another DSM ($DSM' = [d'_{ij}]$) is constructed, where each entry d'_{ij} is 1 if and only if some cluster contains both node i and node j simultaneously. Then, DSM' is compared to the given DSM. For every mismatched entry, where $d'_{ij} \neq d_{ij}$, a description to indicate where the mismatch occurred (i and j) is needed and one additional bit to indicate whether the mismatch is zero-to-one or one-to-zero. Define a mismatch set $S = \{(i,j) | d'_{ij} \neq d_{ij}\}$. The mismatched data description length is given by:

$$\Sigma_{(i,j)\in S} \left(\log n_n + \log n_n + 1 \right). \tag{18}$$

The first $\log n_n$ in the bracket indicates i, the second one indicates j, and the additional one bit indicates the type of mismatch.

The MDL clustering metric is given by the summation of the model description length and the mismatched data description. With some arithmetic manipulations, the metric can be expressed as follows:

$$f_{DSM}(M) = \log n_n \Sigma_{i=1}^{n_c} (cl_i + 1) + |S|(2 \log n_n + 1), \tag{19}$$

where n_c is the number of clusters, n_n is the number of nodes in the DSM, cl_i is the size of the i-th cluster, and S is a mismatch set.

With the above metric, the DSM clustering problem is converted into an optimization problem: Given a DSM, the objective is to find a DSM clustering arrangement (model, M) to minimize the above metric (f_{DSM}). A steepest descent algorithm was adopted by Yu, Goldberg, Yassine, and Chen (2003) to optimize the DSM clustering problem. Based on the MDL metric, it add/remove one node to/from one cluster at each iteration. The steepest descent algorithm stops when no further improvement is possible. Further details can be found elsewhere (Yu, Goldberg, Yassine, & Chen, 2003).

7 Fitness Inheritance Functional Alignment via Overlapping BBs

As described in the previous section, DSM clustering is able to identify overlapping building blocks, a crucial element to create an accurate surrogate for the multiplexer problem. Using DSM clustering technique requires being able to express the interaction information among variables as a DSM. To achieve this goal we cannot directly used the transformation method proposed by DSMGA (Yu & Goldberg, 2006), since it is engineered to deal with binary populations—not the case of χeCCS. However, if we can define a transformation from a rule set to a DSM that retains the interaction properties of the variables, then DSM cluster will provide us with the proper BB identification mechanism.

We can regard a rule r as a set of interactions among specific values. For instance, given the 6-input multiplexer rule `001###` → `1`, the first three positions contain specific values that need to interact with each other to properly assemble the rule. This given a rule r which condition is defined among the set of possible variables X, we can define the interaction δ_s between the *ith* position and the *jth* position of the rule as:

$$\delta_s(r_i, r_j) = \begin{cases} 1, & \text{if } r_i \neq \# \wedge r_j \neq \# \wedge i \neq j \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

That is, an interaction between position i and j exists if and only if both positions in the rule contain specific values. Thus, we should be able to define a matrix C where c_{ij} is the count of interactions among variables i and j in X. Such a matrix can be defined as follows:

$$c_{ij} = \sum_{r \in R} \sum_{i \in X} \sum_{j \in X} \delta_s(r_i, r_j) \tag{21}$$

The count matrix C is the base of the DSM, where we define each of each entries d_{ij} as the normalized C where interactions belong to the [0,1] domain. This can be simply achieve by defining d_{ij} as

$$d_{ij} = \frac{c_{ij}}{max(C)} \tag{22}$$

We now have to decide which rules should be used as a rule set R when defining the DSM. The answer is easy, only the distinct accurate ones—as was already suggested elsewhere (Butz, Pelikan, Llorà, & Goldberg, 2005). This can be easily achieved by filtering the rules of a population based on their computed error $\varepsilon(r)$. Only rules with no error should be used to build the DSM to cluster.

Another important consideration is if rules belonging to different classes should be mixed together in R. The answer for χeCCS is simple, due to the use of a default rule—close world assumption—while solving binary classification problems, only rules belonging to one class are evolved. For non-binary classification problems we should construct one DSM per rule class. Thus, we would have a better flexibility to express complex interaction without having to deal with the introduction of spurious interclass interactions.

Given the evolved rules for the hidden `XOR` presented in section 5, for the 6-input multiplexer presented below,

Rules:	`001###`	$\rightarrow$ `1`	$f(r) = 0.625$
	`01#1##`	$\rightarrow$ `1`	$f(r) = 0.625$
	`10##1#`	$\rightarrow$ `1`	$f(r) = 0.625$
	`11###1`	$\rightarrow$ `1`	$f(r) = 0.625$
	`0#11##`	$\rightarrow$ `1`	$f(r) = 0.625$
	`#01#1#`	$\rightarrow$ `1`	$f(r) = 0.625$
	`1###11`	$\rightarrow$ `1`	$f(r) = 0.625$
	`#1#1#1`	$\rightarrow$ `1`	$f(r) = 0.625$
	default	$\rightarrow$ `0`	

the DSM matrix for both problem are

$$\mathbf{DSM}_{\mathbf{XOR}} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{23}$$

$$\mathbf{DSM}_{\mathbf{MUX}} = \begin{pmatrix} 0 & 1 & 0.5 & 0.5 & 0.5 & 0.5 \\ 1 & 0 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0 & 0.25 & 0.25 & 0 \\ 0.5 & 0.5 & 0.25 & 0 & 0 & 0.25 \\ 0.5 & 0.5 & 0.25 & 0 & 0 & 0.25 \\ 0.5 & 0.5 & 0 & 0.25 & 0.25 & 0 \end{pmatrix} \tag{24}$$

Using this DSMs, DSM clustering process returns `[i0 i1 i2] [i3][i4][i5]` as model for the hidden `XOR` problem. As we presented it in section 5, this is the right model the leads to the creation of an accurate fitness surrogate. On the other hand, the DSM clustering produces the model `[i0 i1] <i2 i3 i4 i5>` indicating that there is a building block `[i0 i1]` that interacts with a bus of variables `<i2 i3 i4 i5>`. Hence, the bus can be expanded as `[i0 i1 i2][i0 i1 i3][i0 i1 i4][i0 i1 i5]` giving the right overlapping model—see section 5—to build an accurate surrogate for the multiplexer problem allowing a first fitness inheritance scheme for genetic-based machine learning approaches.

8 Conclusions and Further Work

We have shown how fitness inheritance for genetics-based machine learning techniques is possible. A surrogate fitness based on substructural information and least square fitting is able to accurately predict the fitness of the rules evolve by χeCCS. Such a surrogate can replace the cost of computing the fitness of a rule against large data sets. However, to make such a usage possible, we have shown that the model building process used and the surrogate fitness need to be functionally aligned. We have seen that χeCCS is able to solve hidden `XOR` and multiplexer problems quickly, reliably, and accurately by using approximating probabilistic models of the population of rules based on non-overlapping BBs. However, such rough approximation models are not enough to build a proper surrogate fitness model. An accurate surrogate fitness function requires an accurate probabilistic model able to express overlapping BBs—as empirically shown for the multiplexer problem. In order to obtain such models we have defined a transformation from a set of accurate rules into a DSM that when properly clustered using the DSM clustering method provides accurate overlapping BBs of the population of rules. Future research should focus on introducing probabilistic model-building GAs able to express overlapping into the χeCCS. Only then we should be able to take full advantage of the fitness inheritance schemes build around the fitness surrogate used in this paper.

9 Acknowledgments

This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant FA9550-06-1-0096, the National Science Foundation under ITR grant DMR-03-25939 at Materials Computation Center and under grant ISS-02-09199 at the National Center for Supercomputing Applications, UIUC. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, or the U.S. Government.

References

Barthelemy, J.-F. M., & Haftka, R. T. (1993). Approximation concepts for optimum structural design—a review. *Structural Optimization*, *5*, 129–144.

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

Bernadó-Mansilla, E., Llorà, X., & Traus, I. (2005). *MultiObjective Machine Learning* (Chapter MultiObjective Learning Classifier System, pp. 261–288). Springer.

Björk, Å. (1996). *Numerical method for least squares problems.* Philadelphia, PA: SIAM.

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

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

Draper, N. R., & Smith, H. (1966). *Applied regression analysis.* New York, USA: John Wiley & Sons.

Fernandez, C. (1998). Integration Analysis of Product Architecture to Support Effective Team Co-location. Master thesis, Massachusetts Institute of Technology.

Goldberg, D. E. (1989). Genetic algorithms and Walsh functions: Part I, a gentle introduction. *Complex Systems*, *3*, 129–152. (Also IllIGAL Report No. 88006).

Goldberg, D. E. (2002). *The design of innovation: Lessons from and for competent genetic algorithms.* Norwell, MA: Kluwer Academic Publishers.

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

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

Grefenstette, J. J., & Fitzpatrick, J. M. (1985). Genetic search with approximate function evaluations. *Proceedings of the International Conference on Genetic Algorithms and Their Applications*, 112–120.

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

Harik, G. R., Lobo, F. G., & Sastry, K. (2006). Linkage learning via probabilistic modeling in the ECGA. In Pelikan, M., Sastry, K., & Cantú-Paz, E. (Eds.), *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications* (Chapter 3. Berlin: Springer. (Also IlliGAL Report No. 99010).

Haykin, S. (1996). *Adaptive filter theory.* Prentice Hall.

Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing Journal*, *9*(1), 3–12.

Kailath, T., Sayed, A. H., & Hassibi, B. (2000). *Linear estimation.* Upper Saddle River, NJ: Prentice-Hall.

Llorà, X., & Sastry, K. (2006). Fast rule matching for learning classifier systems via vector instructions. In *Proceedings of the 2006 Genetic and Evolutionary Computation Conference (GECCO 2006)* pp. 15131520. (Also IlliGAL Report No. 2006001).

Llorà, X., Sastry, K., Goldberg, D. E., & de la Ossa, L. (2006). The χ-ary extended compact classifier system: Linkage learning in pittsburgh lcs. In *Proceedings of the 2006 Genetic and Evolutionary Computation Conference (GECCO 2006) Workshops: International Workshop on Learning Classifier Systems* (Also IlliGAL Report No. 2006015).

Llorà, X., Sastry, K., Goldberg, D. E., Gupta, A., & Lakshmi, L. (2005, 25-29 June). 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 pp. 1363–1370. Washington DC, USA: ACM Press.

Llorà, X., Sastry, K., Goldberg, D. E., Gupta, A., & Lakshmi, L. (2005). Combating user fatigue in iGAs: Partial ordering, support vector machines, and synthetic fitness. *Proceedings of the Genetic and Evolutionary Computation Conference*, 1363–1370. (Also IlliGAL Report No. 2005009).

Mahfoud, S. W. (1994). Population size and genetic drift in fitness sharing. *Foundations of Genetic Algorithms*, *3*, 185–224. (Also IlliGAL Report No. 94005).

Pelikan, M. (2005). *Hierarchical bayesian optimization algorithm: Toward a new generation of evolutionary algorithm.* Berlin: Springer Verlag.

Pelikan, M., & Sastry, K. (2004). Fitness inheritance in the bayesian optimization algorithm. *Proceedings of the Genetic and Evolutionary Computation Conference*, *2*, 48–59. (Also IlliGAL Report No. 2004009).

Rao, C. R., & Toutenburg, H. (1999). *Linear models: Least squares and alternatives*. Berlin: Springer.

Rissinen, J. (1978). Modeling by shortest data description. *Automatica*, *14*, 465–471.

Sastry, K. (2001). *Evaluation-relaxation schemes for genetic and evolutionary algorithms*. Master's thesis, University of Illinois at Urbana-Champaign, General Engineering Department, Urbana, IL. (Also IlliGAL Report No. 2002004).

Sastry, K., Abbass, H. A., Goldberg, D. E., & Johnson, D. D. (2005). Sub-structural niching in estimation of distribution algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference*, 671–678. (Also IlliGAL Report No. 2005003).

Sastry, K., & Goldberg, D. E. (2003). Probabilistic model building and competent genetic programming. In Riolo, R. L., & Worzel, B. (Eds.), *Genetic Programming Theory and Practise* (Chapter 13, pp. 205–220). Kluwer.

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

Sastry, K., Lima, C. F., & Goldberg, D. E. (2006). Evaluation relaxation using substructural information and linear estimation. In *Proceedings of the 2006 Genetic and Evolutionary Computation Conference (GECCO 2006)* pp. 419–426. (Also IlliGAL Report No. 2006003).

Sastry, K., Pelikan, M., & Goldberg, D. E. (2004). Efficiency enhancement of genetic algorithms via building-block-wise fitness estimation. *Proceedings of the IEEE International Conference on Evolutionary Computation*, 720–727. Also IlliGAL Report No. 2004010.

Sharman, D., Yassine, A., & Carlile, P. (2002, Sept.). Characterizing modular architectures. ASME 14th International Conference, DTM–34024.

Smith, R., Dike, B., & Stegmann, S. (1995). Fitness inheritance in genetic algorithms. In *Proceedings of the ACM Symposium on Applied Computing* pp. 345–350. New York, NY, USA: ACM.

Steward, D. V. (1981). The design structure system: A method for managing the design of complex systems. IEEE Transactions on Engineering Managment, *28*, 77–74.

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

Yassine, A., Falkenburg, D. R., & Chelst, K. (1999). Engineering design management: An informatoin structure approach. International Journal of production research, *37*(13), 2957–2975.

Yu, T.-L., & Goldberg, D. E. (2006). Conquering hierarchical difficulty by explicit chunking: Substructural chromosome compression. In *Proceedings of the 2006 Genetic and Evolutionary Computation Conference (GECCO 2006) Workshops: International Workshop on Learning Classifier Systems* pp. 1385–1392. (Also IlliGAL Report No. 2006007).

Yu, T.-L., Goldberg, D. E., Yassine, A., & Chen, Y.-P. (2003). A genetic algorithm design inspired by organizational theory: Pilot study of a dependency structure matrix driven genetic algorithm. *Artificial Neural Networks in Engineering*, 327–332. (Also IlliGAL Report No. 2003007).