# Partitionnement optimal et élagage fonctionnel pour la détection de ruptures multiples. Application à des données d'Hybridation Génomique Comparative (CGH).

Robert Maidstone [1] & Paul Fearnhead [1] & Guillem Rigaill [2]

[1] *STOR-i CDT, Lancaster University, United Kingdom*
[2] *INRA-CNRS-Université d'Evry, URGV, 2 Rue Gaston Crémieux, 91057 Evry Cedex*

**Résumé.**

Nous nous intéressons à la détection de ruptures multiples dans un signal de taille $n$ corrompu par un bruit gaussien. Nous cherchons à identifier les ruptures qui maximisent la log-vraisemblance ou minimisent l'erreur quadratique. Nous illustrons ce problème sur des données d'hybridation génomique comparative où $n$ est de l'ordre du million.

Nous présentons un algorithme, fpop, intégrant des techniques d'élagage fonctionnel à l'algorithme de partitionnement optimal. Cet algorithme est exact et rapide. Nous démontrons que élagage opéré par fpop est au moins aussi efficace que celui de deux algorithmes récemment proposés PELT et pDPA. En pratique, sur des données simulées et réelles, nous montrons que fpop est plus rapide que PELT et pDPA.

**Mots-clés.** Détection de ruptures multiples, programmation dynamique, élagage fonctionnel, partitionnement optimal, Hybridation Génomique Comparative (CGH)

**Abstract.** We consider the problem of detecting multiple change-points in a signal of size $n$ corrupted by a Gaussian noise. We aim at detecting the change-points that maximize the log-likelihood or minimize the quadratic loss. We illustrate this problem on real Comparative Genomic Hybridization data where $n$ is typically larger than $10^5$ or $10^6$.

We describe an algorithm, fpop, which combines optimal partitioning and function pruning. This algorithm is exact and fast. We demonstrate that the pruning of fpop is at least as efficient as the pruning of two recently proposed algorithms PELT and pDPA. We empirically show on simulated and real data that the runtime of fpop is faster than PELT and pDPA.

**Keywords.** Multiple change-points, dynamic programming, function pruning, optimal partitioning, Comparative Genomic Hybridization (CGH)

# 1 Contexte

Nous nous intéressons à la détection de ruptures multiples dans la moyenne d'un signal de taille $n$ corrompu par un bruit gaussien. C'est un problème que l'on rencontre dans de nombreux domaines d'application, par exemple en économétrie [1], en traitement du signal [2] et en biologie moléculaire [3]. Dans le cas des données d'Hybridation Génomique Comparative (CGH) l'objectif est d'identifier des régions du génome où le nombre de copie d'ADN est anormal, c'est à dire différent de deux copies [3]. Ces régions altérées se traduisent par des changements abrupts dans la moyenne du signal. Certaines se détectent facilement, comme par exemple sur la figure 1, d'autres sont plus difficiles à repérer. L'identification précise de ces régions suscite beaucoup d'intérêt en cancérologie car elle contiennent très probablement des gènes impliqués dans le développement du cancer. Une manière naturelle de les détecter est de partitionner ou de segmenter le signal en segments homogènes.

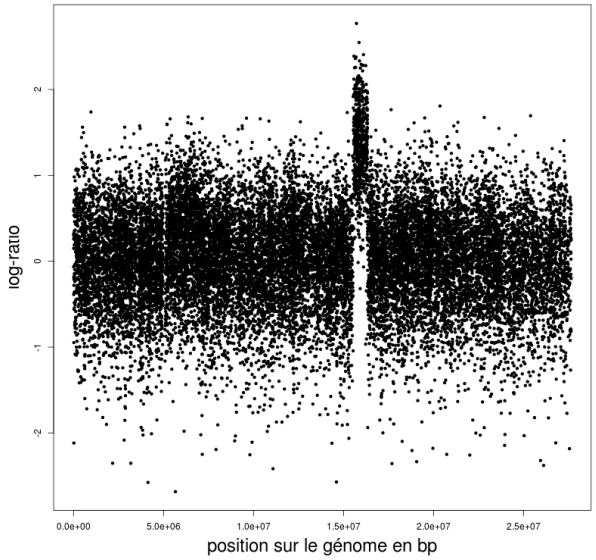


FIGURE 1 – Un profil d'Hybridation Génomique Comparative (CGH). Ce graphe représente la mesure du logarithme du nombre de copie d'ADN d'un tissu tumoral rapporté à celui d'un tissu sain en fonction de la position le long du chromosome en paire de bases (bp). Ici, seul les 20 milles premières observations du chromosome 2 sont représentées. Ce profil provient du package R SegAnnot. Aux alentours de 15 millions bp on observe un changement abrupt de la moyenne du signal qui correspond à une altération du nombre de copies d'ADN.

**Modèle statistique** Considérons $n$ observations $y_1, \cdots, y_n$ réalisations de $n$ variables aléatoires indépendantes de loi normale. Nous supposons que la moyenne de ces lois est affectée par $K-1$ changements ou ruptures abruptes à des instants $\tau$ non déterminés : $1 < \tau_2 \cdots < \tau_K < n+1$. Nous adoptons la convention $\tau_1 = 1$ et $\tau_{K+1} = n+1$. La variance du signal reste, elle, inchangée. Formellement nous supposons le modèle suivant :

$$\forall i \quad | \quad \tau_k \ \leq \ i \ < \ \tau_{k+1} \ , \qquad Y_i \sim \mathcal{N}(\mu_k, \sigma^2) \quad \text{indépendant.} \tag{1}$$

L'ensemble $m$ des ruptures, $m = \{\tau_1, \cdots, \tau_{K+1}\}$, détermine une segmentation du signal en $K$ morceaux ou segments. Nous appellerons $\mathcal{M}_K$ l'ensemble des segmentations en $K$ morceaux. Étant donné le modèle (1) une manière d'estimer la position des ruptures et de retrouver la segmentation de log-vraisemblance pénalisée maximale. Cela revient ici à minimiser l'erreur quadratique pénalisée. Pour une segmentation $m$ particulière cette erreur s'écrit

$$\sum_k \sum_{i \in k} C_{\tau_k, \tau_{k+1}} + pen(m), \qquad \text{avec} \qquad C_{\tau_k, \tau_{k+1}} = \text{argmin}_{\mu_k} \sum_{i=\tau_k}^{\tau_{k+1}-1} (Y_i - \mu_k)^2. \tag{2}$$

# 2 Algorithmes élagués et exacts

De manière générale, il est possible de trouver la segmentation $m$ qui minimise le critère (2) en utilisant un algorithme de programmation dynamique de complexité $\mathcal{O}(Kn^2)$. Toutefois, le temps de calcul correspondant est prohibitif pour de grands profils avec $n \geq 10^5$. Deux algorithmes, PELT [4] et pDPA [5], permettent de retrouver plus rapidement la segmentation qui minimise le critère (2) pour des formes particulières de la pénalité $pen(m)$ ou pour un nombre de ruptures maximales petit. Ces deux algorithmes reposent sur un élagage efficace de l'espace des segmentations qui est très grand, $card(\mathcal{M}_K) = \binom{n-1}{K-1}$. Ci-dessous, nous décrivons rapidement ces deux algorithmes sans rentrer dans les détails techniques de leurs implémentations et des structures qu'ils manipulent.

**PELT, un élagage basé sur des inégalités** L'algorithme PELT [4] retrouve la segmentation $m$ qui minimise le critère de l'équation (2) pour une pénalité de la forme $pen(m) = \lambda \ . \ card(m)$ où $\lambda$ est un paramètre choisi par l'utilisateur. PELT est une amélioration de l'algorithme de partitionnement optimal [6]. Dans PELT chaque segmentation $m$ candidate est représentée par son erreur pénalisée courante qui prend ses valeurs dans $\mathbb{R}$. PELT élague l'espace des segmentations en comparant à chaque étape $i$ le coût courant des segmentations candidates. L'élagage repose sur une inégalité [7]. Il est démontré dans [4] que, si le vrai nombre de ruptures croît linéairement avec $n$, PELT a une complexité en espérance de $\mathcal{O}(n)$. Si le vrai nombre de ruptures est petit devant $n$ la vitesse de l'algorithme se dégrade (voir [4] ou la figure 2 droite).

**pDPA, un élagage fonctionnel** L'algorithme pDPA [6] retrouve la segmentation en $k$ segments qui minimisent la perte quadratique pour toutes les valeurs de $k$ entre 1 et $K_{max}$, où $K_{max}$ est un paramètre choisit par l'utilisateur. Une fois ces $K_{max}$ segmentations obtenues il ne reste plus qu'à sélectionner celle qui a la meilleure erreur quadratique pénalisée. Dans pDPA chaque segmentation $m$ en $k$ segments est représentée par une fonction polynomiale du second degré $P_m(\mu_k)$ où $\mu_k$ est la moyenne du dernier segment. $P_m$ n'est autre que l'erreur quadratique de la segmentation en fonction de $\mu_k$. pDPA élague l'espace des segmentations en comparant à chaque étape $i$ les $P_m$ des segmentations candidates. On parlera d'élagage fonctionnel [7]. Dans [5] il est démontré que pDPA est au pire en $\mathcal{O}(K_{max}\ n^2)$. En pratique l'algorithme est en $\mathcal{O}(K_{max}\ n\ \log(n))$ (voir [5] et [8]). Les performances de pDPA se dégradent si le nombre de ruptures est important.

# 3 Un nouvel algorithme fpop

Les algorithmes PELT et pDPA sont complémentaires au sens où PELT est efficace quand le vrai nombre de ruptures est grand et pDPA est efficace quand le vrai nombre de ruptures est faible. Ici nous présentons un algorithme fpop qui vise à être efficace dans les deux cas. Comme PELT, fpop est une amélioration de l'algorithme de partitionnement optimal [6] et maximise le critère de l'équation (2) pour une pénalité de la forme $pen(m) = \lambda\ .\ card(m)$. La différence entre PELT et fpop est que fpop repose sur un élagage fonctionnel similaire à celui de pDPA alors que PELT repose sur un élagage basé sur des inégalités. Nous ne décrirons pas dans ce résumé l'algorithme fpop dans les détails. Nous nous contentons de décrire quelques résultats que nous avons obtenus.

**Résultat théorique** Nous avons démontré que l'élagage de fpop est au moins aussi efficace que celui de PELT et de pDPA, au sens ou toute segmentation candidate qui a été élaguée par PELT ou pDPA à l'étape $i$ sera élagué par fpop au plus tard à l'étape $i$. Cette preuve garantit donc que fpop combine les avantages de PELT et de pDPA.

**Temps de calcul en pratique** Nous avons implémenté fpop en $R/C++$. Nous avons comparé son temps de calcul à ceux de PELT, pDPA et de l'heuristique de segmentation binaire [9, 10]. Nous parlons ici d'heuristique au sens où la segmentation binaire cherche bien à minimiser le critère (2) mais ne garantit pas de retrouver exactement la segmentation qui minimise (2). Quand le nombre de ruptures est faible la segmentation binaire est très compétitive en terme de temps de calcul, sa complexité est au pire en $\mathcal{O}(K_{max}n)$. Nous avons comparé les temps de calcul sur des données simulées avec un grand ou petit nombre de ruptures. Nous avons également testé fpop sur un benchmark de données CGH proposé par [11]. fpop est plus rapide que PELT et pDPA aussi bien quand le vrai nombre de ruptures est grand (voir la figure 2 à gauche) que quand le vrai nombre de ruptures est faible (voir la figure 2 à droite). Sur le benchmark proposé par [11] fpop est plus rapide

que PELT et pDPA et a un temps de calcul comparable à celui de l'heuristique binaire. Par ailleurs pour un faible nombre de ruptures son temps de calcul est comparable à celui de l'heuristique de segmentation binaire qui est en $\mathcal{O}(n \log(K_{max}))$.

grand nombre de ruptures petit nombre de ruptures

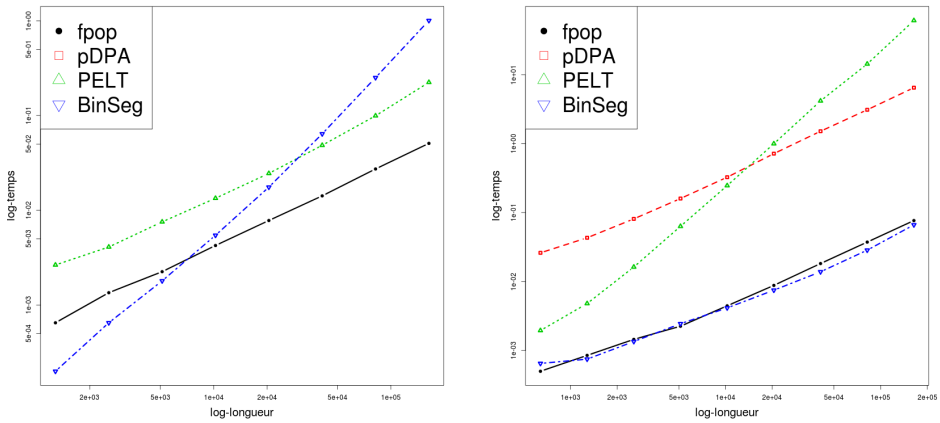


FIGURE 2 – Log du temps de calcul de PELT, pDPA, fpop et de l'heuristique de segmentation binaire en fonction de la log-longueur du signal pour un grand nombre de ruptures réelles ou un faible nombre de ruptures réelles

| algorithme | médiane du temps en s | max du temps en s |
|---|---|---|
| PELT | 14 | 14 |
| pDPA | 3.3 | 3.4 |
| fpop | 0.076 | 0.076 |
| Binseg | 0.043 | 0.044 |

TABLE 1 – Temps de calcul médian en seconde sur le benchmark proposé dans [11] ($n = 153663$). Nous avons choisi les mêmes valeurs des paramètres $K_{max}$ et $\lambda$ que dans [11]. C'est à dire $\lambda = log(n)$ pour PELT et fpop et $K_{max} = 15$ pour l'heuristique de segmentation binaire (Binseg) et pDPA.

# 4 Remerciement

Nous remercions Pierre Neuvial et Patrick Rigaill d'avoir relu ce résumé.

# Bibliographie

[1] J. Bai et P. Perron (2003), Computation and analysis of multiple structural change models, Journal of Applied Econometrics, 1–22
[2] O. Gillet, S. Essid et G. Richard (2007), On the correlation of automatic audio and visual segmentation of music videos, IEEE Transactions on Circuits and Systems for Video Technology
[3] F. Picard, S. Robin, M. Lavielle, C. Vaisse et J-J. Daudin (2005), A statistical approach for array CGH data analysis, BMC Bioinformatic, 6, 1471-2105
[4] R. Killick et P. Fearnhead et I. Eckley (2012). Optimal detection of changepoints with a linear computational cost. JASA, 107(500) :1590–1598
[5] G. Rigaill. Pruned dynamic programming for optimal multiple change-point detection. Arxiv :1004.0887, April 2010. URL http ://arxiv.org/abs/1004.0887
[6] B. Jackson et J.D. Sargle et D. Barnes et S. Arabhi et A. Alt et P. Gioumousis et E. Gwin et P. Sangtrakulcharoen et L. Tan et T.T. Tsai (2005). An algorithm for optimal partitioning of data on an interval. IEEE, Signal Processing Letters, 12(2) :105–108
[7] R. Maidstone et P. Fearnhead et A. Letchford (2013). Changepoint Detection Using Pruned Dynamic Programming. Poster URL :
http ://www.lancaster.ac.uk/pg/maidston/ICPposter.pdf
[8] A. Cleynen et M. Koskas et E. Lebarbier et G. Rigaill et S. Robin (2014). Segmentor3IsBack : an R package for the fast and exact segmentation of Seq-data Algorithms for Molecular Biology 2014, 9 :6 (10 March 2014)
[9] S. Gey et E. Lebarbier (2008). Using CART to detect multiple change-points in the mean for large samples.
[10] A.J. Scott et M. Knott (1974). A cluster analysis method for grouping means in the analysis of variance. Biometrics, 30(3) :507512.
[11] T.D. Hocking (2013). Comparing least squares segmentation code. URL :
http ://sugiyama-www.cs.titech.ac.jp/ toby/org/HOCKING-ml-seg-compare.pdf