Journal de la Société Française de Statistique
Vol. 156 No. 4 (2015)

A pruned dynamic programming algorithm to recover the best segmentations with 1 to K_{max} change-points.

Titre: Un algorithme de programmation dynamique élagué pour trouver les meilleures segmentations avec 1 à K_{max} ruptures.

Guillem Rigaill[1]

Abstract: A common computational problem in multiple change-point models is to recover the segmentations with 1 to K_{max} change-points of minimal cost with respect to some loss function. Here we present an algorithm to prune the set of candidate change-points which is based on a functional representation of the cost of segmentations. We study the worst case complexity of the algorithm when there is a unidimensional parameter per segment and demonstrate that it is at worst equivalent to the complexity of the segment neighbourhood algorithm: $\mathcal{O}(K_{max}n^2)$. For a particular loss function we demonstrate that pruning is on average efficient even if there are no change-points in the signal. Finally, we empirically study the performance of the algorithm in the case of the quadratic loss and show that it is faster than the segment neighbourhood algorithm.

Résumé : Dans les modèles de ruptures multiples la recherche des segmentations de coût minimal, au sens d'une certaine fonction de perte, avec une à K_{max} ruptures est un problème algorithmique courant. Ici, nous présentons un algorithme, basé sur une représentation fonctionnelle des segmentations, qui élague l'ensemble des ruptures candidates. Nous étudions la complexité au pire de cet algorithme quand il y a un paramètre unidimensionnel par segment. Nous démontrons dans ce cas que ça complexité est équivalente à celle de l'algorithme « segment neighborhood » en $\mathcal{O}(K_{max}n^2)$. Pour une fonction de perte particulière nous démontrons que l'élagage est en moyenne efficace quand il n'y a pas de ruptures dans le signal. Enfin nous étudions empiriquement les performances de l'algorithme dans le cas de la perte quadratique et montrons qu'il est plus rapide que l'algorithme « segment neighborhood ».

Keywords: multiple-change-point detection, dynamic programming, functional cost, segment neighbourhood
Mots-clés : détection de ruptures multiples, programmation dynamique, coût fonctionnel
AMS 2000 subject classifications: 35L05, 35L70

1. Introduction

A common computational problem in multiple change-point models is to recover segmentations with 1 to K_{max} change-points of minimal cost where the cost is some well chosen criteria, such as minus the log-likelihood or the quadratic loss (Bai and Perron, 2003; Picard et al., 2005; Harchaoui and Cappé, 2007; Guédon et al., 2007; Killick and Eckley, 2011; Arlot et al., 2012; Cleynen and Lebarbier, 2014; Cleynen et al., 2014a). Many algorithms have been proposed to exactly solve this problem (Bellman, 1961; Fisher, 1958; Auger and Lawrence, 1989; Bai and Perron, 2003; Guédon, 2008). All these are dynamic programming algorithms and have a complexity which

[1] Institut of Plant Sciences Paris-Saclay (IPS2), UMR 9213/UMR1403, CNRS, INRA, Université Paris-Sud, Université d'Evry, Université Paris-Diderot, Sorbonne Paris-Cité, Bâtiment 630, 91405 0rsay, France
E-mail: rigaill@inra.evry.fr

 ISSN: 2102-6238

is linear in K_{max} and quadratic in the length of the signal n: $\mathcal{O}(K_{max}n^2)$. We will refer to these algorithms as segment neighbourhood algorithms.

In practice, this quadratic complexity in n is a problem. Indeed, if n is larger than 10^5 or 10^6 one run of the algorithm can take several hours or even days. In applications such as DNA copy number studies the signal length is typically of this order: $n = 10^5$-10^6. Several strategies have been developed to cope with this problem, the most famous is probably the binary segmentation heuristic (Scott and Knott, 1974). A common idea is to first identify a restricted set of candidate change-points with any fast heuristic and then run a segment neighbourhood algorithm on this restricted set (Kolesnikov and Fränti, 2003; Harchaoui and Lévy-Leduc, 2010; Gey et al., 2008). These heuristics typically have a complexity which is linear in n. From a computational perspective the main drawback of these approaches is that they lack optimality.

Most algorithms and heuristics that aim at recovering the segmentations of minimal cost operate on segmentations through their costs. Here we propose a new representation of the segmentations that we call the functional cost where the cost of a segmentation is represented as a function of a, possibly multidimensional, parameter. We demonstrate that using this functional cost it is theoretically possible to prune the set of segmentations while searching for segmentations of minimal costs with 1 to K_{max} change-points and thus to carry out the calculation only for a small subset of candidate segmentations. We call the resulting algorithm pruned dynamic programming algorithm (pDPA).

From an intuitive point view, if we consider a random sequence of n data-points with K true abrupt change-points, we expect that the segmentation with those K change-points will greatly outperform other possible segmentations in terms of cost. Hence it makes sense that it is possible to efficiently prune the set of segmentations. On the contrary, if we now consider a random sequence of n data-points without any change-points, all segmentations will have roughly the same cost and we do not expect an efficient pruning of the set of segmentations. This intuition is indeed true if segmentations are represented through their cost. In the context of another change-point optimization problem, the optimal partitioning problem (Jackson et al., 2005), this idea was made particularly clear by Killick et al. (2012).

For this problem, the PELT algorithm, that prunes segmentations based on their cost, was proven to be linear if the true number of change-points is linear in the number of data-points. If it is not linear, in particular if there are no change-points, PELT's pruning is less efficient.

Here we argue that if we consider the functional costs of segmentations, rather than their costs, we can have an efficient pruning even for random sequences without any change-points. As a proof of concept we study the complexity of the pDPA for change-point models with a unidimensional parameter per segment. In that case, we demonstrate that the algorithm is at worst as efficient as segment neighbourhood algorithms. For a special loss function we demonstrate that on average pruning is efficient even if we consider random sequences without change-points and we retrieve an average complexity which is sub-quadratic in $\mathcal{O}(n\log(n))$. We implemented the algorithm for the commonly used quadratic loss and empirically show that it is faster than segment neighbourhood algorithms.

Related works Since the first pre-print of this work (Rigaill, 2010), the pDPA has been implemented for other losses (Cleynen et al., 2014b), and in the context of DNA copy number

 ISSN: 2102-6238

analysis its competitive runtime compared to the segment neighbourhood algorithm, PELT or other approaches for this problem was confirmed by others (Hocking et al., 2013; Hocking, 2013; Maidstone et al., 2014). Furthermore, we demonstrate in this new version of the paper, for a simple, yet non-trivial, loss function that on average the pDPA pruning is efficient even if we consider random sequences without change-points.

Finally, the main contribution of this paper is of a computational nature, *i.e* an algorithm that recovers the segmentations with 1 to K_{max} change-points of minimal cost. From a computational point of view several methods boil down to this specific problem. Importantly, the statistical properties of these methods have been studied theoretically and empirically through simulations and on real data by many (Yao and Au, 1989; Horváth, 1993; Lavielle and Moulines, 2000; Lavielle, 2005; Lebarbier, 2005; Birgé and Massart, 2007; Boysen et al., 2009; Cleynen and Lebarbier, 2014; Zhang and Siegmund, 2007; Lai et al., 2005; Cleynen et al., 2014a), and recently the pDPA as implemented in the R cghseg package for the quadratic loss was shown to reach state of the art performances for the segmentation of DNA copy number profiles (Hocking et al., 2013).

Outline Section 2 describes the change-point framework used in the paper. Section 3 gives a quick overview of segment neighbourhood algorithms and informally describes the functional cost and the pDPA. Section 4 gives a detailed description of the functional cost and the pDPA. Section 5 is about the worst case, average and empirical complexity of the pDPA.

2. Change-point framework and cost minimization

Data We assume that we have a sequence of n observations. We denote this sequence $Y = \{Y_t\}_{1\leq t\leq n}$ and denote $Y_{i:j}$ a subsequence between data-point i and data-point $j-1$, *i.e.* $Y_{i:j} = \{Y_t\}_{i\leq t<j}$.

Segmentations We define a segmentation m of Y by a set of K change-points splitting the sequence in $K+1$ segments. We denote the positions of these K change-points τ_k for $k = 1,\dots,K$ and we also set the convention that $\tau_0 = 1$ and $\tau_{K+1} = n+1$. We call r_k the k-th segment of a segmentation: $r_k = \tau_{k-1} : \tau_k = \{i|\tau_{k-1} \leq i < \tau_k\}$. τ_{k-1} is thus the first data-point belonging to r_k and τ_k is the first data-point not belonging to r_k. For $K > 0$, we define $\mathscr{M}^K_{1:t}$ as the set of all possible segmentations with exactly K change-points of the sequence $Y_{1:t}$. There are $\binom{t-2}{K}$ such segmentations and so for the whole sequence we have $|\mathscr{M}^K_{1:n+1}| = \binom{n-1}{K}$.

Statistical model Our goal is to infer from the data both the positions and the number of change-points. Many methods, statistical models and model selection criteria have been proposed to address this problem for various types of data and various types of changes, *i.e.* change in the mean, in the variance, in the distribution etc...(Lavielle, 2005; Cleynen and Lebarbier, 2014; Arlot et al., 2012). In practice most of these methods critically rely on algorithms or heuristics that explore the set of all possible segmentations in search for a list of optimal segmentations w.r.t. some well chosen statistical criteria. However, this set of segmentations is extremely large ($|\mathscr{M}^K_{1:n+1}| = \binom{n-1}{K}$) and it is hard to explore it efficiently, especially when n is large.

From a computational point of view this exploration problem is often re-formulated as a cost minimization problem under the constraint that the number of change-points is $K = 1, 2 \cdots K_{max}$,

 ISSN: 2102-6238

where K_{max} is defined by the user (Bellman, 1961; Fisher, 1958; Auger and Lawrence, 1989; Hawkins, 2001; Bai and Perron, 2003; Harchaoui and Cappé, 2007; Guédon, 2008; Arlot et al., 2012; Cleynen and Lebarbier, 2014; Guédon, 2013). To be more specific, the aim is to find the segmentations of minimal cost in $\mathscr{M}_{1:n+1}^{K}$ for K from 1 to K_{max}, where the cost of a segmentation is the sum of the costs of its segments:

$$R_m = \sum_{k=1}^{K+1} c_{\tau_{k-1}:\tau_k}, \tag{1}$$

with $c_{\tau_{k-1}:\tau_k}$ is the cost of the k-th segment of m.

A smaller class of problems In this paper, we consider a smaller class of models and methods for which it is possible to write the cost of a segment as follows:

$$c_{\tau_{k-1}:\tau_k} = \min_{\mu} \left\{ \sum_{t=\tau_{k-1}}^{\tau_k - 1} \gamma(Y_i, \mu) + g(\mu) \right\},$$

where γ is a loss function, depending on the data-point Y_i and the (possibly multi-dimensional) parameter μ, and the function g is a regularization penalty. For this class of models we show in the following that it is theoretically possible to prune the set of segmentations and we explain how to implement this pruning if μ is unidimensional.

The quadratic loss belongs to this class of model: $\gamma(Y_i, \mu) = (Y_i - \mu)^2$. This framework also includes maximum likelihood inference of identically distributed and independent data-points. Indeed, it suffices to take γ equal to minus the log-likelihood: $\gamma(Y_i, \mu) = -\log(p(Y_i, \mu))$. Let us give some examples.

1. We can consider a linear model. To do that we take $Y_i = (Z_i, x_{i1}, \ldots, x_{ip})$ in $\mathbb{R}^{p+1}$, $\mu = (\beta_0, \beta_1, \ldots, \beta_p, \sigma^2)$ in $\mathbb{R}^{p+1} \times \mathbb{R}^+$ and within a segment $\tau_{k-1} : \tau_k$ we assume that

   $$Z_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} + \varepsilon_i \quad \text{with} \quad \varepsilon_i \sim \mathscr{N}(0, \sigma^2) \quad \text{i.i.d.}$$

   In that case minus the log-likelihood is

   $$\gamma(Y_i, \mu) = \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}(Z_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2.$$

   If the variance is known this simplifies to $\gamma(Y_i, \mu) = (Z_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2$.

2. We can also consider the segmentation in the mean of a p-dimensional sequence. Here we take $Y_i = (X_{i1}, \ldots X_{ip})$ in $\mathbb{R}^p$, $\mu = (\beta_1, \ldots, \beta_p, \sigma^2)$ in $\mathbb{R}^p \times \mathbb{R}^+$ and for all i within a segment $\tau_{k-1} : \tau_k$ we assume that

   $$X_{ij} = \beta_j + \varepsilon_{ij} \quad \text{with} \quad \varepsilon_{ij} \sim \mathscr{N}(0, \sigma^2) \quad \text{i.i.d.}$$

   In that case minus the log-likelihood is

   $$\gamma(Y_i, \mu) = \frac{p}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\sum_{j=1}^{p}(X_{ij} - \beta_j)^2.$$

   If the variance is known this simplifies to $\gamma(Y_i, \mu) = \sum_{j=1}^{p}(X_{ij} - \beta_j)^2$.

 ISSN: 2102-6238

3. We could also consider categorical data. Indeed, we can take $Y_i \in \{1,\dots,p\}$, $\mu = (\pi_1,\dots,\pi_p)$ in $[0,1]^p$ with $\sum_j \pi_j = 1$ and assume that within a segment, Y_i are independent and follow a multinomial distribution with $P(Y_i = j) = \pi_j$. In that case minus the log-likelihood is

$$\gamma(Y_i,\mu) = -\sum_{j=1}^{p} \mathbb{I}\{Y_i = j\}\log(\pi_j),$$

where $\mathbb{I}$ is the indicator function.

Many segmentation models do not include a regularization penalty ($g(\mu) = 0$), however, in some cases it can be of interest to include a regularization penalty such as the ridge penalty. We could take this into account by setting $g(\mu) = \lambda\mu^2$. For simplicity, in the rest of this paper we will only consider the case $g(\mu) = 0$, however extension to $g(\mu) \neq 0$ is straightforward.

3. Exact segment neighbourhood algorithm and pruning

In this section, we first describe the main update rule of the standard dynamic programming algorithm - often called the segment neighbourhood algorithm - used to recover the segmentations of minimal cost for $K = 1$ to K_{max}:

$$\mathbf{Cost}^K_{1:n+1} = \min_{m \in \mathscr{M}^K_{1:n+1}} \{R_m\}, \tag{2}$$

Then we present the functional cost representation of a segmentation and informally explain how using this representation it is possible to prune the search space of the segment neighbourhood algorithm. In the last subsection (3.5) we explicitly describe the optimal partitioning problem and the advantages and disadvantages of functional pruning over inequality based pruning (as is done in PELT (Killick et al., 2012)).

3.1. The segment neighbourhood algorithm update rule

R_m is segment additive, thus the Bellman optimality principle applies and if a segmentation is optimal any sub-segmentation of this segmentation is also optimal. Mathematically, this can be expressed as the following update rule:

$$\mathbf{Cost}^K_{1:t} = \min_{\tau<t}\{\mathbf{Cost}^{K-1}_{1:\tau} + c_{\tau:t}\}, \tag{3}$$

This update rule can be performed with an $\mathscr{O}(t)$ time complexity. Many algorithms developed for various statistical models implement this particular update rule (Fisher, 1958; Bellman, 1961; Auger and Lawrence, 1989; Bai and Perron, 2003; Guédon, 2008). This algorithm was called the segment neighbourhood algorithm by Auger and Lawrence (1989). To recover $\mathbf{Cost}^K_{1:n+1}$ we need to apply update rule (3) for every t smaller than $n+1$ and for every K smaller than K_{max}. The overall time complexity is thus in $\mathscr{O}(K_{max}n^2)$. Most of these algorithms have an $\mathscr{O}(n^2)$ space complexity. However some, like Guédon (2008), have an $\mathscr{O}(K_{max}n)$ space complexity.

 ISSN: 2102-6238

3.2. *Functional cost*

The segment neighbourhood algorithm and in fact most algorithms and heuristics that aim at recovering the segmentation of minimal cost operate on segmentations through their costs, R_m, which are numbers in $\mathbb{R}$. Our pruned DPA algorithm is different and uses a slightly more complex representation that we call the functional cost of a segmentation. We define this function of μ as:

$$\widetilde{R}_m(\mu) = \sum_{k=1}^{K} c_{\tau_{k-1}:\tau_k} + \widetilde{c}_{\tau_K:\tau_{K+1}}(\mu) \tag{4}$$

$$\begin{aligned} \text{with } \widetilde{c}_{\tau_K:\tau_{K+1}}(\mu) = & \quad \textstyle\sum_{t=\tau_K}^{\tau_{K+1}-1} \gamma(Y_i,\mu) \quad \text{if } \tau_{K+1} > \tau_K \\ \text{and } \widetilde{c}_{\tau_K:\tau_{K+1}}(\mu) = & \quad 0 \qquad\qquad\quad \text{if } \tau_{K+1} = \tau_K \end{aligned}$$

In words, the functional cost, $\widetilde{R}_m(\mu)$, is the cost of segmentation m if the parameter of m's last segment is set to μ, rather than the optimal value $\hat{\mu} = \arg\min\{\widetilde{c}_{\tau_K:\tau_{K+1}}(\mu)\}$. The minimum value of the functional cost is simply the cost: *i.e.* $\min_\mu\{\widetilde{R}_m(\mu)\} = R_m$

The functional cost is a more complex representation of a segmentation than the cost, however, it leads to some great simplifications. Namely the functional cost is point additive, *i.e* if we consider a segmentation m of $\mathscr{M}^K_{1:t}$ with change-points $\tau_1, \tau_2, ..., \tau_K$ and the segmentation m' of $\mathscr{M}^K_{1:t+1}$ with the same change-points we have:

$$\widetilde{R}_{m'}(\mu) = \widetilde{R}_m(\mu) + \gamma(Y_t,\mu).$$

In the next sub-section (3.3) we explain informally how this functional formulation and the point additiveness makes it theoretically possible to prune the set of segmentations.

3.3. *Functional cost and pruning*

The pDPA searches for:

$$\widetilde{\mathbf{Cost}}^K_{1:n+1}(\mu) = \min_{m\in\mathscr{M}^K_{1:n+1}} \{\widetilde{R}_m(\mu)\}, \tag{5}$$

for $K = 1$ to K_{max}. This is the functional formulation of equation (2). Like the cost (see update rule (3)), the functional cost is segment additive and a similar update rule applies:

$$\widetilde{\mathbf{Cost}}^K_{1:t}(\mu) = \min_{\tau<t}\{\mathbf{Cost}^{K-1}_{1:\tau} + \widetilde{c}_{\tau:t}(\mu)\}. \tag{6}$$

Thanks to the point-additiveness of $\widetilde{R}_m(\mu)$, it is possible to further simplify this update rule. If we consider a given value of μ, two last change-points τ and τ' and a time $t' > t$ we get the following implication:

$$\left\{\mathbf{Cost}^{K-1}_{1:\tau} + \widetilde{c}_{\tau:t}(\mu) \le \mathbf{Cost}^{K-1}_{1:\tau'} + \widetilde{c}_{\tau':t}(\mu)\right\} \implies \left\{\mathbf{Cost}^{K-1}_{1:\tau} + \widetilde{c}_{\tau:t'}(\mu) \le \mathbf{Cost}^{K-1}_{1:\tau'} + \widetilde{c}_{\tau':t'}(\mu)\right\}. \tag{7}$$

 ISSN: 2102-6238

In other words, if for a given t and μ the functional cost of last change-point τ is less than the functional cost of τ', then this will still be the case for any time point in the future (*i.e.* for any $t' > t$). Hence for the parameter value μ we can discard τ' as it will never be minimal.

If we consider only one value for μ, this pruning rule can be formalised as the following update rule:

$$\widetilde{\mathbf{Cost}}^K_{1:t+1}(\mu) = \min\left\{\widetilde{\mathbf{Cost}}^K_{1:t}(\mu) + \gamma(Y_t,\mu)\ ,\ \mathbf{Cost}^{K-1}_{1:t} + \gamma(Y_t,\mu)\right\}, \tag{8}$$

and this update rule can be performed with an $\mathcal{O}(1)$ time complexity. We only need to compare the values of $\widetilde{\mathbf{Cost}}^K_{1:t}(\mu) + \gamma(Y_t,\mu)$ and $\mathbf{Cost}^{K-1}_{1:t} + \gamma(Y_t,\mu)$.

3.4. Implementing update rule (8) for all possible μ

Update rule (8) cannot be implemented in practice, because the rule would have to be applied to all possible values of μ and in most cases the set of all μ is uncountably infinite. A key idea to cope with this problem is to consider the set of μ for which a particular last change-point τ is better than any other change-point τ' in the sense that $\mathbf{Cost}^{K-1}_{1:\tau} + \widetilde{c}_{\tau:t}(\mu)$ is smaller than $\mathbf{Cost}^{K-1}_{1:\tau'} + \widetilde{c}_{\tau':t}(\mu)$. We will call this set $\mathscr{S}^K_{1:t,\tau}$. We will properly define and study the properties of $\mathscr{S}^K_{1:t,\tau}$ in section 4. As yet let us have a look at these sets on a small example. We will consider the segmentations with one change-point ($K = 1$) of a four-point signal with the quadratic loss: $\gamma(y_i,\mu) = (y_i - \mu)^2$. Our observations are $y_1 = 0,\ y_2 = 0.5,\ y_3 = 0.4,\ y_4 = -0.5$ (see Figure 1).

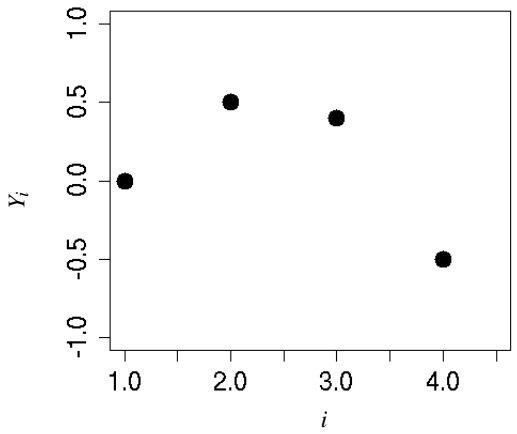
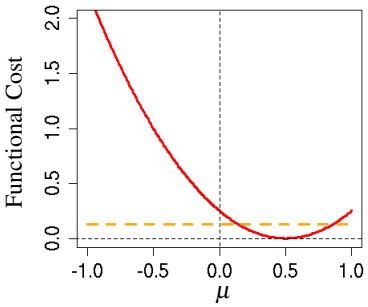


FIGURE 1. *Four-point signal. y_i as a function of i.* $y_1 = 0,\ y_2 = 0.5,\ y_3 = 0.4,\ y_4 = -0.5$

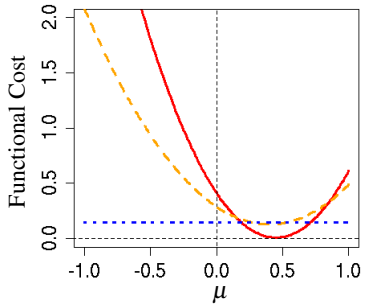
Functional cost of $Y_{1:3}$ Let us first consider the first two data-points: $Y_{1:3}$. The first segmentation we need to consider is the one with a change-point at $\tau = 2$. The functional cost of this segmentation is simply the cost of its first segment, $r_{1:2}$, which is 0, plus the functional cost of the last segment, which is the polynomial function $\mu \to (y_2 - \mu)^2$. We should also consider a segmentation with a change-point at $\tau = 3$. Indeed, for $t = 2$ its functional cost is well defined: it is the cost of its first segment, $r_{1:3}$, which is 0.125, plus the functional cost of its last segment, $r_{3:3}$.

 ISSN: 2102-6238



τ	Functional cost	$\mathscr{S}^1_{1:3,\tau}$
2	$0.25-\mu+\mu^2$	$[\,0.146\,,\,0.854\,]$
3	0.125	$[\,-\infty,0.146\,]\cup[\,0.854,+\infty\,]$

FIGURE 2. *Functional cost of $Y_{1:3}$ for $K=1$ using the quadratic loss. (Left) Functional cost as a function of μ of segmentations having a change-point at $\tau=2$ (solid red) and $\tau=3$ (orange dashed). (Right) Analytical expression of the functional costs for $\tau=2$ and $\tau=3$ and the set of μ, for which they are optimal: $\mathscr{S}^1_{1:3,\tau}$.*



τ	Functional cost	$\mathscr{S}^1_{1:4,\tau}$
2	$0.41-1.8\mu+2\mu^2$	$[\,0.190\,,\,0.709\,]$
3	$0.285-0.8\mu+\mu^2$	$\emptyset$
4	0.14	$[\,-\infty,0.190\,]\cup[\,0.709,+\infty\,]$

FIGURE 3. *Functional cost of $Y_{1:4}$ for $K=1$ using the quadratic loss. (Left) Functional cost of a segmentations having a change-point at $\tau=2$ (solid red) $\tau=3$ (orange dashed) and $\tau=4$ (blue dotted). (Right) Analytical expression of the functional costs for $\tau=2$, 3 and 4 and the set of μ, for which they are optimal: $\mathscr{S}^1_{1:4,\tau}$.*

$r_{3:3}$ is empty and so its functional cost is simply the zero function: $\mu\to 0$. These two functional costs are represented in figure 2-left as a solid red line for $\tau=2$ and a dashed orange line for $\tau=3$. Simple calculations show that the set of μ for which $\tau=2$ is best, $\mathscr{S}^1_{1:3,2}$, is an interval centered on 0.5: $[\,0.146\,,\,0.854\,]$. $\mathscr{S}^1_{1:3,3}$ is a union of two intervals: $[\,-\infty,0.146\,]\cup[\,0.854,+\infty\,]$. These intervals are also given in figure 2-right.

Functional cost of $Y_{1:4}$ Now if we consider the first three data-points: $Y_{1:4}$. We have to update the functional cost of the segmentations having a change-point at $\tau=2$ and 3. We do this by adding the function $\mu\to(y_3-\mu)^2$ to both of them (point additiveness). We also need to consider another possible change-point: $\tau=4$. Its functional cost is simply the cost of segment $r_{1:4}$, which is 0.14, plus the function cost of segment $r_{4:4}$ which is the zero function: $\mu\to 0$. These three functional costs are represented in figure 3-left as a solid red line for $\tau=2$, a dashed orange line for $\tau=3$ and a blue dotted line for $\tau=4$. Simple calculations show that the set of μ for which $\tau=2$ is best, $\mathscr{S}^1_{1:4,2}$, is an interval centered on 0.45: $[\,0.190\,,\,709\,]$. $\mathscr{S}^1_{1:4,4}$ is a union

 ISSN: 2102-6238

of two intervals: $[\ -\infty, 0.190\ ] \cup [\ 0.709, +\infty\ ]$. $\mathscr{S}^1_{1:4,3}$ is strikingly empty. Based on equation (7), this means that irrespective of the last data-point it is sure that the segmentation with a last change-point at $\tau = 3$ will always have a greater cost that those with a change-point at 2 or 4. Thus for all possible values of μ we can discard $\tau = 3$. Note however that neither $\tau = 2$ nor $\tau = 4$ are better than $\tau = 3$ for all μ.

This simple example illustrates that by keeping track of the $\mathscr{S}^K_{1:t,\tau}$ we might be able to prune some candidate change-points very early in the process. We will make this idea explicit in the next sections (section 4 and 5).

3.5. Segment Neighbourhood, Optimal partitioning and PELT

Since 2010 another pruned algorithm, PELT, has been proposed for change-point detection problems by Killick et al. (2012). pDPA and PELT are different in nature. First, they do not solve the same problem. PELT is an extension of the optimal partitioning algorithm (Jackson et al., 2005) rather than the segment neighbourhood algorithm. More precisely, if we call $\mathscr{M}_{1:n+1}$ the set of all segmentations, the optimal partitioning problem is to find:

$$\mathbf{OPCost}_{1:n+1} = \min_{m \in \mathscr{M}_{1:n+1}} \{R_m + \lambda |m|\}, \tag{9}$$

where R_m is defined in equation (1), λ is a user defined scalar and $|m|$ is the number of change-points of segmentation m. Intuitively, the number of change-points of the recovered solution decreases with λ. If the penalty λ is known in advance solving the optimal partitioning problem is faster than solving the segment neighbourhood problem. The advantage of solving the segment neighbourhood problem is that this gives optimal segmentations for a range of numbers of change-points.

Second PELT and the pDPA do not prune segmentations in the same way. PELT's prune segmentations based on their cost and the pDPA based on their functional cost. In a recent pre-print, Maidstone et al. (2014) called these two ways of pruning respectively inequality based pruning (IP) and functional based pruning (FP) and studied their relationship. The benefit of IP over FP is that it is more widely applicable than FP, i.e. the assumptions on the segment cost to apply FP imply the assumptions to apply IP. Furthermore the implementation of IP is usually straightforward which is not the case of FP. The advantage of FP over IP is that it can prune efficiently the set of change-points even if there are only a few change-points in the sequence. We prove this for a special loss function and under some restrictive conditions in sub-section 5.2. We empirically confirm this result for the quadratic loss in sub-section 5.3. Furthemore, Maidstone et al. (2014) proved in Theorem 6.1 for the optimal partitioning and segment neighbourhood problem that any candidate change-point pruned by IP will also be pruned by FP at worst at the same time. In other words FP cannot prune less than IP.

4. Functional pruning of the set of segmentations

In this section we first describe the key quantities and properties used by the pruned DPA algorithm. Then we describe the algorithm.

 ISSN: 2102-6238

4.1. Key quantities and their properties

The four main quantities of interest in the pDPA are given below.

1. the optimal functional cost with K change-points:
$$\widetilde{\mathbf{Cost}}^K_{1:n+1}(\mu) = \min_{m \in \mathscr{M}^K_{1:n+1}} \{\widetilde{R}_m(\mu)\},$$
where the functional cost of a segmentation $\widetilde{R}_m(\mu)$ is defined in equation (4).
2. the optimal functional cost with K change-points if the last change-point is $\tau \leq t$:
$$\widetilde{C}^K_{1:t,\tau}(\mu) = \mathbf{Cost}^{K-1}_{1:\tau} + \widetilde{c}_{\tau:t}(\mu), \tag{10}$$
where $\mathbf{Cost}^{K-1}_{1:\tau}$ is the optimal cost with $K-1$ change-points up to τ defined in equation (2) and $\widetilde{c}_{\tau:t}(\mu)$ is defined in equation (4).
3. the set of μ for which the last change-point τ is optimal:
$$\mathscr{S}^K_{1:t,\tau} = \{\mu \mid \widetilde{C}^K_{1:t,\tau}(\mu) = \widetilde{\mathbf{Cost}}^K_{1:t}(\mu)\}.$$
4. the set of last change-points which are optimal for at least one μ [1]
$$\tau^K_{1:t} = \left\{\tau < t \mid \mathscr{S}^K_{1:t,\tau} \neq \emptyset\right\}$$

Below are the properties of these quantities. The proofs of these properties are given afterwards.

Property 1. *The optimal functional cost may be computed using only the (pruned) set of change-points which are optimal for at least one μ.*

$$\begin{aligned}\widetilde{\mathbf{Cost}}^K_{1:t}(\mu) = & \min_{\tau<t}\{\widetilde{C}^K_{1:t,\tau}(\mu)\} \\ = & \min_{\tau \in \tau^K_{1:t}}\{\widetilde{C}^K_{1:t,\tau}(\mu)\}.\end{aligned}$$

Property 2. *The functional cost with a last change-point τ is easy to update using point additivity.*

$$\widetilde{C}^K_{1:t+1,\tau}(\mu) = \widetilde{C}^K_{1:t,\tau}(\mu) + \gamma(Y_t, \mu).$$

Property 3. *The set of μ for which a change-point τ is optimal decreases with t (w.r.t. set inclusion) and can be pruned.*

$$\forall\, \tau < t, \quad \mathscr{S}^K_{1:t+1,\tau} = \mathscr{S}^K_{1:t,\tau} \bigcap \{\mu \mid \widetilde{C}^K_{1:t,\tau}(\mu) \leq \mathbf{Cost}^{K-1}_{1:t}\} \tag{11}$$

$$\mathscr{S}^K_{1:t+1,t} = \bigcap_{\tau \in \tau^K_{1:t}} \{\mu \mid \mathbf{Cost}^{K-1}_{1:t} \leq \widetilde{C}^K_{1:t,\tau}(\mu)\} \tag{12}$$

$$\mathscr{S}^K_{1:t,\tau} = \emptyset \quad \Longrightarrow \quad \forall\, t' \geq t,\ \mathscr{S}^K_{1:t',\tau} = \emptyset. \tag{13}$$

[1] Depending on the properties of γ it might be possible to consider only the set of last change-points τ for which $\mathscr{S}^K_{1:t,\tau}$ is not restricted to a finite set of μ. This is the case for the quadratic loss which is continuous.

 ISSN: 2102-6238

Property 4. *The set of candidate change-points can be pruned and computed recursively.*

$$\tau^K_{1:t+1} = \quad \{\tau \in (\tau^K_{1:t} \cup \{t\}) \mid \mathscr{S}^K_{1:t+1,\tau} \neq \emptyset\}.$$

Property 1 is very similar to the update rule of the segmentation neighbourhood algorithm (equation (3)). The main difference is that the set of last change-points to consider is not necessarily all those before t: $\{\tau|\tau < t\}$ but rather those that still have a chance to be optimal at this stage: $\tau^K_{1:t}$. At worst this $\tau^K_{1:t}$ is equal to $\{\tau|\tau < t\}$ but based on property 4 this set could be smaller. Indeed any change-point that has been discarded from $\tau^K_{1:t}$ will never be included again.

The four previous properties are in fact simple consequences of the point additiveness and segment additiveness of the functional cost. However given their combinatorial nature they might look a bit tedious. To clarify those properties we provide detailed proofs in the following paragraphs.

Proof of property 1 As the cost in the segment neighbourhood algorithm, the functional cost of a segmentation is segment additive, thus the Bellman optimality principle holds and we recover the first part of property 1. The second part follows by definition of $\tau^K_{1:t}$. Indeed, if τ is the optimal last change-point then there must exist a μ for which its functional cost is equal to $\widetilde{\mathbf{Cost}}^K_{1:t}(\mu)$ and thus τ is in $\tau^K_{1:t}$ ■

Proof of property 2 By definition of $\widetilde{C}^K_{1:t,\tau}(\mu)$ (see equation (10)) we have:

$$\begin{aligned}
\widetilde{C}^K_{1:t+1,\tau}(\mu) = &\ \mathbf{Cost}^{K-1}_{1:\tau} + \textstyle\sum_{i=\tau}^{t} \gamma(Y_i,\mu) \\
= &\ \mathbf{Cost}^{K-1}_{1:\tau} + \textstyle\sum_{i=\tau}^{t-1} \gamma(Y_i,\mu) \quad + \ \gamma(Y_t,\mu) \\
= &\ \widetilde{C}^K_{1:t,\tau}(\mu) + \ \gamma(Y_t,\mu) \qquad \blacksquare
\end{aligned}$$

Proof of property 3 Using property 1 and by definition of $\mathscr{S}^K_{1:t+1,\tau'}$ we get that for $\tau' < t+1$:

$$\begin{aligned}
\mathscr{S}^K_{1:t+1,\tau'} = &\ \left\{\mu \mid \widetilde{C}^K_{1:t+1,\tau'}(\mu) \leq \min_{\tau<t+1}\{\widetilde{C}^K_{1:t+1,\tau}(\mu)\}\right\} \\
= &\ \bigcap_{\tau<t+1} \left\{\mu \mid \widetilde{C}^K_{1:t+1,\tau'}(\mu) \leq \widetilde{C}^K_{1:t+1,\tau}(\mu)\right\}.
\end{aligned}$$

Now, using property 2 we also have for $\tau < t+1$:

$$\{\mu \mid \widetilde{C}^K_{1:t+1,\tau'}(\mu) \leq \widetilde{C}^K_{1:t+1,\tau}(\mu)\} = \{\mu \mid \widetilde{C}^K_{1:t,\tau'}(\mu) \leq \widetilde{C}^K_{1:t,\tau}(\mu)\}.$$

Finally, by defintion we also have $\widetilde{C}^K_{1:t,t}(\mu) = \mathbf{Cost}^{K-1}_{1:t}$.

Combining these three facts for $\tau' < t$ we get:

$$\begin{aligned}
\mathscr{S}^K_{1:t+1,\tau'} = &\ \bigcap_{\tau<t+1} \left\{\mu \mid \widetilde{C}^K_{1:t,\tau'}(\mu) \leq \widetilde{C}^K_{1:t,\tau}(\mu)\right\} \\
= &\ \mathscr{S}^K_{1:t,\tau'} \bigcap \left\{\mu \mid \widetilde{C}^K_{1:t,\tau'}(\mu) \leq \widetilde{C}^K_{1:t,t}(\mu)\right\} \\
= &\ \mathscr{S}^K_{1:t,\tau'} \bigcap \left\{\mu \mid \widetilde{C}^K_{1:t,\tau'}(\mu) \leq \mathbf{Cost}^{K-1}_{1:t}\right\}
\end{aligned}$$

 ISSN: 2102-6238

and we recover equation (11) of property 3.

We recover equation (12) of property 3 by taking $\tau' = t$:

$$\begin{aligned}
\mathscr{S}^K_{1:t+1,t} &= \bigcap_{\tau<t} \left\{ \mu \mid \mathbf{Cost}^{K-1}_{1:t} \leq \widetilde{C}^K_{1:t,\tau}(\mu) \right\} \\
&= \left\{ \mu \mid \mathbf{Cost}^{K-1}_{1:t} \leq \min_{\tau<t} \{\widetilde{C}^K_{1:t,\tau}(\mu)\} \right\} \\
&= \left\{ \mu \mid \mathbf{Cost}^{K-1}_{1:t} \leq \min_{\tau\in\tau^K_{1:t}} \{\widetilde{C}^K_{1:t,\tau}(\mu)\} \right\} \\
&= \bigcap_{\tau\in\tau^K_{1:t}} \left\{ \mu \mid \mathbf{Cost}^{K-1}_{1:t} \leq \widetilde{C}^K_{1:t,\tau}(\mu) \right\}.
\end{aligned}$$

Finally, if $\mathscr{S}^K_{1:t,\tau} = \emptyset$ using equation (11) we have that

$$\mathscr{S}^K_{1:t+1,\tau} = \mathscr{S}^K_{1:t,\tau} \bigcap \{\mu \mid \widetilde{C}^K_{1:t,\tau}(\mu) \leq \mathbf{Cost}^{K-1}_{1:t}\} = \emptyset$$

and by induction we recover equation (13) of property 3 ■

Proof of property 4 By definition of $\tau^K_{1:t+1}$ we have

$$\tau^K_{1:t+1} \quad \supseteq \quad \{\tau \in (\tau^K_{1:t} \cup \{t\}) \mid \mathscr{S}^K_{1:t+1,\tau} \neq \emptyset\}.$$

Now suppose τ is in $\tau^K_{1:t+1}$. If $\tau < t$, then using equation (13) of property 3 we see that τ must also be in $\tau^K_{1:t}$. If $\tau = t$, then by definition of $\tau^K_{1:t+1}$ we have that $\mathscr{S}^K_{1:t+1,t}$ is not empty. Thus we recover:

$$\tau^K_{1:t+1} \quad \subseteq \quad \{\tau \in (\tau^K_{1:t} \cup \{t\}) \mid \mathscr{S}^K_{1:t+1,\tau} \neq \emptyset\} \blacksquare$$

4.2. The pDPA algorithm

The pruned dynamic programming algorithm uses the four properties described in subsection 4.1 to update the optimal functional cost, the set of optimal last change-points and the set of μ for which these last change-points are optimal. For every $t \leq n$ and $K \leq K_{max}$ the algorithm:

1. updates for each τ in $\tau^K_{1:t}$ the set of μ for which they are optimal (equation (11));
2. initialises the set of values for which a change-point at t is optimal (equation (12));
3. updates the set of possible last change-points (property 4);
4. updates the functional cost of possible last change-points (property 2);
5. computes the minimal cost with K change-points at t (property 1).

 ISSN: 2102-6238

More formally the algorithm can be described as:

Data: A sequence $Y_{1:n+1}$
Result: Two matrices $D_{K,t}$ and $I_{K,t}$ of size $(n+1) \times K_{max}$ containing the optimal cost and optimal last change-points
for $K = 1$ *to* K_{max} **do**
  $\tau^K_{1:K+1} \leftarrow \{K\}$
  $\widetilde{C}^K_{1:K+1,K}(\mu) \leftarrow \mathbf{Cost}^{K-1}_{1:K} + \gamma(Y_K, \mu)$
  for $t = K+1$ *to* n **do**
    for $\tau \in \tau^K_{1:t}$ **do**
      $\mathscr{S}^K_{1:t+1,\tau} \leftarrow \mathscr{S}^K_{1:t,\tau} \quad \bigcap \quad \{\mu \mid \widetilde{C}^K_{1:t,\tau}(\mu) \leq \mathbf{Cost}^{K-1}_{1:t}\}$
    end
    $\mathscr{S}^K_{1:t+1,t} \leftarrow \quad \bigcap_{\tau<t} \{\mu \mid \mathbf{Cost}^{K-1}_{1:t} \leq \widetilde{C}^K_{1:t,\tau}(\mu)\}$
    $\tau^K_{1:t+1} \leftarrow \quad \{\tau \in (\tau^K_{1:t} \cup \{t\}) \mid \mathscr{S}^K_{1:t+1,\tau} \neq \emptyset\}$
    for $\tau \in \tau^K_{1:t+1}$ **do**
      $\widetilde{C}^K_{1:t+1,\tau}(\mu) \leftarrow \quad \widetilde{C}^K_{1:t,\tau}(\mu) + \gamma(Y_t, \mu)$
    end
    $D_{K,t+1} \leftarrow \quad \min_{\tau \in \tau^K_{1:t+1}} \left\{ \min_\mu \{\widetilde{C}^K_{1:t+1,\tau}(\mu)\} \right\}$
    $I_{K,t+1} \leftarrow \quad \arg\min_{\tau \in \tau^K_{1:t+1}} \left\{ \min_\mu \{\widetilde{C}^K_{1:t+1,\tau}(\mu)\} \right\}$
  end
end

Algorithm 1: Pruned DPA algorithm

Importantly $\widetilde{\mathbf{Cost}}^K_{1:t}(\mu)$, $\mathscr{S}^K_{1:t,t}$ and $\tau^K_{1:t}$ are used only at step K,t and $K,t+1$. So they need not be stored, they can be discarded or overwritten immediately.

Implementing the pruned DPA The pruned DPA critically relies on the possibility of easily updating the functional cost ($\mu \to \widetilde{C}^K_{1:t,\tau}(\mu)$) and the set of μ for which one particular last change-point is optimal ($\mathscr{S}^K_{1:t,\tau}$). If this is not the case then the algorithm and the fact that the set of last change-points can be functionally pruned is purely theoretical.

Representing and updating the functional cost is easy typically if there exists a simple analytical decomposition of the functional cost. For example, if we consider the quadratic loss, the functional cost is a second degree polynomial function and it can be represented by the three coefficients of this polynomial function. Other loss functions can be represented in such a way, and in fact the pruned DPA has been implemented for other losses since its first pre-print (namely the Poisson and negative binomial log-likelihood losses, Cleynen et al. (2014b))

Representing the sets $\mathscr{S}^K_{1:t,\tau}$ is a priori more difficult. Their representation depends on the dimensionality of μ and the characteristic of the loss function γ. In the next section we study theoretically and empirically the expected and worst case complexity of functional pruning in the simple case where μ is a unidimensional parameter in $\mathbb{R}$ and all functions $\mu \to \sum \gamma(Y_i, \mu)$ are

 ISSN: 2102-6238

unimodals. Here by unimodal we mean functions such that for any c in $\mathbb{R}$ the set $\{x|f(x) \leq c\}$ is an interval. In that case it is simpler to keep track of the sets $\{\mu \mid \mathbf{Cost}_{1:t}^{K-1} \leq \widetilde{C}_{1:t,\tau}^{K}(\mu)\}$ and $\mathscr{S}_{1:t,\tau}^{K}$ as they are respectively intervals and union of intervals. Functional pruning is theoretically possible even if those conditions are not valid and in particular if μ is multidimensional. However the implementation and the theoretical study of functional pruning in a multidimensional case is not straightforward and is outside the scope of this paper.

5. Complexity of the algorithm for a unidimensional μ

In this section we study the complexity of the pruned DPA for μ in $\mathbb{R}$. We also assume that updating the functional cost is in $\mathscr{O}(1)$, which is the case if there is a simple analytical decomposition of the functional cost. Finally we will assume that any function $\mu \to \sum_i \gamma(Y_i, \mu)$ is a unimodal function of μ. This last assumption is true if γ is convex, which is often the case if we take γ to be minus the log-likelihood. Under those conditions all $\mathscr{S}_{1:t,\tau}^{K}$ are unions of intervals [2].

A key question then is how many intervals do we precisely need. In this section, we propose a bound of this number and using it we bound the worst case complexity of the pDPA and show that it is at worst as efficient as the segment neighbourhood algorithm (see subsection 5.1). Then we theoretically study the average complexity of the pDPA algorithm for a particular loss function for a random sequence without change-points (see subsection 5.2). Finally, we empirically study the complexity of the algorithm for the quadratic loss (see subsection 5.3).

5.1. *Worst case complexity of the pDPA*

In this section we prove that the pDPA is at worst as efficient as the segment neighbourhood algorithm. More precisely we have the following proposition.

Propostion 5. *If all $\sum_j \gamma(Y_j, \mu)$ are unimodal in μ and if both minimising $\widetilde{C}_{1:t,\tau}^{K}(\mu)$ and finding the roots of $\widetilde{C}_{1:t,\tau}^{K}(\mu) = \mathbf{Cost}_{1:t}^{K-1}$ are in $\mathscr{O}(1)$, the pDPA is at worst in $\mathscr{O}(K_{max}n^2)$ time and in $\mathscr{O}(K_{max}n)$ space.*

Proof. The key quantity to control is the number of intervals needed to represent $\mathscr{S}_{1:t,\tau}^{K}$. For a given K and at step t the number of candidate last change-points is obviously bounded by t. If all $\sum_{j=\tau+1}^{t+1} \gamma(Y_j, \mu)$ are unimodal, using theorem 8 (proved in appendix A) we get that the total number of intervals is bounded by $2t-1$. Thus at each step there is at most t last change-points and $2t-1$ intervals to update. By summing all these bounds from 1 to n and for every possible K we retrieve an $\mathscr{O}(K_{max}n^2)$ worst case time complexity.
As for the worst case space complexity, we need to store two $(n+1) \times K_{max}$ matrices ($D_{K,t}$ and $I_{K,t}$) and at each step there is at most t candidates and $2t-1$ intervals. This gives an $\mathscr{O}(K_{max}n)$ space complexity ■

[2] Indeed, if all $\mu \to \sum_i \gamma(Y_i, \mu)$ are unimodal functions then all $\{\mu \mid \widetilde{C}_{1:t,\tau}^{K}(\mu) \leq \mathbf{Cost}_{1:t}^{K-1}\}$ are intervals as $\mathbf{Cost}_{1:t}^{K-1}$ is a constant in $\mathbb{R}$ and all $\widetilde{C}_{1:t,\tau}^{K}(\mu)$ are unimodal. Then using property 3 we get by induction that all $\mathscr{S}_{1:t,\tau}^{K}$ are unions of intervals.

 ISSN: 2102-6238

The previous theorem provides a worst case bound on the complexity. One can wonder in which cases this quadratic complexity in n is reached. Let us consider the sequence such that $\forall i,\ Y_i = i$, segmentations with one change-point and the quadratic loss. The best segmentation of $Y_{1:t}$ has a change-point at $t/2$. So at step t of the pDPA the change-point $\tau = t/2$ has not been pruned and in fact all change-points from $t/2$ to t haven't been pruned. Hence, at each step we have at least $t/2$ candidates to update. Hence by summing from $t = 1$ to n we recover a quadratic complexity: $\mathcal{O}(n^2)$.

However, as illustrated with an example in subsection 3.4 and demonstrated for a specific loss function in the following subsection (5.2) one can hope that in many cases the pruning will be more efficient than that.

5.2. *Average pruning of the pDPA with a special loss function and* $K_{max} = 1$

In this subsection, we prove for a particular loss function and $K_{max} = 1$ that on average we get an efficient pruning with the functional cost representation even when we consider a random sequence without true change-points. As explained in the introduction this result is not intuitive. In subsection 5.3 we empirically show that this is also the case for the quadratic loss function.

Here, we will use the negative log-likelihood loss of a continuous uniform distribution defined on $[0, \mu]$. This loss function is:

$$\begin{cases} \gamma(Y_i, \mu) = \log(\mu) & \text{if } 0 \leq Y_i \leq \mu \\ \gamma(Y_i, \mu) = \infty & \text{otherwise} \end{cases}$$

For this particular loss function [3] and for $K_{max} = 1$, it is possible to bound the average number of candidate last change-points, $E(|\tau^K_{1:t}|)$.

Property 6. *For the negative log-likelihood loss,* $K_{max} = 1$*, and for,* $Y_{1:n+1}$*,* n *independent and identically distributed random variables of density* f *and continuous distribution* F*,* $E(|\tau^1_{1:n}|) = \mathcal{O}(\log(n))$ *and the average time complexity of the pDPA is in* $\mathcal{O}(n\log(n))$*.*

Proof The proof of $E(|\tau^1_{1:t}|) = \mathcal{O}(\log(t))$ is given in appendix B. We obtain this result by studying the set $\mathcal{S}^1_{1:n,\tau}$. More precisely we characterize some simple events for which $\mathcal{S}^1_{1:n,\tau}$ is empty and compute the probability of these events. Then by taking the expectation and summing over all possible τ we get the desired result.

For the complexity using theorem 8 we know that the number of intervals stored by the pruned DPA is always smaller than 2 times the number of candidate change-points. Thus for $K_{max} = 1$, for every $t \leq n$ the pruned DPA updates on average $\mathcal{O}(\log(t))$ functional costs and intervals. From this the complexity follows ■

5.3. *Empirical complexity of the pDPA*

In this section, we empirically assess the efficiency of the pDPA to analyze both simulated and real data in the case of the quadratic loss, $\gamma(Y_i, \mu) = (Y_i - \mu)^2$. The pDPA was implemented in C++

[3] note that any $\mu \to \sum_i \gamma(Y_i, \mu)$ is unimodal.

 ISSN: 2102-6238

and was run on a Intel Core i7-3687U3 2.10 GHz. The code is available in the cghseg package on the CRAN at the following webpage (http://cran.r-project.org/web/packages/cghseg/index.html). Here is an example code using this function:

```
install.packages("cghseg")
library(cghseg)
Kmax <- 40; n <- 10^5;
y <- rnorm(n)
system.time(res_ <- cghseg:::segmeanCO(y, Kmax))
```

5.3.1. *Synthetic data*

We simulated a number of sequences with a constant, a sinusoid or a rectangular wave signal. We considered an additional gaussian noise of variance 1, a uniform noise and a Cauchy noise. We also considered the worst case scenario for the pDPA which is, as explained at the end of section 5.1, achieved for $y_i = i$. We compared the pDPA with the segment neighbourhood algorithm for $n \leq 81\,920$. For all these tests, we set $K_{max} = 40$. We then ran the pDPA alone for larger sequences $n \leq 2\,621\,440$. The R code to ran the pDPA on these simulations is given in appendix C.

Figures 4-A and B show that for all simulated sequences (coloured dotted and dashed lines) except the worst case scenario (black dashed and crossed line) the pDPA is faster than segment neighbourhood (black solid line). It took roughly 150 seconds for the segment neighbourhood algorithm to process a sequence of 81 920 data-points. In the worst case scenario the pDPA was able to process 40 960 data-points in the same amount of time, and for other simulated signals the pDPA was able to process more than 2.6 million data-points in the same amount of time. The runtime of the pDPA depends on the nature of the noise, for example it is faster for a Cauchy noise (green dotted lines in figure 4-A, B and green box-plots in figure 4-C).

5.3.2. *Real data*

We download the publicly available *GSE*17359 project from Gene Expression Omnibus (GEO: http://www.ncbi.nlm.nih.gov/geo/). This data set is made of SNP (Single Nucleotide Polymorphism) array experiments. SNP arrays enable the study of DNA copy number gains and losses along the genome. For this kind of sequence, a multiple change-point model based on the quadratic loss is often used (Picard et al., 2005). This model has been shown to reach state of the art performances (Lai et al., 2005; Hocking et al., 2013).

Each SNP array experiment can be viewed as a sequence of 1.8 million data-points. We assessed the runtime of the pDPA to process these sequences from data-point 1 to t for various values of t and again with $K_{max} = 40$. The R code is given in appendix D. Runtimes are reported in Figure 4-D. The runtime to process the 1.8 million data-points was on average 28 seconds and was always smaller than 33 seconds. Note that the runtime performances of the pDPA on such SNP array datasets have been confirmed by Hocking et al. (2013).

 ISSN: 2102-6238

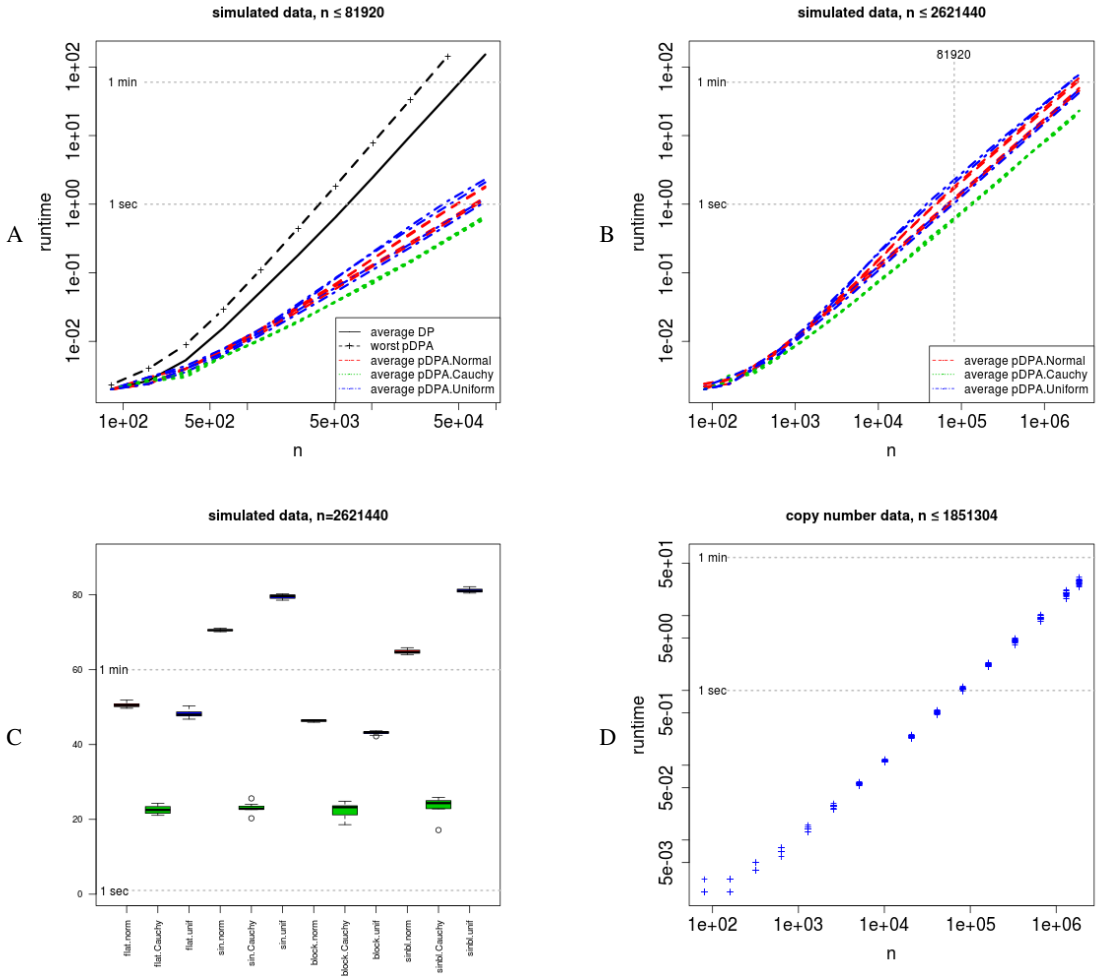


FIGURE 4. *Runtimes as a function of n in log-scale. A: Mean runtimes in seconds of the segment neighbourhood algorithm (black solid line) and pDPA (dotted, dashed, and dashed dotted line) for sequences of less than* 81 920 *data-points. The black dashed and crossed line is the runtime of the pDPA in the worst case scenario. Coloured dashed and dotted lines correspond to sequences simulated with or without sine or block waves plus an additional normal (red), Cauchy (green) or uniform noise (blue) (see appendix C). B: Same as A for sequences of less than* 2.6 *million data-points. C: Boxplot of runtimes (in seconds) of the pDPA to process simulated sequences of size* $n = 2621440$. *D: Runtimes (in seconds) of the pDPA to process the sequences of the GSE*17359 *dataset from data-point* 1 *to t.*

Number of intervals We also directly assess the efficiency of the pruning by counting the number of intervals stored by the pDPA at every time step of the algorithm. Figure 5-A, B and C show, for sequences with a constant or sine wave signal plus an additional gaussian noise and for sequences of the *GSE*17359 project, the limited number of intervals stored by the pDPA. Indeed, for all these sequences we observed less than 50 intervals. If there was no pruning we would expect at least $n = 1.8$ million and at worst $2n - 1$ intervals (see subsection 5.1).

 ISSN: 2102-6238

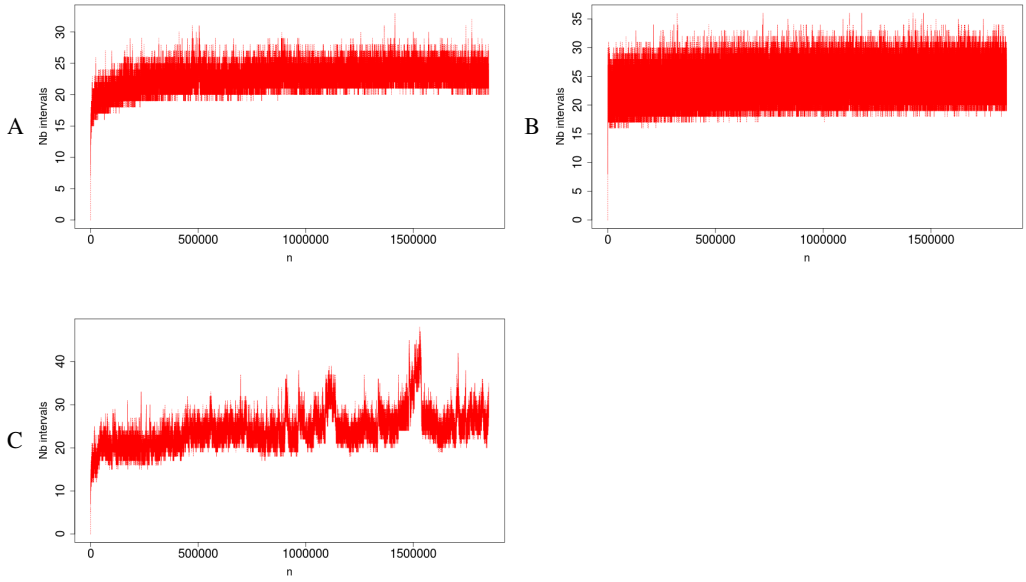


FIGURE 5. *Maximum number of intervals stored by the pDPA at each point of the sequence for* $K = 1$. *A: For 100 sequences of* $1.8\ 10^6$ *points simulated with a constant signal plus an additional normal noise of variance 1. B: For 100 sequences of* $1.8\ 10^6$ *points simulated with a sine wave signal plus an additional normal noise of variance 1. C: For the 18 profiles of length* $1.8.\ 10^6$ *of the GSE*17359 *dataset.*

6. Conclusion

This paper presented the pDPA that recovers exactly the best segmentations with 1 to K_{max} change-points of an n point sequence w.r.t a loss function. This algorithm is based on a functional representation of the cost of segmentations. For loss functions with a unidimensional parameter per segment the worst case complexity is in $\mathcal{O}(K_{max}n^2)$, thus the pDPA is at worst equivalent to segment neighbourhood algorithms. For a special loss function it is proven that the pDPA allows to prune the set of candidate change-points efficiently even when there are no change-points in the signal. Finally, in the case of the quadratic loss, it was shown empirically that the pDPA has a small runtime compared to segment neighbourhood algorithms.

Future work Here we theoretically proved and empirically assessed that functional pruning is efficient under a restrictive set of assumptions. It would be interesting to more generally characterise what are the conditions for functional pruning to be efficient. In particular functional pruning is theoretically possible for a p dimensional parameter. Recently, Maidstone et al. (2014) proved that functional pruning prunes at least as much as inequality based pruning. Based on this, it follows that if the conditions of Theorem 3.2 of Killick et al. (2012) are met then functional pruning with optimal partitioning (Fpop) is efficient. However whether functional pruning is on average efficient when p is large and the number of change-points is fixed is to our knowledge an open question.

Another question is how to implement functional pruning in the multidimensional case. Indeed

 ISSN: 2102-6238

this would require an efficient way to represent and update the set of parameter values for which a particular change is optimal. To the best of our knowledge these sets do not have particularly simple properties and are not necessarily easy to handle, except maybe for some special loss functions such as the ℓ_∞ loss.

Acknowledgement I would like to thank Alice Cleynen, Michel Koskas, Robert Maidstone, Toby Hocking, Paul Fearnhead, Emilie Lebarbier, Stéphane Robin for fruitful discussions that helped me to better understand the pDPA. Special thanks to Toby, Alban and Patrick.

References

Arlot, S., Celisse, A., and Harchaoui, Z. (2012). Kernel change-point detection. *arXiv preprint arXiv:1202.3878*.

Auger, I. E. and Lawrence, C. E. (1989). Algorithms for the optimal identification of segment neighborhoods. *Bulletin of mathematical biology*, 51(1):39–54.

Bai, J. and Perron, P. (2003). Computation and analysis of multiple structural change models. *Journal of Applied Econometrics*, 18(1):1–22.

Bellman, R. (1961). On the approximation of curves by line segments using dynamic programming. *Communications of the ACM*, 4(6):284.

Birgé, L. and Massart, P. (2007). Minimal penalties for Gaussian model selection. *Probability theory and related fields*, 138(1-2):33–73.

Boysen, L., Kempe, A., Liebscher, V., Munk, A., and Wittich, O. (2009). Consistencies and rates of convergence of jump-penalized least squares estimators. *The Annals of Statistics*, pages 157–183.

Cleynen, A., Dudoit, S., and Robin, S. (2014a). Comparing segmentation methods for genome annotation based on rna-seq data. *Journal of Agricultural, Biological, and Environmental Statistics*, 19(1):101–118.

Cleynen, A., Koskas, M., Lebarbier, E., Rigaill, G., and Robin, S. (2014b). Segmentor3isback: an r package for the fast and exact segmentation of seq-data. *Algorithms for Molecular Biology*, 9:6.

Cleynen, A. and Lebarbier, E. (2014). Segmentation of the Poisson and negative binomial rate models: a penalized estimator. *ESAIM, P&S*, 18:750–769.

Fisher, W. D. (1958). On grouping for maximum homogeneity. *Journal of the American Statistical Association*, 53(284):789–798.

Gey, S., Lebarbier, E., et al. (2008). Using cart to detect multiple change points in the mean for large sample. *SSB preprint*.

Guédon, Y. (2008). Exploring the segmentation space for the assessment of multiple change-point models. *hal.inria.fr pre-print inria-00311634*.

Guédon, Y. (2013). Exploring the latent segmentation space for the assessment of multiple change-point models. *Computational Statistics*, pages 1–38.

Guédon, Y., Caraglio, Y., Heuret, P., Lebarbier, E., and Meredieu, C. (2007). Analyzing growth components in trees. *Journal of Theoretical Biology*, 248(3):418–447.

Harchaoui, Z. and Cappé, O. (2007). Retrospective multiple change-point estimation with kernels. In *IEEE Workshop on Statistical Signal Processing*.

Harchaoui, Z. and Lévy-Leduc, C. (2010). Multiple change-point estimation with a total variation penalty. *Journal of the American Statistical Association*, 105(492).

Hawkins, D. M. (2001). Fitting multiple change-point models to data. *Computational Statistics & Data Analysis*, 37(3):323–341.

Hocking, T. D. (2013). Comparing least squares segmentation code. *available at http://sugiyama-www.cs.titech.ac.jp/ toby/org/HOCKING-ml-seg-compare.pdf*.

Hocking, T. D., Schleiermacher, G., Janoueix-Lerosey, I., Boeva, V., Cappo, J., Delattre, O., Bach, F., and Vert, J.-P. (2013). Learning smoothing models of copy number profiles using breakpoint annotations. *BMC bioinformatics*, 14(1):164.

Horváth, L. (1993). The maximum likelihood method for testing changes in the parameters of normal observations. *The Annals of statistics*, pages 671–680.

 ISSN: 2102-6238

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

Killick, R. and Eckley, I. A. (2011). Changepoint: an r package for changepoint analysis. *R package version 0.6, URL http://CRAN. R-project. org/package= changepoint*.

Killick, R., Fearnhead, P., and Eckley, I. (2012). Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598.

Kolesnikov, A. and Fränti, P. (2003). Reduced-search dynamic programming for approximation of polygonal curves. *Pattern Recognition Letters*, 24(14):2243–2254.

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

Lavielle, M. (2005). Using penalized contrasts for the change-point problem. *Signal Processing*, 85(8):1501–1510.

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

Lebarbier, É. (2005). Detecting multiple change-points in the mean of Gaussian process by model selection. *Signal processing*, 85(4):717–736.

Maidstone, R., Hocking, T., Rigaill, G., and Fearnhead, P. (2014). On optimal multiple changepoint algorithms for large data. *arXiv preprint arXiv:1409.1842*.

Picard, F., Robin, S., Lavielle, M., Vaisse, C., and Daudin, J.-J. (2005). A statistical approach for array cgh data analysis. *BMC bioinformatics*, 6(1):27.

Rigaill, G. (2010). Pruned dynamic programming for optimal multiple change-point detection. *arXiv preprint arXiv:1004.0887*.

Scott, A. and Knott, M. (1974). A cluster analysis method for grouping means in the analysis of variance. *Biometrics*, 30(3):507–512.

Yao, Y.-C. and Au, S. (1989). Least-squares estimation of a step function. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 370–381.

Zhang, N. R. and Siegmund, D. O. (2007). A modified Bayes information criterion with applications to the analysis of comparative genomic hybridization data. *Biometrics*, 63(1):22–32.

Appendix A: Bound on the number of intervals

In this section, we study a special class of functions that we denote $\mathscr{B}$ and demonstrate theorem 8. A direct application of this theorem shows that if there are t candidate last change-points, then the pDPA need to store at most $2t-1$ intervals. We used this theorem in section 4 to prove the worst-case complexity of the pDPA.

A.1. $\mathscr{B}$ functions

Definition A.1.1. *Let $\mathscr{B}_n$ denote the set of all functions $B : \mathbb{R} \to \mathbb{R}$ such that*

$$\forall\, \mu \in \mathbb{R},\ B(\mu) = min_{\{t \in \{1\ldots n\}\}} \{u_{B,t} + \sum_{j=t+1}^{n+1} f_{B,j}(\mu)\}$$

where all $u_{B,t}$ are real numbers and all $f_{B,j}$ are unimodal functions of μ [4] *and any $\sum_j f_{B,j}(\mu)$ is also unimodal . Note that $\mathscr{B}_n \subset \mathscr{B}_{n+1}$. Let $\mathscr{B} = \bigcup \mathscr{B}_n$.*

[4] Here by unimodal we mean functions such that for any c in $\mathbb{R}$ the set $\{x|f(x) \leq c\}$ is an interval.

 ISSN: 2102-6238

If the loss function $\gamma(y_i,\mu)$ and any $\sum_i \gamma(y_i,\mu)$ are unimodal in μ, $\widetilde{\mathbf{Cost}}^K_{1:t}(\mu)$ is part of $\mathscr{B}_{t-1}$ because we have: $\widetilde{\mathbf{Cost}}^K_{1:t}(\mu) = \min_{\tau<t}\{\widetilde{C}^K_{1:t,\tau}(\mu)\}$.

Definition A.1.2. *For any $B \in \mathscr{B}_n$ and A a subset of $\{1\dots n\}$ we define the function B_A as*

$$\forall\ \mu,\ B_A(\mu) = min_{\{t\ \in\ A\}}\ \{u_{B,t} + \sum_{j=t+1}^{n+1} f_{B,j}(\mu)\}$$

Property 7. $B_A \in \mathscr{B}_{card(A)}$

It is easily shown for $A = \{1\dots n\}\setminus\{i\}$ with $i \in \{1\dots n\}$ and thus by induction it is true for any A.

Definition A.1.3. *The rank of a function $B \in \mathscr{B}$ is $\mathscr{R}(B) = min\{n\ \in \mathbb{N}^*\ | B \in \mathscr{B}_n\}$*

A.2. Decomposition in intervals and order of $\mathscr{B}$

Definition A.2.1. *Let $\mathscr{I}$ be a partition of $\mathbb{R}$ in a finite set of intervals $\mathscr{I} = \{I_j\}_{\{j\in\{1\dots k\}\}}$. $\mathscr{I}$ is a k-decomposition of a function $B\ \in\ \mathscr{B}$ if*

$$\forall\, I_j,\ \exists\, i,\ \forall x\ \in\ I_j \qquad B_i(x) = B(x)$$

The set of all $\mathscr{B}$ functions with a k-decomposition is denoted $\mathscr{B}^k$. Similarly the set of all $\mathscr{B}_n$ functions with a k-decomposition is denoted $\mathscr{B}^k_n$.

Definition A.2.2. *The order $O(B)$ of a $\mathscr{B}$ function is $min\{k \in \mathbb{N}^* | B\ \in\ \mathscr{B}^k\ \}$*

Théorème 8. *For all $B \in \mathscr{B}$, we have $O(B) \le 2\times\mathscr{R}(B) - 1$.*

Proof We demonstrate this theorem by induction. It is true if $\mathscr{R}(B) \le 1$. Assume it is true for any B with $\mathscr{R}(B) \le n$. Let $B \in \mathscr{B}$ with $\mathscr{R}(B) = n+1$. We have:

$$\begin{aligned}\forall \mu \in \mathbb{R}, \qquad B(\mu) &= min\{B_{\{1\dots n\}}(\mu),\ B_{\{n+1\}}(\mu)\}\\ B(\mu) &= min\{C(\mu),\ u_{B,n+1}\} + f_{B,n+2}(\mu),\end{aligned}$$

where $C \in \mathscr{B}_n$:

$$C(\mu) = \min_t\{u_{B,t} + \sum_{j=t+1}^{n+1} f_{B,j}(\mu)\}$$

Let $\mathscr{I} = \bigcup_{j\in\{1\dots k\}} I_j$ be the smallest set of intervals such that:

$$\forall\, I_j\ \in\ \mathscr{I},\ \forall\ \mu\ \in\ I_j \qquad u_{B,n+1} >\ C(\mu)$$

Let A_k be the subset of $\{1\dots n\}$ defined as $\{i\ \ |\ \ \exists\, x\ \in\ I_k\ \ C_{\{i\}}(x) < u_{B,n+1}\}$.

As $\mathscr{R}(B) = n+1$ and as for all i in $\{1\dots n\}$ $C_{\{i\}}$ is unimodal, there exists a unique j such that $i \in A_j$ and therefore $\sum_{j=1}^k card(A_j) = n$. In each interval I_k, we have $C(\mu) = C_{A_k}(\mu)$. By induction, $O(C_{A_k}) \le 2\times card(A_k) - 1$. Overall, for any B with $\mathscr{R}(B) = n+1$, we have:

$$O(B) \le\ \ \sum_{j=1}^k O(C_{A_k}) + (k+1) \le\ \ 2\sum_{j=1}^k card(A_k) + 1 \le 2n+1 \le 2\mathscr{R}(B) - 1 \quad \blacksquare$$

 ISSN: 2102-6238

Appendix B: Lower bound on the probability that $\mathscr{S}^1_{1:n,\tau}$ is empty

Model We consider the negative log-likelihood loss of a continuous uniform distribution defined on $[0,\mu]$. The loss function is:

$$\begin{aligned}\gamma(y_i,\mu) &= \log(\mu) \quad \text{if } \mu \geq Y_i\\ \gamma(y_i,\mu) &= \infty \quad \text{otherwise}\end{aligned}$$

This loss function is unimodal and any sum of $\gamma(Y_i,\mu)$ is also unimodal. Indeed, for any $t \geq \tau \geq 1$ we have:

$$\begin{aligned}\sum_{\tau+1}^{t}\gamma(Y_i,\mu) &= (t-\tau)\log(\mu) \quad \text{if } \mu \geq \hat{y}_{\tau,t}\\ \sum_{\tau+1}^{t}\gamma(Y_i,\mu) &= \infty \quad \text{otherwise}\end{aligned}$$

We define $\hat{y}_{\tau:t} = \max_{i\in\{\tau\dots t-1\}}\{y_i\}$. We have

$$\mathbf{Cost}^0_{1:t} = \min_{\mu\in\mathbb{R}^+_*}\{\sum_{1}^{t-1}\gamma(Y_i,\mu)\} = (t-1)\log(\hat{y}_{1:t}).$$

Finally note that the loss is continuous to the right and as explained in the footnote page 189 we can further restrict $\tau^K_{1:t}$ to the set of τ such that $\mathscr{S}_{1:t,\tau}$ is not restricted to a single value.

Proof Using equation 12 we get that for any t greater than 2 we have:

$$\begin{aligned}\mathscr{S}^1_{1:t+1,t} &\subset \{\mu \mid \mathbf{Cost}^0_{1:t} \leq \widetilde{C}^1_{1:t,t}(\mu)\} &(14)\\ &=]0,y_{t-1}] \cup [\hat{y}_{1:t}.\left(\frac{\hat{y}_{1:t}}{\hat{y}_{1:t-1}}\right)^{t-1},+\infty[, &(15)\\ & &(16)\end{aligned}$$

with $\left(\frac{\hat{y}_{1:t}}{\hat{y}_{1:t-1}}\right)^{t-1} \geq 1$. There are two cases in which it can be shown that $\mathscr{S}^1_{1:n,t}$ is empty or restricted to a single value and thus can be discarded.

Case 1 : If $y_{t-1} < y_t < \hat{y}_{1:t}$ we have that $\hat{y}_{1:t+1} = \hat{y}_{1:t} = \hat{y}_{1:t-1}$. Thus we have

$$\begin{aligned}\mathscr{S}^1_{1:t+1,t} &\subset]0,y_{t-1}] \cup [\hat{y}_{1:t},+\infty[,\\ \{\mu \mid \widetilde{C}^1_{1:t+1,t}(\mu) &\leq \mathbf{Cost}^0_{1:t+1}\} = [y_t,\hat{y}_{1:t+1}].\end{aligned}$$

and using equation 11 from property 3 that:

$$\mathscr{S}^1_{1:t+2,t} \subset (]0,y_{t-1}] \cup [\hat{y}_{1:t},+\infty[) \cap [y_t,\hat{y}_{1:t}] = \{y_t\}$$

and by induction we get $\mathscr{S}^1_{1:n,t} \subset \{y_t\}$.

 ISSN: 2102-6238

Case 2 : If $y_t < y_{t-1} < \hat{y}_{t:n}$ we have $\hat{y}_{1:t+1} = \hat{y}_{1:t}$ and we get using equation 11:

$$\begin{aligned}\mathscr{S}^1_{1:t+2,t} \subset & \left(]0, y_{t-1}] \cup [\hat{y}_{1:t}.\left(\tfrac{\hat{y}_{1:t}}{\hat{y}_{1:t-1}}\right)^{t-1}, +\infty[\right) \quad \cap \quad [y_t, \left(\hat{y}_{1:t+1}.\tfrac{\hat{y}_{1:t+1}}{\hat{y}_{1:t}}\right)^t] \\ \subset & \qquad\qquad [y_t, y_{t-1}].\end{aligned}$$

Now we also have:

$$\{\mu \mid \widetilde{C}^1_{1:n,t}(\mu) \quad \leq \quad \mathbf{Cost}^0_{1:n}\} = \left[\hat{y}_{t:n}, \left(\hat{y}_{1:n}.\frac{\hat{y}_{1:n}}{\hat{y}_{1:t}}\right)^{\frac{t-1}{n-t}}\right]$$

Using equation 11 we get that $\mathscr{S}^1_{1:n,t} \subset [y_t, y_{t-1}] \cap \{\mu \mid \widetilde{C}^1_{1:n,t}(\mu) \leq \mathbf{Cost}^0_{1:n}\} = \emptyset$, as $y_t < \hat{y}_{t:n}$.

Let us now compute the probability of these two events.

All Y_i are independent, they have the same density f and continuous distribution F, thus the distribution of $\hat{Y}_{1:t}$ is F^{t-1} and $P(\hat{Y}_{1:t} < x) = P(\hat{Y}_{1,t} \leq x) = F^{t-1}(x)$. As Y_t is independent of $\hat{Y}_{1:t}$ we get that $P(\hat{Y}_{1:t} < Y_{t:t+1}) = \int_{\mathbb{R}} 2f(x)F(x)P(\hat{Y}_{1:t} < x)\mathrm{d}x = \int_{\mathbb{R}} 2f(x)F(x)^t \mathrm{d}x = \frac{2}{t+1}$.

Using this, we see that case 1 happens with probability:

$$P(Y_{t-1} < Y_t < \hat{Y}_{1:t}) = P(Y_{t-1} \leq Y_t \leq \hat{Y}_{1:t-1}) = \frac{1}{2}(1 - \frac{2}{t}).$$

and case 2 happens with probability:

$$P(Y_t < Y_{t-1} < Y_{t:n}) = P(Y_t \leq Y_{t-1} \leq Y_{t+1:n}) = \frac{1}{2}(1 - \frac{2}{n-t}).$$

Furthermore case 1 and case 2 are disjoint events. Thus the probability that the two occur is the sum $1 - \frac{2}{t} - \frac{2}{n-t}$. Thus the probability that the change-point t has not been discarded at step n is lower than $\frac{2}{t} + \frac{2}{n-t}$. So taking the expectation and summing across all t smaller than n, we recover that the expected number of non discarded candidate change-points is in $\mathscr{O}(\log(n))$. ■

Appendix C: R code to assess the runtime of pDPA on simulated data

Here we provide the R code we used to assess the performances of the pDPA. The code of the pDPA is in C++.

First here is a function to simulate different type of signals.

```
#### simple simulation function
funSimu_ <- function(n, noise="norm", signal="flat"){
### signal
if(signal == "flat"){
x <- numeric(n)
}
if(signal == "sin"){## sin
x <- 2*sin(1:n/100)
}
```


 ISSN: 2102-6238

```
if(signal == "block"){## 10 blocks
nb=5
x <- rep(rep(c(0,2), nb), each=n/(2*nb))[1:n]
}
if(signal == "sinblock"){## 10 blocks + sin
nb=5
x <- rep(rep(c(0,2), nb), each=n/(2*nb))[1:n] + 2*sin(1:n/100)
}
if(signal == "linear"){
x <- 1:n
}

### noise
if(noise == "none") ### do nothing
if(noise == "norm") x <- x + rnorm(n)
if(noise == "Cauchy") x <- x +rcauchy(n)
if(noise == "unif") x <- x+ runif(n,0, 1) -0.5
return(x)
}
```

Second, here is a simple function to record the runtime of the algorithm.

```
simpleStat <- function(x){c(mean(x), sd(x), range(x))}
#### simple function to test the runtimes for various n
testRunTime <- function(Kmax = 50, ns=c(100, 200), rep=5,
funSimu=rnorm, noise, signal){
rTime <- list()
for(n in ns){
cat("Size:", n , ", Repetition 1/", rep,  "\n")
for(j in 1:rep){
x <- funSimu(n, noise=noise, signal=signal)
rTime[[length(rTime)+1]] <- c("pDPA", n, j,
system.time(res_p <- cghseg:::segmeanCO(x, Kmax=Kmax))[3])
}
}
### formating the runtimes
dTime <- data.frame(matrix(unlist(rTime), ncol=4, byrow=TRUE))
colnames(dTime) <- c("Algo", "n", "rep", "time")
dTime$time <- as.numeric(as.vector(dTime$time))
dTime$n <- as.numeric(as.vector(dTime$n))
mTime <- aggregate(dTime$time, by=list(dTime$Algo, dTime$n),
FUN= simpleStat, simplify=TRUE )
mTime <- cbind(mTime[, 1:2], mTime[, 3][, 1:4])
colnames(mTime) <- c("Algo", "n", "mean", "sd", "min", "max")
```


 ISSN: 2102-6238

```
mTime$signal <- signal
mTime$noise <- noise
mTime <- mTime[order(mTime$n), ]
return(mTime)
}
```

Here is a code to run the algorithm for different types of signal (constant, sinusoid, block), various types of noise (gaussian, uniform and Cauchy) and different values of n.

```
install.packages("cghseg")
require(cghseg)

### 20 000 +
Kmax=40
ns <- 40* 2^(1:16)
respDPA_l <- list()


for(signal in c("flat", "sin", "block", "sinblock")){
print(paste("###", signal))
for(noise in c("norm", "Cauchy", "unif")){
print(noise)
respDPA_l[[length(respDPA_l)+1]] <- testRunTime(Kmax = Kmax, ns=ns, rep=5,
funSimu=funSimu_, noise=noise, signal=signal)
}}
save(respDPA_l, file="respDPA_l.Rdata", compress=TRUE)
```

Here is a code to run the algorithm in the worst case scenario, $Y_i = i$.

```
Kmax=40
ns <- 40* 2^(1:10)
respDPA_w <- list()

respDPA_w <- list()
for(signal in c("linear")){
print(paste("###", signal))
for(noise in c("none")){
print(noise)
respDPA_w[[length(respDPA_w)+1]] <- testRunTime(Kmax = Kmax, ns=ns, rep=5,
funSimu=funSimu_, noise=noise, signal=signal)
}}
save(respDPA_w, file="respDPA_w.Rdata", compress=TRUE)
```


 ISSN: 2102-6238

Appendix D: R code to assess the runtime of the pDPA on the GSE17359 data set

Here we provide the R code we used to assess the performance of the pDPA on the data from the GSE17359 data set.

```
dat <- read.table("GSE17359_affy_snp_ratios_matrix.txt", header=TRUE, sep="\t")
require(cghseg)
ie <- which(apply(is.na(dat), 1, sum) == 0)
dat <- dat[ie,]
ns <- c(40* 2^(1:15), nrow(dat))
Kmax = 40
rTime <- list()
for(iSample in 2:ncol(dat)){
print(iSample)
for(n in ns){
x <- dat[1:n, iSample]
rTime[[length(rTime)+1]] <- c("pDPA", n, iSample,
system.time(res_p <- cghseg:::segmeanCO(x, Kmax=Kmax))[3])
}
}

dTime <- data.frame(matrix(unlist(rTime), ncol=4, byrow=TRUE))
```


 ISSN: 2102-6238