Changepoint Detection via Subset Chains

Alexis Huet, Jose Manuel Navarro, and Dario Rossi

Huawei Technologies Co., Ltd., France
{first(.mid).last}@huawei.com

Abstract. Given a time series, the change point detection task consists in finding the instants where the statistical distribution of the series abruptly changes. The classic approach based on optimization techniques are too rigid and entangled, for which recent approaches advocate building a complete *solution path*, ranking all the possible change points of the series. This article extends this paradigm by providing a *chain of subsets*, corresponding to a hierarchy of changes, where different levels imply a finer detection granularity. Our proposal is to compute all levels from a single score vector through a recursive thresholding mechanism, where the threshold maps to the desired detection granularity.
We contrast our proposal against state-of-the-art approaches on public benchmarks with human expert labeling, showing: (i) best-in class performance (overall F1-score of 0.87), (ii) a statistically significant and remarkable improvement over the state of the art in the practical case where a single cost function and threshold setting is selected over multiple levels (F1-score of 0.76) and (iii) a qualitative alignment with different human experts for different levels, suggesting that each experts may find a different suitable level in practice.

Keywords: Change Point Detection · Multi-level Methods · Recursive Algorithms.

1 Introduction

Change point detection (CPD) focuses in detecting the changes in the distribution of a time series (such as those in mean, variance, or higher-order statistics), whose need arises in many domains including traffic analysis, finance, speech-analysis, and bio-informatics.

Formally, CPD consists in partitioning the indexes of a time series into intervals $I_1, \ldots, I_{K+1}$ (with K typically unknown) such that each interval of the partition conserves a property of interest (e.g., the homogeneity of a statistical moment), whereas consecutive intervals break it. This property of interest is usually measured by a cost function C (e.g., quadratic error loss for the first moment) mapping a sub-sequence x_I (the restriction of a time series x to an interval I) to a numeric value. Typical CPD formulations [29] take the form of a global optimization problem, trading-off the *total cost* $\sum_{k=0}^{K} C(x_{I_{k+1}})$ decreasing with further partitioning of the series vs a *penalty term* increasing with the number K of detected changes. However, as shown in the left of Fig. 1, the penalty term

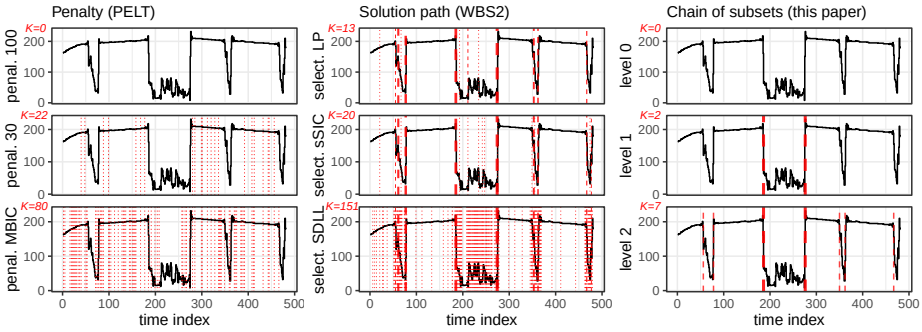


Fig. 1. Comparison of detection methods on the `scanline_42049` dataset [8], where K denotes the number of detected changes. (Left) *Penalty* methods treat all changes equally and are difficult to tune. (Middle) *Solution path* approaches capture relative change importance but lack conciseness. (Right) Our proposed *chain of subsets* method captures the relative importance between *groups* of changes, enhancing interpretability.

is hard to set, and small changes yield very different change points (i.e., a segmentation $S_{\beta'}$ for a lower penalty $\beta' < \beta$ is not guaranteed to verify $S_{\beta'} \supset S_\beta$), which undermines the practical usability of the methods. To counter that, *solution path* approaches have been introduced that maintain coherence among change points: at the same time, as the middle of Fig. 1 shows, the resulting set is still cluttered and of difficult interpretability for the human expert. In this paper, we introduce a new methodology to produce a *chain of subsets*, corresponding to multiple levels of changes, that is not only guaranteed to contain an *increasing sequence* of subsets but that also significantly eases interpretability as shown in the right of Fig. 1. Summarizing our main contributions:

- We introduce *scoring* and *recursive thresholding* mechanisms producing subset chains parametrized by (i) a cost function and (ii) a threshold. We show these provide decomposable (w.r.t. the levels) and flexible (w.r.t. the thresholds) results that align with human experts (each matching a different level).
- Our experimental campaign (over 1000 combinations from 18 algorithms in 3 families, across 42 datasets) shows our proposal outperforms the state of the art, even with a single cost function and threshold setting performed *over the whole public benchmark*.
- We open source an R package `scoth` (standing for scores and thresholds) in [21].

2 Related work

While research in change point detection in the last ten years has been driven and influenced by the statistics and signal processing communities, recent literature started to appear that approaches the field with a spirit closer to the

Table 1. Compact taxonomy of the approaches experimentally compared in this work

F	$\lvert\mathbf{F}\rvert^{\dagger}$	**[Ref.] Search**	**Par**	**(#)**‡	**C(·)**
Penalty	**425**	[20] AMOC	2	(60)	●
		[4] SEGNEIGH	2	(57)	●
		[16] Lasso	3	(90)	●
		[7] ADWIN	2	(18)	○
		[23] PELT	2	(60)	●
		[18] CPNP	2	(80)	○
		[12] RFPOP	2	(60)	●
Misc	**504**	[1] BOCPD	4	(480)	●
		[27] ECP	3	(9)	○
		[17] KCPA	1	(15)	○
Solution Path	**113**	[30] BS	2	(60)	●
		[13] WBS	2	(12)	●
		[15] WBS 2	2	(12)	●
		[14] TGUH	1	(4)	●
		[2] IDetect	1	(4)	●
		[6] NOT	2	(12)	○
		[26] ChangeForest	2	(9)	○
		[8] Naive	n.a.		○
		Chain of subset (this paper)	2	(48)	●

† *Number of overall experimental settings for each family* $\boldsymbol{F}$.
‡ *Number of parameters and of experimental settings for each approach.*
Cost function can be ● *parametric or* ○ *non-parametric.*

machine learning community. We briefly describe the existing families of approaches, namely *penalty*-based and *solution path*-based, as well as additional approaches (Bayesian, kernel-based and energy-based), which we directly experimentally compare with and that are summarized in Table 1.

Penalty-based approaches. The classical [28] approach for change point detection, still popular nowadays [29], sets the problem as a *global optimization* task, involving the *total cost* of a series regularized by a *penalty* function. In this family of approaches, the *penalty term* should be assigned before applying a search method: automatic selection occurs using criteria (AIC, BIC), or a fixed number of change points K can be defined beforehand (SEGNEIGH [4]). Once the cost function and penalty terms have been assigned, a *search method* is used for solving the optimization problem: the simplest search method is to select at most one change (AMOC [20]), but exact solutions can also be found efficiently (PELT [23]). Since the underlying model of real time series is usually unknown, approaches have been developed to improve the *robustness of the detection*: strategies are modifying the cost function (RFPOP [12], to cope with the presence of outliers, and Lasso [16]). Another possibility is to consider non-parametric approaches by relying on the empirical distribution function (ADWIN [7], CPNP [18]). As observed earlier, the *main drawback* of this family is that even slight changes of the penalty term yield completely different solutions.

Solution path approaches. Those methods decouple (i) the *ranking* of the potential change points from (ii) the *selection* of a subset of the top ranked elements. Compared to the penalty-based class, it allows the user to *postpone the selection* of the number of change points to a later stage and to quantify the relative importance between the changes. This class of approaches originates with Binary Segmentation (BS) [30], a divisive "greedy" method consisting in iteratively finding the next change point by maximizing a discrepancy function. As

this approach does not behave well with multiple changes, recent approaches focus on local intervals containing at most one change point (IDetect [2], NOT [6]), possibly relying on random subsets of intervals (WBS [13] , WBS2 [15]). A complementary agglomerative strategy consists in adopting a "generous" bottom-up approach as in TGUH [14]. Non-parametric counterparts also exist, with NOT [6] and ChangeForest [26]. Even if the selection step can be achieved in a similar fashion to the penalty-based approaches (with a criterion, e.g., sSIC [13], or on a grid), the strength of the methodology lies in possible refinements of the selection: Local Pruning (LP) [9] selects only a subset of the changes detected by the sSIC criteria based on additional local computations, while SDLL [15] employs an heuristic similar to the elbow method in clustering. The *main drawback* of these methods is to still output a *unique prediction set*, with no explicit hierarchy: as the methods result in possibly too numerous detected changes, practical usability is therefore undermined.

Miscellaneous approaches. Other families leverage probabilistic, kernel- or energy-based methods, which we also include in our comparison. Bayesian methods [1,11] compute the probability distribution of the time elapsed since the last change point and, due their probabilistic nature, provide additional confidence intervals regarding the position of the detected changes. The initial BOCPD [1] model was extended with model selection (BOCPDMS [24]) and robustness to outliers (RBOCPDMS [25]). Additional methods can be based on the divergence energy statistics (ECP [27]) or on the maximum kernel Fisher discriminant ratio (KCPA [17]). As reported in Table 1, we include representative examples of such methods, as well as naive method (i.e., zero detected change points) also used as benchmark in [8], in our experimental comparison.

3 Chains of subsets methodology

We formalize here our proposed method to yield a hierarchical chain of subsets (Sec. 3.1) by adopting a *scoring* step (Sec. 3.2) followed by a recursive *thresholding* step (Sec. 3.3), that we also illustrate with an example (Sec. 3.4).

3.1 Problem formulation

Notation. We adapt the standard notation of existing literature [29] and extend it to our context wherever necessary. We consider a series $x = (x_1, \dots, x_T)$ of length T, that can be univariate or multivariate. A segmentation $S = (b_1, \dots, b_K)$ of the series is a sequence of integers verifying $1 < b_1 < b_2 < ... < b_K < T + 1$. This segmentation contains $K \in [\![0, T-1]\!]$ change points and, by defining the dummy indexes $b_0 := 1$ and $b_{K+1} := T + 1$, it is associated to the partition $[1, T+1) = \bigcup_{k=0}^{K} [b_k, b_{k+1})$ consisting of $K+1$ intervals $I_1, \dots, I_{K+1}$. Following the notations in [13], any such interval $[b_k, b_{k+1})$ may also be denoted as $[s, e)$ with start- and end-points s and e.

A *cost function* $C(\cdot)$ quantifies as a numeric value a property of interest (e.g., mean) of a consecutive sub-sequence $x_s, \dots, x_{e-1}$, and is noted $C(x_{[s,e)})$. For instance, the mean can be measured by the quadratic error loss, which corresponds to $C_{\mathrm{L2}}(x_{[s,e)}) := \sum_{i=s}^{e-1} \|x_i - \bar{x}\|_2^2$, with $\bar{x}$ the (univariate or multivariate) mean of the sub-sequence. The *total cost* $V(\cdot)$ serves as an internal measure of the performance of a segmentation S, and is defined, for a series x and a cost C, by:

$$V(S; x, C) := \sum_{k=0}^{K} C(x_{[b_k, b_{k+1})}). \tag{1}$$

The total cost is written $V(S)$ when the series and the cost function are fixed. In particular, $V(\varnothing)$ is the initial cost $C(x_{[1,T+1)})$.

Global vs local optimization and discrepancy. Change point detection on series x aims to identify a segmentation S where each interval maintains the property of interest (measured by cost C), while consecutive intervals break it. For a specified number of change points K, this becomes an optimization problem: finding a K-length segmentation that minimizes total cost V. However, K is generally unknown and V serves only as a guide: rather than the *global optimization* using a penalty term (as seen in Sec. 2), we adopt a *local optimization* approach based on discrepancy. The *discrepancy function* $D(\cdot)$ is a convenient way to locally assess the property of interest through the cost $C(\cdot)$. It is defined for any cut $b \in [\![s+1, e-1]\!]$ of a subset $[s, e)$ by:

$$D(b; x_{[s,e)}, C) := C(x_{[s,e)}) - \big(C(x_{[s,b)}) + C(x_{[b,e)})\big). \tag{2}$$

Discrepancy is a good measure for the intervals $[s, e)$ that are sufficiently local to contain at most one change point, but may fail for intervals that contain more changes (cf illustration in Fig. 2(a) in Sec. 3.4). When adding a cut b to a segmentation $S = (b_1, \dots, b_K)$, the discrepancy then naturally appears when computing $V(S) - V(S \cup b)$ thanks to the additive form of the total cost $V(\cdot)$. We define in that context the *gain* $G(\cdot)$, for any $b_k \in S$, and note that since the total cost V has an additive form, when computing $V(S \setminus b_k) - V(S)$ (where $S \setminus b_k$ represents the set S with element b_k removed), all the terms that do not involve b_k cancel out:

$$G(b_k, S) := V(S \setminus b_k) - V(S). \tag{3}$$

$$G(b_k, S) = D(b_k; x_{[b_{k-1}, b_{k+1})}, C). \tag{4}$$

Solution path vs chain of subsets. A complete *solution path* [15] $\mathcal{P}$ is a nested sequence of segmentations $\varnothing =: S_0 \subsetneq S_1 \subsetneq \dots \subsetneq S_{T-1} := [\![2, T]\!]$ that verifies: $|S_k| = k$ for $k \in [\![0, T-1]\!]$. In particular, S_{k+1} only differs from S_k by one element. We further introduce the notion of *chain of subsets*, defined as a nested sequence of segmentations: $\varnothing =: L_0 \subset L_1 \subset L_2 \subset \dots$ Compared to a solution path, the condition on the cardinal is discarded, so that each step $L_{\mathrm{level}} \subset [\![2, T]\!]$ can differ by one or more elements from the previous level $L_{\mathrm{level}-1}$. A chain of subsets can be a solution to the CPD problem that is both *recursive*, as the next levels contain the former, and *hierarchical*, as changes of less important magnitude can be stratified in increasing levels.

Algorithm 1 Scoring through maximum normalized gains vector

1: **Inputs** $x = (x_1, \dots x_T)$ series; C cost function.
2: $S \leftarrow [\![2, T]\!]$ ▷ initial segmentation, as a fully segmented set
3: score $\leftarrow (0)_{[\![2,T]\!]}$ ▷ initial scores, vector of zeros of length $T-1$
4: while $S \neq \varnothing$:
5: for $b \in S$:
6: score$(b) \leftarrow \max(\text{score}(b), G(b, S))$ ▷ update scores
7: $S \leftarrow S \setminus \text{argmin}_{b \in S}\ \text{score}(b)$ ▷ remove the worst cut
8: **Outputs** score$/V(\varnothing)$ ▷ returns normalized scores $\in [0, 1]$.

3.2 Scores

Scoring algorithm. Given a cost function C and a series x, a scoring function $b \mapsto \text{score}(b)$ maps from $[\![2, T]\!]$ to $[0, 1]$. Intuitively, indexes b that are clear change points for the current cost function (i.e. with large local gain) should have large scores, and vice versa. We propose to build the segmentation path along with the scoring of the indexes. The iterative procedure introduced in Algorithm 1 follows a bottom-up approach [22], with the additional storage (at line 6) and usage (at line 7) of the *maximum* normalized gain observed along the constructed path. The use of the maximum in line 6 has a theoretical foundation that ensures two properties linking the output scores and the corresponding solution path, whose proof is omitted due to space constraints, but that is exemplified in Sec. 3.4: (i) the scores are always decreasing w.r.t. the solution path, and (ii) given an existing path produced by the scores, it is possible to reconstruct the exact same scores by simply following the path. In a nutshell, the algorithm calculates for all possible points their score (the normalized gain associated with segmenting the series in that point), updates it if it is higher than their previously assigned one (line 6) and removes from the pool the point with the smallest score (line 7). The process is repeated until all points are removed from the pool (line 4).

Complexity analysis. The subtle part in the algorithm implementation is the update of the score function at line 6, since the naive implementation of computing the gain for all b at each step of the iteration would lead to a quadratic complexity (w.r.t. the number of calls of the cost function). Instead, we first initialize two vectors A and B for tracking (a) the cost of each interval and (b) the cost of each consecutive pair of intervals induced by the segmentation S. At each iteration, the removed cut induces a modification of A that has been already computed by B, and a modification of B that necessitates at most two calls of the cost function. Overall, the process is linear both w.r.t. the number of calls of the cost function and w.r.t. memory.

3.3 Thresholds

Basic thresholding. The scores vector computed in Sec. 3.2 and normalized to $[0, 1]$ offers a convenient setting for thresholding: the most basic mechanism

Algorithm 2 Recursive thresholdings

1: **Inputs** threshold $\in [0, 1]$; score vector; x series; C cost
2: $L \leftarrow []$ ▷ placeholder for the changes at each level
3: $L[0] \leftarrow \varnothing$ ▷ level 0
4: level $\leftarrow 1$
5: $L[\text{level}] \leftarrow \{b \ ; \ \text{score}(b) \geq \text{threshold}\}$ ▷ level 1 changes
6: while $L[\text{level}] \neq L[\text{level} - 1]$:
7: $\quad S \leftarrow L[\text{level}]$ ▷ all the changes marked up to now
8: $\quad L[\text{level} + 1] \leftarrow L[\text{level}] \cup \{b \notin S; \text{score}(b) \frac{V(\varnothing)}{V(S)} \geq \text{threshold}\}$
9: $\quad$ level $\leftarrow$ level $+ 1$
10: **Outputs** L ▷ list of changes at each level.

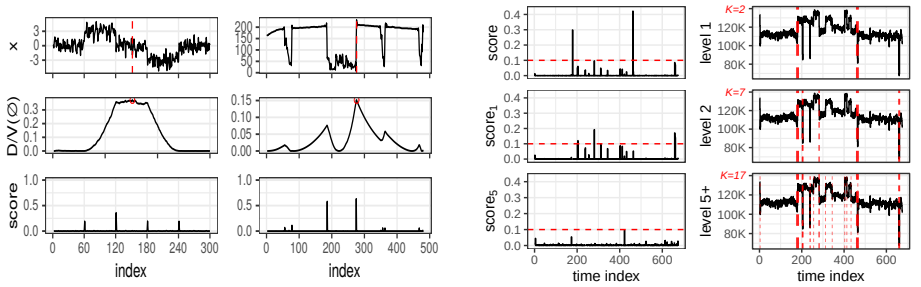
consists of setting a single value within $[0, 1]$ as threshold and to detect as change points all the indexes that are greater than this value. This threshold has a direct interpretation in terms of gain: a change point is selected whenever there exists a step (in the solution path) that offers a normalized gain (i.e. the difference of total costs with and without this index, normalized by the initial cost) that is greater than this threshold. Note that the threshold can be tuned without additional computational cost, as the scores are computed once: this is in contrast to penalty-based strategies that requires re-running the optimization problem at any threshold change.

Recursive thresholding algorithm. Basic thresholding only returns a single level of change points: we introduce in Algorithm 2 a recursive mechanism that allows the production of chain of subsets $L_0 \subset L_1 \subset L_2 \subset \ldots$, where each level corresponding to a finer granularity of the segmentation. Notice that this formulation does not add any further hyperparameter since, having fixed the cost function $C(\cdot)$, the inputs to the algorithm are the series x, *the scores vector* produced by Algorithm 1, and *a single threshold value in* $[0, 1]$, that can be interpreted as the minimum normalized gain for a change point to be detected.

The initial steps populate level 0 with the absence of changes and level 1 with the basic thresholding result. The next levels are computed recursively by considering the segmentation at the current level, noted S, and by measuring the total cost $V(S)$ relative to the initial cost $V(\varnothing)$. Note that the multiplicative coefficient $V(\varnothing)/V(S)$ is larger than one, and acts as a constant "zooming" factor over all the intervals corresponding to the segmentation S, which facilitates new, smaller scores surpassing the threshold. The additional detected changes are added to the next level and the recursion is applied until the convergence of the segmentation (i.e. no new change point is detected in line 8).

3.4 Illustration of scoring and thresholding steps

Scoring. Fig. 2(a) contrasts the time series (top), the normalized discrepancies $D(b; x, C_{L2})/V(\varnothing)$ (middle) with maximum value highlighted, and scores obtained using Algorithm 1 (bottom), on a synthetic (normally distributed series



(a) Scores vs discrepancies (b) Recursive thresholding at three levels

Fig. 2. Illustration of scores vs discrepancies (a) and recursive thresholding (b). In (a), the normalized discrepancy function is computed on two series, with red points and lines indicating the maximum discrepancy index, and contrasted with our score proposal. In (b), the horizontal dashed lines in the score series are the selected threshold of 0.1. The vertical dashed lines are the segmentation indexes at each level, with bold dashes for lower levels and lighter dashes for higher levels. K is the number of detected changes.

with four changes, top left) and a real series (`scanline_42049`, top right). These cases are interesting because they contain multiple changes of the same magnitude: direct maximization of the discrepancy may lead to inappropriate change point (middle left) or underestimated scores (middle right). Conversely, scorings obtained with Algorithm 1 are still grouped into peaks of same magnitude (bottom).

Recursive thresholding. Fig. 2(b) illustrates the recursion, with a threshold of 0.1 and the C_{L2} cost, for the classic well-log data [31], depicting the two first levels (top and middle), and convergence after five levels (bottom). The left plots depict the single scoring vector containing all the information about the potential change points, which are recursively augmented at each step (the predicted number of change points K is indicated at each level). After the computation of the scores vector using C_{L2} (top-left), the indexes over the threshold of 0.1 are extracted, giving a segmentation S_1 (top-right) from which the coefficient factor $V(\varnothing)/V(S_1) = 1.99$ is computed. This coefficient updates the scores vector using $\text{score}_1 = 1.99 \times \text{score}$ (middle-left), resulting in a new segmentation (middle-right). The process converges after five steps (bottom-right), since no element of $\text{score}_5 = 9.98 \times \text{score}$ exceeds the selected threshold (bottom-left).

4 Evaluation

4.1 Experimental settings

Algorithms and search grid. We evaluate our methodology against classical and recent algorithms coming from penalty, solution path, and other change

point detection approaches, as summarized in Table 1. Our methodology uses as cost functions quadratic [28] and linear [5] losses. The linear loss cost function fits a piecewise linear regression to each signal segment, with the cost defined as the sum of squared residuals. We explore thresholds from 0.03 to 1 across 24 settings.

Datasets and annotations. We evaluate the algorithms on the publicly available benchmark [8] of 42 datasets, comprising univariate and multivariate series from diverse sources (environmental, financial, etc.). The main factor considered for the inclusion of a dataset in our selection is the presence of a change point, understood as in the present article as an abrupt change in the statistical distribution of the series. Nonetheless, the datasets are real series and, thus, often show additional seasonal patterns and outliers. Each series have been labeled by 5 different human annotators randomly selected from a pool of 8. These annotators have been trained on multiple quality control series beforehand for learning the difference between a change point and an outlier. Additional details about each dataset and the annotation collection process are available in [8].

Performance metrics. The main performance indicator used for evaluating the performance is the F1-score computed with a margin of error of $M = 5$ [29]. This metric is computed for each dataset on the *median annotator*, defined as the user reaching the maximum average Jaccard index against all other users (formally, with $S_1, \dots S_5$ the annotations given by the 5 annotators, we select the user $\arg\max_i \sum_{j \neq i} \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$). To facilitate the comparison with prior art, we also compute the alternative F1-score [8] biased by adding a trivial change point that is always detected and named $\mathrm{F1}_{\mathrm{biased}}$ here, and the average covering metric *cover* computed from the Jaccard indexes [3, 8].

Evaluation settings. We group the different algorithms into families that correspond to the different approaches for the CPD task, as summarized in the first column of Table 1. We evaluate the algorithms *quantitatively* in two different settings: (i) the *oracle setting*, in which we report the performance obtained by the best possible method and parameter values per dataset within each family, and (ii) the *best-single setting*, corresponding to the performance of the best method and parameter values across all datasets within each family, measured in terms of average performance. To check the statistical significance of the results, we perform for each setting a Friedman test ($p < 0.05$ for both settings) followed by Nemenyi post-hoc tests on the ranks extracted from the performance values [10, 19]. Finally, we evaluate (iii) the *qualitative* alignment between the results of our methodology with the annotations made by the human annotators for three datasets of the benchmark.

Table 2. Full ablation study of (a) oracle settings and (b) best single settings across the four families of approaches on 42 public datasets (boldface for best and *italic* for second-best approach). Results are displayed as average ± standard deviation.

Family(#)†	**Oracle setting**			**Best-single setting**	
	F1	$F1_{biased}$	cover	F1	selected parameters
Ours (48)	**0.87±0.20**	**0.92±0.09**	**0.82±0.11**	**0.76±0.27**	linear cost; threshold=0.1
Misc (504)	*0.84±0.23*	*0.91±0.11*	*0.75±0.15*	*0.49±0.38*	OCP; λ=100; a=10; b=k=1
Penalty (425)	0.81±0.23	0.90±0.12	0.72±0.18	0.40±0.38	RFPOP; Huber cost; penalty=10
Solution Path (113)	0.77±0.26	0.87±0.12	0.73±0.18	0.43±0.35	NOT with M=10; quadratic cost; IC
Zero	0.17±0.38	0.65±0.20	0.56±0.21	0.17±0.38	none

† *Number of overall experimental settings for each family.*

(a) Oracle setting (b) Best-single setting

Fig. 3. Average rank comparison on F1 score between oracle (left) and best-single (right) setting of each family, with critical distance (CD) with 95% confidence interval.

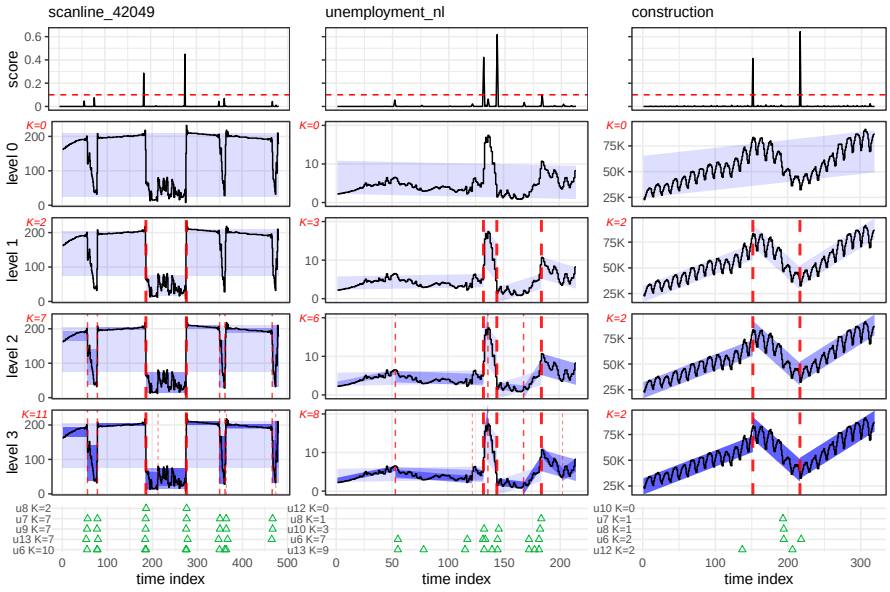


Fig. 4. Qualitative results in applying our recursive methodology using the best-single setting parameters on three datasets from [8] (top) and change points as labeled by five different users (bottom). The dashed line in the score series is the selected threshold. The vertical dashed lines are the segmentation indexes at each level, with bold dashes for lower levels and lighter dashes for higher levels. The shaded bounding boxes represent the mass between the 5th and 95th percentile between consecutive change points within the same level. The numbers in front of the rows in bottom parts are the user id and the number of labeled changes. Details about annotation collection process in [8].

4.2 Experimental results

Oracle setting results. The left parts of Table 2 and Fig. 3 display the oracle setting results. Our proposal has the highest average performance across the three evaluation metrics, with relatively large standard deviations over the different datasets. In terms of significance, our proposal attains the best average rank of all method families (with an average rank close to 2), while being within the critical distance compared to the state of the art. We also observe that the three different evaluation metrics are correlated, with a bias for $F1_{\text{biased}}$ and cover towards 1 due to the added trivial change point.

Best-single setting results. The right parts of Table 2 and of Fig. 3 report results for the best-single setting, for which we explicit the parameters for the selected method. Compared to the oracle setting, we observe an expected decrease in the F1-score, either large (from -0.34 to -0.41 for the 3 state-of-the-art families of approaches) or limited (-0.11 for our methodology). The performance gap between our proposal and other methods has widened, with rank analysis confirming its superiority over the state of the art at 5% confidence level. This larger difference suggests increased robustness to hyper-parameter choice. Computationally, all methods except ECP run in less than one second on an Intel i7 1.90 GHz processor.

Qualitative alignment with different users for different levels. We depict in Fig. 4 the scores and the change points outputs obtained at each level on three time series, using our method with the best-single setting parameters (top), and completed by the annotation for the change points as labeled by five different users, ranked by the number of labeled changes (bottom).

Positive cases. Human annotators show disagreement in segmentation granularity, which our methodology effectively captures. In the `scanline_42049` series, user 8's two changes match our algorithm's level 1 solution, while these changes are subset of seven labels by users 7, 9, and 13 – corresponding to our level 2 solution. These seven labels are themselves subset of user 6's ten labels (except for the last label after 450), aligning with our level 3 solution (e.g., multiple close changes around $t = 350$). Similarly, in `unemployement_nl`, users 6 and 13 mark transition periods as separate changes (e.g., rise and fall near $t = 140$), unlike user 10.

Negative case. Despite outperforming the state of the art, our algorithm has limitations, as illustrated by the `construction` series with its seasonal patterns and long-term piecewise trends. Here, where human labeler agreement is low, our methodology identifies cyclic patterns as change points from level 1, contrary to some users' annotations. Note that our method appropriately does not segment the series into seasonal yearly components unrelated to change points.

5 Discussion

This paper presents a recursive score- and threshold-based methodology for change point detection, with strengths and limitations discussed below.

Strengths. Unlike existing approaches, our proposal constructs a unique solution path where recursive levels correspond to different change detection granularities. Experiments demonstrate both quantitative and qualitative improvements over the state of the art. The recursive *levels of granularity* elegantly accommodate varying human preferences through consistent, incremental refinement with minimal cognitive effort. The method requires only a few intuitive parameters (threshold and cost function), enhancing its practical appeal.

Limitations. Several directions exist to improve algorithms in the new 'chain of subsets' family. The first limitation concerns outliers appearing as paired change points, addressable via refined cost functions as in RFPOP [12]. The second limitation is the offline nature of our method, though our scoring's independence enables future adaptation to online settings. Future work also includes refining threshold tuning guidance. While we currently recommend 0.1 as an effective default in the best-single setting, the thresholding that induces multi-level change points can be tuned independently from scoring computations, offering a valuable opportunity for targeted investigation of threshold mechanisms.

References

1. Adams, R.P., MacKay, D.J.: Bayesian online changepoint detection. arXiv (2007)
2. Anastasiou, A., Fryzlewicz, P.: Detecting multiple generalized change-points by isolating single ones. Metrika **85**(2), 141–174 (2022)
3. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE TPAMI **33**(5), 898–916 (2010)
4. Auger, I., Lawrence, C.E.: Algorithms for the optimal identification of segment neighborhoods. Bulletin of Mathematical Biology (1989)
5. Bai, J.: Least squares estimation of a shift in linear processes. Journal of Time Series Analysis **15**(5), 453–472 (1994)
6. Baranowski, R., Chen, Y., Fryzlewicz, P.: Narrowest-over-threshold detection of multiple change points and change-point-like features. Journal of the Royal Statistical Society Series B: Statistical Methodology **81**(3), 649–672 (2019)
7. Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM ICDM. pp. 443–448. SIAM (2007)
8. Van den Burg, G.J., Williams, C.K.: An evaluation of change point detection algorithms. arXiv preprint arXiv:2003.06222 (2020)
9. Cho, H., Kirch, C.: Two-stage data segmentation permitting multiscale change points, heavy tails and dependence. AISM pp. 1–32 (2022)
10. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. The Journal of Machine learning research **7**, 1–30 (2006)
11. Fearnhead, P., Liu, Z.: On-line inference for multiple changepoint problems. Journal of the Royal Statistical Society Series B: Statistical Methodology **69**(4), 589–605 (2007)

12. Fearnhead, P., Rigaill, G.: Changepoint detection in the presence of outliers. JASA **114**(525), 169–183 (2019)
13. Fryzlewicz, P.: Wild binary segmentation for multiple change-point detection. The Annals of Statistics **42**(6), 2243–2281 (2014)
14. Fryzlewicz, P.: Tail-greedy bottom-up data decompositions and fast mulitple change-point detection. Annals of Statistics **46**(6B), 3390–3421 (2018)
15. Fryzlewicz, P.: Detecting possibly frequent change-points: Wild binary segmentation 2 and steepest-drop model selection. Journal of the Korean Statistical Society **49**(4), 1027–1070 (2020)
16. Harchaoui, Z., Lévy-Leduc, C.: Catching change-points with lasso. In: NIPS. vol. 617, p. 624 (2007)
17. Harchaoui, Z., Moulines, E., Bach, F.: Kernel change-point analysis. NeurIPS **21** (2008)
18. Haynes, K., Fearnhead, P., Eckley, I.A.: A computationally efficient nonparametric approach for changepoint detection. Statistics and computing **27**, 1293–1305 (2017)
19. Herbold, S.: Autorank: A python package for automated ranking of classifiers. Journal of Open Source Software **5**(48), 2173 (2020)
20. Hinkley, D.V.: Inference about the change-point in a sequence of random variables. Biometrika (1970)
21. Huet, A., Navarro, J.M., Rossi, D.: scoth: Changepoint detection via subset chains. `https://github.com/ahstat/scoth-segmentation` (2025)
22. Keogh, E., Chu, S., Hart, D., Pazzani, M.: An online algorithm for segmenting time series. In: Proceedings 2001 IEEE ICDM. pp. 289–296. IEEE (2001)
23. Killick, R., Fearnhead, P., Eckley, I.A.: Optimal detection of changepoints with a linear computational cost. JASA **107**(500), 1590–1598 (2012)
24. Knoblauch, J., Damoulas, T.: Spatio-temporal bayesian on-line changepoint detection with model selection. In: ICML. pp. 2718–2727. PMLR (2018)
25. Knoblauch, J., Jewson, J.E., Damoulas, T.: Doubly robust bayesian inference for non-stationary streaming data with β-divergences. NeurIPS **31** (2018)
26. Londschien, M., Bühlmann, P., Kovács, S.: Random forests for change point detection. Journal of Machine Learning Research **24**(216), 1–45 (2023)
27. Matteson, D.S., James, N.A.: A nonparametric approach for multiple change point analysis of multivariate data. JASA **109**(505), 334–345 (2014)
28. Page, E.: A test for a change in a parameter occurring at an unknown point. Biometrika **42**(3/4), 523–527 (1955)
29. Truong, C., Oudre, L., Vayatis, N.: Selective review of offline change point detection methods. Signal Processing **167**, 107299 (2020)
30. Vostrikova, L.Y.: Detecting "disorder" in multidimensional random processes. In: Doklady akademii nauk. vol. 259, pp. 270–274. Russian Academy of Sciences (1981)
31. Ó Ruanaidh, J.J., Fitzgerald, W.J.: Numerical Bayesian methods applied to signal processing. Springer Science & Business Media (1996)