Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 2337 – 2346

Procedia Computer Science

International Conference on Computational Science, ICCS 2013

A guided hybrid genetic algorithm for feature selection with expensive cost functions

Martin Jung[a,*], Jakob Zscheischler[a,b,c]

[a]*Max Planck Institute for Biogeochemistry, Hans-Knöll-Str. 10, 07745 Jena, Germany*
[b]*Max Planck Institute for Intelligent Systems, Speemanstr. 38, 72076 Tübingen, Germany*
[c]*Institute for Atmospheric and Climate Science, Eidgenössiche Technische Hochschule (ETH) Zurich, 8092 Zurich, Switzerland*

Abstract

We present a guided hybrid genetic algorithm for feature selection which is tailored to minimize the number of cost function evaluations. Guided variable elimination is used to make the stochastic backward search of the genetic algorithm much more efficient. Guiding means that a promising feature set is selected from a population and suggestions (for example by a trained Random Forest) are made which variable could be removed. It uses implicit diversity management and is able to return multiple optimal solutions if present, which might be important for interpreting the results. It uses a dynamic cost function that avoids prescribing an expected upper limit of performance or the number of features of the optimal solution. We illustrate the performance of the algorithm on artificial data, and show that the algorithm provides accurate results and is very efficient in minimizing the number of cost function evaluations.

Keywords: feature selection, genetic algorithm, Random Forests, diversity, niching, cost function

1. Introduction

One of the largest limitations of data-driven modeling is the choice of the right predictor variables. In some cases vast amounts of predictor variables are available but computational limits prohibit the use of all variables and the identification of informative predictors is desired. Feature selection aims to find a suitable (preferably the best) subset of features according to a specified cost function. The cost function can be formulated to choose the set of features with smallest error of the base learner [1], the smallest error of the base learner for a predefined number of included features [2], or the smallest number of included features for a targeted error margin [3].

Variants of sequential search algorithms [4] and genetic algorithms (GA) [3] are among the most frequently used feature selection approaches. While sequential methods tend to provide good results for small to medium sized problems (several tens of features), global search algorithms like GAs tend to be more suitable for large scale problems (> 100 features) since sequential methods get easily stuck in local minima [2]. GAs can also be prone to premature convergence if a 'too good solution' is found too early. Though this can be avoided by mechanisms that maintain genetic diversity in the population [5]. GAs essentially behave like a stochastic backward search: When the GA has found acceptable feature subsets it moves towards fewer features over the

*Corresponding author.
E-mail address: mjung@bgc-jena.mpg.de.

Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science
doi:10.1016/j.procs.2013.05.405

course of the search. However, this stochastic backward search is highly inefficient because of the randomized nature of the genetic operators (selection, mutation, crossover). Hybrid genetic algorithms [2] combine a GA with local search operations (sequential forward or backward selection) to move solutions proposed by the GA to local optima, i.e. the solutions are 'repaired'. But again, such a repairing mechanism is computationally very expensive and not feasible for large scale problems and expensive cost function evaluations. In this paper we propose a Guided Hybrid Genetic Algorithm (GHGA) for the following feature selection problem set:

- The evaluation of the cost function is extremely expensive (for example because it involves training a machine learning algorithm). This justifies operations of GHGA that are more computationally expensive than simple search strategies, while minimizing the number of necessary cost function evaluations.
- The number of candidate variables is large, e.g. in the order of several hundreds.
- The number of variables that are necessary to achieve an acceptable result is not known and should be found by the algorithm.
- The algorithm should be able to return several solutions (if they exist) to aid interpretability of the results (multiple optima).
- The assumption that unnecessary or not informative variables do not decrease the error should be avoided because this is frequently violated, e.g. if the ratio of variables to examples is large.

2. Description of the Guided Hybrid Genetic Algorithm (GHGA)

2.1. Notations

Chromosomes and features. We adopt the nomenclature of genetic algorithms where a chromosome c consists of individual genes. Thus, c is a bit-string and the genes encode which features are included (1) and excluded (0). The length of c is the total number of candidate features (genes) n. We denote the set of included features as c^1, and the set of excluded features as c^0, with n_{c^1} and n_{c^0} being their respective lengths.

Chromosome comparison. In the comparison of two chromosomes c_1 and c_2 we denote $u(c_1, c_2)$ as the set of common genes (either 0 or 1), where $n_{u(c_1,c_2)}$ is its length. We denote $u^1(c_1, c_2)$, and $u^0(c_1, c_2)$ as the set of common genes that are included, and excluded, respectively. Their corresponding lengths are $n_{u^1(c_1,c_2)}$, and $n_{u^0(c_1,c_2)}$ respectively. The set of genes that are not common among c_1 and c_2 is $x(c_1, c_2)$ with $n_{x(c_1,c_2)}$ being its length. We use the following measure to estimate the dissimilarity $d(c_1, c_2)$:

$$d(c_1, c_2) = 1 - \frac{n_{u^1(c_1,c_2)}}{\sqrt{n_{c_1^1} \cdot n_{c_2^1}}} \,. \tag{1}$$

Cost function. We denote the evaluation of the cost function for chromosome c as $[j(c), m(c)] = EVAL(X(c), Y)$, where $X(c)$ is the matrix of explanatory variables for c, and Y is the response variable. The evaluation returns the cost function value $j(c)$, and its modeling efficiency $m(c)$ [6]. The cost function is designed to extract sets with minimum number of included features for which the performance stays within a tolerance margin of the best known performance. The cost function value $j(c)$ is the sum of n_{c^1} and a penalty term p that depends on $m(c)$, the maximum m found over the course of the search $\max(M_A)$ and a specified tolerance ϵ (modified from [3]). Modeling efficiency is computed using cross-validation (e.g. using a test set), where Y_{CV} is the cross-validated predicted response variable.

$$j(c) = n_{c^1} + p(m(c), \max(M_A), \epsilon) \,, \tag{2}$$

$$p(m(c), \max(M_A), \epsilon) = e^{[\max(M_A) - m(c)]\frac{\ln(2)}{\epsilon}} - 1 = 2^{\frac{\max(M_A) - m(c)}{\epsilon}} - 1 \,, \tag{3}$$

$$m(c) = 1 - \frac{var(Y - Y_{CV}(c))}{var(Y)} \,. \tag{4}$$

For p it holds $0 \leq p \leq 1$ if $\max(M_A) - m(c) \leq \epsilon$. In contrary, if $\max(M_A) - m(c) >> \epsilon$, p rapidly increases helping the algorithm to concentrate the search in the promising region of the feature selection lattice. Since $\max(M_A)$ varies over the course of the search, j is recomputed for all individuals if a new $\max(M_A)$ is found. This dynamic cost function avoids prescribing a maximum modeling efficiency (e.g. based on the set with all features included).

	Symbol	Description
chromosome properties	c with length n	chromosome (bit string; 0: excluded, 1: included)
	c^0 with length n_{c^0}	set of excluded features in chromosome c
	c^1 with length n_{c^1}	set of included features in chromosome c
comparison of chromosomes c_1 and c_2	$u(c_1, c_2)$ with length $n_{u(c_1,c_2)}$	set of common genes (included or excluded)
	$u^0(c_1, c_2)$ with length $n_{u^0(c_1,c_2)}$	set of common excluded genes
	$u^1(c_1, c_2)$ with length $n_{u^1(c_1,c_2)}$	set of common included genes
	$x(c_1, c_2)$ with length $n_{x(c_1,c_2)}$	set of different genes (included or excluded)
	$d(c_1, c_2)$	dissimilarity
properties of evaluated chromosomes	$j \in J$	cost function value
	$m \in M$	modeling efficiency value
	$s \in S$	crowding index
	$h \in H$	flag: dominated (1) or not (0) by another individual
	$t \in T$	freq. of selections for guided variable elimination
	$z \in Z$	flag: inert (1) or not (0)
populations	A with length n_A	archive, stores all evaluated chromosomes
	R with length n_R	reproductive population, $R \subset A$
	B with length n_B	candidates for guided variable elimination, $B \subseteq R$

Table 1. Summary of notations.

Populations and sets. GHGA uses a hierarchical population structure:

- A denotes the archive with length n_A, which stores all chromosomes C_A and their corresponding sets of modeling efficiency values M_A, and cost function values J_A;
- R denotes the reproductive population of the GA with $R \subset A$, i.e. elements from C_R can be subject to genetic operators (selection, crossover, mutation);
- B denotes the population from which one individual is selected for local guided feature elimination, $B \subseteq R$.

We denote sets with capital letters, and elements of a set with its corresponding small letter. The length of the set is denoted as n with its corresponding set as subscript, e.g. n_A for the number of elements in A. We use the index k for elements of a population such that e.g. $c = C_A(k)$ and $j = J_A(k)$ is a chromosome from A and its corresponding cost function value at index k. Table 1 summarizes the notation.

2.2. Algorithm overview

Symbol	Description	Default value
ϵ	modeling efficiency tolerance parameter in the cost function	0.005
λ	number of iterations of phase 1	$2 \cdot n$
ψ	frequency (in number of cost function evaluations) of retraining $\mathbb{G}$	50
δ	dissimilarity threshold used in the construction of R	0.5
ρ	maximum number of similar chromosomes in R	5
θ	shape parameter of mutation rate decline used in phase 1	0.5
ϕ	mutation rate used in phase 2	0.05
κ	parameter controlling size of tournament in selection of a chromosome from B	5

Table 2. List of GHGA parameters with default values.

GHGA operates in two main phases. In the first phase only the GA operates with a declining mutation rate. After λ iterations (default = $2 \cdot n$) the second phase starts where the guided feature elimination is employed

in addition. We use a steady-state GA where the reproductive population R is newly extracted from A in each iteration. The reproduction involves the common procedure: parents are selected from R, their chromosomes are recombined using a crossover operation, and the resulting children are subject to mutation. In each iteration two children are generated and their cost function will only be evaluated if they do not have a duplicate in A to avoid redundant evaluations (omitted in Algorithm 1 for clarity). In the second phase of the algorithm, B is created from R and one individual selected from B is subject to local feature elimination using $\mathbb{G}$. $\mathbb{G}$ is a model trained on A and predicts $m(c)$ to guide the feature elimination. $\mathbb{G}$ is retrained after a certain number ψ (default=50) of cost function evaluations have passed since the last training of $\mathbb{G}$. The algorithm terminates if one of the following condition is met:

- no new best cost function value or modeling efficiency estimate or optimal result was found for a certain number cost function evaluations (recommended). Optimal result is defined as a chromosome with a modeling efficiency that is within the tolerance $\max(M_A) - m(c) < \epsilon$ and a number of included features that is equal to the minimum of included features in A where $\max(M_A) - m(c) < \epsilon$,
- the maximum allowed run time has passed,
- the maximum allowed number of function evaluations has passed.

Algorithm 1 gives an overview of GHGA. Below we provide more details on the individual steps of GHGA. All parameters of GHGA (given as greek letters) are listed in Table 2.

Algorithm 1 Guided Hybrid Genetic Algorithm

```
Initialize a fix number of random chromosomes; add a chromosome where all features are included
Evaluate chromosomes to generate the initial archive A (EVAL)
n_it = 1 (iteration counter), retrain = n_A (retrain counter for 𝔾)
while convergence criterion not met do
    Extract the reproductive population R from A (Section 2.3, Algorithm 2)
    Select two chromosomes from C_R for reproduction (Section 2.4)
    Cross them and get children c_1 and c_2 (Section 2.4)
    Calculate mutation rate from n_it; Apply mutation to c_1 and c_2 (Section 2.4)
    Evaluate c_1 and c_2 (EVAL)
    Add c_1, c_2, j(c_1), j(c_2), m(c_1), m(c_2) to A; update n_A
    if n_it > λ (algorithm is in the second phase) then
        if n_A > ψ + retrain (𝔾 should be retrained) then
            Train G based on A; set retrain = n_A
        end if
        Extract a subset B of the population R (Section 2.5, Algorithm 3)
        Select one chromosome c from B
        Do guided feature elimination for c using 𝔾 (involves EVAL; Section 2.5, Algorithm 4)
        Add evaluated chromosomes to A; update n_A
    end if
    n_it = n_it + 1
end while
```

2.3. Extraction of the reproductive population R from A

The way the population of chromosomes is managed is crucial for the performance of GAs in terms of the speed of convergence and diversity management to avoid premature convergence, and to favor the search for multiple distinct optima. The approach adopted here was inspired by work on multi-objective optimization using non-dominated sorting and crowding schemes [7, 8]. We first introduce some further properties of chromosomes: whether it is inert (z), whether it is dominated by other genes (h), and its crowding index (s). The crowding index s measures the crowdedness of chromosomes around chromosome c as:

$$s(c) = \frac{1}{n_{c^1}} \sum_{k=1}^{n_A} n_{u^1(c,C_A(k))}. \tag{5}$$

If there are many chromosomes in the archive that share many included features n_{c^1} then $s(c)$ is large.

A chromosome c is inert ($z = 1$) if the full set of 'leave one variable out' C_{lo} with $n_{C_{lo}} = n_{c^1}$ exists in the the archive. Inert chromosomes are basically flagged as (local) optima, since all combinations were tested with each included feature of c^1 being removed at a time. Inert chromosomes and are not considered when R is created, i.e. they are not allowed to reproduce. A chromosome c_1 is dominated ($h(c_1) = 1$) if there exists at least one other chromosome (c_2) in A where $c_2^1 \subset c_1^1$, where $j(c_2) \leq j(c_1)$. In other words, for dominated chromosomes there is another chromosome in the archive which has a subset of included features and at least equal performance.

A subset of A will be selected in R according to Algorithm 2 that aims at balancing the search in different promising regions of the feature selection lattice. A is sorted from best to worst according to J_A and the best individual $A(1)$ is added to R such that $n_R = 1$. Then we loop over the chromosomes of the sorted archive A using the index k. We assess whether the chromosome is dominated or inert. If not, we compute the dissimilarities D_k between the chromosome $C_A(k)$ and all in C_R, as well as its crowding index $S_A(k)$. We select $A(k)$ in R if its crowding index $S_A(k) < min(S_R)$, and if there are less than ρ (default=5) chromosomes already in R with a smaller dissimilarity to $C_A(k)$ than δ (sometimes referred as niche radius, default=0.5). The crowding scheme and truncating the number of similar chromosomes in R helps in maintaining genetic diversity to avoid premature convergence, and to allow for multiple distinct optima. The algorithm stops accumulating individuals in R if $M_A(k) < median(M_A)$. This is an elitist scheme where poor individuals are not allowed to reproduce.

Computing the crowding index, as well as the domination and inert conditions for each chromosome in A in each iteration would be very expensive. Computational costs can be kept at a minimum if a matrix N_{u^1} with size (n_A, n_A) is used that stores the pairwise number of common included features n_{u^1}. N_{u^1} is completed (updated) only for the relevant chromosomes (see index k in Algorithm 2) and does not need to contain all pairwise n_{u^1}.

Algorithm 2 Extraction of the active population R

```
Sort A ascending (best to worst) according to J_A, add A(1) to R, initialize counter k = 2
while M_A(k) > median(M_A) do
    Update N_u1(k); assess if chromosome is dominated or inert and get h and z
    Calculate dissimilarities of chromosome C_A(k) and all chromosomes in C_R
    Count number of dissimilarities n_δ that are smaller than δ
    if (h = 0 AND z = 0 & S_A(k) < min(S_R) & n_δ < ρ) then
        Add A(k) to R
    end if
    k=k+1
end while
```

2.4. Genetic operators

Here, we briefly describe how two chromosomes are selected from the reproductive population R, crossed, and mutated in the GA.

Selection. The first parent c_1 is selected randomly from R. Binary tournament selection [9] is used to select the second parent. The dissimilarities of two randomly taken individuals from R are compared and the one with a smaller dissimilarity to c_1 is selected as the second parent (c_2). Note that c_1 and c_2 can be identical which causes only mutation and no crossover. Dissimilarities between c_1 and C_R in the tournament selection was chosen to reduce the probability of lethal children which originate from too distant parents due to the search in different regions of the feature selection lattice. In addition this selection scheme has low selection pressure and therefore is beneficial for the diversity of the population.

Crossover. The genes of both selected parents (c_1 and c_2) are recombined based on the subset size oriented commonality crossover (SSOCF [10]), which had been designed for the purpose of feature selection. SSOCF ensures that each child has on average the same number of included features as either of the parents. SSOCF creates two children with $u^1(c_1, c_2)$. The first child inherits features of $x(c_1, c_2)$ with probability $\frac{n_{c^1} - n_{u^1(c_1,c_2)}}{n_{x(c_1,c_2)}}$. The

second child is complementary to the first and inherits those features of $x(c_1, c_2)$ which were not passed to the first.

Mutation. Both children are subject to subset size oriented mutation [2] such that we keep on average n_{c^1} after mutation. A mutation rate r^1 is used to decide which of c^1 are to be excluded. A mutation rate r^0 is used to decide which of c^0 are to be included with $r^0 = r^1 \cdot \frac{n_{c^1}}{n_{c^0}}$ [2]. We use a declining mutation rate $r^1 = \max\left(\frac{\lambda^\theta - n_{it}^\theta}{\lambda^\theta - 1} + \phi, \phi\right)$ as a function of the iteration number n_{it} during the first phase of the search, and keep it at a constant rate ϕ (default = 0.05) during the second phase of the search. λ is a parameter that controls at which iteration number $r^1 = \phi$ (default=$2 \cdot n$) and θ is a shape parameter (default=0.5).

2.5. Guided feature elimination

Extraction of the population B from R. The population B stores potential individuals that can be selected for guided feature elimination. B is identical to R if no inert individuals (local minima) exist yet. Otherwise, B is extracted from R according to Algorithm 3 such that it favors individuals that are more dissimilar from the inert individuals, which represent local minima.

Algorithm 3 Extraction of population B

```
Sort R ascending (best to worst) according to cost function values
For each chromosome in R calculate the smallest distance to all inert chromosomes and get D_R
Add R(1) to B (and D_R(1) to D_B)
for k = 2 : n_R do
    if D_R(k) > max(D_B) then
        Add R(k) to B
    end if
end for
```

Selection from B. One individual from B is selected for guided feature elimination using tournament selection. The size of the tournament scales with n_B and is $\lceil \frac{n_B}{\kappa} \rceil$, where κ is a parameter (default=5). We keep track of how often this chromosome has been selected for guided backward elimination $t(c)$ and increment this by one.

Guide $\mathbb{G}$. The role of $\mathbb{G}$ is to guide variable elimination for a selected chromosome. $\mathbb{G}$ can be in principle anything that provides an anticipated ranking of modeling efficiencies M_{lo}^* for all 'leave one variable out' combinations C_{lo}. Our default mode of $\mathbb{G}$ is Random Forests [11]. Random Forests is trained using A to get $\mathbb{G}$, where C_A is the matrix of explanatory (categorical) variables, and M_A is the response variable. We further implemented two simpler modes of $\mathbb{G}$: a random estimate, and the sum of selection frequencies. The latter simply sums the frequency of a feature being present in A for all features of the current feature subset.

Guided feature elimination algorithm. The chromosome c is declared as master and subject to guided feature elimination using $\mathbb{G}$ where $b(c)$ defines the maximum number of cost function evaluations $fcnt_{max}$ that are spent (Algorithm 4). $\mathbb{G}$ predicts the modeling efficiencies M_{lo}^* for all possible 'leave one feature out' combinations C_{lo}. Then the chromosomes of C_{lo} are evaluated in descending order of M_{lo}^* using the index k. If $J_{lo}(k) < j(c)$ then the search continues with $C_{lo}(k)$ being the new master chromosome. The guided feature elimination stops when the maximum number of function evaluations were reached, or when all possible C_{lo} were evaluated and no chromosome with a smaller cost function value than the master was found.

2.6. Discussion of GHGA parameters

The modeling efficiency tolerance parameter of the cost function ϵ measures the acceptable deviation from the highest (found) accuracy. Thus it controls the parsimony (simplicity-accuracy trade-off) of the result, and should be set according to user requirements. Most other parameters control the exploration-exploitation trade-off. For hard problems (e.g. many candidate features, strong correlation among features, several optima likely), the simplest way to enhance the exploration is to increase number of iterations where only the GA operates λ, and

Algorithm 4 Guided feature elimination algorithm

Define chromosome c as master, Set $fcnt_{max} = b(c)$, Set $flag_{master} = 1$
$fcnt = 0$ (Initialize counter of cost function evaluations)
while $fcnt \leq fcnt_{max}$ & $n_{c^1} \geq 1$ & $flag_{master} = 1$ **do**
  Generate all possible leave one variable out combinations for c and get C_{lo} with length c_{n^1}
  $M^*_{lo} = \mathbb{G}(C_{lo})$ (Predict modeling efficiencies for C_{lo} using $\mathbb{G}$)
  Sort C_{lo} according to descending M^*_{lo}
  $k = 1$ (Initialize counter), Set $flag_{master} = 0$
  while $k \leq c_{n^1}$ & $fcnt \leq fcnt_{max}$ & $flag_{master} = 0$ **do**
    if $C_{lo}(k) \notin A$ (current chromosome was not evaluated before) **then**
      Evaluate $C_{lo}(k)$ $(EVAL)$, add $C_{lo}(k)$ to A; $fcnt = fcnt + 1$
      if $J_{lo}(k) < j(c)$ (current chromosome is better than the master) **then**
        $c = C_{lo}(k)$; $j(c) = J_{lo}(k)$, Set $flag_{master} = 1$ (define $C_{lo}(k)$ as new master)
      end if
    end if
    $k = k + 1$
  end while
end while

to relax the termination criteria. If the search is targeted to find multiple very different optima (few features are shared among them) δ could be set to a large value (e.g. 0.8). If the aim is to find a good solution quickly then more power can be given to the exploitation by decreasing λ ; δ can be set to 1 (all chromosomes are considered similar, no 'niching'), and ρ to a desired maximum population size of R (e.g. 10); setting $\kappa = 1$ ensures that always the currently best chromosome will be selected for guided feature elimination. The suggested default parameters in Table 2 represent a compromise that work well for various problems.

3. Artificial Experiment

3.1. Generation of test data

We generate an artificial data set that has some particular properties that are common to many real world applications in earth sciences:

- several feature combinations have almost the same performance (multiple global optima; equifinality)
- the set of candidate features is large (250) and contains many non-informative variables
- some features are highly correlated with the set of informative variables (part of the global optima)
- the model based on a feature subset can be better than the model that included all features

We construct an artificial multivariate regression problem as follows. We denote a data set containing 250 variables and 1000 samples by X. X_i is the ith variable and $X_i(k)$ is sample k of variable i. Our aim is to construct a target variable Y such that there are four likewise optimal regressions from X to Y with each containing 10 variables out of all 250. X is step by step constructed as follows:

$$
\begin{aligned}
X_i &\sim N(0,1) && \text{for} \quad i = 1,\ldots,250,\\
X_i &= X_i \frac{\sum_{j=1}^{5} X_j}{\sum_{k=11}^{15} X_k} && \text{for} \quad i = 11,\ldots,15,\\
X_i &= X_i \frac{\sum_{j=1}^{10} X_j}{\sum_{k=11}^{25} X_k} && \text{for} \quad i = 16,\ldots,25,\\
X_i &= X_i \frac{\sum_{j=16}^{17} X_j}{\sum_{k=26}^{27} X_k} && \text{for} \quad i = 26,27,\\
X_i &= X_{i-27} + \xi, \quad \xi \sim N(0,0.2), && \text{for} \quad i = 28,\ldots,56.
\end{aligned}
$$

The target variable Y is then defined the sum of the first 10 variables and it can easily be verified that it holds:

$$Y := \sum_{i=1}^{10} X_i = \sum_{i=6}^{15} X_i = \sum_{i=16}^{25} X_i = \sum_{i=18}^{27} X_i \,,$$

i.e., there are four different combinations of respectively 10 regressors which can explain the target variable Y very well. To challenge the algorithm, the four optima were arranged such that two of them share 5 features (1...10; 5...15), and two of them share 8 features (16...25; 18...27). Finally, the data is split equally into a training data set plus some noise X^{tr} and a validation data set X^{val},

$$\begin{array}{lcll} X^{tr}(k) & = & X(k) + \xi & \text{for} \quad k = 1, \ldots, 500 \quad \text{and} \quad \xi \sim N(0, 0.1), \\ X^{val}(k) & = & X(k) & \text{for} \quad k = 501, \ldots, 1000. \end{array}$$

3.2. Experimental set-up

We ran GHGA with default settings on the data set, using a multiple linear regression as base function, and $\epsilon = 0.005$. The regression coefficients are estimated from the training data set, and the modeling efficiency is estimated using the test data set. We ran GHGA in four different modes: no guiding, i.e. just the GA; random guiding; guiding by variable selection frequency; guiding by the Random Forest algorithm. We performed 12 runs for each mode of GHGA to evaluate to what extent the feature selection results were by chance, and to estimate the statistical measures on the required number of cost function evaluations to find one, two, three, or all four global optima. The algorithm was run until all four global optima were found, or when 25,000 cost function evaluations were reached.

3.3. Results and Discussion

Value of a dynamic cost function. Our dynamic cost function approach updates all cost function values when a new maximum modeling efficiency was found, and thereby avoids specifying a maximum modeling efficiency a priori. Specifying, a maximum performance a priori would be done by using all features in the model, which is often inappropriate when many non-informative variables are present or the ratio of features to examples is large [1]. In our data set, modeling efficiency of the chromosome with all features included is 0.967, while the actual largest existing modeling efficiency is 0.999; this difference can be much more drastic in real world applications [12].

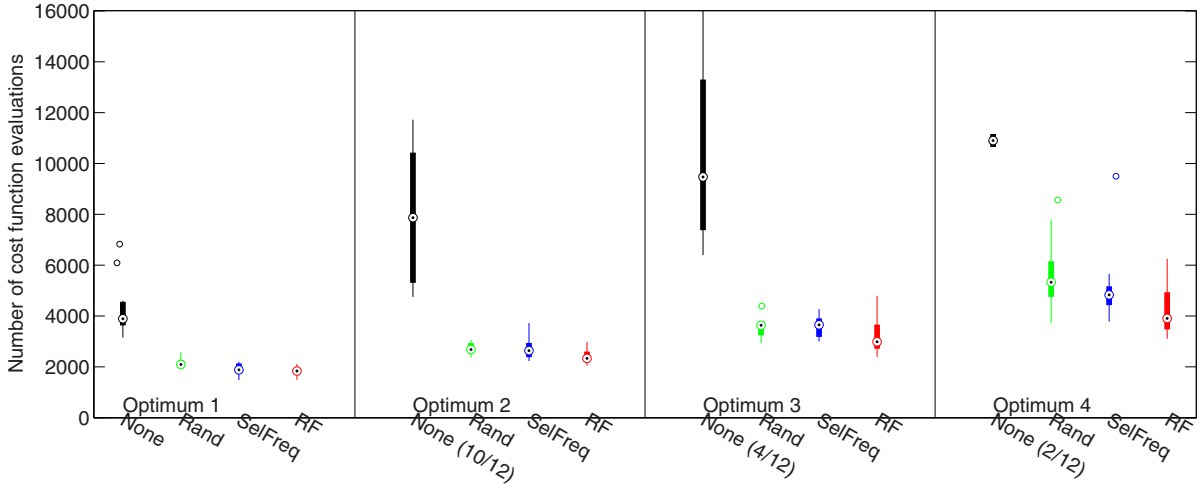

Fig. 1. Number of cost function required to find one, two, three, and all four optima for all modes of GHGA. Boxplots show the distribution of the 12 individual runs. Note that not all of the 12 runs without guidance found all optima (see numbers in parenthesis).

Multiple optima. All runs of all four modes of GHGA found one global optimum, while only the various guided versions found all four global optima for all runs (Figure 1). In 10/12 runs GHGA without guiding found 2 optima; in 4/12 runs it found 3 optima; in 2/12 runs all four optima were found. This demonstrates the capability of GHGA in finding multiple optima, which is based on its diversity management of the reproductive population. The diversity management and the steady state character of the GA setup where the population is created from all tested chromosomes is quite expensive and becomes increasingly expensive as more chromosomes are added to the archive. However, these costs are acceptable when the evaluation of the cost function is very expensive, and we show below that quite few cost function evaluations are needed by the algorithm to find the optima, which in the end greatly offsets the costs due to generation of the population after each iteration.

Efficiency of guiding. The number of cost functions evaluations required to find one optimum was on average half (about 2000) for the guided versions in comparison to the non-guided version. Differences in the number of required cost-function evaluations to find all four optima among the different guiding versions tends to increase with the number of optima. On average guiding by Random Forests required less cost function evaluations (on average 4185) than guiding by selection frequency (on average 5113) than random guiding (on average 5632) to find all four optima, i.e. guiding by Random Forest saves about 22 % and 35 % cost function evaluations in comparison to guiding by selection frequency and random guidance respectively.

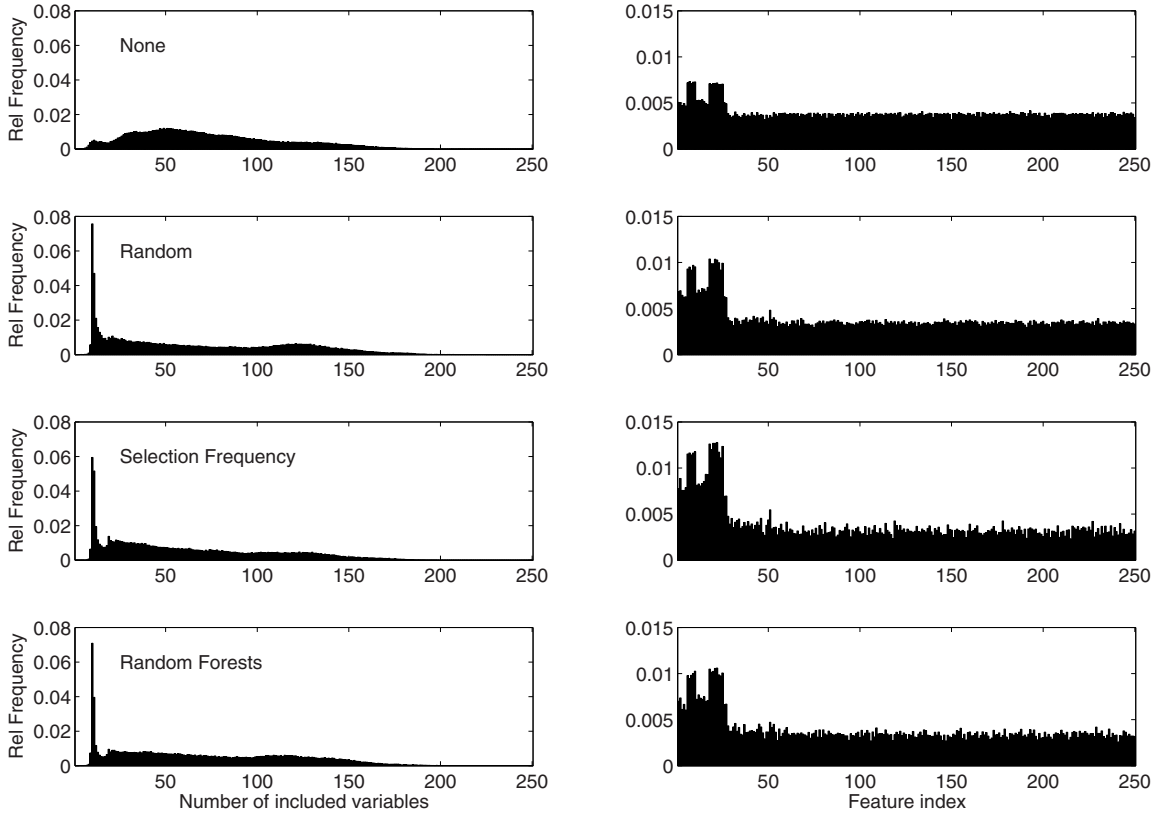

Fig. 2. Number of cost function evaluations as a function of number of included variables for all modes of GHGA (left) and selection frequency of the different features for all modes of GHGA (right). There is a clear peak in the number of cost function evaluations around the true optimum with 10 features included in the various guided versions of GHGA, which does not exist for the non-guided version. The feature selection frequency shows how often a feature was present in the archive and reveals a clear peak of those variables which are part of the optimal sets. The double peak marks those features that are part of two different optimal sets.

Finding four global optima from a data set with 250 variables with about 5000 cost function evaluations is

very efficient, and only a tiny fraction ($2.76 \cdot 10^{-72}$) of exhaustive search ($2^{250} - 1$). Clearly, guiding the search is much more efficient and always found the four optima in comparison to the GA without guiding. Interestingly, even the random guidance works well, which might imply that guiding itself is more important than how it is done. The guiding works essentially because the guiding mechanisms selects a promising chromosome from the population and tests modified chromosomes which have one feature less included. This greatly helps for the stochastic backward search nature of GAs; without guiding chromosomes with one feature less are suggested by the GA by chance, while guiding explicitly suggests chromosomes with one feature less included. Therefore, all the guided versions of GHGA concentrate the search on chromosomes with about 10 features included, while the non-guided version has large difficulties to get there and searches mainly in regions with more features included (Figure 2). The good performance of the random guiding found here might partly be due to the large number of uninformative features; if many uninformative features are present a random suggestion to eliminate a feature has high chances to be successful. Nevertheless, since random guiding or guiding by selection frequency have almost no computational costs in comparison to retraining Random Forests as guiding mechanism, random guiding and guiding by selection frequency are suitable options for less expensive cost function evaluations; i.e. when training Random Forests for guiding takes orders of magnitude more time than the evaluation of cost functions.

4. Conclusions

We presented a hybrid genetic algorithm for feature selection that is tailored to expensive cost function evaluations (GHGA). We illustrated the performance of the algorithm using artificial data, and show that the algorithm provides accurate results and is very efficient in minimizing the number of cost function evaluations. GHGA has many potential applications in Earth Sciences given the growing volume of multivariate data streams that warrant mining. An example is given in a companion paper [12].

Acknowledgements

Our research has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 283080 (GEOCARBON) / nr 244240 (ClimAfrica). JZ is part of the International Max Planck Research School for global Biogeochemical Cycles (IMPRS-gBGC).

References

[1] A. Jain, D. Zongker, Feature selection: Evaluation, application, and small sample performance, IEEE Transactions Pattern Analysis and Machine Intelligence 19 (2) (1997) 153–158.
[2] I.-S. Oh, J.-S. Lee, B.-R. Moon, Hybrid genetic algorithms for feature selection, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (11) (2004) 1424–1437.
[3] W. Siedlecki, J. Sklansky, A note on genetic algorithms for large-scale feature selection, Pattern Recognition Letters 10 (5) (1989) 335–347.
[4] P. Pudil, J. Novovicová, J. Kittler, Floating search methods in feature selection, Pattern recognition letters 15 (11) (1994) 1119–1125.
[5] A. AlSukker, R. Khushaba, A. Al-Ani, Enhancing the diversity of genetic algorithm for improved feature selection, in: International Conference on Systems Man and Cybernetics (SMC), 2010, pp. 1325–1331.
[6] J. Nash, J. Sutcliffe, River flow forecasting through conceptual models part I: A discussion of principles, Journal of Hydrology 10 (3) (1970) 282–290.
[7] D. Corne, J. Knowles, M. Oates, The pareto envelope-based selection algorithm for multiobjective optimization, in: Parallel Problem Solving from Nature PPSN VI, Lecture Notes in Computer Science Vol.1917, 2000, pp. 839–848.
[8] N. Srinivas, K. Deb, Multiobjective optimization using nondominated sorting in genetic algorithms, Evolutionary Computation 2 (3) (1995) 221–248.
[9] B. L. Miller, B. L. Miller, D. E. Goldberg, D. E. Goldberg, Genetic algorithms, tournament selection, and the effects of noise, Complex Systems 9 (1995) 193–212.
[10] C. Emmanouilidis, A. Hunter, J. Macintyre, A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator, in: Proc. of Congress on Evolutionary Computation, 2000, pp. 309–316.
[11] L. Breiman, Random forests, Machine Learning 45 (2001) 5–32.
[12] M. Jung, S. Tautenhahn, C. Wirth, J. Kattge, Estimating basal area of spruce and fir in post-fire residual stands in central siberia using quickbird, feature selection, and random forests, in: Procedia Computer Science, 2013, pp. xxx–xxx.