CovSel: A New Approach for Ensemble Selection Applied to Symbolic Regression Problems

Dominik Sobania
Johannes Gutenberg University
Mainz, Germany
dsobania@uni-mainz.de

Franz Rothlauf
Johannes Gutenberg University
Mainz, Germany
rothlauf@uni-mainz.de

ABSTRACT

Ensemble methods combine the predictions of a set of models to reach a better prediction quality compared to a single model's prediction. The ensemble process consists of three steps: 1) the generation phase where the models are created, 2) the selection phase where a set of possible ensembles is composed and one is selected by a selection method, 3) the fusion phase where the individual models' predictions of the selected ensemble are combined to an ensemble's estimate. This paper proposes CovSel, a selection approach for regression problems that ranks ensembles based on the coverage of adequately estimated training points and selects the ensemble with the highest coverage to be used in the fusion phase. An ensemble covers a training point if at least one of its models produces an adequate prediction for this training point. The more training points are covered this way, the higher is the ensemble's coverage. The selection of the "right" ensemble has a large impact on the final prediction. Results for two symbolic regression problems show that CovSel improves the predictions for various state-of-the-art fusion methods for ensembles composed of independently evolved GP models and also beats approaches based on single GP models.

CCS CONCEPTS

• **Computing methodologies** → **Ensemble methods**; *Supervised learning*; • **Software and its engineering** → *Genetic programming*;

KEYWORDS

Machine learning; Ensemble methods; Genetic programming; Symbolic regression

ACM Reference Format:
Dominik Sobania and Franz Rothlauf. 2018. CovSel: A New Approach for Ensemble Selection Applied to Symbolic Regression Problems. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan,* Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205570

GECCO '18, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205570

1 INTRODUCTION

Symbolic regression is one of the relevant applications for genetic programming (GP). Its application possibilities range from crypto analysis [31] to established products like Eureqa [8, 30]. Symbolic regression's ability to find an explanation for a set of observations in form of a mathematical function makes it attractive for various applied sciences. In contrast to other approaches like neural networks or support vector machines, the results are human-readable and can further be analyzed what can help researchers as well as practitioners to gain a better understanding of the studied problem.

Ensemble methods combine several models' predictions to achieve a better overall result in comparison to a single model [26]. A model can be the output of a machine learning algorithm (for example a GP run) and represents a mathematical term (used for symbolic regression) which can be used to make predictions for a given input. Ensemble methods are widely used in the field of machine learning and can be applied to combine classification or regression results to reach a better accuracy and a higher stability either by simple voting or by a combination of the estimations. Ensemble methods can be used to improve symbolic regression models but the question is how to build ensembles and create a prediction from the models that form an ensemble? Ensembles are often generated by simple greedy hill-climbing or pruning methods [3, 6]. However, these methods are usually bound to a certain fusion method that combines the individual models' predictions to an ensemble's estimate. Therefore, their running time depends mainly on the considered fusion method.

This paper suggests an ensemble selection method based on maximal coverage (CovSel) that ranks ensembles based on the coverage of adequately estimated training points and selects the ensemble with the highest coverage to be used for a fusion method. An ensemble covers a training point if at least one of its models produces an adequate prediction for this point. The more training points are covered, the higher is the ensemble's coverage.

The used models are evolved by isolated tree-based GP runs searching for solutions of symbolic regression problems. From each GP run, the best model on the training points is chosen to become part of a collection of elaborated regression models M. From the models contained in this collection M, we create a set E of candidate ensembles of pre-defined size by randomly selecting models from M. We do not include all possible ensembles as candidate ensembles as the number of combinations would be too large even for small ensembles sizes. CovSel selects the ensemble from E with the highest coverage on the training points. In contrast to other selection approaches, CovSel works independently from the applied fusion method because it focuses on the individual model's performance and estimates how the complete ensemble acts. After selecting the

best ensemble from E, a fusion method combines the individual models' estimates to the ensemble's overall estimation.

In Sect. 2, we present related work by describing the usage of ensembles in GP. Furthermore, we present work on ensemble selection. Section 3 describes CovSel. We specify how we build, select, and combine models. In particular, we introduce a novel selection approach and present four state-of-the-art fusion methods which can be used to combine individual estimations to an ensemble's prediction. In Sect. 4, we describe the experimental setting and present the results. Furthermore, we analyze the achieved results and compare CovSel's performance on different fusion methods.

2 RELATED WORK

Fault-tolerant systems are mainly based on redundancy. To make hardware-based systems fault-tolerant, critical elements can be integrated more than once. If one element fails, there is still a replacement. To make software-based systems fault-tolerant, a simple duplication of critical elements is not reasonable because in the same situation the same error would occur in all elements. For software-based systems, the error is a result of the software's inner logic which is identical in the duplicates. The N-version programming (NVP) method [4, 5] deals with this problem by implementing at least two independent but equivalent pieces of software based on the same specification. The necessary independence can, for example, be achieved through independent development teams. The aim of ensemble methods can be compared to this approach: a collection of trained models working together to correct or compensate errors.

2.1 Ensemble Methods for GP

Imamura et al. [16, 17] introduced N-value genetic programming (NVGP) by applying the idea of NVP to the GP context. NVGP evolves a collection of independent GP models, selects the best ones, and combines the models to perform classification by voting. An extension of NVGP introduces a selection approach, which calculates the expected failure probability of an ensemble (which contains models with an equal performance on the training set) using a method for calculating the failure probabilities of hardware systems [29]. After that, the achieved error on the training set of the considered ensemble is evaluated. Imamura et al. call an ensemble to be NVGP optimal if the difference between the expected error and the error on the training set is lower than a given threshold. NVGP optimal ensembles can be used for classification tasks because their failure rate is close to the expected rate which indicates that the contained models are independent [18].

Just like NVGP, most GP research that makes use of ensemble techniques focusses on classification problems, especially in the early years of ensemble usage in GP [11, 19, 24]. Recent work on classification is often addressing problems due to unbalanced data sets because it may have a large impact on the classifier's overall result if some classes are under-represented in the data set [1, 2, 10]. Such classifiers usually work well on classes with a good representation in the data set but achieve poor results on minority classes.

However, ensemble methods could not only be used for classification but also for regression problems. The aim of ensembles used for classification is to assign the correct class with a high probability by eliminating wrong predictions, e.g. by majority vote. For regression problems, the situation is different as the models' predictions must be combined to an ensemble's estimation in a way that the overall error is lower than the error of a single model. Existing work on this topic ranges from fundamental research [15, 21] to more application-oriented studies [13, 35]. Of particular relevance is the work of Veeramachaneni et al. [34] who compare different fusion methods for regression problems. Furthermore, they developed FlexGP [33], a cloud-based platform using ensemble methods and GP to deal with large regression problems.

2.2 Ensemble Selection

Ensemble methods can improve the prediction quality and stability, both for classification and for regression problems. However, a large ensemble may lead to a high computational effort and memory usage. Ensemble selection approaches usually tackle this problem by limiting the number of models in an ensemble.

In the field of GP, ensemble selection is not a widely used approach compared to the ensemble usage at all. For classification problems, Imamura et al. [18] used some kind of ensemble selection to find an ensemble containing independent GP models. Also for classification, De Stefano et al. [7] used a Bayesian network to select a subset of GP-based models to compose a well performing ensemble. For regression problems, no specialized selection approaches have yet been proposed.

In the field of general machine learning, often greedy ensemble selection methods are used [3, 14, 28]. The basic idea is to iteratively add models to an ensemble to maximize the ensemble's performance until the desired ensemble size is reached. Furthermore, greedy methods starting with a full ensemble and reducing it step-by-step are possible. A disadvantage of the greedy methods is that they are fusion method-dependent as a fusion method is required to evaluate the error of the ensemble during the process.

3 METHODOLOGY

In general, the ensemble process consists of a generation, selection, and fusion phase. We describe how we generate our symbolic regression models, introduce CovSel as a novel ensemble selection approach, and present four state-of-the-art fusion methods that combine individual model's estimates. Figure 1 illustrates how the ensemble process including CovSel works. Table 1 lists our problem notation which is based on Veeramachaneni et al. [34].

3.1 Generation Phase

In the generation phase, a set of candidate models is evolved. Although other machine learning approaches are possible, all of our models are generated by isolated GP runs using the same training points D_{GP}. The generation of models can be done either sequential or in parallel. To assess the quality of a candidate model m during the GP runs, we use as fitness function the *mean squared error (MSE)*

$$f(m) = \frac{1}{n} \sum_{j=1}^{n} (\hat{y}_{mj} - z_j)^2,$$

where $n = |D_{GP}|$. At the end of the generation phase, we choose models from the isolated GP runs to build the set M (Fig. 1). Usually,

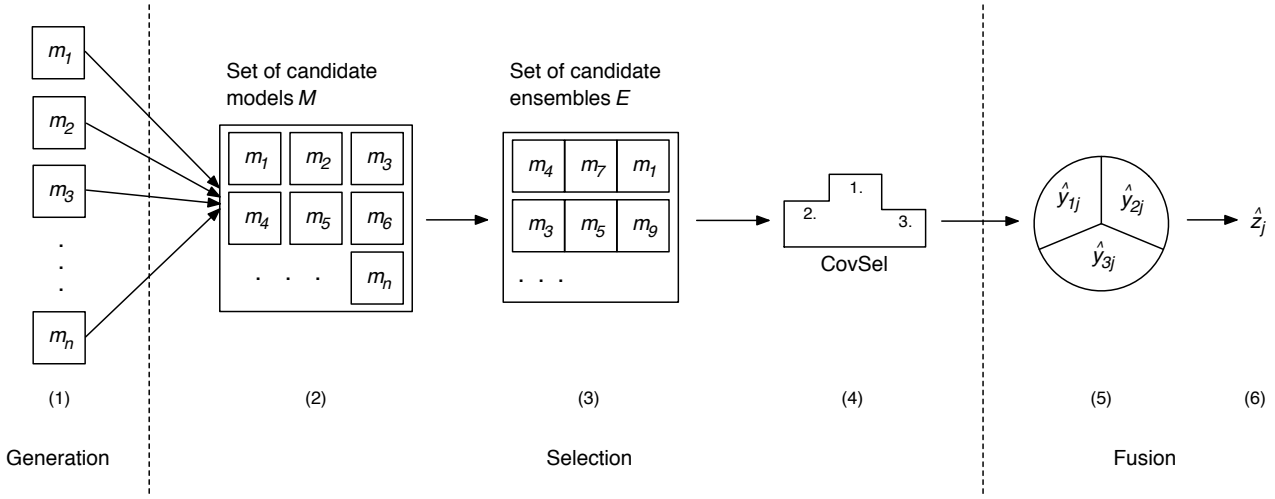

Figure 1: The ensemble process with generation phase, selection phase, and fusion phase.

Table 1: Problem Notation

Notation	Description
D_{GP}	Training points for GP
D_f	Training points for fusion methods
D_t	Testing points
$\bar{\boldsymbol{x}}_j$	Data point containing all coordinates
z_j	Correct result for a training point $\bar{\boldsymbol{x}}_j \in D_f$
m	Model
M	All models generated to select from
$\hat{y}_{mj}$	Model m's prediction for $\bar{\boldsymbol{x}}_j$
$f(m)$	Returns the fitness for a model m
Ω	An ensemble containing models, where $\Omega \subset M$
E	Set of candidate ensembles
$\vec{b}_m$	Model m's behavioral vector
$\vec{b}_\Omega$	Ensemble Ω's behavioral vector
G_Ω	Measure indicating Ω's quality
$d(x)$	Binary decision function for errors on D_{GP}
t	Problem-specific threshold
β	Sum of squared errors
$\hat{z}_j$	Ensemble's estimate

we choose from every isolated GP run the model m with the lowest MSE on D_{GP}. Due to the randomness of GP, the set M usually contains different models.

3.2 Selection Phase

After M is built, the models enter their first selection procedure. We create the set E by randomly selecting models from M (step 3 in Fig. 1). E consists of several candidate ensembles of size $|\Omega|$. The set E contains only a subset of all possible ensembles that can be built using the models in M because considering all possible combinations would be computationally too expensive.

All candidate ensembles in E can be ranked by CovSel (step 4 in Fig. 1). In contrast to voting-based ensembles for classification, which just select the class for which the majority of models in the ensemble votes, ensembles dealing with symbolic regression problems have to combine the individual models' estimates to an ensemble's estimate (and not only the majority's guess). It is important that the ensemble as a whole performs well on the complete range of the searched function. In other words, the ensemble's models have to reach in combination a high coverage. Consequently, we evaluate an ensemble Ω's coverage on the training points D_{GP} by

$$G_\Omega = \sum_{j=1}^{n} \prod_{m=1}^{o} d(|\hat{y}_{mj} - z_j|), \tag{1}$$

where $n = |D_{GP}|$ is the number of the training points, $o = |\Omega|$ is the number of models in Ω, and $d(x)$ is a function that assesses if a model m produces an adequate output for a training point. The coverage of the whole ensemble Ω depends on the coverage of each model m contained in the ensemble Ω. For classification problems, it is straightforward to determine if an estimation is correct as we only have binary decisions. For regression problems, we use a threshold t for this decision. Consequently, the function

$$d(x) = \begin{cases} 0 & \text{if } x \leq t \\ 1 & \text{if } x > t \end{cases}, \tag{2}$$

decides if a model's output is adequate or not. The setting of the threshold t allows us to adjust this approach to various problems.

Figure 2 illustrates the calculation of G_Ω for an example where $|\Omega| = 3$. We plot three behavioral vectors $\vec{b}_m$ ($m \in \{1, 2, 3\}$), which represent the models' results on the training points in D_{GP}. The elements of the behavioral vector $\vec{b}_m = (d(|\hat{y}_{m1} - z_1|), d(|\hat{y}_{m2} - z_2|), \ldots d(|\hat{y}_{mn} - z_n|))$ indicate whether the error is lower or equal than t (value of 0) or larger than t (value of 1). The $\vec{b}_m$ are used to build the ensemble's behavioral vector $\vec{b}_\Omega$. $\vec{b}_\Omega$ has a value of 1 at position j, if all vectors $\vec{b}_m$ have also a 1 at position j (logical AND). Otherwise, $\vec{b}_\Omega$ has a value of 0 at position j. All elements of

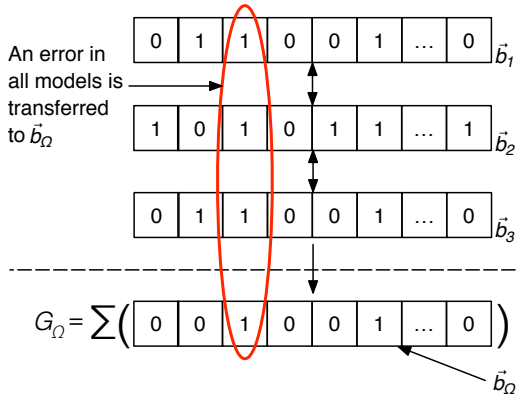

Figure 2: CovSel's calculation of G_Ω.

$\vec{b}_\Omega$ are summed up to get the ensemble's coverage measure G_Ω. A low value of G_Ω indicates a high coverage of an ensemble.

We compute G_Ω for every candidate ensemble in E. After that, we choose the ensemble with the lowest G_Ω for the fusion phase.

3.3 Fusion Phase

The fusion phase takes the ensemble Ω (containing $o = |\Omega|$ models) selected by the selection phase as input and calculates the estimation $\hat{z}_j$ for a testing point $\bar{\boldsymbol{x}}_j$ by usually combining the individual models predictions $\hat{y}_{mj}$ (step 5 and 6 in Fig. 1). For our experiments, we use four state-of-the-art fusion methods trained on the same training points $D_f = D_{GP}$.

(1) The *average (AVG)* method calculates for an input of $\bar{\boldsymbol{x}}_j$ the ensemble's estimate $\hat{z}_j$ by averaging over all $\hat{y}_{mj}$ [21, 35].

(2) The *median method (MED)* determines for an input of $\bar{\boldsymbol{x}}_j$ the median of all $\hat{y}_{mj}$ as estimate $\hat{z}_j$. In our experiments, we use MED instead of the *median average model (MAD)*, because MAD computes the average of the models' predictions $\hat{y}_{mj}$ close to the median and the median itself [21, 34] for a data point $\bar{\boldsymbol{x}}_j$. So for small ensembles this approach works like AVG.

(3) The *adaptive regression mixing (ARM)* estimates $\hat{z}_j$ by applying a weighted average to every $\hat{y}_{mj}$ [34, 38]. To calculate the weights W_m for each model m, we split the set of training points D_f randomly into two parts $D^{(1)}$ and $D^{(2)}$ of equal size $\frac{r}{2}$ (with $r = |D_f|$). For the sake of simplicity, we assume that r is an even number. After that, we compute the maximum likelihood estimate $\hat{\sigma}_m^2$ of the variance of the errors on $D^{(1)}$ and the sum of squared errors $\beta_m = \sum_{j=\frac{r}{2}+1}^{r} (\hat{y}_{mj} - z_j)^2$ on $D^{(2)}$. Then, we calculate the weights for each model m as

$$W_m = \exp(A_m - \log(\sum_{j=1}^{o} A_j)),$$

where

$$A_m = -\frac{r}{2}\log(\hat{\sigma}_m) + \log(\frac{-\hat{\sigma}_m^{-2}\beta_m}{2}).$$

Next, we split D_f randomly again into two new sets $D^{(1)}$ and $D^{(2)}$ and repeat the described steps for a pre-defined number of times. After that, we average the model's weights W_m. Finally, we calculate the weighted average using W_m for each model's estimate $\hat{y}_{mj}$ to get the ensemble's estimation $\hat{z}_j$.

(4) For the *kernel density estimation (KDE)*, we use a multivariate non-parametric kernel density estimator with a Gaussian kernel [9, 37]. The final estimation $\hat{z}_k$ for a test point $\bar{\boldsymbol{x}}_k$ is calculated as

$$\hat{z}_k = \frac{\sum_{j=1}^{n} z_j \prod_{m=1}^{o} K_h(\hat{y}_{mk} - \hat{y}_{mj})}{\sum_{j=1}^{n} \prod_{m=1}^{o} K_h(\hat{y}_{mk} - \hat{y}_{mj})},$$

where $o = |\Omega|$, $n = |D_f|$, and $K_h(x) = \frac{1}{h}\phi(\frac{x}{h})$. $\phi(x)$ is the standard normal density function with bandwidth $h \in \mathbb{R}$. h has a large impact on the result, so it should be a-priori estimated on D_f [34].

An advantage of symbolic regression as application of GP in contrast to other approaches like neural networks or support vector machines is that the results are human-readable. A domain expert could use the resulting expression for further analysis to gain a better understanding of the studied problem. When using ensembles, the combination of the individual models' predictions in the fusion phase is a limiting factor for the human-readability especially when applying more advanced fusion methods like KDE. The individual models can still be used for further analysis, but it must be considered how they contribute to the ensemble's overall prediction. However, when applying GP to symbolic regression problems the hypothesis space is limited by the function and the terminal set and maybe the target function cannot be expressed. The combination of individual models in the fusion phase may help to overcome this limitation by expanding the hypothesis space and so maybe a better approximation can be found [39].

4 EXPERIMENTS

We study the performance of CovSel for each of the four fusion methods presented in Sect. 3.3. As a baseline, we compare CovSel to a randomly selected ensemble from E as well as approaches using either the best or a random single GP model from M.

We evaluate CovSel for two benchmark problems. First, a bivariate function defined by Pagie and Hogeweg [27] using Koza's function set [23]. Second, a bivariate test function by Vladislavleva et al. [36]. Both test functions are taken from the curated list of benchmarks by McDermott et al. [25].

We did not use any other benchmark functions with Koza's function set because these benchmarks can be solved with zero or almost zero error by a single standard GP model. The test functions proposed by Koza [22, 23] and Nguyen et al. [32] contain only elements that can be build straightforward with the elements from the function set. That is why we selected the bivariate function by Vladislavleva et al. as second benchmark.

4.1 Experimental Setting

For generating the candidate models of the set M, we used the GP implementation of the evolutionary computation framework DEAP

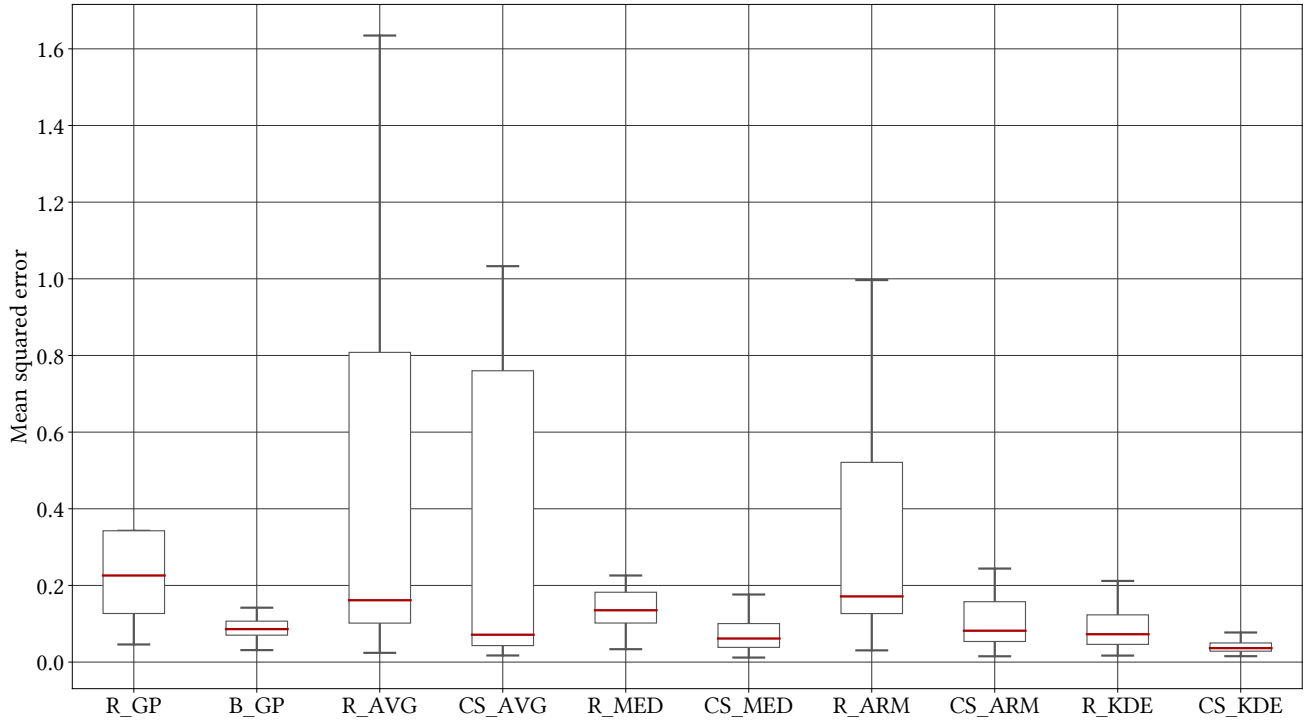

Figure 3: Boxplots of the MSE for the different approaches on Pagie1 with $|\Omega| = 3$ (outliers not plotted).

[12]. The initial generation is generated with the ramped-half-and-half method. As variation operators, we use the framework's default one-point crossover and uniform mutation function. Crossover probability is set to $p_c = 0.8$ and mutation probability is set to $p_m = 0.05$. For selection, we use tournament selection of size 5. We use a population size of 300 and stop each GP run after 35 generations.

We generate GP models for two benchmark functions. The first problem from Pagie and Hogeweg [27] (denoted as Pagie1) is defined as

$$b(x, y) = \frac{1}{1 + x^{-4}} + \frac{1}{1 + y^{-4}},$$

where $x, y \in \mathbb{R}$ are the function arguments. As function set we use the set proposed by Koza [23] without the optional constants

$$\{+, -, *, \%, \sin(n), \cos(n), e^n, \ln(|n|)\}.$$

The terminal set contains just the function arguments x and y. The training points D_{GP} for Pagie1 are the points of a meshed grid with coordinates ranging from -5.0 to 5.0 evenly spaced with distance 0.4. As test points D_t, we use a more dense grid with coordinates ranging from -6.0 to 6.0 evenly spaced with distance 0.1.

The second test problem from Vladislavleva et al. [36] (denoted as Vladislavleva6) is defined as

$$b(x, y) = 6 \sin(x) \cos(y),$$

where $x, y \in \mathbb{R}$ are the function arguments. For this problem, the function set is defined [25, 36] as

$$\{+, -, *, \%, n^2, e^n, e^{-n}, n^\epsilon, n + \epsilon, n\epsilon\},$$

where ϵ is a uniform random value in the interval $[-5.0, 5.0]$. Again, the terminal set contains just the function arguments x and y. The 50 training points D_{GP} are equally distributed in the two-dimensional square $[0.1, 5.9] \times [0.1, 5.9]$. As testing points D_t, we use the points of a meshed grid with coordinates ranging from -0.05 to 6.05 evenly spaced with distance 0.1.

Vladislavleva6 is a challenging test function for a standard GP implementation because 1) the number of training points is low and 2) symbolic regression has to find a way to approximate the functions $\sin(n)$ and $\cos(n)$. Due to the difficulty of the problem, Vladislavleva et al. [36] and Keijzer [20] apply more advanced techniques like interval arithmetic and linear scaling to build qualitative regression models for this benchmark function.

For both test problems, we evolve models by 50 independent GP runs and use from each GP run the model with the lowest MSE on D_{GP}. The selected models form the set M of size $|M| = 50$. The set E contains $|E| = 100$ ensembles and each of the 100 ensembles contains $o = |\Omega|$ randomly selected models from M. As the number of possible ensembles is large and equal to $\binom{|M|}{o}$, E contains only a subset of all possible ensembles. We present results for different ensemble sizes $o \in \{3, 5, 7\}$.

CovSel evaluates all 100 ensembles of E and selects the one with highest coverage on the training points D_{GP}. The threshold t used by CovSel's function $d(x)$ determines the maximum accepted error for a training point $\bar{\boldsymbol{x}}_j$. For Pagie1, we set $t = 0.1$. As Vladislavleva6 is a more difficult problem for standard GP models, we set $t = 1.0$.

Table 2: MSE and IQR (in brackets) for Pagie1 and Vladislavleva6 using different ensemble sizes $|\Omega|$

Benchmark	$\|\Omega\|$	R_GP	B_GP	R_AVG	CS_AVG	R_MED	CS_MED	R_ARM	CS_ARM	R_KDE	CS_KDE
Pagie1	3	0.226 (0.216)	0.086 (0.037)	0.161 (0.706)	0.071 (0.717)	0.135 (0.080)	0.062 (0.062)	0.171 (0.394)	0.082 (0.104)	0.073 (0.077)	**0.036** (**0.021**)
	5	0.226 (0.216)	0.086 (0.037)	0.307 (0.292)	0.114 (0.295)	0.115 (0.054)	0.054 (0.045)	0.153 (0.160)	0.080 (0.165)	0.040 (0.036)	**0.027** (**0.019**)
	7	0.226 (0.216)	0.086 (0.037)	0.193 (0.134)	0.181 (0.181)	0.125 (0.077)	0.067 (0.052)	0.134 (0.145)	0.111 (0.143)	0.029 (0.021)	**0.020** (**0.013**)
Vladislavleva6	3	17.150 (2627.797)	10.136 (130.656)	631.456 (1.3e10)	567.506 (1.8e14)	8.869 (14.938)	8.421 (16.224)	131.668 (5.0e6)	645.808 (1.9e10)	8.048 (**1.364**)	**6.898** (1.655)
	5	17.150 (2627.797)	10.136 (130.656)	39.270 (4.8e12)	1898.552 (3.3e26)	8.871 (5.118)	8.476 (4.526)	29.781 (4.5e6)	1784.984 (1.1e26)	**6.257** (**2.512**)	6.465 (2.649)
	7	17.150 (2627.797)	10.136 (130.656)	220.850 (1.1e26)	2.3e5 (3.0e36)	8.503 (**0.670**)	8.041 (0.686)	123.366 (8.4e12)	216.289 (2.8e22)	6.519 (1.861)	**6.193** (1.836)

For the fusion methods ARM and KDE, we need to define a set of training points D_f. We use the same training points used for evolving the GP models $D_f = D_{GP}$. For KDE, we also need to define the bandwidth h, which we set to $h = 0.5$.

4.2 Results and Discussion

We combined CovSel with four different state-of-the-art fusion methods. The resulting approaches are denoted as CS_AVG (CovSel combined with fusion method AVG), CS_MED (median method), CS_ARM (adaptive regression mixing), and CS_KDE (kernel density estimation), respectively. As benchmark for CovSel, we randomly selected an ensemble from the set E. The resulting methods are denoted as R_AVG (random selection combined with fusion method AVG), R_MED, R_ARM, and R_KDE, respectively. Furthermore, we used as baseline the prediction quality of just a single GP model. The single model is either randomly selected from M (denoted as R_GP) or the model of M that performs best on the set of training points D_{GP} (denoted as B_GP).

The boxplot in Fig. 3 shows the median and the scattering of the MSE of the different approaches for an ensemble size of $|\Omega| = 3$ on the benchmark function Pagie1. The boxplots are based on 100 runs for every approach. As expected, selecting the best model from M (B_GP) outperforms a random model taken from M (R_GP). Furthermore, the median of B_GP is lower than the median of R_AVG, R_MED, and R_ARM. In contrast, the median of all CovSel variants is lower than the median of B_GP. Focusing on the scattering of the results, CovSel leads to lower scattering in comparison to a randomly selected ensemble. The results show that the average fusion methods (CS_AVG and R_AVG) lead to high scattering. This is due to the fact that the averaging fusion methods are sensitive to outliers as they consider the estimates of all models in an ensemble.

Table 2 compares the median MSE as well as the interquartile range (IQR) (in brackets) for Pagie1 and Vladislavleva6 for different ensemble sizes $o \in \{3, 5, 7\}$, where $o = |\Omega|$. Best values are printed in bold fonts. The IQR is the difference between the 75^{th} and the 25^{th} percentile and measures the scattering of a set of values. We use the IQR as a proxy of the MSE's spread. The results for R_GP and B_GP are independent of $|\Omega|$ as these approaches use only a single model $m \in M$.

For Pagie1, CS_KDE has the lowest MSE and also the lowest IQR. When using kernel density estimation (CS_KDE and R_KDE), the median MSE and IQR get lower with larger ensemble size $|\Omega|$. With increasing $|\Omega|$, the differences between CovSel and a randomly selected ensemble are decreasing as the importance of an optimal ensemble selection decreases with a larger size of the ensembles. CS_MED achieves a lower median MSE and IQR for all ensemble sizes $|\Omega|$ compared to R_MED. The results for R_MED and CS_MED are stable over all studied ensemble sizes $|\Omega|$.

For Vladislavleva6, the medians and IQRs are higher compared to Pagie1. Again, B_GP outperforms R_GP on median MSE and IQR. The higher spread of the results confirms that this benchmark is a hard problem for standard GP. All methods using a fusion method based on averaging (R_AVG, CS_AVG, R_ARM, and CS_ARM) do not work well as the performance of the models in one ensemble is too widely spread and does not give a clear direction for averaging methods. CS_KDE as well as R_KDE return ensembles with lowest MSE and a low IQR. Like in Pagie1, when using kernel density estimation (CS_KDE and R_KDE), the median MSE and IQR get lower with larger ensembles sizes $|\Omega|$.

Overall, using CovSel for ensemble selection can improve prediction quality compared to randomly selected ensembles and outperforms methods based on a single GP model. Best results can be obtained when combining CovSel either with the median method or a kernel density estimation.

4.3 Robustness of KDE with Respect to Bandwidth

The quality of R_KDE and CS_KDE depends on the pre-defined bandwidth h [34]. Especially for small training sets, a proper setting of h is difficult. We study the impact of h on the resulting

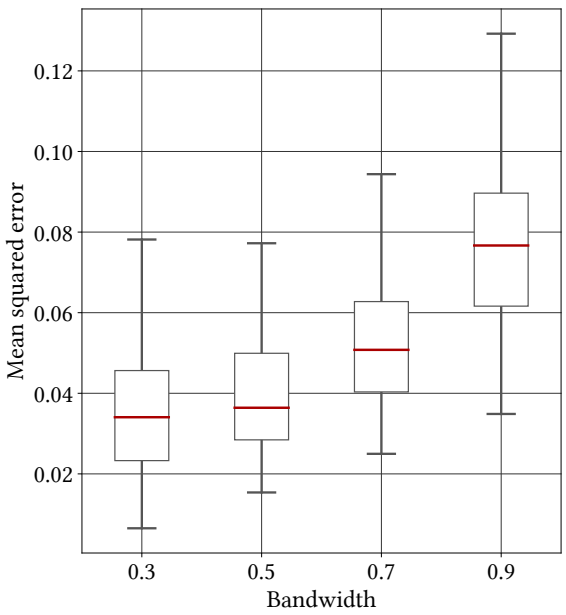

Figure 4: Boxplots of the MSE over bandwidth h for Pagie1

MSE by comparing the MSE of CS_KDE for different values of $h \in \{0.3, 0.5, 0.7, 0.9\}$.

Figure 4 shows boxplots of the MSE over the bandwidth h for CS_KDE with an ensemble size of $|\Omega| = 3$ on the Pagie1 benchmark function. For $h = 0.3$, CS_KDE achieves a median MSE of 0.034 which is even lower than the result presented in Fig. 3 with $h = 0.5$. We find that a larger value of h decreases the prediction quality of CS_KDE for Pagie1. For example, the median for $h = 0.9$ is only 0.077. Nevertheless, the results are robust for different values of h and CS_KDE still returns good results. Furthermore, the IQR remains low for all studied bandwidths.

5 CONCLUSIONS

Symbolic regression is one of the relevant GP applications. Ensemble methods can combine several models to obtain a higher accuracy and stability of the results. Currently, no specialized ensemble selection methods are available for symbolic regression problems.

This paper proposes CovSel, an approach that ranks ensembles based on the coverage of adequately estimated training points and selects the ensemble with the highest coverage to be used for a fusion method. An ensemble covers a training point if at least one of its models returns an adequate prediction for this training point. The more training points are covered this way, the higher is the ensemble's coverage.

The symbolic regression models that form ensembles can be evolved by different, isolated GP runs. From each GP run, the model with the lowest MSE on the training points is chosen to build a collection of elaborated regression models. By using the models in this collection, a set of random candidate ensembles is created. From this set of candidate ensembles, which is a random subset of all possible ensembles, CovSel selects the ensemble with highest coverage. Finally in the fusion phase, the individual models' estimates are combined to the ensemble's estimation. CovSel works independently from the applied fusion method because it focuses on the individual model's performance and estimates how the complete ensemble acts.

We applied CovSel in combination with four state-of-the-art fusion methods and compared ensembles selected by CovSel to randomly selected candidate ensembles as well as the best and a random GP model taken from the set of elaborated regression models. Our results confirm that CovSel outperforms randomly selected ensembles. Furthermore, combining CovSel with kernel density estimation as fusion method outperforms the prediction quality of the single, best model.

6 LIMITATIONS AND FUTURE WORK

CovSel ranks ensembles based on the coverage of adequately estimated training points and selects the ensemble with the highest coverage to be used in the fusion phase. A training point is considered as covered if at least one model approximates the result within the pre-defined threshold t. Thus, CovSel is tuned to work for ensembles dealing with regression problems and should not be used for the selection of ensembles for classification problems because such ensembles attempt to exclude wrong predictions to correct the errors (e.g. by majority vote). In contrast, ensembles for regression problems combine or select the models' predictions to a combined prediction such that the error is reduced and that is why the methods CS_MAD and CS_KDE work that well, because CovSel attempts to provide at least one model with a low error for every part of the searched function.

In future work, we will compare CovSel with current greedy selection approaches and try to come up with a greedy ensemble selection approach that works also independently from the fusion method. Furthermore, we will study the diversity of CovSel and integrate methods which increase the diversity of the models in an ensemble to obtain more diverse ensembles that still have a high coverage. We will apply this new version of CovSel to real-world symbolic regression problems.

REFERENCES

[1] Urvesh Bhowan, Mark Johnston, Mengjie Zhang, and Xin Yao. 2013. Evolving diverse ensembles using genetic programming for classification with unbalanced data. *IEEE Transactions on Evolutionary Computation* 17, 3 (2013), 368–386.
[2] Urvesh Bhowan, Mark Johnston, Mengjie Zhang, and Xin Yao. 2014. Reusing genetic programming for ensemble selection in classification of unbalanced data. *IEEE Transactions on Evolutionary Computation* 18, 6 (2014), 893–908.
[3] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. 2004. Ensemble selection from libraries of models. In *Proceedings of the 21st International Conference on Machine learning*. ACM, New York, NY, 18.
[4] Liming Chen and Algirdas Avizienis. 1977. On the implementation of n-version programming for software fault tolerance during program execution. In *International Computer Software and Applications Conference*. IEEE, New York, NY, 149–155.
[5] Liming Chen and Algirdas Avizienis. 1978. N-version programming: A fault-tolerance approach to reliability of software operation. In *Proceedings of the 8th Annual International Conference on Fault Tolerant Computing*. IEEE Computer Society, Washington, DC, 3–9.
[6] Qun Dai. 2013. A novel ensemble pruning algorithm based on randomized greedy selective strategy and ballot. *Neurocomputing* 122 (2013), 258–265.

[7] Claudio De Stefano, Gianluigi Folino, Francesco Fontanella, and A Scotto Di Freca. 2012. Pruning GP-Based classifier ensembles by Bayesian networks. In *International Conference on Parallel Problem Solving from Nature*. Springer, Berlin, Heidelberg, 236–245.

[8] Renáta Dubčáková. 2011. Eureqa: Software review. *Genetic Programming and Evolvable Machines* 12, 2 (2011), 173–178.

[9] Vassiliy A Epanechnikov. 1969. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications* 14, 1 (1969), 153–158.

[10] Gianluigi Folino and Francesco Sergio Pisani. 2015. Combining ensemble of classifiers by using genetic programming for cyber security applications. In *European Conference on the Applications of Evolutionary Computation*. Springer, Cham, 54–66.

[11] Gianluigi Folino, Clara Pizzuti, Giandomenico Spezzano, et al. 2007. Mining distributed evolving data streams using fractal GP ensembles. In *EuroGP*. Springer, Berlin, Heidelberg, 160–169.

[12] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 2012. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research* 13 (july 2012), 2171–2175.

[13] Crina Grosan and Ajith Abraham. 2006. Stock market modeling using genetic programming ensembles. In *Genetic Systems Programming: Theory and Experiences*. Springer, Berlin, Heidelberg, 131–146.

[14] Daniel Hernández-Lobato, Gonzalo Martínez-Muñoz, and Alberto Suárez. 2006. Pruning in ordered regression bagging ensembles. In *International Joint Conference on Neural Networks*. IEEE, New York, NY, 1266–1273.

[15] Hitoshi Iba. 1999. Bagging, boosting, and bloating in genetic programming. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., Orlando, FL, 1053–1060.

[16] Kosuke Imamura and James A Foster. 2001. Fault-tolerant computing with N-version genetic programming. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 178–178.

[17] Kosuke Imamura, Robert Heckendorn, Terence Soule, and James Foster. 2002. N-version genetic programming via fault masking. *Genetic Programming* 2278 (2002), 172–181.

[18] Kosuke Imamura, Terence Soule, Robert B Heckendorn, and James A Foster. 2003. Behavioral diversity and a probabilistically optimal GP ensemble. *Genetic Programming and Evolvable Machines* 4, 3 (2003), 235–253.

[19] Ulf Johansson, Tuve Löfström, Rikard König, and Lars Niklasson. 2006. Genetically evolved trees representing ensembles. In *International Conference on Artificial Intelligence and Soft Computing*. Springer, Berlin, Heidelberg, 613–622.

[20] Maarten Keijzer. 2003. Improving symbolic regression with interval arithmetic and linear scaling. In *European Conference on Genetic Programming*. Springer, Berlin, Heidelberg, 70–82.

[21] Mark Kotanchek, Guido Smits, and Ekaterina Vladislavleva. 2008. Trustable symbolic regression models: using ensembles, interval arithmetic and pareto fronts to develop robust and trust-aware models. *Genetic Programming Theory and Practice* 5, 1 (2008), 201–220.

[22] John R Koza. 1992. *Genetic Programming II, Automatic discovery of reusable subprograms*. MIT Press, Cambridge, MA.

[23] John R Koza. 1992. *Genetic programming: On the programming of computers by means of natural selection*. Vol. 1. MIT press, Cambridge, MA.

[24] William Langdon, Steven Barrett, and Bernard Buxton. 2002. Combining decision trees and neural networks for drug discovery. *Genetic Programming* 2278 (2002), 76–89.

[25] James McDermott, David R White, Sean Luke, Luca Manzoni, Mauro Castelli, Leonardo Vanneschi, Wojciech Jaskowski, Krzysztof Krawiec, Robin Harper, Kenneth De Jong, et al. 2012. Genetic programming needs better benchmarks. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*. ACM, New York, NY, 791–798.

[26] David W Opitz and Richard Maclin. 1999. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research* 11 (1999), 169–198.

[27] Ludo Pagie and Paulien Hogeweg. 1997. Evolutionary consequences of coevolving targets. *Evolutionary Computation* 5, 4 (1997), 401–418.

[28] Ioannis Partalas, Grigorios Tsoumakas, Evaggelos V Hatzikos, and Ioannis Vlahavas. 2008. Greedy regression ensemble selection: Theory and an application to water quality prediction. *Information Sciences* 178, 20 (2008), 3867–3879.

[29] Dhiraj K Pradhan and Prith Banerjee. 1996. Fault-tolerant multiprocessor and distributed systems: Principles. In *Fault-Tolerant Computer System Design*. Prentice-Hall, Inc., Upper Saddle River, NJ, 135–235.

[30] Michael Schmidt and Hod Lipson. 2009. Distilling free-form natural laws from experimental data. *Science* 324, 5923 (2009), 81–85.

[31] Tomas Smetka, Ivan Homoliak, and Petr Hanacek. 2016. On the application of symbolic regression and genetic programming for cryptanalysis of symmetric encryption algorithm. In *International Carnahan Conference on Security Technology*. IEEE, New York, NY, 1–8.

[32] Nguyen Quang Uy, Nguyen Xuan Hoai, Michael O'Neill, Robert I McKay, and Edgar Galván-López. 2011. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines* 12, 2 (2011), 91–119.

[33] Kalyan Veeramachaneni, Ignacio Arnaldo, Owen Derby, and Una-May O'Reilly. 2015. FlexGP. *Journal of Grid Computing* 13, 3 (2015), 391–407.

[34] Kalyan Veeramachaneni, Owen Derby, Dylan Sherry, and Una-May O'Reilly. 2013. Learning regression ensembles with genetic programming at scale. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*. ACM, New York, NY, 1117–1124.

[35] Kalyan Veeramachaneni, Katya Vladislavleva, Matt Burland, Jason Parcon, and Una-May O'Reilly. 2010. Evolutionary optimization of flavors. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. ACM, New York, NY, 1291–1298.

[36] Ekaterina J Vladislavleva, Guido F Smits, and Dick Den Hertog. 2009. Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. *IEEE Transactions on Evolutionary Computation* 13, 2 (2009), 333–349.

[37] Matt P Wand and M Chris Jones. 1995. *Kernel smoothing*. Chapman & Hall, London.

[38] Yuhong Yang. 2003. Regression with multiple candidate models: selecting or mixing? *Statistica Sinica* 13, 3 (2003), 783–809.

[39] Zhi-Hua Zhou. 2012. *Ensemble methods: foundations and algorithms*. Chapman & Hall, London.