Article

Not peer-reviewed version

Optimal Partitioning Changepoint Analysis

Vittorio Maniezzo * and Lisa Vecchi

Posted Date: 20 March 2026

doi: 10.20944/preprints202603.1640.v1

Keywords: time series; set partitioning problem; segmentation; change point detection

 doi:10.20944/preprints202603.1640.v1

Article

Optimal Partitioning Changepoint Analysis

Vittorio Maniezzo [1,*] **and Lisa Vecchi** [2]

[1] Department of Computer Science, University of Bologna, Italy
[2] University of Bologna, Italy
* Correspondence: vittorio.maniezzo@unibo.it

Abstract

Detecting changepoints in time series is a fundamental task in statistical modeling and data-driven decision-making. We introduce a novel set partitioning-based model for changepoint detection that leverages combinatorial optimization to identify an optimal set of segments explaining the observed data. Unlike conventional methods based on dynamic programming (DP), which often impose strict structural constraints on the objective function (e.g., additivity), our formulation uses Integer Linear Programming (ILP). This approach offers significantly increased expressiveness and flexibility in the cost structure, accommodating diverse, non-additive loss functions and complex penalty schemes. Our formulation guarantees global optimality under this flexible cost structure, a critical advantage over heuristic or approximate approaches. The model's design enables high adaptability to different application domains, including finance, bioinformatics, and industrial monitoring. The efficiency of modern MILP solvers, combined with tailored dominance rules, enables the solution of instances with several hundreds of observations in practical time. Computational results indicate that the approach extends tractability beyond previously studied settings, effectively handling classes of instances whose structural constraints could not be accommodated by existing methods, while retaining robustness and interpretability.

Keywords: time series; set partitioning problem; segmentation; change point detection

1. Introduction

Changepoint detection (CPD) is the task of identifying abrupt changes in the statistical properties of a time series and constitutes a foundational problem across statistics, machine learning, and signal processing [1–3]. The objective of CPD is to partition a temporal sequence into contiguous, non-overlapping segments, each characterized by internally homogeneous behavior. Such segmentation enables a clearer understanding of the underlying data-generating process and facilitates the application of targeted analytical or predictive models at the segment level. As a result, effective changepoint detection plays a critical role not only in descriptive analysis, but also in forecasting, anomaly detection, and data-driven decision-making.

CPD assumes a pivotal function in a wide range of application domains, including financial market analysis (regime shifts), industrial process monitoring (fault detection), climate science (trend changes), and bioinformatics (genomic variation analysis). Across these domains, a CPD method must satisfy two key requirements: (i) accurately segment the data so as to faithfully represent the underlying process, and (ii) provide a globally optimal solution, since suboptimal segmentation can propagate errors downstream and lead to significant real-world costs.

Time series segmentation has therefore attracted extensive attention, leading to the development of a broad spectrum of methodologies, including classical statistical techniques, Bayesian approaches, machine learning algorithms, and hybrid models. Among these, segmentation based on linear models remains particularly prominent due to its computational efficiency and interpretability. This setting, commonly referred to as *piecewise linear regression* [4], assumes that the time series can be approximated by a sequence of linear trends, each corresponding to a distinct segment.

 doi:10.20944/preprints202603.1640.v1

For exact and globally optimal changepoint detection under additive cost structures, the dominant methodological paradigm is dynamic programming (DP). Methods such as Optimal Partitioning exploit the recursive structure of the problem, constructing the optimal segmentation for a sequence of length n from optimal solutions to shorter prefixes. This approach yields polynomial-time algorithms under suitable assumptions and has proven highly effective in practice. However, the effectiveness of the DP framework relies fundamentally on the additive decomposability of segment-wise costs, typically combined with a penalty proportional to the number of segments. Accelerated variants additionally require regularity conditions on the cost function, which can be interpreted as a superadditivity-type inequality, to justify pruning.

This additive and recursive structure imposes a significant modeling limitation. In many practical settings, the analyst wishes to enforce global structural constraints that couple properties of all selected segments. Examples include limiting the total duration of short segments, bounding the number of changepoints within specific time windows, or enforcing regularity in segment lengths to avoid excessively fragmented solutions. While certain such constraints can, in principle, be approximated through state augmentation or penalty tuning, doing so rapidly compromises the tractability and scalability of DP-based methods. As a result, DP frameworks remain largely confined to locally additive objectives, restricting their applicability in scenarios where global control over the segmentation structure is essential.

To overcome this limitation, we propose a novel formulation of the CPD problem as a combinatorial optimization model, specifically a set partitioning problem, possibly extended to its set covering relaxation. In this formulation, all feasible segments are pre-enumerated and treated as decision variables, and changepoint detection reduces to selecting a minimum-cost subset of segments that exactly covers the time series. Unlike breakpoint-indicator or network-flow formulations, this representation models segments as atomic objects, thereby naturally supporting richer interactions and more expressive global constraints.

The key advantage of the MILP framework lies not only in its ability to reproduce the classical additive cost structure, but more importantly in its capacity to enforce complex global constraints through simple linear (in)equalities. These constraints can link arbitrary subsets of segment variables, control aggregate properties such as the number, length, or distribution of segments, and encode domain-specific structural requirements that are difficult or impractical to express within DP-based models. Unlike breakpoint-indicator or network-flow formulations, the proposed set partitioning approach provides a direct and flexible mechanism for global control over the segmentation. While MILP solving is NP-hard in general, recent advances in solver technology, together with problem-specific preprocessing and bounding techniques, make this approach computationally viable for problem sizes of practical interest.

In this paper, we introduce Set-Partitioning based Changepoint Detection (SP-CPD) as a framework for optimal time series segmentation. Our main contributions are as follows:

A set-partitioning MILP formulation with additional constraints for CPD with global structural control. We formulate the changepoint detection problem as a set partitioning model, enabling the direct incorporation of global constraints that are beyond the practical reach of dynamic programming methods.

Enhanced modeling fidelity through global constraints. We demonstrate the practical value of the proposed framework by introducing and analyzing several global constraints that improve the interpretability, robustness, and domain relevance of the resulting segmentations.

Competitive computational performance and solution quality. Through extensive computational experiments, we show that SP-CPD produces higher-quality segmentations while achieving runtimes that, up to mid-sized series, are comparable to state-of-the-art DP-based and heuristic methods.

The paper further illustrates the versatility of the proposed approach through a range of real-world applications, including financial time series, epidemiological data, and environmental monitoring.

 doi:10.20944/preprints202603.1640.v1

Common challenges such as noise, high variances, and non stationarity are included, highlighting the advantages of MIP-based changepoint detection in addressing domain-specific structural requirements.

The remainder of the paper is organized as follows. Section 2 reviews the relevant literature. Section 3 presents the proposed set partitioning model. Section 4 reports the computational results. Finally, Section 5 concludes and outlines directions for future research.

2. Related Literature

The problem of optimal changepoint detection and time series segmentation has been extensively studied, and a wide range of exact and approximate methods have been proposed. Comprehensive surveys are provided in [2,3], which classify approaches into statistical, Bayesian, algorithmic, and learning-based families. In this section, we focus specifically on exact optimization-based formulations for changepoint detection and highlight their modeling assumptions and limitations relative to the proposed approach.

Dynamic programming (DP) constitutes the dominant paradigm for exact changepoint detection under additive cost structures. Classical methods such as Segment Neighborhood [5] and Optimal Partitioning [6] formulate the problem as the minimization of a sum of segment-wise losses plus a penalty on the number of changepoints. By exploiting the optimal substructure of the problem, these methods guarantee global optimality and admit polynomial-time algorithms under suitable assumptions.

Substantial effort has been devoted to improving the practical efficiency of DP-based CPD. Notable examples include the Pruned Exact Linear Time (PELT) algorithm [7], which uses inequality-based pruning to reduce the search space, and functional pruning approaches [8,9], which exploit convexity properties of the segment cost. These methods have become state of the art for large-scale problems with additive objectives.

Despite their efficiency, DP-based models fundamentally rely on the separability of the objective function across segments. This reliance severely limits their ability to incorporate constraints that couple decisions across the entire segmentation. While constrained DP and state-augmented formulations have been proposed for specific problems [10], the resulting state spaces typically grow exponentially, making such approaches impractical beyond very limited forms of global control. Consequently, DP methods remain best suited for problems with simple additive objectives and limited structural constraints.

Several optimal CPD models reinterpret the segmentation problem as a shortest-path problem on a directed acyclic graph, where nodes correspond to time indices and arcs represent feasible segments weighted by their associated costs [11,12]. This representation is then solved by DP models that yield efficient shortest-path algorithms.

Although graph-based formulations offer valuable conceptual insight and algorithmic flexibility, they inherit the same structural limitations as the DP at their core. In particular, it is difficult to express global constraints that do not decompose additively along a path without introducing additional dimensions or complex side constraints, which compromise tractability once again.

Mixed-Integer Programming (MIP) has been previously applied to changepoint detection and piecewise regression, primarily through formulations based on breakpoint indicator variables or segment assignment variables. Early work on segmented regression using integer programming can be traced back to [13] and [14], with more recent formulations incorporating modern MIP solvers for exact piecewise linear fitting [15], while a recent quadratic formulation based on big-M constraints can be found in [16].

In breakpoint-based formulations, binary variables indicate whether a changepoint occurs at a given time index, while continuous variables model segment-specific parameters. These models allow regression and segmentation decisions to be optimized jointly and can accommodate certain side constraints. However, constraints involving segment-level properties—such as length distributions, aggregate statistics of selected segments, or interactions between non-adjacent segments—typically

 doi:10.20944/preprints202603.1640.v1

require additional auxiliary variables and complex logical constraints, leading to weak linear relaxations and limited scalability. Assignment-based formulations, in which observations are assigned to segments via binary variables, suffer similar difficulties when enforcing contiguity and global structural constraints. In both cases, segmentation is encoded implicitly, rather than directly treating segments as decision objects.

The approach proposed in this paper differs fundamentally from the above paradigms by adopting a set-based perspective. Each feasible segment is explicitly represented as a decision variable, and the segmentation problem is formulated as a set partitioning (or covering) problem in which the time axis must be covered (at least) once. While set covering and set partitioning models are classical tools in combinatorial optimization [17,18], they have received comparatively little attention in the changepoint detection literature.

By associating segments with decision variables, the set-partitioning formulation provides a natural mechanism for expressing global structural constraints directly at the segment level. Aggregate constraints on the number, length, distribution, or interaction of segments can be imposed through simple linear inequalities, without resorting to state augmentation or intricate logical constructs. This distinguishes the proposed SP-CPD framework from existing MIP formulations and enables a level of modeling expressiveness that is difficult to achieve within DP-based, shortest-path, or breakpoint-indicator models.

In summary, existing optimal CPD models exhibit a clear trade-off between computational efficiency and modeling flexibility. Dynamic programming and graph-based approaches provide excellent performance for additive objectives but offer limited support for global structural constraints. Prior MIP formulations increase modeling power but typically encode segmentation implicitly and encounter scalability issues when complex segment-level constraints are introduced.

The proposed SP-CPD framework occupies a complementary position in this landscape. By combining a set-partitioning formulation with modern MILP techniques and problem-specific preprocessing, it delivers global optimality while substantially expanding the class of constraints that can be handled in a principled and scalable manner. This positions SP-CPD as a flexible and extensible optimization framework for changepoint detection in applications where global structure and domain-specific requirements play a central role.

3. Extended Set Partitioning and Covering Models

The CPD problem can be naturally modeled from a set partitioning perspective: all subsequences satisfying predefined structural properties—termed feasible runs—are generated and scored according to a segment-specific quality metric, under an additivity assumption on the cost function. The segmentation problem is then formulated as the selection of a collection of runs that partitions the time index set while optimizing a global objective based on residual loss or model fit. The resulting formulation is a 0–1 integer linear program, which can be viewed as a special case of a mixed-integer linear program (MILP), thereby allowing the use of state-of-the-art MILP solvers [19]. This framework provides considerable flexibility, as additional modeling requirements can be seamlessly encoded via linear constraints.

A Set Partitioning Problem (SPP) lies at the core of the formulation. The objective is to minimize an aggregate residual-based cost, while the constraints enforce that each time index of the series is covered exactly once by the selected segments. No restriction is imposed on the functional form of the segment model: it may consist of linear regression (yielding piecewise linear regression), but also of nonlinear or nonparametric specifications. Moreover, different runs may be associated with different model classes without altering the structure of the optimization problem.

Formally, let $X = [x_j]$, $j = 1, \ldots, nruns$, denote the binary decision variables, where $x_j = 1$ if run j is selected and $x_j = 0$ otherwise. Feasible runs may be generated so as to incorporate preliminary dominance rules, for instance by excluding segments that are too short or too long. Each run j is assigned a cost c_j, computed according to any suitable goodness-of-fit or loss metric proposed in

 doi:10.20944/preprints202603.1640.v1

the literature, including R^2, SER, χ^2, RMSE, variance-based criteria, etc. The partitioning / covering constraints ensure that every series point is included in (exactly) one selected run.

The resulting Time Series Set Partitioning (TSSP) model is defined as follows. Let $a_{ij} \in \{0,1\}$ be a coefficient equal to 1 if and only if run j covers time index i, for $i = 1,\ldots,npoints$ and $j = 1,\ldots,nruns$. In formulation TSSP we also included a constraint on the cardinality of the solution set.

$$(TSSP) \quad z_{TSSC} = \min \sum_{j=1}^{nruns} c_j x_j \tag{1}$$

$$s.t. \sum_{j=1}^{nruns} a_{ij} x_j = 1, \qquad i = 1,\ldots,npoints \tag{2}$$

$$\sum_{j=1}^{nruns} x_j \leq maxruns \tag{3}$$

$$x_j \in \{0,1\}, \qquad j = 1,\ldots,nruns \tag{4}$$

Additional variables can be included in the model associating them with the X decision variables, for example it is possible to associate the starting time s_i, the end time e_i with each variable $x_i \in X$. While redundant in this formulation, they might become important to express the constraints listed in subsection 3.2.

Unfortunately, in real-world applications the number of feasible runs can be very large, making the resulting SPP computationally demanding and, in some cases, intractable within acceptable time limits. From a polyhedral perspective, the set partitioning formulation is typically tighter but also harder to explore computationally, as the equality constraints induce a more restrictive feasible region and a combinatorially demanding branching structure. To mitigate this issue, we can relax the equality constraints 3 into inequalities, thereby transforming the formulation into a Set Covering Problem (SCP). The covering polyhedron is generally easier to handle algorithmically and can be solved in significantly larger dimensions, albeit with a weaker linear programming relaxation.

The SCP relaxation allows multiple runs to cover the same series points, and therefore a postprocessing phase is required to extract a feasible segmentation from the (possibly overlapping) covering solution. The resulting set covering formulation is denoted by TSSC.

State-of-the-art MIP solvers are highly effective on SCP formulations; however, very large instances may still entail substantial computational effort. To further enhance scalability, [20] proposed a Lagrangian matheuristic [21] for the TSSC problem where all covering constraints (2) are relaxed by associating a Lagrangian multiplier λ_i with each constraint, $1 \leq i \leq npoints$, while retaining only the relaxed covering constraint in the formulation. The resulting Lagrangian model is solved via a subgradient optimization scheme. The corresponding Lagrangian subproblem is computationally simple: at each iteration, it amounts to selecting at most the *maxruns* variables with the most negative reduced costs. A simple fixing heuristic is embedded within the procedure: the incumbent solution of the subproblem, which may be infeasible with respect to the relaxed covering constraints, is iteratively augmented by adding selected variables until feasibility is restored. This relaxation makes it possible to easily include constraints such as the maximum number of segments allowed. However, in most cases, additional constraints must be added to the subproblem or relaxed themselves, which makes the approach impractical.

In any case, after solving a set covering relaxation, postprocessing is required to transform any SCP solution into a feasible SPP segmentation. This can be accomplished in linear time by scanning each region of overlapping runs and identifying the splitting point that minimizes the total cost obtained by assigning the preceding observations to the preceding segment and the subsequent observations to the following segment.

 doi:10.20944/preprints202603.1640.v1

3.1. Cost Function

Models TSSP and TSSC are suitable for offline (or anytime) analysis assuming additive costs. The model is independent on the specific function used to quantify the cost associated with each segment. The CPD literature proposes different cost functions, including negative log-likelihood (Horvath 1993; Chen and Gupta 2000), quadratic loss or cumulative sums (e.g., Inclan and Tiao 1994; Rigaill 2010), but any statistical measure of fit can be used for the task. Each measure favors different characteristics of the model and the results can be widely different.

In practical applications of CPD, model quality is often judged less by its formal optimality properties and more by the interpretability of the resulting segmentation. In many real-world settings, the most useful segmentation is not necessarily the one that minimizes a given objective function, but rather the one that aligns with human perception of structural changes or provides a coherent narrative for subsequent decision-making. Accordingly, the utility of a cost function may depend as much on the clarity and explanatory power of its segmentation as on its predictive accuracy.

To reconcile algorithmic outputs with visual or domain-based expectations, many selection procedures incorporate penalty terms that regulate the number of detected changepoints. These penalizations effectively allow practitioners to adjust the segmentation to better match subjective expectations. However, the resulting models should be regarded as practically useful approximations rather than representations of an underlying "true" model, which is unlikely to be included among the finite set of candidate models considered in practice [10].

A further limitation of user-defined penalization schemes is that they implicitly require human feedback. In heterogeneous datasets, appropriate penalty calibration often needs to be tailored on a per-series basis, which limits scalability and automation. To reduce this dependency on manual tuning, we evaluated several alternative cost functions drawn from both information-theoretic and optimization-based frameworks, with the aim of identifying a more robust and resilient formulation suitable for automated CPD analysis.

We evaluated a broad range of cost functions, including AIC, BIC, Chi-square statistics, MSE, QRMSE, R^2, RMSE, SER, and simple variance; comparative visual results on a standard benchmark instance are represented in Figure 1. No single criterion consistently dominates the others across all benchmark instances. Performance varies depending on the underlying signal structure, noise characteristics, and segment length distribution, supporting the claim that cost-function performance is inherently context-dependent.

For the subsequent analysis, we focus on two representative and complementary criteria: AIC and QRMSE. AIC is well established in the CPD literature, particularly through its equivalence to penalized negative log-likelihood formulations, and provides a theoretically grounded likelihood-based benchmark.

QRMSE (Quarter-root-RMSE) [20], by contrast, lies between MSE and RMSE in terms of segment-length sensitivity. Relative to MSE, it places less emphasis on short segments, while avoiding the stronger preference for long segments induced by RMSE. This intermediate behavior makes it attractive in applications where neither aggressive segmentation nor excessive smoothing is desirable. Specifically, QRMSE is obtained from the residual sum of squares as

$$QRMSE = \sqrt{\frac{SSE}{\sqrt{n}}}$$

This formulation reduces the implicit penalty associated with long segments compared with classical RMSE or MSE-based criteria. As a consequence, longer segments are penalized less strongly and the resulting segmentations tend to be smoother.

Importantly, although QRMSE is additive across segments and can therefore be used within optimal partitioning frameworks, it does not satisfy the inequality condition required for pruning in algorithms such as PELT. In particular, the cost of a merged segment may be lower than the sum of the costs of two adjacent subsegments. This violation prevents the use of the pruning guarantees that

 doi:10.20944/preprints202603.1640.v1

underpin the computational efficiency of such methods, limiting the practical applicability of QRMSE in large-scale settings.

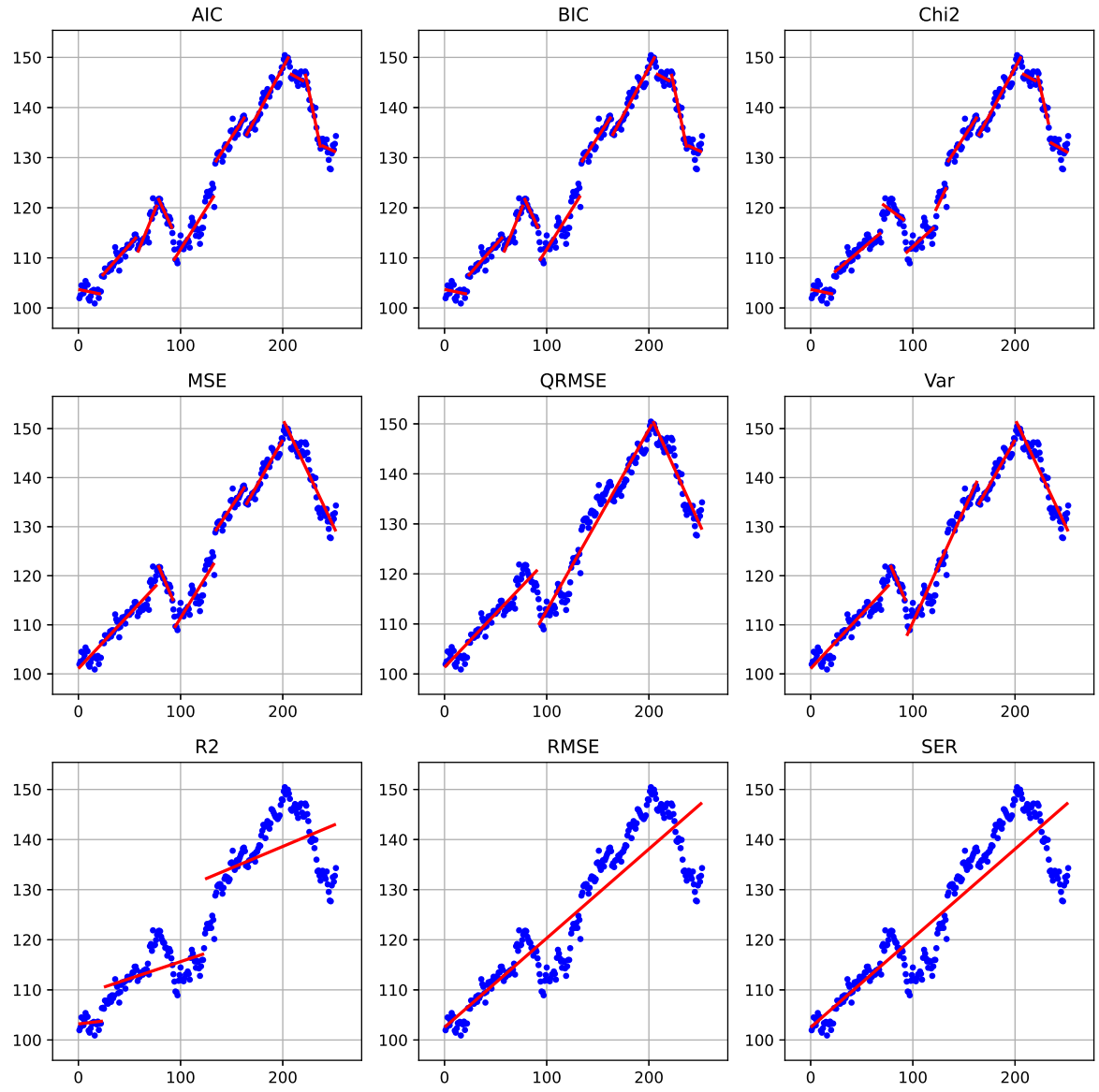


Figure 1. Alternative cost functions on series ROST of benchmark M6.

3.2. Additional Constraints

In many real-world applications, the analyst is not only interested in minimizing a fitting error, but also in enforcing interpretable and domain-driven properties on the resulting segmentation. Many changepoint frameworks encode these ideas via global penalization rather than hard constraints—using penalties on the number of changepoints (*L0*), total variation (*L1*), or graph-structured transitions that enforce sign alternation or monotone regimes across the entire sequence. These approaches provide a way to control global behaviors, such as roughness or regime structure, and are usually implemented by dynamic programming or specialized algorithm limiting the possibility to flexibly expand the proposed model.

These are the constraints that "operate over the whole segmentation" and are discussed in the global constraint catalogues and changepoint detection literature (Beldiceanu, Régin; O'Hearn; Gionis; Killick; etc.), as well as in application papers spanning different areas, including bioinformatics, econometerics, signal processing among others.

 doi:10.20944/preprints202603.1640.v1

Let $\mathcal{I} = 1, \ldots, nruns$ be the index set of feasible segments, where each segment is associated with its start and end indices (s_i, e_i), $i \in I$, with length $\ell_i = e_i - s_i + 1$, slope β_i (in the case of linear regression), fitting cost c_i and endpoint values (v_i^1, v_i^2).

For the sake of our discussion, it is important to distinguish between two classes of constraints: those that affect the feasibility of individual segments independently of the remainder of the sequence, and those that impose conditions on the sequence as a whole. Constraints in the first class can be naturally incorporated into the segment generation process and are readily accommodated within dynamic programming frameworks. In contrast, constraints in the second class require the introduction of additional state variables to track global properties, thereby increasing the computational burden of dynamic programming approaches.

3.2.1. Local Segment Constraints

Segment-level constraints restrict the admissible properties of each segment independently of the rest of the segmentation. These constraints can be implemented in the data generation phase, do not require additions to formulations TSSP or TSSC and are easily included in DP approaches. Typical segment-level constraints include:

- *Minimum and maximum segment length.* Bounds on the length of each segment (minimum spacing between changepoints) prevent over-fragmentation and excessively long segments In optimal partitioning approaches, minimum segment size constraints can be naturally incorporated. For instance, the PELT algorithm [7] allows restricting the set of admissible segmentations by enforcing a lower bound on segment length and also by limiting admissible segment endpoints according to a predefined minimum segment size. This constraint is expressed as

$$\ell_{\min} \leq e_i - s_i + 1 \leq \ell_{\max} \qquad i \in \mathcal{I} \tag{5}$$

- *Slope constraints*. When segments are modeled via linear regression, constraints can be imposed on segment-specific parameters, such as the slope β_i. More in general, constrained changepoint detection under affine inequality restrictions between segment parameters has been studied [22]. Typical examples include bounded slopes, or directional constraints enforcing locally non-decreasing or non-increasing behavior.

$$|\beta_i| \leq \beta_{\max} \qquad i \in \mathcal{I} \tag{6}$$

- *Boundary exclusion*. Changepoints can be disallowed in the first and last K observations of the series to ensure stable parameter estimation. This can be enforced by restricting admissible start and end indices:

$$s_i \geq K + 1 \quad \text{and} \quad e_i \leq n - K \qquad i \in \mathcal{I} \tag{7}$$

  where n denotes the length of the series.
- *Admissible breakpoint positions*. Changepoints may be restricted to a predefined set $\mathcal{A} \subseteq \mathcal{I}$ of admissible positions, such as calendar or seasonal markers. For example, in the Ruptures framework [3], the jump parameter enforces that changepoints can only be selected on a subsampled grid of indices, effectively reducing the admissible set of breakpoint locations. In this case, segment boundaries must satisfy:

$$s_i, e_i \in \mathcal{A}. \tag{8}$$

  These restrictions are useful both for computational efficiency and for incorporating prior structural information into the segmentation model.

3.2.2. Global Series Constraints

In contrast to segment-level constraints, global series constraints impose conditions that couple multiple segments and operate on the segmentation as a whole. Mixed-Integer Programming for-

 doi:10.20944/preprints202603.1640.v1

mulations can support several types of constraints belonging to this class through possibly linear inequalities linking multiple segment-selection variables.

A list of global constraints reported in the literature is presented below. We note that several of these constraint classes generate a number of constraints that grows superlinearly with the number of feasible segments (often quadratically in practice). Explicitly including all such constraints in the model is therefore impractical. Instead, we anticipate incorporating them via a cutting-plane strategy: constraints are added on demand as violated cuts because their separation can be performed efficiently for the constraint types considered.

- *Global bound on the number of segments.* A global bound on the number of selected segments controls the overall model complexity. This constraint can be written as

$$\sum_i x_i \leq n_{\max}. \tag{9}$$

  Constrained dynamic programming algorithms under this setting are discussed in [22].
- *Global budget on segmentation cost*. The total cost of the segmentation can be bounded:

$$\sum_i c_i x_i \leq B, \tag{10}$$

  where c_i denotes the cost of segment j. This constraint can be seen as the constrained version of penalized segmentation model, which can be then relaxed in a Lagrangian fashion to obtain the penalized model itself [7]. A framework for selecting number of changepoints via information criteria is also proposed in [23].
- *Global total variation*. A similar global constraint places a bound on the aggregate magnitude of all changes across the series:

$$\begin{aligned} v_{min} &\leq min(v_i^1 x_i, v_i^2 x_i) \qquad i \in \mathcal{I} \\ v_{max} &\leq max(v_i^1 x_i, v_i^2 x_i) \qquad i \in \mathcal{I} \\ v_{max} - v_{min} &\leq \delta_{max} \end{aligned} \tag{11}$$

  This constraint controls the overall amount of structural variation in the segmentation by limiting the cumulative magnitude of successive jumps between chosen segments. It is closely related to total variation regularization and to the lasso formulation of [24], where a penalty is applied to successive parameter differences.
- *Global shape constraints (monotonicity, convexity, symmetry).* These constraints impose structural properties on the segmentation as a whole, rather than on individual segments. For example, global monotonicity or convexity can be enforced by requiring the sequence of segment parameters to satisfy a prescribed ordering, such as ensuring a nondecreasing evolution of segment means. The formulation here is more complex as it is necessary to identify successive segments in the model. To this end, let $\mathcal{A} = \{(i,j) \in \mathcal{I} \times \mathcal{I} : s_j = e_i + 1\}$ be the set of indices of successive, adjacent segments and let $\xi_{ij}, (i,j) \in \mathcal{A}$, be binary decision variables equal to 1 iff the two adjacent segments of indices i and j are in the solution. Then a monotonic nondecrease of means can be expressed as

$$\begin{aligned} \sum_{j:(i,j)\in\mathcal{A}} \xi_{ij} = x_i; \sum_{i:(i,j)\in\mathcal{A}} \xi_{ij} = x_j \qquad i \in \mathcal{I} \\ \mu_i - \mu_j \leq M(1 - \xi_{ij}) \qquad (i,j) \in \mathcal{A} \end{aligned} \tag{12}$$

  where the μ_i are the mean values of the segments indexed by $i \in \mathcal{I}$. More generally, convexity or concavity can be imposed by constraining first or second differences of the segment parameters to follow a specified sign pattern.

 doi:10.20944/preprints202603.1640.v1

A related formulation limits the number of monotone episodes across the segmentation. For example, one may allow at most n_r runs of consecutive nondecreasing (or nonincreasing) segments, thereby restricting the global alternation of trends.
Symmetry constraints also fall within this class. These may require changepoint locations to be symmetric with respect to the midpoint of the series, or impose mirrored jump magnitudes, as appropriate for symmetric processes.

- *Global pattern constraints on change directions.* These constraints require the sequence of segment transitions to follow a prescribed qualitative pattern. For example, the model may be forced to follow a structure such as increase–plateau–decrease, to alternate increases and decreases in a regular manner, or to follow a periodic regime sequence (e.g., stable–volatile–stable). Similarly, one may regulate the occurrence of peaks, where a peak is defined as an increase in segment means immediately followed by a decrease.
  Such constraints may either enforce a specific pattern or restrict its complexity (e.g., by bounding the number of direction alternations) [25]. They can also be formulated negatively as forbidden-pattern constraints, excluding certain global configurations. Examples include limiting the total number of alternations to at most n_a, or forbidding recurring structures such as repeated "short–long–short" segment sequences. These formulations amount to global pattern-avoidance constraints on the segmentation.
- *Global structural break frameworks* (econometrics). Econometric "structural break" models can be seen as global constraints, as they formalize constraints on the solution such as maximum number of structural breaks, with simultaneous estimation of break locations [26,27].
- *Equal or bounded segment sizes*: Require near-equal segment lengths or bound length variability (e.g., max/min segment length ratio), affecting the global partition shape. If δ_{len} is the maximum allowed difference among segment lengths ($\delta_{len} = 0$ in case of equal segment lengths), the constraint is

$$|(e_i - s_i) - (e_j - s_j)| \leq \delta_{len} \qquad i, j \in \mathcal{I}. \tag{13}$$

  The acceptable length difference δ_{len} can given as input, or it can be optimized itself.

The above lists are not intended to be exhaustive of all possible requirements that may be imposed on time series segmentation problems; rather, they are restricted to constraints that can be incorporated within an extension of our current modeling framework. For example, a global constraint on the number of distinct regimes—commonly used in regime-switching models and implemented in systems such as SETAR [28]—would require an explicit regime identification mechanism, which is not presently supported by our framework.

Similarly, certain global structural constraints, such as net drift constraints that enforce the signed sum of jumps to equal a specified value (e.g., zero), thereby ensuring that the process begins and ends at comparable levels, are not directly accommodated. Ensuring feasibility under such requirements would necessitate a joint search over segmentations and model parameters. In particular, restricting the search to a set of a priori fitted segments would be insufficient; instead, parameter optimization would need to be performed within the segmentation procedure itself.

Finally, we point out that the explicit modelling of such constraints can, in most cases, be effectively achieved also using constraint programming techniques [29]. Constraint programming is deeply rooted in mathematical programming itself, and in recent years the two areas have arguably moved closer to one another. In particular, for global CPD constraints, one can exploit classical constraint programming constructs such as *regular*, which enforces that a sequence of decision variables is accepted by a finite automaton, or global constraints such as *nvalues* or *count*, which bound, respectively, the number of distinct values or the occurrences of specific values across a set of variables. Nevertheless, mixed-integer programming remains a more general and flexible modelling framework, capable of accommodating a broader range of constraints than current constraint-programming systems.

 doi:10.20944/preprints202603.1640.v1

4. Computational Experience

We implemented models TSSP and TSSC in C++ and ran them on a Windows 11 Intel i7 machine equipped with 32 Gb of RAM. The MIP solver used was CPLEX 22.11. All codes and data are available from the project repository.

4.1. Dynamic Programming Approaches

This section reviews three established baselines for offline CPD: exact dynamic programming with a fixed number of change points (called *Dynp* in ruptures), the PELT algorithm for penalized optimal partitioning and the Wild Binary Segmentation (WBS). The first two methods are globally optimal for their respective objective function under a segment-additive structure while the WBS method is a randomized recursive strategy that does not solve a single global optimization problem.

4.1.1. Additive Segmentation Framework

Let $y_{1:n} = (y_1, \ldots, y_n)$ be a univariate time series. A segmentation with k change points is defined by indices $0 = \tau_0 < \tau_1 < \cdots < \tau_k < \tau_{k+1} = n$. Consequently, the k changepoints split the data into $k+1$ segments, with the i-th segment containing $y_{(\tau_{i-1}+1):\tau_i}$.

A broad class of CPD methods assumes that the signal can be modeled piecewise, and that the quality of a segmentation can be expressed through an additive objective of the form

$$\min_{\tau_1,\ldots,\tau_k} \sum_{j=1}^{k+1} \mathcal{C}(\tau_{j-1}, \tau_j), \tag{14}$$

where $\mathcal{C}(\tau_{j-1}, \tau_j)$ denotes the cost associated to the segment $y_{(\tau_{j-1}+1):\tau_j}$.

The specific form of $\mathcal{C}$ depends on the modeling assumptions and leads to different segmentation results.

Two optimization settings arise from (14):

1. Constrained formulation: the number of change points k is fixed.
2. Penalized formulation: the number of change points is selected automatically by minimizing

$$\min_{k,\tau_1,\ldots,\tau_k} \sum_{j=1}^{k+1} \mathcal{C}(\tau_{j-1}, \tau_j) + \beta k, \tag{15}$$

where $\beta > 0$ is a penalty parameter.

In particular, the Dynp algorithm solves the constrained problem exactly, whereas the PELT algorithm solves the penalized problem exactly through pruning.

4.1.2. Dynp: eXact Dynamic Programming with Fixed k

When the number of change points k is fixed, the global minimizer of (14) can be computed via dynamic programming. This approach corresponds to the classical Segment Neighbourhood algorithm introduced by [5], and is implemented in the Ruptures library under the name *Dynp* [30].

Define

$$F(t, k)$$

as the minimum cost of segmenting the prefix $y_{1:t}$ using exactly k change points. The recursion is

$$F(t, k) = \min_{s<t} \{F(s, k-1) + \mathcal{C}(s, t)\}, \tag{16}$$

with initialization

$$F(t, 0) = \mathcal{C}(0, t).$$

The optimal segmentation with k change points is obtained from $F(n, k)$, and the corresponding change points are recovered by backtracking the minimizers s at each stage. Any optimal segmentation

 doi:10.20944/preprints202603.1640.v1

of $y_{1:t}$ with k change points must consist of an optimal segmentation of some prefix $y_{1:s}$ with $k-1$ change points, followed by one final segment $(s,t]$. The computational complexity is quadratic in n for each k, leading to an overall complexity of order $\mathcal{O}(kn^2)$ (up to the cost of evaluating $\mathcal{C}$). The Ruptures implementation includes practical mechanisms to reduce computational burden: a minimum segment length constraint unstable very short segments and a jump parameter restricts candidate change point positions.

4.1.3. PELT: Penalized Optimal Partitioning with Pruning

The PELT algorithm [7] solves the penalized problem (15) exactly. Let

$$G(t)$$

denote the minimum penalized cost of segmenting $y_{1:t}$. The dynamic programming recursion is

$$G(t) = \min_{s<t}\{G(s) + \mathcal{C}(s,t) + \beta\}, \tag{17}$$

with initialization $G(0) = -\beta$.

The number of change points k is not fixed but determined by the optimization through the penalty term. The key contribution of PELT is a pruning rule that maintains a candidate set of potential last changepoints and discards those that can be proven suboptimal for all future $t' > t$. This rule relies on a regularity condition on the segment cost function, typically expressed as a superadditivity-type inequality up to an additive constant,

$$\mathcal{C}(s+1,t) \geq \mathcal{C}(s+1,u) + \mathcal{C}(u+1,t) + K, \quad s < u < t, \tag{18}$$

for some constant K. This condition is satisfied by several commonly used costs, such as those based on (minus) log-likelihood or AIC-type criteria, but may fail for others, such as RMSE-based costs.

It can be shown that the expected number of retained candidates remains bounded as n increases. Consequently, the expected computational complexity becomes linear in n. As in Dynp, the Ruptures implementation of PELT includes minimum segment length and candidate grid restrictions.

4.1.4. Wild Binary Segmentation (WBS)

Wild Binary Segmentation [31] is conceptually different from the previous two methods. It does not directly optimize (14) or (15). Instead, it extends classical binary segmentation through randomized interval localization. Classical binary segmentation detects a single change point over the full interval $(0,n]$, splits the signal at that location, and recurses on the left and right subintervals. However, when multiple change points are close to each other, the global constrant over the full interval may blur their individual effects. WBS addresses this limitation by sampling M random intervals

$$[s_m, e_m] \subseteq \{1,\ldots,n\}, \quad m = 1,\ldots,M.$$

On each interval, a single change point contrast statistic is computed and the location maximizing the absolute constrast within that interval is identified. Among all intervals, the candidate achieving the largest constrast is selected as the estimated change point, provided the contrast exceeds a predefined threshold. The procedure then recurses on the subintervals induced by the detected change point, using only random intervals fully contained in each subproblem. Unlike Dynp and PELT, WBS does not guarantee global optimality with respect to an additive objective. Its performance depends on the number of sampled intervals M, the constrast definition and the stopping rule (threshold-based). Nevertheless, it is often computationally efficient when change points are numerous.

 doi:10.20944/preprints202603.1640.v1

4.2. Benchmarking and Comparative Performance

The computational results are presented on a selection of time series drawn from the M3, M4, and M6 forecasting competitions. These datasets originate from the well-known Makridakis (M) competitions [32], which constitute one of the most influential large-scale benchmarking initiatives in forecasting research. Because of their size, diversity, and public availability, the M-competition datasets have become standard testbeds for evaluating methodological advances in time series analysis and forecasting. The M5 dataset was not included due to the substantially greater length of its daily series and its large hierarchical structure, which would require computational and modeling adaptations beyond the focus of this work.

The M3 competition, conducted in 2000, comprised 3,003 real-world time series of varying frequencies (yearly, quarterly, monthly, and others) drawn from domains such as economics, finance, industry, and demography. The heterogeneity of the M3 dataset—both in terms of series length and underlying dynamics—makes it particularly suitable for assessing the robustness of segmentation and structural change detection procedures.

The M4 competition, concluded in 2018, significantly expanded the scale of the benchmark to 100,000 series, again covering multiple sampling frequencies (yearly, quarterly, monthly, weekly, daily, and hourly). The M4 dataset includes both macroeconomic and microeconomic indicators, financial series, demographic data, and other business-related measurements.

More recently, the M6 competition (2022) focused on financial time series and incorporated ex-ante forecasting in a real-time setting and emphasized portfolio construction and risk-aware evaluation. The dataset primarily consists of financial asset returns and related variables, thereby introducing features such as higher volatility, regime shifts, and structural breaks that are particularly relevant for changepoint detection methodologies.

Together, the M3, M4, and M6 benchmarks offer a diverse and demanding experimental framework. By evaluating our approach on selected series from these competitions, we aim to assess its effectiveness across different domains, sample sizes, and structural regimes representative of real-world forecasting applications. The tests were run on 5 randomly selected series for each category in M3 and M4, and on 10 series of M6 drawn from the different asset types (stock and ETF) and different categories (Health Care, Energy, Real Estate, IT, Consumer Discretionary, Equities, Commodities, Volatility) in the benchmark set.

4.2.1. Cost Functions

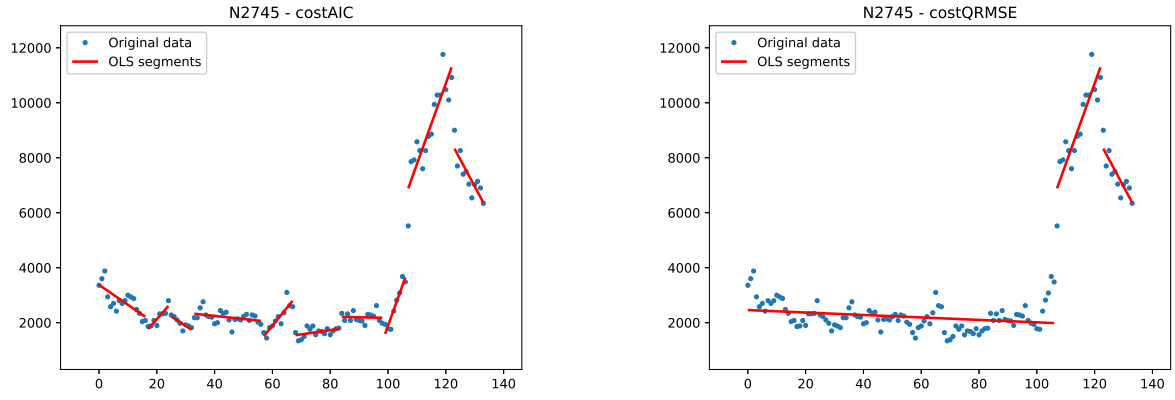
Section 3.1 introduced the considerations motivating the choice of the cost functions used in our experiments. In particular, we focused on two alternatives: AIC and QRMSE. The former is well established in the changepoint literature, while the latter, *Quarter-root RMSE*, an in-between MSE and RMSE which effectively corresponds to scaling the residual sum of squares with an exponent proportional to $n^{1/4}$ (see Section 3.1), was found in our experiments to produce segmentations that better align with visual expectations than the commonly used RMSE, of which it represents a minor variant.

Figure 2 illustrates the typical impact of these two criteria on the segmentation results for series N2745. In both cases, model TSSP was applied under the constraints that the minimum segment length be 8 observations and the maximum number of segments be 10.

When AIC is used as the cost function (Figure 2(**a**)), the resulting model exploits the full fragmentation allowed by the constraints, producing a segmentation with the maximum number of segments in order to achieve a close fit to the observed data points. In contrast, when QRMSE is used (Figure 2(**b**)), the resulting segmentation tends to favor longer segments, tolerating a higher variance of observations around the fitted segments and thereby producing a smoother representation of the series.

These results demonstrate the impact of the choice of cost function on the balance between fidelity to local fluctuations and the reduction of the number of segments.

 doi:10.20944/preprints202603.1640.v1



(**a**) AIC cost segmentation (**b**) QRMSE cost segmentation

Figure 2. Segmentations derived from different cost functions, M3 dataseries N2745, maximum 10 segments.

We emphasize that there is no universally "correct" cost function for segmentation. Different criteria emphasize different characteristics of the fitted model and may therefore align better or worse with the expectations placed on the resulting segments. Beyond these modeling preferences, the statistical properties of the series itself also play an important role in determining which cost function performs more satisfactorily.

Figure 3 reports the optimal TSSP segmentation, obtained under the same constraints as in the previous experiment, for four series belonging to different economic categories. The results highlight the considerable heterogeneity present in these datasets. Even within a single category, series may exhibit markedly different structural properties, such as variability, seasonality, or frequency of structural changes.

Because of this diversity, relying on a single cost function often entails a human-in-the-loop process in which the analyst repeatedly adjusts optimization control parameters to obtain satisfactory segmentations. In contrast, selecting a cost function whose properties are known to match the statistical characteristics of the series can already guide the optimization toward the expected segmentation, reducing or eliminating the need for manual intervention.

4.2.2. Unconstrained Segmentation

In this section we report the results obtained using the three unconstrained changepoint detection algorithms described earlier, namely PELT, Dynp and Wild Binary Segmentation (WBS). The experiments were conducted on the M3, M4 and M6 benchmark datasets in order to provide baseline results for comparison with the proposed set-covering MILP framework. Both PELT and Dynp, which are exact optimization methods, were run using the implementation provided by the Ruptures library [30].

To ensure a consistent experimental setting across all algorithms, the minimum segment length was fixed to 8 observations. This choice prevents the detection of extremely short segments and aligns the unconstrained methods with the structural constraints that will later be considered.

The parameters of the three algorithms were selected through preliminary calibration experiments carried out on a small number of representative series drawn from different dataset categories. For each method, several parameter configurations were tested and the final parameter values were chosen so as to obtain segmentations whose number of segments is broadly comparable with the segmentation structures considered in the experiments with the proposed set-covering MILP framework.

For the PELT algorithm, the penalty parameters were set to $\beta = 40$ for both the AIC function cost and the QRMSE function cost, with `jump` $= 1$ and `min_size` $= 8$.

For Dynp, the number of breakpoints was fixed to 9, corresponding to a target segmentation of 10 segments. This choice was motivated by the fact that the experiments with the proposal framework

 doi:10.20944/preprints202603.1640.v1

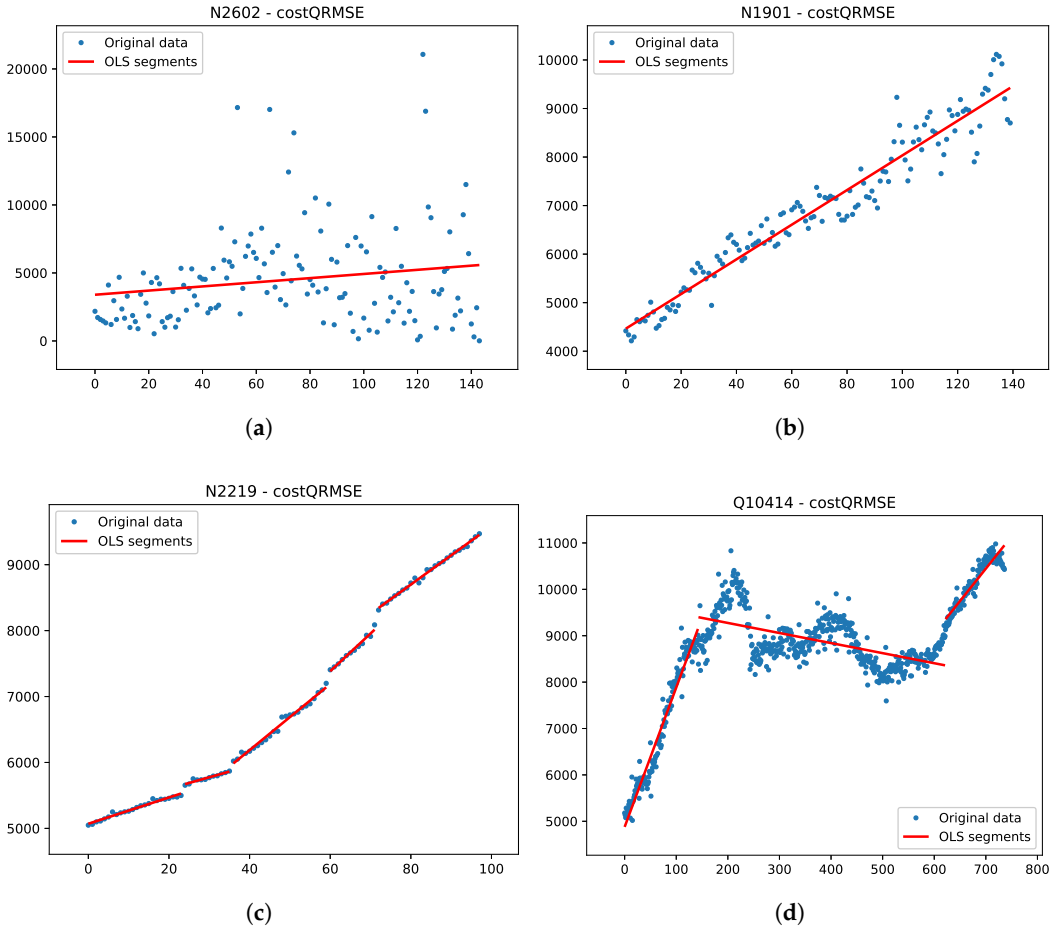


Figure 3. Different dataseries categories (**a**) Financial. (**b**) Industry. (**c**) Macroeconomics. (**d**) Microeconomics.

consider segmentations with up to 10 segments. The minimum segment length was again set to 8 observations. For shorter series, the requested number of breakpoints may not always be feasible under this constraint; in those cases, the number of breakpoints was automatically reduced to the largest admissible value.

For WBS, the number of randomly generated intervals was fixed to $M = 400$, while the minimum segment length was set to 8 observations. The threshold parameters were calibrated separately for the two cost functions and set to $\texttt{thr}_{\text{AIC}} = 20$ and $\texttt{thr}_{\text{QRMSE}} = 10$.

The results reported in Tables 1–3 highlight several differences among the three approaches. The columns show the benchmark of origin of the series, *Bench*, the category of the set of series reported in the row, *Category*, then separately for AIC and for QRMSE the median and maximum number of segments in the proposed solutions, *med.n*, *max.n*, and the median and maximum CPU time for solving, *med.t*, *max.t*. In tables 1 and 2 we also added for reference the median and maximum CPU time used by MIP (inclusive of the setup time) for producing the solutions forcing the same number of segments as in Dynp, *med.t*,*max.t*, while in table 3 we report the percentage cost gap between the solutions proposed by WBS and MIP, *med.gap* and *max.gap*.

The first observation is that the number of segments detected varies substantially depending on the algorithm and cost function used. The AIC criterion generally favours more fragmented segmentations, whereas the QRMSE criterion tends to produce smoother segmentations with fewer change points.

The influence of this formulation is clearly evident in Table 1. When the PELT algorithm is used with the same penalty parameter, $\beta = 40$, for both cost functions, the QRMSE criterion generally produces fewer segments than the AIC criterion across most dataset categories. For instance, the

 doi:10.20944/preprints202603.1640.v1

average number of detected segments decreases notably in several M4 categories under QRMSE, reflecting the criterion's reduced tendency to split long segments.

Of the three methods, PELT often produces the fewest segments, reflecting the influence of the penalty term in the objective function. Dynp, on the other hand, aims to generate segmentations closer to the desired number of ten segments since the number of breakpoints is fixed. Finally, WBS typically exhibits the greatest variability in the number of detected segments, due to the interval sampling procedure.

Of the two algorithms, PELT is usually the fastest, since its pruning strategy enables it to discard many candidate changepoints during the search process. The Dynp algorithm generally takes slightly longer to run because it relies on a dynamic programming procedure whose complexity increases with the length of the series and the number of requested breakpoints. In contrast, WBS is not an exact optimization method; it repeatedly evaluates change-point statistics over a large number of randomly generated intervals ($M = 400$ in our experiments), which increases the computational burden. Consequently, despite its heuristic approach, WBS typically requires longer CPU times compared with the other two methods.

We also ran the TSSP model while constraining it to generate exactly the same number of segments as Dynp, in order to allow a fair performance comparison in a setting compatible with the other algorithms. In this case, both TSSP and Dynp yield optimal solutions; therefore, only the computational times are reported for TSSP.

The higher asymptotic complexity and longer setup time of TSSP are evident in Tables 1 and 2. In particular, the degradation in TSSP performance becomes apparent for longer time series.

Nevertheless, computational times remain below three minutes even for instances leading to constraint matrices with nearly 1000 rows and more than 300,000 columns (see Table 6). This demonstrates the practical viability of relying on state-of-the-art solvers even for time series whose length would normally motivate data aggregation (e.g., subsampling or smoothing) for visualization purposes.

Table 1. PELT: Data series results, AIC-QRMSE cost.

		AIC				QRMSE				TSSP	
Bench	Category	med.n	max.n	med.t	max.t	med.n	max.n	med.t	max.t	med.t	max.t
M3	Demographic	3.0	7.0	0.0	0.0	3.0	4.0	0.0	0.0	0.41	0.59
M3	Finance	3.0	9.0	0.0	0.0	2.0	7.0	0.0	0.0	0.40	0.56
M3	Industry	1.0	2.0	0.0	0.0	1.0	12.0	0.0	0.0	0.55	0.69
M3	Macro	3.0	5.0	0.0	0.0	2.0	3.0	0.0	0.0	0.43	0.59
M3	Micro	1.0	2.0	0.0	0.0	1.0	1.0	0.0	0.0	0.33	0.5
M3	Other	3.0	4.0	0.0	0.0	1.0	4.0	0.0	0.0	0.27	0.45
M4	Demographic	4.0	26.0	0.0	0.1	8.0	10.0	0.0	0.0	0.30	170.68
M4	Finance	5.0	17.0	0.0	0.1	5.0	10.0	0.0	0.0	0.33	88.24
M4	Industry	5.0	10.0	0.0	0.1	6.0	7.0	0.0	0.0	0.58	71.77
M4	Macro	5.0	35.0	0.0	0.1	2.0	17.0	0.0	0.0	0.41	153.33
M4	Micro	3.0	9.0	0.0	0.1	4.0	6.0	0.0	0.0	0.30	69.87
M4	Other	7.0	28.0	0.0	0.0	2.0	9.0	0.0	0.0	1.03	55.1
M6	Finance	6.5	8.0	0.0	0.0	1.0	1.0	0.0	0.0	2.72	3.10

 doi:10.20944/preprints202603.1640.v1

Table 2. DYNP: Data series results, AIC-QRMSE cost.

		AIC				QRMSE				TSSP	
Bench	Category	med.n	max.n	med.t	max.t	med.n	max.n	med.t	max.t	med.t	max.t
M3	Demographic	10.0	10.0	0.0	0.0	10.0	10.0	0.0	0.1	0.41	0.59
M3	Finance	10.0	10.0	0.0	0.1	10.0	10.0	0.0	0.0	0.40	0.56
M3	Industry	10.0	10.0	0.0	0.0	10.0	10.0	0.0	0.0	0.55	0.69
M3	Macro	10.0	10.0	0.0	0.1	10.0	10.0	0.0	0.0	0.43	0.59
M3	Micro	8.0	10.0	0.0	0.0	8.0	10.0	0.0	0.0	0.33	0.5
M3	Other	8.0	10.0	0.0	0.0	8.0	10.0	0.0	0.0	0.27	0.45
M4	Demographic	10.0	10.0	0.0	2.5	10.0	10.0	0.0	2.8	0.30	170.68
M4	Finance	10.0	10.0	0.0	2.5	10.0	10.0	0.0	2.7	0.33	88.24
M4	Industry	10.0	10.0	0.0	2.5	10.0	10.0	0.0	2.7	0.58	71.77
M4	Macro	10.0	10.0	0.0	3.7	10.0	10.0	0.0	4.1	0.41	153.33
M4	Micro	10.0	10.0	0.0	2.5	10.0	10.0	0.0	2.7	0.30	69.87
M4	Other	10.0	10.0	0.1	2.2	10.0	10.0	0.1	2.4	1.03	55.1
M6	Finance	10.0	10.0	0.2	0.2	10.0	10.0	0.2	0.2	2.72	3.10

Finally, Table 3 compares the relative effectiveness of TSSP and WSB by reporting the percentage deviation of the cost of the solution produced by WSB from that of TSSP, computed on the same instance and for the same number of segments. The median percentage error of WSB is generally small, with values below 3% and equal to 0% for the M6 series. However, in some cases the heuristic fails to converge to solutions of acceptable quality.

Table 3. WBS: Data series results, AIC-QRMSE cost.

		AIC				QRMSE				TSSP	
Bench	Category	med.n	max.n	med.t	max.t	med.n	max.n	med.t	max.t	med.gap	max.gap
M3	Demographic	10.0	11.0	0.3	0.3	5.0	8.0	0.2	0.3	1.30	9.26
M3	Finance	9.0	11.0	0.2	0.2	6.0	12.0	0.2	0.3	1.28	8.62
M3	Industry	5.0	12.0	0.2	0.5	2.0	14.0	0.1	0.4	0.89	64.41
M3	Macro	8.0	11.0	0.2	0.3	5.0	6.0	0.2	0.3	0.25	3.84
M3	Micro	2.0	6.0	0.1	0.3	1.0	1.0	0.0	0.1	0.47	1.83
M3	Other	5.0	6.0	0.1	0.1	4.0	6.0	0.1	0.1	0.49	7.61
M4	Demographic	9.0	55.0	0.2	2.9	11.0	37.0	0.2	2.6	5.57	29.54
M4	Finance	10.0	56.0	0.2	2.7	9.0	39.0	0.2	2.3	7.15	60.91
M4	Industry	10.0	27.0	0.3	2.7	8.0	11.0	0.3	2.0	3.31	26.15
M4	Macro	9.0	69.0	0.2	3.3	7.0	50.0	0.2	3.2	3.30	44.30
M4	Micro	8.0	22.0	0.2	2.6	7.0	9.0	0.2	2.1	2.29	19.43
M4	Other	10.0	51.0	0.4	2.5	8.0	31.0	0.3	2.2	3.58	29.82
M6	Finance	17.0	21.0	0.7	0.8	1.0	3.0	0.2	0.4	0.00	33.05

4.2.3. Constrained Segmentation

In this section, we consider globally constrained series; therefore, the only suitable models for application are the TSSP and its heuristic relaxation, the TSSC. In the examples, the models were constrained to propose a maximum of 10 segments (rather than an exact number of 10, as managed by Dynp), as well as a minimum segment length of 8. However, any of the additional constraints listed in 3.2 could have been included.

The results are summarized in tables 4 and 5. The columns show compatible data in both tables, that is the benchmark set of origin of the series (*bench*), the category of the series (*category*), the average and maximum number of data points in the group series (*avg.n* and *max.n*), the corresponding average and maximum number of generated runs (*avg.nrun* and *max.nrun*), the average and maximum number of chosen segments (*avg.nseg* and max.nsegm), and the average and maximum CPU time in seconds to solve the extended SCP model, inclusive of the time needed to generate the runs (*avg.t* and *max.t*).

 doi:10.20944/preprints202603.1640.v1

Table 4. Data series results, QRMSE cost

bench	category	avg.n	max.n	avg.nrun	max.nrun	avg.nseg	max.nseg	avg.t	max.t
M3	Demographic	122.6	138	7073.2	8646	3.8	7	0.6	1
M3	Finance	124.8	144	7267.6	9453	3.6	9	0.1	0.1
M3	Industry	139	144	8789.2	9453	1	1	0.1	0.1
M3	Macro	128	144	7532.2	9453	2	5	0.1	0.1
M3	Micro	91.8	126	4027.8	7140	1	1	0.1	0.1
M3	Other	80.8	120	2952.2	6441	3.4	6	0.1	0.1
M4	Demographic	241.8	735	54101.2	245350	4	7	57.6	288
M4	Finance	246	738	54945.4	247456	6.2	10	61.6	308
M4	Industry	258.4	737	56492	246753	4.6	7	60	298
M4	Macro	280	866	74559.8	339900	5.6	10	106.6	532
M4	Micro	232.8	737	53371.2	246753	2.8	7	58.4	292
M4	Other	260.4	701	53106.2	222778	4.8	10	57.6	286
M6	Finance	257.8	265	30394.2	32131	4.2	7	7.0	25

Table 4 reports aggregate results obtained with the TSSP model using QRMSE as the cost function. The results show that relatively short series—up to a few hundred observations—lead to integer programming instances of moderate size that can be solved very efficiently even on standard hardware.

As the length of the series increases, the non-polynomial complexity of the exact integer programming search becomes more apparent. When the series approaches about 500 observations, solution times begin to require minutes of CPU time. This increase is primarily due to the rapid growth of the IP formulation, whose constraint matrix (the "tableau") may reach dimensions close to 1000 rows and more than 300,000 columns.

Table 5. Data series results, AIC cost.

bench	category	avg.n	max.n	avg.nrun	max.nrun	avg.nseg	max.nseg	avg.t	max.t
M3	Demographic	122.6	138	7073.2	8646	9.4	10	0.2	1
M3	Finance	124.8	144	7267.6	9453	9.6	10	0.1	0.1
M3	Industry	139	144	8789.2	9453	10	10	0.1	0.1
M3	Macro	128	144	7532.2	9453	10	10	0.1	0.1
M3	Micro	91.8	126	4027.8	7140	6.6	10	0.1	0.1
M3	Other	80.8	120	2952.2	6441	7.4	9	0.1	0.1
M4	Demographic	241.8	735	54101.2	245350	10	10	58.4	292
M4	Finance	246	738	54945.4	247456	10	10	65.2	325
M4	Industry	258.4	737	56492	246753	8.8	10	62.4	310
M4	Macro	280	866	74559.8	339900	10	10	105.8	528
M4	Micro	232.8	737	53371.2	246753	10	10	60.4	302
M4	Other	260.4	701	53106.2	222778	10	10	57.8	285
M6	Finance	257.8	265	30394.2	32131	10.0	10	5.9	7

Table 5 reports the results obtained on the same set of series when AIC is used as the cost function. Since the cost function affects only the data preparation phase—namely the computation of the costs associated with the feasible segments—the size of the resulting IP instances remains unchanged, and the dimensions of the constraint matrices are therefore identical to those reported in Table 4.

The CPU time required for the optimization, however, is influenced by the numerical values of the costs, although this effect is relatively limited. The most notable difference between Tables 4 and 5 concerns the number of segments in the optimal solutions. When AIC is used, the optimal segmentation frequently saturates the imposed upper bound on the number of segments, whereas QRMSE tends to produce solutions with a consistently smaller number of segments.

 doi:10.20944/preprints202603.1640.v1

Table 6. Partitioning vs. covering comparison.

name	rows	cols	nonzeros	truns	GAP	tSSP	tSSC
Q13050	736	245350	66244500	2	0.06	219	162
Q22221	739	247456	67060576	4	0.81	294	119
Q15697	738	246753	66787812	4	0.16	275	201
Q5109	867	339900	108201500	6	0.60	467	211
Q10414	738	246753	66787812	3	0.33	326	197
Q23661	702	222778	57476724	6	0.99	242	143

A final remark concerns the relative effectiveness of the two models, TSSP and TSSC. This comparison is reported in Table 6. The columns show: the instance name *name*, the number of rows *rows* and columns *cols* of the constraint matrix, and the number of nonzero coefficients *nonzeros*. The remaining columns report the CPU time, in seconds, required to generate all feasible segment runs *truns*, to solve the instance using model TSSP *tSSP*, and to solve it using model TSSC *tSSC*. The latter includes the time required by the fixing heuristic used to ensure feasibility of the final TSSC solution. Column *GAP* reports the percentage cost gap between the heuristic TSSC solution and the optimal TSSP solution.

The table summarizes results for the longest series included in our benchmark set. The figures highlight the limited computational burden associated with the preliminary generation of segment runs, as well as the effectiveness of modern MIP solvers, which are able to handle matrices of substantial size and densities exceeding 36% within a few minutes of CPU time. As expected, the *TSSP* model requires longer solution times, whereas *TSSC* is on average about 40% faster. The fixing heuristic runs in negligible time and produces solutions whose average optimality gap with respect to *TSSP* remains consistently below 1%.

4.2.4. Nonlinear Models

We emphasize that the validity of the proposed framework is not restricted to piecewise linear regression: any parametric or nonparametric trend function can be adopted for computing the segment-specific error cost. Figure 4 illustrates the segmentation obtained by applying the model to the time series of total infected patients in Italy during the first year and a half (500 data points) of the Covid-19 pandemic. The epidemic evolved through successive waves, each of which can be effectively approximated by a trend following a negative binomial pattern. Accordingly, the formulation of Section 3 is applied by computing, for each feasible run, the fitting error associated with a negative binomial model. The subsequent optimization procedure remains unchanged.

As a final remark, the changepoints identified on this Covid-19 series admit a clear epidemiological interpretation, as they correspond to phases in which a new wave of contagion emerged or was reasonably expected to emerge.

 doi:10.20944/preprints202603.1640.v1

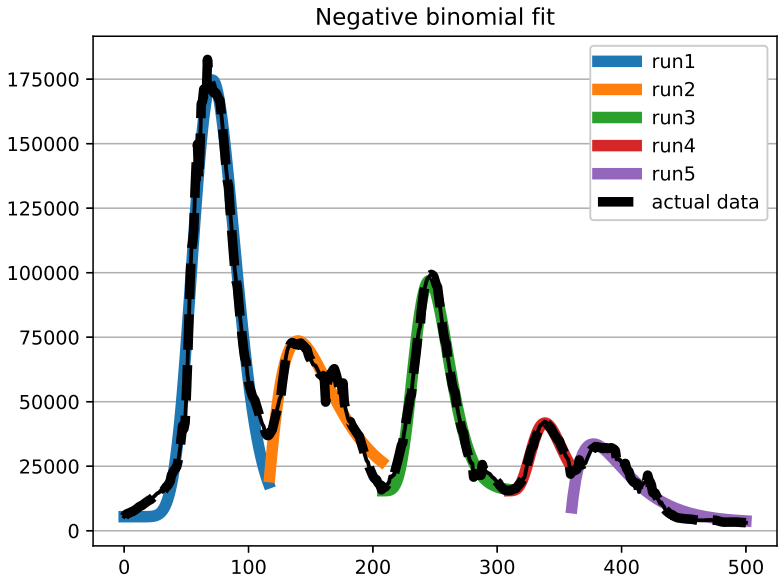


Figure 4. Covid diffusion in Italy.

5. Conclusions

In this work, we proposed a novel formulation of the changepoint detection (CPD) problem based on an extended set partitioning (SPP) model, together with a relaxed set covering (SCP) variant. The proposed approach provides an exact optimization framework that remains computationally competitive with state-of-the-art dynamic programming methods on instances of moderate size in settings where such methods are applicable.

A key advantage of the proposed formulation lies in its modeling flexibility. By representing feasible segments as atomic decision variables, the framework can naturally accommodate a wide range of structural and global constraints that are difficult or impossible to incorporate within classical dynamic programming approaches. This allows the method to address a broader class of CPD problems, extending its applicability beyond traditional settings.

Computational results on a diverse collection of benchmark time series, including those from the M3, M4, and M6 competitions, demonstrate the practical viability of the approach. The experiments show that the method is capable of producing high-quality segmentations across heterogeneous datasets while maintaining reasonable computational effort.

Several directions for future research emerge from this work. A natural extension is the development of formulations that jointly optimize both the segmentation and the parameters of the segment models, rather than relying on a preprocessing phase to enumerate feasible segments. Further improvements may be obtained by integrating advanced decomposition techniques, such as column generation or branch-and-price methods, to handle larger instances more efficiently. Finally, the design of specialized cutting planes and problem-specific heuristics could further enhance scalability and solution quality.

Author Contributions: All authors contributed equally to all phases of the work.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data and codes used in the analysis will be made public upon acceptance.

Acknowledgments: During the preparation of this manuscript/study, the authors used GenAI only for the purposes of syntax and style checking.

 doi:10.20944/preprints202603.1640.v1

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MIP Mixed Integer Programming
NP Nondeterministic Polynomial

References

1. Lovrić, M.; Milanović, M.; Stamenković, M. Algorithmic Methods for Segmentation of Time Series: An Overview. *Journal of Contemporary Economic and Business Issues* **2014**, *1*, 31–53.
2. Aminikhanghahi, S.; Cook, D.J. A Survey of Methods for Time Series Change Point Detection. *Knowledge and Information Systems* **2017**, *51*, 339–367. https://doi.org/10.1007/s10115-016-0987-z.
3. Truong, C.; Oudre, L.; Vayatis, N. Selective Review of Offline Change Point Detection Methods. *Signal Processing* **2020**, *167*, 107299. https://doi.org/10.1016/j.sigpro.2019.107299.
4. Keogh, E.; Chu, S.; Hart, D.; Pazzani, M. An online algorithm for segmenting time series. *IEEE International Conference on Data Mining* **2002**, pp. 289–296.
5. Auger, I.E.; Lawrence, C.E. Algorithms for the optimal identification of segment neighborhoods. *Bulletin of Mathematical Biology* **1989**, *51*, 39–54.
6. Jackson, B.; Scargle, J.D.; Barnes, D.; Arabhi, S.; Alt, A.; Gioumousis, P.; Gwin, E.; Sangtrakulcharoen, P.; Tan, L.; Tsai, T.T. An algorithm for optimal partitioning of data on an interval. *IEEE Signal Processing Letters* **2005**, *12*, 105–108.
7. Killick, R.; Fearnhead, P.; Eckley, I.A. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association* **2012**, *107*, 1590–1598.
8. Rigaill, G. Pruned dynamic programming for optimal multiple change-point detection. *arXiv preprint arXiv:1004.0887* **2010**.
9. Johnson, N.A. A dynamic programming algorithm for the fused lasso and L0-segmentation. *Journal of Computational and Graphical Statistics* **2013**, *22*, 246–260.
10. Hocking, T.D.; Rigaill, G.; Fearnhead, P.; Bourque, G. Learning sparse penalties for change-point detection using max-margin interval regression. *International Conference on Machine Learning (ICML)* **2013**.
11. Bellman, R. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM* **1961**, *4*, 284.
12. Picard, D.; Cook, R.D. Cross-validation of regression models. *Journal of the American Statistical Association* **1975**, *79*, 575–583.
13. Hudson, D.J. Fitting segmented curves whose join points have to be estimated. *Journal of the American Statistical Association* **1966**, *61*, 1097–1129.
14. Fisher, W.D. On grouping for maximum homogeneity. *Journal of the American Statistical Association* **1958**, *53*, 789–798.
15. Bertsimas, D.; King, A.; Mazumder, R. Best subset selection via a modern optimization lens. *Annals of Statistics* **2016**, *44*, 813–852.
16. Prokhorov, A.; Radchenko, P.; Semenov, A.; Skrobotov, A. Change-Point Detection in Time Series Using Mixed Integer Programming **2025**. [arXiv:econ.EM/2408.05665].
17. Balinski, M.L. *Integer Programming: Methods, Uses, Computation*; Academic Press, 1969.
18. Schrijver, A. *Combinatorial Optimization: Polyhedra and Efficiency*; Springer, 2003.
19. Fischetti, M.; Lodi, A.; Salvagnin, D. Just MIP It! In *Matheuristics*; Maniezzo, V.; Stützle, T.; Voß, S., Eds.; Springer, 2009; Vol. 10, *Annals of Information Systems*, pp. 1–20.
20. Maniezzo, V. Extended Set Covering for Time Series Segmentation. In Proceedings of the 15th Metaheuristics International Conference, MIC 2024; Saveaux, M., Ed. Springer, 2024, Vol. 14753, *Lecture Notes in Computer Science*, pp. 193.–199.
21. Maniezzo, V.; Boschetti, M.A.; Stützle, T. *Matheuristics: Algorithms and Implementations*; EURO Advanced Tutorials on Operations Research, Springer, 2021. https://doi.org/10.1007/978-3-030-64219-4.
22. Hocking, T.D.; Rigaill, G.; Fearnhead, P.; Bourque, G. A Log-Linear Time Algorithm for Constrained Changepoint Detection. *arXiv preprint arXiv:1703.03352* **2017**.
23. Yao, Y.C. Estimating the number of change-points via Schwarz' criterion. *Statistics & Probability Letters* **1988**, *6*, 181–189.

 doi:10.20944/preprints202603.1640.v1

24. Tibshirani, R.; Saunders, M.; Rosset, S.; Zhu, J.; Knight, K. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **2005**.
25. Hocking, T.D.; Rigaill, G.; Fearnhead, P.; Bourque, G. Generalized Functional Pruning Optimal Partitioning (GFPOP) for Constrained Changepoint Detection in Genomic Data. *Journal of Statistical Software* **2022**, *101*, 1–31.
26. Bai, J.; Perron, P. Estimating and Testing Linear Models with Multiple Structural Changes. *Econometrica* **1998**, *66*, 47–78.
27. Casini, A.; Perron, P. Structural Breaks in Time Series, 2019.
28. Tong, H.; Lim, K.S. Threshold autoregression, limit cycles, and cyclical data. *Journal of the Royal Statistical Society: Series B (Methodological)* **1980**, *42*, 245–292.
29. Rossi, F.; van Beek, P.; Walsh, T., Eds. *Handbook of Constraint Programming*; Elsevier, 2006.
30. Ruptures Documentation. https://centre-borelli.github.io/ruptures-docs/. Accessed 2025-12-01.
31. Fryzlewicz, P. Wild binary segmentation for multiple change-point detection. *Annals of Statistics* **2014**, *42*, 2243 – 2281.
32. Makridakis, S.; Hibon, M. Accuracy of forecasting: An empirical investigation. *Journal of the Royal Statistical Society, Series A* **1979**, *142*, 97–145.