Comparing CI Methods for Prediction Models in Environmental Engineering *

Oliver Flasch, Thomas Bartz-Beielstein, Artur Davtyan,
Patrick Koch, Wolfgang Konen, Tosin Daniel Oyetoyan, and Michael Tamutan

February 19, 2010

Abstract

The prediction of fill levels in stormwater tanks is an important practical problem in water resource management. In this study state-of-the-art CI methods, i.e. Neural Networks (NN) and Genetic Programming (GP), are compared with respect to their applicability to this problem. The performance of both methods crucially depends on their parametrization. We compare different parameter tuning approaches, e.g. neuro-evolution and Sequential Parameter Optimization (SPO). In comparison to NN, GP yields superior results. By optimizing GP parameters, GP runtime can be significantly reduced without degrading result quality. The SPO-based parameter tuning leads to results with significantly lower standard deviation as compared to the GA-based parameter tuning. Our methodology can be transferred to other optimization and simulation problems, where complex models have to be tuned.

1 Introduction

The prediction of fill levels in stormwater tanks based on rainfall data is an important problem in implementing predictive control of sewage networks [6, 11]. This problem will be referred to as the *stormwater problem* in the remainder of this paper. Solving the stormwater problem efficiently reduces environmental pollution and costs associated with sewer deterioration due to over- or under-loading. Additionally, this problem is a typical example of a large class of predictive control problems in environmental engineering, and it is realistic to hope that acceptable solutions for this problem are adaptable to similar problems.

The stormwater problem belongs to the class of timeseries regression problems [7]. Although many different methods, ranging from classical statistical regression to modern computational statistics, can be used for time series regression, CI-methods offer an attractive tradeoff between ease-of-deployment and prediction quality [5, 10]. Furthermore, CI-methods are robust to changes in the underlying system, which leads to low maintenance costs of CI-based systems. These findings make CI-based systems particularly suitable for application in environmental engineering, therefore we focus this paper on CI-methods.

The most prominent CI-methods applied in environmental engineering are Neural Networks (NN), while symbolic regression via Genetic Programming (GP) offers an interesting alternative. In contrast to the black-box models of NN, GP models are presented as human-readable and -interpretable mathematical formulas. Good results can be reported from the area of financial forecasting, where our project partner (DIP Dortmund Intelligence Project GmbH) applied GP successfully for several years. This is why a comparison of NN methods with GP in the domain of water resource management is of great interest. We use Sequential Parameter Optimization (SPO), as implemented in the SPOT toolbox, for tuning the parameters of all compared algorithms to enable fair and unbiased results [4]. To improve readability, the

*Oliver Flasch, Thomas Bartz-Beielstein, Artur Davtyan, Patrick Koch, Wolfgang Konen, Tosin Daniel Oyetoyan, and Michael Tamutan are with the Department of Computer Science and Engineering Science, Cologne University of Applied Science, 51643 Gummersbach, Germany (email: {oliver.flasch, thomas.bartz-beielstein, artur.davtyan, patrick.koch, wolfgang.konen, tosin.oyetoyan, michael.tamutan}@fh-koeln.de)

term SPO will be used for SPOT throughout the rest of this article. We also compare the results of SPO-tuned algorithms with the results of hand-tuning and tuning via Genetic Algorithms (GA) to verify that SPO finds near-optimal parameters for the algorithms in this study.

This paper is organized as follows: Section 2 introduces the real-world optimization problem of fill level prediction in stormwater tanks and its related objective function. Section 3 presents setup and results from four case studies: The first case study uses a classical CI-approach, which is also known as neuro-evolution: A GA is used to tune the parameters of a NN [1, 17]. The second case study applies SPO to the same NN, allowing for a direct comparison of GA and SPO as parameter tuners for NN. The third case study employs GP/INT2, a hybrid analytical and GP approach, with hand-tuned parameters. In the fourth case study, SPO is used to tune the parameters of the GP system of the third case study. Section 4 summarizes results and gives an outlook to further research.

2 Prediction of Fill Levels in Stormwater Tanks

The main goal of this study is the prediction of fill levels in stormwater overflow tanks based on rainfall data in order to implement predictive control of water drain rate. Such predictions are of immense practical utility in preventing costly and damaging over- or under-loading of the sewage system connected to these stormwater overflow tanks. The task of predicting the current fill levels from the past rain data alone—not using past fill levels—is rather challenging since the hidden state of the surrounding soil influences the impact of rain in a nonlinear fashion [5, 13].

2.1 Objective Function

The quality of a fill level predictor is measured as the root mean squared error (RMSE) between true fill level and predicted fill level on the test dataset defined in Sec. 2.2:

$$\text{RMSE}(\mathsf{y}_{pred}, \mathsf{y}_{real}) := \sqrt{\text{mean}[(\mathsf{y}_{pred} - \mathsf{y}_{real})^2]} \tag{1}$$

A fill-level predictor is a function from a rainfall time series to a scalar fill level prediction. When predicting the fill level at time t, a fill level predictor has the rainfall time series up to time t available as input. A time series of predicted fill levels is obtained by iteratively applying a fill level predictor to a time series of rainfall data.

2.2 Test Data

Training and test time series data for this study consist of $25,344$ data records. This data comprises measurements of the current fill level and the current rainfall at a real stormwater tank in Germany. These measurements were taken every 5 minutes, ranging from April, 21th 2007 (00:00 a.m.) to July, 17th 2007 (11:55 p.m.). We divided this dataset into a training dataset, ranging from April, 21th 2007 (00:00 a.m.) to April, 28th 2007 (00:00 a.m.) and a test dataset, ranging from April, 28th 2007 (00:05 a.m.) to July, 17th 2007 (11:55 p.m.). The results presented in the following are based on the test dataset, while all methods were trained on the training dataset. The training dataset consists of a short but balanced sample of dry and rainy days.

Making predictions on the training dataset had to be embedded into the fitness function of GP, therefore this dataset is comparatively small.[1] We used the same training dataset in each case study to keep results comparable. While fill levels responds differently to rain depending on season, the response stays very stable within a season. This is why the size of our training dataset should not pose a problem to the methods under study, and also why our test dataset spans the late-spring/early summer season, but not more. In practice, the methods under study would be retrained for each season.

[1] In absolute terms, the training dataset contains 2016 data points.

3 Case Studies

We conducted four case studies to assess the relative performance of different CI-based prediction methods and to test the following two scientific hypotheses:

H-1 A study by Bartz-Beielstein et al. [5] indicated that Nonlinear AutoRegressive with eXogenous inputs (NARX) [18] neural networks weren't able to yield good accuracy on the stormwater problem, but only used SPO to tune the NARX network structure and parameters. By using an established neuro-evolution approach, it is possible to evolve a NARX network structure that yields an improved prediction performance.

H-2 Flasch et al. [10] described an approach that yields good accuracy on the stormwater problem. This approach employs GP to optimize a set of integral equations and will be referred to as the GP/INT2 model in the following. A drawback of this approach is its compute-intensive GP training process. It is possible arrive at comparable results with a much shorter GP training time budget by using optimized GP parameter settings.

The first two case studies are based on NARX to test hypothesis *H-1*, while the last two case studies are based on the hybrid GP/INT2 approach to test hypothesis *H-2*. Fig. 1 gives an overview of the setups for these four case studies. We use manual tuning, GA, and SPO to obtain optimized parameter settings for NARX and GP/INT2. Since both NARX and GP/INT2 employ randomized algorithms, each optimized parameter setting was repeated 10 times on consecutive random seeds. We report descriptive statistics on these 10 runs.

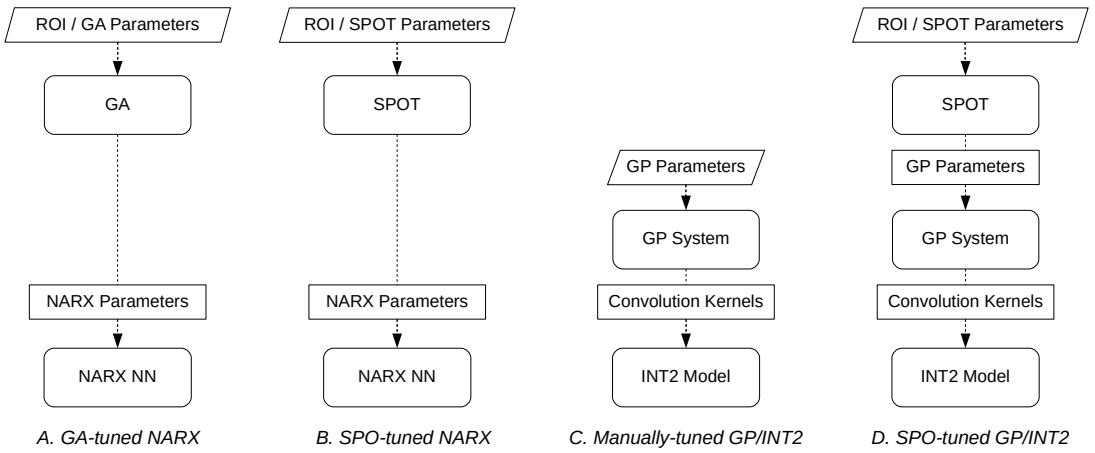

Figure 1: Case study architecture overview. This figure illustrates the setup of the four case studies, see Sec. 3. From left to right: The neuro-evolution approach (A), which is directly compared to the SPO-tuned neural network (B). The setup of the manually-tuned GP/INT2 approach (C), which is directly compared to the SPO-tuned GP/INT2 approach (D).

3.1 GA-tuned NARX

In the first case study, we employed a GA to tune the structural parameters of a NARX network (see Fig. 1.A).

3.1.1 NARX

The NARX recurrent neural network was chosen for this case study because of its dynamical neural architecture which makes it suitable for the stormwater problem. As opposed to other conventional recurrent

neural models like Elman and Hopfield neural networks, the NARX model has a limited feedback architecture, i.e. feedback comes only from the output neuron instead as from hidden neurons. The NARX model is defined as:

$$y(t+1) := f[u(t)..., u(t-d_u+1), y(t)..., y(t-d_y+1)] \tag{2}$$

$u(t)$ and $y(t)$ denote, respectively, the input and output of the model at discrete time t and d_u, d_y are the input and output memory delays, were $d_u \geq 1$, $d_y \geq 1$, and $d_u \geq d_y$. The function f is a nonlinear function which can be approximated by a multilayer perceptron.

We used the MATLAB Neural Network Toolbox 6.0.3 implementation of NARX neural networks [9]. A NARX network was trained using the training dataset. The trained NARX network was then used for multi-step time series prediction on the test dataset (see Sec. 2.2). We employed the Bayesian regulation backpropagation algorithm [9] during training to find the connection weights of the neurons for each neural network structure configuration. The NARX neural network's most influential parameters were subject to tuning as covered in Sec. 3.1.3.

3.1.2 GA for Parameter Tuning

We used a GA as an automatic tuning method for NARX structure and parameters, following a neuro-evolution approach. We mainly conducted this case study to evaluate the effectiveness of NARX parameter tuning by SPO in comparison with a well-known algorithm. Neuro-evolution is an well-established algorithm for finding near-optimal neural network structures [1, 17].

We used the GA implementation of MATLAB's Genetic Algorithm and Direct Search Toolbox 2.4.2 [12] with parameter settings as shown in Table 1.

Table 1: Parameter settings of the MATLAB GA.

GA Parameter	Value
Budget	200 Fitness Evaluations
Creation Function	Stochastic Uniform
Crossover Fraction	0.8
Crossover Function	Scatter
Elite Count	2
Fitness Scaling Function	Rank
Generations	20
Initial Penalty	10
Mutation Function	Adaptive Feasible
Penalty Factor	100
Population Size	10
Selection Function	Stochastic Uniform

These parameter settings were determined by performing several preliminary experiments. The algorithm had an allowed budget of 200 fitness evaluations, i.e., it could evaluate the RMSE of 200 NARX network structures on the test dataset.

3.1.3 Coupling NARX and GA

NARX and GA are coupled with the aim of finding improved NARX parameters. The NARX parameters with most influence on the networks performance in the given problem domain are the number of hidden layers, the number of neurons in each hidden layer, and the number of delay steps. These parameters determine the NARX network structure and were chosen to be subject of the optimization by the GA. Their respective region of interests (search spaces) are shown in Table 2.

Table 2: ROIs for tuned NARX parameters.

NARX Parameter	Region of Interest
Number of Delays	$[2, 40]$
Number of Neurons	$[(1, 1, 1, 1, 1), (10, 10, 10, 10, 10)]$
Number of Layers	$[1, 5]$

The genotype of a NARX network parametrization is an l-dimensional vector of natural numbers of the form $(d, l, n_1, \ldots, n_{lmax})$, where l denotes the number of hidden layers, $lmax$ the upper bound of the ROI chosen for l, and n_i the number of neurons in the ith hidden layer. If $l < lmax$, surplus n_is are simply ignored during the (trivial) genotype-phenotype mapping.

The fitness of a GA individual is calculated by creating a new NARX network (the phenotype) according to the parameters encoded in the individual's genotype, then training this network on the training dataset, and finally evaluating the RMSE of this trained network on the test dataset. The result of a GA parameter optimization run is the parameter setting encoded by the individual of the lowest RMSE in the population, after the algorithm has exceeded its predefined budget of fitness evaluations.

The best NARX parametrization found, shown in Table 6, was then tested on different random seeds and statistical results are reported in Sec. 3.5.

3.2 SPO-tuned NARX

In this case study, we used SPO to optimize the parameters of a NARX network, in order to assess the performance of SPO in comparison with the established neuro-evolution approach described in Sec. 3.1 (see Fig. 1.B). We will give a short introduction to SPO first. Section 3.1.1 describes the neural network which is tuned by SPO, the experiment setup for this case study is described in Sec. 3.2.2.

3.2.1 SPO

The sequential parameter optimization approach is a flexible and general framework which can be applied in many situations. Here, we introduce the *sequential parameter optimization toolbox* (SPOT) as one possible implementation of this framework. The SPO *toolbox* was developed over recent years by Thomas Bartz-Beielstein, Christian Lasarczyk, and Mike Preuss [4]. The main purpose of SPO is to determine improved parameter settings for optimization algorithms to analyze and understand their performance. SPO was successfully applied to numerous optimization algorithms [3, 20, 15, 4, 8, 2, 19, 4].

During the first stage of experimentation, SPO treats the algorithm A as a black box. A set of input variables, say $\vec{x}$, is passed to A. Each run of the algorithm produces some output, $\vec{y}$. SPO tries to determine a functional relationship F between $\vec{x}$ and $\vec{y}$ for a given problem formulated by an objective function $f : \vec{u} \rightarrow \vec{v}$. Since experiments are run on computers, pseudorandom numbers are taken into consideration if:

(i) The underlying objective function f is stochastically disturbed, e.g., measurement errors or noise occur, and/or

(ii) The algorithm A uses some stochastic elements, e.g., mutation in evolution strategies.

This situation can be described as follows:

$$\text{Objective function:} \qquad \vec{u} \xrightarrow{f} \vec{v}, \tag{3}$$

$$\text{Algorithm:} \qquad \vec{x} \xrightarrow{F} \vec{y}. \tag{4}$$

We will classify elements from (3) and (4) in the following manner:

1. Variables that are necessary for the algorithm belong to the algorithm design, whereas

Algorithm 1: Sequential parameter optimization (SPO).

```
   // phase 1, building the model:
 1 let A be the algorithm to be tuned;
 2 generate an initial population X = {x̄^1, ..., x̄^m} of m parameter vectors;
 3 let b = b_0 be the initial number of tests for determining the estimated performance;
 4 foreach x̄ ∈ X do
 5     run A with x̄ b times to determine the estimated performance y of x̄;
   // phase 2, using and improving the model:
 6 while termination criterion not true do
 7     let ā denote the parameter vector from X with best estimated utility;
 8     let b the number of repeats already computed for ā;
 9     build prediction model f based on X and {y^1, ..., y^|X|};
10     generate a set X' of l new parameter vectors by random sampling;
11     foreach x̄ ∈ X' do
12         calculate f(x̄) to determine the expected improvement f(x̄) of x̄;
13     select set X'' of d parameter vectors from X' with best expected improvement (d ≪ l);
14     run A with ā once and recalculate its estimated performance using all b + 1 test results; // (improve
           confidence)
15     let b = b + 1;
16     run A b times with each x̄ ∈ X'' to determine the estimated performance x̄;
17     extend the population by X = X ∪ X'';
```

2. variables that are needed to specify the optimization problem f belong to the problem design.

SPO employs a sequentially improved model to estimate the relationship between algorithm input variables and its output. This serves two primary goals. One is to enable determining good parameter settings, thus SPO may be used as a tuner. Secondly, variable interactions can be revealed that help to understand how the tested algorithm works when confronted with a specific problem or how changes in the problem influence the algorithm's performance. Concerning the model, SPO allows for insertion of virtually every available model. However, regression and Kriging models or a combination thereof are most frequently used.

Algorithm 1 presents a formal description of the SPO scheme that is realized by SPO. This scheme consists of two phases, namely the first construction of the model (lines 1–5) and its sequential improvement (lines 6–17). Phase 1 determines a population of initial designs in algorithm parameter space and runs the algorithm b times for each design. Phase 2 consists of a loop with the following components: By means of the obtained data, the model is built or updated, respectively. Then, a possibly large set of design points is generated and their expected improvement computed by sampling the model. A small set of the seemingly best design points is selected and the algorithm is run $b + 1$ times for each of these. The algorithm is also run once for the current best design point and b is increased by one. The new design points are added to the population and the loop starts over if the termination criterion is not reached (usually a preset budget is granted to the process). In consequence, this means that the number of repeats is always increased by one if the current best design point stays at the top of the list or a newly generated one gets there. Due to nondeterministic responses of the algorithm, it may however happen that neither of these is found at the top of the list after finishing the loop. In this case, b may effectively shrink as performance comparisons have to be fair and thus shall be based on the same number of repeats.

Sequential approaches are generally more efficient, i.e., require fewer function evaluations, than approaches that evaluate the information in one step only. Extensions of this sequential framework are discussed in Bartz-Beielstein et al. [3] and Lasarczyk [14].

We used SPO to tune parameters of NARX to minimize the RMSE on the test dataset. As already mentioned, the SPOT implementation was used to solve this problem. A main goal of this study was to compare the optimization results obtained by SPO with the results obtained by GA. For that reason, we chose the same underlying modeling method (NARX) for both parameter tuning methods.

3.2.2 Coupling NARX and SPO

This section describes the specific setup used for the tuning of NARX parameters with SPO. The tuned parameters are the same as in Sec. 3.1, i.e. the number of hidden layers, the number of neurons in each hidden layer, and the number of delay steps. The region of interest for all parameters are also the same as in the last case study (see Table 2). As in the coupling of NARX and GA, the algorithm had an allowed budget of 200 fitness evaluations. The additional computational cost of applying SPO in comparison with applying GA was negligible, both algorithms are bounded by the NARX training and evaluation time. Results of this case study are reported in Sec. 3.5.

3.3 Manually-tuned GP/INT2

Bartz-Beielstein et al. [5] and Konen et al. [13] introduced an analytic approach to predictive control in environmental engineering. They developed an analytical regression model which was customized for the stormwater problem, the so-called INT2 model. By integrating this existing analytical approach into GP, it is possible to pre-structure the GP search space to allow for a much more effective evolutionary search. This significantly improves on the results obtainable by applying standard GP alone or GP/INT2 alone [10].

The INT2 model describes the causal relationships between incoming rain $r(t)$ and the resulting stormwater tank fill level $y(t)$ by the following integral equations:

$$L(t) = \int_{-\infty}^{t} \beta_L r(\tau) e^{-\alpha_L (t-\tau)} d\tau \quad (5)$$

$$K(t) = \max(0, L(t) - \Delta) \quad (6)$$

$$y(t) - B = \int_{-\infty}^{t} r(\tau - \tau_{\text{rain}}) g(t-\tau) d\tau + \int_{-\infty}^{t} K(\tau - \tau_{\text{rain}}) h(t-\tau) d\tau \quad (7)$$

The "leaky rain" $L(t)$ of equation 6 is a leaky integration of the past rainfall and thus helps to characterize the hidden state of the soil. $K(t)$ is simply a clipped version of $L(t)$. Equation 7 models the stormwater tank fill level as the convolution of rainfall and leaky rain with certain filter kernels $g(t), h(t)$. These filter kernels are generated by symbolic regression using GP.

Our GP implementation is a slightly generalized version of *vTrader*, a commercial typed graph GP system provided by DIP Dortmund Intelligence Project GmbH[2]. vTrader represents GP individuals as term graphs of the expressions of a strict and strongly-typed functional programming language. These graphs are stored in a layered array data structure to allow efficient mutation and interpretation.

Fig. 1.C shows the layered system architecture used in this case study. Table 3 lists the parameter settings for the GP system. To give a baseline for comparison with the SPO-tuned parameters of the last case study (see Sec. 3.4), the mutation strength and population size parameters were tuned by hand by observing the results of 10 GP runs. We started with reasonable parameter settings and tried to reach better results by informed guessing, decreasing the magnitude of the changes at later tuning steps.

3.4 SPO-tuned GP/INT2

This case study builds on previous work by Flasch et al. [10] on applying GP/INT2 to the stormwater problem. The best result reported was an RMSE of 11.91573, calculated on a comparable test dataset. The termination condition of the GP was set to 2500 generations, which amounts to a total runtime of about 20 hours on a Intel Xeon 5500 (2.93 GHz, 4 GiB RAM). In this case study, we tried to improve this result by using SPO to tune the GP system parameters, and by reducing the overall compute time by setting a time

[2] *vTrader* is primarily used in financial time series prediction and portfolio optimization. See `http://www.dortmundintelligence.com/` for more information.

Table 3: Parameter settings of the vTrader GP system.

GP Parameter	Value
Objective	Find optimal convolution kernels $g(t)$ and $h(t)$ for the INT2 model.
Terminal Set	$\{t\} \cup [0.0, 200.0]$
Function Set	$\{\text{polyline}\}$
Fitness	RMSE(fill level$_{pred}$, fill level$_{real}$)
Selection	Tournament selection (tournament size 2).
Initialization	Random graphs with maximum depth of 2 levels and maximum size of 100 nodes. The population size is 10 graphs, with a maximum depth of 2 levels and a maximum size of 200 nodes.
Variation	Normal random perturbation of constant nodes.
Termination	Terminate after consuming a compute time budget of 120 minutes.
Mutation Strength	0.1
Population Size	10

budget of 2 hours as termination condition for the GP runs. The main question was if the GP system is able to generate comparable results in much shorter time when operating with optimized parameters, which refers to hypothesis *H-2*.

The experiment setup is a version of the setup used in Sec. 3.3 extended by an SPO-Layer (see Fig. 1.D). Fig. 2 shows this setup in more detail: The GP system is used to evolve near-optimal convolution kernels for the INT2 model. Subsequently, the INT2 model uses the optimized kernel to calculate the predicted time series of the stormwater tank fill level. SPO optimizes GP system parameters based on RMSE feedback, generating new parameter settings for the GP system, until the preset number of optimization steps is reached.

We chose to tune only the parameters mutation strength and population size, because preliminary experiments indicated that these parameters have the largest influence on the quality of the generated convolution kernels. The region of interest for each parameter tuned is shown in table 4. A benefit of using this small parameter space is that the number of SPO steps needed to arrive at a good setting can be very small, in this case we used 10 steps. The results of this case study are reported in Sec. 3.5.

Table 4: ROIs for tuned vTrader GP system parameters.

GP Parameter	Region of Interest
Mutation Strength	$[0.0001, 1.0]$
Population Size	$[3, 10]$

3.5 Results

We repeated each algorithm 10 times with random different seeds configured to the best parametrization found by each tuning method. Table 5 shows a statistical summary of the resulting RMSE values. While a comparison of the mean and median RMSE obtained by GA-tuned NARX and SPO-tuned NARX reveals

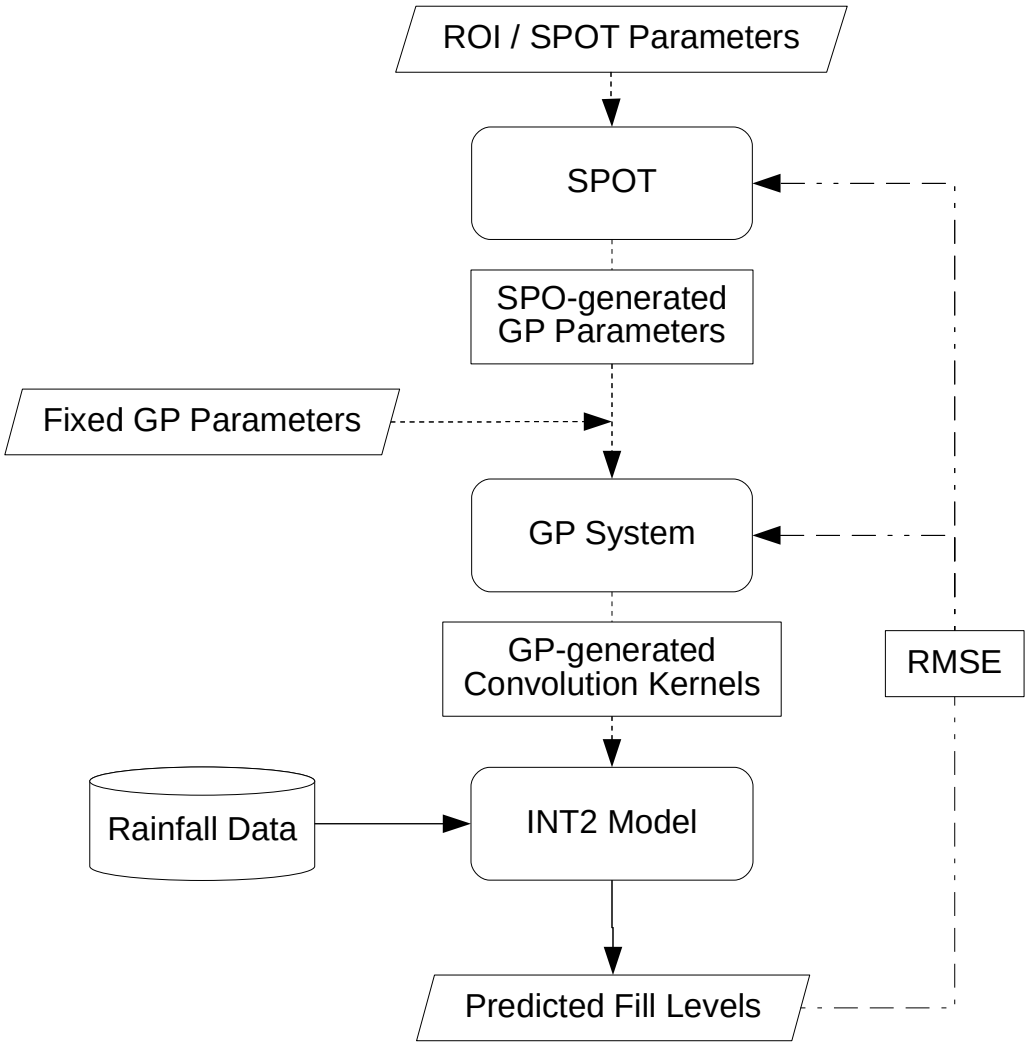

Figure 2: SPO-tuned GP/INT2 algorithm setup. SPOT parametrizes the GP system based on an ROI definition and a set of fixed parameters. The GP system then generates convolution kernels for the INT2 model, which predicts the fill level based on rainfall data. The prediction RMSE is used as an objective function for both the GP system and SPOT.

Table 5: Summary of case-study results (best values are shown in boldface).

Case Study	Algorithm	RMSE Min	RMSE Max	RMSE Median	RMSE Mean	RMSE SD
A	GA-tuned NARX	24.904	39.224	26.645	28.203	4.136
B	SPO-tuned NARX	25.789	26.635	26.562	26.436	0.343
C	Manually-tuned GP/INT2	11.639	13.347	11.745	11.966	0.535
D	SPO-tuned GP/INT2	**11.226**	**11.349**	**11.289**	**11.287**	**0.038**

Table 6: Best NARX parametrization found by GA and SPO (result RMSE values are shown in Table 5).

NARX Parameter	Best GA	Best SPO
Number of Delays	24	37
Number of Neurons	$(2, 9)$	(2)
Number of Layers	2	1

no significant difference between the two methods, the RMSE standard deviation of the SPO-tuned NARX parametrization is lower by one order of magnitude. The results obtained by SPO-tuned GP/INT2 are comparable to the results of manually-tuned GP/INT2, excluding the RMSE standard deviation. Again, the RMSE standard deviation of the SPO-tuned GP/INT2 parametrization is lower by one order of magnitude when compared to that of the manually-tuned GP/INT2 parametrization. When applied to the stormwater problem, the GP/INT2 approach is superior to the NARX approach.

Table 6 shows the best NARX parametrizations found by GA and SPO. Both parameter-tuning methods result in networks with only few layers but relatively many delay steps. Table 7 shows the best GP parametrizations found by manual tuning and SPO. While we were driven to a small mutation strength in our manual tuning, SPO chose a relatively large value for this parameter. Both tuning methods resulted in large population sizes. Due to the fixed compute time budget, larger population sizes lead to a reduced number of GP generations.

3.6 Discussion and Analysis

The comparison of GA with SPO for tuning the parameters of a NARX network shows that SPO is able to find a neural network configuration that consistently performs at least as good as the best neural network structure found by GA (see Tables 5 and 6). Notice the standard deviation of the results obtained by evaluating the best network structure found by SPO, which is one order of magnitude lower than the standard deviation of the results of the GA-tuned network. At least on this task, SPO presents a viable alternative to established neuro-evolution algorithms.

The comparison of SPO-tuned NARX with SPO-tuned GP/INT2 shows a clear advantage of the GP/INT2 approach (see Table 5). This is also readily apparent when plotting predicted fill levels versus real fill levels, as done in Fig. 3 and Fig. 4 for both prediction methods on a subset of the test dataset.

Table 7: Best GP parametrization found by manual tuning and SPO (result RMSE values are shown in Table 5).

GP Parameter	Best Manual	Best SPO
Mutation Strength	0.1	0.846
Population Size	10	8

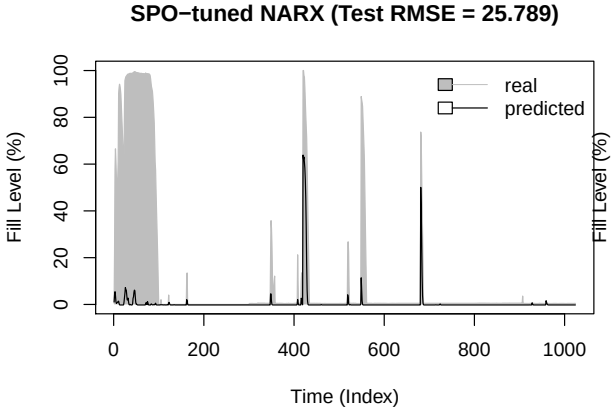

Figure 3: Real versus NARX-predicted fill levels, based on a subset of the test dataset. NARX parameters were tuned by SPO.

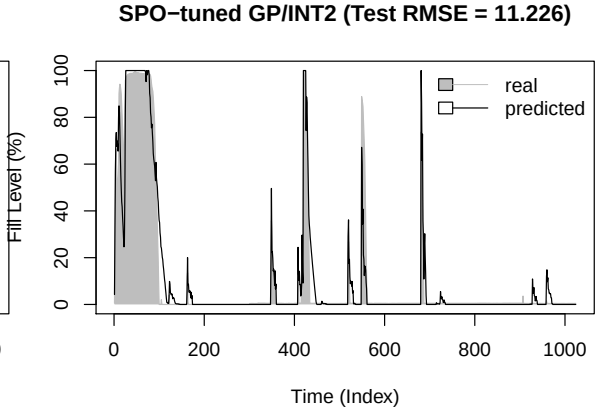

Figure 4: Real versus GP/INT2-predicted fill levels, based on a subset of the test dataset. GP parameters were tuned by SPO.

Regarding hypothesis *H-1* (see the beginning of Sec. 3), neither GA-tuned nor SPO-tuned NARX neural networks were able to reach an acceptable accuracy on the stormwater problem. We conclude that the difficulties in applying NARX to this specific problem are independent of the method used to tune the NARX network structure and parameters. We therefore have to reject hypothesis *H-1*.

Compared to previous results for the stormwater problem obtained with a combined GP/INT2 approach [10], we were able to significantly reduce the runtime required from about 20 hours to 2 hours. This improvement was due to a new GP system parametrization that was optimized to the stormwater problem. This optimization was performed completely automatically by SPO. We also tried to manually optimize the GP parameters to assess the quality of the SPO results and found them comparable to our manual results (see Table 5). When comparing the best GP parametrization found by manual tuning with the one found by SPO (see table 7), the rather large difference in the mutation strength parameter might surprise at first. The estimated function plot generated by SPO's Kriging approach shown in Fig. 5 gives an explanation: There seems to exist a local minimum at small mutation strength values. We were drawn to this local minimum in our manual optimization experiments. In summary, these findings allow us to accept hypothesis *H-2* (see the beginning of Sec. 3).

Results from all four case studies show that SPO was able to determine parametrizations with significantly smaller standard deviations compared to the other parameter tuning approaches studied (see the rightmost column of Table 5). This may be caused by SPO's modelling approach, which considers both performance and variation [16].

4 Summary and Outlook

In summary, this work makes the following contributions:

- Replacing SPO-based NARX parameter tuning by a neuro-evolution approach does not improve the results of applying NARX to the stormwater problem, therefore hypothesis *H-1* has to be rejected.
- When compared to a GA-based neuro-evolution approach, SPO finds NARX network structures and parameters that yield results with significantly lower standard deviation.
- By tuning the GP parameters in the GP/INT2 approach to the stormwater problem, the GP runtime can be reduced by one order of magnitude while still giving results of comparable quality, leading to the acceptance of hypothesis *H-2*.

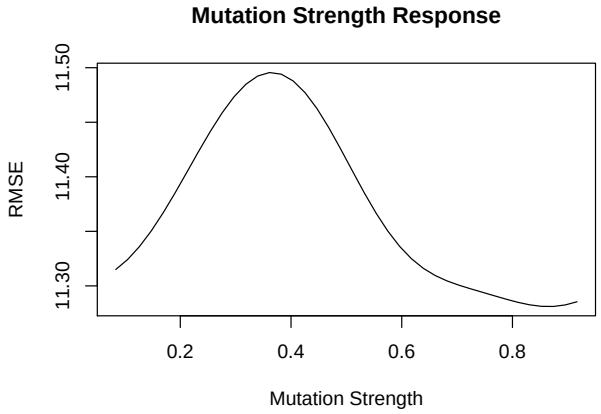

Figure 5: Plot of the estimated function for the mutation strength parameter of the vTrader GP system. This plot is based on SPO's Kriging approach as described in Sec. 3.2.1. Although Kriging is a good interpolator, it is not well suited for extrapolation. Therefore, the plot does not allow any conclusion about the shape of the curve for mutation strength values larger than 0.9. From experience with manual tuning, we would predict that the error increases rapidly for mutation strength values larger than 0.9.

- SPO is able to find parameter settings comparable with hand-tuned parameters on highly complex algorithms like GP/INT2.

In further research, we plan to study the use of GP to improve analytical models in more detail and in other applications from the domain of environmental engineering. SPO proved to be a powerful tool for tuning the parameters of GP, hence in further studies we want to extend its use to a broader set of GP parameters.

Acknowledgments

This work has been supported by the Bundesministerium für Forschung und Bildung (BMBF) under the grants FIWA (AIF FKZ 17N2309, "Ingenieurnachwuchs") and SOMA (AIF FKZ 17N1009, "Ingenieurnachwuchs") and by the Cologne University of Applied Sciences under the research focus grant COSA. We are grateful to Prof. Dr. Michael Bongards and his research group (GECO-C) for discussions and for the stormwater tank data and to Dr. Wolfgang Kantschik (DIP GmbH) for making the vTrader GP system available to us as well as for many helpful discussions.

References

[1] Peter J. Angeline, Gregory M. Saunders, and Jordan P. Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5(1):54–65, January 1994.

[2] Thomas Bartz-Beielstein. Evolution strategies and threshold selection. In M. J. Blesa Aguilera, C. Blum, A. Roli, and M. Sampels, editors, *Proceedings Second International Workshop Hybrid Metaheuristics (HM'05)*, volume 3636 of *Lecture Notes in Computer Science*, pages 104–115, Berlin, Heidelberg, New York, 2005. Springer.

[3] Thomas Bartz-Beielstein, Marco Chiarandini, Luis Paquete, and Mike Preuss, editors. *Empirical Methods for the Analysis of Optimization Algorithms*. Springer, Berlin, Heidelberg, New York, 2009. Im Druck.

[4] Thomas Bartz-Beielstein, Christian Lasarczyk, and Mike Preuß. Sequential parameter optimization. In B. McKay et al., editors, *Proceedings 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, volume 1, pages 773–780, Piscataway NJ, 2005. IEEE Press.

[5] Thomas Bartz-Beielstein, Tobias Zimmer, and Wolfgang Konen. Parameterselektion für komplexe Modellierungsaufgaben der Wasserwirtschaft – Moderne CI-Verfahren zur Zeitreihenanalyse. In R. Mikut and M. Reischl, editors, *Proc. 18th Workshop Computational Intelligence*, pages 136–150. Universitätsverlag, Karlsruhe, 2008.

[6] M. Bongards. Online-Konzentrationsmessung in Kanalnetzen – Technik und Betriebsergebnisse. Technical report, Cologne University of Applied Sciences, 2007.

[7] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer, New York NY, 2002.

[8] Marcel de Vegt. Einfluss verschiedener Parametrisierungen auf die Dynamik des Partikel-Schwarm-Verfahrens: Eine empirische Analyse. Interner Bericht der Systems Analysis Research Group SYS–3/05, Universität Dortmund, Fachbereich Informatik, Germany, Dezember 2005.

[9] Howard Demuth, Mark Beale, and Martin Hagan. Matlab neural network toolbox user's guide version 6. *The MathWorks Inc., Natick, MA*, 2009.

[10] Oliver Flasch, Thomas Bartz-Beielstein, Patrick Koch, and Wolfgang Konen. Genetic programming applied to predictive control in environmental engineering. In Rank Hoffmann and Eyke Hllermeier, editors, *Proceedings 19. Workshop Computational Intelligence*, pages 101–113, Karlsruhe, 2009. KIT Scientific Publishing.

[11] J. Gironas, L. Roesner, L. Rossman, and J. Davis. A new applications manual for the storm water management model (swmm). *Environmental Modelling and Software*, 2009.

[12] The MathWorks Inc. Matlab genetic algorithm and direct search toolbox user's guide version 2. *The MathWorks Inc, Natick, MA*, 2006.

[13] Wolfgang Konen, Tobias Zimmer, and Thomas Bartz-Beielstein. Optimierte Modellierung von Füllständen in Regenüberlaufbecken mittels CI-basierter Parameterselektion. *at – Automatisierungstechnik*, 57(3):155–166, 2009.

[14] Christian W. G. Lasarczyk. *Genetische Programmierung einer algorithmischen Chemie*. PhD thesis, Technische Universität Dortmund, 2007.

[15] Sandor Markon, Hajime Kita, Hiroshi Kise, and Thomas Bartz-Beielstein, editors. *Modern Supervisory and Optimal Control with Applications in the Control of Passenger Traffic Systems in Buildings*. Springer, Berlin, Heidelberg, New York, 2006.

[16] M. Schonlau. *Computer Experiments and Global Optimization*. PhD thesis, University of Waterloo, Ontario, Canada, 1997.

[17] Nils T. Siebel and Gerald Sommer. Evolutionary reinforcement learning of artificial neural networks. *International Journal of Hybrid Intelligent Systems*, 4(3):171–183, October 2007.

[18] H. T. Siegelmann, B. G. Horne, and C. L. Giles. Computational capabilities of recurrent NARX neural networks. Technical Report UMIACS-TR-95-12 and CS-TR-3408, 1995.

[19] Catalin Stoean, Mike Preuss, Ruxandra Gorunescu, and Dan Dumitrescu. Elitist Generational Genetic Chromodynamics - a New Radii-Based Evolutionary Algorithm for Multimodal Optimization. In B. McKay et al., editors, *Proc. 2005 Congress on Evolutionary Computation (CEC'05)*, volume 2, pages 1839 – 1846, Piscataway NJ, 2005. IEEE Press.

[20] L.G. Volkert. Investigating EA based training of HMM using a sequential parameter optimization approach. In Gary G. Yen, Simon M. Lucas, Gary Fogel, Graham Kendall, Ralf Salomon, Byoung-Tak Zhang, Carlos A. Coello Coello, and Thomas Philip Runarsson, editors, *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pages 2742–2749, Vancouver, BC, Canada, 16-21 July 2006. IEEE Press.