A Genetic Decomposition Algorithm for Predicting Rainfall within Financial Weather Derivatives

Sam Cramer
University of Kent
sc649@kent.ac.uk

Michael Kampouridis
University of Kent
M.Kampouridis@kent.ac.uk

Alex Freitas
University of Kent
A.A.Freitas@kent.ac.uk

ABSTRACT

Regression problems provide some of the most challenging research opportunities, where the predictions of such domains are critical to a specific application. Problem domains that exhibit large variability and are of chaotic nature are the most challenging to predict. Rainfall being a prime example, as it exhibits very unique characteristics that do not exist in other time series data. Moreover, rainfall is essential for applications that surround financial securities such as rainfall derivatives. This paper is interested in creating a new methodology for increasing the predictive accuracy of rainfall within the problem domain of rainfall derivatives. Currently, the process of predicting rainfall within rainfall derivatives is dominated by statistical models, namely Markov-chain extended with rainfall prediction (MCRP). In this paper, we propose a novel algorithm for decomposing rainfall, which is a hybrid Genetic Programming/Genetic Algorithm (GP/GA) algorithm. Hence, the overall problem becomes easier to solve. We compare the performance of our hybrid GP/GA, against MCRP, Radial Basis Function and GP without decomposition. We aim to show the effectiveness that a decomposition algorithm can have on the problem domain. Results show that in general decomposition has a very positive effect by statistically outperforming GP without decomposition and MCRP.

CCS Concepts

•Computing methodologies → Classification and regression trees; Genetic algorithms; Genetic programming;

Keywords

Rainfall derivatives, Rainfall prediction, Decomposition, Genetic Programming

GECCO '16, July 20 - 24, 2016, Denver, CO, USA

ISBN 978-1-4503-4206-3/16/07…$15.00

DOI: http://dx.doi.org/10.1145/2908812.2908894

1. INTRODUCTION

Regression based problems are some of the most challenging research opportunities. The problem difficulty is not just based on how effective a certain technique is, but can be of a direct result of the internal structure of the data. Complex and chaotic data structures that exhibit few trends or patterns can be exceptionally hard to predict, especially if tackling the problem as a whole. One such data series that exhibits high volatility, few reoccurring patterns and fluctuating trends is rainfall. Such data sets are critical to predict with the highest of accuracy when considering the application domains where rainfall prediction is key.

Rainfall is one of the most important phenomena within a climate system. It is well known that the variability and intensity of rainfall impacts water resource planning, agriculture and biological systems. In the financial aspects, predicting the amount of rainfall is also a vital component for predicting financial securities. Over recent years, the abilities in understanding and predicting rainfall has increased, due to numerous models developed for increasing the accuracy of prediction in rainfall amounts. Subsequently, such efforts can lead to the correct predictions for weather derivatives. Rainfall derivatives share similar principles with weather derivatives and other regular derivatives, defined as contracts between two or more parties. The value of the contracts is dependent upon the underlying financial asset.
Hence, in the case of weather derivatives, the underlying asset is a weather type, such as rainfall. Another significant difference between common derivatives and weather derivatives is that weather derivative are not tradable. Therefore, many existing methods in the literature for other derivatives become unsuitable for the prediction of weather derivatives.

Rainfall derivatives is a new method for reducing the financial risk posed by adverse or uncertain weather circumstances. Moreover, they are a better alternative than insurance, because it can be hard to prove that the rainfall has had an impact unless it is destructive, such as severe floods or drought. Similar contracts exist for other weather variables, such as temperature.

The pricing of rainfall derivatives consists of two problems. The first problem is the prediction of accumulated rainfall over a specified period. The second problem is developing a pricing framework[1]. The latter has its own unique problematic features, as rainfall derivatives constitute an incomplete market[2]. This paper focuses on the first aspect of

[1]Where deriving a fair price for a contract is key (i.e. how much a unit of rainfall costs)

[2]In incomplete markets, the derivative can not be replicated

predicting the level of rainfall. Note it is important to have a model that can accurately predict the level of rainfall before pricing derivatives, because the contracts are priced on the predicted accumulated rainfall over a period of time. Hence, we want to reduce issues of mispricing.

Prediction in rainfall derivatives poses many obstacles, both in research and in financial practice. There is a light amount of literature researched in rainfall derivatives, because the concept is fairly new, as well as rainfall can be very difficult to accurately measure. In financial practice, investors also share the same kind of difficulties, which minimise the trades of rainfall derivatives in financial markets. Therefore, our motivation is to develop a methodology for accurate rainfall prediction.

The statistical approaches of Markov-chain extended with rainfall prediction (MCRP) [9] is the most commonly and successfully used rainfall prediction method in the literature[3]. Where the process is broken down into two stages. The first stage is to produce an occurrence pathway using a Markov-chain (i.e. a sequence of rainy or dry days). The second stage is to generate a random rainfall amount (from a distribution) for every rainy day in the sequence. We refer the reader to [9] for a complete description and to [1] where MCRP was most recently applied for rainfall derivatives.

MCRP is a popular approach, at the same time is also very simplistic. MCRP is heavily reliant on past information being reflective of the future, by taking the past average to be the major contribution to future rainfall. Thus, producing weak predictive models as the annual deviations in rainfall are not explicitly captured. Moreover, the model for each city needs to be specifically tuned as each exhibits different statistical properties, i.e. a new model for each city.

Due to the disadvantages highlighted above, the literature has diverted away from the dominance of statistical methods, with machine learning methods becoming more popular over recent years. Typical applications within machine learning revolve around short term predictions (e.g. rainfall-runoff models up to a few hours [4] or monthly amounts [10] [7]). For daily predictions, [8] used a feed-forward back-propagation neural network for daily rainfall prediction in Sri Lanka, which was inspired by the chain-dependent approach from statistics. [5] applied Genetic Programming (GP) to daily rainfall data, but the GP performed poorly by itself, although when assisted by wavelets the predictive accuracy did improve. In the context of rainfall derivatives there exists only one application of GP [2], which showed that GP statistically outperformed the most common approach (MCRP). However, this implementation was a standard GP with a few modifications to tailor towards the problem. Despite few applications in rainfall prediction, GP is very popular and has been applied successfully in many different regression based problems. Therefore, we continue using GP, but due to the regression problems with extreme and high variability, we will first decompose rainfall.

In this paper, the goal is to produce more accurate rainfall amount predictions, so that we avoid mispricing, which as we have mentioned is the second step. Thus, we propose a new methodology using a hybrid GP/GA for predicting rainfall amounts using a divide and conquer strategy by decomposing the problem. The motivation allows us to break the problem of rainfall prediction into smaller partitions. As a consequence, this reduces the difficulty when dealing with data sets with high volatility and extreme values. Therefore, rather than attempting to tackle the problem as a whole, the GP/GA will evolve multiple equations. Thus, each regression equation is tailored to a specific partition of the data, rather than one size fits all. We construct our algorithm to be suitable within the final application of rainfall derivatives, but could be easily adapted to suit other application fields as well.

Due to breaking the prediction process down into partitions, we require a classification technique to help assist predicting in the correct class. For this paper we propose to extend the GP individual representation and operators (based on trees) with a GA-like linear representation and operators that first simplify the regression problem, by restricting the range of values to be predicted to one of three classes (low, medium, high), in order to facilitate the final prediction to be performed by the GP component of the hybrid GP/GA. Thus, by combining GP/GA throughout the evolution process, we aim to increase the overall predictive performance.

Hence, our main contribution is the development of a specialised algorithm for the prediction of rainfall, based on a new hybrid combination of GP and GA. In order to compare the overall predictive performance of our approach against the current approaches of GP and MCRP.

The remainder of this paper is organised as follows. Section 2 discusses the setup of the data including the data sets that will be used. Section 3 describes the GP used for rainfall prediction. Section 4 examines the criteria for partitioning the data for GP to handle (i.e. the decomposition of rainfall amounts). Section 5 presents the GA component of the hybrid GP/GA. Section 6 will discuss the techniques required for integration of GP and GA component. Section 7 discusses the experimental setup, and Section 8 will discusses the results from decomposition. Finally, Section 9 will conclude findings and suggest future research.

2. DATA SETUP

The daily rainfall data used is summarised in Table 1, which includes a total of 21 cities from around Europe. We use the same cities that were implemented in [2] and also initialise each data set into the most recent 10 years for training and 1 year for testing.

Following the same procedure in [2], we transform the daily rainfall data into accumulated amounts for a particular contract length[4], given by:

$$r_{t_s} = \sum_{t=t_s}^{t_e} r_t, \tag{1}$$

where, r_{t_s} is the accumulated amount of rainfall for a given day over a contract period from t_s till t_e. The effects of this transformation are shown in Figure 1. Thus, this allows us to determine more easily patterns that exist within the data.

via cash and the underlying asset; this is because one can not store, hold or trade weather variables.

[3]Our interests are in the approaches used within the rainfall derivatives literature, instead of methods used for different problem domains such as rainfall-runoff.

[4]The contract length refers to the period coverage of the derivative one is looking to price, e.g. March's contract length would be 31 days, whereas, April would be 30 days

Table 1: The list of all cities whose daily rainfall amounts will be used for experiments.

Cities used for daily rainfall			
Amsterdam	(Netherlands)	Ljubljana	(Slovenia)
Arkona	(Germany)	Luxembourg	(Luxembourg)
Basel	(Switzerland)	Marseille	(France)
Bilbao	(Spain)	Oberstdorf	(Germany)
Bourges	(France)	Paris	(France)
Caceres	(Spain)	Perpignan	(France)
Castricum	(Netherlands)	Potsdam	(Germany)
De Kooy	(Netherlands)	Regensburg	(Germany)
Delft	(Netherlands)	Santiago	(Portugal)
Gorlitz	(Germany)	Strijen	(Netherlands)
Hamburg	(Germany)		

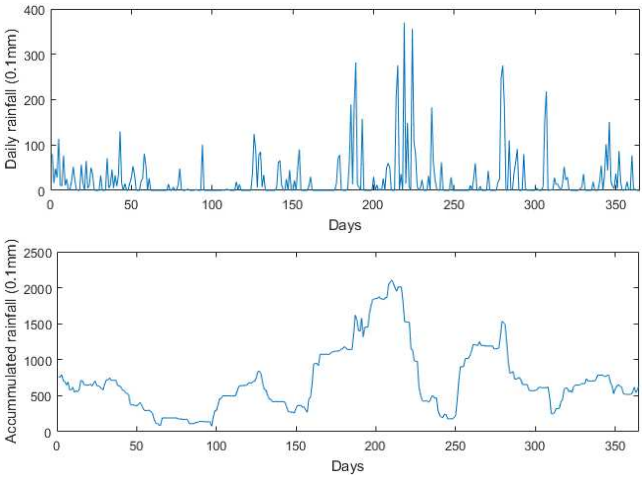

Figure 1: The effect of transforming Luxembourg rainfall data using a sliding window for the year 2014.

3. GENETIC PROGRAMMING

Each individual of the hybrid GP/GA consists of a GP-like part and a GA-like part. The GP-like part consists of 3 expression trees, where nodes represent functions or terminals as usual in GP. The GA-like part consists of a linear chromosome with a string of n rules, each with 5 genes. n can either be specified by the user, or can be randomly generated during the evolution period. We define our n as 12, 5 genes for each month. In this section we describe the GP-like part of the individual representation, which is based on a Strongly-Typed GP (STGP) with modifications used in [2] for the problem of rainfall prediction. Hereafter we use the terms GP and GA, for short, to refer to the GP and GA components of the hybrid GP/GA.

3.1 Terminals

The variables are defined by the r_t's and r_y's calculated based on the data from Section 2, where r_t is the accumulated rainfall amount in the last known non overlapping sliding window t periods ago. Similarly, r_y is the accumulated rainfall amount in the current sliding window y years ago. The attributes for a single instance are shown in Figure 2.

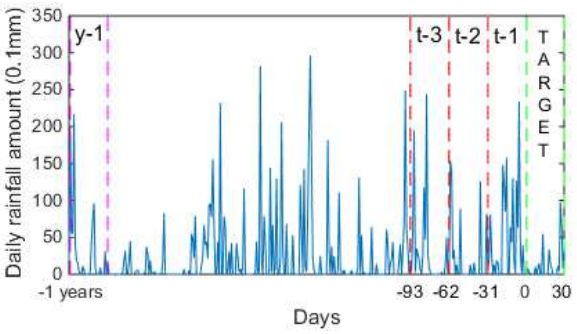

Figure 2: The sliding window value with the targets day amount with its respective t's and y's. The daily rainfall amounts within each boundary would be accumulated.

The second element is an ephemeral random constant (ERC), which will pick a uniformly distributed random number. The third element is a set of constants from -4 to 4, at 0.25 intervals, which will take a separate type from the terminals already discussed. These are constants that are specific to the power function. Due to using STGP, we can ensure that the second argument of the power function is always one of these constants and does not create an illegal tree.

Table 2: GP function and terminal sets.

Set	Value
Functions	ADD, SUB, MUL, DIV, POW, SQRT, LOG
Terminals	11 r_t periods $\{r_{t-1}, r_{t-2}, \ldots, r_{t-11}\}$, 10 r_y periods $\{r_{y-1}, r_{y-2}, \ldots, r_{y-10}\}$, ERC, Constants in the range [-4,4]

3.2 Function set

The function set includes: Add (ADD), Subtract (SUB), Multiply (MUL), Divide (DIV), power (POW), square root (SQRT), and log (LOG). The functions LOG, SQRT and DIV are protected. Additionally, the second argument for POW will be a constant in a specified range as mentioned in 3.1. Since we allow for fractional powers, we force a whole number for the second argument, if the first argument is negative.

3.3 Management of Trees

Due to rainfall being a strictly non-negative data set, a wrapper around each individual is included to modify the prediction to zero if the tree evaluates to a negative amount. The final adjustment is to ensure a balance between variables and random numbers in an individual. Thus, the first child of each node is either a function or a variable. Whereas, the second child of each node can be a variable, ERC or a function. We initialise the population using ramped-half-and-half method.

3.4 Fitness Function

The fitness used for evaluation will be the root mean squared error, given by:

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(r_t - \bar{r}_t)^2}, \quad (2)$$

where N is the length of the training set, r_t represents the predicted rainfall amount and $\bar{r}_t$ represents the actual rainfall amount for the t^{th} data point (time index).

4. DECOMPOSING RAINFALL AMOUNTS

In order to decompose rainfall, we propose partitioning the data into three different partitions (low, medium and high rainfall amounts), thus simplifying the prediction process. More could be considered, but we anticipate that three partitions is sufficient by analysing previous experimentation, where the low and high levels of rainfall received little coverage by a single regression equation. We discuss the process of splitting the data in Section 4.1. Then, in Section 4.2, we will discuss how GP will be adapted to create multiple regression equations for all partitions.

4.1 Splitting the data

We require a lower criterion LC and upper criterion UC to split our data into partitions as shown by Figure 3. Thus, anything below LC is considered low rainfall, anything between LC and UC is considered medium rainfall and above UC is considered high rainfall. Each GP individual will have its own LC and UC based on the accumulated rainfall amounts in the training data. For simplicity we keep these values constant over time, but it would be an area of interest to investigate how these values could change over time.

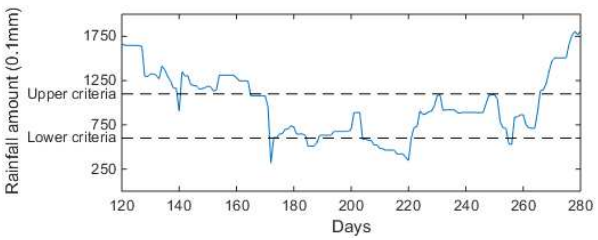

Figure 3: Luxembourg rainfall data split into three partitions according to a lower criterion and upper criterion.

4.2 Genetic Programming Trees

Based on the LC and UC, we require an independent equation that predicts the level of rainfall within each data partition. Therefore, we map each partition to a particular GP branch (b_n), shown by Figure 4. Having independent equations allows the GP to evolve each branch to maximise the predictive performance within each partition. To keep the independency between branches we create a crossover and mutation operator that can only evolve the same branch amongst individuals. The procedure is similar to the standard genetic operators, but is performed branch wise, rather than once per individual. A random node/leaf chosen from b_1 individual$_1$ can only crossover with b_1 individual$_2$ and not b_2. Similarly, the two randomly chosen individuals must be used for each of the three branches.

Elitism places an equal number of b_1, b_2 and b_3 into the next generation based on the predictive performance of each branch. In order to create the elite individual, we merge the best from b_1, b_2 and b_3, creating the best overall individual

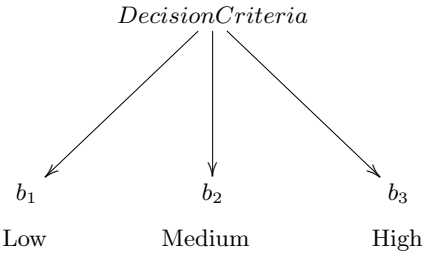

Figure 4: The representation of the decision criteria and the three branches for regression. Upon evaluation of the decision criteria, this leads to one of the three branches; each branch is a different GP tree, representing a different rainfall prediction equation.

from the previous generation. For this paper, we use b_1 to represent low rainfall, b_2 to represent medium rainfall and b_3 to represent high rainfall, as shown by Equation 3.

$$\text{Parent} \begin{cases} b_1 & \text{if } r_{t-1} \leq LC \\ b_3 & \text{if } r_{t-1} \geq UC \\ b_2 & \text{otherwise.} \end{cases} \quad (3)$$

5. THE GA COMPONENT OF THE GP/GA

In this section we outline the GA to classify each data point into the correct partition. First, we introduce the representation of our GA in Section 5.1. Then, we discuss the fitness criteria to be used in Section 5.2. Finally, the breeding of our GA in Section 5.3.

5.1 Decomposing the Problem with the GA Component

Partitioning the data (based on an individuals LC and UC) increases the complexity by having to choose which branch (GP equation) to use. We propose using a GA-like linear representation, as part of a hybrid GP/GA individual, to classify. Figure 3 shows the importance of classifying correctly, especially when considering the impact of misclassifying by more than one class. For example, if the the actual rainfall amount is within the high rainfall partition (amounts $>$ 110mm) and a classifier predicts low rainfall, then this will point to the wrong branch (tree) in the GP-part representation of the GP/GA individual, leading to an equation predicting much lower rainfall amounts, possibly in the range of less than 50mm, thus causing an error of at least 50%.

The decision criteria for a given time point consists of a rule list the size of the number of chosen partitions (for our experimentation 3), making the procedure very intuitive and very comprehensive to understand. We will be using the same attributes (r_t's and r_y's) as used within the terminal set for GP given in Table 2. We have decided not to use more complex rules for the decision criteria, as this could affect the comprehensibility of the results. Furthermore, the rules will be based on a single term and we do not consider rules involving logical operators such as AND, OR, and NOT.

The GA-part of the GP/GA individual representation consists of 5 genes; predictor, period, lower criterion, upper criterion and order. The predictor refers to one of the features from Table 2, e.g. r_{t-1}, r_{t-2} and so on. Period refers to the number of days covered by a rule e.g., a value of 31

Table 3: All the possible values for each gene, except for order. As we have a rule for each month, only the total number of days per month is given.

Genes of the GA-part of an individual	
Predictor	$\{r_{t-1}, r_{t-2} \dots r_{t-11}\}$, $\{r_{y-1}, r_{y-2} \dots r_{y-10}\}$
Period	31, 30 and 28
Lower Criteria	0.05 - 0.45
Upper Criteria	0.55 - 0.95

would cover the next 31 days. The lower and upper criteria is the decision threshold for choosing which class to predict, *predLC* and *predUC* respectively. For our experimentation we define the *predLC* and *predUC* in terms of percentiles of the training set, but this can be modified accordingly to any real number or function. The complete list (excluding order) of criteria is specified in Table 3. The order is one of the unique permutations of the three branches, given below:

Order reference

$$\begin{matrix}[\ 1\ ] & [\ 2\ ] & [\ 3\ ] & [\ 4\ ] & [\ 5\ ] & [\ 6\ ] \end{matrix}$$

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \begin{bmatrix} b_1 \\ b_3 \\ b_2 \end{bmatrix} \begin{bmatrix} b_2 \\ b_1 \\ b_3 \end{bmatrix} \begin{bmatrix} b_2 \\ b_3 \\ b_1 \end{bmatrix} \begin{bmatrix} b_3 \\ b_1 \\ b_2 \end{bmatrix} \begin{bmatrix} b_3 \\ b_2 \\ b_1 \end{bmatrix}$$

where each permutation corresponds to the following criteria:

$$\begin{bmatrix} predictor < predLC \\ predLC < predictor < predUC \\ predictor > predUC \end{bmatrix}. \tag{4}$$

For example order 3, whenever the predictor is less than *predLC* we classify medium rainfall (b_2). If greater than *predUC* we classify high rainfall (b_3), otherwise low rainfall (b_1). For order 5, whenever the predictor is less than *predLC* we classify high rainfall (b_3), if greater than *predUC* we classify medium rainfall (b_2), otherwise low rainfall (b_1).

Due to rainfall features exhibiting very complex and chaotic processes, it is highly unlikely that a single predictor can classify accurately. Such low probability in classification motivates us to allow larger number of rules to be created throughout the year, which is able to simplify complexity in rainfall, hence the period criteria. To best describe the characteristics of each month throughout each year, we set 12 rules, one for each corresponding month. However, the number of rules can be adjusted according to user's or models preferences. Furthermore, the order *predLC* and *predUC* is an important aspect within the classification process, because the same predictor could be used in a different month under different criteria. Figure 5, shows a sample representation of the above description, where we demonstrate the rules for January, February and December.

$$\overbrace{[r_{t-1},\ 31,\ 37,\ 91,\ 2}^{\text{January}}, \overbrace{r_{y-3},\ 28,\ 22,\ 77,\ 2}^{\text{February}} \dots \overbrace{r_{t-1},\ 31,\ 11,\ 64,\ 6]}^{\text{December}}$$

Figure 5: An example of a GA for 3 out of 12 months

The classification rules for January, February and December is shown in Equation 5, Equation 6 and Equation 7 respectively, showing the impact of a different order (by cross-referencing Equation 5.1 with Figure 5) and the different criteria to split the predictor. The period refers to the number of days the rules cover and is expressed in each equation as the days covered during a year. Therefore, the rules shown below are the same for every day in the respective months.

$$\text{January (Days 1-31)} \begin{cases} b_1 & \text{if } r_{t-1} \leq 37^{th} \text{percentile} \\ b_2 & \text{if } r_{t-1} \geq 91^{st} \text{percentile} \\ b_3 & \text{otherwise,} \end{cases} \tag{5}$$

$$\text{February (Days 32-60)} \begin{cases} b_1 & \text{if } r_{y-3} \leq 28^{th} \text{percentile} \\ b_2 & \text{if } r_{y-3} \geq 77^{st} \text{percentile} \\ b_3 & \text{otherwise,} \end{cases} \tag{6}$$

$$\text{December (Days 335-365)} \begin{cases} b_3 & \text{if } r_{t-1} \leq 11^{th} \text{percentile} \\ b_1 & \text{if } r_{t-1} \geq 64^{th} \text{percentile} \\ b_2 & \text{otherwise,} \end{cases} \tag{7}$$

5.2 Fitness Criteria

Each individual of the hybrid GP/GA will have the output of its GP component (which is partly determined by the values of the GA-component genes) evaluated using RMSE as outlined in Section 3.4. However, we also need to compute the fitness of the GA-part of an individual separately, for the purpose of operators like elitism (explained later). To compute that GA-part's fitness we use Kendall's tau correlation coefficient, which is used to measure the rank correlation between two variables taking into account the natural ordering of nominal classes. Kendall's tau is given by:

$$\tau_B = \frac{n_c - n_d}{\sqrt{(n_0 - n_1)(n_0 - n_2)}}, \text{where}$$

$$n_0 = \frac{n(n-1)}{2}, n_1 = \sum_i \frac{t_i(t_i-1)}{2}, n_2 = \sum_j \frac{u_j(u_j-1)}{2},$$

where n_c = Number of concordant pairs, n_d = Number of discordant pairs. t_i = Number of tied values in the i^th group of ties for the first quantity and u_j = Number of tied values in the j^{th} group of ties for the second quantity. Let $(p_1, a_1), (p_2, a_2), \dots, (p_n, a_n)$ be a set of observations, in our case the predicted class and the actual class, where n refers to the number of training instances. A pair is concordant if the ranks for (p_i, a_i) and (p_j, a_j) both agree, such that $p_i > p_j$ and $a_i > a_j$ or $p_i < p_j$ and $a_i < a_j$ and vice versa if discordant.

5.3 Evaluating and Breeding of Genetic Algorithm

The GA will be evaluated based on the Kendall's correlation mentioned above, which will return a value in the range of $[-1, 1]$. For our experimentation a value of 1 represents a perfect agreement between rankings. Once the population has been evaluated, selected individuals undergo genetic operations. The GA-part of the individuals can undergo point mutation and uniform crossover. The mutation procedure will choose a random point within the individual and replace it with a random variable or value that is of the same type. Therefore, one can not replace a predictor (r_{t-4}) with *predLC*, only with another predictor (r_{y-5}).

Uniform crossover is applied separately to each of the 12 rules (corresponding to 12 months) encoded in the parent individuals, so that for each month, each of the 5 genes has its value swapped between parents with a fixed probability (0.5). We choose each parent based on the GA-parts performance using tournament selection. We will cover the process of elitism in the next section, because it requires the interaction between the GP and GA components of the hybrid GP/GA.

6. INTEGRATING THE GP AND GA COMPONENTS

In this section we outline three aspects of the integration of the GP-part and GA-part of the individual representation of the hybrid GP/GA, namely: penalizing the regression trees, elitism, and the evolution of the LC and UC criteria to partition the data for classification.

6.1 Penalising GP Regression Trees

Following the decomposition approach, it is key that each regression equation (a GP tree) predicts values within its respective partition. For example, it makes little sense for an equation responsible for low rainfall class, predicting values in medium and high rainfall class. Therefore, we implement a penalty function based on the distance away from the correct partition, as shown in Figure 6. To integrate the GP and GA components (hereafter GP and GA for short) and together and maximise the usefulness of this idea, we implement a simple check before choosing whether to penalise or not. The GP-related penalty will only apply to situations where the GA has correctly classified. Therefore, we are not penalising GP for making a wrong prediction given that the GA was at fault. This modification should influence GP to predict within a range similar to that of the specified partition. From Figure 6 any deviation denoted by the dashed vertical lines is penalised by Equations 8.

$$
\begin{aligned}
&\text{Actual class is low}\\
p^{new} &= \begin{cases} p^{old} + m(p^{old} - LC) & \text{if } \; c_p = c_a \,\&\, p^{old} > LC \\ 0 & \text{otherwise.} \end{cases}\\
&\text{Actual class is medium}\\
p^{new} &= \begin{cases} p^{old} + m(p^{old} - UC) & \text{if } \; c_p = c_a \,\&\, p^{old} > UC \\ p^{old} + m(p^{old} - LC) & \text{if } \; c_p = c_a \,\&\, p^{old} < LC \\ 0 & \text{otherwise.} \end{cases}\\
&\text{Actual class is high}\\
p^{new} &= \begin{cases} p^{old} + m(p^{old} - UC) & \text{if } \; c_p = c_a \,\&\, p^{old} > UC \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
\tag{8}
$$

Where p^{new} represents the predicted rainfall amount by GP after penalising and p^{old} represents the predicted rainfall amount originally predicted by GP. m represents a scaling function on the penalty, c_p is the predicted class and c_a is the actual class (i.e. the classified rainfall amount). UC and LC are the upper and lower criteria for splitting the data into its respective classes. For example, let us assume that $c_p = c_a$, if GP predicted 1000 tenths of mm (p^{old}), where the UC is 1100 and m was 2, but the true class is high rainfall. We would then update p^{new} by $1000 + 2 \times (1100 - 1000)$, hence p^{new} is penalised to 800. The idea is for GP to deter from this individual given the large penalty effect.

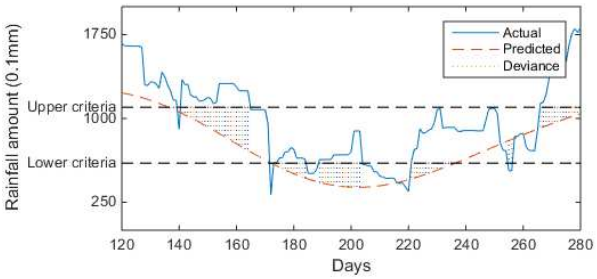

Figure 6: The distance from the predicted amount to either the lower bound or upper bound when GP predicts a rainfall amount in the wrong partition. The deviance is then used to calculate a penalty.

6.2 Elitism Merging Different Individuals

The use of elitism in our evolution process relies on exchanging information to create the best individual to put into the next generation. Typically, elitism would take the best GP trees and GA genes separately and put them into the next generation. However, due to the close integration between the GP and GA components of an individual, we create our own elitism strategy.

The first consideration was mentioned in Section 4.2, where we merge the best performing branches b^{rank}_{number} together in ranking order. Thus, the best branch b^1_1 will merge with b^1_2 and b^1_3. We do not allow a random pairing where b^1_1 could merge with b^2_2 or b^4_3. However, the GA component is jointly responsible for achieving a better RMSE. For instance, b^1_1, b^1_2 and b^1_3 may not come from the same parent using the same GA-based partition rules. Potentially, we may have 3 different GA-based rule lists influencing the performance. Thus, we need an intermediate step to decide which of the GA-based rule lists is responsible for the best overall individual using all 3 branches. Therefore, we evaluate in turn each GA-based rule list associated with the best branches merged together. We also evaluate the best overall performing partition rule list, which may not be attached to any branch. After re-evaluating the newly merged offspring, the partition rule list that was responsible for returning the best fitness in RMSE is moved into the next generation as part of the offspring.

This helps evolve the partition rules that can perform the best classification across the training period, helping the GP to solve the regression problem.

6.3 Evolution of LC and UC

The last aspect of the hybrid GP/GA is the process of evolving LC and UC (our decomposition approach). This criteria is required for the GP component to construct regression equations (trees) to predict within each data partition and for the GA-based rule lists to classify into the relevant classes. Recall that each individual consists of 3 GP regression trees and a GA-based rule list including LC and UC values.

LC and UC allows us to focus on the whole problem, which is what we are most interested in, but does not guarantee that the GA with the highest correlation is moved into the next generation, likewise the GP branch that can return the lowest RMSE. Meaning, that our evolution process is based on finding an equilibrium that can best split

our data into its respective partitions that minimises the overall RMSE. Unlike the previous two aspects, this aspect is more subtle and directly affects the performance of both GP and GA, and helps guide the evolution process of both in turn.

To ensure the split points for decomposition are evolved, during crossover the two parents' LC and UC values undergo uniform crossover to create the future offspring.

7. EXPERIMENTAL SETUP

The purpose of our experiments is to compare our decomposition algorithm (GP/GA) against a previously implemented GP [2], the most commonly used method in the literature (MCRP) and Radial Basis Function (RBF). We choose to use RBF, because it is a popular algorithm for regression problems and has already been applied to the rainfall problem [10]. We used a package called iRace [6] to find the optimal parameters for GP/GA and RBF (using the training set only), presented in Table 4. Additionally, we use the same parameter settings as [2] for GP.

Table 4: The optimal configuration of GP/GA and RBF found by iRace and the parameters of GP used in [2]. Parameters with a * represent parameters used by both the GP-part and GA-part of GP/GA.

GP Parameters	GP [2]	GP/GA
Max depth of tree	8	8
Population size	1400	1000*
Crossover	76%	99%*
Mutation	69%	30%*
Primitive	55%	32%
Terminal/Node bias	20%	64%
Elitism	3%	3%*
Number of generations	30	70*
ERC negative low	-495.36	-288.42
ERC negative high	-102.56	-224.31
ERC positive low	100.77	210.43
ERC positive high	438.58	432.23
RBF Parameters		
Minimum standard deviation		25.3373
Number of clusters		2
Ridge		3.2443

We will run the proposed GP/GA and compare its predictive performance against GP [2] and MCRP, on all 21 data sets specified in Table 1. We compare against GP (first benchmark), since it is the only one in the rainfall derivatives literature to outperform MCRP (second benchmark), which is the most popular and successful approach for the prediction of rainfall within rainfall derivatives.

We train GP/GA from 01/Jan/2004 - 31/Dec/2013 before testing on the unseen test set (01/Jan/2014 - 31/Dec/2014). As GP and RBF are stochastic algorithms, we run each for 50 times on each city (with parameters optimized by iRace), and report the average predictive accuracy on the test set over those 50 runs. For completeness we will include the average performance of MCRP over 10,000 runs.

8. RESULTS

The performance of GP/GA, GP, MCRP, and RBF is presented in Table 5 based on the average RMSE performance from the testing set for each city.

Table 5: The average RMSE performance in tenths of mm for each of our approaches across each city. The best performance is shown in bold.

Data	GP [2]	GP/GA	MCRP	RBF
Amsterdam	412.49	391.62	**373.42**	374.87
Arkona	310.45	280.81	290.72	**272.11**
Basel	425.13	412.68	467.23	**397.40**
Bilbao	519.67	**470.13**	659.03	474.69
Bourges	375.46	**321.74**	382.62	363.11
Caceres	438.41	403.11	687.74	**390.75**
Castricum	438.88	**390.34**	465.77	416.51
De Kooy	343.66	328.85	358.27	**297.46**
Delft	404.47	383.33	449.82	**344.64**
Gorlitz	304.38	279.29	304.92	**254.08**
Hamburg	408.25	389.95	409.56	**376.89**
Ljubljana	1027.62	**901.20**	1058.86	962.41
Luxembourg	517.34	**458.46**	585.23	461.39
Marseille	505.93	**471.15**	956.35	483.05
Oberstdorf	677.36	640.43	671.99	**538.22**
Paris	277.79	**271.93**	280.4	283.64
Perpignan	760.73	731.80	968.74	**709.39**
Potsdam	320.66	302.11	327.98	**265.37**
Regensburg	330.33	314.51	331.07	**296.13**
Santiago	1062.60	**975.67**	1085.1	1071.90
Strijen	298.62	293.43	343.21	**278.94**

GP/GA achieved the lowest RMSE over all data sets compared to GP and against MCRP except for one. Despite being outperformed by RBF in terms of victories, this is a very good result given we comprehensively beat the GP from [2], and also MCRP. Moreover, we perform similarly to a state-of-the-art algorithm RBF. From the results the percentage gains in lower RMSE from GP/GA over GP ranges from 2% to 14%, with the most notable gains including Bourges (15%), Ljubljana (12%), Luxembourg (11%), Castricum (11%) and Bilbao (10%).

To check which approach performed better, we compute the mean rank based on Table 5 — the lower the rank, the better the performance. Furthermore, in order to determine whether the above results are statistically significant, we compare the four approaches by using the Friedman test [3]. The Friedman test is a nonparametric test for testing the difference in mean rank between between multiple related samples. The null hypothesis is that there is no significant difference between the mean rank of the four approaches. We apply the test at the 5% significance level.

Table 6 shows the mean rank of the four approaches, RBF, GP/GA, GP and MCRP, with values of 1.57, 1.67, 3.05 and 3.71 respectively, where a lower rank indicates better performance. Therefore, across all cities on average GP/GA outperformed GP and MCRP and performed similarly to RBF, with RBF just ranking marginally higher. As we can observe, the Friedman test result has a p value of $6.06\text{x}10^{-8}$, which is less than the 5% significance level. Therefore, there is strong evidence to reject the null hypothesis, and con-

Table 6: The mean rankings of the four approaches, and the Friedman test statistic

Approach	Ranking
RBF	**1.57**
GP/GA	1.67
GP	3.05
MCRP	3.71
Friedman p-value	**6.06x10^{-8}**

clude that there is a statistical difference between the three approaches.

We perform the Holm post-hoc test, due to there being a statistical difference. The pairwise comparison results show that, GP/GA does statistically outperform GP and MCRP, with a p value of 5.2794x10^{-4} and 2.7550x10^{-7} respectively which is less than the Holm score of 0.0167 and 0.01. Therefore, these results shows that decomposing the problem of rainfall is a beneficial approach. Moreover, GP/GA performs *as well as* RBF, a highly regarded regression algorithm, by there being no statistical difference, with a p value of 0.81.

From the above results, we can conclude that the use of decomposition is highly beneficial for predicting rainfall in the context of weather derivatives by three key results.

1. GP/GA outperforms the currently popular approach of MCRP.
2. GP/GA outperforms a standard GP, because of its decomposition abilities
3. GP/GA performs *as well as* RBF, a state-of-the-art regression algorithm.

These results will help with the second stage of the rainfall derivatives problem of pricing, by reducing potential mispricing.

9. CONCLUSIONS

This paper introduces a novel algorithm for dealing with complex data sets, which exhibit extreme values and volatility. Our novelty is the proposed use of decomposition on the problem of rainfall, where it is critical to predict the accumulated rainfall amounts for such applications as financial securities (e.g. rainfall derivatives). We developed a hybrid Genetic Programming (GP) and Genetic Algorithm (GA) method in order to predict the accumulated rainfall amounts. The motivation for this paper is to make the process of rainfall prediction a simpler problem space. Therefore, we decompose the overall problem of rainfall prediction into a set of partitions, where the GP component of the hybrid GP/GA can create multiple regression equations (trees) to predict each partition more accurately. By implementing the GP proposed in [2], we extended it with the proposed GA component for classification (before performing regression). A classification technique is required, because we need to determine which regression equation to evaluate.

We evaluated the performance on cities from around Europe using our proposed decomposition algorithm, and find sufficient evidence that the proposed algorithm has a superior predictive power than a stand alone GP. Thus, we can show that our decomposition algorithm is a significant contribution by the consistent gains in observed predictive power. Additionally, we also perform *as well as* a state-of-the-art regression algorithm Radial Basis Function.

Future work will include testing other regression and classification techniques, also to extend the GA component proposed in this paper. Potential extensions could include, variable lengths of coverage per rule, more complex rule creations and a more dynamic approach for specifying the partitions through time. Finally, since we have achieved significantly better results than a stand alone GP and MCRP, this will improve the pricing rainfall derivatives.

10. REFERENCES

[1] B. L. Cabrera, M. Odening, and M. Ritter. Pricing Rainfall Derivatives at the CME. *SFB 649 Discussion Papers*, Jan. 2013.

[2] S. Cramer, M. Kampouridis, A. A. Freitas, and A. Alexandridis. Predicting rainfall in the context of rainfall derivatives using genetic programming. In *Computational Intelligence for Financial Engineering and Economics, 2015 IEEE Symposium Series on*, pages 711–718, Dec 2015.

[3] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, Dec. 2006.

[4] N. Q. Hung, M. S. Babel, S. Weesakul, and N. K. Tripathi. An artificial neural network model for rainfall forecasting in bangkok, thailand. *Hydrology and Earth System Sciences*, 13(8):1413–1425, 2009.

[5] O. Kisi and J. Shiri. Precipitation forecasting using wavelet-genetic programming and wavelet-neuro-fuzzy conjunction models. *Water Resources Management*, 25(13):3135–3152, 2011.

[6] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The Rpackageirace package, iterated race for automatic algorithm configuration. Technical report, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.

[7] Mislan, Haviluddin, S. Hardwinarto, Sumaryono, and M. Aipassa. Rainfall monthly prediction based on artificial neural network: A case study in tenggarong station, east kalimantan - indonesia. *Procedia Computer Science*, 59:142 – 151, 2015. International Conference on Computer Science and Computational Intelligence (ICCSCI 2015).

[8] H. Weerasinghe, H. Premaratne, and D. Sonnadara. Performance of neural networks in forecasting daily precipitation using multiple sources. *Journal of the National Science Foundation of Sri Lanka*, 38(3), 2010.

[9] D. S. Wilks. Multisite generalization of a daily stochastic precipitation generation model. *Journal of Hydrology*, 210:178–191, Sept. 1998.

[10] J. Wu, J. Long, and M. Liu. Evolving {RBF} neural networks for rainfall prediction using hybrid particle swarm optimization and genetic algorithm. *Neurocomputing*, 148:136 – 142, 2015.