Genetic Programming Applied to Predictive Control in Environmental Engineering

Oliver Flasch, Thomas Bartz-Beielstein, Patrick Koch, and Wolfgang Konen

Fakultät für Informatik und Ingenieurwissenschaften, Fachhochschule Köln
E-Mail: {oliver.flasch | thomas.bartz-beielstein | patrick.koch | wolfgang.konen}@fh-koeln.de

Abstract

We introduce a new hybrid *Genetic Programming* (GP) based method for time-series prediction in predictive control applications. Our method combines existing state-of-the-art analytical models from predictive control with a modern typed graph GP system. The main idea is to pre-structure the GP search space with existing analytical models to improve prediction accuracy. We apply our method to a difficult predictive control problem from the water resource management industry, yielding an improved prediction accuracy, compared with both the best analytical model and with a modern GP method for time series prediction. Even if we focus this first study on predictive control, the automatic optimization of existing models through GP shows a great potential for broader application.

1 Introduction

Predictive control is a key technology for improving the efficiency of systems and processes in environmental engineering. It can efficiently reduce environmental pollution. Today, low priced sensors are available to measure data which describe the state of the system. These data can easily be stored on modern computer systems. One possible barrier which prevents a wide applicability of predictive control is the difficulty to rapidly develop and deploy the required software. Time series regression constitutes one core functionality of predictive control software. Although many different methods, ranging from classical statistical regression to modern computational statistics, can be used for time series regression [1], it is often not at all clear which method offers acceptable precision for a concrete regression problem.

Genetic Programming (GP) is unique among the methods of Evolutionary Computation (EC) in the sense that it can easily incorporate and recombine other techniques, especially time series regression methods. GP is able to automatically generate solution programs for arbitrary application problems in a very flexible manner [2]. The downside of this generality is a very large and unstructured search space. For many difficult real-world problems, the size and complexity of this search space quickly becomes unmanageable; our first experiments on applying GP on predictive control illustrate these difficulties (see subsection 3.3).

Bartz-Beielstein et al. [3] and Konen et al. [4] introduced an *analytic approach* to predictive control in environmental engineering. They developed an analytical regression model which was customized for this problem, the so-called INT2 model. Integrating this existing analytical approach into GP, we are able to *pre-structure* the search space to allow for a much more effective evolutionary search. We are able to significantly improve on the results obtained by applying standard GP, as well as on the results obtained by applying existing analytical time series regression methods alone. The pre-structuring does not necessarily mean that we have to abandon the complete generality of the GP approach, because every GP algorithm configuration creates a certain bias in the evolutionary search [5]. By pre-structuring the search space with existing solutions, this bias is simply adjusted from randomly chosen to more promising directions.

We conduct a first case study to illustrate how our approach works in a real-world setting. This case study is devoted to the prediction of water levels in storm-water overflow tanks based on current 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 storm-water 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. We pre-structure the GP search space with the INT2 model [3, 4]. The case study compares the prediction RMSE of our method with the prediction RMSE of standard GP and INT2.

Training and verification time series for this case study consist of more than $100,000$ data records, ranging from April to July 2007. New data from 2008 and 2009 will be used in the near future of our ongoing research project and will validate the generalizability of our approach. These time series are real historical data, collected from storm-water overflow tanks in Germany.

This paper is structured as follows: Section 2 describes the GP approach and shows how we apply this approach to time series prediction. Section 3 presents and discusses first experimental results. Section 4 concludes this paper with a short summary of the results and presents directions for further research.

2 Optimization of existing analytical models through GP

GP is an EC method that automatically generates solutions from high-level problem definitions. GP proceeds by generating a population of computer programs and then refining them in an evolutionary process of variation and selection. To guide this evolutionary process, a solution quality measure, called a fitness function, has to be provided. This fitness function maps candidate solutions (called individuals, taking the form of computer programs) to numerical fitness values. In contrast to other Evolutionary Computation methods, like Evolutionary Systems (ES) or Genetic Algorithms (GA), the structure of the solution does not have to be known in advance. In contrast to solutions generated by sub-symbolic EC methods, like Artificial Neural Networks or Support Vector Machines (SVM), solutions generated by GP are symbolic expressions. This has the benefit of GP solutions being readable and understandable by humans. Furthermore, GP's symbolic solutions are accessible to formal methods of program construction and verification. First, this means that GP solutions are modular and scalable; they can easily be combined to form larger units. Second, the correctness of GP solutions can (at least in principle) be formally proved.

This flexibility of GP comes at a certain cost: The algorithm has an exceptionally large parameter space, containing not only typical EC parameters, like population size and

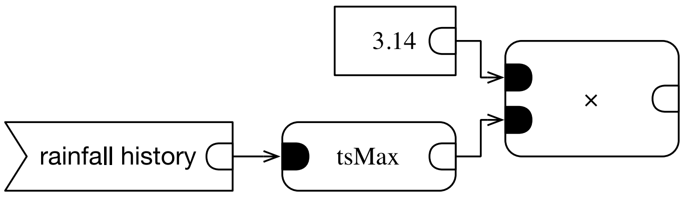

Figure 1: A simple example for an individual graph in our GP system.

selection method, but also much more complex parameters concerning program representation (e.g. linear, tree, or graph), program variation (mutation, crossover), and initial program generation (random, seeding with pre-specified individuals). A highly important parameter of GP is the set of primitive building blocks programs are created from, called the GP function and terminal set. Automatic parameter optimization techniques, such as *Sequential Parameter Optimization* (SPO) [6], can be employed to alleviate the difficulty of tuning this large parameter space.

An even more difficult problem is posed by the GP search space, the space of all expressions in a certain formalism (programming language). The computationally tractability of this search space depends on a complex combination of several factors, including the fitness function, the function and terminal sets, and the variation operators. For an effective evolutionary search, a certain causality between a program and its fitness value has to be maintained, in the sense that a small mutation of the program must not lead to an unbounded mutation of its fitness value. Maintaining this causality when parameterizing GP can be surprisingly difficult, for with many function and terminal sets, small changes to a program can cause radical changes in behaviour. Even if an effective evolutionary search is maintained by careful parameterization, the GP search space may be much too large to reliably find good solutions for more difficult real-world problems. In practice, it is often unavoidable to pre-structure this search space, as we show in section 2.3.

2.1 Typed graph GP for general time series prediction

Our GP implementation is a slightly generalized version of *vTrader*, a commercial typed graph GP system provided by DIP Dortmund Intelligence Project GmbH[1]. vTrader represents GP individuals as term graphs of the expressions of the strict and strongly-typed functional programming language gp$\mathcal{L}$. These graphs are stored in a layered array data structure to allow efficient mutation and interpretation.

As gp$\mathcal{L}$ is strongly-typed, we define the set of types T first: A type $\tau \in T$ is exactly either one of the base types $B := \{$`boolean`, `integer`, `double`, `string`, `date_time`, `time_series`$\}$, or a n-ary function type $(\delta_1, ..., \delta_n) \rightarrow \nu$, for base types $\delta_1, ..., \delta_n$, and ν.

The set of terms of gp$\mathcal{L}$ is then defined as follows: A term of in gp$\mathcal{L}$ is exactly either a constant literal c of base type $\beta \in B$, or an input variable x of base type $\beta \in B$, or a n-ary function application $f(t_1, ..., t_n)$ of type $\tau \in T$, where f must be of type $(\delta_1, ..., \delta_n) \rightarrow \tau$ and the terms t_i of base types $\delta_i \in B$ for $i \in [1, n]$.

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

Consider as an example the valid gp$\mathcal{L}$ term $3.14 \times \mathrm{tsMax}(\textsf{rainfall history})$. This term is a function application and has the base type `double`. The function $\times$ is of type $(\texttt{double}, \texttt{double}) \rightarrow \texttt{double}$, the function tsMax is of type $(\texttt{time\_series}) \rightarrow \texttt{double}$, the constant literal 3.14 is of type `double`, and the input variable rainfall history is of type `time_series`. Figure 1 shows how this term would be represented as term graph in our GP system.

Building on this basic structure, vTrader provides a comprehensive default function and terminal set consisting of literals of all supported scalar types, arithmetic functions, statistical functions, boolean operators, as well as if-then-else branches and comparison functions. The resulting language is minimal in the sense that it does not contain higher-order or generic functions, but it is expressive enough to compactly represent interesting algorithms for time series prediction.

Populations can be initialized by randomly generating graphs of the required type, or by seeding with prefabricated graphs. The required graph type depends on the application problem and on the available input data. vTrader implements grow-, shrink-, strong-, and weak-mutation operators. These operators respectively add, remove, change, and perturbate random graph nodes. The strong-mutation operator replaces subgraphs with randomly generated graphs of matching type, while the weak-mutation operator perturbates numerical literals. An implementation of a crossover operator is planned, but not finished at the time being.

To employ our GP system for time series regression, we evolve individuals of the type $(\texttt{time\_series}, ..., \texttt{time\_series}) \rightarrow \texttt{double}$. To use such an individual as a predictor, we just apply it to a window of our current set of input time series and interpret the resulting `double` value as the predicted value. The prediction horizon, as well as which time series to predict, is encoded in the fitness function as follows: We use a training set containing historical data of S_{target}, the time series to predict, as well as of $\mathcal{S} := \{S_{input_1}, ..., S_{input_n}\}$, the set of input time series (S_{target} may be contained in $\mathcal{S}$). To measure the prediction accuracy of an individual I at time t with horizon length h, we first window all input time series to the interval $[t-(h+w), t-h]$ for some fixed window length w. We then apply I to the set of windowed input time series to calculate I's prediction of $S_{target}(t)$ and record the squared error SE between the real value at $S_{target}(t)$ and this prediction. To approximate the prediction RMSE, we take the root mean of a number of randomly sampled values of SE on the training dataset, and take this approximation as the fitness value of I.

When later deploying an individual I as a predictor in a predictive control application, I simply has to be applied as a function to a window (of length w) of the current real-time input dataset.

2.2 The INT2 model for predictive control of stormwater tanks

We investigated in our previous work [3, 4] the stormwater tank problem as described in Sec. 1 with different modeling approaches, among them FIR, NARX, ESN, a dynamical system based on ordinary differential equations (ODE) and a dynamical system based on integral equations (INT2). All models were systematically optimized using SPO [6]. Among these models the INT2 approach turned out to be the best one, although the full modeling potential was not yet assessed in our previous work [3]. In the following we

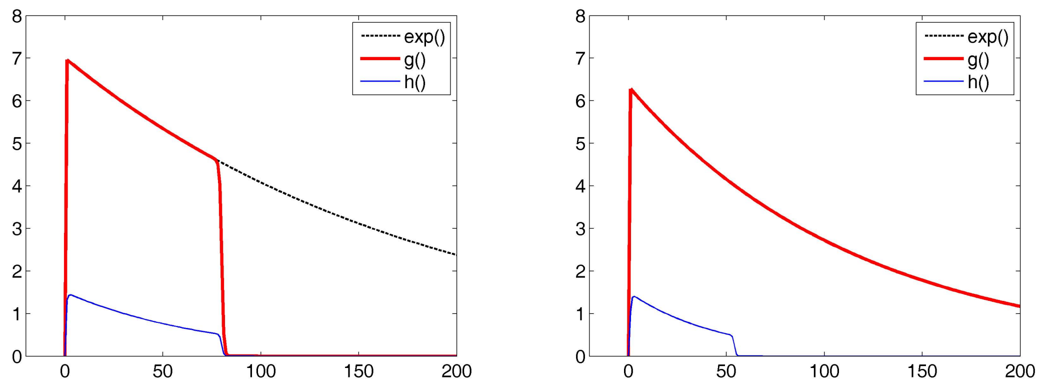

Figure 2: Analytic convolution kernels of the INT2 model. Left: manual setting, right: after SPO-optimization (from [3]).

describe briefly the INT2 model (for a more detailed presentation see [4]) and show how GP can be utilized to improve this model further.

The INT2 model

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

$$
\begin{aligned}
L(t) &= \int_{-\infty}^{t} \beta_L r(\tau) e^{-\alpha_L (t-\tau)} d\tau && (1)\\
K(t) &= \max(0, L(t) - \Delta) && (2)\\
y(t) - B &= \int_{-\infty}^{t} r(\tau - \tau_{\text{rain}}) g(t-\tau) d\tau \\
&+ \int_{-\infty}^{t} K(\tau - \tau_{\text{rain}}) h(t-\tau) d\tau && (3)
\end{aligned}
$$

As an intermediate quantity we introduce in Eq. (2) the 'leaky rain' $L(t)$ which is a leaky integration of the past rainfall and thus may help to characterize the hidden state of the soil. $K(t)$ is simply a clipped version of $L(t)$. Eq. (3) models the stormwater tank fill level as the convolution of rainfall and leaky rain with certain filter kernels $g(t), h(t)$. A specific filter kernel like

$$g(t) = \beta e^{-\alpha t}$$

would correspond to the leaky integration of Eq. (1). But other forms are equally well possible. As an example one might consider truncated exponential functions with smooth on- and offset

$$
\begin{aligned}
g(t) &= \beta \sigma(t - \tau_{in}) e^{-\alpha t} \sigma(\tau_{out} - t) && (4)\\
h(t) &= h_0 \sigma(t - \tau_{in3}) e^{-\alpha_H t} \sigma(\tau_{out3} - t) && (5)\\
\text{with} \quad \sigma(t) &= \tfrac{1}{2}(1 + \tanh(\kappa t)), && (6)
\end{aligned}
$$

which are depicted in Figure 2.

Kernel optimization The flexibility of the INT2 model—in principle any causal kernel function can be used—constitutes an opportunity and a burden at the same time. The opportunity is that a rich set of pre-structured model functions is at hand. The burden is that it is manually impossible to investigate in a systematic way a large set of possible functions. The parameterization example Eqs. (4), (5)—although containing 9 tunable parameters—characterizes only a small set of possible kernel functions. The application of GP was motivated by the need to investigate a much richer set of possible kernel functions. the following.

2.3 Optimizing INT2 through GP

Even though the INT2 model has been established as the best alternative in our previous work (see [3]), it still has some drawbacks, as already described in Sec. 2.2. Especially the requirement to choose a specific kernel function is an important problem, which can have a significant impact on the results. Of course it might be possible to build other kernel functions by the practitioner, but one can not be sure to find the optimal kernel. Due to the infinite function space, it seems to be as looking for the needle in a haystack.

With the integration of the kernel function into the GP system, this additional burden for the practitioner can be removed. Also, this integration might be a promising way to evolve more suitable kernel functions. When automatically evolving kernel functions for the INT2 model, it is of course of large interest whether it is possible to achieve an improvement over the hand- and SPO-tuned model.

Pre-structured GP

One can think of integrating other models as pre-structuring elements into the GP system and to compare the results of these new GP-evolved models with the results of the constituent models. The optimization of models based on ordinary differential equations (ODE) might be such an alternative. However, as we have achieved the best results with the INT2 model in our previous research, we first concentrate on joining the INT2 model with our GP system. We identified three possible approaches for doing so:

- *Pre-structuring individuals* for the initial population of the GP algorithm: Individuals are initialized with the INT2 model itself. These individuals are optimized by the general evolutionary process, e.g., mutation and recombination. The underlying concept of this approach can be described as follows: The INT2-initialized population provides an established model in the solution set from the beginning of the search which might accelerate the evolution. This approach is very general and powerful, but requires the reformulation and reimplementation of the established model in the programming language used as the individual representation of the underlying GP system (gp$\mathcal{L}$ in our case).
- *Integration of the INT2 model* into the function set of the GP system: In this approach, variation operators can choose the model as a building block for creating new individuals. With this approach to model integration, the function set gains a much more powerful element in comparison to the standard functions generally used in GP.

- *Embedding the INT2 model* into the fitness function of the GP system: Each time the fitness function is evaluated, the INT2 model is calculated based on GP-generated convolution kernels. The prediction RMSE of the INT2 model is then used as a fitness value for the GP system.

We choose the third approach in this work, because it offers a simple and practicable way to use the INT2 model with the GP system. The main advantages of this solution are that there is no need to extend the standard function set of GP, or to restrict the search space to certain areas, as done in the first option. The integration into the fitness function does not require any changes to the GP system itself and is therefore easy to implement in any given GP system. Also, the individuals themselves can remain rather simple and do not need to grow too much during the search, which avoids a drawback of genetic programming (see e.g. *bloat* in genetic programming). It is of cause possible to combine all three approaches to model integration, for example by implementing only some components of the model as GP functions to allow for free recombination of these components, and keeping other parts of the model fixed in the fitness function. The exploration of these possibilities is a theme for further research.

3 Experimental results

3.1 Experimental setup

In this section, we present first experimental results of applying our method to the real-world problem of water level prediction in stormwater tanks based on historic rainfall data. To generate a baseline for comparisons, we applied the INT2 model, as the current best known prediction model for this problem. We then applied standard GP, i.e. GP with a standard function set, to this problem. Finally, we applied our GP-optimized version of INT2. All experiments were performed on the same set of input data.

Our input data consists of measurements of the current water level and the current rainfall at a real stormwater tank in Germany. These measurements were taken every 5 minutes, ranging from 01 April 2007 (10:15) to 18 July 2007 (10:10). We divided this dataset into a training dataset, ranging from 21 April 2007 (00:00) to 28 April 2007 (00:00), and a test dataset, ranging from 28 April 2007 (00:05) to 18 July 2007 (10:10). The first 20 days of April 2007 were nearly completely dry, so they were omitted from the training dataset. All the results presented in the following are based on the test dataset.

Since both GP and SPO are randomized algorithms, each experimental setting was repeated 5 times on consecutive random seeds. Our experiments show that the quality of the results did not vary much. We only present the best results obtained in the 5 runs of each experimental setting. The experimental settings involving GP were parameterized to allow them a comparable quota of computing time. Each of the total of 10 GP runs took 4 hours of computing time a single core of an Intel Xeon E5530 (2.4 GHz) CPU. Each run was allowed a maximum memory usage of 4 GiB.

3.2 INT2 with analytic kernels

As a baseline for comparisons, we applied the INT2 model to our test data, using the parameters as described in [3, 4]. In these studies, INT2 was found to give the best results

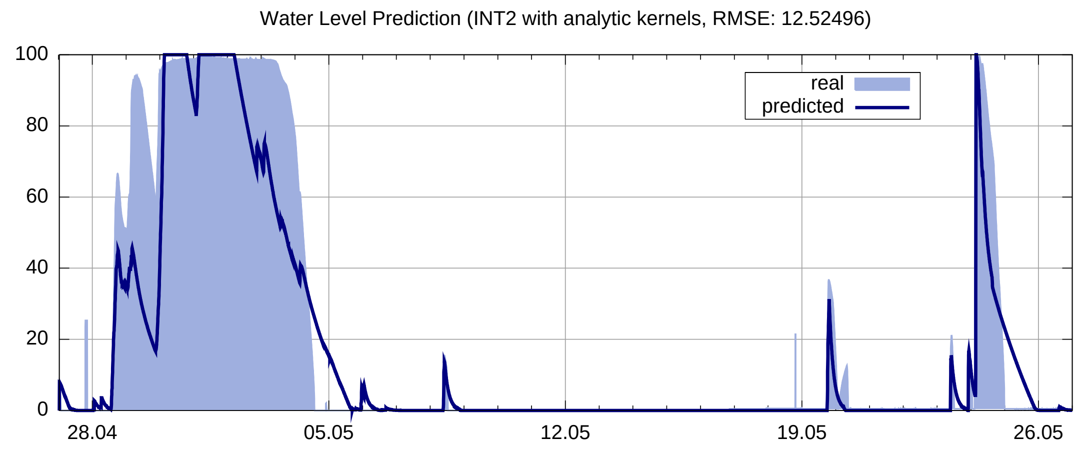

Figure 3: Water level prediction with INT2, using SPO-optimized analytic kernels.

of all investigated modeling approaches, including FIR, NARX, ESN, and a dynamical system based on ordinary differential equations (ODE).

We used the SPO-optimized analytical INT2 convolution kernels $g(t)$ and $h(t)$ based on the parametrization of Eqs. (4), (5), and shown on the right of Figure 2. The parametrized functions of these kernels were defined manually, taking background knowledge about the water level prediction into account. Then the 9 parameters of these functions together with other model parameters were optimized using SPO, giving a near optimal parameterization of the INT2 model. In contrast, with our GP method, we will try in Sec. 3.4 to evolve suitable kernels without any problem-specific background knowledge or human intervention. Figure 3 shows the results of applying the INT2 model with SPO-optimized convolution kernels to our test dataset. The filled curve represents the real water levels, while the dark solid line represents the predicted water levels.

3.3 Standard GP

To our knowledge, this is the first application of GP to the stormwater tank water level prediction problem. We proceeded as described in Sec. 2.1, using the GP parameters shown in Table 1. We deliberately choose a simple "standard" function set, containing only basic arithmetic, logic, comparison, control flow, and time series processing functions. These functions should be available by default on every modern typed GP system. Another reason for choosing this simple function set is that practitioners often do have neither the time nor the resources to extensively tune a GP function set.

We parameterized the GP system so that only rainfall data could be used as input to the evolved predictor functions. The predictions of the best individual found are shown in figure 4. This rather disappointing result was to be excepted, given the vast search space of predictor functions and given the other reasons already mentioned in Sec. 2.

A customized problem-specific function set would probably greatly improve GP performance. But for our problem, as for many other real-world problems, it is very difficult to define such an problem-specific function set without detailed background knowledge. That is why we turn to existing analytical models for guidance. Our approach of integrating these existing models into GP can be seen as a first step towards formalizing

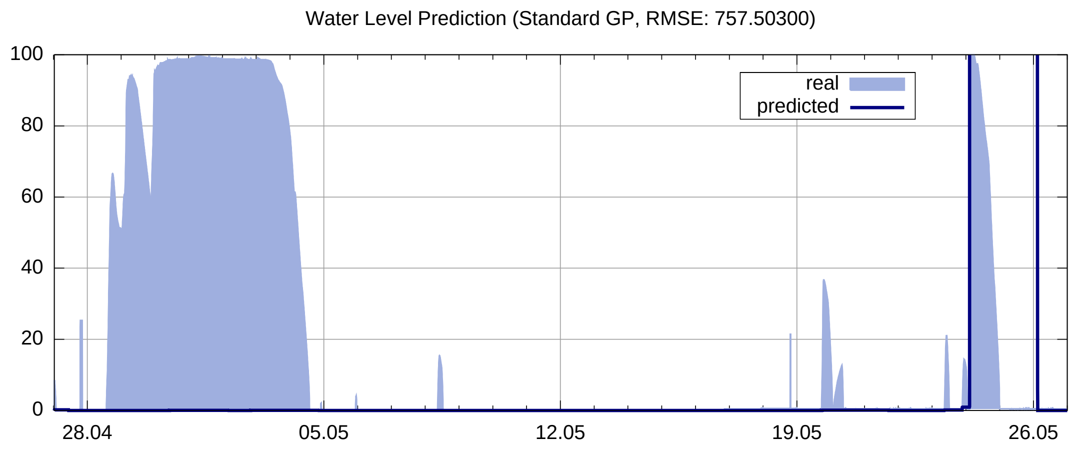

Figure 4: Water level prediction through GP with a standard function set.

the process of inferring improved evolutionary algorithms for real-world problems from existing analytical models.

Table 1: Parameters used for standard GP runs.

Objective:	Find a predictor function for the water level in stormwater tank, based on historical rainfall data. The prediction horizon is 0 minutes.
Terminal set:	$\{\text{rainfall history}\} \cup [-1.0, 100.0]$
Function set:	$\{+, \times, -, \%(\textit{safe divide}), \mathrm{abs}, \max, \min, \ln, \mathrm{ifThenElse}, <, <=, >, >=$ $, \mathrm{not}, \mathrm{and}, \mathrm{or}, \mathrm{tsMax}, \mathrm{tsMin}, \mathrm{tsSlope}, \mathrm{tsArithmeticMean}, \mathrm{tsStdDev}, \mathrm{tsSum}\}$ Functions prefixed with ts are defined on time series.
Fitness:	$\mathrm{RMSE}(\text{water level}_{predicted}, \text{water level}_{real})$
Selection:	Tournament selection with tournament size 2.
Initialization:	Random graphs with maximum depth of 20 and maximum size of 100 nodes. The population size is 20 graphs, with a maximum depth of 40 and a maximum size of 200 nodes.
Variation:	3 steps per individual variation, with a probability for grow-mutation of 0.6, shrink-mutation of 0.0 , strong-mutation of 0.8, and weak-mutation of 0.2 at each step.
Termination:	Terminate after $2,500$ generations ($25,000$ fitness evaluations).

3.4 INT2 with GP-optimized kernels

In the following experiments, we evolved the INT2 convolution kernels $g(t)$ and $h(t)$ through our GP system, using the parameterization shown in Table 2. Both kernels are functions of type $(\texttt{double}) \rightarrow \texttt{double}$ that are simultaneously co-evolved in two separate populations in our GP system. Settings that only evolved $g(t)$ or $h(t)$, using the respective SPO-optimized kernels $h(t)$ or $g(t)$ to complete the model, were also tried. While these settings delivered good results in less time due to lesser problem complexity, the results after $2,500$ generations were comparable to the results of the co-evolution setting described here.

To ensure an unbiased GP search space of manageable size, the GP function set in this configuration contains the $\mathrm{polyline}$ function only. This function behaves as a piecewise linear function of its first parameter defined by the control points given as its remaining

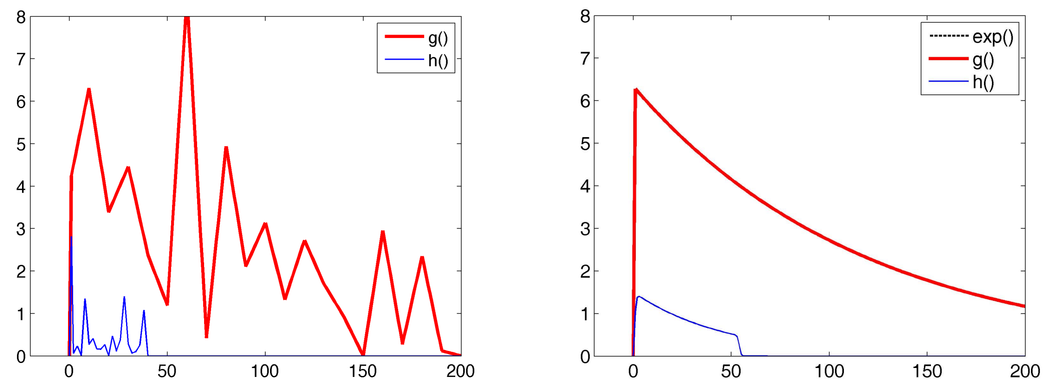

Figure 5: Left: GP-evolved convolution kernels for the INT2 model. Right: SPO-optimized kernels for comparison.

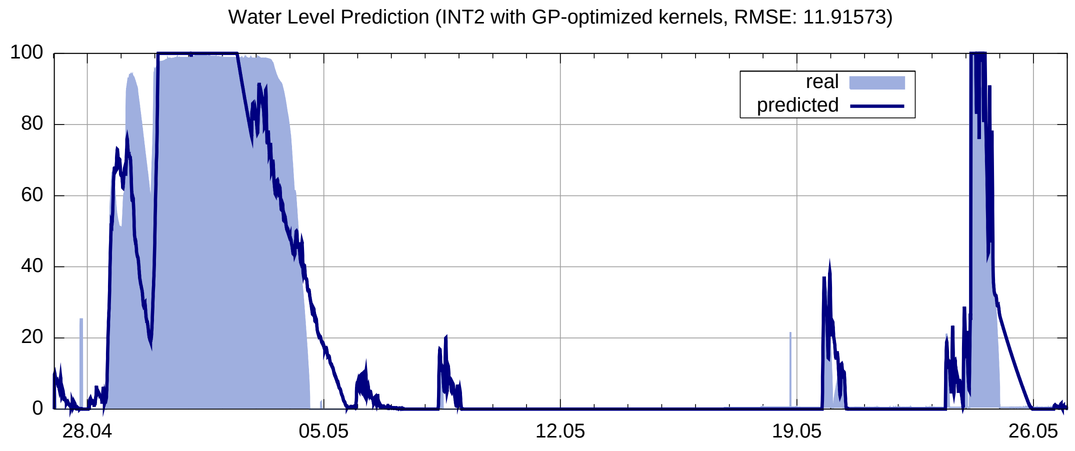

Figure 6: Water level prediction with INT2, using the GP-evolved kernels of figure 5.

parameters. In this setting, we used 20 control points, given by 40 values of type `double`, resulting in a polyline function of arity 41. A standard GP function set, as in subsection 3.3 could have also been used, which would be topic for further research. We chose the a piecewise linear function in this setting to enable an unbiased search for kernel functions of all forms, while keeping the search space reasonable small.

Figure 5 shows the best kernels found in 5 runs. The predictions of the INT2 model using these kernels are shown in Figure 6. As can be seen in this figure, these results are slightly better than the results based on SPO-optimized analytic kernels presented in subsection 3.2. It might be possible to further improve on these results by employing SPO to optimize the remaining INT2 model parameters (such as τ_{rain}) for the GP-evolved kernels. Here, we just used the same settings as in subsection 3.2 for these parameters.

3.5 Discussion

Table 3 summarizes our experimental results. The prediction RMSE column shows the best result out of 5 runs for each method. It was calculated on the test dataset defined in Sec. 3.1. With a prediction RMSE more than 60 times larger than the other two methods,

Table 2: Parameters used for GP-optimized INT2 runs.

Objective:	Find optimal convolution kernels $g(t)$ and $h(t)$ for the INT2 model.
Terminal set:	$\{t\ (i.e.\ input\ variable\ the\ kernel\ function)\} \cup [0.0, 200.0]$
Function set:	$\{\text{polyline}\}$
Fitness:	$\text{RMSE}(\text{water level}_{predicted}, \text{water level}_{real})$
Selection:	Tournament selection with tournament size 2.
Initialization:	Random graphs with maximum depth of 2 and maximum size of 100 nodes. The population size is 10 graphs, with a maximum depth of 2 and a maximum size of 200 nodes.
Variation:	5 steps per individual variation, with a probability for grow-mutation of 0.0, shrink-mutation of 0.0 , strong-mutation of 0.0, and weak-mutation of 1.0 at each step.
Termination:	Terminate after $2,500$ generations ($12,500$ fitness evaluations).

GP with a standard function set does not lead to sensible results. Pre-structuring the GP search space with the INT2 model on the other hand gives a result even slightly better than the so far best known approach for this problem, INT2 with analytic kernels.

Table 3: Overview of the experimental results. The prediction RMSE for each method was calculated on the test dataset.

Method	*prediction RMSE*
INT2 with analytic kernels	12.52496
Standard GP	757.50300
INT2 with GP-optimized kernels	11.91573

At first sight it might seem a bit awkward that the GP-evolved kernels in Figure 5 show such a rugged shape. Wouldn't a smooth shape be more natural? Although this might seem reasonable, one has to keep in mind that there was no evolution pressure towards smooth kernel functions and that the rain data itself show on the relevant timescale a very non-smooth, burst-like behaviour (sharp 'needles'). Therefore it is quite natural for the GP-system to 'invent' also a non-smooth kernel function $g(t)$ which tries to optimize the effects of certain needles on the subsequent fill levels in our training data. Keep in mind that the time span of the training data in this study is rather small. We expect the GP-evolved kernels to become smoother when we are using more training data, which has however to be confirmed by future research.

4 Conclusions and outlook

GP is difficult to apply directly to real-world time series prediction problems, at least when using a standard function set. Hand-tuning the function set promises improved results, but is often very difficult to do in practice. Furthermore, for many real-world problems, it is not at all clear if it is easier to find a suitable GP function set than to find a suitable modeling function directly. GP shows its strength in domains where there are many modeling functions that work only some of the time, without clear rules when each of these modeling functions should be applied. The most striking example for this kind of problem domain is financial time series prediction, the problem domain our GP system vTrader has been designed for. The vTrader GP function set contains building blocks for a wide array of financial time series regression models, which are then selected and recombined during GP search based on the accuracy the give on the most current financial time series data.

We showed that by pre-structuring the GP search space with an established analytical model of the prediction problem to solve, it is possible to improve on the results of even an SPO-optimized version of this model. Given an easy to apply pre-structuring method for the GP search space, GP could be very useful in optimizing arbitrary components of existing models. Such a method could be perceived not so much as an extension of GP, but as a design method for application-specific evolutionary algorithms.

In this first study, by integrating our analytical model in the GP fitness function, we only implemented the simplest approach to model integration. By implementing the other approaches to model integration described in subsection 2.3, many other interesting opportunities for applying GP in the optimization of analytical models would open. By integrating the constituent parts of the INT2 model as functions in the GP function set, the GP system could recombine these parts in novel ways to find a better configuration of the model. By controlling the granularity of these parts, the size of the GP search space can be controlled.

As already pointed out in Sec. 3.5, we currently are in the process of applying our methods to larger datasets (bigger time span, more different stormwater tanks) in order to test the generalizability of our model and to see whether GP-evolved kernels trained on larger datasets would become smoother.

Additional directions of further research would be the evaluation of function sets other than the one used in subsection 3.4 for the GP-evolution of INT2 convolution kernels. Also, applying our approach to other problems in predictive control that have existing analytical models is of great interest.

5 Acknowledgements

This work has been supported by the Bundesministerium für Forschung und Bildung (BMBF) under the grants FIWA 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] Brockwell, P. J.; Davis, R. A.: *Introduction to Time Series and Forecasting*. New York NY: Springer. 2002.

[2] Smits, G.; Vladislavleva, E.: Ordinal Pareto Genetic Programming. In: *Proceedings of the 2006 IEEE Congress on Evolutionary Computation* (Yen, G. G.; Lucas, S. M.; Fogel, G.; Kendall, G.; Salomon, R.; Zhang, B.-T.; Coello, C. A. C.; Runarsson, T. P., Hg.), S. 3114–3120. Vancouver, BC, Canada: IEEE Press. URL http://ieeexplore.ieee.org/servlet/opac?punumber=11108. 2006.

[3] Bartz-Beielstein, T.; Zimmer, T.; Konen, W.: Parameterselektion für komplexe Modellierungsaufgaben der Wasserwirtschaft – Moderne CI-Verfahren zur Zeitreihenanalyse. In: *Proc. 18th Workshop Computational Intelligence* (Mikut, R.; Reischl, M., Hg.), S. 136–150. Universitätsverlag, Karlsruhe. 2008.

[4] Konen, W.; Zimmer, T.; Bartz-Beielstein, T.: Optimierte Modellierung von Füllständen in Regenüberlaufbecken mittels CI-basierter Parameterselektion. *at – Automatisierungstechnik* 57 (2009) 3, S. 155–166.

[5] Poli, R.; Langdon, W. B.; McPhee, N. F.: *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`. URL `http://www.gp-field-guide.org.uk`. (With contributions by J. R. Koza). 2008.

[6] Bartz-Beielstein, T.: *Experimental Research in Evolutionary Computation—The New Experimentalism*. Natural Computing Series. Berlin, Heidelberg, New York: Springer. 2006.