Optimally weighted ensembles of surrogate models for sequential parameter optimization

Echtenbruck, M.M.

Chapter 5

Automated Model Selection in SPO

In Chapter 4 a method was developed that automatically builds an ensemble from a large set of heterogeneous models by computing an optimally weighted convex linear combination. It was shown that regarding regression tasks, none of the possible ensembles could perform worse than the weakest of the available base models. Moreover, it is possible that an ensemble combination exists that performs even better, in terms of RMSE, when fitting the regarded data, than any of the base models. Additionally, a method was supplied to automatically determine the best performing model from the possible ensemble combinations. In this chapter, the CCM building method is adapted for the use in SPO. Since the overarching goal of this thesis is to release the user from the burden to select the right surrogate model, also and especially in time-consuming optimization tasks, the driving questions are:

- Can the CCM building method adapted such that it works reliably and accurately in SPO?
- Can optimized (and dynamically adapted) CCM compete with fixed base models in SPO?
- How does the CCM approach compete with approaches that dynamically update the surrogate model selection or apply the same ensemble throughout the entire optimization process?
- Is it possible to adapt the method such that it performs feasible, in terms

of calculation time, despite a large number of models involved?

The CCM already showed good results on regression tasks. A central focus of this chapter lies on the characteristics of the SPO that come with the sequential step. Some of these characteristics may be taken advantage of to improve the algorithm further. Other characteristics make it harder for the CCM to perform reliably. Possible solutions to approach these difficulties are introduced, analyzed and incorporated into the ensemble building method. Additionally, the algorithm is further adjusted such that it functions reliably even if one or more models of the set fails during the optimization process.

This chapter is structured as follows. In Section 5.1 the CCM method is introduced in detail, and further adaptations needed for the application in SPO are discussed and implemented. In Section 5.2 the dynamically adapted CCM method is then thoroughly tested for its performance during sequential parameter optimization on a large set of diverse objective functions. Also, it is compared to the base models and two strong ensemble competitors. Finally, the results of the experiments are presented and discussed.

5.1 Adapting to the Sequential Step

The experiments carried out in Chapter 4 were all regression tasks of a static nature. A given objective function had to be fitted via a given set of points generated by a space-filling design. For the evaluation of the fit of the model to the function, the RMSE was calculated. For a reliable result, the experiment has been repeated several times without any changes in the experiment setup besides the positioning of the points of the design that are to a given extent chosen randomly.

With the step from this static experiment to a sequential experimental setup, some alterations of the method have to be done to make it work and to ensure it functions efficiently and reliably.

The most noticeable change that has to be addressed is the fact that in each sequential step only one set of data is available for evaluating the models and finding the best ensemble combination. So instead of running repetitions, the evaluation is now done in a single 10-fold cross-validation step. The single fitness value resulting from this evaluation is crucial for the choice of the best ensemble.

The main steps of our proposed CCM building method are presented in Algo-

rithm 2 which, up to this degree of detail, is self-explanatory. The approach takes a set of n points that have been evaluated with an expensive black box evaluation function; they are denoted with $(x_1, y(x_1)), \ldots, (x_n, y(x_n))$. Then, it minimizes the cross-validation error over the set of CCM, thereby performing a parameter optimization of the model weights over the simplex $\{\alpha \in \mathbb{R} | \sum_{i=1}^{n} \alpha_i = 1, \alpha_i \geq 0\}$.

Algorithm 2: CCM Ensemble-building using weighted 10-fold cross-validation

Data: Previously evaluated n data-points $(x_1, y(x_1)), \ldots, (x_n, y(x_n))$;

Result: CCM function $\hat{y} : \mathbb{R}^d \to \mathbb{R}$

1: **begin**

2: For all base models, compute cross-validation on previously evaluated n data-points;

3: Search for best weights using box-constrained optimization and data generated in Step 2;

4: Generate ensemble-predictor function $\hat{y}$ using best weights;

To apply these models in sequential optimization requires several adaptations, e.g., to adjust the models continuously in the presence of dynamically changing and non-uniform data sets. Several adaptations will be applied in order to integrate the CCM approach in sequential parameter optimization. These are:

- Periodically rebuilding of models and temporarily suspending models, to take dynamical updates into account.
- Local density weighted cross-validation, to deal with non-uniform point distributions.
- Adaptation of (1+1)-ES used for weight optimization, to deal with large ensemble sets.

Next, these adaptations will be introduced one-by-one before in the next section further experimental justification of their usefulness is added.

5.1.1 Building Intervals and Suspension of Models

The characteristic of the SPO that is of highest interest for the adaptation of the presented approach is the steadily expanding dataset. With each step of the

optimization additional points are added to the set of known data points D. With the growth of the set of known points also the knowledge about the underlying objective function grows. Characteristics about the function that can be read from the known points may be changing over time, and the ensemble should adapt to that change. Therefore, it should be beneficial to update the weight combination over time to adopt the model to features of the objective function that were not known before.
An additional parameter τ is introduced to the method specifying a fixed number of steps, such that the proposed CCM building method updates the ensemble combination in every τ-th step of the optimization.
This parameter controls the ability of the ensemble to adjust itself, in terms of giving more weight to a more appropriate model during optimization. We suppose this ability, in general, to be beneficial for the performance of the ensemble but the choice of the parameter has to be taken carefully. While choosing too large a value for the parameter reduces the ensembles ability to adapt to changes, it is not given that an adaptation of the ensemble in every step of the optimization is still beneficial. Also, the ensemble building process is rather expensive in terms of computation time. Though in real-world optimization the objective function is the expensive part, a reasonable calculation time for the model may still be desired and thus also might be considered when specifying the rebuilding interval.

The computation time of an SPO process when using a CCM not only depends on the frequency with that the ensemble is rebuilt but also on the number of models that are part of the set. While a large set of heterogeneous models is in general desirable, the evaluation of a large set of base models also takes its time. Moreover, not all models might make a beneficial contribution to the ensemble at any time of the optimization. We expect only a smaller subset of the models to be a beneficial contribution to the ensemble, and with the set of known data steadily growing, this choice of beneficial models may steadily shift. Based on this assumption the method is adapted to allow for temporary suspension of models that do not contribute to the ensemble. A possible suspension of a base model is checked after every CCM building process, whereas re-inclusion of these models is done every λ-th step of the optimization.

It is obvious that the model building interval τ has a large influence on the computation time since the majority of the time is needed for the cross-validation. However, the influence of the suspension interval λ on the calculation time may vary since it depends on the number of models that are suspended.

These considerations embed the actual CCM building part described in Algorithm 2. In case of the actual step being a λ's step, all suspended models are

added to the set again. The CCM building is then started if this is a τ's step of the optimization and at least two models are not suspended. If all but one model are suspended, this model is the new CCM response.
Potential suspension of models is considered only after completion of the CCM building process. Models that do not contribute to the ensemble, and therefore gained no weight in the CCM building process, are suspended.

5.1.2 Local Density Weighted Cross-Validation

The main step of the CCM building is the cross-evaluation that is carried out on the previously evaluated n data-points to evaluate the fit of the base models (cf. Algorithm 2, Line 2). This step runs the risk to be the most time-consuming step of the CCM building method, but it is also a crucial step of the Algorithm since the whole CCM building process depends on the evaluation of the models that is done in this step.

Methods to gain control over the computation time were already introduced in Section 5.1.1. However, additional precautions to ensure a reliable performance are also introduced in this step. Since the overall goal is to allow for the incorporation of a large set of heterogeneous models with characteristics that may not be known to the user, these models may also show unreasonable calculation times or do not perform reliably. To encounter such problems models may be excluded from the set during the optimization process. Models that exceed a predefined time limit, fail or return defective predictions (i.e., NAN[1] values) during the fitting process are immediately excluded from the system.

However, the more critical aspect is the appropriate evaluation and weighting of the models since also the performance of the CCM depends on it. For the experiments carried out in Chapter 4 the use of the RMSE lead to good results. Nonetheless, it has to be considered that for these experiments the experimental setup was chosen such, that it enables the best circumstances for evaluating the fit of the models to the objective function consistently over the entire region of interest by choosing a space filling design for the automatic generation of the experimental data.

In general, SPO starts the optimization process by evaluating an initial set of points which are derived from a space filling design. However, it is expected

[1] NAN is the abbreviation for 'Not a Number' which is, in general, return when division by zero is attempted.

that during the progress of an SPO evaluated points will cluster at local optima. Without taking this into account during cross-validation, we risk putting too much emphasis on prediction errors close to those local optima, which leads to over-fitting in these areas.

A possible solution to this problem would be to exclude points from the data set before evaluation, to ensure an even distribution of the data for cross-validation. However, for this approach, it would have to be specified how many and which points have to be excluded from the data set for cross-validation also risking to exclude important data from the set.
Instead, to overcome the problem, for the evaluation of the overall fit of the model, we propose to reduce the importance of points that are located in areas with a very dense neighborhood by weighting their squared errors. Weights are depending on the density of the close neighborhood, and therefore on the position of the regarded point. Hence weighting occurs evenly and without harsh steps in weights between points that are close to each other since they also share parts of the same neighborhood.

The resulting weighted Root Mean Square Error (wRMSE), which is implemented as quality indicator, is calculated as follows:

$$wRMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\beta_i(y_i - \hat{y}_i)^2}$$

The weights $\beta_i \in [0,1]$ that are applied to the prediction errors $(y_i - \hat{y}_i)$ at the points $\mathbf{x}_i \in \mathbb{R}^d,\ \ i = 1,\ldots,n$, are derived from the density of the point's direct neighborhood. For the calculation of this density, the k closest neighbors of each point are utilized.

The density ρ_i of a point i is then calculated as the median distance to these neighbours:

$$\rho_i = \text{median}\left\{ \sqrt{\sum_{j=1}^{d}(\mathbf{x}_i^j - \mathbf{x}_l^j)^2} \mid l = 1,\ldots,k \right\}$$

Since only points are to be weighted that are located in areas with a dense neighborhood, density values that are exceeding the overall mean density are truncated to the mean value. This is done to ensure that all points with a neighbourhood of mean density, or sparser, get full weights in the cross-validation.

In order to obtain the weights $\beta_i \in [0,1]$, the determined density values ρ_i are normalized to the $[0,1]$ range, by computing $\beta_i = \rho_i / max\{\rho_0,\ldots,\rho_n\}$. The lower

bound has not to be taken in account here, since the density values ρ_j have to be positive per definition because of the distances used for calculation and it is not intended to force zero weight on points with the highest density.

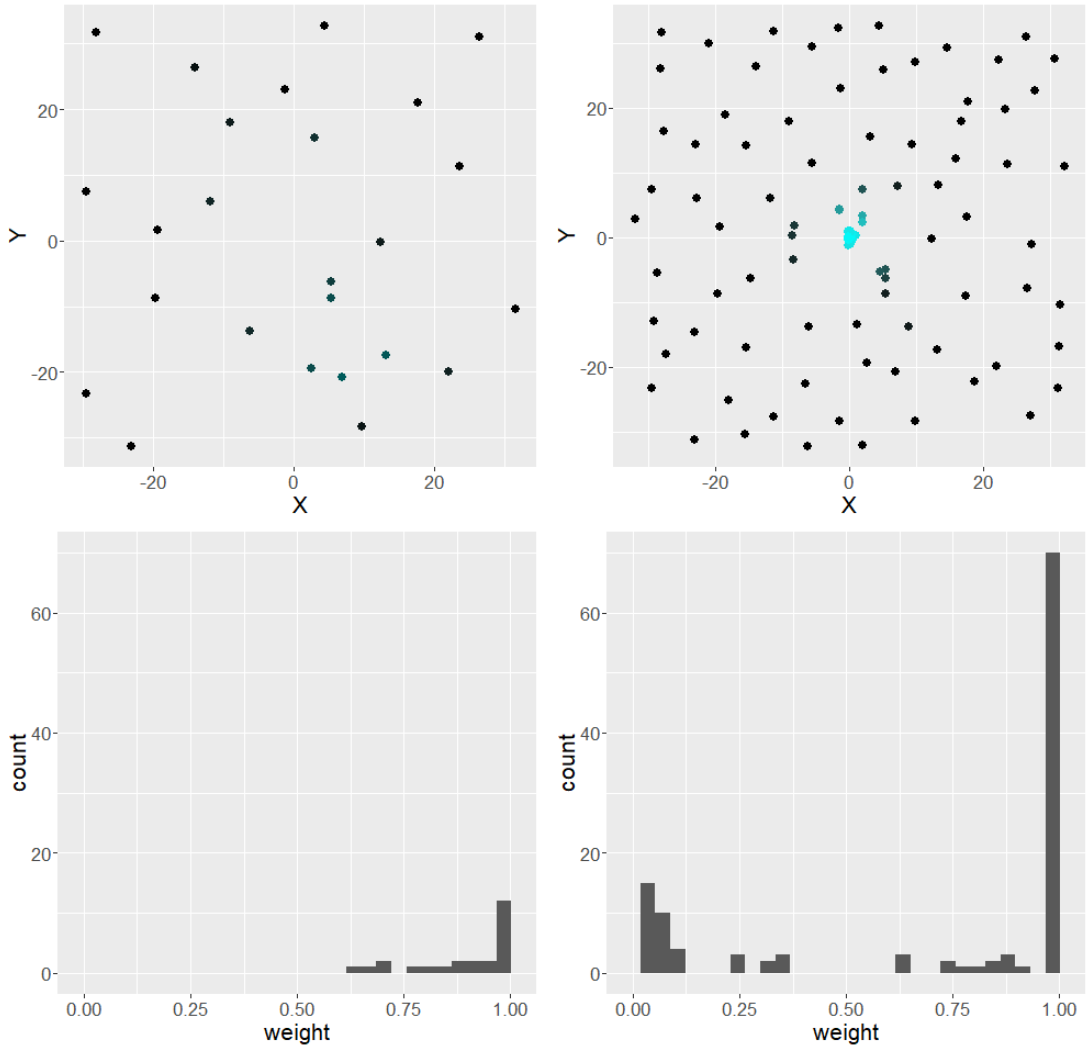

Figure 5.1: The plots show the impact of the weighting procedure on the points at the beginning and the end of an optimization process on a 2D Ackley function. Points that are colored black are fully taken into account, and lighter blue points are weighted. The lower row shows the related distribution of weights used. It can be seen that points near the cluster are rigorously weighted.

Applying these weights to the squared errors of the RMSE during cross-validation ensures that all points with a neighborhood not denser than the mean-density are

considered with full weight and only points located in denser neighborhoods are weighted according to the density of their neighborhoods.

At the beginning of the SPO, this approach has only a small impact on the weighting of the predictions during cross-validation since initial designs of experiments preferably are space-filling and as such avoid clustering of points. Only with the optimization process proceeding points will eventually cluster at local optima. The denser this clustering will be the higher gets the impact of the weighting during cross-validation on points located in or near clusters.

Figure 5.1 shows the effect of the weights applied to data points during an sequential optimization on a two dimensional Ackley function. The plots belong to two different steps of the same optimization process. In the left column, the situation after the first sequential steps of the optimization is shown, only five additional points were evaluated. In the right column, the situation after the optimization process has stopped is shown.
In the upper row, the points are depicted; the color indicates the applied weights. Here a point colored black means no weighting or a full weight of one respectively, while a blue colored point means that this point was weighted. In the lower row, the distribution of the weights is shown.
It can be seen that in the early steps of the SPO the weighting has nearly no impact, some points are only slightly weighted while the majority of the points get full weight. The histogram in the lower row also confirms this.
In the last step of the SPO altogether 120 points were evaluated and clustering has taken place in the center of the regarded area. The points that are densely clustered also are severely weighted, while points that are not located near a cluster get full recognition for the evaluation. Again, this can be read from the histogram, which shows that the majority of the points gets full weight, but a smaller subset of the points is rigorously weighted.

5.1.3 Optimization of the Ensemble Weights

After finishing the cross-validation of the base models, we use a (1+1)-Evolutionary Strategy (ES) with 1/5th success probability rule for step size adaptation to find the best ensemble weights (cf. Algorithm 2, Line 3).

In Section 4.4 several adaptations of the algorithm were proposed to ensure that the method functions properly on large sets of base models. From these adaptations, the additional approaches were at an advantage in some cases. The experiments also proved that a model can make a beneficial contribution to an

ensemble although a better-ranked model is not able to do. Therefore, in the experiments carried out in this chapter the adaptation introduced as 'Additive ES without stop on stagnation' (cf. Section 4.4.6) is used. Algorithm 3 depicts the main steps of the modified $(1 + 1)$-ES method which are introduced in more detail in the following.

Algorithm 3: Adapted (1+1)-evolution strategy (ES) with 1/5th success probability rule for step size adaptation and

Data: Initial population $P \subset [0, 1]^s$
Available models $\mathbf{av} = (av_1, \cdots, av_s) \in \{0, 1\}$
Result: Complete population P

```
1: begin
2:     Choose the best individual as the first parent individual ;
3:     Specify initial search space (active models) act = (act_1, ..., act_s) ∈ {0,1};
4:     Initialize ES max step count C_max ← 5|act|^2 ;
5:     Initialize ES step size adaption interval φ ← 5|act| ;
6:     Initialize ES initial step size σ ← σ_init;
7:     while Stop criteria not met do
8:         if Steps for this level exhausted then
9:             Adjust search space;
10:            Reset ES step size σ ← σ_init ;
11:            Adjust ES max step count C_max ← C_max + 5s^2 ;
12:        Generate offspring by perturbation of all active α_i with a normal
           distribution with standard deviation σ (step size);
13:        Evaluate offspring;
14:        Select new parent individual by choosing the offspring or the current
           parent depending on the objective function value;
15:        if Step count is multiple of φ then
16:            Adjust step size σ using 1/5th success rule;
```

The search starts with an initial population P, e.g., the corners of the search space, representing the active base models and a vector $\mathbf{av} \in \{0, 1\}^s$ of flags, where s corresponds to the number of base models in the set, stating which models are currently suspended ($\mathbf{av}_i = 0$) and which models are to be used for

this optimization process ($\mathbf{av}_i = 1$), and thus defining the search space. For each base model, all predictions made during cross-validation, as well as each model's wRMSE value is known. From these data, any CCM given by a specific weights combination can be derived.

Due to the incremental data update that is characteristic for sequential optimization, the position of the optimal weight combination in the current iteration is likely to not deviate much from the one obtained in the previous optimization. Therefore, the previously obtained solution is also added to the initial population P.

The individual that is used as a starting point for the search is chosen by its fitness value (Algorithm 3, Line 2). However, the search favors a combination of models over a single model only if its overall fit in terms of wRMSE is strictly better. Therefore, the ensemble from the previous optimization step is also chosen only if strictly better.

To enable the search to handle large sets of base models, the search starts on a smaller subset of the available models (Algorithm 3, Line 3) and then stage-wise extends or adjusts the search space throughout the search. Which models are currently active is specified in an additional vector $\mathbf{act} \in \{0,1\}^s$ of flags.
For the initial subset, the base models that are part of the individual chosen as the starting point are automatically added. If needed, additional base models are chosen by their fitness values to ensure, that at least three base models are part of the initial search space.

Then, the remaining parameters for the ES are initialized (Algorithm 3, Line 4-6). Finally, the main optimization loop is started. This loop is terminated solely when all available models ($\mathbf{av}_i = 1$) were part of the search space, at least for the length of the stage that they were added in. As mentioned before, the actual search is performed stagewise, doing restarts with different subsets of models. With each stage the search space is adjusted, the parameters of the ES are reset and if needed adjusted to the new search space. For this, the last model added to the search space is removed again if its addition did not lead to a better solution. However, the initial three models are never removed from the search space. Then the best performing model that has not yet been part of the search space is added. The number of search steps granted for this stage depends on the number of models that are part of the search space at this stage ($|\mathbf{act}|$). (cf. Algorithm 3, Lines 9-11).

One offspring is then generated per mutation from the best individual of the

actual population P (cf. Algorithm 3, Line 12). This is done by a random mutation of the active models' weights. We assume that the benefit of a model with a contribution to the ensemble of less than two percent is negligible. Thus the mutation step of the ES is adopted such that single weights are at least two percent or zero. This correction is done randomly, so that also when the step size of the search algorithm is rather small, the weight has a chance to surpass this barrier.

Next the offspring is evaluated for its fitness (cf. Algorithm 3, Line 13). For this the prediction of the model is to be evaluated, that is a mixture of the predictions of the base models. These base model predictions have been previously determined (cf. Algorithm 2, Line 2) so that the mixture now can be easily calculated. The offspring individual is chosen as the new parent individual if its fitness value is better than the parent's fitness value.

The search is finished when all stages are completed, which means, that all models that are currently available were part of the search space at least for the duration of one complete stage.

5.1.4 Building the Ensemble Function

With the completion of the search, the best weights combination for the ensemble is known. Models that do not contribute to the ensemble ($\beta_i = 0$) are suspended for the remainder of the actual suspension interval ($\mathbf{av}_i := 0$). All models that contribute to the ensemble are fitted to the complete data set. With the weights and the fitted base models an ensemble function, that represents the CCM, is built and returned (cf. Algorithm 2, Line 4).

5.2 Sequential Optimization Using Dynamic Ensembles

The necessary adaptations to apply the CCMs, developed in Chapter 4, to sequential optimizations were made in Section 5.1.

In this section, the dynamically adapted CCM method is thoroughly tested to obtain further experimental justification of the usefulness of these adaptations.

Furthermore, the questions posed at the introduction of this chapter are to be addressed as follows:

- To investigate the performance and reliability of the proposed dynamically adapted CCM method, it is tested in sequential optimization processes on a large set of diverse objective functions.
- The performance of the CCM is compared to the performance of the base models to evaluate if the CCM can still compete with the base models in a sequential optimization process.
- Additionally, the performance of the CCM method is compared to the performance of two strong ensemble approaches that both hold only some of the features of the CCM method.
- Finally, the CCM method is analyzed for its performance in terms of calculation time.

On each of the functions, several independent complete optimization processes are carried out using the SPO Framework as introduced in Section 2.2. The experimental setup for these experiments and the functions used is given in Section 5.2.1 and the general setup for the CCM is presented in Section 5.2.2. In Section 5.2.3, the competitors that the CCM is also compared to is introduced. Experiments using different settings for the rebuild interval τ and the suspension interval λ are carried out in Section 5.2.4. The results are discussed with a special focus on the impact of the settings of these intervals on the performance and the computation time of the CCM. Finally, the performances of two CCMs using different settings for τ and λ are compared to the performances of the base models and the two competitors presented in Section 5.2.3.

5.2.1 Experimental Setup and Objective Functions

To obtain meaningful insights about the performance of the proposed dynamically adapted CCM method it is aimed for a set of functions as diverse as possible. Therefore, a set of 10 objective functions is chosen, all showing different characteristics. Part of the set are two Gaussian landscape generator functions (GLG4D, GLG8D), four instances of two classical test problems for optimization algorithms (Ackley2D, Ackley4D, Rosenbrock4D, Rosenbrock8D), and four test functions based on physical models (piston function, robot arm function, otl-circuit function, wing weight function) (cf. Section 2.3). The Gaussian landscape generator functions are instantiated with 80 Gaussian process realizations

for the four-dimensional GLG function and 320 Gaussian process realizations for the eight-dimensional GLG function respectively.

Function	Dimension	Initial Design Size	Number Sequential Steps
GLG4D	4	60	100
GLG8D	8	100	220
Ackley2D	2	20	100
Ackley4D	4	60	100
Rosenbrock4D	4	60	100
Rosenbrock8D	8	160	100
Otl-circuit function	6	30	50
Piston function	7	110	50
Robot function	8	110	50
Wing weight function	10	280	100

Table 5.1: Settings used for the experimental setups per function.

Table 5.1 gives an overview of the different experimental setups. The initial design size used for each function is determined in a preliminary experiment and depends on the difficulty of the function. As before, the defining criterion is the share of base models that perform better than the mean predictor.
For all functions, distinct initial designs are generated in advance to ensure that all experiments have the same precondition. For the GLG functions as well as for the classical optimization problems 20 repetitions of the experiment are performed, for the functions based on physical models, ten repetitions are carried out. Each of the repetitions starts from one of the predefined initial designs.

The initial designs for the Ackley function as well as for the otl-circuit function contain a smaller number of points since previous experiments showed that the optimum is reached quite early with few points already. However, for the wing weight function, the initial design contains 280 data points since the function is rather hard.

5.2.2 Setup of the CCM Building Method

The CCM building algorithm as described in Section 5.1 has a few settings which have to be considered. Some of these settings are easy to be set to reasonable values; some of them allow for several options to be negotiable. Table 5.2 gives an overview of all of these settings and the values chosen for the experiments.

Description	Variable	Value
Ensemble building interval	τ	{1, 5, 10, 20}
Base model suspension interval	λ	{1, 5, 10, 20}
Exclusion time limit	-	300s
Number of neighbours considered for density calculation	k	20
Model-accept weight		2%

Table 5.2: Settings of the CCM building method used in the experiments

The variables τ and λ are crucial for the CCM building method. As stated before (cf. Section 5.1) these values have a significant influence on the overall computation time of the CCM method but may also influence the performance of the CCM. With too large values chosen for τ and λ, the CCM may not be able to adapt fast enough to changes in the underlying data while too small values will unnecessarily increase the computation time. The experiments presented consider a range of different settings for these values.

The algorithm allows setting a fixed time limit for fitting a single predictor during cross-validation. Models that exceed the preset time limit are excluded from the system. This ensures reasonable calculation times by enabling the algorithm to exclude models that need unreasonable long calculation times. However, this parameter should be chosen carefully since this exclusion is final; models are not reincluded after the expiration of the suspension interval. Also, it has to be taken into account that computation time is no quality indicator; sometimes longer calculation times might be preferable. In our experiments, the exclusion time limit is set to 300 seconds. This time limit is considered as safety switch only, in case that a model takes extraordinary much time for fitting and is not expected to be reached. With this setting, the use of entirely unknown models may be encouraged.

The number of nearest neighbors that are considered for the density calculation of the point's neighborhood is set to 20 points. This is assumed to be large enough

to obtain a reliable value and yet not too large so that the calculation focuses on the closer neighborhood.

The model-accept weight specifies the minimum weight that a model should have to be accepted as part of the model. In the experiments presented here this value is set to 2% since we do not intend to restrict the algorithm more than needed but consider a weight of less than 2% as negligible.

Description	Variable	Value
Designated search steps per stage	-	$10 \cdot s_{act}^2$
Maximum search steps without improvement	-	$\frac{2}{3}(10 \cdot s_{act}^2)$
Interval for $\frac{1}{5}$-success rate check	-	$5 \cdot s_{act}$
Initial step size	σ_{init}	0.4
Minimum step size	σ_{min}	0.1
Learning rate (step size variation factor)	η	0.9

Table 5.3: Settings of the (1+1)-ES used for searching the best weights

Additional parameters that are used for the (1+1)-ES are specified in Table 5.3. The learning rate η is chosen in the recommended range $[0.817, 1)$, but slightly higher than the recommended value of 0.817 to improve exploration. Also, the other settings follow the recommended settings in Bäck et al. [110]. As mentioned before (cf. Section 5.1) the search algorithm is adapted stage wise. Some of the parameters specified here are reset or adjusted with the beginning of every stage. The designated number of search steps per stage is one of these variables. It also depends on the number of base models that are part of the search space in the considered stage of search (s_{act}). However, the search on one stage may be finished earlier if no progress is made. In the experiments presented here a stage is been terminated when no progress is made in 2/3 of the total number of search steps allowed for this stage. If progress is made the model is supposed to be valuable to the ensemble and is granted the designated amount of search steps.

In a fixed interval of search steps, the success rate of the search is calculated, and the search step width is adapted accordingly. The length of this interval also depends on the number of models that are part of the search space at that time (s_{act}). The step width of the ES is reset to its initial value with the start of each stage.

5.2.3 Competitors

To investigate the performance of the proposed dynamically adapted CCM method its performance is compared to the performance of the base models as well as to two ensemble-like approaches. These approaches will hereafter be referred to as 'Initial' and 'Choose'.

'Initial' builds the CCM only in the first iteration of SPO ($\tau = \infty$). In case of failure of one base models during the prediction, this model is excluded from the ensemble. For this prediction, the weights of the remaining models are adjusted to keep their relation and fulfill the requirement to sum up to one, despite the missing model. Additionally, the malfunctioning model is directly and finally excluded from the set of model choices. The CCM building process is then newly started in the next step of the SPO. By using this method, we want to get some insights into the benefits of adjusting the ensemble during the optimization process.

'Choose' selects a single base model in every λ-th iteration of the SPO. This method also uses the weighted 10-fold cross-validation like the CCM building method, since in preliminary experiments it showed better performance using weighted cross-validation than by using the standard cross-validation. However, in contrast to the CCM building method Choose is restricted to choosing a single best model only. Thereby we assess the benefits of using mixtures of models instead of selecting and updating only single base models.

5.2.4 The Impact of Rebuild- and Suspension Intervals

Before turning to the main experiments, some thought should be given on the settings of the values τ and λ. As stated before, these values have a crucial influence on the computation time of the CCM building method as well as on its prediction performance during sequential optimization. To get some insights into the impact of these values on the quality of the optimization result, experiments are run with all reasonable combinations for $\tau \in \{1, 5, 10, 20\}$ and $\lambda \in \{1, 5, 10, 20\}$. Settings, where the model suspension interval λ is shorter than the model rebuild interval τ, are not considered as reasonable since the reactivation of temporarily suspended models only comes into account during the next model rebuilding process. Experiments are run on the complete set of objective functions as presented in Section 5.2.1.

Table 5.4 gives an overview of the results of these experiments. The table is ar-

$\tau = 1$	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 20$
ackley2D	4.736 (±1.8360)	3.082 (±2.1987)	2.421 (±2.2847)	**1.954** (±2.0943)
ackley4D	8.453 (±2.6557)	7.728 (±3.4242)	6.932 (±3.1568)	6.163 (±3.5904)
GLG4D	21.96 (±11.6650)	21.92 (±11.9038)	24.21 (±11.3581)	23.77 (±11.4260)
GLG8D	30.45 (±8.9146)	28.11 (±11.1735)	**26.05** (±9.9418)	26.67 (±11.3850)
otl-circuit	2.621 (±0.050386)	**2.605** (±0.002278)	**2.605** (±0.001937)	**2.605** (±0.001924)
piston	**0.1647** (±0.000499)	0.1655 (±0.002251)	0.1676 (±0.004043)	0.1679 (±0.004456)
robot	**0.01078** (±0.014679)	0.01592 (±0.020619)	0.01768 (±0.027308)	0.0227 (±0.026355)
rosenbrock4D	2.673 (±1.599095)	**2.236** (±1.728283)	2.412 (±1.649369)	2.988 (±1.545986)
rosenbrock8D	4075 (±2482.58)	3684 (±2353.88)	**3119** (±1606.20)	4302 (±3218.83)
wingweight	177.7 (±12.7954)	179.6 (±6.154)	176.2 (±13.9320)	178.2 (±16.4675)
Md RankSums	56	46	**27.5**	45
Mn RankSums	60	43	**33**	46
Md Rank	5	3	**1**	2
Mn Rank	6	2	**1**	3
$\tau = 5$		$\lambda = 5$	$\lambda = 10$	$\lambda = 20$
ackley2D		3.68 (±2.1033)	2.936 (±2.1041)	2.427 (±1.7135)
ackley4D		7.685 (±2.4899)	7.122 (±2.6919)	6.323 (±3.5570)
GLG4D		20.58 (±12.8548)	24.01 (±11.0480)	20.66 (±14.1705)
GLG8D		30.21 (±8.2603)	30.36 (±10.3108)	29.78 (±9.2280)
otl-circuit		2.645 (±0.066244)	2.606 (±0.003172)	2.611 (±0.017736)
piston		0.1692 (±0.005782)	0.1684 (±0.004193)	0.1709 (±0.007741)
robot		0.01772 (±0.027462)	0.02818 (±0.0283594)	0.02828 (±0.024504)
rosenbrock4D		2.452 (±1.600449)	18.95 (±72.166486)	7.898 (±21.774267)
rosenbrock8D		3812 (±1473.23)	3939 (±1807.51)	3895 (±1645.08)
wingweight		**173** (±9.1062)	176.3 (±13.1074)	178.9 (±9.8847)
Md RankSums		60	72	62
Mn RankSums		54	65	63
Md Rank		6	10	7
Mn Rank		5	9	7
$\tau = 10$			$\lambda = 10$	$\lambda = 20$
ackley2D			3.268 (±2.0659)	2.641 (±2.0901)
ackley4D			7.452 (±3.0298)	6.386 (±3.2157)
GLG4D			26.26 (±7.4316)	25.68 (±9.8755)
GLG8D			30.7 (±12.4646)	28.56 (±12.0462)
otl-circuit			2.632 (±0.056935)	2.624 (±0.052627)
piston			0.1672 (±0.003480)	0.1694 (±0.008199)
robot			0.0191 (±0.021059)	0.02285 (±0.020645)
rosenbrock4D			2.66 (±1.941899)	3.005 (±2.081024)
rosenbrock8D			4429 (±2114.36)	4155 (±1879.44)
wingweight			178.8 (±11.0519)	175.9 (±18.6308)
Md RankSums			68	65
Mn RankSums			74	64.5
Md Rank			9	8
Mn Rank			10	8
$\tau = 20$				$\lambda = 20$
ackley2D				2.323 (±2.2779)
ackley4D				**6.09** (±2.9370)
GLG4D				**18.27** (±12.5356)
GLG8D				27.31 (±11.0315)
otl-circuit				2.61 (±0.012475)
piston				0.1693 (±0.006238)
robot				0.0211 (±0.020513)
rosenbrock4D				25.45 (±96.6529205)
rosenbrock8D				4348 (±3245.30)
wingweight				175.9 (±14.9929)
Md RankSums				48.5
Mn RankSums				47.5
Md Rank				4
Mn Rank				4

Table 5.4: Experiment results for the comparison of different settings for τ and λ. Given is the mean and standard deviation of the optimization results of the ensemble for each reasonable combination of τ and λ. The two lowest rows give the ranking results of these runs in comparison. Best results are marked bold.

ranged in four major rows; each row gives the results for one value of the model rebuild interval τ while each column represents one setting for the model suspension interval λ. Each entry names the mean optimization result with standard deviation that has been achieved during the optimization processes using the corresponding CCM. The best values that were achieved on each function are marked bold. To allow for an evaluation of the methods over the set of functions with so strongly differing features function wise rankings are used. Although the result table only names the mean optimization results the median optimization results also have been ranked. The sums of these ranks are shown in the rows 'Md RankSums' and 'Mn Ranksums' respectively. The last two rows of each major row 'Md Rank' and 'Mn Rank' only shows the rankings of 'Md RankSums' and 'Mn Ranksums' respectively.

It can be seen, that the CCM using $\tau = 1$ and $\lambda = 10$ is ranking first place for mean optimization value as well as for median. Looking at the RankSums one can say that the result is not even tight. Furthermore, the three best-performing settings can be found in the first major row ($\tau = 1$). Also noteworthy is the fact, that the model which is ranked fourth, in both mean and median optimization result, is the CCM using $\tau = 20$ and $\lambda = 20$.

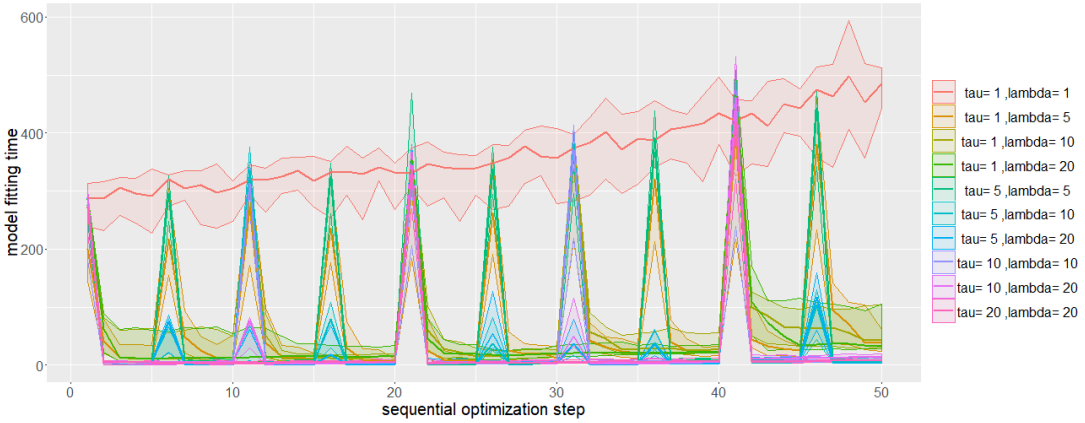

Figure 5.2: Average model fitting times during sequential optimization for the different settings of τ and λ per sequential step. The solid line marks the mean time while the opaque area shows the standard deviation of the calculation time. Most time-consuming is the approach using $\tau = \lambda = 1$ since it is rebuilding the ensemble on the complete model set in every step. The least time consuming is the one using $\tau = \lambda = 20$, which shows a peak in calculation time only in every 20th step.

However, as mentioned before, these settings do not only have an impact on the

performance of the models but also on its calculation times. Figure 5.2 gives an insight into the behavior of the different setups.
Three different situations may be encountered during the model fitting step. These are model rebuild steps where all models are part of the search space, model rebuild steps where some models are suspended and steps where the model is not rebuilt.
As expected the steps where the ensemble has to be rebuilt using the complete set of base models are the most expensive steps, in terms of calculation time. Whereas the steps where no new ensemble is built are negligible. This behavior can be easily seen in Figure 5.2, comparing the lines for $\tau = 1$ and $\lambda = 1$ versus $\tau = 20$ and $\lambda = 20$. The calculation times per sequential step for the CCM with $\tau = 1$ and $\lambda = 1$ ranges, slowly increasing, between 200 and 600 seconds. Whereas the CCM with $\tau = 20$ and $\lambda = 20$ has calculation times close to zero in most of the sequential steps, only every 20-th sequential step the ensemble is rebuilt and the calculation time shows a peak with calculation times corresponding to the calculation times of the CCM using $\tau = 1$ and $\lambda = 1$.

The calculation time needed in such steps, where some models are suspended from the system, is strongly depending on the number of models that are not suspended. The line depicting the times for the setting using $\tau = 1$ and $\lambda = 20$ shows this. Although the model is built in every step, the calculation time is negotiable and steadily decreases step by step, as further models are suspended. In step 20 and step 40 respectively all models are reincluded to the system, which has a strong impact on the calculation time of this CCM after step 40.

Table 5.5 shows exemplary running times for the same choice of configurations of λ and τ summed up for an entire optimization process on two different functions. The configurations where λ is set to the same value as τ are not affected by the suspension since it is released in the same step as the model is rebuilt, and therefore in every model building step the full model set is available.

The results suggest that two choices are best, depending on the priorities set. For one the CCM with $\tau = 1$ and $\lambda = 10$ seems to be the best choice when calculation time has not to be taken into consideration too much. For another, the CCM with $\tau = 20$ and $\lambda = 20$ seems to be a worthwhile choice.
With a setting of $\tau = 1$ and $\lambda = 10$, the CCM is rebuilt in every step and thus can quickly adapt to changes in the underlying data. The suspension interval is large enough to reduce the calculation time remarkably and small enough to allow for an adaptation of the active models throughout the optimization process.
Using a CCM with $\tau = 20$ and $\lambda = 20$ results in an ensemble, that shows a good performance while having the best results in terms of calculation time. Though we

		λ			
		1	5	10	20
τ	1	17659 (± 4956)	4717 (± 1435)	3763 (± 720)	3068 (± 943)
	5		3765 (± 1011)	2464 (± 452)	1782 (± 422)
	10			2096 (± 801)	1772 (± 509)
	20				1421 (± 432)

Table 5.5: Average computation times (s) for a complete sequential optimization process on the GLG4D function depending on the values for τ and λ. On the diagonal, the results for such settings are shown where λ has no influence since it matches to the value of τ. In these cases, the full set of available models is used in the ensemble building process. The same applies for combinations of τ and λ where $\lambda < \tau$, therefore these fields are left blank.

aim for problem setups with high-cost objective functions where the calculation time of the model is neglectable, this still might be an interesting choice in some cases.

Recapitulating it can be said that the choice of the values for τ and λ should be well considered. The results show that these values have a heavy impact on the performance of the model in terms of prediction quality as well as computation time. A smaller value for the model rebuild interval seems to be preferable though the results show that also larger values can lead to good results (cf. Table 5.4, $\tau = 20$, $\lambda = 20$). However, a smaller value for the suspension interval must not necessarily lead to better results (cf. Table 5.4, $\tau = 1$, $\lambda = 1$). We assume that a large set of base models still makes it harder to build the best fitting model. So it might indeed be beneficial to suspend models that are not contributing to the system for some time. Of course, these values also should be chosen considering the number of sequential steps that are to be performed.

5.2.5 The Performance of Dynamical Adapted CCM in SPO

In the following, the performance of the CCM in sequential optimization processes is closer investigated. For these experiments the CCM with $\tau = 1$ and $\lambda = 10$ is chosen as well as the CCM using $\tau = 20$ and $\lambda = 20$. The ensembles are compared to the base models as well as to the competitors 'Choose' and 'Initial' as introduced in Section 5.2.3. Again, experiments are run on the complete set

of objective functions as presented in Section 5.2.1.

Table 5.6 gives an overview of the Results of the experiments. The structure of the table resembles the structure of Table 5.4 in the previous section. Main difference is the columns 'Best Base Model'. Since for the main experiment setup the chosen CCM methods are compared against 'Choose' and 'Initial' as well as all base models that are part of the set, the complete result is condensed to the relevant information to preserve the readability.

FUN	$\tau = 1, \lambda = 10$	$\tau = 20, \lambda = 20$	Choose	Initial	Best Base Model	
ackley2D	2.496 (±1.931)	1.934 (±2.041)	5.250 (±2.604)	0.869 (±1.392)	**0.343** (±0.694)	MLP
ackley4D	6.850 (±3.179)	6.022 (±3.242)	9.189 (±3.182)	**4.403** (±3.101)	5.206 (±1.074)	Lm
GLG4D	22.43 (±12.00)	23.58 (±13.02)	**14.31** (±14.47)	25.07 (±7.83)	21.10 (±13.17)	corrgauss
GLG8D	29.62 (±9.42)	30.34 (±7.17)	40.02 (±13.39)	35.31 (±10.42)	**28.70** (±11.39)	corrgauss
otl-circuit	2.605 (±0.002)	2.610 (±0.0123)	2.695 (±0.080)	2.619 (±0.046)	**2.604** (±0.0002)	Earth, MLP
piston	**0.167** (±0.003)	0.170 (±0.008)	0.172 (±0.004)	0.168 (±0.001)	0.167 (±0.001)	MLP
robot	0.013 (±0.020)	0.018 (±0.018)	0.040 (±0.031)	0.021 (±0.022)	**0** (±0)	MLP, Neuralnet
rosenbrock4D	**2.414** (±1.522)	4.198 (±2.088)	2.665 (±1.331)	160.4 (±208.2)	3.554 (±2.490)	corrgauss
rosenbrock8D	4127 (±2989)	3224 (±2042)	5755 (±4100)	3171 (±1758)	**697** (±582)	Lm
wingweight	182.5 (±13.11)	**173.0** (±10.35)	180.4 (±15.10)	174.0 (±15.65)	174.8 (±14.12)	correxp
Md RankSums	29	27.5	41	32	**20.5**	
Mn RankSums	28	31	42	31	**18**	
Md Rank	3	2	5	4	**1**	
Mn Rank	2	3	5	4	**1**	

Table 5.6: Results for the main experiment setup. Depicted are mean optimization result with standard deviation. The presentation of the results for the base models is consolidated to only depict the performance of the best model for each function.

Thus, the column 'Best Base Model' names the mean optimization result with standard deviation and the name of the corresponding base model which performed best on this function. In case that two base models showed the same performance in terms of mean optimization result, both base models are named here. For differing performances in terms of standard deviation only, the smaller standard deviation value is shown. Additionally, the complete results for the comparison of the two CCMs to all base models is given in the Appendix A.2.

Inspecting the overall rankings of the models depicted, it can not be denied that the base model is ranked best. However, it has also to be taken into account, that the best base model is changing, depending on the objective function. Given that the most appropriate base model is not known to the user, the next best choice is the CCM. We assume the better choice to be the CCM with $\tau = 1$ and $\lambda = 10$, but the differences in performance seem to be negligible, at least for these experiments.

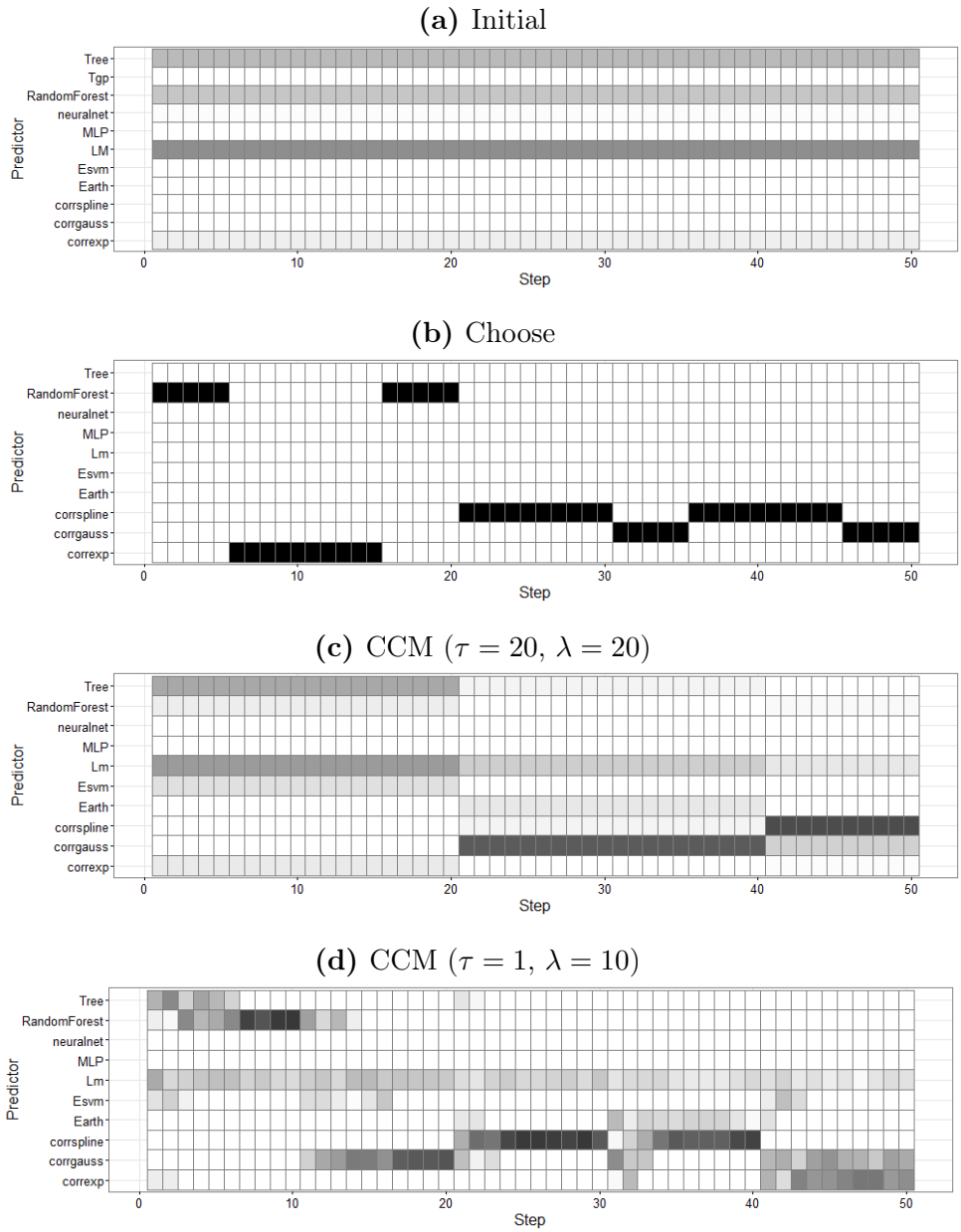

Figure 5.3: Distribution of weights during one exemplary sequential optimization process on the GLG4D function. 'Initial' is forced to use the same ensemble throughout the whole optimization process, while 'Choose' switches to the best choice in every fifth step. The ensemble ($\tau = 1, \lambda = 10$) starts with a setup that resembles the setup of 'Initial', but adjusts its weights in the next steps, often giving large parts of the weight also, but not exclusively, to the model also preferred by 'Choose'.

Figure 5.3 gives further insights into the behavior of the different types of models. As introduced in Section 5.2.3, 'Initial' builds its ensemble only in the very first step and then sticks to it. Whereas 'Choose' selects the best performing model every fifth step. Here it can be seen, that the preferred base model chosen by 'Choose' switches several times. The two CCM models start with a similar ensemble as 'Initial', but while the CCM with $\tau = 20$ and $\lambda = 20$ keeps its setup for 20 steps without changes the CCM with $\tau = 1$ and $\lambda = 10$ slightly modifies its setup in every step. This improved ability to adapt to changes may also explain the fact, that the distribution of weights resembles more to the choice of 'Choose' than the weights distribution of the CCM using $\tau = 20$ and $\lambda = 20$.

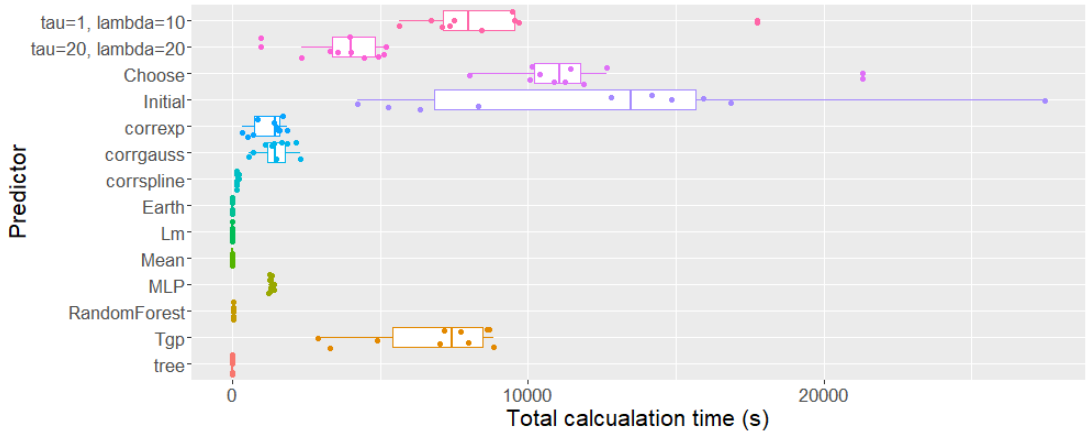

Figure 5.4: Shown is a boxplot of the calculation times that are needed by the different models during a complete sequential optimization process on the wing weight function. Noteworthy is, that on this function 'Choose' needs significantly more time than both ensembles. Also, the calculation times of 'Initial' are surprisingly long with an also rather large variance. Therefore, also 'Initial' takes more calculation time than both ensembles, in most cases.

Figures 5.4 and 5.5 give another insight into the calculation times of the different models. The times depicted here each represent the calculation times of a complete optimization process carried out on the wing weight function, the most expensive function in terms of model calculation time, and on the GLG4D function respectively. Although the calculation time on this wing weight function builds an exception, the relation of the calculation times between the different models is similar on all considered functions.

Remarkable in these results is, that the computational cost needed for the two competitors 'Choose' and 'Initial' is not necessarily less than for the two CCM

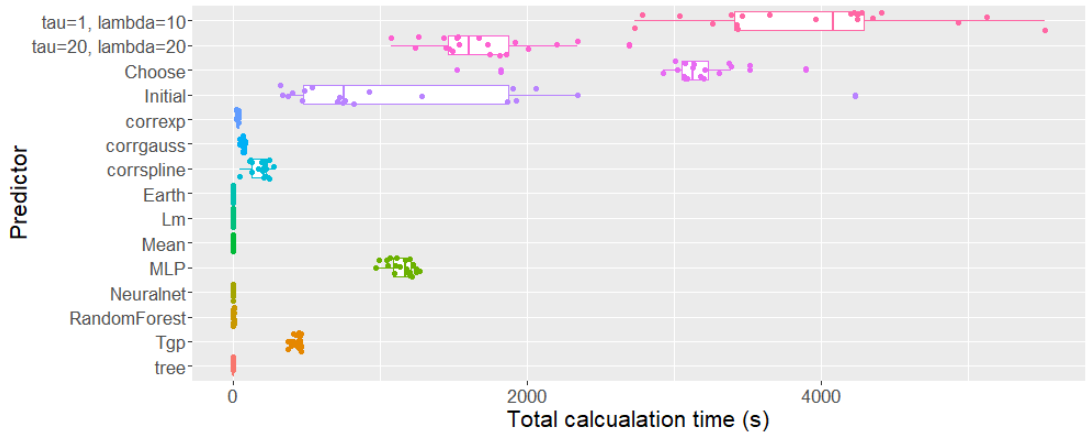

Figure 5.5: Shown is a boxplot of the calculation times that are needed by the different models during a complete sequential optimization process on the GLG4D function. Noteworthy is, that 'Choose' takes significantly more time than the CCM using $\tau = 20$ and $\lambda = 20$. 'Initial' has a large variance in its calculation times but in most cases performs faster than both ensembles.

instances. On the wing weight function, the computation times for 'Choose' are significantly longer than for both ensembles, while on the GLG4D function it still performs slower than the CCM using $\tau = 20$ and $\lambda = 20$. This may be based on the fact, that 'Choose' performs a cross-validation on the full set of base models in every fifth step whereas the ensembles use the full set of base models only every tenth and twentieth step respectively.
The calculation times for 'Initial' are surprisingly long, taking into account, that the cross-validation of the full set of base models is done only in the very first step of the SPO process. A possible explanation for this might be an unfortunate choice of base models in the first step. I.e., in most of the twenty repetitions carried out on the GLG4D function 'Initial' chose at least one, sometimes even more of the slower performing base models. However, this is only an assumption and requires further investigation.

5.3 Conclusion

The primary goal of this chapter was to adapt the CCM method developed in Chapter 4 such that it can be applied to SPO and thereby release the user from the burden to select the right surrogate model, especially in sequential optimization

tasks. As before, it was aimed for a method that in any situation can compete with the best performing base model. To achieve this, in Section 5.1 several adjustments were made to the basic CCM method to improve its accuracy, reliability and computation time in sequential optimization processes. In Section 5.2 the dynamically adapted CCM method was thoroughly tested for these virtues in different setups and compared to the base models as well as to two strong ensemble competitors.

The results showed that the dynamically adapted CCM performs reliably and, in terms of accuracy, can compete with the base models as well as the competitors. It was shown that in most cases a base model showed the best optimization result. However, in these cases, a CCM was placed second in general. Moreover, though the 'BestBaseModel' was ranked first in the overall evaluation, it has to be taken into account that the concrete best base model was differing for each function. Therefore, in comparison to the base models the CCM would be the best choice, given that the best base model is not known beforehand.
Concerning the ensemble competitors, it could be shown that the CCM method has a clear advantage over approaches like 'Choose' that only select a single best base model in fixed intervals. The difference to 'Initial', a CCM instance using $\tau = \infty$, is a bit smaller; still the CCM instances that updated their ensemble setup throughout the optimization process showed better results.
The comparison of CCM instances using different values for τ and λ gave further evidence for the advantageousness of regular updates of the ensemble setup. Though it was visible that an update on the full set of available models is neither needed nor useful, the results showed that a regular update on a reduced set of base models is beneficial.

Concerning the calculation time, the CCM method and the parameter τ and λ showed the envisioned behavior. As expected, in terms of calculation time, the CCM cannot compete to a single base model. However, the difference between the ensemble competitors and the CCM instances was not that clear. As expected 'Choose' performed slower than the CCM using $\tau = 20$ and $\lambda = 20$ but depending on the function it even performed slower than the CCM using $\tau = 1$ and $\lambda = 10$. 'Initial' showed a surprisingly large variance in its computation times and therefore was, other than expected, not always performing faster than the CCM instances.

However, as emphasized in Section 3.6, it was aimed for a strategy that works reliably and as accurately as possible on arbitrary objective functions knowingly accepting that this is probably going to happen at the expense of the ensembles computation time. Still, the computation time of the CCM depends on the choice

of the values of τ and λ and can, within bounds, be adapted to the needs of the user and with a focus on expensive real-world applications is expected to be neglectable.

To a great extent, this chapter is based on the articles *'Weighted Ensembles in Model-based Global Optimization'* by Friese et al. [74] and *"Optimally Weighted Ensembles of Surrogate Models for Sequential Parameter Optimization"* by Friese et al. [75]. Major parts from the original articles were adopted verbatim. Of course, the text was adapted to fit the structure and notation of this thesis.