Substructural Surrogates for Learning Decomposable Classification Problems

Albert Orriols-Puig[1,2], Kumara Sastry[2], David E. Goldberg[2], and Ester Bernadó-Mansilla[1]

[1]Grup de Recerca en Sistemes Intel·ligents, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Quatre Camins 2, 08022 Barcelona (Spain)
[2]Illinois Genetic Algorithms Laboratory, Department of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign
aorriols@salle.url.edu, ksastry@uiuc.edu, deg@uiuc.edu, esterb@salle.url.edu

Abstract. This paper presents a learning methodology based on a substructural classification model to solve decomposable classification problems. The proposed method consists of three important components: (1) a structural model, which represents salient interactions between attributes for a given data, (2) a surrogate model, which provides a functional approximation of the output as a function of attributes, and (3) a classification model, which predicts the class for new inputs. The structural model is used to infer the functional form of the surrogate. Its coefficients are estimated using linear regression methods. The classification model uses a maximally-accurate, least-complex surrogate to predict the output for given inputs. The structural model that yields an optimal classification model is searched using an iterative greedy search heuristic. Results show that the proposed method successfully detects the interacting variables in hierarchical problems, groups them in linkages groups, and builds maximally accurate classification models. The initial results on non-trivial hierarchical test problems indicate that the proposed method holds promise and also shed light on several improvements to enhance the capabilities of the proposed method.

1 Introduction

Nearly decomposable functions play a central role in the design, analysis, and modeling of complex engineering systems [28,6,8]. A design decomposition principle has been proposed for the successful design of scalable genetic algorithms (GAs) [8,18,20], genetic programming [25], and learning classifier systems and genetics based machine learning (GBML) [4,16]. For example, in [4], estimation of distribution algorithms (EDAs) were applied over the rule-based knowledge evolved by XCS [32,33] to discover linkages between the input variables, permitting XCS to solve hierarchical problems that were intractable with first-generation XCS.

Nonetheless, previous approaches used the probabilistic models built by EDAs—GAs that replace variation operators by building and sampling probabilistic models of promising solution—for recombination. However, the probabilistic models can also be used to induce the form of surrogates which can be used for efficiency enhancement of GAs [27,19,26] and GBML [17]. In this paper, we use the substructural surrogates for learning from decomposable problems with nominal attributes. Similar to Sastry, Lima, and Goldberg [26], we use the structural model of EDAs to induce the form of the surrogate and linear regression for estimating the coefficients of the surrogate. The surrogate is subsequently used to predict the class of unknown input instances.

In this paper, we discuss the critical components of the proposed methodology and outline several ways to implement it. We then propose a greedy search heuristic for discovering the structural model that minimizes the test error of the classification model constructed from it. We address this method as *greedy Extraction of the Structural Model for Classification* (gESMC). We artificially design a set of hierarchical problems by means of concatenating essential blocks whose output, provided by a boolean function, serves as the input of another function that determines the global output of the example. Thus, these problems may be decomposed and essential blocks should be correctly processed to predict the correct output. gESMC is able to detect the interactions between variables and build accurate classification models. Moreover, the system is compared to C4.5 and SMO. The comparison highlights that extracting the problem structure is essential to solve hierarchical problems. Finally, we review the limitations of applying a greedy search to obtain the best structural model, show in which circumstances these limitations may appear, and propose approaches to overcome them.

The paper is organized as follows. Section 2 discusses the proposed methodology followed by a description of gESMC. The test problems designed and used in this study are discussed in Sect. 4. Section 5 compares the results of gESMC with C4.5 and SMO on the hierarchical problems. Section 6 discusses some enhancements that are yet to be investigated. Section 7 provides summary and conclusions.

2 Methodology for Learning χ-Ary Input Problems

In this section, we discuss a methodology for learning the structural and the classification model from a set of labeled examples. The methodology consists of three layers: (1) the *structural model layer*, (2) the *surrogate model layer*, and (3) the *classification model layer*. The *structural model layer* extracts the dependencies between the attributes of the examples in the dataset. These dependencies can be expressed in form of linkage groups [9,10], matrices [35], or Bayesian networks [18]. However the dependencies are represented, the key idea is that the salient interactions between attributes are used as a basis for determining the output. The *surrogate model layer* uses the *structural model* to infer the functional form of the surrogate, and the coefficients of the surrogate are determined using linear regression methods. The resulting surrogate is a function that approximates the output of each input instance. Finally, the *classification model layer* uses the surrogate function to predict the class of new input instances.

In essence, we infer the structure of the surrogate from the structural models of attribute interactions and then use linear regression methods to estimate the value of the coefficients (or the partial contributions of subsolutions to the output) of the resulting surrogate function. Finally the surrogate is used to predict the class of new input instances. Details of each of the three components are discussed in the following sections.

2.1 Structural Model Layer

The *structural model layer* is responsible for identifying salient interactions between attributes, which need to be processed together to determine their contribution to the output. For example, consider a problem with two binary attributes (x_1, x_2) and whose output is determined by the *x-or* boolean function. If we considered each of the attributes independently, we cannot evolve a function that computes the output accurately for all possible inputs. However, when we consider the two attributes together, we can easily create a function that accurately predicts the output for all possible input sequences.

A number of linkage-learning methods [8] can be used to implement the structural model layer. Here, we use estimation of distribution algorithms (EDAs) [18,20], which learn the salient interactions between decision variables by building probabilistic models of promising candidate solutions. These probabilistic models can be expressed in different forms such as (i) linkage groups [9,10], i.e., groups of variables that have a salient interaction; matrices [35], which express the relationship between pairs or groups of variables, permitting to detect overlaps in the groups of interacting variables; or Bayesian networks [18], in which the nodes represent variables and the connections denote salient interactions between variables. The implementation proposed in the next section uses linkage groups to express the salient interactions, although it can be extended to other representations.

In the realm of learning classifier systems (LCSs) or genetics-based machine learning (GBML), EDAs have been successfully combined with LCSs to extract the linkages between classifiers' alleles [4,16,17]. However, unlike previous studies which used the structural model as a replacement of recombination, in this study we integrate the structural model and learning with the use of substructural surrogates.

In order to achieve this integration, the first step is to find the structural model of the given data. This can be done in several ways. As with EDAs, given a class of permissible structural models, we can search for the best structural model. Prior and domain-specific knowledge can also be used to propose the structural model, and a search mechanism could be used to refine it [1]. In this study we use a greedy search heuristic that searches for the model structure that results in the most accurate surrogate model, details of which are given in Section 3.

2.2 Surrogate Model Layer

The *surrogate model layer* preprocesses the input examples according to the structural model and builds a regression model from these preprocessed examples, as described in [26]. In this section we summarize the procedure of building such a surrogate. Consider a matrix D of dimension $n \times \ell$ that contains all the input examples (where n is the number of examples and ℓ the number of attributes). Once the structural model is built, every linkage group is treated as a *building block* [11]. Then, we consider all possible input combinations within each linkage group to process the input examples.

For example, consider the following structural model of a binary problem of 3 variables: $\{[x_1, x_3], [x_2]\}$. That is, there is salient interaction between variables x_1 and x_3, which are independent from the variable x_2. In this case, we consider the following schemata: {0*0, 0*1, 1*0, 1*1, *0*, *1*}. In general, given m linkage groups, the total number of schemata m_{sch} to be considered is given by:

$$m_{sch} = \sum_{i=1}^{m} \left[\Pi_{j=1}^{k_i} \chi_{i,j} \right], \tag{1}$$

where $\chi_{i,j}$ is the alphabet cardinality of the j^{th} variable of the i^{th} linkage group, and k_i is the size of the i^{th} linkage group.

Then, each example in D is mapped to a vector of size m_{sch}, creating the matrix A of dimensions $n \times m_{sch}$:

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m_{sch}} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m_{sch}} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m_{sch}} \end{pmatrix}, \tag{2}$$

where $a_{i,j}$ will have value '1' if the *ith* example belongs to the *jth* schemata, and '0' otherwise. Note that, given an example, only one of the schemata for each linkage group can have value '1'.

We map different labels or classes of the examples to numeric values. For example: $\{class_1, class_2, \cdots, class_k\} \longrightarrow \{\mathbb{Z}_1, \mathbb{Z}_2, \cdots, \mathbb{Z}_k\}$. The label or class c_i of each example is also kept in a matrix C of dimensions $n \times 1$:

$$\mathbf{C} = \begin{pmatrix} c_1 \; c_2 \; \cdots \; c_n \end{pmatrix}^{\tau}. \tag{3}$$

Now, the task of designing the surrogate can be formulated into a linear system of equations and our objective is to compute the coefficients of the matrix $\mathbf{x}$ of dimensions $m_{sch} \times 1$ that satisfy the following equality:

$$\mathbf{Ax} = \mathbf{C}. \tag{4}$$

In practice, we may not find an $\mathbf{x}$ that satisfies this expression. For this reason, we use a multi-dimensional least squares fitting approach. That is, the problem is reformulated by estimating the vector of coefficients $\mathbf{x}$ that minimize the square error function χ:

$$\chi^2 = (Ax - C)^T \cdot (Ax - C). \tag{5}$$

The problem of least-squares fitting is well-known, and so we do not provide insight in the resolution methodology herein. The interested reader is referred to [5,23]. Here, we used the multi-dimensional least squares fitting routine available with Matlab [24].

2.3 Classification Model Layer

Once we obtain the matrix x with the regression coefficients, the output for a new example is computed as follows. The example is mapped to a vector $\vec{e}$ of size m_{sch}. The mapping procedure used is identical to that used to create matrix $\mathbf{A}$ and as outlined in the previous section, the elements of $\vec{e}$ will have a value '1' if the example belongs to the corresponding schemata and '0' otherwise. Then, the predicted output is given by:

$$output = \vec{e} \cdot x. \tag{6}$$

Note that the *output* is a continuous value, and has to be transformed to one of the possible class labels. Therefore, we convert the continuous output to the closer integer $\mathbb{Z}_i$ in $\{\mathbb{Z}_1, \mathbb{Z}_2, \cdots, \mathbb{Z}_k\}$, and then, return the class label that corresponds to $\mathbb{Z}_i$.

In essence, the proposed method relies on the structural and the surrogate models extracted from the data to build the classification model. Therefore, we note that if this structural model does not reflect the variable interactions accurately, the accuracy of the classification model will be limited. Thus, a critical task for the success of the proposed methodology is our ability to find reasonably accurate structural models. In the next section we propose an implementation of the methodology that searches iteratively for the best structural model, and uses the classification model to evaluate its quality. We call this implementation *greedy extraction of the structural model for classification* (gESMC).

3 Implementing the Methodology: gESMC

The pseudocode of the implementation of our proposed method is shown in Algorithm 1. In the initialization stage, the algorithm divides the data into training and test sets. We start with a structural model where all variables are treated as independent and build a surrogate function via regression over the training set as explained in the previous section (see Section 2.2). The quality of the classification model is evaluated with the test set and stored in the variable *mdl*.

Similar to the extended compact genetic algorithm (eCGA) [10], in gESMC we use a greedy search heuristic to partition the set of attributes into non-overlapping clusters such that the classification error is (locally) minimized. That is, starting from a model where variables are treated as independent, we continue to merge substructures till either (1) the *mdl* measure becomes less than a user set threshold θ, or (2) the search produces no improvement. In every iteration of the inner loop (lines 10 to 13), we merge two linkage groups from the current best model, create the surrogate and the classification model, and evaluate it. That is, $\binom{m}{2}$ new structural models are formed (where m is the number of substructures in the current best model), and their surrogate functions created and evaluated. Among the evaluated $\binom{m}{2}$ models, the one with the lowest classification error is chosen as the current best model for the next iteration if it *significantly improves*

Algorithm 1. Building of structural and classification model via a greedy search.

```
Data: dataset is the set of labeled examples.
Result: function is the classification function and bestModel the structural
        model.
1  begin
2      i ← 0
3      count ← 0
4      [train, test] ← divideData ( data )
5      bestModel ← [1], [2], ..., [n]
6      function ⟵ createSurrogateFunction ( bestModel, train )  ▷ See Sect. 2.2
7      mdl ← evaluateModel ( bestModel, test )
8      isImproving ← true
9      while mdl > $\theta$ and isImproving do
10         for $i \in \{1, ..., length(bestModel) - 1\}$ do
11             for $j \in \{i+1, ..., length(bestModel) - 1\}$ do
12                 newModel[count] ← joinLinkages( bestModel, i, j)
13                 newFunction[count] ← createSurrogateFunction (
                   newModel[count], train )
14                 newMdl[count] ← evaluateModel ( newModel[count], test )
15                 count ← count + 1
16             end
17         end
18         best ← position min. mdl( newMdl )          ▷ Selects the best model
19         if newMdl[best] significantly improves mdl then
20             bestModel = newModels[best]
21             mdl = newMdl[best]
22         else
23             isImproving=false
24         end
25     end
26 end
```

the current best model; otherwise, we terminate the search, and the current best surrogate and classification models are returned as the (locally) best models.

Three elements of the implementation need further explanation: (1) procedure to divide the data into training and test sets (line 2), (2) evaluation of the model (lines 5 and 12), and (3) procedure for comparing two models and choosing the best one (line 17). Each of the three elements are discussed in the following paragraphs.

Partition of the data. The procedure used to partition the data into training and test sets affects the estimation of classification error. A number of approaches such as holdout validation, k-fold cross validation, and leave-one-out cross-validation methods can be used. Here, we use a k-fold cross validation [29] with $k = 10$.

Evaluation of the model. The quality of a structural model depends on (1) the complexity of this model, and (2) the test error of the classification model created from it. Again a number of measures such as minimum description length metrics and multiobjective approaches could be used to measure the relative quality of a given surrogate and classification model. We use the k-fold cross validation which provides a measure of both the test error and the model complexity in terms of overfitting the training data. That is, if the structural model is more complex than necessary, the surrogate function will tend to overfit the training instances, and the test error will increase. Nonetheless, we acknowledge that direct measures for model complexity could be included in the evaluation.

Comparison of models. Given a current-best model, in gESMC we consider all pairwise merges of the substructures of the current-best model. We need to choose the best model among all the models created via the pairwise merges and compare it to the current-best model. Again, this could be done in a number of ways. For example, we could accept the new model if its classification error is lower than that of the current-best model. However, this might lead to spurious linkages and more complex models might be accepted, especially if the data set is noisy. To avoid getting unnecessarily complicated structural models, we can say that a model m_1 is significantly better than a model m_2 if:

$$error_{m_1} < error_{m_2} - \delta, \tag{7}$$

where δ is a user-set threshold. Alternatively, we can use different statistical tests as well. In our implementation, we use a paired t-test to determine if a new, more complex, structural model is better than the current best model [29]. That is, we feed the t-test with the errors corresponding to the ten different folds obtained with the current best model and the new model. We use a significance level of $\alpha = 0.01$.

Before proceeding with a description of the test functions, we note two important properties of gESMC. First, in the current implementation gESMC, the structural model is a partition of the variables into non-overlapping groups. However, this limitation can easily be relaxed by using other structural models [35,18]. Second, because of the greedy procedure, we need some guidance from lower-order, sub-optimal structural models toward an optimal structural model. This limitation can be alleviated by replacing the greedy search heuristic with another optimization method such as genetic algorithms [11,7]

4 Test Problems

In this section, we present a general set of decomposable problems to investigate the capabilities of gESMC in correctly identifying salient substructures and building an accurate classification model. Following the idea of designing hierarchical artificial problems proposed in [4,3], we design a class of two-level

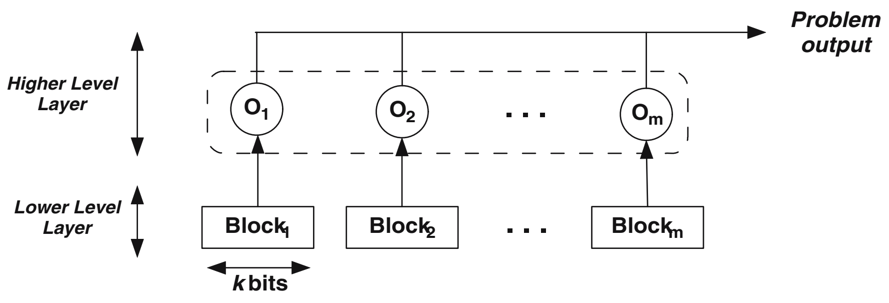

Fig. 1. Example of the design of a two-level hierarchical problem. In the low level layer, m blocks of k bits are concatenated. Each block is evaluated resulting in the correspondent output. All the outputs are groups in an input string that is used to determine the global output.

hierarchical problems where the lower level consists of functions that operate on a set of binary-input blocks and the upper level consists of functions that operate on the lower-level function values to produce the output (see a schematic illustration of these types of problems in Fig. 1). The lower- and upper-level function used in the study are explained in Section 4.1 and 4.2, respectively.

4.1 Lower Level of the Hierarchy

At the lower level of the hierarchy, we considered the following two binary functions which operate independently on m blocks with k variables in each block. Moreover the variables within a block interact with each other and determine the output of the function.

The position problem. The *position* problem [2] is defined as follows. Given a binary input of length ℓ, the output is the position of the left-most one-valued bit. For example, f($\underline{1}$00)=3, f(0$\underline{1}$0)=2, f(00$\underline{1}$)=1, and f(000)=0. Note that every variable is linked to all the variables on its left.

The parity problem. The *parity* problem [15] is a two-class binary problem defined as follows. Given a binary input of length ℓ, the output is the number of one-valued bits modulo two. For example, f(110)=0, f(100)=1, and f(111)=1. To predict the output accurately for the parity problem, all the variables have to be jointly processed. Additionally, for a k-bit parity problem, the structural model that represents that all the variables are independent yields a classification model of the same accuracy as the one that contains substructures of size $k-1$ or less. That is, till we get a structural model that groups all k variables together, the accuracy of the classification model does not increase.

4.2 Higher Level of the Hierarchy

At the higher level of the hierarchy, we use the following problems where each variable contributes independently to the output. That is, the structural information is contained in the lower-level of the hierarchy and the upper level function affects the salience of the substructures. Notice that χ-ary strings are permitted in the higher level. That is, both problems defined as follows for the higher level can deal with χ-ary strings.

The decoder problem. The decoder problem [2] is a binary-input multi-class problem defined as follows. Given an input of length ℓ, the output is determined by the decimal value of the input. For example, $f(111) = 7$, $f(101) = 5$, and $f(000) = 0$. Note that each variable independently contributes to the output. That is, starting with class equal to zero, a '1' in i^{th} position adds 2^i to the output, irrespective of other variable values.

The count-ones problems. The *count-ones* is defined as follows. Given a binary input of size ℓ, the output is the number of one-valued bits. Again, the output of the count-ones problems can be predicted by treating the input variables independently.

As mentioned earlier, we concatenated m blocks of k bits of the two lower level problems with the two higher level problems to create four different hierarchical test problems. Specifically, we used the *position* at the lower level with the *decoder* (HPosDec) and the *count-ones* (HPosCount) in the higher level. Similarly, low order *parity* blocks were combined again with the *decoder* (HParDec) and the *count-ones* (HParCount). Additionally, we added some irrelevant bits, which do not contribute to the output, to see if our method was capable of ignoring them. Therefore, in our case, $\ell \geq m \cdot k$, where ℓ is the length of the input string.

With the above description of the test problems, the following section presents the results of gESMC and compares them with those of C4.5 and SMO.

5 Results

This section analyzes the behavior of gESMC for learning hierarchical problems, and compares the results to those obtained with two highly competitive learners in terms of performance and interpretability.

5.1 Experimental Methodology

We use the four hierarchical problems designed in the previous section to analyze the performance of gESMC. We start with concatenations of three minimum-order blocks in the lower level hierarchy (that is, k=2) and add 9 irrelevant bits to the input. Our aim is to analyze the capabilities of gESMC in (i) identifying salient substructures of interacting variables, and (ii) ignoring irrelevant variables. Next, we increase the order of the lower-order blocks with a two-fold objective. For the problem with position blocks, we analyze if the system is able to identify and efficiently handle larger groups of linked variables. For the problems with parity blocks, we want to investigate the behavior of gESMC when there is a lack of guidance toward an accurate substructural model.

Table 1. Test error and standard deviation obtained with gESMC, SMO and C4.5 on the problems HPosDec, HPosCount, HParDec, HParCount with ℓ=15, $m = 3$, and $k = 2$. Results are averaged over ten runs with different holdouts and random seeds.

	gESMC	**C4.5**	**SMO**
HPosDec	0.00% ± 0.00%	0.00% ± 0.00%	0.00% ± 0.00%
HPosCount	0.00% ± 0.00%	0.00% ± 0.00%	21.89% ± 0.13%
HParDec	0.00% ± 0.00%	3.32% ± 2.90%	89.11% ± 0.94%
HParCount	0.00% ± 0.00%	5.15% ± 4.43%	62.72% ± 0.27%

To illustrate the need for detecting linkage groups in classification tasks, we compare the results obtained with gESMC to those of two widely used learners: C4.5 [22], and SMO [21]. C4.5 is a decision tree, derived from ID3, which has been widely used because of its ability to tackle a wider range of problems and because of the interpretability of the extracted knowledge. SMO is a *support vector machine* [30] that implements the *Sequential Minimal Optimization* algorithm. Although the interpretability is more difficult since it represents the knowledge as function weights, its competence has been demonstrated in different kinds of problems. Both methods were run using WEKA [34]. Unless otherwise noted, for C4.5 we used the default configuration, and for SMO, we used a polynomial kernel of order 1.

The three methods are compared in terms of performance (that is, test accuracy) and comprehensibility of the knowledge generated by the learner. As the datasets had a large number of instances, we used the *holdout* methodology[1] to estimate the test accuracy; that is, 70% of the instances were randomly selected and placed in the training set, and the rest formed the test set. We repeated the experiments with ten different holdouts. To compare the performance of each pair of learners on a given problem, we applied a paired Student t-test [29] on the results. We fed the results for each different seed to the t-test. To study the interpretability of each method, we qualitatively compared the structural and the classification models evolved by gESMC to the decision trees generated by C4.5, and the weights extracted by SMO.

5.2 Results with 2-Bit Low Order Blocks

We first show performances for gESMC, C4.5, and SMO on the problems HPosDec, HPosCount, HParDec, and HParCount with $\ell = 15$, $m = 3$, and $k = 2$. Therefore, the problems were formed by three lower level blocks of two bits and 9 irrelevant bits at the end of the binary input. Next, we compare the results in terms of performance and interpretability.

Comparison of the Performance. Table 1 summarizes the test errors resulting of applying gESMC, C4.5, and SMO on the four hierarchical problems. All

[1] A holdout is the simplest cross-validation approach where the data is divided in two sets, the train and the test set.

Table 2. Structural models and surrogate functions build by gESMC for the problems HPosDec, HPosCount, HParDec, HParCount with ℓ=15, $m = 3$, and $k = 2$

HPosDec	*link. groups*	$[x_0x_1][x_2x_3][x_4][x_5][x_6][x_7][x_8][x_9][x_{10}][x_{11}][x_{12}][x_{13}][x_{14}]$
	surr. func.	$16.75 + 9(1 - \overline{x}_0x_1) - 6\overline{x}_2\overline{x}_3 - 3\overline{x}_2x_3 - 1.5x_4 + 0.5x_5$
HPosCount	*link. groups*	$[x_0x_1][x_2x_3][x_4][x_5][x_6][x_7][x_8][x_9][x_{10}][x_{11}][x_{12}][x_{13}][x_{14}]$
	surr. func.	$0.75 + \overline{x}_0x_1 + (1 - \overline{x}_2x_3) + 0.5\overline{x}_4 + 0.5x_5$
HParDec	*link. groups*	$[x_0x_1][x_2x_3][x_4x_5][x_6][x_7][x_8][x_9][x_{10}][x_{11}][x_{12}][x_{13}][x_{14}]$
	surr. func.	$2 + 4(\overline{x}_0x_1 + x_0\overline{x}_1) - 2(\overline{x}_2\overline{x}_3 + x_2x_3) - (\overline{x}_4x_5 + x_4\overline{x}_5)$
HParCount	*link. groups*	$[x_0x_1][x_2x_3][x_4x_5][x_6][x_7][x_8][x_9][x_{10}][x_{11}][x_{12}][x_{13}][x_{14}]$
	surr. function	$\overline{x}_0x_1 + x_0\overline{x}_1 + \overline{x}_2x_3 + x_2\overline{x}_3 + \overline{x}_4x_5 + x_4\overline{x}_5$

the results were averaged over ten runs, each with a different holdout partition and random seed.

The results show that gESMC obtained 0% test error for all the problems tested. This indicates that the method is able to process the variable linkages and build maximally accurate classification models. None of the other learners could achieve 0% error in all the problems. C4.5 achieved 0% test error for the problems HPosCount and HPosDec, the ones formed by position blocks. Nonetheless, on the problems that consist of parity blocks, C4.5 was significantly outperformed by gESMC according to a paired t-test on a confidence level of 0.99. Finally, SMO presents the worst behavior of the comparison. The learner could accurately generalize over the input data only on the HPosDec problem. For the problems HPosCount, HParDec, and HParCount, the results of SMO significantly degraded those obtained with gESMC and C4.5. Note the big difference in the test errors; for HParDec, SMO has 89.11% test error, C4.5 has 3.32%, and gESMC is maximally accurate. We repeated the experiments with a Gaussian kernel [13] to promote the discovery of the linkage groups, but no significant improvement was found.

These results highlight the importance of learning and incorporating the structural model into the classification model. gESMC found highly accurate classification models only after discovering the problem structure (examples of some structural and classification models are shown in the next section). However C4.5 and SMO failed since they were not able to identify this structure. Note that the problems formed by parity blocks resulted more problematic for both learners than the problems based on position blocks. This could be explained as follows. The variables linkages in the position are weaker than in the parity. That is, in the position problem every variable processed from left to right reduces the uncertainty of the output. In the parity, looking at a single variable does not reduce the uncertainty, and so, processing the linkages is crucial. We hypothesize that, for this reason, problems formed by parity are more difficult to learn for C4.5 and SMO.

Comparison of the Interpretability. We now analyze the interpretability of the models created by gESMC, and qualitatively compare them to those obtained by C4.5 and SMO. Table 2 shows the structural models and the associated surrogate functions built for each problem. For HPosDec and HPosCount, gESMC correctly detects the linkages between the groups of variables $[x_0, x_1]$ and $[x_2, x_3]$; all the other variables are considered independent. Variables x_4 and x_5 are incorrectly identified as independent because gESMC reaches the termination criteria of 0% test error. For the problems HParDec and HParCount, gESMC discovers the linkage groups $[x_0, x_1]$, $[x_2, x_3]$, and $[x_4, x_5]$. Differently from the position problem, now gESMC needs to discover all the existing parity groups to remove the uncertainty, and so, build the most accurate classification model.

The availability of the structural model with gESMC is another advantage over other conventional classification techniques in terms of interpretability. The structural model facilitates easy visualization of the salient variable interactions; moreover, it permits a better understanding of the resulting surrogate function. Note that for all the problems, gESMC built easily interpretable functions and also efficiently ignored irrelevant variables. For example, consider the problem HParCount, in which the output is the number of '1s' resulting from the evaluation of each low-order parity block. The function evolved clearly indicates that if any of the linkage groups has the schemata '01' or '10' (values from which the parity would result in '1'), the output is incremented by one.

Let us now compare this knowledge representation to those obtained with C4.5 and SMO. For this purpose, we consider the size of the trees built by C4.5, and the machines constructed by SMO. For HPosDec and HPosCount,

```
at1 = 0
|   at2 = 0
|   |   at3 = 0
|   |   |   at4 = 0
|   |   |   |   at5 = 0
|   |   |   |   |   at6 = 0: 0 (153.0)
|   |   |   |   |   at6 = 1: 1 (150.0)
|   |   |   |   at5 = 1: 2 (314.0)
|   |   |   at4 = 1
|   |   |   |   at5 = 0
|   |   |   |   |   at6 = 0: 3 (154.0)
|   |   |   |   |   at6 = 1: 4 (139.0)
|   |   |   |   at5 = 1: 5 (311.0)
|   |   at3 = 1
|   |   |   at5 = 0
|   |   |   |   at6 = 0: 6 (277.0)
|   |   |   |   at6 = 1: 7 (299.0)
|   |   |   at5 = 1: 8 (632.0)
                . . .
```

Fig. 2. Portion of the tree built by C4.5 for the HPosDec problem

C4.5 built a tree with 53 nodes, from which 27 were leaves. The resulting trees specified the output for each combination of the six relevant bits (see a portion of a tree for HPosDec in Fig. 2). Although these trees detail the output given the value of the first six variables, they do not show the variable interactions. For HParCount, C4.5 built trees that, on average, had 136 leaves and 270 nodes. For HParDec, the trees had 142 leaves and 283 nodes. This high number of nodes makes the interpretability of tree very hard. Additionally, all the trees had some irrelevant attributes in the decision nodes. That is, C4.5 was overfitting the training instances to reduce the training error, resulting in more complicated trees, further hindering the interpretability of the classification model.

In contrast to gESMC and C4.5, SMO presented the less interpretable results. In general, SMO creates a machine for each pair of classes, and adjust $\ell + 1$ weights for each machine (where ℓ is the number of attributes of the problem). For HPosDec, SMO built 351 machines with 16 weights ranging from 0 to 1. For HPoscount, HParDec, and HParCount, 6, 6, and 28 machines were respectively created, all them with 16 weights ranging from 0 to 1. Although some of these weights were zero, the machines evolved could not be interpreted at all. Thus, the human expert would not be able to extract any information from these knowledge models.

Although both SMO and gESMC represent the knowledge in weights of functions that partition the search space, gESMC yields the structural model which permits easy visualization of salient variable interactions. Additionally, while SMO weights the input variables, gESMC weights the different subsolutions of the identified substructures. Classification models obtained via gESMC show the relative influence of subsolutions to the output and therefore gESMC's models are more easily interpretable than the ones created by SMO.

5.3 Results Increasing the Low Level Block Size

We now increase the interaction order of the lower-level blocks to analyze its effect on the performance of gESMC. Additionally, for the test problems with parity blocks, we also want to investigate the effect of having no guidance from lower-order substructures. Specifically, we want to analyze if this lack of guidance thwarts the search of gESMC toward the best structural form of the surrogate. For this purpose, we use HPosDec, HPosCount, HParDec, and HParCount with $\ell = 15$, $m = 2$, and $k = 3$, and compared gESMC to C4.5 and SMO.

Table 3 shows the test errors for gESMC, C4.5, and SMO. For HPosDec and HPosCount, gESMC obtained 0% error test and for both problems, two different structural models were created during independent runs:

$$Model_1 : [x_0 x_1][x_3 x_4 x_5][x_2][x_6][x_7][x_8][x_9][x_{10}][x_{11}][x_{12}][x_{13}][x_{14}],$$
$$Model_2 : [x_0 x_1 x_2][x_3 x_4][x_5][x_6][x_7][x_8][x_9][x_{10}][x_{11}][x_{12}][x_{13}][x_{14}].$$

In both the above models, the variables of one of the lower-level blocks are correctly identified, and only two variables of the other lower-level blocks form a linkage group. As observed in the previous section, this is because gESMC meets

Table 3. Test error and standard deviation obtained with gESMC, SMO and C4.5 on the problems HPosDec, HPosCount, HParDec, HParCount with ℓ=15, $m = 3$, and $k = 3$. Results are averages over ten runs with different holdouts and random seeds.

	gESMC	**C4.5**	**SMO**
HPosDec	0.00% ± 0.00%	0.00% ± 0.00%	0.00% ± 0.00%
HPosCount	0.00% ± 0.00%	0.00% ± 0.00%	14.24% ± 1.03%
HParDec	24.00% ± 25.50%	9.01% ± 5.86%	76.91% ± 2.01%
HParCount	49.99% ± 0.00%	12.25% ± 6.69%	49.94% ± 0.22%

the convergence criteria of 0% test errors even when one of the substructures is partially identified.

The surrogate functions evolved are qualitatively similar to those obtained in the previous section. In all cases, only the six relevant variables were taken in consideration, and specifically, some of their schemata. For example, one of the surrogate functions created for the HPosDec is

$$14.9\overline{x}_5 + 15x_5 - 2.5\overline{x}_3\overline{x}_4 + \overline{x}_3x_4 - 12\overline{x}_0\overline{x}_1\overline{x}_2 - 4\overline{x}_0x_1\overline{x}_2 - 8\overline{x}_0\overline{x}_1x_2 - 4\overline{x}_0x_1x_2, \tag{8}$$

which only contains the six relevant variables $x_0, x_1, \cdots, x_5$.

As expected, for HParDec and HParCount, gESMC yielded poorer results. For HParDec, the average test error was 24% with a high standard deviation. This high deviation is because gESMC yielded a maximally accurate classification model for 50% of the runs. For the rest 50%, gESMC could not discover an accurate structural model. Further investigation showed that this is due to the stochasticity of the holdout estimation. Since we randomly selected 70% of the instances as the training set, the symmetry of the parity problem may be broken leading the greedy search heuristic to yield the accurate structural model. This indicates that introduction of stochasticity might break symmetry of parity-like functions and render the accurate structural model hill-climbable. However, the efficacy of adding exogenous noise to break symmetry needs to be further investigated.

For HPosDec, gESMC was not able to discover the accurate structural model in any of the cases, and therefore yielded a test error of 50%. As mentioned earlier, the reason for this failure is due of the greedy search of the structural model. For the k-bit parity function, since all structural models with substructures of order $k-1$ or lower yield classification models with the same error, the optimal structural model is not hill-climbable. Therefore, the greedy search heuristic fails to identify the accurate structural model and so yields inaccurate classification models. This limitation can easily be alleviated in several number of ways, some of which are outlined in the next section.

Finally, we compare the results of gESMC to those obtained with C4.5 and SMO. All three algorithms perform equivalently in tackling HPosDec; differently, gESMC and C4.5 outperform SMO on HPosCount. However, on HParDec and HParCount, C4.5 outperforms both gESMC and SMO. Nonetheless, as with the

2-bit lower-order blocks, the trees of C4.5 had some irrelevant variables in the decision nodes indicating overfitting to the training data.

These results clearly show that gESMC can discover the accurate structural model provided that it is hill-climbable from lower-order structural models. In the following section, we discuss some approaches to relax this limitation of gESMC. We also discuss ways to represent structural models with overlapping substructures.

6 Discussion

The results presented in the previous section highlighted both the strengths and limitations of gESMC. In this section we discuss some approaches to overcome the limitations of gESMC which have to be further investigated. We discuss approaches to discover accurate structural models even when there is a lack of guidance from lower-level structural models. Moreover, we also address two new issues: how to deal with problems that present non-linearities in the high order function and also discuss ways to represent structural models with overlapping substructures.

6.1 Lack of Guidance from Lower-Order Substructures

As mentioned earlier, the greedy search used in gESMC needs some guidance from lower-order substructural models towards the optimal structural model. That is, in order to discover a k-variable substructure the greedy search needs a classification model built with at least one of the substructures of order 2 to be more accurate than that with substructures of order 1, and the classification model built with at least one of the substructures of order 3 has to be more accurate than those with substructures of order 2 and so on. In the absence of such a guidance, the greedy search may stop because it cannot find any structural model that decreases the classification error. To alleviate this limitation, we propose the following two approaches:

Increase the order of substructural merges. We can increase the order of the linkages that the greedy search does if the test error is high and no better structural model is found. That is, at each iteration, instead of pairwise merges, we could permit higher-order merges if the pairwise merges yield no improvement.

We implemented this approach and tested gESMC on the four hierarchical problems. The results show that gESMC obtained 0% test error in all the four problems, and the structural models were correctly evolved. However, the limitation of this approach is the increase in the complexity and cost of the algorithm which is dictated by the maximum order of linkages permitted (ℓ_{max}):

$$Cost = \binom{\ell}{2} \cdot s + \binom{\ell}{3} \cdot s + \cdots + \binom{\ell}{\ell_{max}} \cdot s, \tag{9}$$

where s is the cost of building a surrogate. Note that the cost of this approach increases with ℓ_{max}. For this reason, we do not consider this approach as a general solution, although it can be really useful in certain problem domains.

Select randomly one of the new structural models. If the test error is high, and the greedy search cannot find any structural model that significantly decreases this test error, a new structural model can be chosen randomly. More sophisticated approaches could be followed, such as using a technique based on *simulated annealing* [14]. In this case, we would accept a structural model with a higher error with the hope of getting a better model in the subsequent iterations.

Preliminary results using this strategy indicates that gESMC yields maximally accurate classification models for problems consisting of lower-order parity blocks with $k > 2$. However, the structural models evolved are slightly more complicated and contain spurious interactions between variables. Nevertheless, these spurious linkages can be removed by analyzing the classification model and the relative contribution of different schemata to the output.

6.2 Non-linearities in the High Order Functions

The problems designed for the experimentation consisted in higher order functions in which each variable contributed independently to the output. The search procedure could be easily replaced to be able to tackle problems with non-linearities in the higher order functions [4] more efficiently. For example, the greedy search of gESMC could be easily replaced by population-based search methods such as *genetic algorithms* [11,7]. This would permit to solve non-linearities in the higher order of the hierarchy at the cost of slightly increasing the computational time, since a population of candidate solutions should be evaluated and evolved.

6.3 Creating Structural Models with Overlapping Substructures

Finally, we look at problems with overlapping linkages where some variables interact with different groups of variables depending on the input. A widely used test problem with overlapping linkages is the *multiplexer* problem [12,31], which is defined as follows. Given a bit string of length ℓ, where the first $\log_2 \ell$ bits are the *address bits* and the remaining bits are the *position bits*, the output is the value of the position bit referred by the decimal value of the address bits. For example, for the 6-bit multiplexer, f($\underline{00}$ $\underline{0}$101)=0 and f($\underline{10}$ 10$\underline{1}$1)=1. Thus, a surrogate with a group formed by all the address bits and the corresponding position bit as a basis accurately determines the output.

We tested gESMC on the 6-bit and 11-bit multiplexer problems. The structural models evolved contained all the address and the position bits in the same linkage group. For example, we obtained the following structural model for the 6-bit multiplexer:

$$[x_0 x_1 x_2 x_3 x_4 x_5],$$

which resulted in a 0% test error. Since gESMC builds structural models with non-overlapping substructures, one way to handle overlapping substructures is by grouping the substructures together. However, such a merger is unnecessary and other methods which can build structural model with overlapping surrogates such as the design structure matrix genetic algorithm (DSMGA) [35,17], can evolve a structural model such as:

$$[x_0 x_1 x_2][x_0 x_1 x_3][x_0 x_1 x_4][x_0 x_1 x_5],$$

The above structural model also yields a surrogate with 0% test error, and gives more information than the former one. Therefore, we will investigate the use of DSMGA and other similar methods that can discover structural models with overlapping variables.

7 Summary and Conclusions

In this paper, we proposed a methodology for learning by building a classification model that uses the structural and surrogate model of a data set. First, we discover the structural model of a set of examples, identifying salient groups of interacting variables to determine the output. Then, the structural model is used to infer the functional form of a surrogate function and the coefficients of the surrogate are estimated using linear regression. Finally, using the substructural surrogate, we build a classification model to predict the class of a given new set of inputs.

We presented gESMC, an implementation of the methodology which uses a greedy search heuristic to search for the structural, surrogate, and classification models that minimize the classification error. Without any problem knowledge, gESMC starts with a simplest model of independent variables and proceeds to explore more complex structural models untill the classification error no longer improves or is below a user-defined threshold.

We ran gESMC on four hierarchical test problems. We compared the models evolved by gESMC with those created by C4.5 and SMO. The empirical observations evidenced that gESMC significantly outperforms C4.5 and SMO in problems that consisted of 2-bit low order blocks in terms of learning accuracy and interpretability. Moreover, one of the main differences between gESMC and other learners is highlighted: gESMC detects the structure of the data and uses it to predict the class of given inputs. In essence, gESMC not only yields accurate classification models, but also the classification models evolved are *interpretable*. That is, gESMC not only provides the classification model, but also the structure of the data, making it amenable to human interpretation.

Along with these strengths, the results also highlighted some limitations of the particular implementation of the methodology, gESMC. Specifically, the accuracy of the structural model to capture salient variable interactions depends on the guidance from lower-order substructures. Therefore, the accuracy of the structural model and consequently the accuracy of the classification model suffers when there is no guidance from lower-order substructures. This limitation is expected provided that we use a minimum description length style metric and also a greedy search heuristic that only considers pairwise merges of the substructures. Several approaches were outlined to overcome this limitation, serving as a basis for further research on substructural surrogates for learning decomposable classification problems.

Acknowledgments

We thank the support of *Enginyeria i Arquitectura La Salle*, Ramon Llull University, *Ministerio de Ciencia y Tecnología* under project TIN2005-08386-C05-04, and *Generalitat de Catalunya* under Grants 2005FI-00252 and 2005SGR-00302.

This work was also sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-03-1-0129, the National Science Foundation under grant ITR grant DMR-03-25939 at the Materials Computation Center. 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

1. Baluja, S.: Incorporating a priori Knowledge in Probabilistic-Model Based Optimization. In: Pelikan, M., Sastry, K., Cantú-Paz, E. (eds.) Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications ch. 9, pp. 205–219. Springer, Berlin (2006)
2. Bernadó-Mansilla, E., Garrell, J.M.: Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks. Evolutionary Computation 11(3), 209–238 (2003)
3. Butz, M.V.: Rule-Based Evolutionary Online Learning Systems: A Principled Approach to LCS Analysis and Design. In: Studies in Fuzziness and Soft Computing, vol. 109. Springer, Heidelberg (2006)
4. Butz, M.V., Pelikan, M., Llorà, X., Goldberg, D.E.: Automated Global Structure Extraction for Effective Local Building Block Processing in XCS. Evolutionary Computation 14(3), 345–380 (2006)
5. Drapper, N.R., Smith, H.: Applied Regression Analysis. John Wiley & Sons, New York (1966)
6. Gibson, J.J.: The Ecological Approach to Visual Perception. Lawrence Erlbaum Associates, Mahwah (1979)
7. Goldberg, D.E.: Genetic Algorithms in Search, Optimization & Machine Learning, 1st edn. Addison Wesley, Reading (1989)
8. Goldberg, D.E.: The Design of Innovation: Lessons from and for Competent Genetic Algorithms, 1st edn. Kluwer Academic Publishers, Dordrecht (2002)
9. Harik, G.: Linkage Learning via Probabilistic Modeling in the ECGA. Technical report. University of Illinois at Urbana-Champaign, Urbana, IL (January 1999) (IlliGAL Report No. 99010)

10. Harik, G.R., Lobo, F.G., Sastry, K.: 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 ch. 3, pp. 39–61. Springer, Berlin (2006) (Also IlliGAL Report No. 99010)
11. Holland, J.H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press (1975)
12. De Jong, K.A., Spears, W.M.: Learning Concept Classification Rules Using Genetic Algorithms. In: Proceedings of the International Joint Conference on Artificial Intelligence, Sidney, Australia, pp. 651–656 (1991)
13. Keerthi, S.S., Lin, C.J.: Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel. Neural Computation 15(7), 1667–1689 (2003)
14. Korst, J., Aarts, E.: Simulated Annealing and Boltzmann Machines. Wiley-Interscience, New York (1997)
15. Kovacs, T.: Deletion Schemes for Classifier Systems. In: GECCO 1999: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 329–336. Morgan Kaufmann, San Francisco (1999)
16. Llorà, X., Sastry, K., Goldberg, D.E., de la Ossa, L.: The χ-ary extended compact classifier system: Linkage learning in Pittsburgh LCS. In: Proceedings of the 2006 Genetic and Evolutionary Computation Conference Workshop Program. ACM Press, Berlin (2006) (Also IlliGAL Report No. 2006015)
17. Llorà, X., Sastry, K., Yu, T.-L., Goldberg, D.E.: Do not match, inherit: Fitness surrogates for genetics-based machine learning. In: Proceedings of the 2007 Genetic and Evolutionary Computation Conference, vol. 2, pp. 1798–1805 (2007)
18. Pelikan, M.: Hierarchical Bayesian Optimization Algorithm: Toward a new Generation of Evolutionary Algorithms. Springer, Berlin (2005)
19. Pelikan, M., Sastry, K.: Fitness inheritance in the Bayesian optimization algorithm. In: Proceedings of the 2004 Genetic and Evolutionary Computation Conference, vol. 2, pp. 48–59 (2004) (Also IlliGAL Report No. 2004009)
20. Pelikan, M., Sastry, K., Cantú-Paz, E. (eds.): Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications. Studies in Computational Intelligence, vol. 33. Springer, Heidelberg (2006)
21. Platt, J.: Fast Training of Support Vector Machines using Sequential Minimal Optimization. In: Advances in Kernel Methods - Support Vector Learning, pp. 557–563. MIT Press, Cambridge (1998)
22. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo (1995)
23. Rao, C.R., Toutenburg, H.: Linear Models: Least Squares and Alternatives. Springer, Berlin (1999)
24. Recktenwald, G.: Numerical Methods with MATLAB: Implementations and Applications. Prentice Hall, Englewood Cliffs (2000)
25. Sastry, K., Goldberg, D.E.: Probabilistic Model Building and Competent Genetic Programming. In: Riolo, R.L., Worzel, B. (eds.) Genetic Programming Theory and Practise, ch. 13, pp. 205–220. Kluwer, Dordrecht (2003)
26. Sastry, K., Lima, C.F., Goldberg, D.E.: Evaluation Relaxation Using Substructural Information and Linear Estimation. In: GECCO 2006: Proceedings of the 8th annual Conference on Genetic and Evolutionary Computation, pp. 419–426. ACM Press, New York (2006)
27. Sastry, K., Pelikan, M., Goldberg, D.E.: Efficiency enhancement of genetic algorithms via building-block-wise fitness estimation. In: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 720–727 (2004) (Also IlliGAL Report No. 2004010)

28. Simon, H.A.: Sciences of the Artificial. MIT Press, Cambridge (1969)
29. Dietterich, T.G.: Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. Neural Comp. 10(7), 1895–1924 (1998)
30. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
31. Wilson, S.W.: Quasi-Darwinian Learning in a Classifier System. In: 4th IWML, pp. 59–65. Morgan Kaufmann, San Francisco (1987)
32. Wilson, S.W.: Classifier Fitness Based on Accuracy. Evolutionary Computation 3(2), 149–175 (1995)
33. Wilson, S.W.: Generalization in the XCS Classifier System. In: 3rd Annual Conf. on Genetic Programming, pp. 665–674. Morgan Kaufmann, San Francisco (1998)
34. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
35. Yu, T.-L.: A matrix approach for finding extrema: Problems with modularity, hierarchy, and overlap. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL (2006)