Geometric-Based Pruning Rules For Change Point Detection in Multiple Independent Time Series

Liudmila Pishchagina, Guillem Rigaill, Vincent Runge

Geometric-Based Pruning Rules For Change Point Detection in Multiple Independent Time Series.

Liudmila Pishchagina[a], Guillem Rigaill[a,b], Vincent Runge[a]

[a] *Université Paris-Saclay, CNRS, Univ Evry, Laboratoire de Mathématiques et Modélisation d'Evry, 91037, Evry-Courcouronnes, France.*
[b] *Université Paris-Saclay, CNRS, INRAE, Univ Evry, Institute of Plant Sciences Paris-Saclay (IPS2), Orsay, France.*

Abstract

We consider the problem of detecting multiple changes in multiple independent time series. The search for the best segmentation can be expressed as a minimization problem over a given cost function. We focus on dynamic programming algorithms that solve this problem exactly. When the number of changes is proportional to data length, an inequality-based pruning rule encoded in the PELT algorithm leads to a linear time complexity. Another type of pruning, called functional pruning, gives a close-to-linear time complexity whatever the number of changes, but only for the analysis of univariate time series.

We propose a few extensions of functional pruning for multiple independent time series based on the use of simple geometric shapes (balls and hyperrectangles). We focus on the Gaussian case, but some of our rules can be easily extended to the exponential family. In a simulation study we compare the computational efficiency of different geometric-based pruning rules. We show that for small dimensions (2, 3, 4) some of them ran significantly faster than inequality-based approaches in particular when the underlying number of changes is small compared to the data length.

Keywords: Multivariate time series, multiple change point detection, dynamic programming, functional pruning, computational geometry

Introduction

A National Research Council report [1] has identified change point detection as one of the "inferential giants" in massive data analysis. Detecting change points, either a posteriori or online, is important in areas as diverse as bioinformatics [2, 3], econometrics [4, 5], medicine [6, 7, 8], climate and oceanography [9, 10, 11, 12], finance [13, 14], autonomous driving [15], entertainment [16, 17, 18], computer vision [19] or neuroscience [20]. The most common and prototypical change point detection problem is that of detecting changes in mean of a univariate Gaussian signal and a large number of approaches have been proposed to perform this task (see among many others [21, 22, 23, 24, 25] and the reviews [26, 27]).

Penalized cost methods. Some of these methods optimize a penalized cost function (see for example [22, 28, 29, 11, 30, 31]). These methods have good statistical guarantees [21, 32, 22] and have shown good performances in benchmark simulations [33] and on many applications [34, 35]. From a computational perspective, they rely on dynamic programming algorithms that are at worst quadratic in the size of the data, n. However using inequality-based and functional pruning techniques [30, 11, 31] the average run times are typically much smaller allowing to process very large profiles ($n > 10^5$) in a matter of seconds or minutes. In detail, for one time series:

- if the number of change points is proportional to n both PELT (inequality-based pruning) and FPOP (functional pruning) [11, 31] are on average linear.
- if the number of change points is fixed, FPOP is quasi-linear (on simulations) while PELT is quadratic [31].

Multivariate extensions. In this paper we focus on the multivariate problem assuming the cost function or log-likelihood of a segment (denoted $\mathcal{C}$) can be

decomposed as a sum over all p dimensions. Informally that is

$$\mathcal{C}(segment) = \sum_{k=1}^{p} \mathcal{C}(segment,\ \text{time series } k)\,.$$

In this context, the PELT algorithm can easily be extended for multiple time series. However, as for the univariate case, it will be algorithmically efficient only if the number of change points non-neglectible compare to n. In this paper, we study the extension of functional pruning techniques (and more specifically FPOP) to the multivariate case.

At each iteration, FPOP updates the set of parameter values for which a change position τ is optimal. As soon as this set is empty the change is pruned. For univariate time series, this set is a union of intervals in $\mathbb{R}$. For multi-parametric models, this set is equal to the intersection and difference of convex sets in $\mathbb{R}^p$ [36]. It is typically non-convex, hard to update, and deciding whether it is empty or not is not straightforward.

In this work, we present a new algorithm, called Geometric Functional Pruning Optimal Partitioning (GeomFPOP). The idea of our method consists in approximating the sets that are updated at each iteration of FPOP using simpler geometric shapes. Their simplicity of description and simple updating allow for a quick emptiness test.

The paper has the following structure. In Section 1 we introduce the penalized optimization problem for segmented multivariate time series. We then review the existing pruned dynamic programming methods for solving this problem. We define the geometric problem that occurs when using functional pruning. The new method, called GeomFPOP, is described in Section 2 and based on approximating intersection and exclusion set operators. In Section 3 we introduce two approximation types (sphere-like and rectangle-like) and define the approximation operators for each of them. We then compare in Section 4 the empirical efficiency of GeomFPOP with PELT on simulated data.

1. Functional Pruning for Multiple Time Series

1.1. Model and Cost

We consider the problem of change point detection in multiple time series of length n and dimension p. Our aim is to partition time into segments, such that in each segment the parameter associated to each time series is constant. For a time series y we write $y = y_{1:n} = (y_1, \ldots, y_n) \in (\mathbb{R}^p)^n$ with y_i^k the k-th component of the p-dimensional point $y_i \in \mathbb{R}^p$ in position i in vector $y_{1:n}$. We also use the notation $y_{i:j} = (y_i, \ldots, y_j)$ to denote points from index i to j. If we assume that there are M change points in a time series, this corresponds to time series splits into $M+1$ distinct segments. Each segment $m \in \{1, \ldots, M+1\}$ is generated by independent random variables from a multivariate distribution with the segment-specific parameter $\theta_m = (\theta_m^1, \ldots, \theta_m^p) \in \mathbb{R}^p$. A segmentation with M change points is defined by the vector of integers $\tau = (\tau_0 = 0, \tau_1, \ldots, \tau_M, \tau_{M+1} = n)$. Segments are given by the sets of indices $\{\tau_i + 1, \ldots, \tau_{i+1}\}$ with i in $\{0, 1, \ldots, M\}$.

We define the set S_t of all possible change point locations related to the segmentation of data points between positions 1 to t as

$$S_t = \{\tau = (\tau_0, \tau_1, \ldots, \tau_M, \tau_{M+1}) \in \mathbb{N}^{M+2} | 0 = \tau_0 < \tau_1 < \cdots < \tau_M < \tau_{M+1} = t\}.$$

Usually the number of changes M is unknown, and has to be estimated. Many approaches to detecting change points define a cost function for segmentation using the opposite log-likelihood (times two). Here the opposite log-likelihood (times two) linked to data point y_j is given by function $\theta \mapsto \Omega(\theta, y_j)$, where $\theta = (\theta^1, \ldots, \theta^p) \in \mathbb{R}^p$. Over a segment from i to t, the parameter remains the same and the segment cost $\mathcal{C}$ is given by

$$\mathcal{C}(y_{i:t}) = \min_{\theta \in \mathbb{R}^p} \sum_{j=i}^{t} \Omega(\theta, y_j) = \min_{\theta \in \mathbb{R}^p} \sum_{j=i}^{t} \left(\sum_{k=1}^{p} \omega(\theta^k, y_j^k) \right), \tag{1.1}$$

with ω the atomic likelihood function associated with Ω for each univariate time series. This decomposition is made possible by the independence hypothesis between dimensions. Notice that function ω could have been dimension-dependent

with a mixture of different distributions (Gauss, Poisson, negative binomial, etc.). In our study, we use the same data model for all dimensions.

We consider a penalized version of the cost by a penalty $\beta > 0$, as the zero penalty case would lead to segmentation with n segments. Summing over all segments we end up with a penalty that is linear in the number of segments. Such choice is common in the literature ([37],[11]) although some other penalties have been proposed ([38],[22],[39]). The optimal penalized cost associated with our segmentation problem is then defined by

$$Q_n = \min_{\tau \in S_n} \sum_{i=0}^{M} \{\mathcal{C}(y_{(\tau_i+1):\tau_{i+1}}) + \beta\} . \tag{1.2}$$

The optimal segmentation τ is obtained by the argminimum in Equation (1.2).

1.2. Functional Pruning Dynamic Programming Algorithm

The idea of the Optimal Partitioning (OP) method [29] is to search for the last change point defining the last segment in data $y_{1:t}$ at each iteration (with $Q_0 = 0$), which leads to the recursion:

$$Q_t = \min_{i \in \{0,\dots,t-1\}} \Big(Q_i + \mathcal{C}(y_{(i+1):t}) + \beta \Big) .$$

Functional description. In the FPOP method we introduce a last segment parameter $\theta = (\theta^1, \dots, \theta^p)$ in $\mathbb{R}^p$ and define a functional cost $\theta \mapsto Q_t(\theta)$ depending on θ, that takes the following form:

$$Q_t(\theta) = \min_{\tau \in S_t} \Big(\sum_{i=0}^{M-1} \{\mathcal{C}(y_{(\tau_i+1):\tau_{i+1}}) + \beta\} + \sum_{j=\tau_M+1}^{t} \Omega(\theta, y_j) + \beta \Big) .$$

As explained in [31], we can compute the function $Q_{t+1}(\cdot)$ based only on the knowledge of $Q_t(\cdot)$ as for each integer t from 0 to $n-1$. We have:

$$Q_{t+1}(\theta) = \min\{Q_t(\theta), m_t + \beta\} + \Omega(\theta, y_{t+1}) , \tag{1.3}$$

for all $\theta \in \mathbb{R}^p$, with $m_t = \min_\theta Q_t(\theta)$ and the initialization $Q_0(\theta) = 0$, so that $Q_1(\theta) = \Omega(\theta, y_1)$. By looking closely at this relation, we see that each function

Q_t is a piece-wise continuous function consisting of at most t different functions on $\mathbb{R}^p$, denoted q_t^i:

$$Q_t(\theta) = \min_{i \in \{1,\dots,t\}} \left\{ q_t^i(\theta) \right\} ,$$

where the q_t^i functions are given by explicit formulas:

$$q_t^i(\theta) = m_{i-1} + \beta + \sum_{j=i}^{t} \Omega(\theta, y_j) , \quad \theta \in \mathbb{R}^p , \quad i = 1, \dots, t.$$

and

$$m_{i-1} = \min_{\theta \in \mathbb{R}^p} Q_{i-1}(\theta) = \min_{j \in \{1,\dots,i-1\}} \left\{ \min_{\theta \in \mathbb{R}^p} q_{i-1}^j(\theta) \right\}. \tag{1.4}$$

It is important to notice that each q_t^i function is associated with the last change point $i-1$ and the last segment is given by indices from i to t. Consequently, the last change point at step t in $y_{1:t}$ is denoted as $\hat{\tau}_t$ ($\hat{\tau}_t \leq t-1$) and is given by

$$\hat{\tau}_t = \operatorname*{Arg\,min}_{i \in \{1,\dots,t\}} \left\{ \min_{\theta \in \mathbb{R}^p} q_t^i(\theta) \right\} - 1.$$

Backtracking. Knowing the values of $\hat{\tau}_t$ for all $t = 1, \dots, n$, we can always restore the optimal segmentation at time n for $y_{1:n}$. This procedure is called backtracking. The vector $cp(n)$ of ordered change points in the optimal segmentation of $y_{1:n}$ is determined recursively by the relation $cp(n) = (cp(\hat{\tau}_n), \hat{\tau}_n)$ with stopping rule $cp(0) = \emptyset$.

Parameter space description. Applying functional pruning requires a precise analysis of the recursion (1.3) that depends on the property of the cost function Ω. In what follows we consider three choices based on a Gaussian, Poisson, and negative binomial distribution for data generation. The exact formulas of these cost functions are given in Appendix A.

We denote the set of parameter values for which the function $q_t^i(\cdot)$ is optimal as:

$$Z_t^i = \left\{ \theta \in \mathbb{R}^p | Q_t(\theta) = q_t^i(\theta) \right\}, \quad i = 1, \dots, t.$$

The key idea behind functional pruning is that the Z_t^i are nested ($Z_{t+1}^i \subset Z_t^i$) thus as soon as we can prove the emptiness of one set Z_t^i, we delete its associated

q_t^i function and do not have to consider its minimum anymore at any further iteration (proof in next Section 1.3). In dimension $p = 1$ this is reasonably easy. In this case, the sets Z_t^i $(i = 1, \dots, t)$ are unions of intervals and an efficient functional pruning rule is possible by updating a list of these intervals for Q_t. This approach is implemented in FPOP [31].

In dimension $p \geq 2$ it is not so easy anymore to keep track of the emptiness of the sets Z_t^i. We illustrate the dynamics of the Z_t^i sets in Figure 1 in the bi-variate Gaussian case. Each color is associated with a set Z_t^i (corresponding to a possible change at $i - 1$) for t equal 1 to 5. This plot shows in particular that sets Z_t^i can be non-convex.

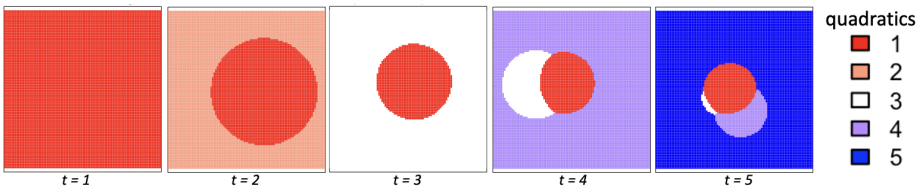

Figure 1: The sets Z_t^i over time for the bi-variate independent Gaussian model on time series without change $y = ((0.29, 1.93), (1.86, -0.02), (0.9, 2.51), (-1.26, 0.91), (1.22, 1.11))$. From left to right we represent at time $t = 1, 2, 3, 4$, and 5 the parameter space (θ^1, θ^2). Each Z_t^i is represented by a color. The change 1 associated with quadratics 2 is pruned at $t = 3$. Notice that each time sequence of Z_t^i with i fixed is a nested sequence of sets.

1.3. Geometric Formulation of Functional Pruning

To build an efficient pruning strategy for dimension $p \geq 2$ we need to test the emptiness of the sets Z_t^i at each iteration. Note that to get Z_t^i we need to compare the functional cost q_t^i with any other functional cost q_t^j, $j = 1, \dots, t$, $j \neq i$. This leads to the definition of the following sets.

Definition 1. *(S-type set) We define S-type set S_j^i using the function Ω as*

$$S_j^i = \left\{ \theta \in \mathbb{R}^p \mid \sum_{u=i+1}^{j} \Omega(\theta, y_u) \leq m_j - m_i \right\}, \text{ when } i < j$$

and $S_i^i = \mathbb{R}^p$. *We denote the set of all possible S-type sets as* $\mathbf{S}$.

To ease some of our calculations, we now introduce some additional notations. For $\theta = (\theta^1, \ldots, \theta^p)$ *in* $\mathbb{R}^p$, $1 \leq i < j \leq n$ *we define* p *univariate functions* $\theta^k \mapsto s_{ij}^k(\theta^k)$ *associated to the* k*-th time series as*

$$s_{ij}^k(\theta^k) \quad = \sum_{u=i+1}^{j} \omega(\theta^k, y_u^k), \quad k = 1, \ldots, p\,. \tag{1.5}$$

We introduce a constant Δ_{ij} *and a function* $\theta \mapsto s_{ij}(\theta)$*:*

$$\begin{cases} \Delta_{ij} = m_j - m_i\,, \\ s_{ij}(\theta) = \sum_{k=1}^{p} s_{ij}^k(\theta^k) - \Delta_{ij}\,, \end{cases} \tag{1.6}$$

where m_i *and* m_j *are defined as in (1.4). The sets* S_j^i *for* $i < j$ *are also described by relation*

$$S_j^i = s_{ij}^{-1}(-\infty, 0]\,. \tag{1.7}$$

In Figure 2 we present the level curves for three different parametric models given by $s_{ij}^{-1}(\{w\})$ *with* w *a real number. Each of these curves encloses an S-type set.*

At time $t = 1, \ldots, n$ we define the following sets associated to the last change point index $i - 1$:

`past set` $\mathcal{P}^i$

$$\mathcal{P}^i = \{S_i^u,\, u = 1, \ldots, i-1\}\,.$$

`future set` $\mathcal{F}^i(t)$

$$\mathcal{F}^i(t) = \{S_v^i,\, v = i, \ldots, t\}\,.$$

We denote the cardinal of a set $\mathcal{A}$ as $|\mathcal{A}|$. Using these two sets of sets, the Z_t^i have the following description.

Proposition 1. *At iteration* t*, the functional cost* $Q_t(\cdot)$ *defines the subsets* Z_t^i *(*$i = 1, \ldots, t$*), each of them being the intersection of the sets in* $\mathcal{F}^i(t)$ *minus the union of the sets in* $\mathcal{P}^i$*.*

$$Z_t^i = (\cap_{S \in \mathcal{F}^i(t)} S) \setminus (\cup_{S \in \mathcal{P}^i} S)\,, \quad i = 1, \ldots, t. \tag{1.8}$$

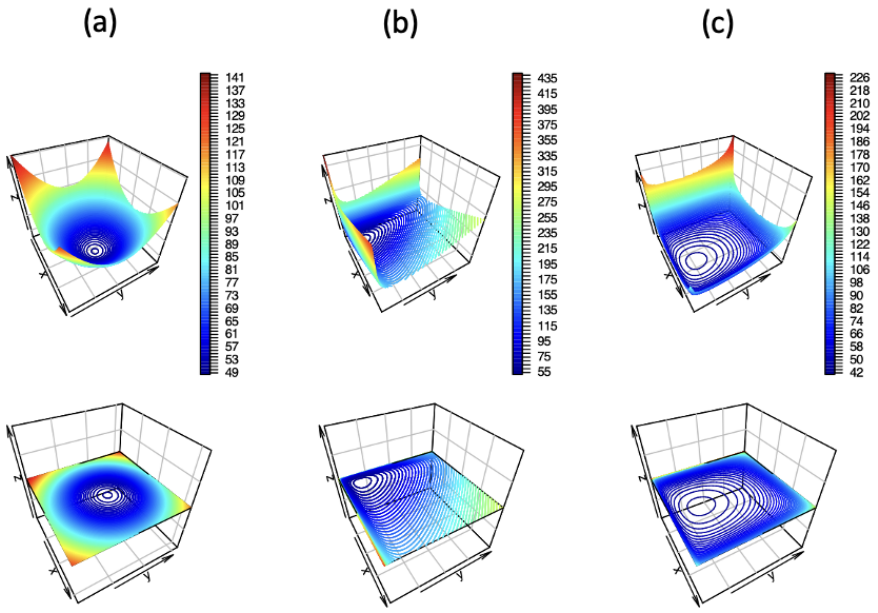

Figure 2: Three examples of the level curves of a function s_{ij} for bi-variate time series $\{x, y\}$. We use the following simulations for univariate time series : (a) $x \sim \mathcal{N}(0,1)$, $y \sim \mathcal{N}(0,1)$, (b) $x \sim \mathcal{P}(1)$, $y \sim \mathcal{P}(3)$, (c) $x \sim \mathcal{NB}(0.5,1)$, $y \sim \mathcal{NB}(0.8,1)$.

Proof. Based on the definition of the set Z_t^i, the proof is straightforward. Parameter value θ is in Z_t^i if and only if $q_t^i(\theta) \leq q_t^u(\theta)$ for all $u \neq i$; these inequalities define the past set (when $u < i$) and the future set (when $u > i$). By convention we assume that, in case $i = t$, $\cap_{S \in \mathcal{F}^i(t)} S = \mathbb{R}^p$. □

Corollary 1. *The sequence $\zeta^i = (Z_t^i)_{t \geq i}$ is a nested sequence of sets.*

Indeed, Z_{t+1}^i is equal to Z_t^i with an additional intersection in the future set. Based on Corollary 1, as soon as we prove that the set Z_t^i, is empty, we delete its associated q_t^i function and, consequently, we can prune the change point $i-1$. In this context, functional and inequality-based pruning have a simple geometric interpretation.

Functional pruning geometry. The position $i-1$ is pruned at step $t+1$, in $Q_{t+1}(\cdot)$, if the intersection set of $\cap_{S\in\mathcal{F}^i(t)}S$ is covered by the union set $\cup_{S\in\mathcal{P}^i}S$.

Inequality-based pruning geometry. The inequality-based pruning of PELT is equivalent to the geometric rule: position $i-1$ is pruned at step $t+1$ if the set S_t^i is empty. In that case, the intersection set $\cap_{S\in\mathcal{F}^i(t)}S$ is empty, and therefore Z_t^i is also empty using equation (1.8). This shows that if a change is pruned using inequality-based pruning it is also pruned using functional pruning. For the dimension $p=1$ this claim was theoretically proved in [31].

The construction of set Z_t^i using proposition 1 is illustrated in Figure 3 for a bi-variate independent Gaussian case: we have the intersection of three S-type sets and the subtraction of three S-type sets.

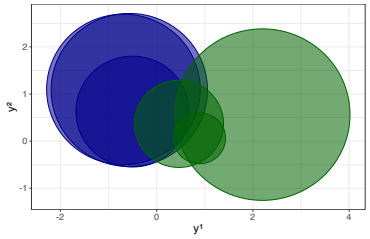

Figure 3: Examples of building a set Z_t^i with $|\mathcal{P}^i|=|\mathcal{F}^i(t)|=3$ for the Gaussian case in 2-D ($\mu=0,\sigma=1$). The green disks are S-type sets of the past set $\mathcal{P}^i$. The blue disks are S-type sets of the future set $\mathcal{F}^i(t)$.

2. Geometric Functional Pruning Optimal Partitioning

2.1. General Principle of GeomFPOP

Rather than considering an exact representation of the Z_t^i, our idea is to consider a hopefully slightly larger set that is easier to update. To be specific, for each Z_t^i we introduce $\tilde{Z}_t^i$, called *testing set*, such that $Z_t^i\subset\tilde{Z}_t^i$. If at time t $\tilde{Z}_t^i$ is empty thus is Z_t^i and thus change $i-1$ can be pruned. From 1 we have that starting from $Z=\mathbb{R}^p$ the set Z_t^i is obtained by successively applying two types of operations: intersection with an S-type set S ($Z\cap S$) or subtraction

of an S-type set S ($Z \setminus S$). Similarly, starting from $\tilde{Z} = \mathbb{R}^p$ we obtain $\tilde{Z}_t^i$ by successively applying approximation of these intersection and subtraction operations. Intuitively, the complexity of the resulting algorithm is a combination of the efficiency of the pruning and the easiness of updating the testing set.

A Generic Formulation of GeomFPOP. In what follows we will generically describe GeomFPOP, that is, without specifying the precise structure of the testing set $\tilde{Z}_t^i$. We call $\widetilde{\mathbf{Z}}$ the set of all possible $\tilde{Z}_t^i$ and assume the existence of two operators $\cap_{\tilde{Z}}$ and $\setminus_{\tilde{Z}}$. We have the following assumptions for these operators.

Definition 2. *The two operators $\cap_{\tilde{\mathbf{Z}}}$ and $\setminus_{\tilde{\mathbf{Z}}}$ are such that:*

(1) *the left input is a $\tilde{Z}$-type set (that is an element of $\tilde{\mathbf{Z}}$);*

(2) *the right input is a S-type set;*

(3) *the output is a $\tilde{Z}$-type set;*

(4) *$\tilde{Z} \cap S \subset \tilde{Z} \cap_{\tilde{\mathbf{Z}}} S$ and $\tilde{Z} \setminus S \subset \tilde{Z} \setminus_{\tilde{\mathbf{Z}}} S$.*

We give a proper description of two types of testing sets and their approximation operators in section 3.

At each iteration t GeomFPOP will construct $\tilde{Z}_{t+1}^i$ from $\tilde{Z}_t^i$, $\mathcal{P}^i$ and, $\mathcal{F}^i(t)$ iteratively using the two operators $\cap_{\tilde{\mathbf{Z}}}$ and $\setminus_{\tilde{\mathbf{Z}}}$. To be specific, we define S_j^F the j-th element of $\mathcal{F}^i(t)$ and S_P^j the j-th element of $\mathcal{P}^i$, we use the following iteration:

$$\begin{cases} A_0 = \tilde{Z}_t^i\,, \quad A_j = A_{j-1} \cap_{\tilde{\mathbf{Z}}} S_j^F\,, \quad j = 1, \ldots, |\mathcal{F}^i(t)|\,, \\ B_0 = A_{|\mathcal{F}^i(t)|}\,, \quad B_j = B_{j-1} \setminus_{\tilde{\mathbf{Z}}} S_P^j\,, \qquad j = 1, \ldots, |\mathcal{P}^i|\,, \end{cases}$$

and define $\tilde{Z}_{t+1}^i = B_{|\mathcal{P}^i|}$. Using the fourth property of Definition 2 and Proposition 1, we get that at any time of the algorithm $\tilde{Z}_t^i$ contains Z_t^i.

The pseudo-code of this procedure is described in Algorithm 1.

The `select`($\mathcal{A}$) step in Algorithm 1, where $\mathcal{A} \subset \mathbf{S}$, returns a subset of $\mathcal{A}$ in $\mathbf{S}$. By default, `select`($\mathcal{A}$) := $\mathcal{A}$.

Algorithm 1 Geometric update rule of $\tilde{Z}_t^i$

procedure `updateZone`$(\tilde{Z}_{t-1}^i, \mathcal{P}^i, \mathcal{F}^i(t), i, t)$
  $\tilde{Z}_t^i \leftarrow \tilde{Z}_{t-1}^i$
  for $S \in$ `select`$(\mathcal{F}^i(t))$ **do**
    $\tilde{Z}_t^i \leftarrow \tilde{Z}_t^i \cap_{\tilde{\mathrm{Z}}} S$
  end for
  for $S \in$ `select`$(\mathcal{P}^i)$ **do**
    $\tilde{Z}_t^i \leftarrow \tilde{Z}_t^i \setminus_{\tilde{\mathrm{Z}}} S$
  end for
  return $\tilde{Z}_t^i$

We denote the set of candidate change points at time t as τ_t. Note that for any $(i-1) \in \tau_t$ the sum of $|\mathcal{P}^i|$ and $|\mathcal{F}^i(t)|$ is $|\tau_t|$. With the default `select`() procedure we do $\mathcal{O}(p|\tau_t|)$ operations in Algorithm 1. By limiting the number of elements returned by `select`() we can reduce the complexity.

Remark 1. *For example, if the operator* $\mathcal{A} \mapsto$ `select`$(\mathcal{A})$*, regardless of* $|\mathcal{A}|$*, always returns a subset of constant size, then the overall complexity of GeomFPOP is at worst equal to that of PELT with* $\sum_{t=1}^{n} \mathcal{O}(p|\tau_t|)$ *time complexity.*

Using this `updateZone`() procedure we can now informally describe the GeomFPOP algorithm. At each iteration the algorithm will

(1) find the minimum value for Q_t, m_t; and the best position for last change point $\hat{\tau}_t$ (note that this step is standard: as in the PELT algorithm we need to minimize the cost of the last segment defined in equation 1.1);
(2) compute all sets $\tilde{Z}_t^i$ using $\tilde{Z}_{t-1}^i$, $\mathcal{P}^i$, and $\mathcal{F}^i(t)$ with the `updateZone`() procedure.
(3) Remove changes such that $\tilde{Z}_t^i$ is empty.

To simplify the pseudo-code of GeomFPOP, we also define the following operators:

(1) `bestCost&Tau`(t) operator returns two values: the minimum value of Q_t, m_t, and the best position for last change point $\hat{\tau}_t$ at time t (see Section 1.2);

(2) `getPastFutureSets`(i,t) operator returns a pair of sets $(\mathcal{F}^i(t),\ \mathcal{P}^i)$ for change point candidate $i-1$ at time t;

(3) `backtracking`$(\hat{\tau}, n)$ operator returns the optimal segmentation for $y_{1:n}$.

The pseudo-code of GeomFPOP is presented in Algorithm 2.

Algorithm 2 GeomFPOP algorithm

procedure `GeomFPOP`$(y, \Omega(\cdot,\cdot), \beta)$
$m_0 \leftarrow 0, \quad Q_0(\theta) \leftarrow 0\,, \quad \tau_0 \leftarrow \emptyset, \quad \{\tilde{Z}^i_{i-1}\}_{i\in\{1,\dots,n\}} \leftarrow \mathbb{R}^p$
for $t = 1, \dots, n$ **do**
  $Q_t(\theta) \leftarrow \min\{Q_{t-1}(\theta), m_{t-1} + \beta\} + \Omega(\theta, y_t)$
  $(m_t, \hat{\tau}_t) \leftarrow$ `bestCost&Tau`(t)
  for $i - 1 \in \tau_t$ **do**
    $(\mathcal{P}^i, \mathcal{F}^i(t)) \leftarrow$ `getPastFutureSets`(i,t)
    $\tilde{Z}^i_t \leftarrow$ `updateZone`$(\tilde{Z}^i_{t-1}, \mathcal{P}^i, \mathcal{F}^i(t), i, t)$
    if $\tilde{Z}^i_t = \emptyset$ **then**
      $\tau_t \leftarrow \tau_t \backslash \{i-1\}$
    end if
  end for
  $\tau_t \leftarrow (\tau_{t-1}, t-1)$
end for
return $cp(n) \leftarrow$ `backtracking`$(\hat{\tau}, n)$

3. Approximation Operators

The choice of the geometric structure and the way it is constructed directly affects the computational cost of the algorithm. We consider two types of testing set $\tilde{Z} \in \tilde{\mathbf{Z}}$, a S-type set $\tilde{S} \in \mathbf{S}$ (see Definition 1) and a hyperrectangle $\tilde{R} \in \mathbf{R}$ defined below.

Definition 3. *(Hyperrectangle) Given two vectors in* $\mathbb{R}^p$, $\tilde{l}$ *and* $\tilde{r}$ *we define the set* $\tilde{R}$*, called hyperrectangle, as:*

$$\tilde{R} = [\tilde{l}_1, \tilde{r}_1] \times \cdots \times [\tilde{l}_p, \tilde{r}_p] \,.$$

We denote the set of all possible sets $\tilde{R}$ *as* $\mathbf{R}$*.*

To update the testing sets we need to give a strict definition of the operators $\cap_{\tilde{Z}}$ and $\setminus_{\tilde{Z}}$ for each type of testing set. To facilitate the following discussion, we rename them. For the first type of geometric structure, we rename the testing set $\tilde{Z}$ as $\tilde{S}$, the operators $\cap_{\tilde{\mathrm{Z}}}$ and $\setminus_{\tilde{\mathrm{Z}}}$ as $\cap_{\mathrm{S}}$ and $\setminus_{\mathrm{S}}$ and $\tilde{Z}$-type approximation as S-type approximation. And, likewise, we rename the testing set $\tilde{Z}$ as $\tilde{R}$, the operators $\cap_{\tilde{\mathrm{Z}}}$ and $\setminus_{\tilde{\mathrm{Z}}}$ as $\cap_{\mathrm{R}}$ and $\setminus_{\mathrm{R}}$ and $\tilde{Z}$-type approximation as R-type approximation for the second type of geometric structure.

3.1. S-type Approximation

With this approach, our goal is to keep track of the fact that at time $t = 1, \ldots, n$ there is a pair of changes (u_1, u_2), with $u_1 < i < u_2 \leq t$ such that $S^i_{u_2} \subset S^{u_1}_i$ or there is a pair of changes (v_1, v_2), with $i < v_1 < v_2 \leq t$ such that $S^i_{v_1} \cap S^i_{v_2}$ is empty. If at time t at least one of these conditions is met, we can guarantee that the set $\tilde{S}$ is empty, otherwise, we propose to keep as the result of approximation the last future S-type set S^i_t, because it always includes the set Z^i_t. This allows us to quickly check and prove (if $\tilde{S} = \emptyset$) the emptiness of set Z^i_t.

We consider two generic S-type sets, S and $\tilde{S}$ from $\mathbf{S}$, described as in (1.5) by the functions s and $\tilde{s}$:

$$s(\theta) = \sum_{k=1}^{p} s^k(\theta^k) - \Delta \,, \qquad \tilde{s}(\theta) = \sum_{k=1}^{p} \tilde{s}^k(\theta^k) - \tilde{\Delta} \,.$$

Definition 4. *For all* S *and* $\tilde{S}$ *in* $\mathbf{S}$ *we define the operators* $\cap_{\mathrm{S}}$ *and* $\setminus_{\mathrm{S}}$ *as:*

$$\tilde{S} \cap_{\mathrm{S}} S \;\; = \begin{cases} \emptyset \,, & \text{if } \tilde{S} \cap S = \emptyset \,, \\ \tilde{S} \,, & \text{otherwise} \,. \end{cases}$$

$$\tilde{S} \setminus_{\mathrm{S}} S \qquad = \begin{cases} \emptyset \,, & \text{if } \tilde{S} \subset S \,, \\ \tilde{S} \,, & \text{otherwise} \,. \end{cases}$$

As a consequence, we only need an easy way to detect any of these two geometric configurations: $\tilde{S} \cap S$ and $\tilde{S} \subset S$.

In the Gaussian case, the S-type sets are p-balls and an easy solution exists based on comparing radii (see Appendix B for details). In the case of other models (as Poisson or negative binomial), intersection and inclusion tests can be performed based on a solution using separative hyperplanes and iterative algorithms for convex problems (see Appendix C). We propose another type of testing set solving all types of models with the same method.

3.2. R-type Approximation

Here, we approximate the sets Z_t^i by hyperrectangles $\tilde{R}_t^i \in \mathbf{R}$. A key insight of this approximation is that given a hyperrectangle R and an S-type set S we can efficiently (in $\mathcal{O}(p)$ using proposition 3) recover the best hyperrectangle approximation of $R \cup S$ and $R \setminus S$. Formally we define these operators as follows.

Definition 5. *(Hyperrectangles Operators $\cap_{\mathrm{R}}$, $\setminus_{\mathrm{R}}$) For all $R, \tilde{R} \in \mathbf{R}$ and $S \in \mathbf{S}$ we define the operators $\cap_{\mathrm{R}}$ and $\setminus_{\mathrm{R}}$ as:*

$$R \cap_{\mathrm{R}} S = \cap_{\{\tilde{R} | R \cap S \subset \mathbf{R}\}} \tilde{R}\,,$$
$$R \setminus_{\mathrm{R}} S = \cap_{\{\tilde{R} | R \setminus S \subset \mathbf{R}\}} \tilde{R}\,.$$

We now explain how we compute these two operators. First, we note that they can be recovered by solving a $2p$ one-dimensional optimization problems.

Proposition 2. *The k-th minimum coordinates $\tilde{l}_k$ and maximum coordinates $\tilde{r}_k$ of $\tilde{R} = R \cap_{\mathrm{R}} S$ (resp. $\tilde{R} = R \setminus_{\mathrm{R}} S$) is obtained as*

$$\tilde{l}_k \text{ or } \tilde{r}_k = \begin{cases} \min\limits_{\theta_k \in \mathbb{R}} \text{ or } \max\limits_{\theta_k \in \mathbb{R}} \theta_k\,, \\ \text{subject to } \varepsilon s(\theta) \leq 0\,, \\ \qquad\qquad l_j \leq \theta_j \leq r_j\,, \quad j = 1, \ldots, p\,, \end{cases} \tag{3.1}$$

with $\varepsilon = 1$ (resp. $\varepsilon = -1$).

To solve the previous problems ($\varepsilon = 1$ or -1), we define the following characteristic points.

Definition 6. *(Minimal, closest and farthest points) Let $S \in \mathbf{S}$, described by function $s(\theta) = \sum_{k=1}^{p} s^k(\theta^k) - \Delta$ from the family of functions (1.6), with $\theta \in \mathbb{R}^p$. We define the minimal point $\mathbf{c} \in \mathbb{R}^p$ of S as:*

$$\mathbf{c} = \left\{\mathbf{c}^k\right\}_{k=1,\dots,p}, \qquad \textit{with} \qquad \mathbf{c}^k = \underset{\theta^k \in \mathbb{R}}{\text{Arg min}}\{s^k(\theta^k)\}. \tag{3.2}$$

Moreover, with $R \in \mathbf{R}$ defined through vectors $l, r \in \mathbb{R}^p$, we define two points of R, the closest point $\mathbf{m} \in \mathbb{R}^p$ and the farthest point $\mathbf{M} \in \mathbb{R}^p$ relative to S as

$$\mathbf{m} = \left\{\mathbf{m}^k\right\}_{k=1,\dots,p}, \qquad \textit{with} \qquad \mathbf{m}^k = \underset{l^k \le \theta^k \le r^k}{\text{Arg min}} \left\{s^k(\theta^k)\right\},$$

$$\mathbf{M} = \left\{\mathbf{M}^k\right\}_{k=1,\dots,p}, \qquad \textit{with} \qquad \mathbf{M}^k = \underset{l^k \le \theta^k \le r^k}{\text{Arg max}} \left\{s^k(\theta^k)\right\}.$$

Remark 2. *In the Gaussian case, S is a ball in $\mathbb{R}^p$ and*

- **c** *is the center of the ball;*
- **m** *is the closest point to* **c** *inside R;*
- **M** *is the farthest point to* **c** *in R.*

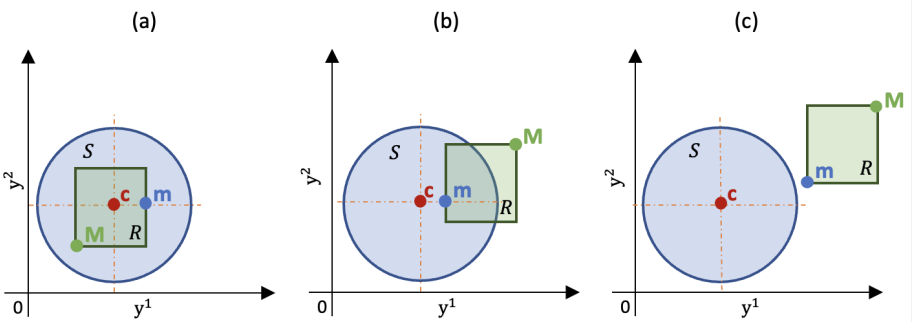

Figure 4: Three examples of minimal point **c**, closest point **m** and farthest point **M** for bi-variate Gaussian case: (a) $R \subset S$; (b) $R \cap S \neq \emptyset$; (c) $R \cap S = \emptyset$.

Proposition 3. *Let $\tilde{R} = R \cap_{\mathrm{R}} S$ (resp. $R \setminus_{\mathrm{R}} S$), with $R \in \mathbf{R}$ and $S \in \mathbf{S}$. We compute the boundaries $(\tilde{l}, \tilde{r})$ of $\tilde{R}$ using the following rule:*

(i) We define the point $\tilde{\theta} \in \mathbb{R}^p$ *as the closest point* $\mathbf{m}$ *(resp. farthest* $\mathbf{M}$*). For all* $k = 1, \dots p$ *we find the roots* θ^{k_1} *and* θ^{k_2} *of the one-variable* (θ^k) *equation*

$$s^k(\theta^k) + \sum_{j \neq k} s^j(\tilde{\theta}^j) - \Delta = 0\,.$$

If the roots are real-valued we consider that $\theta^{k_1} \leq \theta^{k_2}$*, otherwise we write* $\left]\theta^{k_1}, \theta^{k_2}\right[= \emptyset$*.*
(ii) We compute the boundary values $\tilde{l}^k$ *and* $\tilde{r}^k$ *of* $\tilde{R}$ *as:*

- *For* $R \cap_{\mathrm{R}} S$ $(k = 1, \dots, p)$*:*

$$\left[\tilde{l}^k, \tilde{r}^k\right] = \left[\theta^{k_1}, \theta^{k_2}\right] \cap \left[l^k, r^k\right]. \tag{3.3}$$

- *For* $R \setminus_{\mathrm{R}} S$ $(k = 1, \dots, p)$*:*

$$\left[\tilde{l}^k, \tilde{r}^k\right] = \begin{cases} \left[l^k, r^k\right] \setminus \left[\theta^{k_1}, \theta^{k_2}\right], & \text{if} \quad \left[\theta^{k_1}, \theta^{k_2}\right] \not\subset \left[l^k, r^k\right], \\ \left[l^k, r^k\right], & \text{otherwise}. \end{cases}$$

If there is a dimension k *for which* $\left[\tilde{l}^k, \tilde{r}^k\right] = \emptyset$*, then the set* $\tilde{R}$ *is empty.*

The proof of Proposition 3 is presented in Appendix D.

4. Simulation Study of GeomFPOP

In this section, we study the efficiency of GeomFPOP using simulations of multivariate independent time series. For this, we implemented GeomFPOP (with S and R types) and PELT for the Multivariate Independent Gaussian Model in the R-package 'GeomFPOP' (`https://github.com/lpishchagina/GeomFPOP`) written in R/C++. By default, the value of penalty β for each simulation was defined by the Schwarz Information Criterion proposed in [21] ($\beta = 2p \log n$).

Overview of our simulations. First, as a quality control we made sure that the output of PELT and GeomFPOP were identical on a number of simulated profiles. Second, we studied cases where the PELT approach is not efficient,

that is when the data has no or few changes relative to n. Indeed, it was shown in [11] and [31] that the run time of PELT is close to $\mathcal{O}(n^2)$ in such cases. So we considered simulations of multivariate time series without change (only one segment). By these simulations we evaluated the pruning efficiency of GeomFPOP (using S and R types) for dimension $2 \leq p \leq 10$ (see Figure 5 in Subsection 4.1). For small dimensions ($2 \leq p \leq 4$) we also evaluated the run time of GeomFPOP and PELT and compare them (see Figure 6 in Subsection 4.2). In addition, we considered another approximation of the Z_t^i where we applied our $\cap_{\mathrm{R}}$ and $\setminus_{\mathrm{R}}$ operators only for a randomly selected subset of the past and future balls. In practice, this strategy turned out to be faster computationally than the full/original GeomFPOP and PELT (see Figure 7 in Subsection 4.3). For this strategy we also generated time series of a fixed size (10^6 data points) and varying number of segments and evaluated how the run time vary with the number of segments for small dimensions ($2 \leq p \leq 4$). Our empirical results confirmed that the GeomFPOP (R-type : `random/random`) approach is computationally comparable to PELT when the number of changes is large (see Figure 9 in Subsection 4.3.2).

4.1. The Number of change point Candidates Stored Over Time

We evaluate the functional pruning efficiency of the GeomFPOP method using simulations with 10^4 data points (without change, i.e. i.i.d $\mathcal{N}_p(0, I_p)$). For such signals, PELT typically does not pruned (e.g. for $t = 10^4$, $p = 2$ it stores almost always t candidates).

We report in Figure 5 the percentage of candidates that are kept by GeomFPOP as a function of n, p and the type of pruning (R or S). Regardless of the type of approximation and contrary to PELT, we observe that there is some pruning. However when increasing the dimension p, the quality of the pruning decreases.

Comparing Figure 5 left and the right we see that for dimensions $p = 2$ to $p = 5$ R-type prunes more than the S-type, while for larger dimensions the S-type prunes more than the R-type. For example, for $p = 2$ at time $t = 10^4$ by

GeomFPOP (R-type) the number of candidates stored over t does not exceed 1% versus 3% by GeomFPOP (S-type). This intuitively makes sense. One the one hand the R-type approximation of a sphere gets worst with the dimension. On the other hand with R-type approximation every new approximation is included in the previous one. For small dimensions this memory effect outweight the roughness of the approximation.

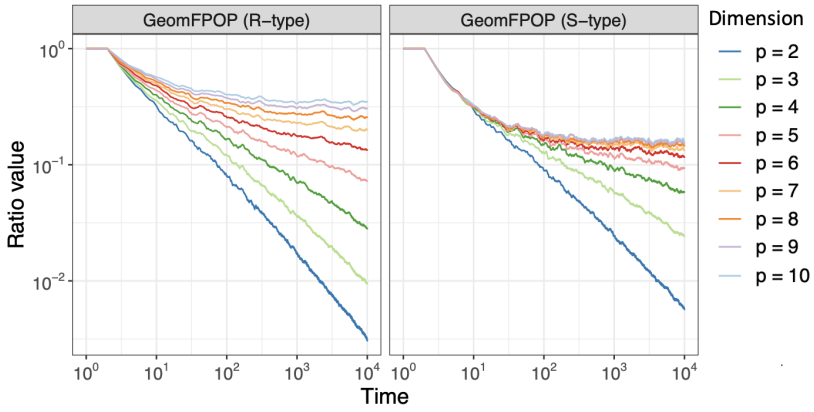

Figure 5: Percentage of candidate change points stored over time by GeomFPOP with R (left) or S (right) type pruning for dimension $p = 2, \ldots, 10$. We simulated 100 i.i.d Gaussian data $\mathcal{N}_p(0, I_p)$ and report the average.

Based on these results we expect that R-type pruning GeomFPOP will be more efficient than S-type pruning for small dimensions.

4.2. Empirical Time Complexity of GeomFPOP

We studied the run time of GeomFPOP (S and R-type) and compared it to PELT for small dimensions ($p = 2, 3, 4$). Run times were limited to three minutes and were recorded for simulations (without change, i.e i.i.d $\mathcal{N}_p(0, I_p)$). The results are presented in Figure 6. We observe that GeomFPOP is faster than PELT only for $p = 2$. For $p = 3$ run times are comparable and for $p = 4$ GeomFPOP is slower. This lead us to consider a randomized version of GeomFPOP (see next subsection).

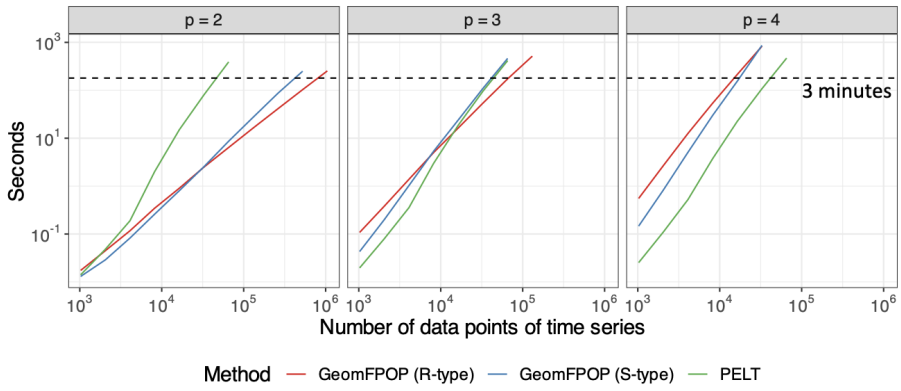

Figure 6: Run time of GeomFPOP (S and R types) and PELT using multivariate time series without change points. The maximum run time of the algorithms is 3 minutes. Averaged over 100 data sets.

4.3. Empirical Time Complexity of a randomized GeomFPOP

R-type GeomFPOP is designed in such a way that at each iteration we need to consider all past and future spheres of change i. In practice, it is often sufficient to consider just a few of them to get an empty set. Having this in mind, we propose a further approximation of the Z_t^i where we apply our $\cap_{\mathrm{R}}$ and $\setminus_{\mathrm{R}}$ operators only for a randomly selected subset of the past and future sets. In detail, we propose to redefine the output of the `select`() function in Algorithm 1 on any sets $\mathcal{P}^i$ and $\mathcal{F}^i(t)$ as:

- `select`$(\mathcal{P}^i)$ returns one random set from $\mathcal{P}^i$.
- `select`$(\mathcal{F}^i(t))$ returns the last set S_t^i and one random set from $\mathcal{F}^i(t)$.

Thus, we consider the following geometric update rule:

- (`random`/`random`) At time t we update hyperrectangle:
  (1) by only two intersection operations: one with the last S-type set S_t^i from $\mathcal{F}^i(t)$, and one with a random S-type set from $\mathcal{F}^i(t)$;
  (2) by only one exclusion operation with a random S-type set from $\mathcal{P}^i$.

In this approach at time t we do no more than three operations to update the testing set $\tilde{Z}_t^i$ for each $(i-1) \in \tau_t$. According to the Remark 1, even with large values of p, the overall complexity of GeomFPOP should not be worse than that of PELT. We investigated other randomized strategies but this simple one was sufficient to significantly improve run times. The run time of our optimization approach and PELT in dimension $(p = 2, \ldots, 10, 100)$ are presented in Figure 7. As in Subsection 4.2, run times were limited to three minutes and were recorded for simulations of length ranging from 2^{10} to 2^{23} data points (without change, i.e i.i.d $\mathcal{N}_p(0, I_p)$).

Although the (`random`/`random`) approach reduces the quality of pruning (see Appendix E), it gives a significant gain in run time compared to PELT in small dimensions. To be specific, with a run time of five minutes GeomFPOP, on average, processes a time series with a length of about 8×10^6, 10^6 and $2,5 \times 10^5$ data points in the dimensions $p = 2, 3$ and 4, respectively. At the same time, PELT manages to process time series with a length of at most $6,5 \times 10^4$ data points in these dimensions.

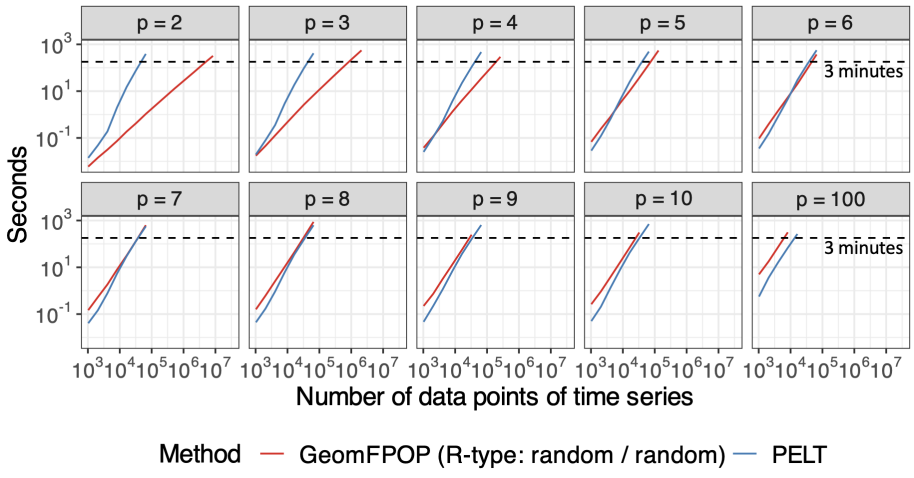

Figure 7: Run time of the (`random`/`random`) approach of GeomFPOP (R-type) and PELT using p-variate time series without change points $(p = 2, \ldots, 10, 100)$. The maximum run time of the algorithms is 3 minutes. Averaged over 100 data sets.

4.3.1. Empirical complexity of the algorithm as a function of p

We also evaluate the slope coefficient α of the run time curve of GeomFPOP with random sampling of the past and future candidates for all considered dimensions. In Figure 8 we can see that already for $p \geq 7$ α is close to 2.

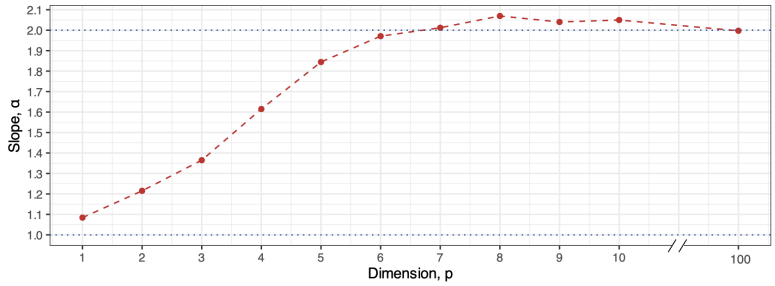

Figure 8: Run time dependence of (`random`/`random`) approach of GeomFPOP (R-type) on dimension p.

4.3.2. Run time as a function of the number of segments

For small dimensions ($2 \leq p \leq 4$) we also generated time series with 10^6 data points with increasing number of segments. We have considered the following number of segments: $(1, 2, 5) \times 10^i$(for $i = 0, \ldots, 3$) and 10^4. The mean was equal to 1 for even segments, and 0 for odd segments. In Figure 9 we can see the run time dependence of the (`random`/`random`) approach of GeomFPOP (R-type) and PELT on the number of segments for this type of time series. Interestingly, the run time of GeomFPOP (`random`/`random`) is comparable to PELT even when the number of segment is large. For smaller number of segments (as already observed) GeomFPOP (`random`/`random`) is an order of magnitude faster.

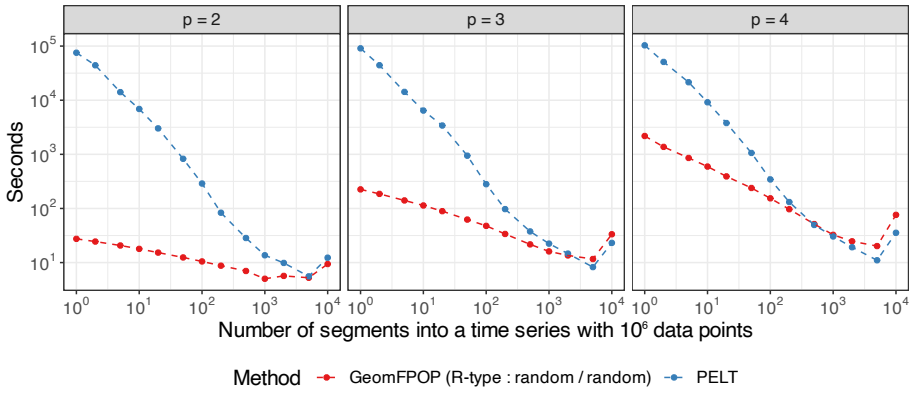

Figure 9: Run time dependence of (`random`/`random`) approach of GeomFPOP (R-type) on the number of segments in time series with 10^6 data points.

Acknowledgment

We thank Paul Fearnhead for fruitful discussions.

Appendix A. Examples of Likelihood-Based Cost Functions

We define a cost function for segmentation as in (1.1) by the function $\Omega(\cdot,\cdot)$ (the opposite log-likelihood (times two)). In Table A.1 is the expression of this function linked to data point $y_i = (y_i^1, \ldots, y_i^p) \in \mathbb{R}^p$ for three examples of Parametric Multivariate Models.

Table A.1: Likelihood-based cost function for the Gaussian, Poisson and Negative Binomial models. We suppose that the over-dispersion parameter ϕ of the Negative Binomial distribution is known.

Distribution	$\Omega(\theta, y_i)$
Gaussian	$\sum_{k=1}^{p}(y_i^k - \theta^k)^2$
Poisson	$2\sum_{k=1}^{p}\left\{\theta^k - \log\left(\frac{(\theta^k)^{y_i^k}}{y_i^k!}\right)\right\}$
Negative Binomial	$-2\sum_{k=1}^{p}\log\left((\theta^k)^{y_i^k}(1-\theta^k)^{\phi}\binom{y_i^k+\phi-1}{y_i^k}\right)$

Appendix B. Arrangement of Two p-balls

We define two p-balls, S and S' in $\mathbb{R}^p$ using their centers c, $c' \in \mathbb{R}^p$ and radius R, $R' \in \mathbb{R}^+$ as

$$S = \{x \in \mathbb{R}^p, ||x - c||^2 \leq R^2\} \text{ and } S' = \{x \in \mathbb{R}^p, ||x - c'||^2 \leq R'^2\},$$

where $||x - c||^2 = \sum_{k=1}^{p}(x^k - c^k)^2$, with $x = (x^1, ..., x^p) \in \mathbb{R}^p$, is the Euclidean norm. The distance between centers c and c' is defined as $d(c, c') = \sqrt{||c - c'||^2}$. We have the following simple results:

$$S \cap S' = \emptyset \iff d(c, c') > R + R' \,,$$

$$S \subset S' \text{ or } S' \subset S \iff d(c, c') \leq |R - R'| \,.$$

Appendix C. Intersection and Inclusion tests

Remark 3. *For any $S_j^i \in \mathbf{S}$ its associated function s can be redefine after normalization by constant $j-i+1$ as:*

$$s(\theta) = a(\theta) + \langle b, \theta \rangle + c,$$

with $a(\cdot)$ is some convex function depending on θ, $b = \{b^k\}_{k=1,\dots,p} \in \mathbb{R}^p$ and $c \in \mathbb{R}$.

For example, in the Gaussian case, the elements have the following form:

$$a : \theta \mapsto \theta^2\,, \quad b^k = 2\bar{Y}_{i:j}^k\,, \quad c = \bar{Y}_{i:j}^2 - \Delta_{ij}\,,$$

where $\bar{Y}_{i:j}^k = \frac{1}{j-i+1}\sum_{u=i+1}^{j} y_u^k$ and $\bar{Y}_{i:j}^2 = \frac{1}{j-i+1}\sum_{u=i+1}^{j}\sum_{k=1}^{p}(y_u^k)^2$.

Definition 7. *For all $\theta \in \mathbb{R}^p$ and $S_1, S_2 \in \mathbf{S}$ with their associated functions, s_1 and s_2, we define a function h_{12} and a hyperplane H_{12} as:*

$$h_{12}(\theta) := s_2(\theta) - s_1(\theta)\,, \quad H_{12} := \{\theta \in \mathbb{R}^p | h_{12}(\theta) = 0\}\,.$$

We denote by $H_{12}^+ := \{\theta \in \mathbb{R}^p | h_{12}(\theta) > 0\}$ and $H_{12}^- := \{\theta \in \mathbb{R}^p | h_{12}(\theta) < 0\}$ the positive and negative half-spaces of H_{12}, respectively. We call $\mathbf{H}$ the set of hyperplanes.

For all $S \in \mathbf{S}$ and $H \in \mathbf{H}$ we introduce a `half-space` operator.

Definition 8. *The operator* `half-space` *is such that:*

(1) *the left input is an S-type set S;*
(2) *the right input is a hyperplane H;*
(3) *the output is the half-spaces of H, such that S lies in those half-spaces.*

Definition 9. *We define the output of* `half-space`(S, H) *by the following rule:*

(1) *We find two points, $\theta_1, \theta_2 \in \mathbb{R}^p$, as:*

$$\begin{cases} \theta_1 = & \underset{\theta \in S}{\operatorname{Arg\,min}}\, s(\theta), \\ \theta_2 = & \begin{cases} \underset{\theta \in S}{\operatorname{Arg\,min}}\, h(\theta), & \text{if } \theta_1 \in H^+, \\ \underset{\theta \in S}{\operatorname{Arg\,max}}\, h(\theta), & \text{if } \theta_1 \in H^-. \end{cases} \end{cases}$$

(2) *We have:*

$$\texttt{half-space}(S, H) = \begin{cases} \{H^+\}, & \textit{if } \theta_1, \theta_2 \in H^+, \\ \{H^-\}, & \textit{if } \theta_1, \theta_2 \in H^-, \\ \{H^+, H^-\}, & \textit{otherwise.} \end{cases}$$

Lemma 1. *$S_1 \subset H_{12}^- \Leftrightarrow \partial S_1 \subset H_{12}^-$, where $\partial(\cdot)$ denote the frontier operator.*

The proof of Lemma 1 follows from the convexity of S_1.

Lemma 2. *$S_1 \subset S_2$ (resp. $S_2 \subset S_1$) $\Leftrightarrow S_1, S_2 \subset H_{12}^-$ (resp. $S_1, S_2 \subset H_{12}^+$).*

Proof. We have the hypothesis $\mathcal{H}_0 : \{S_1 \subset S_2\}$, then

$$\forall \theta \in \partial S_1 \quad \begin{cases} s_1(\theta) = 0\,, & \text{[by Definition 1]} \\ s_2(\theta) \leq 0\,, & \text{[by } \mathcal{H}_0\text{]} \end{cases} \Rightarrow \theta \in H_{12}^- \quad \Rightarrow \partial S_1 \subset H_{12}^-.$$

Thus, according to Lemma 1, $S_1 \subset H_{12}^-$.

We have now the hypothesis $\mathcal{H}_0 : \{S_1, S_2 \subset H_{12}^-\}$, then

$$\forall \theta \in S_1 \quad \begin{cases} s_1(\theta) \leq 0, & \text{[by Definition 1]} \\ h_{12}(\theta) < 0, & \text{[by } \mathcal{H}_0, \text{ Definition 1]} \end{cases} \Rightarrow \theta \in S_2 \quad \Rightarrow S_1 \subset S_2.$$

Similarly, it is easy to show that $S_2 \subset S_1 \Leftrightarrow S_1, S_2 \subset H_{12}^+$. □

Lemma 3. *$S_1 \cap S_2 = \emptyset \Leftrightarrow H_{12}$ is a separating hyperplane of S_1 and S_2.*

Proof. We have the hypothesis $\mathcal{H}_0 : \{S_1 \subset H_{12}^+, S_2 \subset H_{12}^-\}$. Thus, H_{12} is a separating hyperplane of S_1 and S_2 then, according to its definition, $S_1 \cap S_2 = \emptyset$.

We have now the hypothesis $\mathcal{H}_0 : \{S_1 \cap S_2 = \emptyset\}$ then

$$\forall \theta \in S_1 \quad \begin{cases} s_1(\theta) \leq 0\,, & \text{[by Definition 1]} \\ s_2(\theta) > 0\,, & \text{[by } \mathcal{H}_0, \text{ Definition 1]} \end{cases} \Rightarrow \theta \in H_{12}^+.$$

$$\forall \theta \in S_2 \quad \begin{cases} s_1(\theta) > 0\,, & \text{[by } \mathcal{H}_0, \text{ Definition 1]} \\ s_2(\theta) \leq 0\,, & \text{[by Definition 1]} \end{cases} \Rightarrow \theta \in H_{12}^-.$$

Consequently, H_{12} is a separating hyperplane of S_1 and S_2. □

Proposition 4. *To detect set inclusion $S_1 \subset S_2$ and emptiness of set intersection $S_1 \cap S_2$, it is necessary:*

(1) *build the hyperplane H_{12};*

(2) *apply the* `half-space` *operator for couples* (S_1, H_{12}) *and* (S_2, H_{12}) *to know in which half-space(s)* S_1 *and* S_2 *are located;*

(3) *check the conditions in the Lemmas 2 and 3.*

Appendix D. Proof of Proposition 3

For the proof of Proposition 3 we need the following remark.

Remark 4. *With set* $S \in \mathbf{S}$ *the maximum and minimum values for each coordinate in* S *are obtained on the axis going through minimal point* $\mathbf{c}$*.*

Proof. Let $\mathbf{c} = \{\mathbf{c}^k\}_{k=1,\dots,p}$ is the minimal point of S, defined as in (3.2). In the intersection case, we consider solving the optimization problem (3.1) for the boundaries $\tilde{l}^k$ and $\tilde{r}^k$, removing constraint $l^k \leq \theta^k \leq r^k$. If R intersects S, the optimal solution θ^k belongs to the boundary of S due to our simple (axis-aligned rectangular) inequality constraints and we get

$$s^k(\theta^k) = -\sum_{j \neq k} s^j(\theta^j) + \Delta\,. \tag{D.1}$$

We are looking for minimum and maximum values in θ^k for this equation with constraints $l^j \leq \theta^j \leq r^j$ $(j \neq k)$. Using the convexity of s^k and s^j, we need to maximize the quantity in the right-hand side. Thus, the solution $\tilde{\theta}^j$ for each θ^j is the minimal value of $\sum_{j \neq k} s^j(\theta^j)$ under constraint $l^j \leq \theta^j \leq r^j$ and the result can only be l^j, r^j or $\mathbf{c}^j$. This decomposition in smaller problems is made possible thanks to our problem setting with independence. Looking at all coordinates at the same time, the values for $\tilde{\theta} \in \mathbb{R}^p$ corresponds to the closest point $\mathbf{m} = \{\mathbf{m}^k\}_{k=1,\dots,p}$. Having found θ^{k_1} and θ^{k_2} using $\tilde{\theta}$ the result in (3.3) is obvious considering current boundaries l^k and r^k.

In exclusion case, we remove from R the biggest possible rectangle included into $S \cap \{l^j \leq \theta^j \leq r^j\,,\, j \neq k\}$, which correspond to minimizing the right hand side of (D.1), that is maximizing $\sum_{j \neq k} s^j(\theta^j)$ under constraint $l^j \leq \theta^j \leq r^j$ $(j \neq k)$. In that case, the values for $\tilde{\theta}$ correspond to the greatest value returned by $\sum_{j \neq k} s^j(\theta^j)$ on interval boundaries. With convex functions s^j, it corresponds to the farthest point $\mathbf{M} = \{\mathbf{M}^k\}_{k=1,\dots,p}$. □

Appendix E. Optimization Strategies for GeomFPOP(R-type)

In GeomFPOP(R-type) at each iteration, we need to consider all past and future spheres of change i. As it was said in Section 4, in practice it is often sufficient to consider just a few of them to get an empty set. Thus, we propose to limit the number of operations $\cap_{\mathrm{R}}$ no more than two:

- `last`. At time t we update hyperrectangle by only one operation, this is an intersection with the last S-type set S_t^i from $\mathcal{F}^i(t)$.
- `random`. At time t we update the hyperrectangle by only two operations. First, this is an intersection with the last S-type set S_t^i from $\mathcal{F}^i(t)$, and second, this is an intersection with other random S-type set from $\mathcal{F}^i(t)$.

The number of operations $\setminus_{\mathrm{R}}$ we limit no more than one:

- `empty`. At time t we do not perform $\setminus_{\mathrm{R}}$ operations.
- `random`. At time t we update hyperrectangle by only one operation: exclusion with a random S-type set from $\mathcal{P}^i$.

According to these notations, the approach presented in the original GeomFPOP (R-type) has the form (`all/all`). We show the impact of introduced limits on the number of change point candidates retained over time and evaluate their run times. The results are presented in Figures E.10 and E.11.

Even though the (`random/random`) approach reduces the quality of pruning in dimensions $p = 2, 3$ and 4, it gives a significant gain in the run time compared to the original GeomFPOP (R-type) and is at least comparable to the (`last/random`) approach.

References

[1] C. Data, C. Statistics, B. Applications, D. Sciences, N. Council, Frontiers in massive data analysis, The National Academies Press., 2013. doi:10.17226/18374.

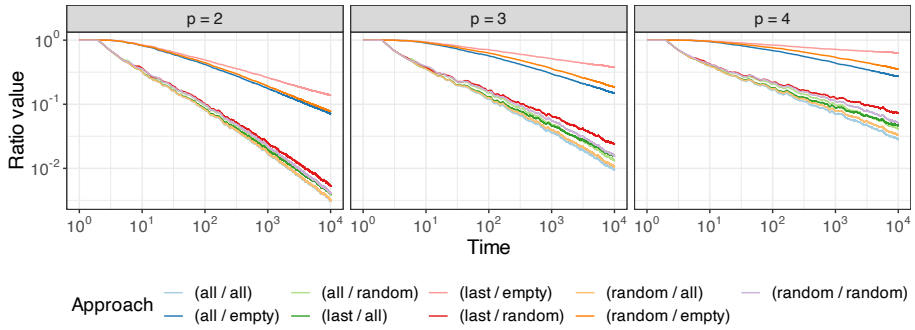

Figure E.10: Ratio number of candidate change point over time by different optimization approaches of GeomFPOP (R-type) in dimension $p = 2, 3$ and 4. Averaged over 100 data sets without changes with 10^4 data points.

[2] A. Olshen, E. Venkatraman, R. Lucito, M. Wigler, Circular binary segmentation for the analysis of array-based dna copy number data., Biostatistics (Oxford, England) 5 (2004) 557–72. doi:10.1093/biostatistics/kxh008.

[3] F. Picard, S. Robin, M. Lavielle, C. Vaisse, J.-J. Daudin, A statistical approach for array cgh data analysis, BMC Bioinformatics 6 (2005) np. doi:10.1186/1471-2105-6-27.
URL https://hal.archives-ouvertes.fr/hal-01222433

[4] J. Bai, P. Perron, Computation and analysis of multiple structural-change., Journal of Applied Econometrics 18 (01 2003).

[5] A. Aue, L. Horváth, M. Hušková, P. Kokoszka, Change-point monitoring in linear models., The Econometrics Journal 9 (3) (2006) 373–403.
URL http://www.jstor.org/stable/23114925

[6] M. Bosc, F. Heitz, J.-P. Armspach, I. Namer, D. Gounot, L. Rumbach, Automatic change detection in multimodal serial mri: application to multiple sclerosis lesion evolution., NeuroImage 20(2) (2003) 643–656doi:https://doi.org/10.1016/S1053-8119(03)00406-3.

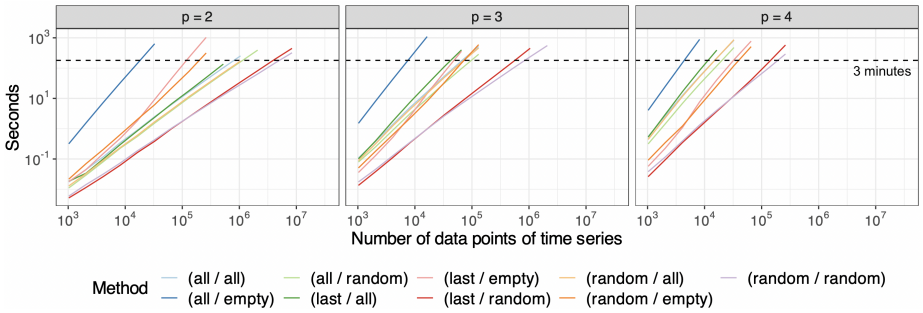

Figure E.11: Run time of different optimization approaches of GeomFPOP (R-type) using multivariate time series without change points. The maximum run time of the algorithms is 3 minutes. Averaged over 100 data sets.

URL https://www.sciencedirect.com/science/article/pii/S1053811903004063

[7] M. Staudacher, S. Telser, A. Amann, H. Hinterhuber, M. Ritsch-Marte, A new method for change-point detection developed for on-line analysis of the heart beat variability during sleep., Physica A-statistical Mechanics and Its Applications 349 (2005) 582–596.

[8] R. Malladi, G. P. Kalamangalam, B. Aazhang, Online bayesian change point detection algorithms for segmentation of epileptic activity., 2013 Asilomar Conference on Signals, Systems and Computers (2013) 1833–1837.

[9] J. Reeves, J. Chen, X. L. Wang, R. Lund, Q. Q. Lu, A review and comparison of changepoint detection techniques for climate data., Journal of Applied Meteorology and Climatology 46 (6) (2007) 900 – 915. doi:10.1175/JAM2493.1.
URL https://journals.ametsoc.org/view/journals/apme/46/6/jam2493.1.xml

[10] J.-F. Ducré-Robitaille, L. A. Vincent, G. Boulet, Comparison of techniques for detection of discontinuities in temperature series., International Journal of Climatology 23 (2003).

[11] R. Killick, P. Fearnhead, I. A. Eckley, Optimal detection of changepoints with a linear computational cost., Journal of the American Statistical Association 107 (500) (2012) 1590–1598. arXiv:https://doi.org/10.1080/01621459.2012.737745.

[12] I. Naoki, J. Kurths, Change-point detection of climate time series by nonparametric method., Lecture Notes in Engineering and Computer Science 2186 (10 2010).

[13] E. Andreou, E. Ghysels, Detecting multiple breaks in financial market volatility dynamics., Journal of Applied Econometrics 17 (5) (2002) 579–600.
URL http://www.jstor.org/stable/4129273

[14] P. Fryzlewicz, Wild binary segmentation for multiple change-point detection., The Annals of Statistics 42 (6) (dec 2014). doi:10.1214/14-aos1245.
URL https://doi.org/10.1214%2F14-aos1245

[15] E. Galceran, A. Cunningham, R. Eustice, E. Olson, Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment., Autonomous Robots 41 (08 2017). doi:10.1007/s10514-017-9619-z.

[16] D. Rybach, C. Gollan, R. Schluter, H. Ney, Audio segmentation for speech recognition using segment features., in: 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, 2009, pp. 4197–4200. doi:10.1109/ICASSP.2009.4960554.

[17] R. Radke, S. Andra, O. Al-Kofahi, B. Roysam, Image change detection algorithms: a systematic survey., IEEE Transactions on Image Processing 14 (3) (2005) 294–307. doi:10.1109/TIP.2004.838698.

[18] R. A. Davis, T. C. Lee, G. A. Rodriguez-Yam, Structural break estimation for nonstationary time series models., Journal of the American Statistical

Association 101 (2006) 223–239.
URL https://EconPapers.repec.org/RePEc:bes:jnlasa:v:101:y:2006:p:223-239

[19] A. Ranganathan, Pliss: Labeling places using online changepoint detection., Auton. Robots 32 (4) (2012) 351–368. doi:10.1007/s10514-012-9273-4.
URL https://doi.org/10.1007/s10514-012-9273-4

[20] S. Jewell, P. Fearnhead, D. Witten, Testing for a change in mean after changepoint detection. (2019). doi:10.48550/ARXIV.1910.04291.
URL https://arxiv.org/abs/1910.04291

[21] Y.-C. Yao, Estimation of a noisy discrete-time step function: Bayes and empirical bayes approaches., The Annals of Statistics 12 (4) (1984) 1434–1447. doi:10.1214/aos/1176346802.
URL https://doi.org/10.1214/aos/1176346802

[22] E. Lebarbier, Detecting multiple change-points in the mean of gaussian process by model selection., Signal Processing 85 (2005) 717–736. doi:10.1016/j.sigpro.2004.11.012.

[23] Z. Harchaoui, C. Lévy-Leduc, Multiple change-point estimation with a total variation penalty., Journal of the American Statistical Association. 105 (492) (2010) 1480–1493.
URL http://www.jstor.org/stable/27920180

[24] K. Frick, A. Munk, H. Sieling, Multiscale change-point inference. (2013). doi:10.48550/ARXIV.1301.7212.
URL https://arxiv.org/abs/1301.7212

[25] A. Anastasiou, P. Fryzlewicz, Detecting multiple generalized change-points by isolating single ones., Metrika 85 (02 2022). doi:10.1007/s00184-021-00821-6.

[26] C. Truong, L. Oudre, N. Vayatis, Selective review of offline change point detection methods, Signal Processing 167 (2020) 107299.

[27] S. Aminikhanghahi, D. J. Cook, A survey of methods for time series change point detection., Knowledge and information systems 51 (2) (2017) 339–367.

[28] I. E. Auger, C. E. Lawrence, Algorithms for the optimal identification of segment neighborhoods., Bulletin of Mathematical Biology 51 (1) (1989) 39–54. doi:10.1007/BF02458835.
URL https://doi.org/10.1007/BF02458835

[29] B. Jackson, J. D. Scargle, D. Barnes, S. Arabhi, A. Alt, P. Gioumousis, E. Gwin, P. Sangtrakulcharoen, L. Tan, T. T. Tsai, An algorithm for optimal partitioning of data on an interval, IEEE Signal Processing Letters 12 (2) (2005) 105–108.

[30] G. Rigaill, A pruned dynamic programming algorithm to recover the best segmentations with 1 to k_{max} change-points, Journal de la société française de statistique 156 (4) (2015) 180–205.
URL http://www.numdam.org/item/JSFS_2015__156_4_180_0/

[31] R. Maidstone, T. Hocking, G. Rigaill, P. Fearnhead, On optimal multiple changepoint algorithms for large data., Statistics and Computing 27 (2) (2017) 519–533.

[32] M. Lavielle, E. Moulines, Least-squares estimation of an unknown number of shifts in a time series., Journal of time series analysis 21 (1) (2000) 33–59.

[33] P. Fearnhead, R. Maidstone, A. Letchford, Detecting changes in slope with an l0 penalty., Journal of Computational and Graphical Statistics (2018) 1–11.

[34] W. R. Lai, M. D. Johnson, R. Kucherlapati, P. J. Park, Comparative analysis of algorithms for identifying amplifications and deletions in array cgh data., Bioinformatics 21 (19) (2005) 3763–3770.

[35] A. Liehrmann, G. Rigaill, T. D. Hocking, Increased peak detection accuracy in over-dispersed chip-seq data with supervised segmentation models, BMC bioinformatics 22 (1) (2021) 1–18.

[36] V. Runge, Is a finite intersection of balls covered by a finite union of balls in euclidean spaces?, Journal of Optimization Theory and Applications 187 (2) (2020) 431–447.

[37] Y.-C. Yao, Estimating the number of change-points via schwarz' criterion, Statistics & Probability Letters 6 (3) (1988) 181–189.
URL https://EconPapers.repec.org/RePEc:eee:stapro:v:6:y:1988:i:3:p:181-189

[38] N. Zhang, S. David, A modified bayes information criterion with applications to the analysis of comparative genomic hybridization data., Biometrics 63 (2007) 22–32. doi:10.1111/j.1541-0420.2006.00662.x.

[39] N. Verzelen, M. Fromont, M. Lerasle, P. Reynaud-Bouret, Optimal change-point detection and localization. (2020). doi:10.48550/ARXIV.2010.11470.
URL https://arxiv.org/abs/2010.11470