July 1, 2017

Optimizing the Decomposition of Time Series using Evolutionary Algorithms: Soil Moisture Analytics

Aniruddha Basak
Ole J Mengshoel
Chinmay Kulkarni
Kevin Schmidt
Prathi Shastry, et al.

Optimizing the Decomposition of Time Series using Evolutionary Algorithms: Soil Moisture Analytics

Aniruddha Basak
Carnegie Mellon University
abasak@cmu.edu

Ole J. Mengshoel
Carnegie Mellon University
ole.mengshoel@sv.cmu.edu

Chinmay Kulkarni
Carnegie Mellon University
chinmaymkulkarni@gmail.com

Kevin Schmidt
U. S. Geological Survey
kschmidt@usgs.gov

Prathi Shastry
Carnegie Mellon University
prathi.shastry@sv.cmu.edu

Rao Rapeta
Intel Corporation
rao.j.rapeta@intel.com

ABSTRACT

Soil moisture plays a crucial part in earth science, with impact on agriculture, ecology, hydrology, landslides, and water resources. Extremes in soil moisture, which we denote as peaks and valleys, caused by heavy rainfalls and subsequent dry weather, are very important when predicting future soil moisture or even landslides. Existing methods, like moving averages, have limitations when it comes to smoothing time series data while preserving peaks and valleys. In this work, we propose a novel method, HyperSTL, for extrema-preserving smoothing of soil moisture time series. The method optimizes an existing time series decomposition technique, Seasonal Decomposition of Time Series by Loess (STL). HyperSTL optimizes STL's control parameters, which we call hyperparameters, using an objective function over the decomposed components. We demonstrate in experiments with nine soil moisture datasets that using HyperSTL generally results in improved predictions compared to using other smoothing methods.

CCS CONCEPTS

•Mathematics of computing → Time series analysis; Evolutionary algorithms; •Applied computing → Earth and atmospheric sciences;

KEYWORDS

Time Series Decomposition; Smoothing; STL; Stochastic Optimization; Soil Moisture; Smoothing Spline

ACM Reference format:
Aniruddha Basak, Ole J. Mengshoel, Chinmay Kulkarni, Kevin Schmidt, Prathi Shastry, and Rao Rapeta. 2017. Optimizing the Decomposition of Time Series using Evolutionary Algorithms: Soil Moisture Analytics . In *Proceedings of the Genetic and Evolutionary Computation Conference 2017, Berlin, Germany, July 15–19, 2017 (GECCO '17),* 8 pages.
DOI: 10.1145/3071178.3071191

GECCO '17, July 15-19, 2017, Berlin, Germany

ACM ISBN 978-1-4503-4920-8/17/07…$15.00
http://dx.doi.org/10.1145/3071178.3071191

1 INTRODUCTION

Context. Soil moisture plays an essential role in agriculture, ecosystem health, and disasters such as flash flooding and landslides. As an example, we consider post-wildfire erosion monitoring in Figure 1. The figure shows three soil moisture measurements at different depths and one rainfall time series. As a metric for soil moisture, we employ near-surface Volumetric Water Content (VWC). The task of measuring VWC, using currently available dielectric sensors, is made challenging due to (i) the presence of (short-term) instrumental variability, (ii) temperature-driven diurnal variations, and (iii) "noise" in the recorded time series. We aim to smooth out these three types of variability, while keeping the extrema intact.

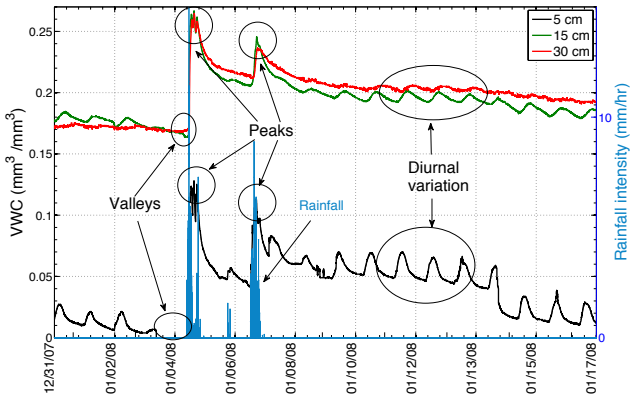


Figure 1: Volumetric Water Content (VWC) form three depths (5, 15 and 30 cm) and rainfall time series data.

Problem. The problem we study can be denoted as *extrema-preserving smoothing* of a time series. Specifically, we want to smooth time series data in order to aid forecasting, with emphasis on maintaining maxima (with surrounding peaks) and minima (with surrounding valleys), see Figure 1.

In the case of soil moisture, we want to fit the long-term (weeks or months) variations and maintain the shapes and magnitudes of extrema. Diurnal variations, caused by the Earth's rotation around the Sun, might skew a model to fit uninteresting parts of a signal. Local maxima (peaks) in soil moisture, typically caused by rainfall, are critical for forecasting future soil moisture, landslides, and runoff-driven erosion and flooding. Local minima (valleys) in soil moisture, on the other hand, typically represent maximally dry

conditions before a rainfall. Such minima have strong implications in agriculture and maintenance of ecosystem.

Many traditional signal smoothing techniques such as moving averages, splines, and LOESS[1] smoothing exist and may appear suitable to solve our extrema-preserving smoothing problem. Unfortunately, when applied to our soil moisture datasets, these traditional smoothing techniques often diminish local maxima ("reduce peaks"), introduce time shifts, or create other problematic artifacts (Figure 2). Landslides occur at high soil moisture levels, and thus we do not want to drastically smooth out or displace such maxima.

Contribution. In order to perform extrema-preserving smoothing of soil moisture time series, we develop a novel method, HyperSTL, for optimizing the decomposition of time series. HyperSTL is a wrapper method for STL, a time series decomposition method based on loess [3]. We use STL to decompose a VWC time series into trend, seasonality, and remainder components. Unfortunately, STL with the default parameter setting produces similar results to traditional smoothing methods. Often, the setting of one or more STL parameters is quite complicated and calls for substantial trial and error, perhaps even statistical expertise.

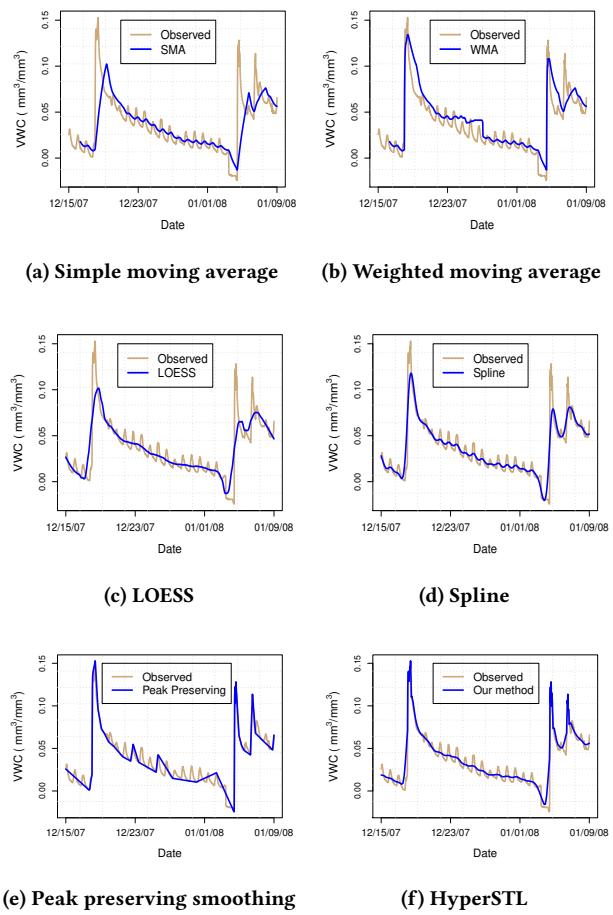
Our HyperSTL method[2] successfully reduces the types of variability identified above while preserving maxima, minima, and important signal detail (see Figure 2f), thus also improving forecasting results. The main contributions of this paper are as follows:

- We establish the significant impact of varying the hyperparameters of STL for several soil moisture data sets. While STL was introduced in 1990, we are not aware of any other studies that highlights this issue.
- We formulate an objective function for smoothing of soil moisture time series, and study its optimization via evolutionary and other optimization algorithms in HyperSTL.
- In experiments with nine soil moisture datasets, we demonstrate the smoothing benefit of HyperSTL. We also show that our HyperSTL method typically leads to improved forecasts compared to five existing smoothing methods.

Overview. In the rest of this paper, which expands on an abstract [17], we first present related work. Then we discuss our novel extrema-preserving smoothing method HyperSTL along with a smoothing objective function. Next, we describe soil moisture datasets and show smoothing results, including a comparison between existing methods and HyperSTL. Further, we demonstrate that soil moisture forecasting results typically improve when HyperSTL is used. Finally, we conclude and outline future work.

2 RELATED WORK

Here we discuss several existing methods for smoothing of time series, and demonstrate their use on a soil moisture dataset, the Canyon 5cm dataset (see Section 4.2 for details). Due to certain limitations of these traditional methods, we turn to a more complex class of methods, namely time-series decomposition methods. These methods show great promise, but their performance relies on the setting of several hyperparameters, a topic which appears to not have been investigated in the time-series decomposition literature. Consequently, we also briefly discuss previous work on hyperparameter optimization in this section.



(a) Simple moving average (b) Weighted moving average

(c) LOESS (d) Spline

(e) Peak preserving smoothing (f) HyperSTL

Figure 2: Comparison of various existing smoothing methods on the Canyon-5 dataset. Methods (a)–(d) reduce the peak heights while (e) adds bumps in the output. Our novel smoothing algorithm, HyperSTL, is shown in (f).

Smoothing Methods. The methods we study include simple moving average (SMA), weighted moving average (WMA), and Local Polynomial Regression Fitting (LOESS) [4], spline [11], and a peak preserving smoothing algorithm [9].

SMA (Figure 2a) calculates the average of the last n measurements in a time series. WMA (Figure 2b), provides an option of having weights on individual measurements. For an input time series $\{Y_i : i \in [1, T]\}$, SMA and WMA generate smooth series as $\text{SMA}(Y_i) = \frac{1}{n}\sum_{j=0}^{n} Y_{i-j}$ and $\text{WMA}(Y_i) = \frac{1}{n}\sum_{j=0}^{n} Y_{i-j} * W_j$ respectively. We chose the weights of WMA, W_j, to be much higher in regions containing peaks and valleys. We experimented with multiple window sizes (n), ranging from 12-24 hours. This parameter was tuned to minimize the time shift introduced by the moving average algorithms. In the smoothing result of SMA (Figure 2a), the

[1]The word "loess" has two related but slightly different meanings in this paper. We use "LOESS" to denote a smoothing technique and "loess" to denote the non-parametric regression method.

[2]See GitHub https://github.com/olemengshoel/hyperstl for the R source code of HyperSTL.

peaks are shifted in time and heights are diminished. Using higher weights on peaks and valleys we tuned n to minimize time shifts of the smooth output. Although WMA preserves the peak heights better than SMA, it increases the widths, see Figure 2b.

LOESS (see Figure 2c) performs local regression over a subset of data and repeats the process for the rest of the dataset. We tuned the *span* (ranging from 30 minutes to 6 hours) of the local dataset to optimize the results. However, we observe dislocation of valleys and peaks in this case.

The smoothing spline fits a smooth curve to the noisy input, see Figure 2d. Compared to moving averages and LOESS, it produces less distortion of the signal. But, in some cases, the peaks are significantly diminished.

A peak preserving smoothing algorithm [9] identifies discontinuities in the data by comparing three linear fits and then producing a smooth estimate by applying the linear fits in between the discontinuities. Although the algorithm preserves peaks, it changes the nature of the signal by making peaks bumpy and abrupt (Figure 2e). This algorithm can also create discontinuous output [27].

In contrast to existing smoothing methods, HyperSTL (described in Section 3) smooths a time series while preserving the shapes, locations, and heights of peaks and valleys (see Figure 2f).

Time Series Decomposition Methods. As the existing smoothing algorithms (discussed above) appear to have some limitations, we consider time series decomposition methods which have a wide range of applications in time series analysis including forecasting and smoothing. Shamsudduha *et al.* [23] used STL for estimating long term trends in groundwater level. To find the appropriate decomposition, the authors experimented with different STL parameter settings and picked one from the visualization of the results. Similarly, knowledge of vegetation activity was required to choose trend and seasonality neighborhood sizes in decomposition of Normalized Difference Vegetation Index (NDVI) time series [14]. Despite many applications of time series decomposition methods including STL, to the best of our knowledge, systematic goal-driven optimization of STL's parameter has never been studied before.

Hyperparameter Optimization Methods. Due to space limitations, we refer to the literature [6, 15] and only discuss the most closely related research here. In the machine learning community, there has been research on optimizing the parameters of machine learning algorithms. In classification, for instance, hyperparameter optimization methods are often used to configure neural networks [1]. In the evolutionary algorithm community, there has been interest in optimizing the parameters of the evolutionary algorithms themselves [24], and also using evolutionary algorithms to optimize the parameters of non-evolutionary algorithms.

General-purpose parameter tuning methods, such as IRACE [18] and ParamILS [12], have also been developed. Both methods optimize the configuration of algorithm parameters, guided by a cost function. The relatively recent method IRACE implements an iterative race, where the new configurations are sampled based on previously found best configurations.

Uniqueness of Our Work. From the study of existing smoothing methods, we observe that they have many limitations. HyperSTL can perform much improved smoothing, see Figure 2. Of course, this is only one example. But HyperSTL also works well for other datasets and the smooth output of HyperSTL improves time series forecasting (see Section 4).

3 OPTIMIZATION OF TIME-SERIES DECOMPOSITION

3.1 Time Series Decomposition

Time series decomposition has been done primarily on the basis of two different factors: rate of change and predictability. The rate of change approach attempts to decompose a time series into multiple components with variations of different periodicity. The typical components of such decompositions are: trend or long term (T), seasonal (S), cyclical (C), and random or residual (R). The predictability approach decomposes a time series into deterministic and non-deterministic components.

Additive and multiplicative structures are two commonly used decomposition models. When the components are independent of each other, the additive model provides a good decomposition. In contrast, the multiplicative model is more suitable when the seasonal and residual fluctuations follow a specific pattern in relation to a trend. Multiplicative models often apply in economic time series [13].

There is often a need to optimize the hyperparameters of a decomposition algorithm. Model fitting problems are usually formulated as optimization problems:

$$\min_{\Theta} F(\Theta; Y, T, S, R); \text{ subject to } Y = T + S + C + R, \tag{1}$$

where Y is the input time series and F is a loss function (*e.g.*, mean squared error, log likelihood [19]) for additive decomposition. If F is convex, a convex optimization method can be chosen to solve this optimization problem efficiently. However, we can not make the convexity assumption here.

3.2 STL and Hyperparameters

STL Method. We now turn to a specific nonparametric decomposition method, namely Seasonal Trend Decomposition Based on Loess (STL) [3], along with its optimized application to soil moisture time series data. The inputs and outputs of STL can be defined as follows:

$$Y \overset{\text{STL}}{\underset{\Theta}{=}} T + S + R. \tag{2}$$

STL decomposes, according to the hyperparameters Θ, the input time series (Y) into three components: trend (T), seasonality (S), and remainder (R). STL enables its users to control the nature of T, S, and R, including smoothness of the trend component T. This requires carefully manipulating Θ: neighborhood (or window) size (s.window, t.window, l.window: $\theta_1, \theta_2, \theta_3$) and degree of polynomial (s.degree, t.degree, l.degree: $\theta_4, \theta_5, \theta_6$) for seasonality, trend and loess. Other parameters include robustness (boolean) of loess fitting and a few dependent variables of $\{\theta_1, \cdots, \theta_6\}$.

STL Hyperparameter Optimization. The default values of the STL hyperparameters Θ often do not provide very effective smoothing according to a scientific objective. Manual tuning of Θ can be time consuming, error prone, and the relationship to the resulting curves for T, S, and R may be unclear. Consequently, we propose to find an appropriate assignment to Θ via optimization,

formulated in terms of the decomposed components T, S, and R as discussed in Section 3.3.

3.3 HyperSTL Objective Function

To preserve the extrema of Y, we first identify regions without extrema. Since rainfall is the main driving force behind peaks in soil moisture, we also use a rainfall time series X as input. Before rain events in X, data-points in Y are valleys and after rain events in X we find peaks in Y (see Figure 1). Placing a 12 hour-window before and after contiguous rain events, we identify regions in Y with extrema. We denote the subsequence of Y consisting of all data-points *between extrema* as $Y_{\text{b.e.}}$. Similarly, $T_{\text{b.e.}}$ corresponds to a subsequence of the trend T.

As the smoothing task is to extract a smooth replica of the input Y from the trend component of STL, we formulate this task using the following objective function:

$$\min_{\theta} f(\theta; Y, T, S, R) = w_1\sqrt{Var[R]} + w_2 \text{ Range}(R) + \frac{w_3}{\sqrt{m}}||T_{\text{b.e.}} - \hat{Y}_{\text{line, b.e}}||_2. \quad (3)$$

Here, $\text{Range}(R) = |\max(R) - \min(R)|$ by definition and the subscript "b.e." corresponds to m data points between two extrema. $\hat{Y}_{\text{line, b.e.}}$ is a piece-wise linear approximation of $Y_{\text{b.e.}}$, *i.e.*, a collection of linear segments fitted between every two consecutive extrema. The weights on the three terms of f are $w_1 \in \mathbb{R}^+, w_2 \in \mathbb{R}^+$, and $w_3 \in \mathbb{R}^+$. These weights control the relative importance of the three smoothing goals. As scaling the weights does not change the optimization problem, we choose $w_1 + w_2 + w_3 = 1$.

Algorithm 1: Compute smoothing objective function

1 function $f(\theta, Y)$;
Input : STL parameters θ and time series Y
Output: $f(\theta)$
2 Find a drying region in Y:
3 $Y_{\text{b.e.}} \leftarrow$ subsequence of Y between extrema (see text);
4 $m \leftarrow$ number of elements in $Y_{\text{b.e.}}$;
5 $\hat{Y}_{\text{line, b.e}} \leftarrow$ Linear fit of $Y_{\text{b.e.}}$;
6 $(T, S, R) \leftarrow \text{STL}(Y, \theta)$; `// Equation 2`
7 $f_1 \leftarrow Var[R]$; $f_2 \leftarrow Range(R)$; $f_3 \leftarrow \frac{1}{\sqrt{m}}||T_{\text{b.e.}} - \hat{Y}_{\text{line, b.e}}||_2$;
8 return $w_1\sqrt{f_1} + w_2 f_2 + w_3 f_3$;

Analogous to regression, we attempt in the first term of (3) to minimize the variance, $Var(R)$, of the remainder R. This is to reduce the information content in R. However, a small variance solution can still have some abrupt peaks (or valleys) in R. Thus we explicitly add the range of R, $Range(R)$, as a second term in (3), to reduce the chance of peaks (or valleys) in Y being passed to R. The third term in (3) is designed to make the trend line smooth.

We note that Equation 3 is not the only possible objective function. In particular, the objective function definition may depend on the application, and reasonable alternatives are likely to exist.

Algorithm 1 implements the objective function in (3). In order to keep all terms in the scale of the data, we use the square roots for f_1 and f_3. This simplifies the choice of weights. In our experiments we use $(w_1, w_2, w_3) = (0.1, 0.3, 0.6)$, putting higher importance on making the trend curve smooth. We found that the decomposition method is robust to minor changes of the weights and one can choose other weights for different smoothing purposes.

3.4 HyperSTL Algorithm

HyperSTL, as implemented by Algorithm 2, minimizes (in line 3) $f(\theta)$ (see Algorithm 1) to obtain the most suitable hyperparameter setting of STL.

It is hard to expect convexity[3] with $Range(R)$ in $f(\theta)$. In Section 4, we will show the non-convex objective function landscape for a particular soil moisture dataset. Thus we do not make convexity assumptions and use stochastic optimization (denoted by SO) algorithms to solve (3) and obtain $\theta^*_{SO} = \arg\min f(\theta)$. Algorithm 1 is used by the SO as a sub-routine. We decompose Y using STL with θ^*_{SO} as hyperparameters. The trend line T of this decomposition will be a smoothed version of Y with the extrema preserved.

Algorithm 2: HyperSTL

Input : Time series Y
Output: Smoothed version of Y
1 Set the range of θ: $\theta^{\max} \leftarrow \{\theta_1^{\max}, \theta_2^{\max}, \theta_3^{\max}\}$, $\theta^{\min} \leftarrow \{\theta_1^{\min}, \theta_2^{\min}, \theta_3^{\min}\}$;
2 Minimize $f(\theta)$: $\theta^*_{SO} \leftarrow \text{SO}(f, \theta^{\max}, \theta^{\min})$;
3 Compute decomposition: $(T, S, R) \leftarrow \text{STL}(Y, \theta^*_{SO})$;
4 return trend T;

The processing steps of HyperSTL are: applying STL to the input time series, computing the smoothing objective based on the decomposition output, and carefully tuning the hyperparameters to optimize the smoothing objective. The output of HyperSTL is a smooth replica T of the input Y; T is useful in its own right and can be used for forecasting also.

Computing the decomposition using STL is computationally intense and grows fast with the length of the time series. Therefore, a brute force search of the entire space of STL parameters is computationally intractable for soil moisture time series spanning a few months or more.

In order to optimize STL's hyperparameters, HyperSTL introduces new parameters in the objective function (3). However, these parameters are more intuitive than STL's hyperparameters and can easily be tuned (by the user) by observing changes in the STL outputs T, S, and R for different assignments to w_1, w_2, and w_3.

4 SOIL MOISTURE DATA ANALYSIS

We now discuss the implementation and application of our HyperSTL decomposition optimization method for smoothing soil moisture datasets. Data is collected from three different locations, at three different soil depths for each location, see Table 1.

4.1 Experimental Setup

We use an R implementation of STL[4] (denoted by STL^{R}) for our experiments. All arguments of STL^{R} (or all elements in Θ) do

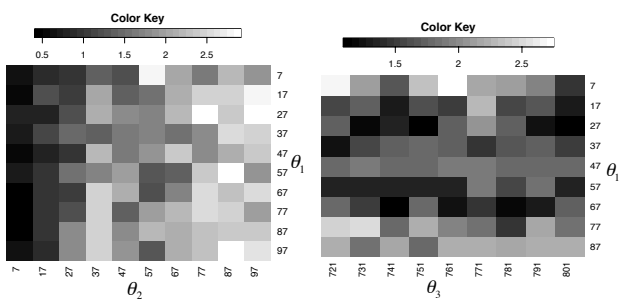
[3] We are using discrete convexity in this paper, see Figure 3b.
[4] https://stat.ethz.ch/R-manual/R-devel/library/stats/html/stl.html

Location	Depth (cm)	Dataset Id
Field: Canyon Fire	5	Canyon-5
	15	Canyon-15
	30	Canyon-30
Field: Gap Fire	5	Gap-5
	15	Gap-15
	30	Gap-30
Controlled Experiment: Bucket	10	Bucket-10
	20	Bucket-20
	28	Bucket-28

Table 1: Summary of nine soil moisture datasets.

not influence the decomposition significantly in our case. After pilot experiments with STL^{R} on our soil moisture data, we found that only the three loess window sizes (θ_1, θ_2, and θ_3 as defined in Section 3.2) have a significant impact on the resulting trend and seasonal curves. Therefore, we pick the subset $\theta = \{\theta_1, \theta_2, \theta_3\} \subset \Theta$ as the hyperparameters to be optimized.

We analyze the objective function landscape of $f(\theta)$ by varying two parameters in θ while keeping the third constant. Figures 3a and 3b show the objective function landscapes of $f(\theta)$ when `l.window` and `t.window` are kept constant at 761 and 57 respectively.[5] We observe non-convex and non-smooth landscapes in both cases. Moreover, there are multiple local peaks in both cases.



(a) $\theta_3 = 761$ **(b)** $\theta_2 = 57$

Figure 3: Heat maps of the fitness landscape of $f(\theta)$. Darker regions have better fitness values (lower $f(\theta)$). Hence, a black spot surrounded by gray areas indicate a (desirable) local minimum.

Therefore we attempt to find a suitable optimization algorithm for this non-convex problem. We compare several kinds of optimization methods: single- and multi-objective evolutionary algorithms, meta-heuristic global optimization algorithms, and statistical parameter tuning algorithms. The single-objective evolutionary algorithms include Genetic Algorithms (GAs) and Differential Evolution (DE) [25]. We use both a real-coded GA (RGA) and a binary-coded GA (BGA) to solve (3). We also include an adaptive algorithm JADE [29] in our comparison. Simulated Annealing (SA) [2], being a widely adopted probabilistic search technique, is our choice of a meta-heuristic optimizer.

[5]These numbers are chosen based on the input time series Y.

To minimize the three terms in (3) as separate objectives, we use the Non-dominated Sorting Genetic Algorithm II (NSGA II) [5]. NSGA II computes a Pareto optimal set of solutions. Then we pick the best solution from this set by taking the weighted sum of their objectives according to (w_1, w_2, w_3). Among machine learning hyperparameter tuning algorithms, we pick IRACE [18].

The implementations of the aforementioned algorithms in R packages GA [22] (RGA and BGA), DEOptim [20] (DE and JADE), GenSA [28] (Generalized Simulated Annealing), nsga2R[6] (NSGA II), and IRACE [18] are used to implement "SO" in HyperSTL (see Algorithm 2). Table 2 shows the parameter settings of the algorithms (the tournament size of NSGA II is 2). We use default parameters for IRACE.

Using random initialization, we run each algorithm 30 times independently with a per-run budget of 5000 objective function evaluations. For each HyperSTL run, we record the objective function value of the solution obtained and the wall clock time. We use 64 bit Linux machines with Intel Xeon CPU 3.2GHz processors for this experiment.

Parameter	RGA	BGA	DE & JADE	GenSA	NSGA II
Population size (n_P)	50	50	50	1	100
Num. generations (n_G)	100	100	100	5000	50
Crossover prob. (p_{CR})	0.8	0.8	0.5	N/A	0.7
Mutation prob. (p_M)	0.1	0.1	0.8	N/A	0.2

Table 2: Parameter settings for the optimization algorithms.

4.2 Field Data: Canyon and Gap Fires

Canyon Fire Data. The Canyon fire dataset is composed of soil moisture and rainfall measurements from the Santa Monica mountains following the 2007 Canyon fire. The dataset was collected by the U. S. Geological Survey (USGS) near the town of Malibu in Southern California [10, 21]. Prior to the fire, the site was covered by chaparral vegetation, but the fire removed almost all of the vegetation and changed the soil infiltration properties. After wildfires in chaparral ecosystems, increased erosion is triggered by floods and debris flows from steep, burned watersheds. Forecasting runoff conditions, as a result of rainfall, from these burned landscapes is challenging because the hydrologic response of the ground is dramatically altered by the high-temperature fire. Typically, the fire consumes the vegetation, fuses soil particles together, deposits a layer of ash, and locally forms hydrophobic layers.

Soil moisture measurements were collected using probes (Decagon Devices Inc.[7]) measuring VWC by estimating the dielectric constant of the media using capacitance or frequency domain technology [16]. These probes were placed at three different depths in the soil: 5, 15 and 30 cm. Soil moisture measurements were logged every 2 minutes. The soil moisture and rainfall data from December 2007 to April 2008 is shown in Figure 6.

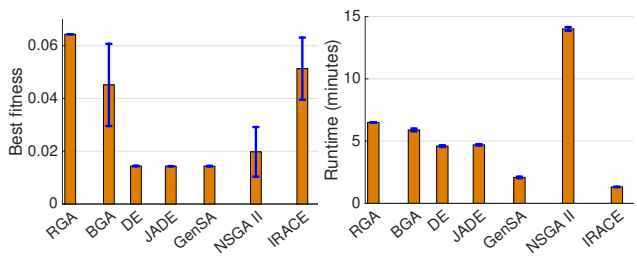
Gap Fire Data. The 2008 Gap fire in the Western Transverse range burned the urban wild-lands above Goleta and Santa Barbara

[6]http://CRAN.R-project.org/package=nsga2R
[7]http://www.decagon.com/products/soils/volumetric-water-content-sensors/

in California. In response to public safety concerns for post-fire erosion and debris flows, the USGS installed monitoring equipment to record rainfall, overland flow, soil suction, and soil moisture.

4.3 Comparison of Optimization Algorithms

In order to pick a suitable optimization algorithm to solve the problem in Equation 3, using HyperSTL, we empirically compare a few optimization methods (see Section 4.1). Using the Canyon fire dataset, we compare the optimization algorithms on the basis of best objective function value and runtime.



(a) Comparison of objective function values. **(b) Runtime comparison the optimization algorithms.**

Figure 4: Comparison of best objective function (fitness) and runtime for six optimization algorithms (RGA, BGA, DE, JADE, GenSA, and NSGA II), over 30 independent runs.

Figure 4 shows the mean and standard deviation for both performance metrics over 30 independent runs. DE, JADE, and GenSA obtained much lower mean-best fitness values than the rest. As the variance of best fitness values for DE and JADE are extremely low, these algorithms can be used to solve the optimization problem with high accuracy.

Even though optimizing the objective function is our main focus, we also compare the running times of algorithms. This is because faster methods allow the user to more easily design an appropriate objective function by inspecting the smoothing result and interactively adjusting (3). The runtime of GenSA is smaller than that of all others. Therefore, GenSA can be a good choice to obtain optimized decompositions of large-scale time series data.

Significance Test. To evaluate the statistical significance of the previous comparison, we perform a Wilcoxon rank-sum test [7] on the best fitness values obtained in 30 independent runs. We consider two algorithms to perform statistically similar if the p-value of the significance test is higher than the 5% significance level. Otherwise, if the p-value is lower than 5%, we determine superiority among the two algorithms based on their mean best fitness values.

From the above analysis, we found that DE and JADE are statistically similar and superior to the other methods, with DE being slightly faster than JADE. In all experiments involving Algorithm 2, we thus use DE in place of *SO* in the following.

Smoothing by STL Optimization. We present the smoothing results on Canyon fire data. Figure 5a shows decomposition of soil moisture data using STL optimization with default parameters (as in STL^{R}). From the figure, it can be noticed that STL with default parameters is able to decompose soil moisture data into trend and seasonal behaviors. The trend curve T is effectively a candidate for smooth version of the input. Unfortunately, the peaks are greatly reduced and significant information is passed to the remainder R. The default parameters do not lead to effective smoothing.

Consequently, we use our decomposition optimization method, and optimize STL's hyperparameters using HyperSTL (Algorithm 2).

The diurnal variations are significantly reduced in the trend curve of the optimized STL (Figure 5b), while the peaks and shapes are intact. Moreover, the amplitude of the seasonal curve quantifies the magnitude of daily variations in VWC. According to geo-scientists, this information can be useful in studying the effect of environmental factors, such as solar radiation and humidity, on soil moisture.

Figure 6 shows the three time series (Y_5, Y_{15}, and Y_{30}) from the Canyon fire dataset overlayed with the outputs (trend curves: T_5, T_{15}, and T_{30}) of Algorithm 2. All trend curves are smooth versions of the inputs and maintain the peaks and valleys in the data. However, an abrupt jump appears in 5 cm trend curve (T_5). It indicates that Algorithm 2 does not perform "perfect" smoothing. It might be close to impossible to obtain perfectly smooth curves while maintaining all extrema in the data.

4.4 Results from Controlled Experiment

To obtain another dataset to study soil moisture dynamics, we performed an experiment in a controlled environment. To imitate the post-fire scenario, we used very dry material in this experiment.

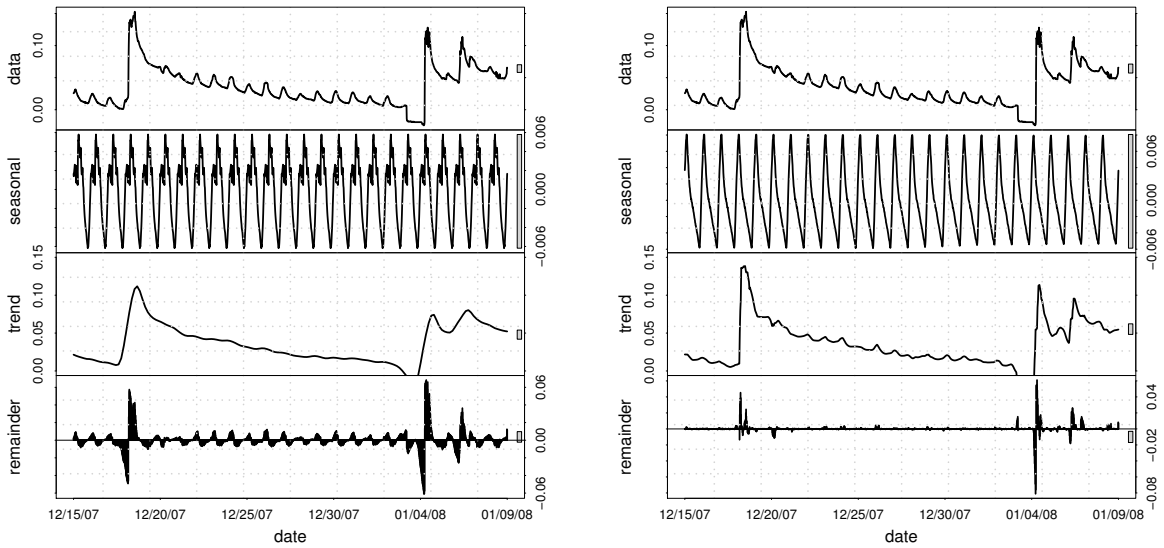
Bucket Data. We performed the experiments in an open space exposed to normal weather conditions at CMU, Moffett Field, California. We filled a 5 gallon plastic bucket with play sand (grain size < 1 mm) and placed three VWC sensors at 10, 20, and 28 cm depths. The same type of capacitance VWC sensors were used as for the field data. Then we added water to the bucket in intervals to bring the soil to near-saturated conditions. Some rainfall events during the time of the experiment also added water to the bucket.

VWC data collected during October and November of 2015 is shown in Figure 7, along with the smoothing results. The wetting and drying behavior of VWC data collected in the controlled experiments are noticeably similar to Canyon Fire field-data shown in Figure 1 with rainfall and dry periods. In particular, similar diurnal variations appear in the recorded VWC time series.

Smoothing by STL Optimization. Applying HyperSTL (Algorithm 2) to the collected time-series from the controlled experiment, we obtain the smooth versions of soil moisture measurements at three different depths as shown in Figure 7. Clearly, the diurnal variations are suppressed without the peaks being distorted. This demonstrates another success of our smoothing method and suggests that it can applied more generally.

4.5 Forecasting Results

To study the effect of different smoothing methods on forecasting, we investigate the predictions of an Antecedent Water Index (AWI) based soil moisture model [8, 26] using all nine datasets in Table 1. For each dataset, we create AWI models using (i) the original data and (ii) the data after smoothing with six different methods, giving a total of seven distinct forecasting models. We split each time series in a 2:1 ratio for training and test sets. We compare the test errors of the trained AWI models on the basis of root mean squared error (RMSE) and maximum absolute error (MAE).



(a) Default STL arguments and "s.window" = periodic.

(b) HyperSTL result: STL arguments set to θ^*_{DE} .

Figure 5: STL decomposition of 5cm soil moisture from Canyon fire field data with different parameter settings.

	Root Mean Squared Error (RMSE)							Maximum Absolute Error (MAE)						
	Original	SMA	WMA	LOESS	Spline	Peak Pres.	Hyper STL	Original	SMA	WMA	LOESS	Spline	Peak Pres.	Hyper STL
Canyon-5	0.015	0.012	0.011	0.011	0.008	0.011	**0.007**	0.086	0.083	0.083	0.061	**0.055**	0.086	0.058
Canyon-15	0.009	0.007	0.006	0.006	0.005	0.008	**0.004**	0.069	0.063	0.063	0.048	**0.044**	0.059	**0.044**
Canyon-30	0.008	0.006	0.005	0.005	0.004	0.010	**0.003**	0.052	0.050	0.050	0.033	**0.031**	0.048	**0.031**
Gap-5	0.007	0.006	0.006	0.005	**0.004**	0.010	**0.004**	0.039	0.032	0.032	0.021	0.018	0.035	**0.012**
Gap-15	0.003	0.003	0.003	0.003	0.002	0.006	**0.001**	0.024	0.016	0.016	0.019	**0.015**	0.019	**0.015**
Gap-30	0.005	0.004	0.004	0.004	0.003	0.007	**0.002**	0.039	0.031	0.031	0.022	0.018	0.040	**0.016**
Bucket-10	0.027	0.035	0.035	0.028	0.025	0.028	**0.024**	0.090	0.119	0.090	**0.076**	0.088	0.097	0.079
Bucket-20	0.009	0.009	0.009	**0.008**	**0.008**	0.009	**0.008**	0.022	0.024	0.024	0.021	**0.020**	**0.020**	**0.020**
Bucket-28	0.018	0.028	0.039	0.025	0.024	**0.014**	**0.014**	0.057	0.059	0.068	0.064	0.054	0.059	**0.053**

Table 3: Prediction errors using no smoothing (Column: Original), five smoothing methods (Columns: SMA, WMA, LOESS, Spline, Peak Preserving), and HyperSTL of trained AWI models on test data from nine different soil moisture datasets.

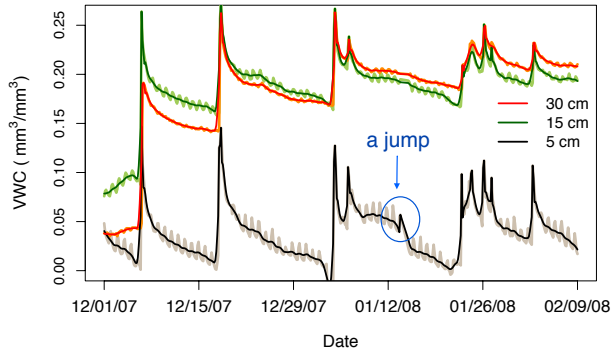


Figure 6: Outputs of HyperSTL applied to three Canyon fire soil moisture. The lighter colors describe the observed data and darker colors represent the smoothed versions.

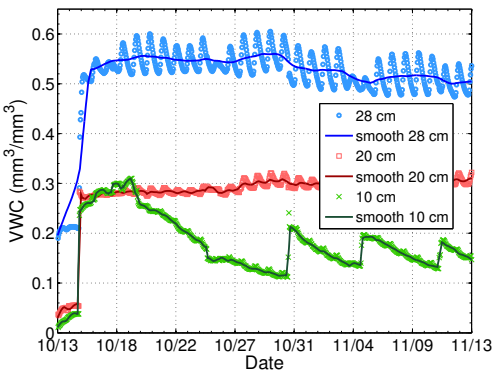


Figure 7: Outputs of HyperSTL applied to the three Bucket soil moisture time series.

Table 3 shows the results of this comparison and Figure 8 shows example predictions of trained models in the test set of the Canyon-5 time series. We see that using the original data usually produces higher error. For the Bucket time series, the original data is superior to a few smoothing methods. However, HyperSTL leads to the lowest RMSE and in most cases a low MAE. Even when the MAE is not minimal for HyperSTL, the model performs well. This comparison suggests wide applicability of HyperSTL for analyzing and forecasting time series datasets.

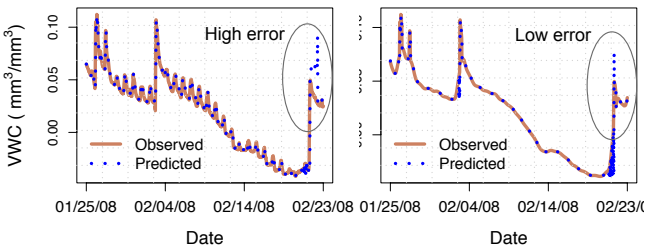


Figure 8: Prediction of two trained soil moisture models (for Canyon-5) on test set: the first uses original data (Left) and the second uses data smoothed by HyperSTL (Right).

5 CONCLUSION AND FUTURE WORK

In this paper, we study the problem of extrema-preserving smoothing for soil moisture time series. The goal is to reduce the diurnal and other short-term variations in the data without distorting the extrema (peaks and valleys). We have taken a time series decomposition approach, using the loess-based decomposition method STL [3]. We designed an objective function that evaluates the STL decomposition according to our smoothing goal. Without a convexity guarantee for the objective function, we used heuristic algorithms including evolutionary algorithms for optimization in our HyperSTL method. Applying HyperSTL to several soil moisture time series datasets, we found that our smoothing technique performed very well compared to existing smoothing methods, and also gave improved forecasting results in most cases.

In the future, we hope to apply our decomposition method to other types of time series analysis tasks. Future work could also be directed towards developing a parallel implementation of HyperSTL, for example by parallelizing the STL function itself or by computing multiple STL function calls in parallel.

REFERENCES

[1] J. Bergstra and Y. Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13, 1 (2012), 281–305.
[2] I. O. Bohachevsky, M. E. Johnson, and M. L. Stein. 1986. Generalized simulated annealing for function optimization. *Technometrics* 28, 3 (1986), 209–217.
[3] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning. 1990. STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics* 6, 1 (1990), 3–73.
[4] W. S. Cleveland, E. Grosse, and W. M. Shyu. 1991. Local regression models. Statistical Models in S (Chambers, J. M. and Hastie, T. J., eds), 309-376. (1991).
[5] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197.
[6] A. E. Eiben and S. K. Smit. 2011. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation* 1, 1 (2011), 19–31.
[7] S. García, D. Molina, M. Lozano, and F. Herrera. 2009. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. *Journal of Heuristics* 15, 6 (2009), 617–644.
[8] J. W. Godt, R. L. Baum, and A. F. Chleborad. 2006. Rainfall characteristics for shallow landsliding in Seattle, Washington, USA. *Earth Surface Processes and Landforms* 31, 1 (2006), 97–110.
[9] P. Hall and D. M. Titterington. 1992. Edge-preserving and peak-preserving smoothing. *Technometrics* 34, 4 (1992), 429–440.
[10] M. N. Hanshaw, K. M. Schmidt, D. P. Jorgensen, and J. D. Stock. 2008. By air and land: estimating post-fire debris-flow susceptibility through high-resolution radar reflectivity and tipping-bucket gage rainfall. In *AGU Fall Meeting Abstracts*.
[11] T. J. Hastie and R. J. Tibshirani. 1990. *Generalized additive models*. Vol. 43. CRC Press.
[12] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. 2009. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research* 36, 1 (2009), 267–306.
[13] R. J. Hyndman and G. Athanasopoulos. 2013. *Forecasting: principles and practice*. [Online]. Availble via www.OTexts.com/fpp.
[14] A. Jacquin, D. Sheeren, and J.-P. Lacombe. 2010. Vegetation cover degradation assessment in Madagascar savanna based on trend analysis of MODIS NDVI time series. *International Journal of Applied Earth Observation and Geoinformation* 12 (2010), S3–S10.
[15] G. Karafotias, M. Hoogendoorn, and A. E. Eiben. 2015. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation* 19, 2 (2015), 167–187.
[16] F. Kizito, C. S. Campbell, G. S. Campbell, D. R. Cobos, B. L. Teare, B. Carter, and J. W. Hopmans. 2008. Frequency, electrical conductivity and temperature analysis of a low-cost capacitance soil moisture sensor. *Journal of Hydrology* 352, 3 (2008), 367–378.
[17] C. Kulkarni, O. J. Mengshoel, A. Basak, and K. M. Schmidt. 2015. Optimizing the decomposition of soil moisture time-series data using genetic algorithms. In *AGU Fall Meeting Abstracts*.
[18] M. López-Ibánez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. 2011. *The irace package, iterated race for automatic algorithm configuration*. Technical Report. Citeseer.
[19] H. Lütkepohl. 2005. *New introduction to multiple time series analysis*. Springer Science & Business Media.
[20] K. Mullen, D. Ardia, D. L. Gil, D. Windover, and J. Cline. 2011. DEoptim: An R package for global optimization by differential evolution. *Journal of Statistical Software* 40, 6 (2011), 1–26.
[21] K. M. Schmidt, J. D. Stock, M. N. Hanshaw, and G. W. Bawden. 2008. Constraining Diffusivity and Critical Slope from Post-Fire Sediment Flux of the Day, Canyon, and Corral Fires, California. In *AGU Fall Meeting Abstracts*.
[22] Luca Scrucca. 2012. GA: a package for genetic algorithms in R. *Journal of Statistical Software* 53, 4 (2012), 1–37.
[23] M. Shamsudduha, R. E. Chandler, R. G. Taylor, and K. M. Ahmed. 2009. Recent trends in groundwater levels in a highly seasonal hydrological system: the Ganges-Brahmaputra-Meghna Delta. *Hydrology and Earth System Sciences* 13, 12 (2009), 2373–2385.
[24] S. K. Smit and A. E. Eiben. 2009. Comparing parameter tuning methods for evolutionary algorithms. In *IEEE Congress on Evolutionary Computation*. 399–406.
[25] R. Storn and K. Price. 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* 11, 4 (1997), 341–359.
[26] R. C. Wilson and G. F. Wieczorek. 1995. Rainfall thresholds for the initiation of debris flows at La Honda, California. *Environmental & Engineering Geoscience* 1, 1 (1995), 11–27.
[27] Z. Xia and P. Qiu. 2015. Jump information criterion for statistical inference in estimating discontinuous curves. *Biometrika* 102, 2 (2015), 397–408.
[28] Y. Xiang, S. Gubian, B. Suomela, and J. Hoeng. 2013. Generalized Simulated Annealing for Global Optimization: the GenSA Package. (2013).
[29] J. Zhang and A. C. Sanderson. 2009. JADE: adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation* 13, 5 (2009), 945–958.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their helpful comments and suggestions. This research has, in part, been supported under Cooperative Agreement G15AC00002 between Carnegie Mellon University and U. S. Geological Survey.