Fore-and-Back for Changepoint Detection

Vittorio Maniezzo[1][0000−0002−1220−1235]

University of Bologna, Cesena, Italy
vittorio.maniezzo@unibo.it
https://www.unibo.it/sitoweb/vittorio.maniezzo/en

Submitted to special session 2: Matheuristics

Abstract. This paper introduces a fore–and–back matheuristic framework for offline changepoint detection in univariate time series, combining the modeling rigor of mathematical programming with a flexible constructive search strategy. Classical exact approaches to optimal partitioning, such as dynamic programming, provide strong optimality guarantees but are typically developed under structural assumptions on the cost function and offer limited support for incorporating additional modeling requirements. The proposed method embeds an optimization-based formulation within an iterative forward–backward search scheme that constructs feasible segmentations and incrementally refines them as additional computational resources become available. This design yields an anytime algorithm that preserves the structural advantages of exact models—such as extensibility and the natural inclusion of constraints—while enabling the treatment of cost functions and problem variants that are difficult to address with existing approaches. Computational experiments illustrate the ability of the fore–and–back framework to produce high-quality solutions across a range of benchmark series and cost functions, highlighting its potential as a general and versatile tool for optimization-based changepoint detection.

Keywords: Changepoint analysis · matheuristics · Fore-and-Back.

1 Introduction

Changepoint detection (CPD) is the problem of identifying locations in a time series where the underlying data-generating process changes abruptly. Such changes may correspond to shifts in mean, variance, trend, or more complex structural properties, and their accurate detection is central to a wide range of applications, including industrial process monitoring, finance, bioinformatics, network security, and environmental analysis. As modern sensing and logging technologies generate increasingly long and high-frequency time series, there is a growing demand for CPD methods that are not only statistically sound but also computationally scalable and adaptable to practical constraints.

A common and widely studied formulation of CPD is the optimal partitioning problem, in which the time series is segmented into contiguous intervals so as to minimize a global objective composed of within-segment costs and

a penalty on the number of changepoints. This framework encompasses many classical likelihood-based and least-squares models and admits exact solution methods with strong optimality guarantees. Dynamic programming (DP) approaches constitute the canonical exact solution technique for this problem and, when segment costs can be computed efficiently, achieve quadratic time and memory complexity in the length of the series. Despite important advances such as pruning strategies and average-case improvements, these methods may still become computationally demanding for very long sequences, complex segment models, or when additional structural constraints are imposed.

In parallel to algorithmic developments, mathematical programming formulations—typically based on mixed-integer linear or quadratic models—have been proposed to address CPD in a unified optimization framework. These approaches offer several advantages, including modeling flexibility, the ability to incorporate global constraints across segments, and conceptual clarity. However, their direct solution via general-purpose solvers often requires exploring a large combinatorial search space, which can limit their practical applicability, especially in low-latency or resource-constrained environments. As a result, there remains a gap between the expressive power of optimization-based CPD models and the computational efficiency demanded by large-scale or time-critical applications.

Matheuristics, which integrate mathematical programming techniques within heuristic or metaheuristic frameworks, provide a promising avenue for bridging this gap. By leveraging the structural strength of exact models while selectively relaxing exhaustive search, matheuristics aim to deliver high-quality solutions with significantly reduced computational effort. In the context of CPD, however, few approaches explicitly exploit this paradigm, and most existing methods fall either on the side of exact optimization or on purely algorithmic heuristics.

This paper proposes a novel matheuristic for univariate changepoint detection based on a *fore–and–back* search strategy. The method starts from a rapidly generated feasible segmentation and alternates between forward and backward construction phases, guided by an underlying mathematical programming formulation. This design enables the algorithm to quickly identify promising regions of the solution space while retaining a global view of the segmentation problem. Importantly, the approach is exact and *anytime* in nature: it produces a valid solution almost immediately and progressively improves solution quality as additional computational resources become available.

The proposed matheuristic preserves key advantages of optimization-based CPD models. In particular, it naturally supports the inclusion of additional constraints on the segmentation, which can be difficult to accommodate within classical dynamic programming frameworks. At the same time, by avoiding a full enumeration of candidate segmentations, the method circumvents the computational burden typically associated with exact solvers.

Preliminary computational experiments suggest that the proposed approach consistently yields optimal or near-optimal solutions while remaining computationally competitive with state-of-the-art methods in contexts where these last can be applied. These results suggest that the fore–and–back matheuris-

tic provides an effective compromise between statistical accuracy and runtime efficiency, making it well suited for large-scale and resource-constrained CPD applications.

The remainder of the paper is organized as follows. Section 2 reviews related work on changepoint detection and optimization-based approaches. Section 2 recaps the fore–and–back matheuristic and its algorithmic components, Section 4 presents computational results, and Section 5 concludes with directions for future research.

2 Changepoint detection

Changepoint detection has been extensively studied across statistics, machine learning, and operations research, leading to a rich body of methodological literature. Early approaches focused on hypothesis testing and likelihood ratio methods applied sequentially or retrospectively, often with limited scalability or weak global optimality guarantees. More recent work has converged on segmentation-based formulations, in which the time series is partitioned into contiguous segments by minimizing a penalized cost function. Comprehensive surveys of this evolution can be found in [2] and [22].

From an algorithmic standpoint, dynamic programming (DP) has emerged as the canonical tool for solving optimal partitioning problems exactly. While classical DP incurs quadratic time and memory complexity, a substantial line of research has focused on pruning strategies to improve practical performance. Notable examples include PELT [14], functional pruning [20], and segment neighborhood methods [3]. Although highly effective under favorable conditions, these methods rely on assumptions about the cost structure or penalty function and may lose effectiveness when segment costs are complex, constraints are added, or worst-case behavior is encountered.

Beyond exact DP-based approaches, a variety of heuristic and approximate methods have been proposed to address scalability concerns. Binary segmentation and its variants [21] [9] iteratively split the series using local test statistics and are computationally efficient, but they are known to suffer from suboptimal global behavior. Wild binary segmentation and related multiscale procedures improve detection power but still lack a global optimization perspective and provide limited control over structural constraints.

Sliding-window and online heuristics, often based on distance or local statistics, have been widely explored for detecting changes very shortly after its occurrence (low-latency detection, e.g., [12] [13]). While computationally efficient, such methods typically rely on local information and offer limited control over global segmentation structure.

Optimization-based heuristics form a smaller but growing subset of the CPD literature. Some authors have proposed mixed-integer programming formulations for changepoint detection, highlighting their flexibility and modeling power [17]. However, due to the combinatorial nature of the resulting problems, these formulations are limited to moderate instance sizes when solved to optimality.

To mitigate this issue, relaxation-based methods, warm-start strategies, and restricted search techniques have been explored, though often without a systematic heuristic framework.

Matheuristics offer a principled way to combine the strengths of these two worlds. By embedding mathematical programming models within heuristic search strategies, matheuristics aim to preserve global structure while avoiding exhaustive enumeration. Despite their success in other combinatorial optimization domains, their application to changepoint detection remains underexplored. Existing approaches typically focus either on speeding up exact solvers or on designing problem-specific heuristics, with limited integration between the two.

The method proposed in this paper contributes to this emerging area by introducing a fore–and–back matheuristic that explicitly leverages a DP formulation while maintaining heuristic flexibility and anytime performance. Unlike classical DP or purely algorithmic heuristics, the proposed approach is designed to accommodate complex cost structures and global constraints, positioning it as a complementary alternative within the broader CPD literature.

3 Fore-and-Back for changepoint analysis

Developed as an enhancement of traditional Beam Search (BS), a variant of standard tree search that limits the number of offsprings that are expanded at each iteration, the Fore-and-Back method (F&B, Fore-n-Back, [4, 18]) aims to increase the overall effectiveness of the standard BS approach [15, 19, 5]. While Fore-and-back functions as an exact solution algorithm when provided with unlimited computational power, its primary intent is to serve as a heuristic. In this capacity, it prioritizes the rapid discovery of high-quality solutions over the rigorous pursuit of optimality proofs.

A defining feature of this approach is its ability to generate bounds on the cost required to complete partial solutions, even though it operates as a *primal-only method*. By calculating these bounds, the algorithm can identify and prune unpromising partial solutions before they are expanded, effectively narrowing the search space and increasing efficiency.

Beam search (BS) does not perform the exhaustive exploration typical of branch-and-bound algorithms. Instead, at each expansion step it maintains a restricted set of partial solutions, called the *beam* $\mathbf{T}$, from which at most δ solutions are retained, where δ is the *beam width*. These solutions are selected according to a ranking criterion that estimates the potential quality of their extensions, for instance by using bounds on the cost of their possible completions.

F&B extends the general beam search paradigm by alternating explorations in opposite expansion directions and by learning from partial solutions generated in previous iterations, which are stored in memory. These learned partial results are then exploited as a look-ahead mechanism to complete partial solutions when the search proceeds in the opposite direction. Consequently, the algorithm is particularly effective for problems that admit a natural direction for constructing partial solutions, which can also be meaningfully reversed.

F&B leverages the BS structure by embedding it into an iterative heuristic framework that adopts a memory-based learning and look-ahead strategy, explicitly exploiting information learned during earlier search phases. The algorithm performs a partial exploration of the solution space through a sequence of beam-search–like trees of two types, referred to as forward and backward trees. Each node at level h in these trees corresponds to a partial solution consisting of h solution components.

At iteration t, a forward tree F^t is constructed if t is odd, whereas a backward tree B^t is generated if t is even. During the construction of a tree, each partial solution X is extended to a feasible complete solution by combining it with learned partial solutions produced in the previous iteration of the complementary tree. The cost of the resulting solution is then used to bound the quality of the best complete solution that can be obtained from X.

A list, denoted by L_h^t, is associated with each level h of the tree built at iteration t. The list contains δ nodes generated but not expanded at level h, where δ is an input control parameter equivalent to the likewise denoted parameter of Beam Search. All nodes so far expanded at level h in all trees at odd iterations are kept in set E^h for forward trees, while those expanded at level h in all trees at even iterations are kept in $\overline{E}^h$ for backward trees. The nodes in the lists L_h^t, $h = 1, \ldots, n$, represent the memory of iteration t, that will be used to guide the exploration of the current tree and of the tree that will be explored in the following iteration $t + 1$.

The core idea of Fore-and-Back is to evaluate the completion cost of partial solutions stored at level h of the tree by means of the partial solutions stored in L_{n-h}^{t-1} in case of forward trees, or, analogously, the completion of partial solutions in L_{n-h}^t by means of solutions in L_h^{t-1} in case of backward trees.

Algorithm 1 presents a pseudocode for F&B. In it, $\mathbf{T}$ denotes alternatively the forward tree $\mathbf{F}$ or the backward tree $\mathbf{B}$, depending on the parity of the iteration counter, and T_h denotes the level h of the tree $\mathbf{T}$, be it forward or backward. The algorithm is controlled by three user-defined parameters: δ, the number of nodes expanded at each level of both forward and backward trees; *maxn*, the maximum total number of nodes to expand; *maxtnodes*, the limit of the number of nodes of each tree.

In order to expand level h of a tree at iteration t, the algorithm computes the value $c(X)$ for each node $X \in T_h$ and builds the set $L_h^t \subseteq T_h$. The list is defined ordering T_h by increasing cost values and keeping in L_h^t only its δ nodes having the smallest cost, which will be further expanded.

A distinctive characteristic of the Fore-and-Back algorithm is that, at each level h of the tree generated at iteration t, it also records the value $\hat{c}_h^t$. This value corresponds to the minimum cost among the nodes in T_h that do not belong to L_h^t. It represents the smallest cost that is expected to be incurred when completing a complementary partial solution without relying on the partial solutions that contributed to the construction of L_h^t. A node expansion is therefore accepted if *i*) it does not violate structural constraints, and *ii*) the sum of the cost

of the partial solution associated with the node and of the completion bound does not exceed the current upper bound.

More details about the algorithm can be found in [18], a sample implementation of Fore-and-Back applied to the GAP problem can be downloaded from [16].

4 Computational results

The Fore-and-Back algorithm for the CPD problem was implemented in C++, compiled with MSVC, and executed on a Windows 11 laptop equipped with an Intel Core Ultra 7 processor and 1 TB of RAM. The computational study is preliminary, and no particular effort has yet been devoted to the use of efficient data structures or to leveraging the multi-core architecture of the machine. Consequently, the reported runtimes should be regarded as indicative rather than representative of the best achievable performance.

Benchmark instances. The instances in the test set were selected to be representative of a variety of application areas in which time series analysis is relevant. We used benchmarks from the M forecasting competitions [1], including instances from the M3, M4, and M6 installments. Specifically, the M3 and M4 datasets include time series from microeconomics, industry, macroeconomics, finance, demographics, and an unspecified "other" category, while the M6 dataset consists of daily adjusted close prices of the competition assets.

The series of the M-benchmark datasets have length up to 5,000 observations. This range corresponds to the regime commonly considered in the changepoint literature where regime changes are semantically unambiguous and do not need additional scale assumptions. For substantially longer time series, meaningful changepoint analysis typically requires preprocessing steps, such as aggregation, smoothing, or multiscale constraints, not only to manage the computational cost, but to avoid mistaking short-lived fluctuations for actual structural changes. As Killick et al. [14] highlight, both algorithmic efficiency and appropriate scale considerations become critical as series length grows, to prevent overfitting and the detection of spurious changepoints.

In general in fact, the segment length must be greater than the noise scale (in turns greater than the sampling interval), to avoid over-segmentation. For long, high-frequency series CPD would detect variance bursts, outliers, or noise-induced artifacts, which are statistically valid changepoints but semantically meaningless.

Benchmark codes. Different codes were used to benchmark the effectiveness of F&B under various application conditions. In the simplest setting, where the objective is solely to identify effective changepoints and the quality of a model is assumed to be additive over its constituent segments, the time points of the series can be mapped to the nodes of a graph, with segments represented as arcs.

Algorithm 1: Algorithm Fore-and-Back

function Fore-and-Back(δ, *maxn*, *maxtnodes*);
Input : δ, beam width, *maxn*, max total num of nodes, *maxtnodes*, max num of nodes per tree
Output: A feasible solution $\mathbf{x}^*$ of value z_{best}
initialize $t = 0$, $noimpr = 0$, $nNodes = 0$, $z_{best} = \infty$;
while $nNodes \leq maxn$ **do** `// Alternate forward and backward trees`
  $t = t + 1$; $noimpr = noimpr + 1$;
  let $L_0^t = L_{n+1}^t = \{\emptyset\}$, $ntNodes$=0;
  foreach *level* $h = 1, ..., n$ **do**
    set $T_h = \{\emptyset\}$; `// generate the node set` T_h
    if t *is odd* **then**
      set $k = h$; $k1 = h - 1$;
    else
      set $k = n - h + 1$; $k1 = n - h + 2$;
    end
    foreach *node* $X \in L_{k-1}^t$ **do** `// Possible expansions`
      foreach *component* $s \in S_k$ **do**
        let $X' = X \cup \{s\}$;
        if X' *meets conditions i) and ii)* **then**
          set $T_k = T_k \cup X'$;
          $ntNodes = ntNodes + 1$;
          $nNodes = nNodes + 1$;
          if $h = n$ *and* $c(X') < z_{best}$ **then** `// feasible solution`
            set $z_{best} = c(X')$, $noimpr = 0$ and $\mathbf{x}^* = X'$;
          end
        end
      end
      set $E^k = E^k \cup X$; `// actual node expansion`
    end
    foreach *node* $X \in T_k$ **do** `// extract the subset` $L_k^t \in T_k$
      if $|T_k| \leq \delta$ **then**
        set $L_k^t = T_k$;
      else
        let L_k^t contain only the δ least cost partial solutions of T_k;
      end
    end
    if $ntNodes > maxtnodes$ **then break;**
  end
  if $noimpr = 2$ **then stop** ; `// sol. not improved for two iterations`
end

Under this formulation, the CPD problem reduces to finding a least-cost path from time 0 to the final time point in the resulting graph. This problem can be efficiently solved, for example, using the DAG-SHORTEST-PATHS (DSP) algorithm [7].

A commonly imposed constraint limits the maximum number of segments in the solution, specified as a parameter. Under this constraint, the problem remains easy to solve, as it requires only a minor adaptation of the Bellman–Ford shortest-path recursion [7], in which the number of dynamic programming stages is capped at the prescribed parameter value.

Finally, the state-of-the-art for CPD solution is represented by the *ruptures* library [22], a Python library for offline change point detection that provides a variety of methods for the analysis and segmentation of non-stationary signals. We include, as baselines, three widely used changepoint detection algorithms implemented in the ruptures library, namely PELT, and dynamic programming (DYNP). These methods are representative of different algorithmic paradigms and trade-offs between optimality and computational efficiency.

Pruned Exact Linear Time (PELT) [14], is an exact method for optimal multiple changepoint detection under an additive cost formulation with a penalty on the number of changepoints. PELT exploits a pruning rule that discards suboptimal candidate changepoints during the dynamic programming recursion, yielding linear expected time complexity under a subadditivity assumption on the cost function and penalty. PELT is often regarded as a state-of-the-art exact method for large-scale offline changepoint detection.

Dynamic programming (DYNP), implements an optimal partitioning algorithm based on dynamic programming with a fixed number of change points. Differently from Bellman–Ford, it explores all possible partitions of the time series and finds the optimal segmentation with exactly a given number of changepoints (i.e., the number of segments minus 1).

Cost function. In the practical application of piecewise linear models, the quality of a model is often related to interpretability or forecasting utility rather than mathematical properties. While statistical literature provides a rigorous framework for identifying structural breaks, practitioners frequently find that the most useful model is the one that aligns with the "naked eye" or provides the most coherent narrative for decision-making. As noted in industry applications, a model is often deemed successful if it identifies breakpoints that correspond to known external events—such as a shift in market policy or a technical failure—regardless of whether it satisfies every formal hypothesis test [8]. Modern methods have moved away from strict cost function optimization toward visual or semantic alignment with human-labeled events [23].

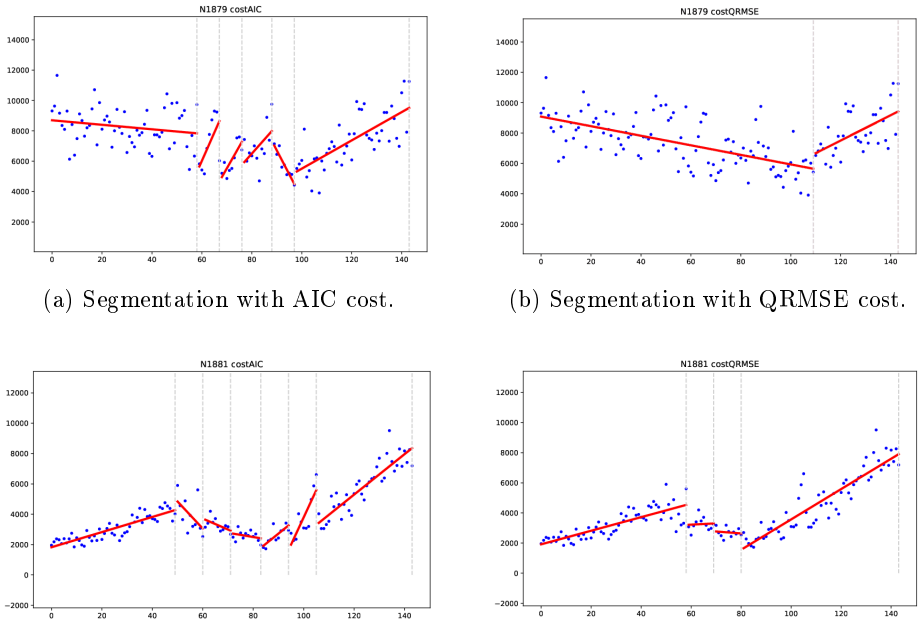
For actual purposes, the evaluation of a changepoint model often falls into two categories:

- *Aesthetic Clarity*: A model must be simple enough for stakeholders to understand its significance. While overly complex models with numerous small segments may achieve a better fit, they are often dismissed as 'noisy' or 'unstructured' [10].
- *Predictive Accuracy*: In time-series contexts, the ultimate indicator of quality is often out-of-sample performance. Although complex models may minimize in-sample residuals, simpler piecewise structures often produce more stable short time forecasts and lower generalization error. A changepoint is deemed meaningful when it impacts decision-making. In many industries, a "perfect" statistical fit on historical data is useless if it doesn't alert you to the moment your forecasting model stops working. Recent work suggests that instead of monitoring the complex raw data, we should monitor the forecast errors [10].

Common cost functions such as AIC, negative log-likelihood, and RMSE are frequently employed not because they represent a "true" physical reality, but because of their computational tractability. Penalized likelihood methods provide a formal stopping rule in the search for changepoints, which does not necessarily reflect the underlying data-generating processes [6, 11]. Information criteria such as AIC or BIC act as analytical tools that automate what would otherwise be a subjective model selection process by introducing a standard penalty for complexity. These metrics are best viewed as instruments for identifying useful approximations rather than a "true" model, which is virtually never contained in real-world candidate sets [10].

While a systematic assessment of forecasting performance lies beyond the scope of this work, we acknowledge here that no single cost function captures the full spectrum of practical objectives. Among standard criteria, the log-likelihood is valued for its mathematical structure and optimization efficiency, while AIC and BIC serve to minimize information loss by penalizing complexity. Conversely, RMSE offers an intuitive measure of predictive error in the original data units. We present results for both AIC and a variant of RMSE, called QRMSE (Quasi-RMSE) [17], which scales the sum of squared errors by the square root of the sample size rather than its mean. This scaling proved able to effectively gauge the complexity versus representativeness of the proposed model by favoring segments that maintain fit stability over longer intervals.

Figure 1 presents two examples of the different results obtained when using AIC as cost function (subfigures 1a and 1c) or QRMSE as cost (subfigures 1b and 1d). It can be observed that QRMSE induces a regularizing effect directly through the cost assessment, thereby emphasizing aesthetic clarity. We stress again that there is no "right" cost function; rather, the choice depends on the desired structure of the resulting segmentation. Moreover, most segmentation codes include a parameter that influences the cost, thereby controlling the trade-off between many well-fit segments and fewer segments with worse fit. However, this typically requires series-specific tuning (see also Table 1), which is impractical in several application contexts.



(a) Segmentation with AIC cost.

(b) Segmentation with QRMSE cost.

(c) Segmentation with AIC cost.

(d) Segmentation with QRMSE cost.

Fig. 1: AIC vs. QRMSE models, series N1879 and N1881

Preliminary results. The results presented here are preliminary, as the code is currently undergoing a revision aimed at improving its efficiency. Tables 1 and 2 report results on series taken from different M Competition benchmark sets, covering multiple application domains and including cases with and without trend, seasonality, stationarity, and high S/N ratio.

Two tables are presented for the two tested cost functions. The first one, Table 1, makes use of AIC, a function that enjoys the subadditivity property and is coherent with most of the relevant literature. This property permits the use of the dominance criterion proposed in [14] and is implemented in PELT and in `ruptures`' DYNP. The columns report the M benchmark and the series name (*name*), the application domain that originated the series (*category*), the number of points in the series (n), followed by the number of segments (*segments*) and the total CPU time ($t.cpu$) for Fore-and-Back, PELT, and DYNP.

All tests were run by optimizing the cost subject to a hard constraint on the minimum allowed segment length and a soft constraint on the maximum number of segments proposed by the model. It can be observed that, for series of this size, the computation times of the three algorithms are comparable, although PELT already appears to scale best, while Fore-and-Back begins to suffer on longer series. The main difference lies in the number of segments proposed. Even though all three algorithms use the same AIC cost function, DYNP enforces models with the exact number of segments specified in the call (8 in this case).

Fore-and-Back allows fewer segments when convenient, a situation that occurs in some cases, whereas PELT being based on a penalty whose value was kept stable across all tests to avoid series-specific fitting, results in widely different number of segments.

Table 1: Preliminary results using AIC cost function.

			FnB		PELT		DYNP	
name	category	n	segments	t.cpu	segments	t.cpu	segments	t.cpu
M3/N1879	Industry	144	8	0.052	2	0.062	8	0.0156
M3/N1881	Industry	144	8	0.054	4	0.109	8	0.0156
M3/N1882	Industry	144	8	0.052	12	0.046	8	0.015
M3/N1883	Industry	144	8	0.042	15	0.046	8	0.015
M3/N1884	Industry	144	8	0.06	19	0.031	8	0.031
M3/N1918	Industry	144	8	0.064	5	0.078	8	0.015
M3/N2830	Micro	104	8	0.024	7	0.031	8	0
M3/N2831	Micro	104	8	0.022	10	0.031	8	0.015
M3/N2832	Micro	104	6	0.024	4	0.062	8	0
M3/N2833	Micro	104	8	0.022	7	0.046	8	0.015
M3/N2834	Finance	104	8	0.026	9	0.046	8	0
M4/Q14640	Industry	115	2	0.036	1	0.015	8	0
M4/Q15444	Industry	126	4	0.05	3	0.031	8	0.015
M4/Q18148	Industry	71	5	0.006	4	0.046	8	0.015
M4/Q18496	Industry	126	8	0.038	5	0.046	8	0
M4/Q19604	Industry	123	8	0.038	8	0.046	8	0.015
M4/Q19637	Industry	123	3	0.044	14	0.031	8	0.015
M4/Q2002	Macro	170	8	0.094	11	0.078	8	0.031
M4/Q22221	Finance	738	8	1.926	58	0.359	8	0.593
M4/Q5109	Macro	866	8	2.858	102	0.296	8	0.828
M4/Q7774	Micro	115	6	0.026	3	0.078	8	0.015
M4/Q8030	Micro	97	8	0.022	6	0.031	8	0.015
M6/ABBV	Finance	253	8	0.196	19	0.109	8	0.062
M6/BTCUSD	Finance	367	8	0.406	22	0.156	8	0.125
M6/EWG	Finance	253	8	0.176	15	0.109	8	0.046
M6/EWQ	Finance	253	8	0.168	14	0.125	8	0.062
M6/EWZ	Finance	253	8	0.182	16	0.125	8	0.046
M6/GBPUSD	Finance	263	8	0.184	17	0.093	8	0.078
M6/PYPL	Finance	253	8	0.162	19	0.109	8	0.062
M6/RBLX	Finance	253	8	0.17	21	0.093	8	0.046
M6/ROST	Finance	253	8	0.172	20	0.078	8	0.062
M6/SLV	Finance	253	8	0.168	23	0.109	8	0.078
M6/USDJPY	Finance	263	8	0.182	22	0.109	8	0.062

Table 2 presents results obtained using the QRMSE cost function. Any cost function obtained by normalizing the residual sum of squares by the square root of the segment length does not satisfy the subadditivity condition required by PELT, since splitting a segment can increase the total cost. More generally, any

concave normalization in the segment length (such as division by the square root of the length, as in RMSE or QRMSE) breaks PELT's subadditivity assumption. Consequently, the correctness of PELT pruning is not guaranteed for such a cost, and we therefore report results only for Fore-and-Back, as the `ruptures` codes, although much faster, do not guarantee optimality in this case.

The table columns report *name*, *category*, and n as in Table 1, followed by *t.tot*, the total CPU time of the optimization; *t.opt*, the time required to reach the optimal value; *matches*, the number of matches, i.e., solutions obtained by chaining a partial solution from a forward pass with a partial solution from a backward pass; *fathomed*, the number of nodes eliminated by the lower bounds computed at each node; and *n.brk*, the number of final breakpoints. All runs again required proposing at most 8 segments, i.e., 7 breakpoints. The impossibility of exploiting PELT's dominance property results in a slower code; however, the table shows that the characteristic components of Fore-and-Back—namely, the matching of partial solutions and the use of primal bounds—remain effective also with this cost function.

5 Conclusions

This paper presents current results on the application of the Fore-and-Back matheuristic framework to the changepoint detection (CPD) problem. Owing to its relevance for time series analysis, CPD has been extensively studied in the literature, and highly efficient codes are available. However, we note that most existing approaches rely on a subadditivity assumption for the cost function. While this assumption is reasonable in many contexts, it excludes families of cost functions that may be of interest in practical applications.

The use of the Fore-and-Back framework allows the design of an exact algorithm for CPD, which naturally becomes an effective heuristic when run under limited computational resources. Characteristic features of the algorithm include the ability to derive bounds on the cost of completing partial solutions in a primal-only fashion and the construction of partial solutions with structural properties that guide their possible chaining.

Computational tests are still ongoing at the time of writing; nevertheless, the proposed matheuristic appears to be computationally comparable with the most efficient available codes, while retaining applicability to cost functions that limit or preclude the use of alternative approaches.

A future extension currently under investigation is to move toward assessments based on forecasting performance rather than on cost-function fit. This would enable the quality of a model to be quantified directly in the context of its intended application, thereby increasing the practical impact of this research.

References

1. Makridakis competitions. https://en.wikipedia.org/wiki/Makridakis_Competitions, accessed: 5 January 2026

Table 2: Preliminary results using QRMSE cost function.

name	category	n	t.tot	t.opt	matches	fathomed	n.brk
M3/N1879	Industry	144	0	0	266	40446	2
M3/N1881	Industry	144	0	0	363	20441	7
M3/N1882	Industry	144	0	0	285	40027	7
M3/N1883	Industry	144	0	0	359	25539	7
M3/N1884	Industry	144	0	0	337	41697	7
M3/N1918	Industry	144	0	0	446	21688	6
M3/N2830	Micro	104	0	0	219	12497	7
M3/N2831	Micro	104	0	0	175	16713	6
M3/N2832	Micro	104	0	0	160	19533	4
M3/N2833	Micro	104	0	0	262	4184	7
M3/N2834	Finance	104	0	0	248	8600	7
M4/Q14640	Industry	115	0	0	0	17037	0
M4/Q15444	Industry	126	0	0	105	11704	3
M4/Q18148	Industry	71	0	0	92	2329	2
M4/Q18496	Industry	126	0	0	252	17004	7
M4/Q19604	Industry	123	0	0	263	18127	7
M4/Q19637	Industry	123	0	0	334	21332	7
M4/Q2002	Macro	170	0	0	519	103936	7
M4/Q22221	Finance	738	14	13	7086	1811047	7
M4/Q5109	Macro	866	38	36	462	5586555	7
M4/Q7774	Micro	115	0	0	102	18649	2
M4/Q8030	Micro	97	0	0	183	6872	7
M6/ABBV	Finance	253	0	0	974	270239	7
M6/BTCUSD	Finance	367	2	2	1157	675562	7
M6/EURCHF	Finance	263	0	0	903	364509	7
M6/EURUSD	Finance	263	0	0	507	264444	7
M6/EWG	Finance	253	0	0	719	185694	7
M6/EWQ	Finance	253	0	0	816	185572	7
M6/EWZ	Finance	253	0	0	618	354703	7
M6/GBPUSD	Finance	263	0	0	580	291430	7
M6/PYPL	Finance	253	0	0	760	214249	7
M6/RBLX	Finance	253	0	0	784	301279	7
M6/ROST	Finance	253	0	0	824	225076	7
M6/SLV	Finance	253	0	0	686	132486	7
M6/USDJPY	Finance	263	0	0	687	263442	7

2. Aminikhanghahi, S., Cook, D.J.: A survey of methods for time series change point detection. Knowledge and Information Systems **51**, 339–367 (2017)
3. Auger, I.E., Lawrence, C.E.: Algorithms for the optimal identification of segment neighborhoods. Bulletin of Mathematical Biology **51**(1), 39–54 (1989)
4. Bartolini, E., Maniezzo, V., Mingozzi, A.: An adaptive memory-based approach based on partial enumeration. In: Maniezzo, V., Battiti, R., Watson, J.P. (eds.) LION 2, LNCS 5313, pp. 12–24. Springer (2008)
5. Blum, C.: Beam-ACO for simple assembly line balancing. INFORMS Journal on Computing **20**(4), 618–627 (2008)
6. Burnham, K., Anderson, D.: Model Selection and Inference: A Practical Information-Theoretic Approach. Springer-Verlag, New York, 2nd edn. (2002)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, 3rd edn. (2009)
8. Datadog Engineering: Piecewise regression: When one line simply isn't enough. https://www.datadoghq.com/blog/engineering/piecewise-regression/ (2017)
9. Fischetti, M., Lodi, A., Salvagnin, D.: Just MIP it! In: Maniezzo, V., Stützle, T., Voß, S. (eds.) Matheuristics, Annals of Information Systems, vol. 10, pp. 1–20. Springer (2009)
10. Gao, Z., Xiao, X., Fang, Y.P., Rao, J., Mo, H.: A selective review on information criteria in multiple change point detection. Entropy **26**(1), 50 (2024)
11. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning: data mining, inference and prediction. Springer, 2 edn. (2009)
12. Keogh, E.J., Chu, S., Hart, D.M., Pazzani, M.J.: An online algorithm for segmenting time series. In: Proceedings of the 2001 IEEE International Conference on Data Mining. pp. 289–296. IEEE, San Jose, CA, USA (2001)
13. Keogh, E.J., Chu, S., Hart, D.M., Pazzani, M.J.: Segmenting time series: A survey and novel approach. https://api.semanticscholar.org/CorpusID:8365617 (2002)
14. Killick, R., Fearnhead, P., Eckley, I.A.: Optimal detection of changepoints with a linear computational cost. Journal of the American Statistical Association **107**(500), 1590–1598 (2012)
15. Lowerre, B.: The HARPY speech recognition system. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA (1976)
16. Maniezzo, V.: MatheuristicsGAP: Accompanying code for Matheuristics, algorithms and implementations. https://github.com/maniezzo/MatheuristicsGAP (2021), accessed: 5 January 2026
17. Maniezzo, V.: Extended set covering for time series segmentation. In: Metaheuristics: 15th International Conference, MIC 2024, Lorient, France, June 4–7, 2024, Proceedings, Part I. p. 193–199. Springer-Verlag, Berlin, Heidelberg (2024)
18. Maniezzo, V., Boschetti, M.A., Stützle, T.: Matheuristics: Algorithms and Implementations. EURO Advanced Tutorials on Operations Research, Springer (2021)
19. Natarajan, B.K.: On the use of beam search in exact inference. IEEE Transactions on Pattern Analysis and Machine Intelligence **11**(8), 883–887 (1989)
20. Rigaill, G.: A pruned dynamic programming algorithm to recover the optimal segmentation of a signal. Journal de la Société Française de Statistique **156**(4), 180–205 (2015)
21. Scott, A.J., Knott, M.: A cluster analysis method for grouping means in the analysis of variance. Biometrics **30**(3), 507–512 (1974)
22. Truong, C., Oudre, L., Vayatis, N.: Selective review of offline change point detection methods. Signal Processing **167**, 107299 (2020)
23. Xu, R., Song, Z., Wu, J., Wang, C., Zhou, S.: Change-point detection with deep learning: A review. Frontiers of Engineering Management **12**, 154–176 (2025)