




Mathematical optimization for time series decomposition

Seyma Gozuyilmaz¹ · O. Erhun Kundakcioglu¹ 

Received: 24 December 2019 / Accepted: 13 May 2021 / Published online: 8 June 2021

© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2021

Abstract

Decomposing time series into trend and seasonality components reveals insights used in forecasting and anomaly detection. This study proposes a mathematical optimization approach that addresses several data-related issues in time series decomposition. Our approach does not only handle longer and multiple seasons but also identifies outliers and trend shifts. Numerical experiments on real-world and synthetic problem sets present the effectiveness of the proposed approach.

Keywords Time series · Seasonal trend decomposition · Mixed integer nonlinear programming

1 Introduction

Data indexed in chronological order are referred to as time series data. Time series data are classified based on output data as univariate or multivariate and with respect to measuring/sampling intervals as discrete or continuous. When time series is analyzed, a usual assumption in most of the statistical procedures is *stationarity*. That implies data sampled from different time windows behave similarly, regardless of where they are in the timeline. However, in real life, an observation is affected by various factors, especially observations from earlier periods. This effect is the primary motivation behind representing time series as an aggregation of components. When decomposed, components are extracted for each data point, which provides further insights and benefits the prediction of future values (Brockwell et al. 2002).

Some of the most frequently studied components are trend, seasonality, and remainder (Cleveland et al. 1990). *Trend* demonstrates the general tendency of data (i.e., increase, decrease, or stagnate) over a long period of time. Short-term fluctuations do not affect trend, which represents a smooth and long term tendency. Therefore, one rule of thumb for the trend is that it moves relatively slowly over time. On the contrary, repetitive patterns are explained by *seasonality*.

✉ O. Erhun Kundakcioglu
erhun.kundakcioglu@ozyegin.edu.tr

¹ Department of Industrial Engineering, Ozyegin University, Istanbul, Turkey

Seasonality portrays fluctuations based on where the data are within a repetitive cycle. Finally, *remainder* describes random and irregular changes that cannot be explained by the trend and seasonality. Therefore, remainders are residuals of time series after trend and seasonality are removed. Time series are usually computed as an addition or multiplication of these three components.

Although there are several decomposition algorithms, a limited number of approaches can handle noise and abnormalities. For instance, it is very likely for real-world data to have *abrupt dips and spikes*, which cannot be explained by trend or seasonality. Classical approaches' performances deteriorate significantly in the presence of such noise, due to residual minimization. Another possible scenario is a rare *sudden change in trend* due to an external factor, which cannot be identified with a classical decomposition approach. To the best of our knowledge, no tool, while maintaining the smoothness of trend and repetitive patterns of (multiple) seasons, can handle abnormalities and sudden changes in trend. To illustrate these issues, consider the toy example in Fig. 1.

The data in Fig. 1a are generally increasing with several changes in pattern. It is important to identify if these changes are temporary (abrupt), permanent (change in trend), recurring (due to seasonality). To correctly identify a change in pattern, we decompose data into trend, seasonality, and remainder. We aim to provide an answer with minimum residuals and best explainability on the changes in the pattern. When decomposed accurately as in Fig. 1b, we observe that

- the data have a generally increasing trend *with a sudden drop around time 35, and a recovery around 200*,
- a seasonal effect of 52 time points is repeated (pictured in the middle), and
- aside from the expected residuals, *there are four abrupt spikes around 40, 50, 100, and 220, and one dip around 160*.

Note that these observations are not always easy to compute from the original data, especially in the presence of sudden trend shifts and random spikes/dips. A decomposition that is robust against these anomalies gives us the ability to understand the underlying phenomenon and make higher accuracy predictions. Therefore, it is crucial to capture issues precisely and compute a cleaned-up decomposition. However, no available method can decompose time series (i) plotting perfectly repetitive seasonality component(s), (ii) plotting a smooth trend with possible unexpected shifts, and (iii) filtering out abnormalities. Earlier seasonal trend decomposition approaches focus only on repetitive seasons and smooth trends (Cleveland et al. 1990). Wen et al. (2019) are the first to highlight the importance of data anomalies and propose a robust seasonal trend decomposition to address the issue. Although the need is well identified and the proposed model successfully decomposes a range of data sets, it may fail to preserve the fundamental phenomena of repetitive seasonality patterns and trend smoothness, especially in the presence of anomalies. In sum, no currently available method can successfully decompose the data in Fig. 1a with a similar output to Fig. 1b, even though it is a relatively simple data set with one seasonality effect.

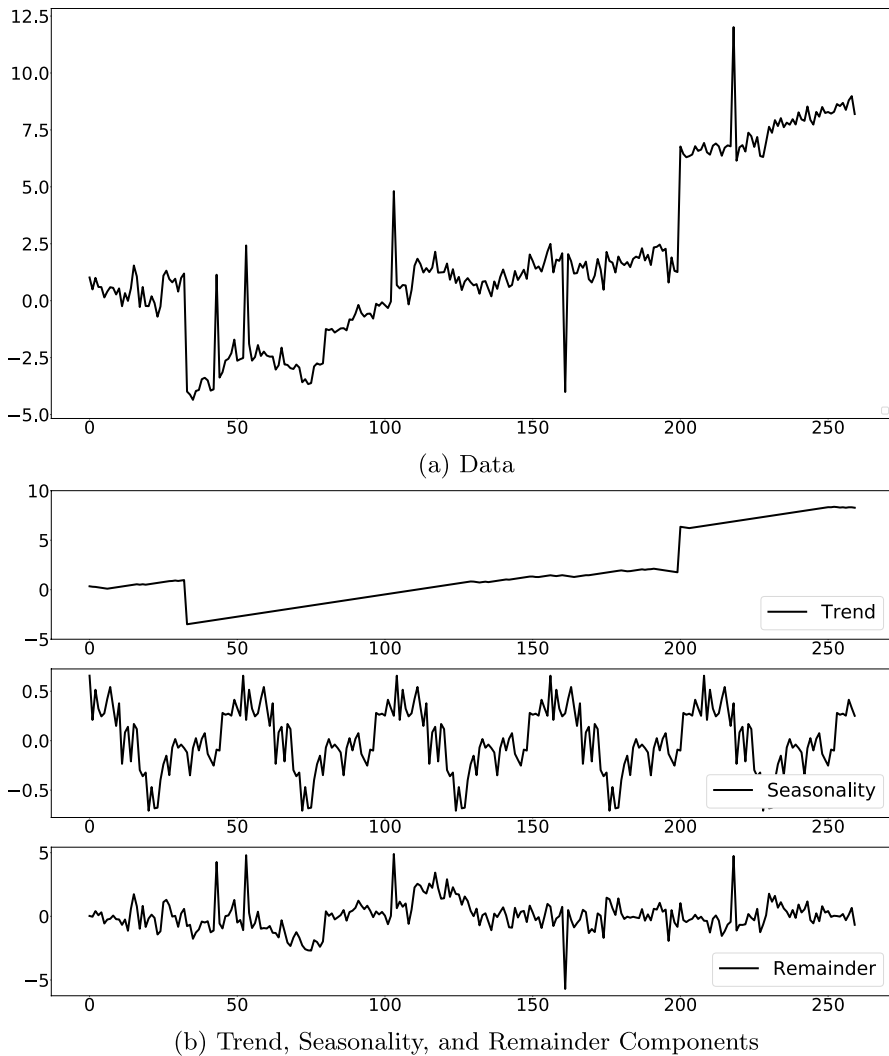


Fig. 1 A toy example with the associated decomposition for one seasonality effect with a season length of 52 time points (sampling intervals)

In this paper, we propose a mixed integer nonlinear programming (MINLP) formulation that decomposes complex time series data, addressing the aforementioned data-related issues. Our main contribution is to propose a mathematical model that can detect multiple seasonality effects and manage abrupt changes in otherwise smooth trends, while maintaining robustness against anomalies. Mathematical optimization also brings explainability, which, unlike black-box models, allows the decision maker to make further adjustments or introduce additional considerations during decomposition. It is valuable from a business operations standpoint to have control over trend and seasonality rules through a coherent model. This ultimately

yields a better understanding of driving forces for prediction and planning strategies, hence the widespread use of decomposition in business operations (Mendez-Jimenez and Cardenas-Montes 2018). Within this context, our model outputs insights that every decision maker needs for various data sources (e.g., sales, operations, or finance) to overcome challenges. We propose directly solving the mathematical model to optimality using a commercial solver. Even though this exact approach is computationally challenging for large data sets, our experiments show promising results in terms of explainability and forecasting performance for a broad spectrum of problems.

The rest of the paper is organized as follows. We review the related literature in Sect. 2. In Sect. 3, we formulate a mathematical model that provides the decomposition of time series while addressing possible data issues. We present numerical experiments and discussions on real and synthetically generated data in Sect. 4. Finally, we provide concluding remarks and possible future extensions in Sect. 5.

2 Literature review

Several alternative approaches are developed for time series analysis, due to its importance in various fields (Yaffee and McGee 2000; Brockwell et al. 2002; Montgomery et al. 2015). Among these, decomposition is one of the most popular methods due to its interpretability and forecasting ability over sub-series (Theodosiou 2011; Arputhamary and Lawrence 2018). The decomposition technique is initially developed by Persons (1919) to isolate and identify the notable features of a time series. After this study, several decompositions with different capabilities are proposed based on average, subspace, regression, and transformation. Next, we present decomposition approaches in each of these categories and present their capabilities and shortcomings on the common data issues we presented in Sect. 1.

Average-based methods are uncomplicated statistical approaches for decomposition. The filtered value is the average result within a predefined time window. A popular technique within this framework is the seasonal trend decomposition based on loess smoothing (STL) (Cleveland et al. 1990). This methodology is nonparametric (except for the season length), straightforward, and produces robust outputs from the data that may have missing values in various fields (Theodosiou 2011). These are the abilities that make STL powerful and flexible in fitting the data; however, this technique is prone to data issues. Ollech (2018) and Wen et al. (2019) report that STL fails to extract the seasonality component accurately, when seasonality shifts and fluctuations exist.

Subspace-based decomposition takes advantage of subspace learning, which extracts stochastic influences by reducing the dimension of time series. One example is the Singular Spectrum Analysis (SSA) (Hassani 2007; Zhigljavsky 2011). SSA assumes series are linear and stationary, and any additional noise follows a normal distribution (Duarte et al. 2019). This strong assumption makes SSA suitable for short time series and impractical for certain real-world data (Hassani 2007; Wen et al. 2019).

Another approach for time series analysis is multiple regression-based decomposition. The Hodrick–Prescott filter is one of the fundamental techniques within this context (Hodrick and Prescott 1981). It works well with slow-changing trends and fast-changing residuals (Hamilton 2017). Dokumentov and Hyndman (2015) introduce an extension, seasonal trend decomposition procedure based on regression (STR), which handles seasonal shifts and multiple seasonalities, and provides confidence intervals for the predicted components. Dokumentov and Hyndman (2015) also propose a robust version to deal with spikes and dips better; however, STR or robust STR cannot handle abrupt changes on trend (Wen et al. 2019).

Transformation-based methods extract stochastic and deterministic influences from time series. Fourier and Wavelet transforms are well-known examples. Fourier Transform (FT) breaks down a time series into a sum of sinusoidal functions (Gomes and Velho 2015). It is applicable in several fields, where deterministic and stochastic parts occur at different frequency intervals. However, FT faces issues when analyzing non-periodic and non-stationary data (Duarte et al. 2019). Non-stationary time series can be made stationary using transformation and differentiation (Chen and Anderson 1998; Brockwell and Davis 2002). Wavelet transform (WT) addresses some issues with FT, extracting components at different frequencies (Liu 2010), but jeopardizes time relationships among observations, deteriorating modeling and forecasting performance (Duarte et al. 2019).

Besides these classical approaches, several extensions and hybrid approaches are developed. For instance, auto-regressive integrated moving average (ARIMA) is a prevalent approach (Bartholomew 1971), with extensions such as X-11-ARIMA, X-12-ARIMA, X-13-ARIMA-SEATS, Seasonal ARIMA (Dagum 1980; Findley et al. 1998; Monsell et al. 2013; Terrell 2019). These techniques contain factors such as external regressors or calendar effects (Dokumentov and Hyndman 2015), and numerous applications benefit from these (Qin et al. 2019). However, one shortcoming is that these methods only scale to small or medium-size data (Ollech 2018; Wen et al. 2019). In addition, the differencing processing utilized is not always satisfactory to handle nonlinearity and long seasons simultaneously (Qin et al. 2019). Another similar extension is the Seasonal Adjustment at Bell Laboratories (SABL) (Cleveland et al. 1978). Although not commonly used, SABL's main feature is that it provides a graphical methodology to assess the performance and behavior of decomposition (Bleikh and Young 2016).

Recently, Wen et al. (2019) propose robust seasonal trend decomposition (Robust-STL) to address some of the data issues and handle long time series. RobustSTL can be considered as a hybrid method that iteratively denoises input signal using bilateral filter, obtains relative trend for L1 sparse model and season using non-local seasonal filtering, and adjusts trend and season until convergence. Various experiments using synthetic and real data sets demonstrate that RobustSTL outperforms three well-known algorithms: STL (seasonal trend decomposition using loess), TBATS (Trigonometric Exponential Smoothing State Space model with Box-Cox transformation, ARMA errors, Trend and Seasonal Components), and STR (seasonal trend decomposition procedure based on regression). As it is the most capable decomposition tool, it is used as the main reference for the discussions in the rest of the paper. Despite the strengths, we observe in (Wen et al. 2019) as well as our numerical

experiments that RobustSTL has the following shortcomings: (i) the seasonality patterns vary from one cycle to the next, (ii) trend follows the input time series very closely, without the expected smoothness quality. Our first observation defeats the purpose of decomposition as it is impossible to identify what seasonality cycle is upcoming and perform forecasting. Our second observation helps with catching rapid changes in trend, but deteriorates the performance on non-problematic time series. It should also be noted that RobustSTL cannot handle more than one seasonality component. Our study aims to address these shortcomings with a mathematical optimization approach around average-based decomposition.

3 Mathematical model

This section presents a mathematical model that divides time series into three additive components (i.e., trend, seasonality, and remainder). The overall behavior and patterns on time series differ from one problem set to another. Still, there are widely accepted expectations on decomposition to have an understanding and produce accurate forecasts. In this context, the trend component demonstrates the continuous tendency of data that usually is a smooth function. Seasonality typically shows regular and periodic patterns within a fixed period. Finally, the remainder captures the remaining gaps, which should be as small as possible. We build our basic model based on these expectations and use the following notation:

Sets

- T : set of time points for which there is a measurement/sampling, indexed by t
 R : set of time points (starting with zero) in a season that recurs, indexed by r

Parameters

- ρ : allowable difference between two consecutive trend values
 B : upper bound on the number of trend shifts
 K : upper bound on the number of anomalies (abrupt spikes and dips)

Decision Variables

- X_t^{Trend} : trend value at time $t \in T$
 X_t^{Season} : seasonality value at time $r \in R$
 $X_t^{\text{Remainder}}$: remainder value at time $t \in T$

The model takes a time series input, denoted by X_t , data sampled at time $t \in T$. Our aim is to decompose this series into seasonality, trend, and remainder components. Before we proceed with the model, we define a many-to-one matching between the set of all time points and those in a season. Seasonality explains periodic and regular fluctuations that recur during each repeating season. Thus, it should get the same value in every iterated $|R|$ time points, where $|R|$ equals the length of a season.

To serve that purpose, we use congruent modulo of each time point, formalized by function $f : T \rightarrow R$ as follows:

$$f(t) \equiv t \pmod{|R|} \quad t \in T \quad (1)$$

As mentioned above, the aim of this study is to break the observed value at time t down into a set of its components. Thus, the base MINLP model is formulated as follows:

$$\min \sum_{t \in T} X_t^{\text{Remainder}^2} \quad (2a)$$

$$\text{s.t. } X_t = X_t^{\text{Trend}} + X_{f(t)}^{\text{Season}} + X_t^{\text{Remainder}} \quad t \in T \quad (2b)$$

$$\sum_{r \in R} X_r^{\text{Season}} = 0 \quad (2c)$$

$$X_t^{\text{Trend}} - \rho \leq X_{t+1}^{\text{Trend}} \leq X_t^{\text{Trend}} + \rho \quad t \in T \quad (2d)$$

In this formulation, the objective function minimizes the sum of squares of all remainder values in T . Remainder values denote residuals, whose convergence to zero from above or below ensures trend and seasonality has more ability to represent a given data. Constraint (2b) shows that we adopt an additive model. The seasonality component is unrestricted, protecting a specific pattern over each season. Constraint (2c) ensures that the sum of seasonality values does not create an artificial boost or reduction over these seasons. Constraints in (2d) ensure that the trend is smooth. In the stable cases (i.e., no shifts in trend), the trend value from time t to $t + 1$ can vary up to allowable difference of ρ . Next, we present how we build extended models on this base model to address some data issues.

3.1 Shifts in trend

The trend component denotes the general tendency of time series over a long period. Even though movements may change in time, trend is assumed to be smooth to provide robust decompositions. However, abrupt shifts in trend may occur in real-world cases. The following decision variables are presented to handle these changes at times, yet provide smoothness elsewhere:

New Decision Variables

Y_t : 1 if trend has an abrupt shift at time t , 0 otherwise

With the inclusion of these new binary variables, the base model is extended by replacing constraint (2d) with

$$X_{t+1}^{\text{Trend}} \leq X_t^{\text{Trend}} + \rho + MY_t \quad t \in T \quad (3a)$$

$$X_{t+1}^{\text{Trend}} \geq X_t^{\text{Trend}} - \rho - MY_t \quad t \in T \quad (3b)$$

$$\sum_{t \in T} Y_t \leq B \quad (3c)$$

$$Y_t \in \{0, 1\} \quad t \in T \quad (3d)$$

where M denotes a large enough number. Y_t indicates if the trend has an abrupt change at time t , or not. Constraints (3a) and (3b) guarantee that the value of trend at time $t + 1$ should remain between certain ranges to keep smooth structure, except for when $Y_t = 1$. Trend at time $t + 1$ can take any value by courtesy of a sufficiently large constant, when $Y_t = 1$, which is expected in the case of a substantial trend shift due to the nature of objective¹. Parameter B denotes the maximum number of trend shifts movements. Constraint (3c) ensures that the allowed number of spike and dip movements cannot be more than B .

3.2 Multi-seasonality

Time series representing real-world activity are often affected by several factors that indicates several seasonal periods. This can be addressed with the contributions of the following changes.

New Sets

J : set of seasons

R_j : set of time points (starting with zero) in season $j \in J$ that recurs

In addition, the many-to-one matching represented by function f in (1) is extended to multiple seasons as follows:

$$f_j(t) \equiv t \pmod{|R_j|} \quad t \in T, j \in J \quad (4)$$

Finally, the formulation of the base model is updated by adding a new index j to decision variables X_r^{Season} in the base model.

Decision Variables

$X_{r,j}^{\text{Season}}$: seasonality value of type $j \in J$ at time $r \in R_j$.

Constraints (2b) and (2c) are updated as follows:

$$X_t = X_t^{\text{Trend}} + \sum_{j \in J} X_{f_j(t),j}^{\text{Season}} + X_t^{\text{Remainder}} \quad t \in T \quad (5a)$$

¹ The user can purposefully set M value and limit trend shifts. We highlight even a large number does not lead to an unrealistic solution due to the rest of the constraints.

$$\sum_{r \in R_j} X_{r,j}^{\text{Season}} = 0 \quad j \in J \quad (5b)$$

Note that, multi-seasonal decomposition is computationally more challenging and it can provide unrealistic results. To overcome this issue and to ensure meaningful segregation of individual seasonal effects, seasonality smoothness can also be introduced to the model.

3.3 Unexpected spikes and dips (anomalies)

In a traditional decomposition procedure, the trend is smooth, and seasonality denotes regular fluctuations. However, unexpected cases such as large spikes or dips may occur in reality, and these are hard to attribute to the decomposed components. These instances do not necessarily stand out in time series as other components may partly alleviate unexpected effects. Thus, these anomalies in data have to be filtered out while simultaneously decomposing the data. We extend the base model to address this issue as follows:

New Decision Variables

$X_t^{\text{Remainder}'}$: actual remainder value (with abnormality) at time $t \in T$
 Z_t : 1 if time series has an unexpected behavior (anomaly) at time $t \in T$, 0 otherwise

The base model is extended by adding the following constraints:

$$X_t^{\text{Remainder}'} \leq X_t^{\text{Remainder}} + MZ_t \quad t \in T \quad (6a)$$

$$X_t^{\text{Remainder}'} \geq X_t^{\text{Remainder}} - MZ_t \quad t \in T \quad (6b)$$

$$\sum_{t \in T} Z_t \leq K \quad (6c)$$

$$Z_t \in \{0, 1\} \quad t \in T \quad (6d)$$

Similar to allowing the trend shifts, constraints (6a) and (6b) guarantee that instances to have virtually unbounded² remainders, unless $Z_t = 0$. Parameter K denotes the maximum number of anomalies and constraint (6c) ensures that no more than K unbounded and unpenalized remainders are allowed. Note that these actual remainders are only penalized in the objective when $Z_t = 0$ and consequently $X_t^{\text{Remainder}} = X_t^{\text{Remainder}'}$. This way, $X_t^{\text{Remainder}}$ penalized in the objective function is filtered from the factors causing anomalies.

² Similar to the trend shift, the user can set M value if spikes or dips are to be bounded.

3.4 Final MINLP model and parameters

Our final model addressing all the data issues is summarized as follows:

$$\min \sum_{t \in T} X_t^{\text{Remainder}^2} \quad (7a)$$

$$\text{s.t. } X_t = X_t^{\text{Trend}} + \sum_{j \in J} X_{f_j(t),j}^{\text{Season}} + X_t^{\text{Remainder}} \quad t \in T \quad (7b)$$

$$\sum_{r \in R_j} X_{r,j}^{\text{Season}} = 0 \quad j \in J \quad (7c)$$

$$X_{t+1}^{\text{Trend}} \leq X_t^{\text{Trend}} + \rho + MY_t \quad t \in T \quad (7d)$$

$$X_{t+1}^{\text{Trend}} \geq X_t^{\text{Trend}} - \rho - MY_t \quad t \in T \quad (7e)$$

$$\sum_{t \in T} Y_t \leq B \quad (7f)$$

$$X_t^{\text{Remainder}'} \leq X_t^{\text{Remainder}} + MZ_t \quad t \in T \quad (7g)$$

$$X_t^{\text{Remainder}'} \geq X_t^{\text{Remainder}} - MZ_t \quad t \in T \quad (7h)$$

$$\sum_{t \in T} Z_t \leq K \quad (7i)$$

$$Y_t, Z_t \in \{0, 1\} \quad t \in T \quad (7j)$$

This formulation uses several parameters for decomposition: length of the season(s) ($|R|$), upper bounds for the number of trend shifts (B) and anomalies (K), and allowable change between two consecutive trend values (ρ). On the contrary, for instance, RobustSTL method, although can handle one seasonality effect only, seems more advantageous as it requires season length only. As the season length is defined by the user and fed to both algorithms, we propose using the time series data and estimate three additional parameters required by the MINLP model using the simple heuristics explained next.

3.5 Estimating B , K , and ρ

Upper bounds on the number of trend shifts and anomalies can be detected using a signal analysis or transformation tool. However, we propose the following heuristics to set these MINLP parameters. The performance of decomposition provided by MINLP depends on how well these parameters are estimated.

We assume the upper bound on the number of trend shifts can be estimated as the number of trend shifts we can identify on the time series. To detect the trend shifts, we check if a point t signals a shift in pattern. For each time point t (except the first and last two in time), the absolute difference between the average of three neighbors (i.e., X_{t-2} , X_{t-1} , X_t) and next three neighbors (i.e., X_{t+1} , X_{t+2} , X_{t+3}) is computed. Our approach's logic is that these absolute differences should be reasonably small, otherwise we conclude there is a significant trend shift. As we do not expect too many considerable trend shifts, we assume that at least 90% of these values should be acceptable. Starting with the 90-th percentile, we include these values in the acceptable set one by one, if they are less than the mean plus three standard deviations of the already included values. If a value cannot be added to the set, that value and all values beyond that are counted as significant, pointing an abrupt trend shift. Figure 2 presents a flow-chart illustrating this simple procedure.

Similarly, we assume the upper bound on the number of anomalies can be estimated as the number of anomalies we detect on the time series. To detect the anomalies, we check if a point stands out compared to its neighbors. For each time point t (except the first and last two in time), this time the absolute difference between the average of four neighbors (i.e., X_{t-2} , X_{t-1} , X_{t+1} , X_{t+2}) and X_t is computed. Next, finding the outliers in this set remains, for which we use the same procedure as above (see Fig. 2).

Finally, as far as the allowable change between two consecutive trend values is concerned, we propose computing ρ as the average change required to reach both higher and lower values of the time series within the given time window. That is, we divide the difference between one of the larger and one of the smaller values by the number of observations (i.e., $|T|$). Note that the minimum and maximum values might be selected, but we conjecture that would be misleading due to possible abnormalities. Therefore, we use 95-th percentile rather than the maximum value and 5-th percentile rather than the minimum.

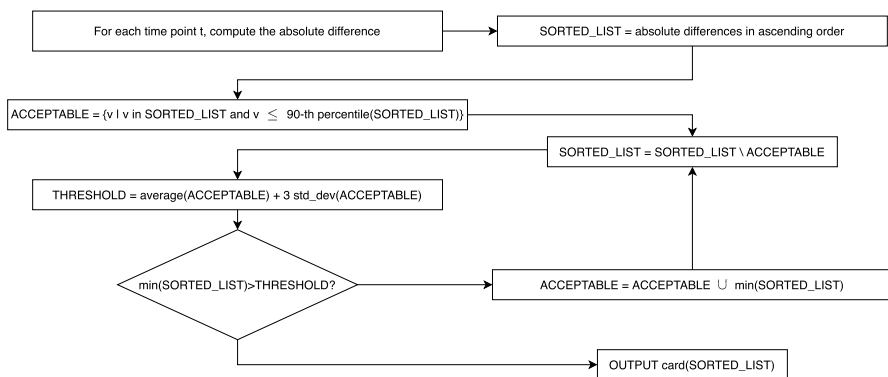


Fig. 2 Flowchart of upper bound computation for anomalies and trend shifts

4 Computational results

This section presents experiments that show the effectiveness of our algorithm on synthetic and real problem sets. Aside from the validation cases, we compare our approach against RobustSTL³. In our benchmarking against RobustSTL, we do not consider multi-seasonality because RobustSTL does not handle multi-seasonality. All computations are performed using Python, calling Gurobi 8.0 (Gurobi Optimization 2020) and LocalSolver (LocalSolver 2021) to solve optimization problems, on a 3.5 GHz Intel Xeon (E5-1650 v2) computer with 16 GB DDR3 ECC (1866 MHz) RAM and the macOS High Sierra operating system.

4.1 Data generation

We generate data sets involving 60, 260, 120, and 520 time points ($|T|$), mimicking five and ten years of data observed on a monthly and weekly basis. In real life, seasonality effects are usually yearly (e.g., annual cash flows or weekly inventory stock), hence season lengths of 12 (months) and 52 (weeks). See Fig. 1 for details on four alternative time series obtained. We estimate parameters needed for the formulation (i.e., B , K , and ρ) using heuristics in Sect. 3.5.

Each time series is generated using the additive variation of the components (i.e., trend, seasonality, and remainder). They are generated based on different experiment types that can be handled in two groups as Type I and Type II. The details are summarized in Table 2.

At each time point, we randomly generate a decision on trend value to decrease, increase or remain the same. We assume the probability that trend increases is 0.6, whereas the probability that it decreases or remains the same is 0.2 each. Thus, we expect an overall increasing trend in all experiments. Furthermore, trend is also open to abrupt shifts (upwards or downwards, 0.5 probability each), with an expectation of once every two years and a value of 3 or once every five years with a value of 5.

Seasonality values are generated using the following three shapes: (i) triangular, (ii) square, and (iii) sinusoidal. As presented in Table 2, we first define the range (maximum and minimum) for seasonality depending on experiment type. We then generate seasonality for each year (12 or 52 observations) based on one of the shapes pictured in Fig. 3).

Finally, we generate noise (to be decomposed as remainder) at each observation from a normal distribution with the given mean and standard deviation in Table 2. Besides these small movements, more substantial anomalies occur randomly with an expectancy of once a year or two years, in the magnitude of 5 or 7, respectively.

Considering the four data and season length sets in Table 1, two types of parameter settings in Table 2, and three seasonality types in Fig. 3, we have 24 different experimental setups. We generate three instances using each setup, leading to a total of 72 instances.

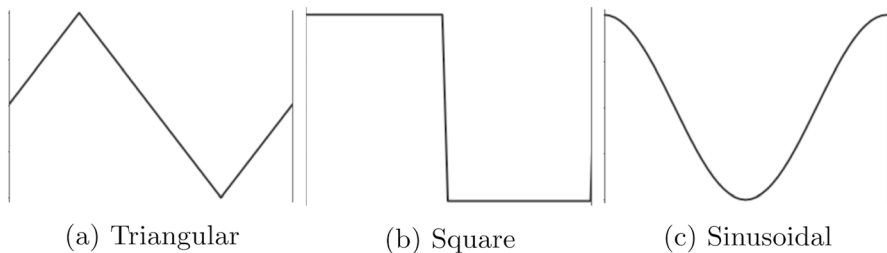
³ The code that solves RobustSTL is available at <https://github.com/LeeDoYup/RobustSTL>.

Table 1 Parameter summary

Parameters	Five years data		Ten years data	
	Monthly	Weekly	Monthly	Weekly
$ T $	60	260	120	520
$ R $	12	52	12	52

Table 2 Parameters for experiment types

Parameter	Type I	Type II
Regular trend deviation	0.25	0.5
Abrupt trend shift magnitude	3	5
Abrupt trend shift interarrival rate	2 years	5 years
Seasonality range	$[-0.5, 0.5]$	$[-1, 1]$
Noise mean	0	0
Noise standard deviation	0.15	0.25
Anomaly magnitude	5	7
Anomaly interarrival rate	1 year	2 years

**Fig. 3** Alternative seasonality shapes in each experiment type

4.2 Verification of the model

In this section, we visualize decomposed values obtained via the MINLP model against RobustSTL on some instances to shed light on our performance. Each figure shows time series data generated using a thick line at the top subplot. These data are generated by synthetically generating each component, which we display using solid lines in the corresponding component's subplot. In the subplots, we also present decomposition results on the time series using dashed lines. Thus, a decomposition is deemed successful when the margin between dashed and solid lines in these subplots is small.

Figure 4 demonstrates results on a time series data that is generally increasing with several changes in the pattern. We show MINLP and RobustSTL

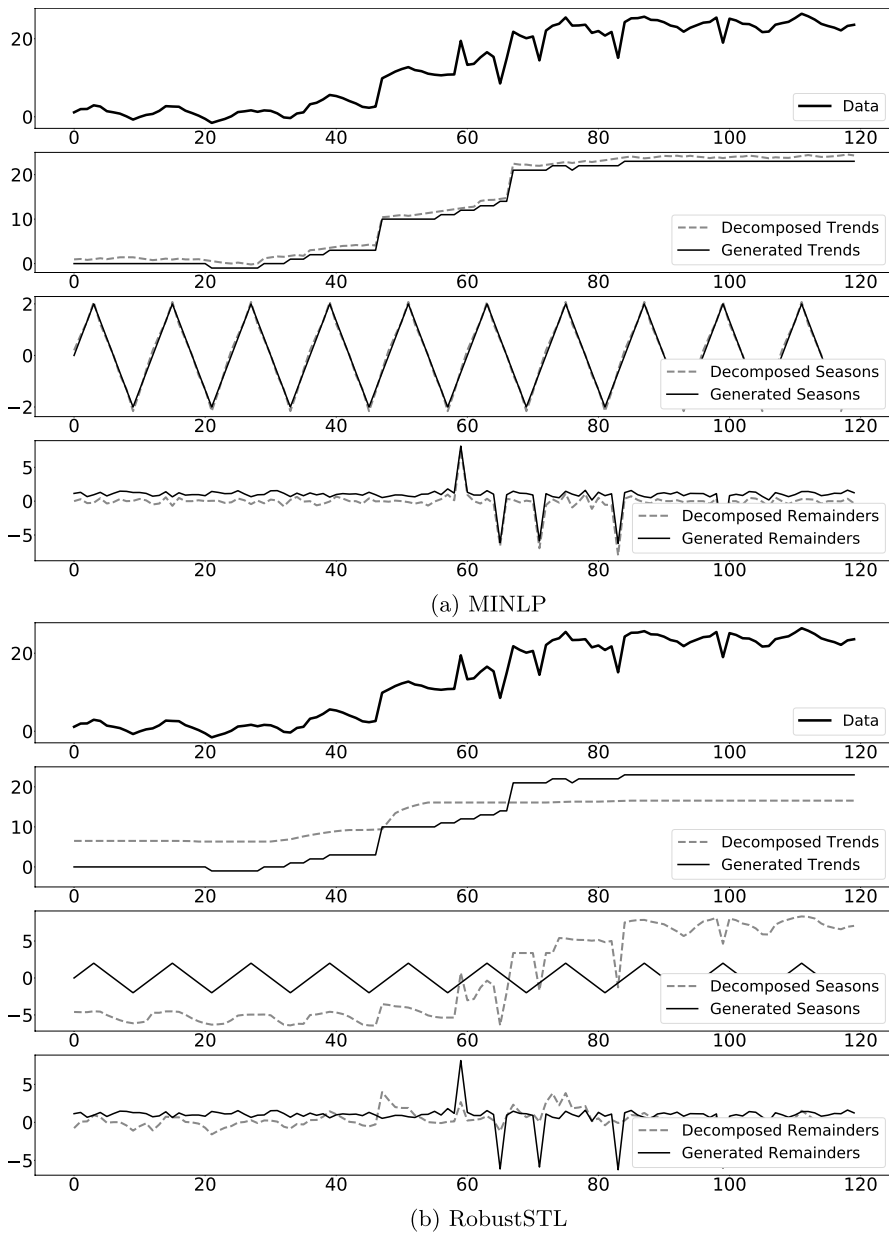


Fig. 4 Example 1 in the presence of trend shifts and anomalies

decompositions into trend, seasonality, and remainder. Our proposed model identifies the two trend shifts which are fully consistent with the generated data; however, RobustSTL fails to closely capture changes in trend. Furthermore, seasonality patterns are off and not cyclic, and abrupt peaks and dips in the data seem to have deteriorated the performance of RobustSTL.

The second example in Fig. 5 has more apparent anomalies. There are four spikes and one dip that are certainly out of cyclic behavior. MINLP detects all anomalies as well as shifts in trend. The seasonality is not as smooth as the generated one, yet it captures the overall pattern. On the other hand, RobustSTL misses some anomalies and more importantly, adds detected anomalies on the seasonality component. Note that the first trend shift around 40 also appears on the seasonality component. That highlights the main shortcoming of RobustSTL—it is not possible to predict what next season will bring, using the non-recurring seasonality component.

Finally, we present the decomposed components for the time series with multiple seasons. As mentioned before, RobustSTL cannot handle multi seasonality. We observe in Fig. 6 that our proposed model catches the overall structure of both seasons, trend, and abrupt shifts in trend. However, there are issues with detecting anomalies and more importantly artificial fluctuations appear in both seasonal effects. We can address these issues by adding some smoothness constraints in the MINLP model for seasonality. This way, we can avoid impractical results and obtain more meaningful decomposed seasonality. This example should suffice to prove our MINLP handles multi-seasonality; however, as our numerical experiments' primary focus is on single-season cases, we do not delve into further analysis and improvements.

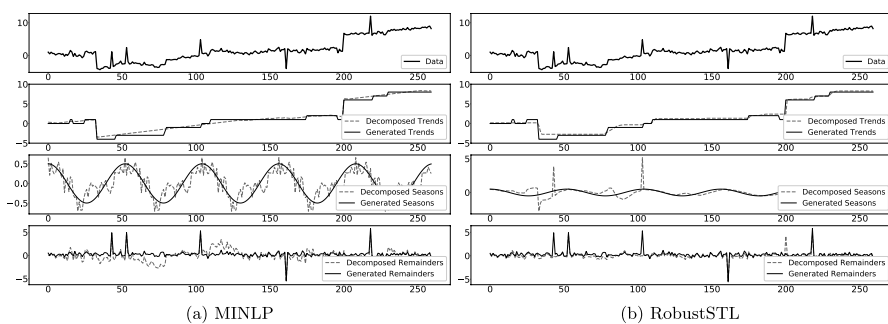


Fig. 5 Example 2 in the presence of trend shifts and anomalies

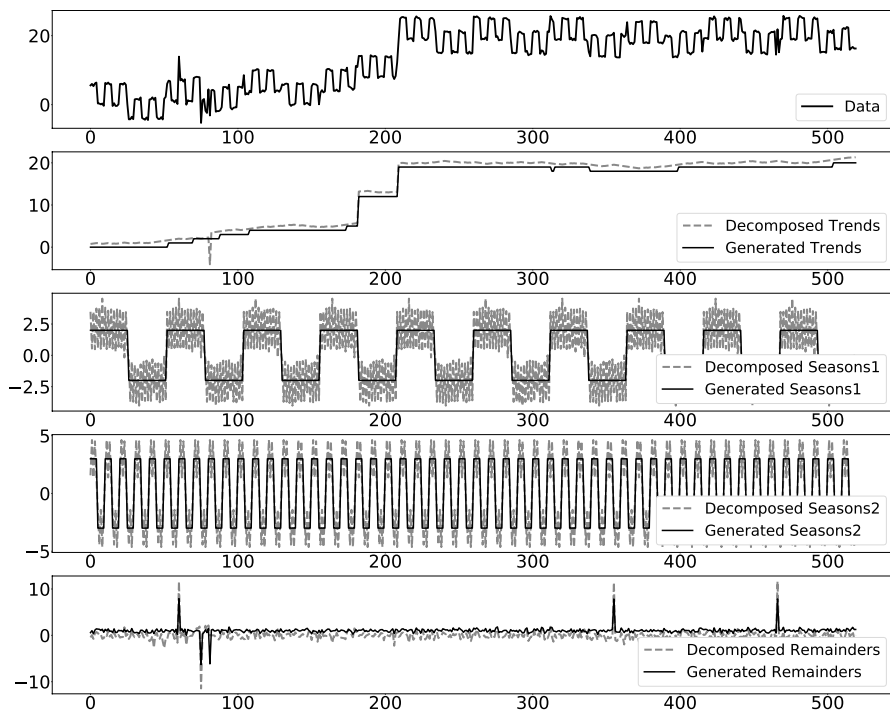


Fig. 6 MINLP decomposition for multiple seasons

4.3 Comparison of methods on synthetic data

We conduct an array of experiments to prove the effectiveness of our proposed algorithm on synthetic data sets. For the MINLP problems Gurobi is used with a one-hour time limit. We create 72 synthetic problem instances as explained above and report the mean absolute error (MAE) between the decomposed and generated trend, season, and remainder. Next, we report how successful our heuristics are in setting the parameters and solution times of algorithms.

We compare mean absolute error (MAE) for both approaches' decomposed trends, seasonalities, and remainders. For completeness, we also present box plots for each component. MAE values for experiments are summarized in Table 3. Each cell displays the average of three data sets that are generated the same way.

In Table 3, we present both algorithms' performance and highlight the algorithm that is closer to the generated trend and seasonality values with bold font. For completeness, we show MAE results for the remainder using the decomposed versus generated values. However, proximity to generated remainders is not necessarily an indicator of success, as the algorithm might have provided better remainders than those generated. For long seasons (i.e., five and ten years of weekly data), Robust-STL outperforms our MINLP in most instances, with exceptions, especially on the longer data sets. Especially on ten years of weekly data, MINLP catches seasonality

Table 3 MAE comparison between MINLP using Gurobi and RobustSTL

		Ten years data					
		Five years data					
		Weekly		Monthly	Weekly		Monthly
		MINLP	RobustSTL	MINLP	MINLP	RobustSTL	MINLP
Type I	Trend	0.3144	0.1457	0.5396	1.3448	0.0812	0.6553
	Seasonality	0.2424	0.1364	0.4951	1.5530	0.0877	0.3153
	Remainder	0.2243	0.1455	1.2589	0.6982	0.1161	1.4628
Type II	Trend	0.2088	0.1781	0.3922	0.5672	0.1189	0.3942
	Seasonality	0.1628	0.1209	0.2393	0.5356	0.1267	0.1688
	Remainder	0.2587	0.1627	0.7840	0.5470	0.1441	1.4992
Type I	Trend	0.2186	0.1038	0.5155	1.5661	0.2602	0.6904
	Seasonality	0.1058	0.1684	0.5018	1.3523	1.0018	0.6628
	Remainder	0.2210	0.1666	1.1947	0.9039	0.1612	0.4362
Type II	Trend	0.2564	0.1823	0.6786	0.4934	0.1791	1.4301
	Seasonality	0.1976	0.1778	0.4709	0.4638	0.1127	0.9378
	Remainder	0.2288	0.1750	1.4240	0.5025	0.1478	0.3897
Type I	Trend	0.3222	0.1354	0.6008	1.3623	0.2471	2.1231
	Seasonality	0.2395	0.1178	0.3795	1.4895	0.1163	0.5180
	Remainder	0.2066	0.1151	1.1269	0.7774	0.1284	0.3804
Type II	Trend	0.1594	0.1507	0.3703	0.8203	0.2021	1.1929
	Seasonality	0.1127	0.1501	0.1969	0.8355	0.1002	0.3791
	Remainder	0.2428	0.1706	0.5678	0.7314	0.0890	0.1886
					1.0013	0.2506	0.1193

Lower MAE is highlighted in bold

better on more instances. For shorter seasons (i.e., monthly data), MINLP outperforms RobustSTL on all instances but one. Judging by the data in this table, there are more settings where MINLP performs better than RobustSTL, compared to those where RobustSTL is better.

In order to have an in-depth look, we present Fig. 7 that compares the distribution of errors in trend on all 72 instances. Likewise, Fig. 8 shows the distribution of errors in seasonality for both algorithms.

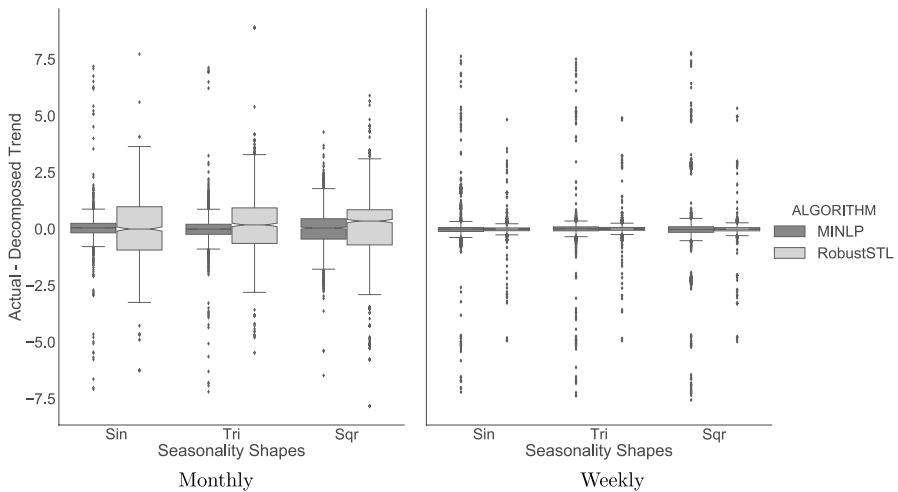


Fig. 7 Actual-decomposed trend comparison

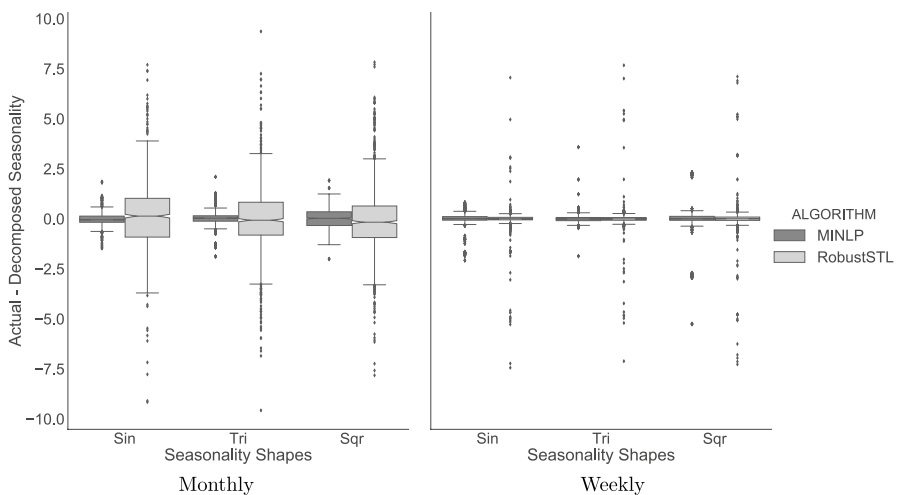


Fig. 8 Actual-decomposed seasonality comparison

These figures illustrate the success of our MINLP better. Starting with the monthly box plots on the left, MINLP outperforms RobustSTL by a large margin on both trend and seasonality, with fewer outliers and more desirable error distribution. The difference between algorithms is more apparent for sinusoidal seasonality. Weekly figures on the right show that the differences between algorithms are subtle. There are more outliers for MINLP trend; however, outliers for the seasonality of RobustSTL are substantially higher. Thus, we conclude that in the weekly cases, RobustSTL is only better in estimating the trend by a small margin.

Next, we present our observations on our heuristic performance and time performance of MINLP, which helps us understand why our approach does not always perform splendidly.

4.3.1 Heuristic Performance on Setting MINLP Parameters

Experimental results on all problems solved demonstrate the effectiveness of the proposed heuristics that set MINLP parameters. In general, ρ is easy to estimate and our parameters estimated are reasonably close to the parameter used in data generation on all instances. However, the more critical B and K parameters are challenging to tune. The following remark with two extreme examples shows that an accurate estimation of these parameters is of vital importance.

Remark 1 Our validation experiments reveal that if B and K are correctly set to the values used in data generation, MINLP outperforms RobustSTL in trend and seasonality accuracy in all of the generated instances. On the contrary, when B and K are manually set to zero, RobustSTL outperforms MINLP (which becomes an NLP) on all instances.

Underestimating B and K is riskier compared to overestimating because of their role in the MINLP. As these parameters are upper bounds, a looser upper bound, while providing more freedom to the model, might not necessarily yield a meaningless decomposition. On the other hand, if underestimated, these parameters cause the MINLP to tighten the feasible region and miss trend shifts or anomalies, worsening the trend and seasonality performance. The challenge with our synthetically generated data is that, for instance, trend shifts are more challenging to catch compared to real-life scenarios that usually shift substantially and recover, converging to the shifted value. Our approach would more comfortably detect such behavior due to the more massive gap among the neighboring time windows. Nonetheless, our simple heuristics underestimate B on only around 20% of the generated instances and K on less than 5%.

4.3.2 Solver Selection and Time to Reach Optimality

One issue with solving a challenging mathematical optimization model to optimality is the computation time. Usually, these decomposition problems do not need to be solved in seconds or minutes, especially in the domain of business analytics. A thorough analysis and accurate predictions are of vital importance. However, in general, speed might also be a concern. Therefore, for the sake of a complete analysis, we report the optimality gap achieved for Gurobi against LocalSolver in one hour.

Table 4 shows that Gurobi generally outperforms LocalSolver in solving our MINLP formulation, hence our solver selection. The decomposition performance is also adversely affected using LocalSolver with MAE values of up to ten times that of Gurobi for weekly data. In fact, in all cases where LocalSolver has a better gap, the objective values achieved by LocalSolver are larger and MAE values for trend and seasonality are worse compared to Gurobi. The three instances of Type II experiments with weekly seasonality that LocalSolver achieved a smaller optimality gap have the following MAE values for trend–seasonality respectively:

- 5 year square: 1.3623–1.2965 (LocalSolver), 0.2564–0.1976 (Gurobi).
- 10 year sinusoidal: 1.0914–0.6532 (LocalSolver), 0.1976–0.0883 (Gurobi).
- 10 year triangular: 1.1854–0.6747 (LocalSolver), 0.1828–0.0919 (Gurobi).

It should also be noted that the only setup LocalSolver solves all instances to optimality is 60 observations with (monthly) square season type for Type I experiments, and it takes 686.33 seconds on average. Even this solution time is excessive compared to Gurobi, as we illustrate next. We present average solution times for Gurobi against RobustSTL in Table 5.

Table 4 Optimality gap achieved in one hour on three generated instances of each synthetic setup

Observations in season	Total observations	Season type	Type I		Type II	
			Gurobi (%)	Local solver (%)	Gurobi (%)	Local solver (%)
12	60	Sinusoidal	0.00	5.79	0.00	63.76
		Square	0.00	0.00	0.00	33.33
		Triangular	0.00	1.39	0.00	83.97
	120	Sinusoidal	0.00	100	0.00	85.52
		Square	0.00	50.91	0.00	92.75
		Triangular	0.00	70.79	0.00	100.00
52	260	Sinusoidal	80.31	100.00	82.37	100.00
		Square	100.00	100.00	99.50	95.33
		Triangular	70.12	77.54	74.69	100.00
	520	Sinusoidal	97.82	100.00	100.00	66.67
		Square	99.65	100.00	100.00	100.00
		Triangular	100.00	100.00	100.00	66.67

Lower optimality gap is highlighted in bold

Table 5 Average time to terminate (in seconds) on three generated instances of each synthetic setup

Observations in season	Total observations	Season type	Type I		Type II	
			Gurobi	RobustSTL	Gurobi	RobustSTL
12	60	Sinusoidal	0.06	0.06	0.06	0.05
		Square	0.05	0.05	0.02	0.05
		Triangular	0.04	0.05	0.07	0.05
	120	Sinusoidal	115.28	0.13	0.12	0.14
		Square	4.75	0.14	0.06	0.13
		Triangular	3.11	0.13	0.32	0.14
52	260	Sinusoidal	3600.00	0.46	3600.00	0.43
		Square	3600.00	35.39	3600.00	0.47
		Triangular	3600.00	4.19	3600.00	0.47
	520	Sinusoidal	3600.00	5.69	3600.00	2.32
		Square	3600.00	6.28	3600.00	1.96
		Triangular	3600.00	2.38	3600.00	1.98

We observe that RobustSTL is faster and more practical, and does not benefit from additional time, as it is an iterative method with no user-defined parameter that affects convergence. The MINLP solutions, on the other hand, take more time, hitting the one-hour time limit on half of the instances. These relatively long CPU times in Table 5 and the associated large optimality gaps in Table 4 are due to the fixed-charge constraints in our formulation and poor quality lower bounds. Time performance is one of the significant drawbacks of the MINLP formulation we propose. However, without anomalies and trend shifts, the MINLP for any synthetically generated instance can be solved in less than three seconds. Despite longer solution times and potential drawbacks of heuristics that set the parameters, we conclude with the following remark.

Remark 2 The results in Table 5, together with Table 3 show that if MINLP terminates at optimality for a setting, it outperforms RobustSTL in decomposition performance.

Aside from these 72 problem instances, one additional time series is generated to replicate the experiment in (Wen et al. 2019). That replication contains 750 observations with a total of 15 seasons, 10 trend shifts, and 14 anomalies. Noise is generated with zero mean and a variance of 0.1. Table 6 shows MAE for both algorithms.

Table 6 MAE comparison between MINLP and RobustSTL

	MINLP	RobustSTL
Trend	0.120	0.066
Seasonality	0.032	0.070
Remainder	0.140	0.083

Lower MAE is highlighted in bold

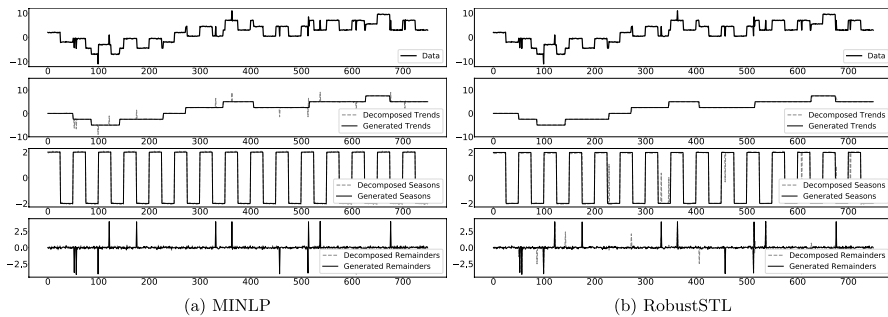


Fig. 9 Decompositions for the data that replicates the experiment in (Wen et al. 2019)

These results show that MINLP detects the seasons better but misses the trend. We present both decomposition results in Fig. 9.

We observe that the only issue with MINLP is that the anomalies are handled as back-to-back trends shifts. This is because our heuristic algorithm incorrectly estimates K as zero due to the small magnitude of anomalies. On the other hand, RobustSTL has severe issues. First, seasons are not recurring, as usual. Some anomalies in data are placed in seasons, but there are also other differences from one season pattern to next. Robust STL also misses some anomalies and creates some artificial peaks and dips, as observed in its remainder. This result confirms our earlier observations, where MINLP performs better, especially when the parameter estimations are accurate. As one of the main goals in decomposition is prediction, we need (i) the recent trend to be accurate, which both methods succeed, and (ii) seasonal effects to be precise, where MINLP excel. Therefore, we conjecture that MINLP is a better forecasting tool, as presented in the following section.

4.4 Comparison of methods on real-world data

Finally, we use real-world data, illustrate the decomposition results of MINLP against RobustSTL, and compare both methods' forecasting performance. We use publicly available real-world data⁴ that involves the minimum *daily* temperatures for ten years (between 1981 and 1990) in Melbourne, Australia. The season length is naturally 365 days. Due to the excessive number of observations, our heuristics provide a liberal number of trend shifts or anomalies. In order to address this and ensure optimality is reached, we restrict the MINLP to no trend shifts or anomalies. The figures below demonstrate the decomposition results on these data using the two algorithms.

Figure 10 shows that there are subtle differences between the decompositions provided by the algorithms. Even though there are similarities in the decomposed

⁴ The data are available at <https://raw.githubusercontent.com/jbrownlee/Datasets/master/daily-min-temperatures.csv>.

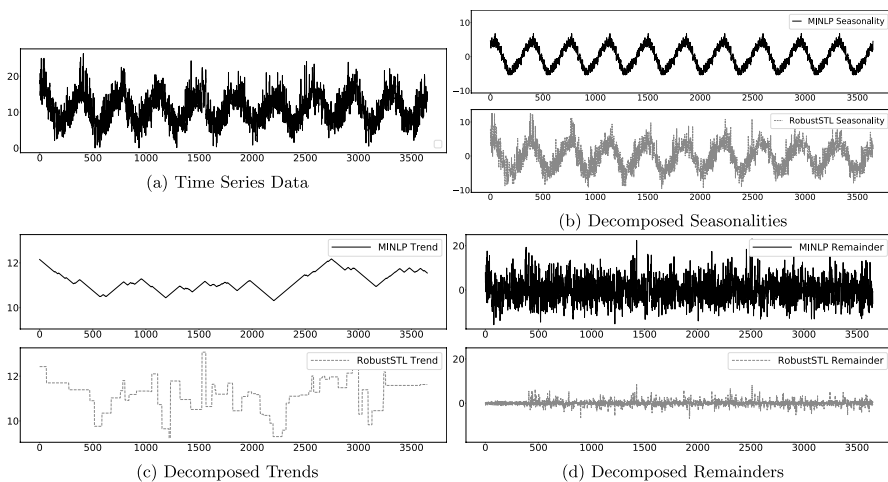


Fig. 10 Comparison of MINLP & RobustSTL decompositions on real-world data

Table 7 Mean absolute error for predictions in years 6–10

Method	Year 6	Year 7	Year 8	Year 9	Year 10
MINLP	2.255	2.255	2.187	2.115	2.135
RobustSTL	2.394	2.659	2.480	2.579	2.394

Better prediction performance with lower MAE is highlighted in bold

trend and seasonality patterns, Fig. 10b, c shows the following: (i) seasonality fluctuates less in MINLP, (ii) seasonalities are not cyclic and do vary among seasons in RobustSTL, (iii) trend is smoother in MINLP, and (iv) remainders are smaller in RobustSTL. However, we know that MINLP returns the minimum possible value for remainders given the hard constraints on the more important trend and seasonality components.

As we do not know the driving forces (i.e., components) of the real-world data, the ultimate KPI in deciding the better algorithm is the prediction performance. Thus, we conduct the following experiment. We decompose the time series for the first n years and test both algorithm's predictions on the entire $n + 1$ -st year. We use this testing approach for the second half of the data, i.e., $n = 6, \dots, 9$. We report the average proximity of both approaches in the following table.

Table 7 shows the mean absolute error (MAE) on each of the five years of the test set. Clearly, MINLP outperforms RobustSTL in predicting each year's value for the available real-world data. It should also be noted that the MINLP solves the problem on an average of 2 seconds, whereas the RobustSTL takes 1012 seconds.

5 Conclusions

Extraction of the time series components provides a guide to make accurate predictions from the analysis of past values. It is significant to have powerful tools for this extraction to understand the time series and enhance forecasting accuracy. Time series occurring in numerous areas (e.g., yearly sales figures, monthly rainfall data, or hourly readings of air temperature) have undesirable properties, making accurate predictions challenging.

We propose a mixed integer nonlinear programming model to decompose complex time series into trend, seasonality, and remainder components with the capability to handle some common data-related issues such as anomalies and trend shifts. Although most of the decomposition approaches in the literature rely on some form of optimization, our proposed approach is one of the very few that directly solve a mathematical optimization model. Experimental results on synthetic and real-world data sets demonstrate the effectiveness of our algorithm in obtaining components accurately. Despite long computation times to achieve optimality, our contribution is to produce better predictions than the current best method on real-world data. The accuracy of the trend and seasonality components is also superior unless there are long seasons and data issues. Our proposed mathematical optimization approach also has better algorithmic transparency, interpretability, and explainability compared to black-box or hybrid methods.

There are several directions for future research. First, some of the shortcomings of our optimization approach can be addressed and more effective formulations can be developed. An improved formulation that utilizes fewer parameters and computes the optimal season length can be studied. For multi-seasonality, the smoothness of seasonality components can be added. Our model assumes the data are always available, but missing data are an issue that can be addressed. As highlighted in the setting replicating an earlier study, trend shifts cannot happen frequently. Constraints addressing this and similar real-life issues can further enhance the model.

Aside from improving the formulation, speedup strategies can be considered. Decomposition results from fast available approaches can be used as a warm start for the optimization model. The fixed-charge constraints with large multipliers are known to cause prolonged runtimes. Three approaches can be taken: (i) timing of shifts and anomalies detected by the heuristics can be fed to the formulation (ii) a lower bounding scheme can be studied for better convergence for an exact method, and (iii) a fast near-optimal algorithm can be proposed that handles fixed-charge constraints (e.g., (Walker 1976)). Instead of heuristically setting bounds, we can also penalize the binary variables associated with anomalies and trend shifts in the objective function. This approach can address outlier robustness with separate ramp or hard margin loss and regularization terms.

Acknowledgements The second author would like to thank Osman Aydemir Yetkin and Nedim Yılmaz, from Bıçakçılar Medical Devices, for helpful feedback on this work's earlier numerical experiments' forecasting performance. The authors also thank two anonymous referees and guest editors for their constructive comments and suggestions.

References

- Arputhamary B, Lawrence DLA (2018) A pragmatic study on time series models for big data. *Int J Emerg Res Manag Technol* 6:67
- Bartholomew DJ (1971) Time series analysis forecasting and control. *J Oper Res Soc* 22(2):199–201
- Bleikh H, Young W (2016) Time series analysis and adjustment: measuring, modelling and forecasting for business and economics. Taylor & Francis, Thames
- Brockwell PJ, Davis RA editors (2002) Nonstationary and seasonal time series models. Springer, New York, pp 179–221
- Brockwell PJ, Davis RA, Calder MV (2002) Introduction to time series and forecasting, vol 2. Springer, Berlin
- Chen Z-G, Anderson OD (1998) Increment-vector methodology: transforming non-stationary series to stationary series. *J Appl Prob* 35(1):64–77
- Cleveland RB, Cleveland WS, McRae JE, Terpenning I (1990) STL: a seasonal-trend decomposition procedure based on loess (with discussion). *J Off Stat* 6:3–73
- Cleveland WS, Dunn DM, Terpenning IJ (1978) SABL: A resistant seasonal adjustment procedure with graphical methods for interpretation and diagnosis. In: National Bureau of Economic Research, Inc, June
- Dagum E (1980) The X-II-ARIMA seasonal adjustment method. In: Essays Collection of Estela Bee Dagum in Statistical Sciences, 01
- Dokumentov A, Hyndman RJ (2015) STR: A seasonal-trend decomposition procedure based on regression. In: Monash econometrics and business statistics working papers 13/15, Monash University, Department of Econometrics and Business Statistics
- Duarte FS, Rios RA, Hruschka ER, de Mello RF (2019) Decomposing time series into deterministic and stochastic influences: a survey. *Digital Signal Process* 95:102582
- Findley DF, Monsell BC, Bell WR, Otto MC, Chen B-C (1998) New capabilities and methods of the X-12-ARIMA seasonal-adjustment program. *J Bus Econ Stat* 16(2):127–152
- Gomes J, Velho L (2015) From Fourier analysis to wavelets, vol 01. Springer, Berlin
- Gurobi Optimization (2020) Gurobi optimizer reference manual,
- Hamilton JD (2017) Why you should never use the Hodrick–Prescott filter. In: Working paper 23429, National Bureau of Economic Research, May
- Hassani H (2007) Singular spectrum analysis: methodology and comparison. University Library of Munich, Germany, MPRA Paper, vol 5, p 01
- Hodrick RJ, Prescott E. (1981) Post-War U.S. business cycles: an empirical investigation. In: Discussion papers 451, Northwestern University, Center for Mathematical Studies in Economics and Management Science, May
- Liu C-L (2010) A tutorial of the wavelet transform. Duke University Community, 01
- LocalSolver (2021) Python api reference,
- Mendez-Jimenez I, Cardenas-Montes M (2018) Time series decomposition for improving the forecasting performance of convolutional neural networks. In: 18th Conference of the Spanish association for artificial intelligence. Springer International Publishing, pp 87–97
- Monsell B, Lytras D, Findley D (2013) Getting started with X-13ARIMA-SEATS input files (accessible version). *United States Census Bureau*, 08
- Montgomery DC, Jennings CL, Kulahci M (2015) Introduction to time series analysis and forecasting. John Wiley & Sons, London
- Ollech D (2018) Seasonal adjustment of daily time series. In: Bundesbank discussion paper 41/2018, Deutsche Bundesbank, Frankfurt a. M
- Persons WM (1919) Indices of business conditions. *Rev Econ Stat*, vol 1
- Qin L, Li W, Li S (2019) Effective passenger flow forecasting using STL and ESN based on two improvement strategies. *Neurocomputing* 356:05
- Terrell C (2019) Predictions in time series using regression models. *EDTECH*
- Theodosiou M (2011) Forecasting monthly and quarterly time series using STL decomposition. *Int J Forecast* 27:1178–1195
- Walker WE (1976) A heuristic adjacent extreme point algorithm for the fixed charge problem. *Manag Sci* 22(5):587–596

- Wen Q, Gao J, Song X, Sun L, Xu H, Zhu S (2019) RobustSTL: A robust seasonal-trend decomposition algorithm for long time series. In: The Thirty-Third AAAI Conference on Artificial Intelligence. AAAI Press, pp 5409–5416
- Yaffee RA, McGee M (2000) An introduction to time series analysis and forecasting with applications of SAS and SPSS. Elsevier, London
- Zhigljavsky A (2011) Singular spectrum analysis for time series. Springer, Berlin, pp 1335–1337

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.