Predicting Oil Prices using LPPL and MGPA

Jules Mourgues-Haroche Alexandre Remiat Thibault Charbonnier
Baptiste Dufour

January 2025

Université Paris Dauphine

1 Abstract

We replicated the findings from the original paper Prediction of Oil Price Using LPPL and MPGA by Fangzheng Cheng, Tijun Fan, Dandan Fan, and Shanling Li, published in 2017 in the journal Energy Economics. In addition to reproducing some of their results, we introduced several extensions, including a simplified version of the LPPL model, three new optimization algorithms, and a comprehensive analysis of other asset classes such as the SP500, SSE, and BTC. Moreover, we developed a trading strategy based on the model turning point prediction. Our findings indicate that MPGA remains the most effective algorithm for predicting WTI crude oil prices. However, for other asset classes like BTC, the Nelder-Mead and Simulated Annealing algorithms demonstrated superior performance. We also offered predictions, based on our findings, regarding the timing of the next BTC cryptocurrency crisis.

2 Introduction

Detecting financial bubbles and predicting their end is vital, particularly after the 2007 financial crisis. While the definition of a bubble remains debated in economic literature, the Log Periodic Power Law (LPPL) model, proposed by Johansen and Sornette, has become a widely used tool for identifying bubbles and forecasting their collapse. The LPPL model, based on concepts from statistical physics, assumes two types of market participants: rational traders and irrational "noise" traders. These traders interact in a way that can create self-sustained bubbles due to positive feedback loops.

This paper serves as a practical guide for modeling and detecting bubbles using the LPPL framework. It outlines the necessary steps for deriving and fitting LPPL models, discusses early criticisms and recent improvements, and presents diagnostic techniques for assessing bubble development. Through these tools, practitioners and researchers can better understand the risks of market bubbles and improve forecasting accuracy.

In this context, the LPPL model is optimized using a Multi-Population Genetic Algorithm (MGPA). Using the MGPA allows for efficient exploration of the parameter space of the LPPL model and helps overcome challenges related to nonlinear optimization. The genetic algorithm provides a flexible approach to search for robust solutions and avoid local minima, which is crucial for accurate modeling of financial bubbles. By optimizing the LPPL model parameters in this way, the accuracy of detecting and predicting market turning points is improved.

3 The Log Periodic Power Law (LPPL) Model for the Prediction of Turning Points

The Log-Periodic Power Law (LPPL) model has been widely used to capture the oscillatory behavior of speculative bubbles in financial markets. This model provides a mathematical framework to predict the critical time, T_c, when a bubble is expected to burst. It describes how asset prices evolve as a bubble grows, and how they oscillate near the turning point, which is characterized by increasing frequency and decreasing amplitude of oscillations.

The LPPL model is formulated as:

$$y(t) = A + B(T_c - t)^{\alpha} + C(T_c - t)^{\alpha} \cos(\omega \ln(T_c - t) + \phi)$$

Where:
- $y(t)$ is the asset price at time t.
- A, B, C, ϕ are the parameters with no structural information.
- T_c is the critical time (or turning point), representing the predicted time when the bubble is expected to burst.
- α is the exponent of the power law, which determines the rate of price acceleration as t approaches T_c.
- ω is the frequency of the oscillatory term, controlling the period of oscillations.
- $\ln(T_c - t)$ is the logarithmic term that introduces discrete scale invariance, helping to capture the self-similar nature of the oscillations.

As t approaches T_c, the price exhibits accelerated growth due to positive feedback mechanisms. The periodic term captures the oscillatory behavior that serves as a corrective mechanism to the power-law growth. The LPPL model provides a robust framework for forecasting potential turning points in financial markets.

4 Methods

To apply the LPPL model in real-world scenarios, the parameters need to be optimized using historical data. This is typically done by fitting the model to time series data, which involves determining the best values for the nonlinear parameters T_c, α, ω, and ϕ. These parameters can be estimated using various optimization techniques, with a particular focus on improving the accuracy of the prediction. We compare the MPGA with several metaheuristic algorithms including Particle Swarm Optimization (PSO), Simple Genetic Algorithm (SGA), Simulated annealing (SA).

4.1 Multi-Population Genetic Algorithm (MPGA) for Optimizing LPPL Parameters

The optimization of the LPPL model involves searching for the optimal values of the nonlinear parameters T_c, α, ω, and ϕ. This is a challenging task, as the search space for these parameters is complex and high-dimensional. The solution to this problem is an NP-hard problem (see proof in the appendix), which makes traditional optimization methods inefficient.

The Multi-Population Genetic Algorithm (MPGA) is an effective heuristic search method that can tackle this optimization problem by exploring the search space more thoroughly. MPGA combines several populations of candidate solutions, which helps avoid local minima and improves the convergence rate. It also uses genetic operators like selection, crossover, and mutation to explore and exploit the parameter space.

The MPGA works as follows:

1. **Initialization**: Generate an initial population of candidate solutions, each representing values for the parameters (T_c, α, ω, ϕ).

2. **Fitness Evaluation**: Evaluate the fitness of each candidate by comparing the model's predicted prices to observed prices.

3. **Selection**: Select the best candidates based on their fitness scores.

4. **Crossover**: Combine pairs of selected candidates to create new solutions through crossover.

5. **Mutation**: Introduce small random changes in the offspring to maintain diversity.

6. **Immigration**: Exchange the least fit chromosome from one population with the best fit from another, promoting diversity and optimization.

7. **Termination**: Repeat the process for several generations until convergence to an optimal or near-optimal solution.

4.2 Simple Genetic Algorithm (SGA)

The Simple Genetic Algorithm (SGA) follows an iterative process inspired by natural selection. It follows the same principal of the MPGA simply with one population.

Although the SGA is powerful for exploring a large search space, it can be slow to converge or get trapped in local optima.

4.3 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is inspired by the collective behavior of animals, such as bird flocks. Each particle represents a potential solution in the search space. Its movements are defined by:

- Its current velocity.
- The best position it has individually achieved (p_{best}).
- The best position achieved by the swarm (g_{best}).

PSO is simple to implement and converges quickly, but it can be sensitive to being trapped in local optima.

4.4 Simulated Annealing (SA)

Simulated Annealing (SA) is inspired by the annealing process of metals. It favors exploration by temporarily accepting worse solutions with decreasing probability.

SA is robust for combinatorial problems, but its performance depends heavily on the precise tuning of parameters (initial temperature, reduction factor).

4.5 Speeding Computational Time using Numba

Numba, a Just-In-Time (JIT) compiler for Python, can significantly accelerate numerical computations by translating Python code into optimized machine code. By leveraging Numba, computationally intensive sections of the LPPL model, such as fitness evaluations or parameter tuning, can be executed with improved efficiency. This enhancement reduces overall execution time without requiring extensive code modifications. Numba's compatibility with NumPy further ensures seamless integration and optimization for array-based operations, making it an ideal choice for accelerating the LPPL model's computational workflow.

4.6 Validation of Turning Points Using the Lomb Periodogram Analysis

To validate the turning points predicted by the LPPL models, it is necessary to determine whether the frequency ($\omega/2\pi$) obtained from the MPGA (where ω is one of the optimal four nonlinear parameters) and the frequency of $y(t) - A - B(tc - t)^a$ are consistent. This is done using a method called Lomb periodogram analysis, which detects periodic oscillations for $y(t) - A - B(tc - t)^a$ and calculates its frequency.

The Lomb periodogram analysis is preferred over other existing methods (e.g., Shanks transform, $(H; q)$-analysis, Fourier transform) because it overcomes some of their major weaknesses. For instance, the Shanks transform method can be subjective when identifying the critical time t_c, and the Fourier transform assumes uniform time series, which is not always the case. In contrast, the Lomb periodogram is suitable for non-uniform time series and provides an objective evaluation of critical time t_c and turning points.

The validation process starts with pre-setting the frequency series freq_i $(i = 1, 2, \ldots, M)$, where M is the length of the given frequency series. For a given frequency f, the power spectral density $P(f)$ is computed using the Lomb periodogram analysis as follows:

$$P(f) = \frac{1}{2\sigma^2}\left[\frac{\left(\sum_{j=1}^{J}(x_j - \bar{x})\cos\left(2\pi f(t_j - \tau)\right)\right)^2}{\sum_{j=1}^{J}\cos^2\left(2\pi f(t_j - \tau)\right)} + \frac{\left(\sum_{j=1}^{J}(x_j - \bar{x})\sin\left(2\pi f(t_j - \tau)\right)\right)^2}{\sum_{j=1}^{J}\sin^2\left(2\pi f(t_j - \tau)\right)}\right]$$

where $x_j = y_j - A - B(t_c - t_j)^a$, $\bar{x}$ is the mean of x_j, and σ^2 is the variance of x_j. The time offset t is calculated as:

$$t = \frac{1}{4\pi f}\arctan\left(\sum_{j=1}^{J}\sin(4\pi f t_j) \Big/ \sum_{j=1}^{J}\cos(4\pi f t_j)\right)$$

Invalid values are then removed from the resulting $P(\text{freq}_i)$ series ($i = 1, 2, \ldots, M$). These invalid values include: 1. $P(f_{\text{mpf}})$ corresponding to the most probable frequency (f_{mpf}), which is caused by random series and is inversely proportional to the length of the given frequency series (J), i.e., $f_{\text{mpf}} \approx 1.5/J$; 2. $P(\text{freq}_i)$ values that are smaller than the critical value, which is computed by $z = -\ln(1 - (1 - p)^{1/M})$, at a given statistical significance level p.

If there are no valid values in the $P(\text{freq}_i)$ series, the Lomb periodogram rejects the conclusion. In other words, the turning points predicted by the LPPL model are not statistically valid. Otherwise, the frequency corresponding to the maximum valid value in the $P(\text{freq}_i)$ series is considered the result of the Lomb periodogram test.

The Lomb periodogram analysis can be summarized as follows:

1. First, an LPPL model corresponding to each subinterval is obtained.
2. The Lomb periodogram analysis computes the frequency value based on the periodic oscillations of the LPPL model.
3. If the frequency value is close to the frequency ($\omega/2\pi$) optimized by the MPGA, the Lomb periodogram analysis concludes that the LPPL model prediction is effective.
4. If not, the predicted turning points are invalid and deleted.

Finally, only the turning points predicted by the LPPL models that pass the Lomb periodogram test are recorded.

4.7 Creation of 30-Day Window Periods

To enable a more flexible approach to predicting turning points, we selected the ten most significant turning points, weighted by their power-law frequency. For the turning point, we extended the range by subtracting fifteen days and adding fifteen days, thereby creating a thirty-day prediction window.

4.8 Bounds and algorithm parameters

As detailed in Section 10.6, the parameter bounds and algorithm configurations are specified for the LPPL optimization.

5 Data

Our data include the WTI spot price from January, 2, 1986 until December 12th 2024, the data is daily and weekly and was retrieve on the U.S Energy Information Administration https://www.eia.gov/dnav/pet/hist/RWTCD.htm

6 Results

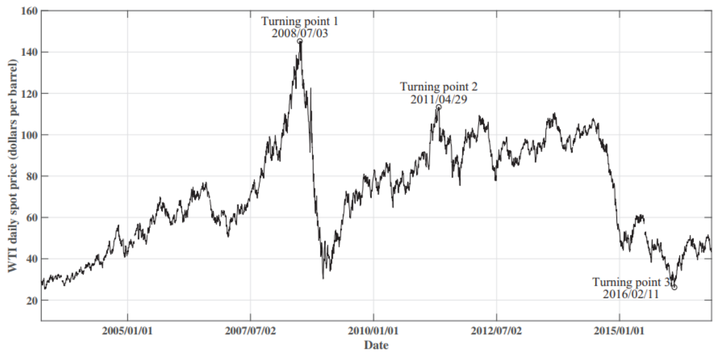
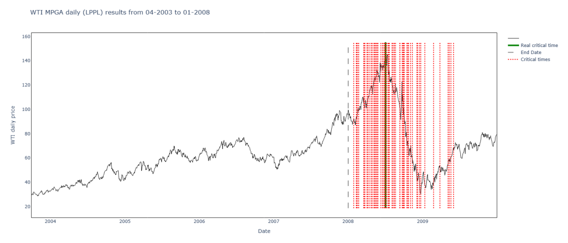
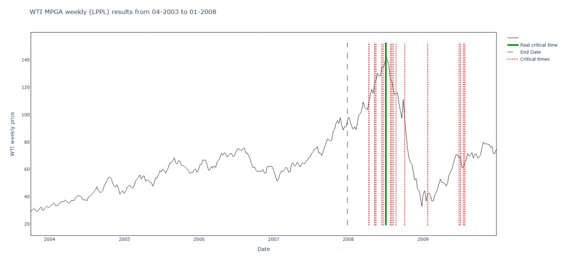


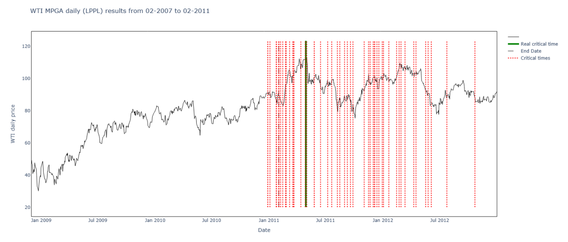
Figure 1: The WTI spot price between April 1, 2003 and November 14, 2016.



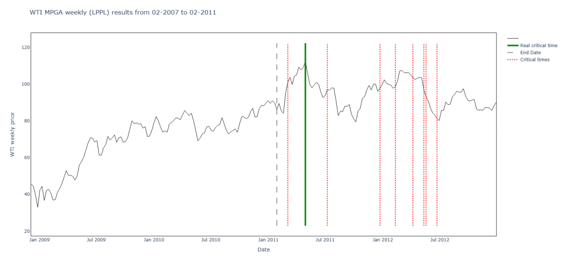
(a) Comparison of daily data all T_c of April 1, 2003 to January 2, 2008.



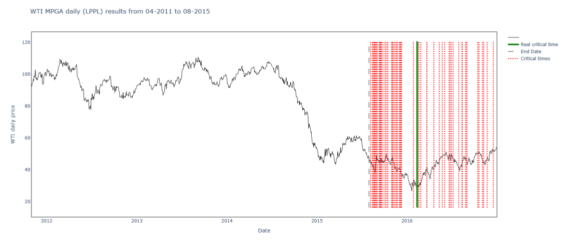
(d) Comparison of weekly data all T_c of February 1, 2007 to February 1, 2011



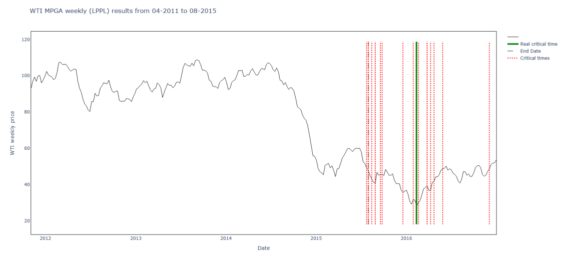
(b) Comparison of daily data all T_c of February 1, 2007 to February 1, 2011.



(e) Comparison of weekly data all T_c of February 1, 2007 to February 1, 2011



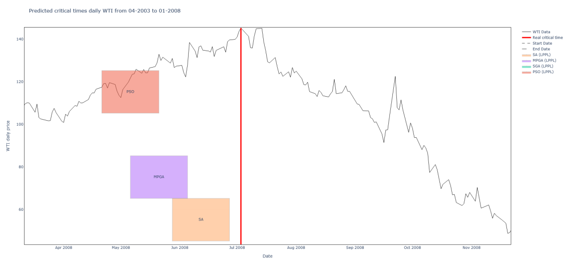
(c) Comparison of daily data all T_c of April 29, 2011 to August 1, 2015.



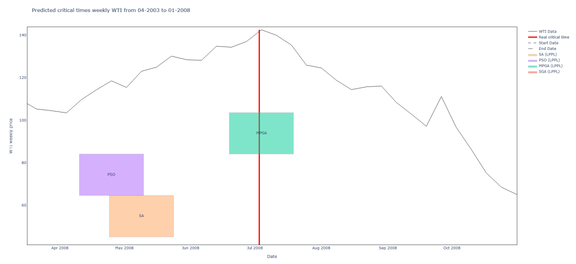
(f) Comparison of weekly data all T_c of April 29, 2011 to August 1, 2015

Figure 2: Comparison of LPPL predictions showing all significative T_c (left side is daily data and right side is weekly data).

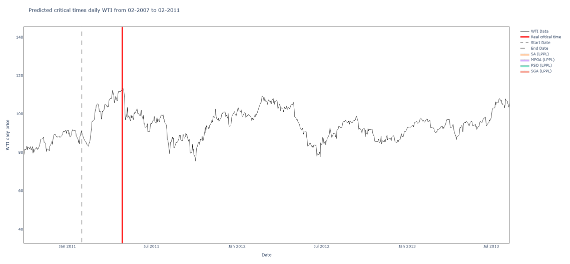
Overall, the prediction identified a numerous number of significant turning points close to the actual T_c, highlighting the need for filtering. Therefore, we decided to weigh the ten most significant $T_c$values based on their power law value, then subtract fifteen days before and add fifteen days after the mean of these points, to create our predicting squares periods.



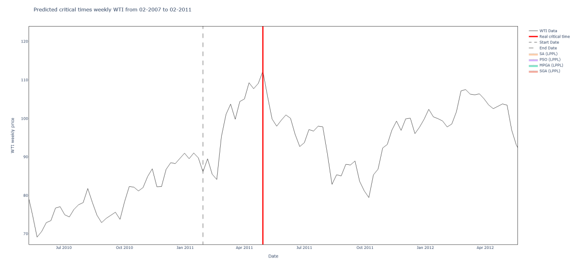
(a) Prediction result of daily data of April 1, 2003 to January 2, 2008



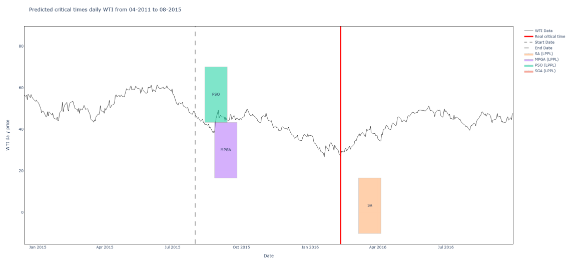
(d) Prediction result of weekly data of April 1, 2003 to January 2, 2008



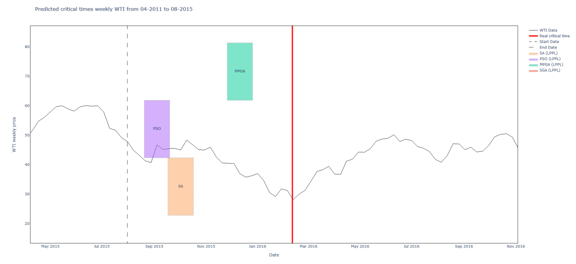
(b) Prediction result of daily data of February 1, 2007 to February 1, 2011



(e) Prediction result of weekly data of February 1, 2007 to February 1, 2011



(c) Prediction result of daily data of April 29, 2011 to August 1, 2015



(f) Prediction result of weekly data of April 29, 2011 to August 1, 2015

Figure 3: Comparison of LPPL predictions (left side is daily data right side is weekly data).

The results showed that PSO, MPGA and SA works best for the first and last turnings points. SGA couldn't predict even closely to the real T_c. Graph d showed MPGA predicted the exact month the turning point happened.

Only one turning point prediction can be considered true, even though PSO and SA showed great results predicting the turning point only 2 months prior to the real one. In their paper, the authors found out that MPGA was superior to all other optimization algorithms followed by SA, SGA and PSO. In our work, MPGA is still superior to the other algorithms but PSO showed good results compare to what they found.

Table 1: Comparison of prediction results of different heuristic algorithms.

Optimization Method	**Turning Point 1**	**Turning Point 2**	**Turning Point 3**
Historical Turning Point	**2008/07/03**	**2011/04/29**	**2016/02/11**
30 days optimized by MPGA	False (2008/05/06–2008/06/05)	False (2019/11/04–2019/12/04)	False (2015/08/26–2015/09/25)
4 weeks optimized by MPGA	True (2008/06/19–2008/07/19)	False (2012/07/26–2012/08/25)	False (2015/11/26–2015/12/26)
30 days optimized by SGA	False (2010/06/13–2010/07/13)	False (2019/07/11–2019/08/10)	False (2018/07/02–2018/08/01)
4 weeks optimized by SGA	False (2010/05/20–2010/06/19)	False (2013/12/19–2014/01/18)	False (2019/11/14–2019/12/14)
30 days optimized by SA	False (2008/05/25–2008/06/25)	False (2019/11/14–2019/12/14)	False (2016/02/22–2016/03/22)
4 weeks optimized by SA	False (2008/04/25–2008/05/25)	False (2016/06/02–2016/07/02)	False (2015/09/13-2015/10/12)
30 days optimized by PSO	False (2008/04/21–2008/05/21)	False (2021/01/11-2021/02/10)	False (2015/08/13-2015/09/12)
4 weeks optimized by PSO	False (2008/04/10–2008/05/10)	False (2013/11/28–2013/12/28)	False (2015/08/20–2015/09/19)

6.1 USO (United State Oil Fund) Analysis

The prediction of turning points of oil price also has some implications of the stock price in energy sector in the financial market since the fluctuations in oil price will inevitably lead to the volatility in equity holdings of oil companies. To verify the link between the two, daily and weekly data of United States Oil (USO) Fund, an exchange-traded fund, in the period between April 10, 2006 and November 14, 2016 were retrieved from Yahoo Finance and analyzed.

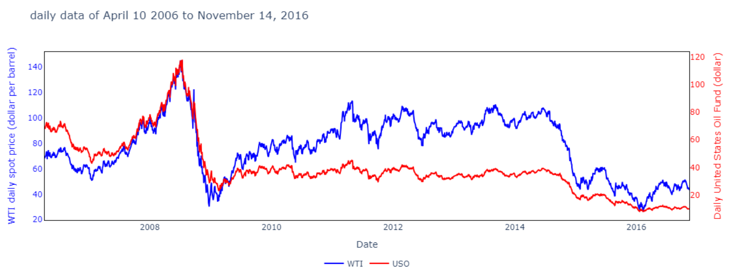


Figure 4: Daily Price of USO (red) and WTI (blue) from April 10, 2006 to November 14, 2016

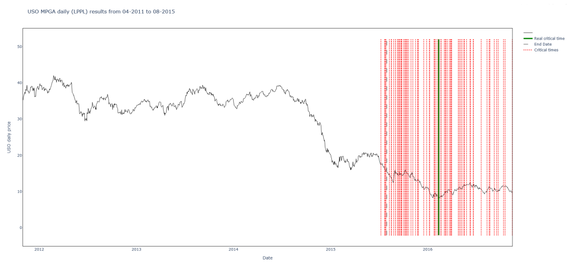


Figure 5: Comparison of daily data USO all T_c of April 29, 2011 to August 1, 2015.

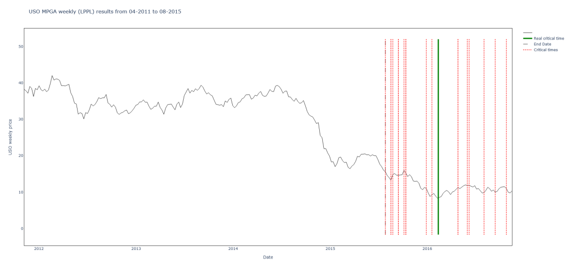


Figure 6: Comparison of weekly data USO all T_c of April 29, 2011 to August 1, 2015.

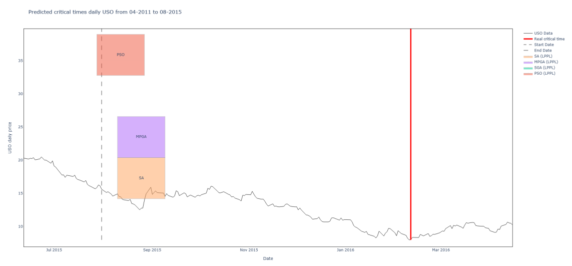


Figure 7: Comparison of daily data USO of April 29, 2011 to August 1, 2015.

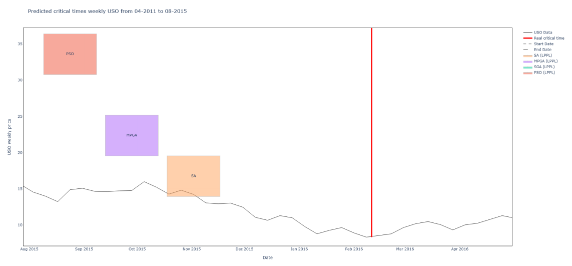


Figure 8: Comparison of weekly data USO of April 29, 2011 to August 1, 2015.

6.2 Conclusion USO analysis

The predicted turning point for the United States Oil Fund confirms the validity of the turning point forecasted for the WTI spot price. Additionally, both predictions align closely with actual historical turning points, demonstrating the robustness of the LPPL model. Lastly, the findings highlight a strong relationship between the WTI spot price, the United States Oil Fund, and the turning point predicted by the LPPL model.

7 Paper replication conclusion

We successfully replicated some of their results, although we were unable to achieve identical outcomes. The lack of detailed information regarding the application of the metaheuristic algorithm, the creation of the thirty-day squared windows, and the rationale behind the selection of tuning point dates presented challenges in fully replicating the original results. Nevertheless, our analysis demonstrated that PSO, MPGA, and SA were the most effective methods for predicting oil prices turnings points.

8 Extensions

8.1 New Optimization Algorithms

Tabu Search

Tabu Search is a metaheuristic optimization algorithm designed to find the global minimum or maximum of a function. It enhances local search methods by avoiding cycles and escaping local optima through the use of a *tabu list*, which temporarily prohibits revisiting recently explored solutions.

Firefly Optimization Algorithm

The Firefly Optimization Algorithm is a swarm intelligence technique inspired by the flashing behavior of fireflies. It is based on the idea that fireflies are attracted to brighter individuals, where brightness corresponds to the objective function value.

8.2 Simplified LPPL (LPPLS

The proposed method aims to reduce the number of nonlinear parameters and eliminate the interdependence between the phase φ and the angular log-frequency ω. This is achieved by transforming the phase φ into two new parameters, C_1 and C_2, simplifying the formulation of the LPPL equation.

The original LPPL equation is expanded as:

$$\ln E[p(t)] = A + B(tc-t)^{\alpha} + C(tc-t)^{\alpha}\cos(\omega\ln(tc-t))\cos\varphi + C(tc-t)^{\alpha}\sin(\omega\ln(tc-t))\sin\varphi.$$

By introducing the new parameters:

$$C_1 = C\cos\varphi, \quad C_2 = C\sin\varphi,$$

the LPPL equation is rewritten as:

$$\ln E[p(t)] = A + B(tc-t)^\alpha + C_1(tc-t)^\alpha \cos(\omega\ln(tc-t)) + C_2(tc-t)^\alpha \sin(\omega\ln(tc-t)).$$

In this new form, the LPPL function has only 3 nonlinear parameters (tc, ω, m) and 4 linear parameters (A, B, C_1, C_2).

To estimate the parameters, a least-squares method is used with the cost function:

$$F(tc,\alpha,\omega,A,B,C_1,C_2) = \sum_{i=1}^{N} \left[\ln p(\tau_i) - A - B(tc-\tau_i)^\alpha - C_1(tc-\tau_i)^\alpha \cos(\omega\ln(tc-\tau_i)) - C_2(tc-\tau_i)^\alpha \sin(\omega\ln(tc-\tau_i))\right]^2 .$$

Slaving the linear parameters to the nonlinear ones, we obtain the optimization problem:

$$\hat{t_c}, \hat{\alpha}, \hat{\omega} = \arg\min_{tc,\alpha,\omega} F_1(tc,\alpha,\omega),$$

where the cost function $F_1(tc,\alpha,\omega)$ is:

$$F_1(tc,\alpha,\omega) = \min_{A,B,C_1,C_2} F(tc,\alpha,\omega,A,B,C_1,C_2).$$

The optimization problem for the linear parameters has a unique solution obtained from a matrix equation:

$$\begin{pmatrix} \sum_i f_i & \sum_i g_i & \sum_i h_i \\ \sum_i f_i & \sum_i f_i^2 & \sum_i f_i g_i \\ \sum_i g_i & \sum_i f_i g_i & \sum_i g_i^2 \\ \sum_i h_i & \sum_i f_i h_i & \sum_i g_i h_i \end{pmatrix} \begin{pmatrix} \hat{A} \\ \hat{B} \\ \hat{C_1} \\ \hat{C_2} \end{pmatrix} = \begin{pmatrix} \sum_i y_i \\ \sum_i y_i f_i \\ \sum_i y_i g_i \\ \sum_i y_i h_i \end{pmatrix}.$$

This modification leads to two important results:

- The dimensionality of the nonlinear optimization problem is reduced from a 4-dimensional space to a 3-dimensional space, reducing the complexity.
- The phase φ is no longer an independent parameter, eliminating the quasi-periodicity of the cost function. This simplifies the optimization process, making it possible to use rigorous search methods instead of metaheuristic ones.

The new formulation results in a smoother cost function, with only a few minima, which can easily be found using local search methods like the Levenberg-Marquardt algorithm or the Nelder-Mead simplex method.

Nelder-Mead Optimization Algorithm

The Nelder-Mead algorithm is a numerical optimization technique used to find the minimum (or maximum) of a scalar objective function $f(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^n$. Unlike gradient-based methods, it does not require the computation of derivatives, making it suitable for non-linear, non-differentiable, or noisy functions.

8.3 Extensions results

In this section, we analyze the results of the LPPLS model applied to WTI data using various algorithms. The figures illustrate comparisons of daily and weekly data across different time periods.

The analysis of the data reveals consistent patterns over the years, with detailed comparisons for specific periods available in the Appendix (e.g., Figure 12 for data from April 2003 to January 2008). The newly introduced algorithms performed well, with the Tabu optimization achieving similar prediction results to MPGA, PSO, and SA on daily data. However, on weekly data, MPGA continued to outperform the others.

Tabu algorithms demonstrated promising results for both daily and weekly data during the 2011–2015 period, as shown in Figure 13.

Figure 14 presents daily data from April 1, 2003, to January 2, 2008, using various optimization algorithms with the simplified LPPLS model. Nelder-Mead optimization produced excellent results, aligning with expectations, as it excels in searching simpler functions a key reason for applying the LPPLS model. Other algorithms behaved similarly to their performance with LPPL, predicting the turning point slightly earlier. Despite this, taking a position based on their predictions would still allow for capitalizing on the subsequent market crash.

For a focus on weekly data, Figure 15 compares the results from February 1, 2007, to February 1, 2011. Unfortunately, none of our algorithms or models successfully predicted the second turning point in this period. It is possible that our algorithms were identifying a turning point closer to July 2014 instead.

Lastly, Figure 16 illustrates the daily data from April 29, 2011, to August 1, 2015. Both Tabu and Nelder-Mead optimization methods demonstrated promising results, aligning closely with MPGA and PSO by predicting the same turning point.

8.4 SP500 analysis with LPPLS

The SP500 dataset has been analyzed using the LPPLS methodology to uncover underlying patterns and predict potential market turning points. Figure 17 presents the daily log prices of the SP500 index, offering a comprehensive view of its behaviour over time.

To focus on specific periods, Figure 18 highlights the daily data from March 1995 to March 2000, capturing significant market dynamics during the tech bubble era, our optimization algorithms using LPPLS, specially PSO, MPGA and Nelder-Mead successfully capture some of the downtrend price movement, predicting thirty days after the real turning point. Additionally, Figure 19 examines the daily data from March 2004 to October 2009, showed that SA, TABU, MPGA and PSO were close to predicting the real turning point.

8.5 SSE analysis with LPPLS

The SSE (Shanghai Stock Exchange) data was also examined for similar insights. Figure 20 displays the daily log prices of the SSE index. A deeper analysis of the daily data from April 2004 to April 2007 is shown in Figure 21, none of the results were promising enough to predict the real turning point.

8.6 BTC analysis with LPPLS

The analysis of Bitcoin (BTC) using the LPPLS model provided insights into the cryptocurrency's volatile behavior. Figure 22 shows the daily log prices of BTC over an extended period.

The LPPLS predictions were applied to various timeframes:

Figure 23 and Figure 24 present the daily data from January 2013 to November 2017 and from March 2018 to September 2021, respectively. The weekly data for the same periods is analyzed in Figure 25 and Figure 26, showcasing broader trends and potential turning points.

On three out of these four graphs, SA predicted the real turning point. PSO, Tabu and Nelder-Mead were able to predict one out of four turning point.

Finally, Figure 27 highlights the LPPLS model's prediction of the next turning point in BTC, starting from January 2022, 2025 and the beginning of 2026, seems safe for the crypto world, however based on the hit rate of the algorithms Nelder Mead and Tabu predict an end of 2026/early 2027 crypto crash while SA predicts a 2028 crypto crash.

Trading Strategy on BTC

Trading Strategy Explanation

- We use the LPPLS model to identify significant market turning points.
- The strategy is primarily positioned long, aiming to benefit from overall market growth.
- **Short Position:**
  - Enter a short position when the model predicts a significant turning point within the next 30 days.
- **Exit Short:**
  - Close the short position after 30 days or earlier if market conditions indicate a change.
- This approach avoids periods of intense downturns while capitalizing on favorable market movements.

The LPPLS-based trading strategy demonstrates its effectiveness in navigating Bitcoin's market volatility. By identifying key turning points, the strategy avoids significant downturns and captures favorable trends. As shown in Figure 28, this approach achieves consistent performance over time. During the 2021 bitcoin crash, the model correctly predicted the crash and entered a short position, creating a huge performance gap.

9 Conclusion

This study explored the predictive capabilities of various heuristic algorithms, including PSO, MPGA, SA, and SGA, for identifying turning points in financial markets using the LPPL and LPPLS models. The analysis revealed that MPGA consistently outperformed other methods, with PSO and SA also delivering competitive results in certain scenarios. In contrast, SGA struggled to provide accurate predictions.

The replication of prior research demonstrated that while we achieved comparable results, challenges in fully replicating the methodologies due to incomplete information on the implementation details highlight the complexity of optimizing these models. However, our extensions using new optimization methods such as Tabu Search and Nelder-Mead showcased promising outcomes, further refining the LPPL and LPPLS models predictive power.

Oil Markets (WTI and USO): Predictions aligned well with historical turning points, underscoring the robustness of the LPPL model. MPGA, PSO, and SA demonstrated strong performance, confirming their applicability to oil price fluctuations.

Equity Markets (SP500 and SSE): The LPPLS model captured significant market movements, with optimization algorithms like SA, Tabu Search, and Nelder-Mead successfully predicting critical downtrends. However, in certain periods, particularly in the SSE dataset, results were less conclusive.

Cryptocurrency Markets (BTC): BTC predictions emphasized the LPPLS model's adaptability to highly volatile markets. Notably, SA showed the highest accuracy, while other methods provided mixed results. Predictions for future market movements suggest a potential cryptocurrency crash between late 2026 and

2028, depending on the algorithm.

The extensions to the LPPLS model, including parameter simplifications and the use of advanced optimization techniques, significantly improved the model's computational efficiency and reduced optimization complexity. These enhancements enabled more accurate predictions, particularly when using methods like Nelder-Mead for smoother cost functions.

Overall, this research reinforces the utility of LPPL and LPPLS models in understanding market dynamics and offers a pathway for further improvement through novel optimization approaches. The ability to predict turning points reliably in various financial markets positions these models as vital tools for risk management and investment decision-making.

References

[1] Cheng, Y., Wang, X., Li, X., & Zhang, W. (2017). *The prediction of oil price turning points with log-periodic power law and multi-population genetic algorithm.* Energy Economics, 64, 47-57.

[2] Sornette, D., & Johansen, A. (2011). *A Stable and Robust Calibration Scheme of the Log-Periodic Power Law Model.*

[3] Johansen, A., Sornette, D., & Ruan, J. (2000). *The Power of Log-Periodic Scaling and the Prediction of Financial Bubbles.* International Journal of Theoretical and Applied Finance, 3(2), 229–267.

[4] Sornette, D., Johansen, A., & Bouchaud, J. P. (2004). *Stock market crashes, Precursors and prediction: Towards a unified theory of speculative bubbles.* Physica A: Statistical Mechanics and its Applications, 299(1-2), 256–275.

[5] Hamilton, J. D. (2009). *Causes and Consequences of the Oil Shock of 2007–08.* Brookings Papers on Economic Activity, 2009(1), 215–283.

[6] Wang, Y., & Li, J. (2015). *Prediction of Turning Points in Oil Prices Using LPPLS Model.* Energy Economics, 50, 221-229.

[7] Sornette, D., & Zhou, W. X. (2005). *From bubble to crash: Dynamics of the 1997–1998 Asian financial crisis.* International Journal of Modern Physics C, 16(08), 1237-1264.

[8] Sornette, D. (2003). *Why Stock Markets Crash: Critical Events in Complex Financial Systems.* Princeton University Press.

[9] Gao, J., Li, X., & Liu, J. (2016). *Turning Point Prediction of Financial Time Series Using Log-Periodic Power Law Singularities.* Physica A: Statistical Mechanics and its Applications, 448, 107–115.

[10] Sornette, D., & Sornette, A. (2007). *The critical market and the role of financial leverage in predicting bubbles and crashes.* Physica A: Statistical Mechanics and its Applications, 383(2), 430-446.

10 Appendix

Proof: The optimization of the four nonlinear parameters in the LPPL model is an NP-hard problem.

Definition. A class of functions F is closed if it satisfies the following four conditions:

1. F contains all linear functions;
2. F is closed under addition, i.e., if $f \in F$ and $g \in F$, then $f + g \in F$;
3. F is closed under multiplication by a positive constant, i.e., if $f \in F$ and $c > 0$, then $c \cdot f \in F$;

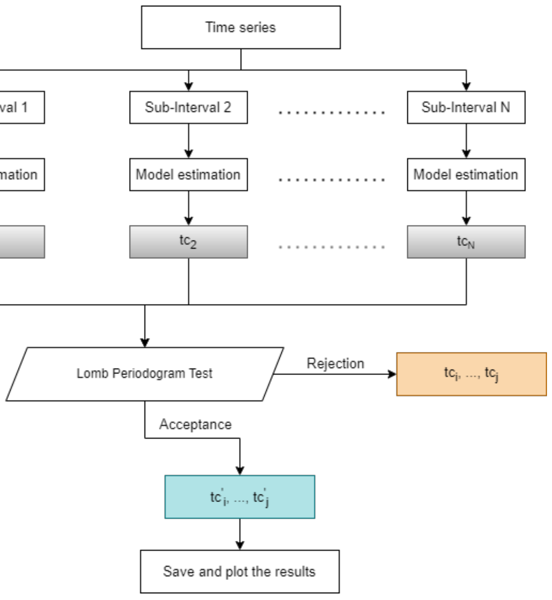
4. F is closed under linear substitution, i.e., whenever $f(x_1, \dots, x_k) \in F$ and c_{i_k} are real numbers, we have

$$f\left(c_{10} + c_{11}x_1 + \cdots + c_{1n}x_n + \cdots + c_{k0} + c_{k1}x_1 + \cdots + c_{kn}x_n\right) \in F.$$

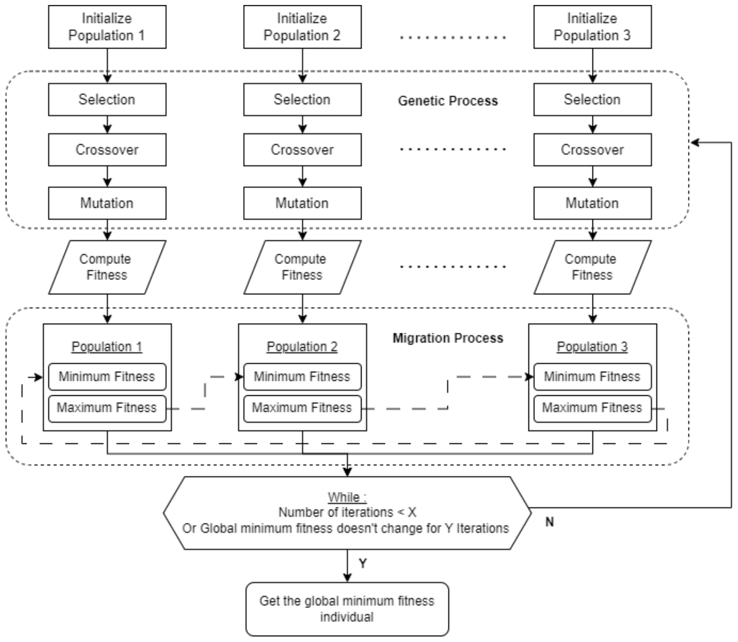
Proof. According to Kreinovich and Kearfott (2005), if a closed class of functions F contains at least one non-convex function and at least one nonlinear convex function, then for this class, the problem of finding the minimum of a given function $f \in F$ on a given feasible region is NP-hard.

Note that in Equation (1), the power law term, $f_j = (t_c - t_j)^a$, is a nonlinear convex function, and the periodic term $g_j = (t_c - t_j)^a \cos[y \ln(t_c - t_j) + 0]$, is a non-convex function. Since Equation (1) can be considered as a closed class and it has at least one nonlinear convex power law term and at least one non-convex periodic term, we conclude that the optimization of the four nonlinear parameters in the LPPL model is an NP-hard problem.

Turning points prediction in practice



MPGA process



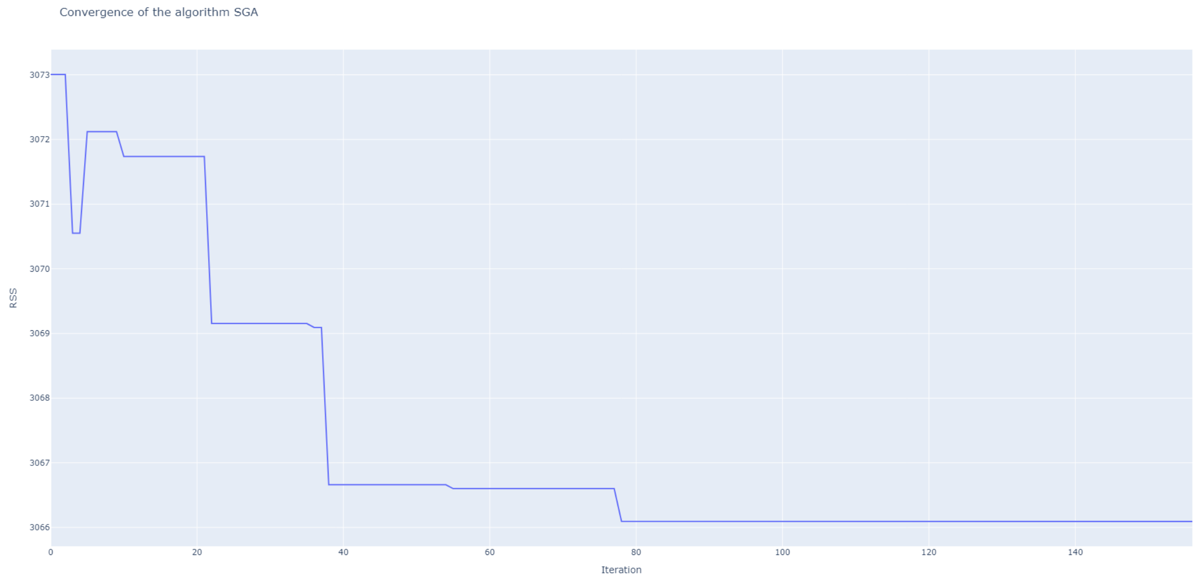


Figure 9: Convergence of SGA algorithm

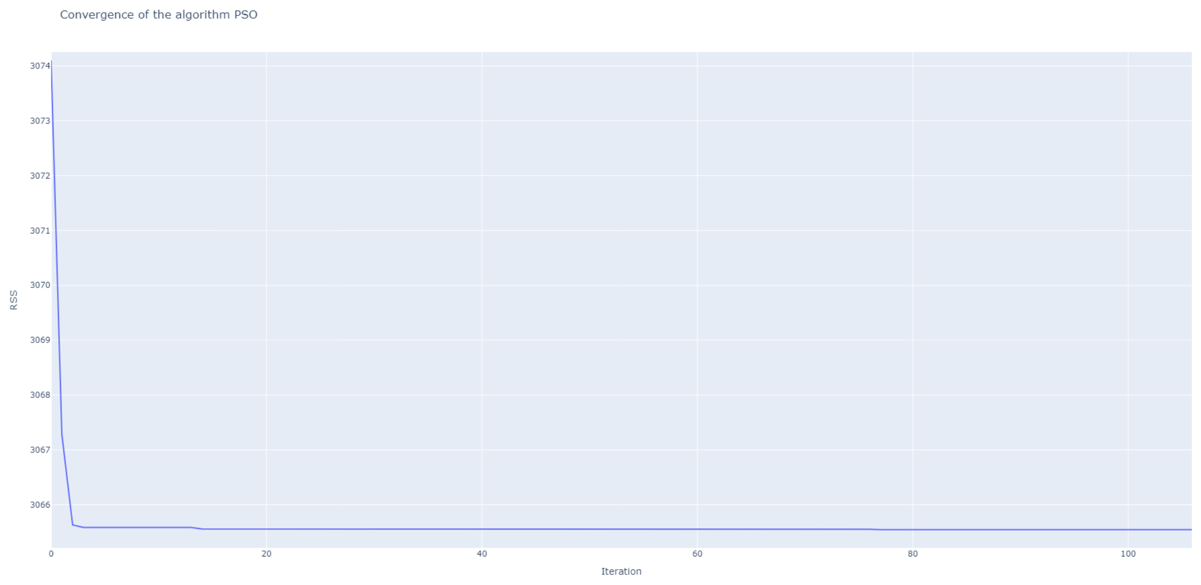


Figure 10: Convergence of PSO algorithm

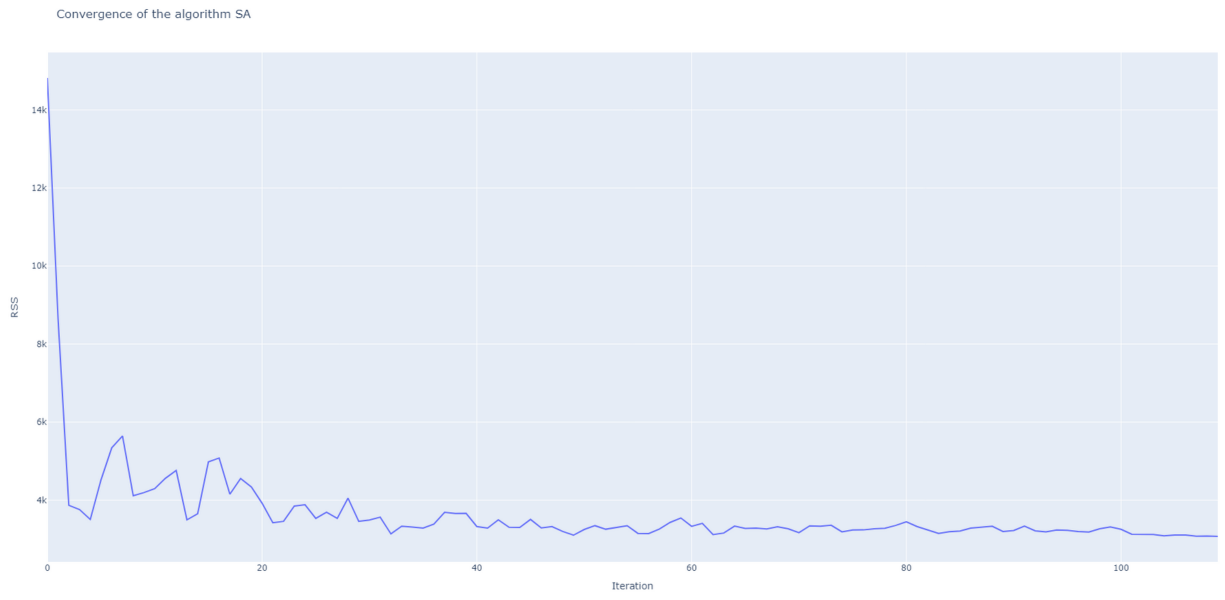


Figure 11: Convergence of SA algorithm

10.1 New optimization algorithms with LPPL

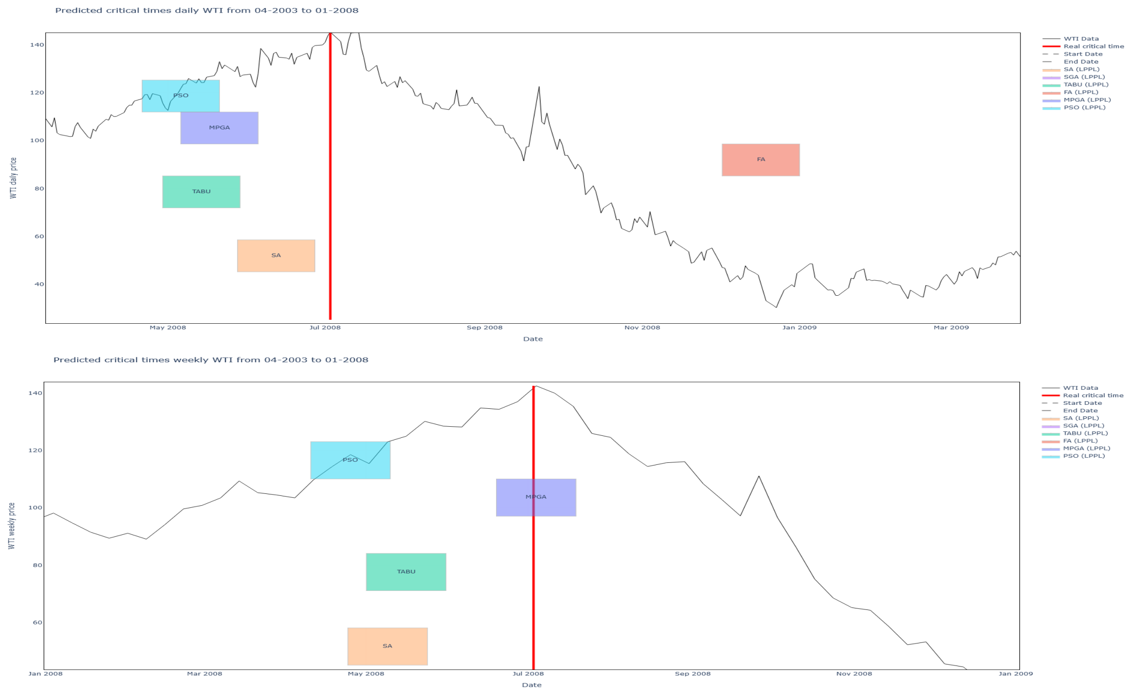


Figure 12: Comparison of data from April 2003 to January 2008 using new algorithms TABU, FA (firefly algorithm), on LPPL model.

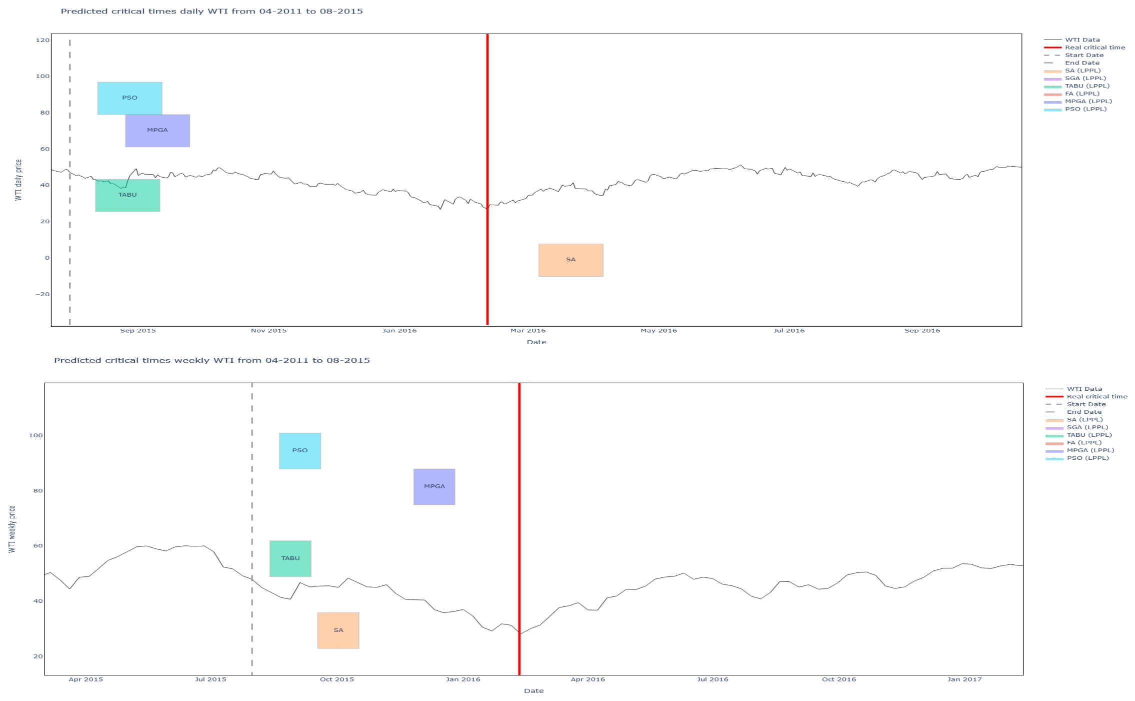


Figure 13: Comparison of data from April 2011 to August 2015 using new algorithms TABU, FA (firefly algorithm), on LPPL model.

10.2 LPPLS all optimizations algorithms

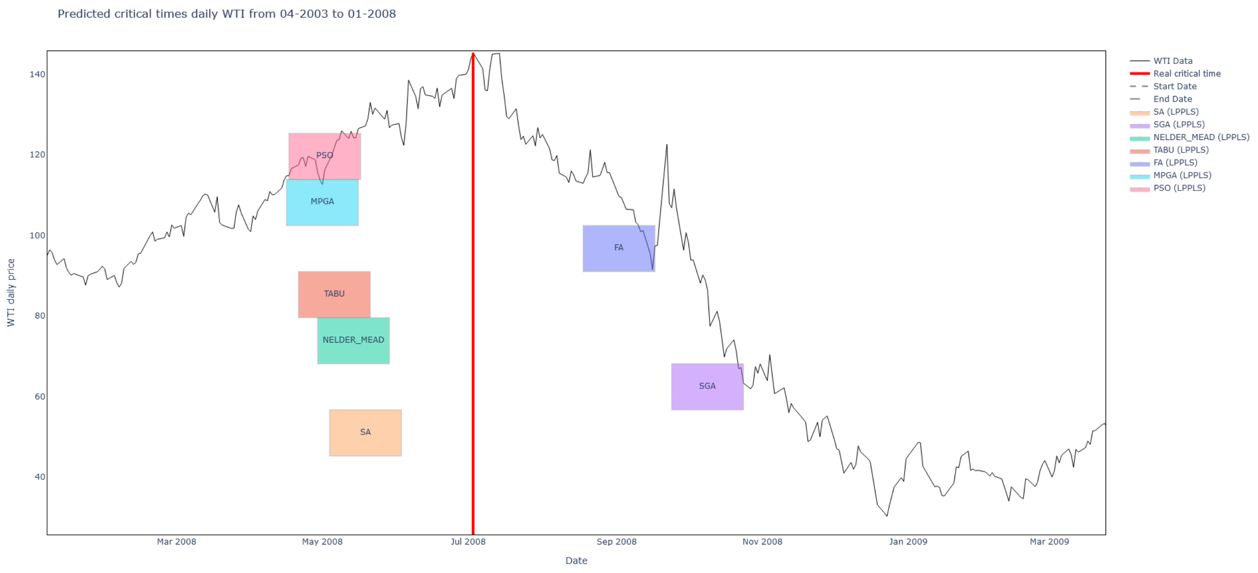


Figure 14: Comparison of daily data from April 1, 2003 to January 2, 2008, using LPPLS model.

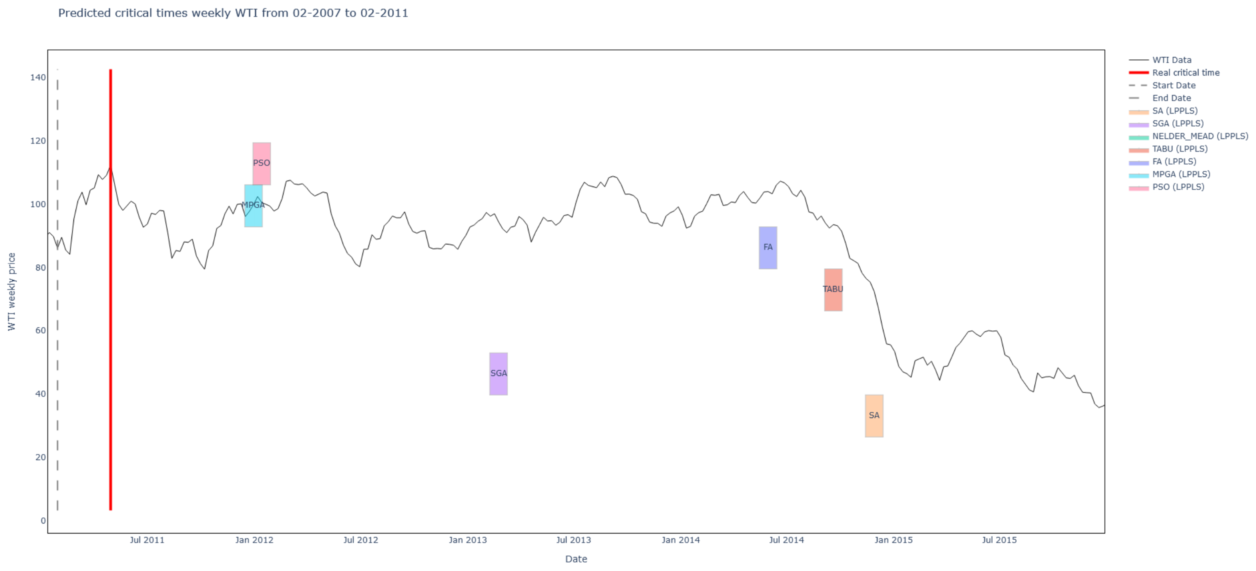


Figure 15: Comparison of weekly data from February 1, 2007 to February 1, 2011, using LPPLS model.

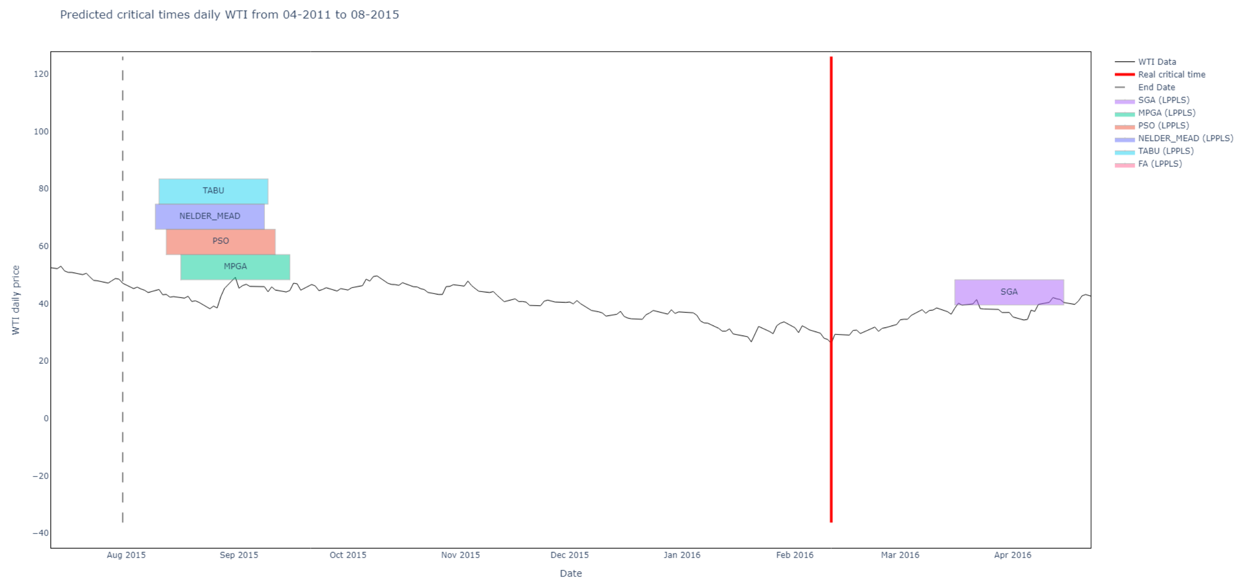


Figure 16: Comparison of daily data from April 29, 2011 to August 1, 2015.

10.3 SP500 analysis with LPPLS

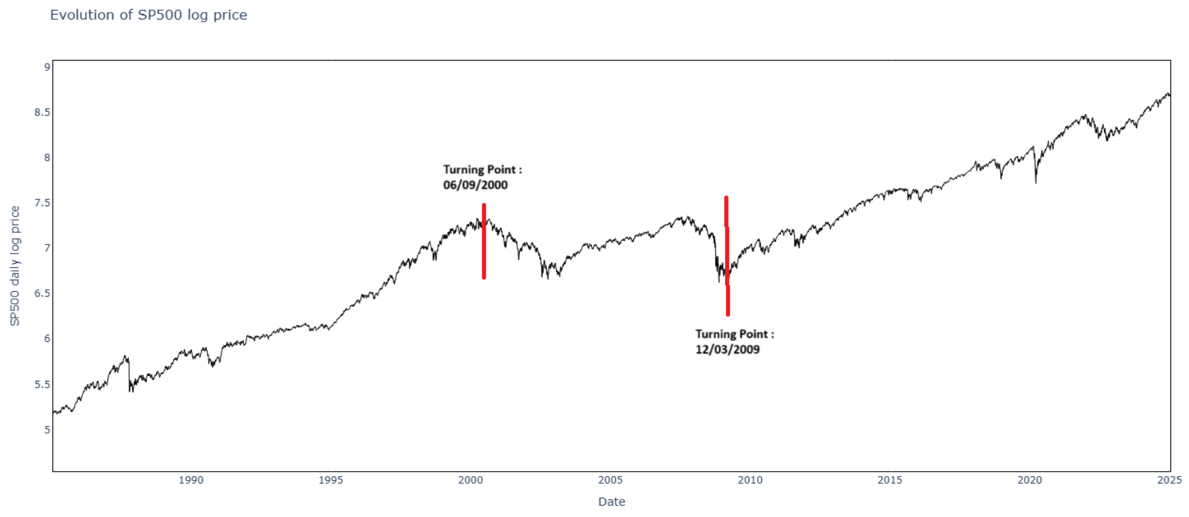


Figure 17: SP500 daily log prices.

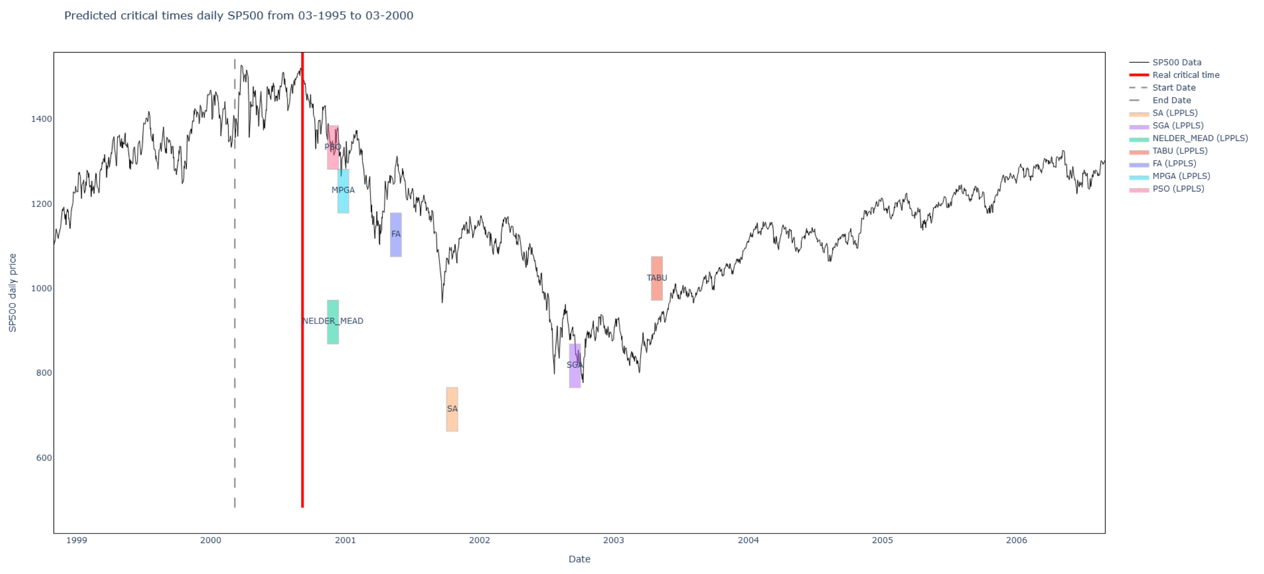


Figure 18: Comparison of daily data from March 1995 to March 2000.

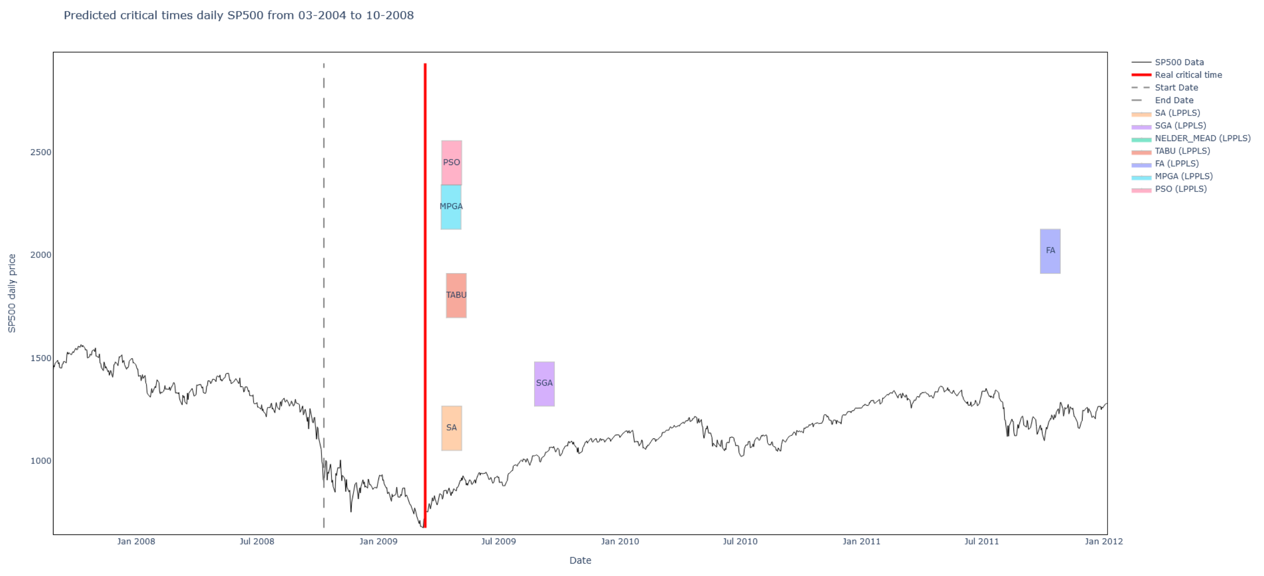


Figure 19: Comparison of daily data from March 2004 to October 2009.

10.4 SSE analysis with LPPLS

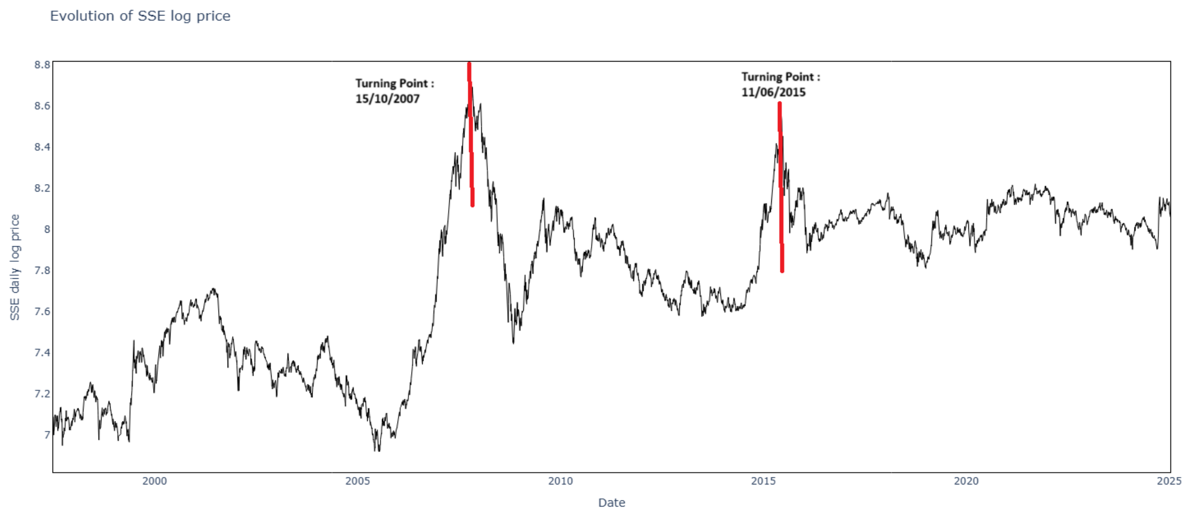


Figure 20: SSE daily log prices.

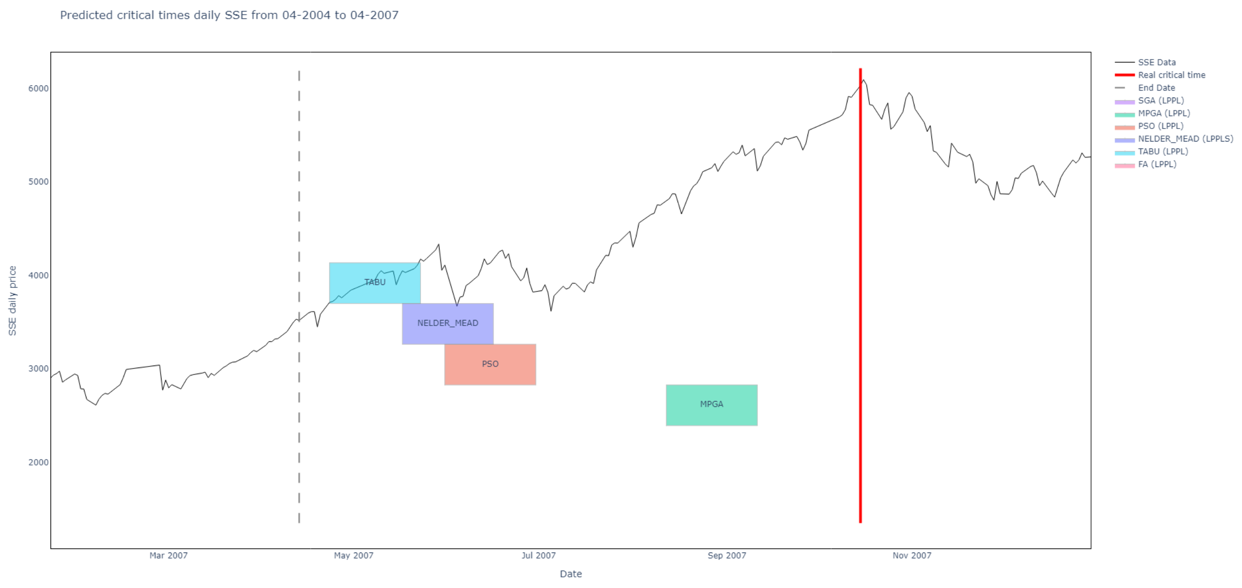


Figure 21: Comparison of daily data from April 2004 to April 2007.

10.5 BTC analysis with LPPLS

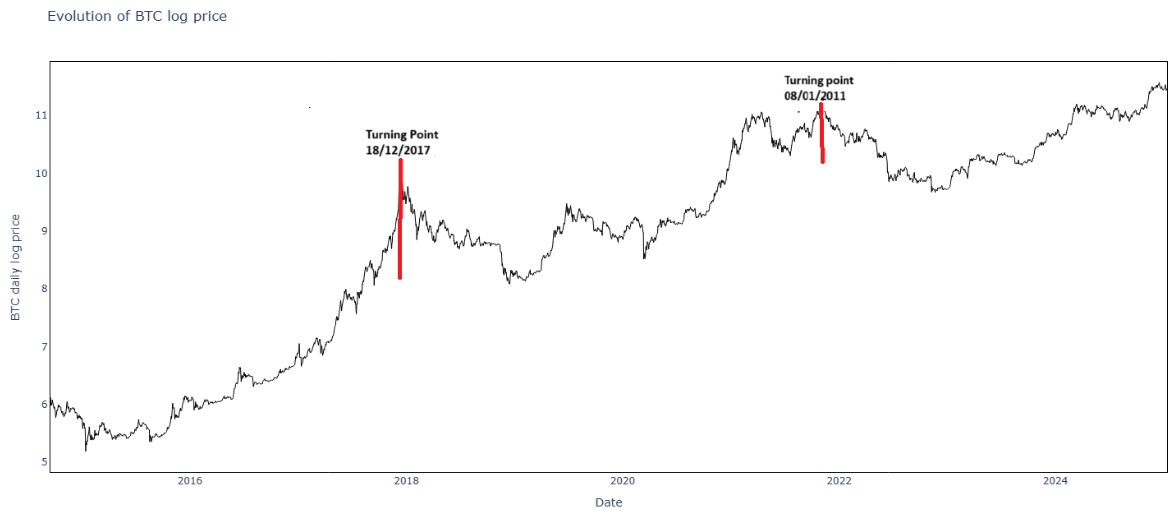


Figure 22: BTC daily log prices.

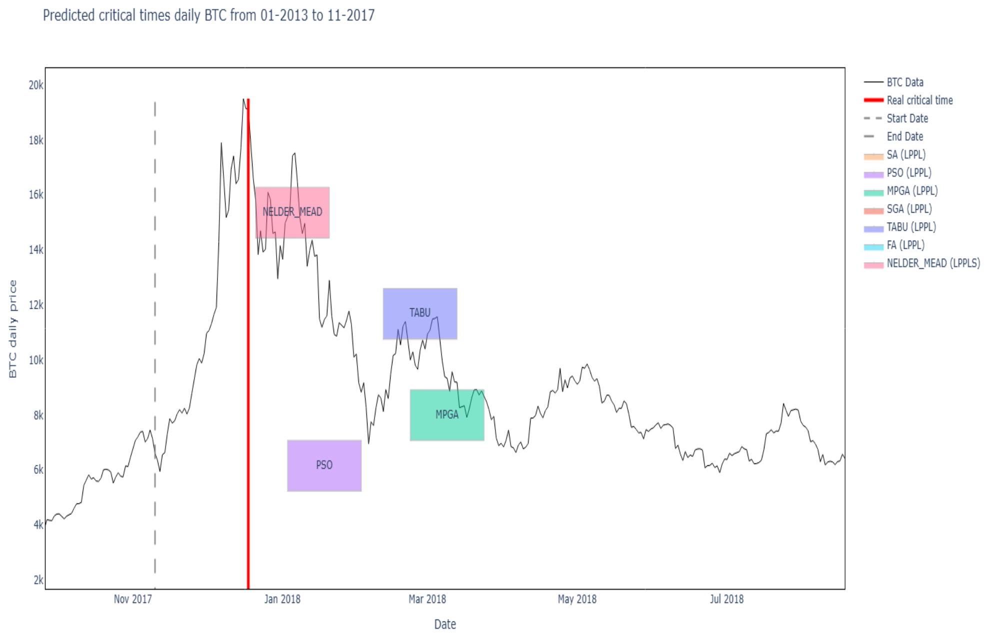


Figure 23: Prediction result of LPPL daily data from January 2013 to November 2017.

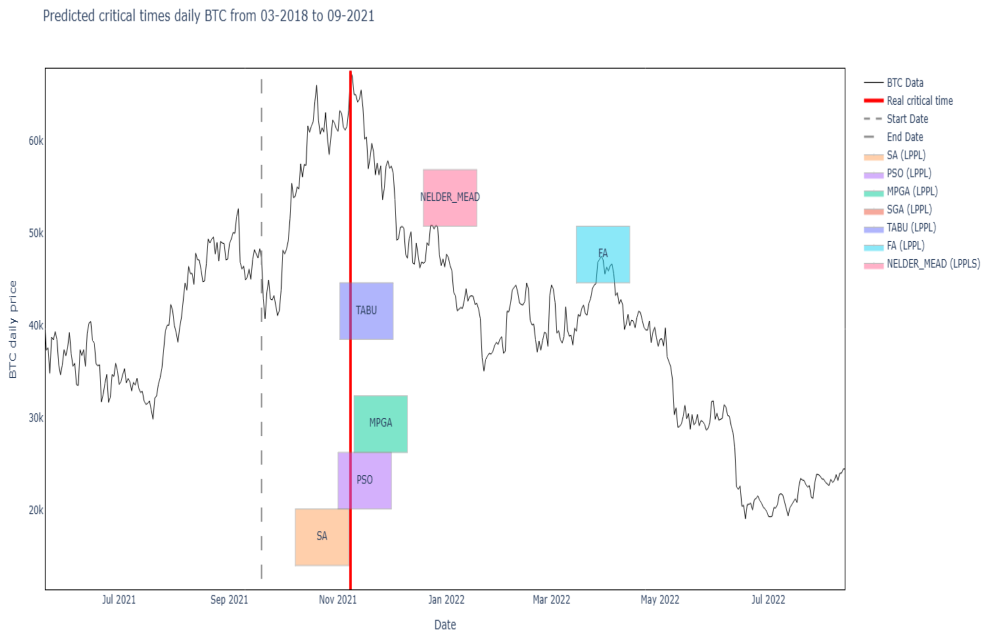


Figure 24: Prediction result of LPPL daily data from March 2018 to September 2021.

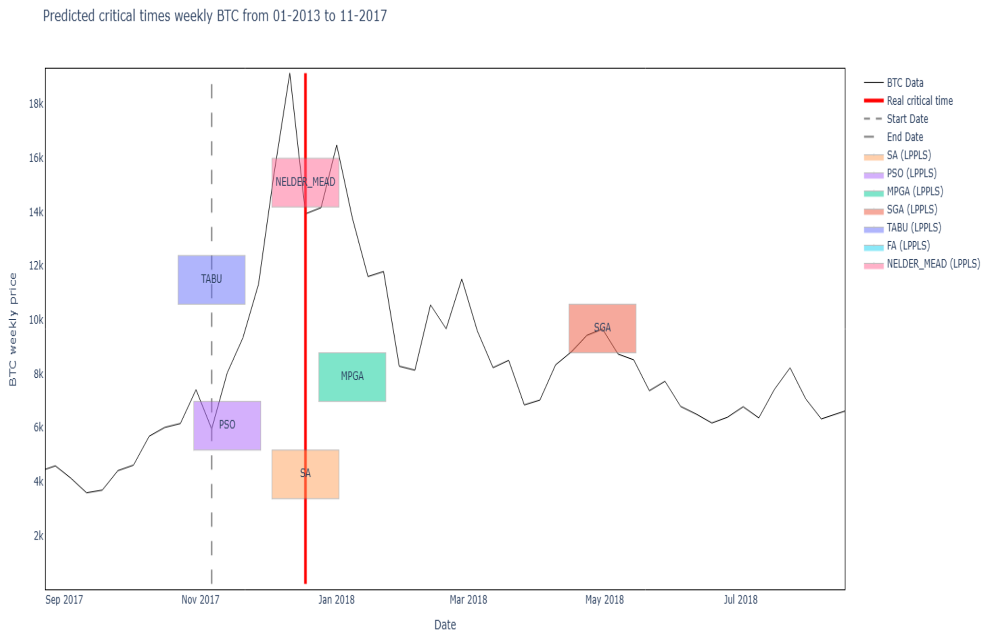


Figure 25: Prediction result of LPPLS weekly data from January 2013 to November 2017.

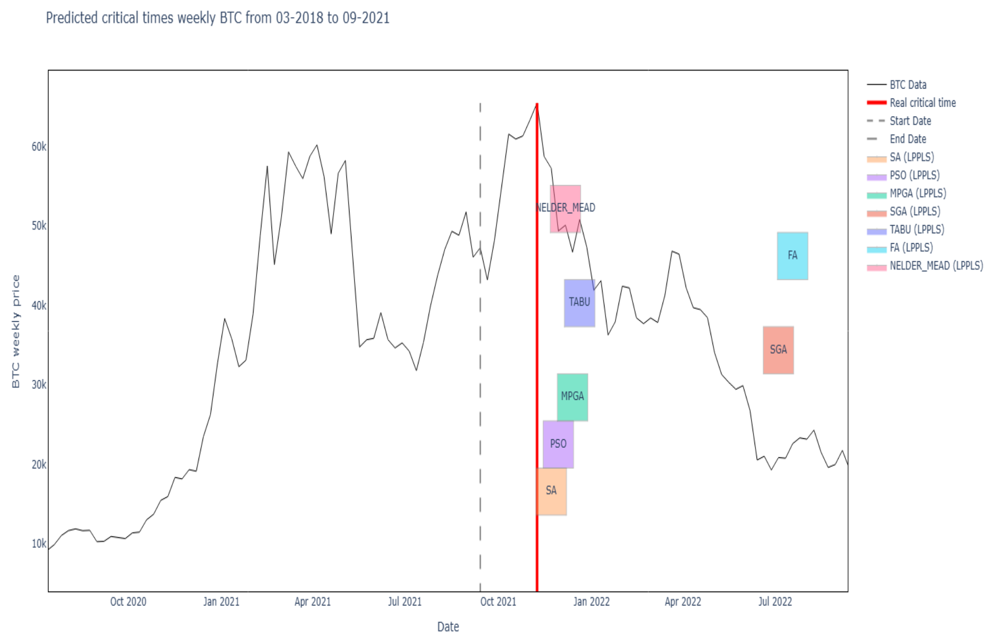


Figure 26: Prediction result of LPPL weekly data from March 2018 to September 2021.

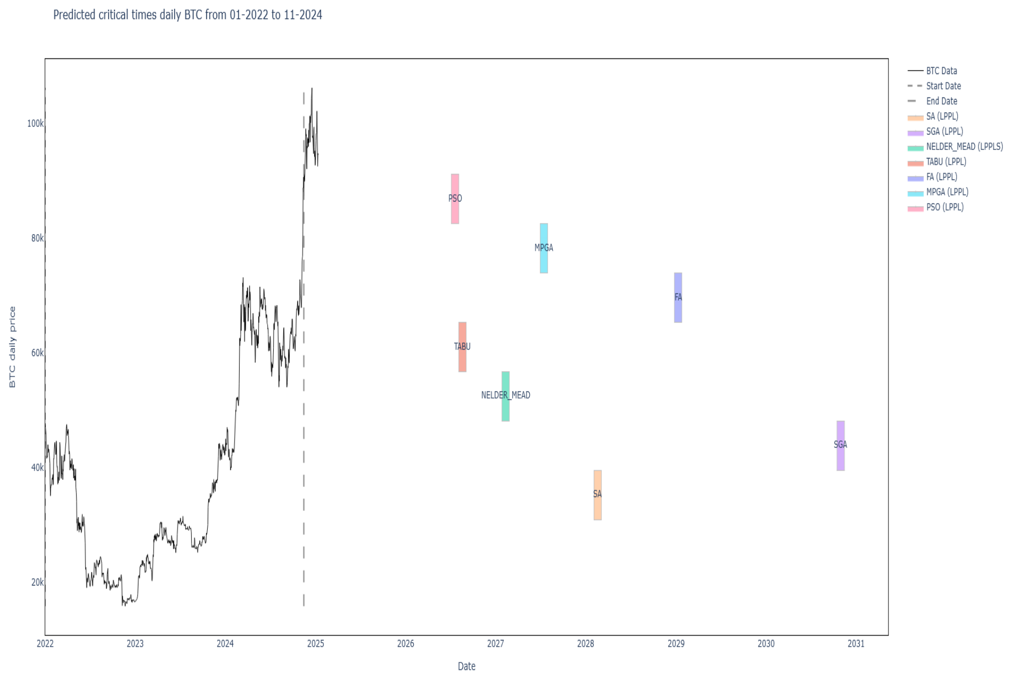


Figure 27: Next prediction of turning point on BTC from January 2022.

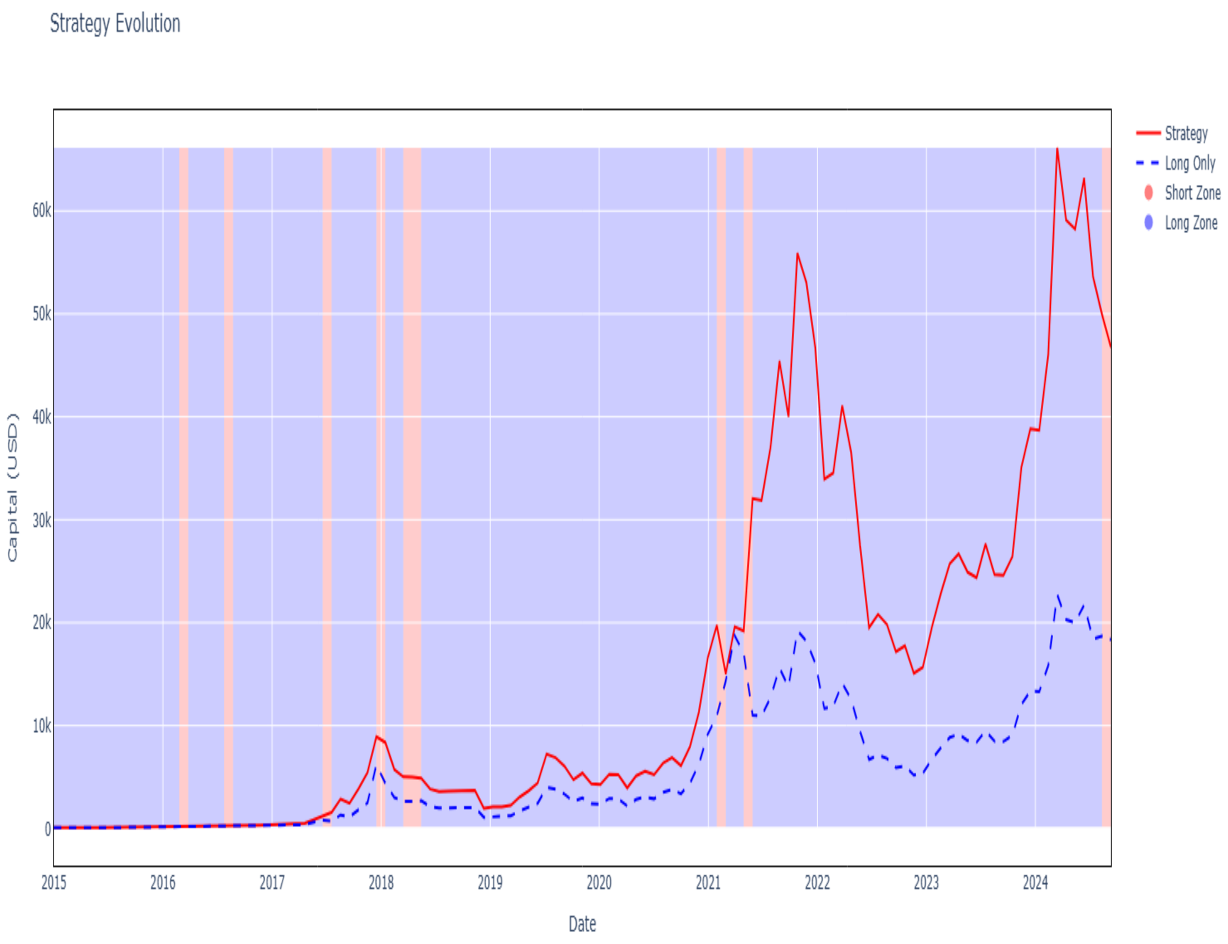


Figure 28: Trading Strategy results from January 2015 to September 2024.

10.6 Bounds and Algorithm Parameters

For the LPPL model, the parameters are optimized within defined bounds for both daily and weekly data. These bounds are specified as follows:

- t_c: Critical time parameter, bounded by $[0, 2520]$ for daily data and $[0, 520]$ for weekly data.
- ω: Angular log-frequency, bounded by $[0, 40]$.
- ϕ: Phase, bounded by $[0, 2\pi]$.
- α: Log-periodicity amplitude exponent, bounded by $[0.1, 0.9]$.

The parameters for each algorithm used in the optimization process are as follows:

Firefly Algorithm (FA):

- Number of fireflies: 20
- Maximum generations: 500

Multi-Population Genetic Algorithm (MPGA):

- Number of populations: 15
- Population size: 150
- Maximum generations: 700
- Stopping criterion: 80 consecutive generations without improvement

Particle Swarm Optimization (PSO):

- Number of particles: 100
- Maximum generations: 500

Simulated Annealing (SA):

- Maximum iterations: 1000
- Initial temperature: 1000
- Cooling rate: 0.99

Standard Genetic Algorithm (SGA):

- Population size: 100
- Maximum generations: 500
- Stopping criterion: 50 consecutive generations without improvement

Tabu Search:

- Maximum iterations: 10,000
- Tabu tenure: 20
- Neighborhood size: 20

These parameter bounds and settings ensure robust optimization of the LPPL model, tailored to the characteristics of the dataset and the specificities of each algorithm.

Algorithm 1 Multi-Population Genetic Algorithm (MPGA) for LPPL Optimization

```
Require: LPPL model type (LPPL or LPPLS), data, and hyperparameters:
    NUM_POPULATIONS: Number of sub-populations
    POPULATION_SIZE: Number of individuals per population
    MAX_GEN: Maximum number of generations
    STOP_GEN: Early stopping threshold for generations without improvement
Ensure: Best fitness value (RSS) and corresponding parameters (BestChromosome)
 1: Initialize parameter bounds (param_bounds) based on the LPPL model.
 2: Initialize random crossover and mutation probabilities for each population.
 3: Create NUM_POPULATIONS random initial populations.
 4: Compute initial fitness values for all populations.
 5: Identify the globally best fitness value (BestObjV) and parameters (BestChromosome).
 6: while gen $\leq$ MAX_GEN and gen0 $<$ STOP_GEN do
 7:     for each population $m$ in NUM_POPULATIONS do
 8:         Perform selection on population $m$ based on fitness.
 9:         Apply crossover to create offspring.
10:         Apply mutation to offspring within parameter bounds.
11:         Update population $m$ with mutated offspring.
12:     end for
13:     Perform immigration operation to share individuals across populations.
14:     Recompute fitness values for all populations.
15:     Identify the globally best fitness value (NewBestObjV) and parameters (NewBestChromosome) from
    all populations.
16:     if NewBestObjV $<$ BestObjV then
17:         Update BestObjV and BestChromosome.
18:         Reset stagnation counter gen0 $\leftarrow 0$.
19:     else
20:         Increment stagnation counter gen0 $\leftarrow$ gen0 $+ 1$.
21:     end if
22:     Increment generation counter gen $\leftarrow$ gen $+ 1$.
23: end while
        return BestObjV, BestChromosome
```

Algorithm 2 Standard Genetic Algorithm (SGA) for LPPL Optimization

```
Require: LPPL model type (LPPL or LPPLS), data, and hyperparameters:
    POPULATION_SIZE: Number of individuals in the population
    MAX_GEN: Maximum number of generations
    STOP_GEN: Early stopping threshold for generations without improvement
Ensure: Best fitness value (RSS) and corresponding parameters (BestChromosome)
 1: Initialize parameter bounds (param_bounds) based on the LPPL model.
 2: Generate random crossover and mutation probabilities (crossover_prob, mutation_prob).
 3: Create an initial random population of size POPULATION_SIZE.
 4: Compute initial fitness values for the population.
 5: Identify the best fitness value (BestObjV) and corresponding parameters (BestChromosome).
 6: while gen $\leq$ MAX_GEN and gen0 $<$ STOP_GEN do
 7:     Perform selection on the population based on fitness.
 8:     Apply crossover to create offspring.
 9:     Apply mutation to offspring within parameter bounds.
10:     Update the population with mutated offspring.
11:     Recompute fitness values for the updated population.
12:     Identify the best fitness value (NewBestObjV) and parameters (NewBestChromosome).
13:     if NewBestObjV $<$ BestObjV then
14:         Update BestObjV and BestChromosome.
15:         Reset stagnation counter gen0 $\leftarrow 0$.
16:     else
17:         Increment stagnation counter gen0 $\leftarrow$ gen0 $+ 1$.
18:     end if
19:     Increment generation counter gen $\leftarrow$ gen $+ 1$.
20: end while
        return BestObjV, BestChromosome
```

Algorithm 3 Simulated Annealing (SA) for LPPL Parameter Optimization

```
Require: LPPL model type (LPPL or LPPLS), data, and hyperparameters:
    MAX_ITER: Maximum number of iterations
    INITIAL_TEMP: Initial temperature
    COOLING_RATE: Rate at which temperature decreases
Ensure: Best fitness value (RSS) and corresponding parameters (BestSolution)
 1: Define parameter bounds (param_bounds) based on the LPPL model.
 2: Randomly initialize the current solution (CurrentSolution) within parameter bounds.
 3: Compute the fitness of the current solution (CurrentFitness).
 4: Initialize the best solution (BestSolution) and fitness (BestFitness) as the current ones.
 5: Initialize the temperature (Temperature $\leftarrow$ INITIAL_TEMP).
 6: while current $\leq$ MAX_ITER do
 7:     Generate a new candidate solution (CandidateSolution) by perturbing CurrentSolution.
 8:     Clip CandidateSolution to stay within parameter bounds.
 9:     Evaluate the fitness of CandidateSolution (CandidateFitness).
10:     Compute the fitness difference: $\Delta \leftarrow$ CandidateFitness $-$ CurrentFitness.
11:     if CandidateFitness $<$ CurrentFitness or random value $< \exp(-\Delta/\text{Temperature})$ then
12:         Accept CandidateSolution as the new CurrentSolution.
13:         Update CurrentFitness to CandidateFitness.
14:     end if
15:     if CurrentFitness $<$ BestFitness then
16:         Update BestSolution and BestFitness.
17:         Reset stagnation counter current $\leftarrow 0$.
18:     else
19:         Increment stagnation counter current $\leftarrow$ current $+ 1$.
20:     end if
21:     Apply the cooling schedule: Temperature $\leftarrow$ Temperature $\cdot$ COOLING_RATE.
22: end while
        return BestFitness, BestSolution
```

Algorithm 4 Particle Swarm Optimization (PSO) for LPPL Parameter Optimization

```
Require: LPPL model type (LPPL or LPPLS), data, hyperparameters:
    w: Inertia weight
    c1: Cognitive coefficient
    c2: Social coefficient
    NUM_PARTICLES: Number of particles in the swarm
    MAX_GEN: Maximum number of generations
Ensure: Best fitness value (RSS) and corresponding parameters (BestSolution)
 1: Define parameter bounds (param_bounds) based on the LPPL model.
 2: Initialize a swarm of NUM_PARTICLES particles:
 3: for each particle $p$ in the swarm do
 4:     Randomly initialize the particle's position within param_bounds.
 5:     Initialize the particle's velocity to zero.
 6:     Compute the particle's fitness (RSS) and store its local best position and fitness.
 7: end for
 8: Set the global best particle as the one with the best local fitness among the swarm.
 9: Initialize global best position (GlobalBestPosition) and fitness (GlobalBestFitness).
10: Set generation counter current $\leftarrow 0$.
11: while current $\leq$ MAX_GEN do
12:     for each particle $p$ in the swarm do
13:         Update the particle's velocity:
                $v_i \leftarrow w \cdot v_i + c1 \cdot r_1 \cdot (\texttt{LocalBestPosition}_i - x_i) + c2 \cdot r_2 \cdot (\texttt{GlobalBestPosition}_i - x_i)$
14:         Update the particle's position:
                $x_i \leftarrow x_i + v_i$
15:         Clip the particle's position to stay within param_bounds.
16:         Recompute the particle's fitness (RSS).
17:         if current fitness < local best fitness then
18:             Update the particle's local best position and fitness.
19:         end if
20:     end for
21:     Update the global best particle based on the new local best fitness values:
22:     if any particle's local best fitness < GlobalBestFitness then
23:         Update GlobalBestPosition and GlobalBestFitness.
24:     end if
25:     Increment the generation counter: current $\leftarrow$ current $+ 1$.
26: end while
        return GlobalBestFitness, GlobalBestPosition
```

Algorithm 5 Tabu Search (TABU) for LPPL Parameter Optimization

```
Require: LPPL model type (LPPL or LPPLS), data, hyperparameters:
    MAX_ITERATION: Maximum number of iterations
    TABU_TENURE: Maximum size of the tabu list
    NEIGHBORHOOD_SIZE: Number of neighboring solutions to explore
Ensure: Best fitness value (RSS) and corresponding parameters (BestSolution)
 1: Define parameter bounds (param_bounds) based on the LPPL model.
 2: Initialize a random solution (CurrentSolution) within param_bounds.
 3: Compute the fitness of the initial solution: CurrentFitness.
 4: Set BestSolution ← CurrentSolution, BestFitness ← CurrentFitness.
 5: Initialize an empty tabu list: TabuList ← [].
 6: for each iteration from 1 to MAX_ITERATION do
 7:     Generate a neighborhood of solutions (Neighborhood) around CurrentSolution.
 8:     Initialize BestNeighbor ← None, BestNeighborFitness ← $+\infty$.
 9:     for each solution Neighbor in Neighborhood do
10:         if Neighbor is not in TabuList then
11:             Compute fitness: NeighborFitness.
12:             if NeighborFitness < BestNeighborFitness then
13:                 BestNeighbor ← Neighbor.
14:                 BestNeighborFitness ← NeighborFitness.
15:             end if
16:         end if
17:     end for
18:     Add CurrentSolution to TabuList.
19:     if size of TabuList > TABU_TENURE then
20:         Remove the oldest solution from TabuList.
21:     end if
22:     if BestNeighbor is not None then
23:         CurrentSolution ← BestNeighbor.
24:         CurrentFitness ← BestNeighborFitness.
25:     end if
26:     if CurrentFitness < BestFitness then
27:         BestSolution ← CurrentSolution.
28:         BestFitness ← CurrentFitness.
29:     end if
30: end for
        return BestFitness, BestSolution
```

Algorithm 6 Nelder-Mead Optimization for LPPL Parameter Optimization

```
Require: LPPL model type (LPPL or LPPLS), data, parameter bounds (param_bounds).
Ensure: Best fitness value (RSS) and corresponding parameters (BestParams).
 1: function TransformParams($params, param\_bounds$)    ▷ Transform parameters to ensure they remain within bounds.
 2:     Initialize transformed ← empty array of same size as $params$.
 3:     for each parameter $i$ in params do
 4:         low, high ← $param\_bounds[i]$
 5:         transformed[i] ← $low + \frac{high-low}{1+\exp(-params[i])}$
 6:     end for
 7:     return transformed.
 8: end function
 9: function InverseTransformParams($params, param\_bounds$)    ▷ Inverse transform to return to the original parameter space.
10:     Initialize inverse_transformed ← empty array of same size as $params$.
11:     for each parameter $i$ in params do
12:         low, high ← $param\_bounds[i]$
13:         inverse_transformed[i] ← $\ln\left(\frac{params[i]-low}{high-params[i]}\right)$
14:     end for
15:     return inverse_transformed.
16: end function
17: procedure NelderMead($data, param\_bounds, lppl\_model$)    ▷ Main Nelder-Mead optimization procedure.
18:     Define ObjectiveFunction(params) ← Compute fitness (RSS) using lppl_model.
19:     Compute initial guess: initial_guess ← mean of param_bounds along axis 1.
20:     Transform initial guess: transformed_initial_guess ← InverseTransformParams(initial_guess, param_bounds).
21:     Perform Nelder-Mead optimization:
22:     result ← minimize(ObjectiveFunction, transformed_initial_guess, method='Nelder-Mead').
23:     Extract best fitness: bestObjV ← result.fun.
24:     Extract and transform best parameters: bestParams ← TransformParams(result.x, param_bounds).
25:     return bestObjV, bestParams.
26: end procedure
```

Algorithm 7 Firefly Algorithm (FA) for LPPL Parameter Optimization

```
Require: LPPL model (LPPL or LPPLS), data, hyperparameters:
    beta0: Initial attractiveness
    gamma: Light absorption coefficient
    alpha: Randomization parameter
    NUM_FIREFLIES: Number of fireflies
    MAX_GEN: Maximum number of generations
Ensure: Best fitness value (RSS) and corresponding parameters (BestFirefly).
 1: Define parameter bounds (param_bounds) based on the LPPL model.
 2: Initialize fireflies' positions (Fireflies) randomly within param_bounds.
 3: Compute initial fitness for each firefly: Fitness[i] ← RSS value of Fireflies[i].
 4: Set BestFirefly ← Fireflies[BestIndex], where BestIndex ← $\arg\min(Fitness)$.
 5: Set BestFitness ← Fitness[BestIndex].
 6: Initialize FitnessHistory ← [].
 7: for each generation $g$ in 1 to MAX_GEN do
 8:     for each firefly $i$ in 1 to NUM_FIREFLIES do
 9:         for each firefly $j$ in 1 to NUM_FIREFLIES do
10:             if Fitness[j] < Fitness[i] then
11:                 Compute distance: $r \leftarrow \|$Fireflies[i] $-$ Fireflies[j]$\|$.
12:                 Compute attractiveness: Attractiveness ← $\exp(-\text{gamma} \cdot r^2)$.
13:                 Update position:
    Fireflies[i] ← Fireflies[i]+beta0·Attractiveness·(Fireflies[j]−Fireflies[i])+alpha·(RandomNoise−0.5).
14:                 Clip position to bounds:
        Fireflies[i][k] ← max(lower_bound, min(Fireflies[i][k], upper_bound)).
15:                 Recompute fitness: Fitness[i] ← RSS value of Fireflies[i].
16:                 if Fitness[i] < BestFitness then
17:                     Update BestFitness ← Fitness[i].
18:                     Update BestFirefly ← Fireflies[i].
19:                 end if
20:             end if
21:         end for
22:     end for
23:     Append BestFitness to FitnessHistory.
24: end for
        return BestFitness, BestFirefly.
```

Algorithm 8 Metropolis-Hastings MCMC for LPPL Parameter Optimization (Not used in the paper)

```
Require:
    Initial parameters (InitialIndividual)
    Data: Time and observed values (data)
    Hyperparameters:
    NB_ITERATION: Number of iterations
    PROPOSAL_STD: Standard deviation of proposal distribution
    BURNIN_PERIOD: Burn-in period
Ensure: Best fitness value (BestFitness) and corresponding parameter set (Expectation)
 1: Initialize CurrentIndividual ← InitialIndividual
 2: Compute initial likelihood: CurrentLikelihood ← log(Likelihood(CurrentIndividual, data))
 3: Initialize IndividualHistory ← [CurrentIndividual]
 4: Initialize FitnessHistory ← [abs(CurrentLikelihood)]
 5: for each iteration $t$ in 1 to NB_ITERATION do
 6:     Propose a new individual: ProposedIndividual $\sim \mathcal{N}(\text{CurrentIndividual}, \text{PROPOSAL\_STD})$
 7:     Compute likelihood of proposal: ProposedLikelihood ← log(Likelihood(ProposedIndividual, data))
 8:     Compute priors:
        CurrentPrior ← ProposalDensity(CurrentIndividual | ProposedIndividual)
        ProposalPrior ← ProposalDensity(ProposedIndividual | CurrentIndividual)
 9:     Compute acceptance ratio:
        $\texttt{AcceptanceRatio} \leftarrow \min\left(\exp(\texttt{ProposedLikelihood - CurrentLikelihood}) \cdot \frac{\texttt{ProposalPrior}}{\texttt{CurrentPrior}}, 1\right)$
10:     Draw a random value: RandomValue $\sim U(0,1)$
11:     if RandomValue < AcceptanceRatio then
12:         Accept the proposal:
            CurrentIndividual ← ProposedIndividual
            CurrentLikelihood ← ProposedLikelihood
13:     else
14:         Reject the proposal (keep current individual).
15:     end if
16:     Append to history:
        IndividualHistory.append(CurrentIndividual)
        FitnessHistory.append(abs(CurrentLikelihood))
17: end for
18: Compute mean parameter values after burn-in period:
        AfterBurninPeriod ← IndividualHistory[BURNIN_PERIOD:]
        Expectation ← mean(AfterBurninPeriod, axis=0)
19: Set BestFitness ← FitnessHistory[-1] return BestFitness, Expectation
```