Pruned dynamic programming for optimal multiple change-point detection

Guillem Rigaill

April 7, 2010

Abstract

Multiple change-point detection models assume that the observed data is a realization of an independent random process affected by $K-1$ abrupt changes, called change-points, at some unknown positions. For off-line detection a dynamic programming (DP) algorithm retrieves the $K-1$ change-points minimizing the quadratic loss and reduces the complexity from $\Theta(n^K)$ to $\Theta(Kn^2)$ where n is the number of observations. The quadratic complexity in n still restricts the use of such an algorithm to small or intermediate values of n. We propose a pruned DP algorithm that recovers the optimal solution. We demonstrate that at worst the complexity is in $O(Kn^2)$ time and $O(Kn)$ space and is therefore at worst equivalent to the classical DP. We show empirically that the run-time of our proposed algorithm is drastically reduced compared to the classical DP algorithm. More precisely, our algorithm is able to process a million points in a matter of minutes compared to several days with the classical DP algorithm. Moreover, the principle of the proposed algorithm can be extended to other convex losses (for example the Poisson loss) and as the algorithm process one observation after the other it could be adapted for on-line problems.

1 Introduction

Segmentation or change-point detection problems arise in many fields, from econometrics [2] to molecular biology [12]. The goal is to partition the signal in K homogeneous and contiguous segments of variable length or time. These segments are delimited by the $K-1$ change-points. These change-points can be identified on-line (sequentially) or off-line (retrospectively) [3]. For the off-line detection problem, a dynamic programming (DP) algorithm recovers the optimal solution with respect to the quadratic loss in $\Theta(Kn^2)$ time and $\Theta(Kn)$ space ([8], [11], [1], [9]). The quadratic complexity in n restricts the use of such an algorithm to small or intermediate values of n.

To get around this problem several approaches have been proposed. These methods rely either on efficient heuristics (see [7]) or on small modification of the optimisation problem (see [13], [10]). Overall these methods will not find the optimal solution with respect to the quadratic loss and in this sense there is a trade-off between run-time and introducing some errors. Many other methods can be used for fast segmentation, for example wavelets denoising (see for example [4]), or particle filter (see [5], [6]) but they do not try to optimize the same criteria.

Here we propose a pruned DP algorithm which retrieves the optimal solution for large values of n. More precisely, we have a sequence of n observations $\{Y_i\}_{i\in[\![1,n]\!]}$ in a set A ($\mathbb{N}$, $\mathbb{R}$, $\cdots$). We call $\mathcal{M}_{K,t}$ the set of all possible segmentations in K segments up to point t ($card(\mathcal{M}_{K,t}) = \binom{t-1}{k-1}$). We denote $r_k(m) = [\![\tau_k, \tau_{k+1}[\![$ the k-th segment of segmentation m delimited by change-points τ_k and τ_{k+1}. With the convention that $\tau_1 = 1$ and $\tau_{K+1} = t+1$, any segmentation m of $\mathcal{M}_{K,t}$ is defined as $\{[\![\tau_1, \tau_2[\![, \cdots, [\![\tau_K, \tau_{K+1}[\![\}$. Our algorithm recovers:

$$min_{\{m \,\in\, \mathcal{M}_{K,t}\}}\{\sum_{r \,\in\, m} min_{\{\mu \,\in\, \mathbb{R}\}}\{\textstyle\sum_{i \,\in\, r}\gamma(Y_i,\mu)\}\}$$

where $\gamma : A \times \mathbb{R} \to \mathbb{R}$ is a convex function of μ. Throughout the paper we will take the quadratic loss, $\gamma(Y_i, \mu) = (Y_i - \mu)^2$, as a leading example.

In Section 2, we give a brief overview of the classical DP ([8], [11], [1]) and give the main idea of our algorithm. In Section 3, we describe our proposed pruned DP algorithm and in Section 4 we describe a simple execution of the algorithm. In Section 5 we demonstrate its worst case complexity. In Section 6, we compare the run-time of our pruned DP algorithm and the classical DP on both simulated and real data.

2 From the classical to the pruned DP algorithms

For any segment r, we define its cost as $g_r(\mu) = \sum_{i \,\in\, r}\gamma(Y_i,\mu)$ and its optimal cost as $c_r = min_\mu\{g_r(\mu)\}$. We define $C_{K,t} = min_{\{m \,\in\, \mathcal{M}_{K,t}\}}\{\sum_{r \,\in\, m} c_r\}$. Because $C_{K,t}$ is segment additive, we retrieve the update equation:

$$\forall\, t \geq K \qquad C_{K,t} = \min_{K-1\leq j<t}\{\, C_{K-1,j} + c_{[\![j+1,t+1[\![}\} \tag{1}$$

Using update equation (1), the classical DP performs t comparisons at each step and therefore retrieves $C_{K,n+1}$ in $\Theta(Kn^2)$.

Assume we know the optimal mean value μ^* of the last segment, then the problem is to minimize $H_{k,t}(\mu^*)$:

$$H_{k,t}(\mu^*) = min_{\{m \,\in\, \mathcal{M}_{k,t}\}}\{\sum_{r=r_1(m)}^{r_{k-1}(m)} c_r + g_{r_k(m)}(\mu^*)\},$$

$H_{k,t}(\mu^*)$, as $C_{K,t}$, is segment additive and $g_{r_k(m)}(\mu^*)$ is point additive, thus we retrieve the update equation:

$$\forall\, t \geq k \qquad H_{k,t+1}(\mu^*) = \min(\, H_{k,t}(\mu^*),\ C_{k-1,t}\,) + \gamma(Y_{t+1},\mu^*) \tag{2}$$

Using update equation (2), we perform one comparaison at each step and we retrieve $H_{k,n+1}(\mu^*)$ in $\Theta(n)$. Most of the time we do not know μ^*. A simple heuristic is to test a large but finite set of possible μ^* denoted here $\{\mu^*_p\}_{p\in[\![1,P]\!]}$. Using update equation (2) on all $H_{k,t}(\mu^*_p)$, we perform P comparaisons at each step and we retrieve all $H_{k,n+1}(\mu^*_p)$ in $\Theta(Pn)$.

To recover a solution as close as possible to the optimal one, we can increase the number of tested values and for each of these μ^*_p store an optimal last change-point.

Intuitively, two close values of μ^*, $\mu^*_{p_1}$ and $\mu^*_{p_2}$, probably share the same optimal last change-points. In other words the heuristic is storing too much information and it seems that the heuristic should stores only critical values of μ^* corresponding to a change in the last optimal change-point. Our pruned DP algorithm is build on this idea.

3 Algorithm

We define $H_{k,t}$ as:

$$H_{k,t}(\mu) = min_{\{m \ \in \ \mathcal{M}_{k,t}\}}\{ \sum_{r=r_1(m)}^{r_{k-1}(m)} c_r + \sum_{i \in r_k(m)} \gamma(Y_i, \mu) \ \}$$

and $h_{k,t,t'}(\mu)$ as:

$$\forall \ t' \ < \ t \qquad h_{k,t,t'}(\mu) = C_{k-1,t'} + \sum_{i=t'+1}^{t} \gamma(Y_i, \mu),$$

where $h_{k,t,t'}$ represents the cost of the best candidate segmentation in k segments having its last change-point at position t' and a last mean value of μ. We have: $H_{k,t}(\mu) = min_{\{t' \ \in \ [\![k,t-1]\!]\}}\{ \ h_{k,t,t'}(\mu) \ \}$. We define $S_{k,t,t'}$ as the set of μ such that a last change-point at t' is optimal:

$$S_{k,t,t'} = \{\mu \mid h_{k,t,t'}(\mu) = H_{k,t}(\mu)\}$$

and the set $I_{k,t,t'}$ of μ such that a last change at t' is better than a change at t:

$$I_{k,t,t'} = \{ \ \mu \mid h_{k,t,t'}(\mu) \ \leq \ C_{k-1,t} \ \}$$

Here we review the key properties of $h_{k,t,t'}$ and $S_{k,t,t'}$ that allow their simple construction in the course of the proposed pruned DP algorithm.

Proposition 3.1

Additivity $\quad \forall \ t' \ < \ t \qquad h_{k,t+1,t'}(\mu) = h_{k,t,t'}(\mu) + \gamma(Y_{t+1}, \mu)$
Stability $\quad h_{k,t,t_0}(\mu) \geq h_{k,t,t_1}(\mu) \quad \Rightarrow \ \forall \ t^* \ \geq \ t \quad h_{k,t^*,t_0}(\mu) \geq h_{k,t^*,t_1}(\mu)$
Pruning

$$\forall \ t > t' \geq k, \qquad S_{k,t+1,t'} = S_{k,t,t'} \ \cap I_{k,t,t'} \tag{3}$$

$$S_{k,t,t'} = \emptyset \qquad \Rightarrow \qquad \forall \ t^* \geq t \quad S_{k,t^*,t'} = \emptyset \tag{4}$$

$$\forall \ t' \geq k, \qquad S_{k,t',t'} = \complement_{\mathbb{R}}(\cup_{t \in [\![k,t'-1]\!]} I_{k,t,t'}) \tag{5}$$

Storage *$I_{k,t,t'}$ is an interval and $S_{k,t,t'}$ is a finite union of intervals.*

In other words, if $S_{k,t,t'}$ is empty we are certain that, whatever the observations after t, the best segmentation does not have a change at t'. The proofs of these properties are simple and left to the reader.

Using these propositions our pruned DP algorithm stores a list, *listOfCandidates*$_k$, of candidate last change-points t', update for each t' a cost function $Cost_{k,t'}$ and a set of winning intervals $Set_{k,t'}$ (see proposition 3.1-Storage). $Cost_{k,t'}$ and $Set_{k,t'}$ store the successive $h_{k,t,t'}$ and $S_{k,t,t'}$.

In the case of the quadratic loss, $\gamma(Y_i, \mu) = (Y_i - \mu)^2$, $Cost_{k,t'}$ is stored as a second degree polynomial function of μ. It is initialized when arriving at observation t' as $Cost_{k,t'} := C_{k-1,t'}$. $Cost_{k,t'}$ is updated with proposition 3.1-Additivity.

$Set_{k,t'}$ is stored as a list of intervals (see proposition 3.1-Storage). It is initialized when arriving at observation t' as $Set_{k,t'} := D$. In the case of the quadratic loss, D can be chosen as $[min_i(Y_i), max_i(Y_i)]$. $Set_{k,t'}$ is updated using proposition 3.1-Pruning (3) and (5) and a candidate change-point is pruned as soon as $Set_{k,t'} = \emptyset$ using proposition 3.1-Pruning (4).

The algorithm is described in more details in Algorithm 1 for a given $k \geq 2$. Algorithm 1 should be iterated for all k up to K. For $k = 1$, all $C_{1,t}$ are computed in $O(n)$ as in the classical DP algorithm. Importantly, enough the algorithm could be modified for on-line problems as the observations are used one after the other.

Algorithm 1 Pruned DP algorithm for segmentation

```
$listOfCandidates_k := \{k-1\}$
$Cost_{k,k-1} := C_{k-1,k-1}$    ;  $Set_{k,t'} := D$
for $t \in [k..n]$ do
  $Cost_{k,t} := C_{k-1,t}$    ;  $Set_{k,t} := D$
  for $l \in listOfCandidates_k$ do
    $Cost_{k,l} := Cost_{k,l} + \gamma(Y_{t+1}, .)$
    $I = \{\mu \mid Cost_{k,l}(\mu) \leq C_{k-1,t}\}$
    $Set_{k,l} := Set_{k,l} \cap I$
    if $Set_{k,l} = \emptyset$ then
      $listOfCandidates_k := listOfCandidates_k \setminus \{l\}$
    end if
    $Set_{k,t} := Set_{k,t} \setminus I$
  end for
  if $Set_{k,t} \neq \emptyset$ then
    $listOfCandidates_k := listOfCandidates_k \cup \{t\}$
  end if
  $C_{k,t} = min_{\{l \in listOfCandidates_k\}}\{min_\mu\,(Cost_{k,l})\}$
end for
```

4 An example

Here we illustrate the different steps of the proposed DP algorithm with a four point signal and for $k = 2$ segments. These four points are respectively $0, 0.5, 0.4, -0.5$ (see Figure 1 top left).

Step 1 The only two segments segmentation up to point $t = 2$ has a break after point 1. Its cost is $h_{2,2,1}(\mu^*) = C_{1,1} + (0.5 - \mu^*)^2 = 0 + (0.5 - \mu^*)^2$ and is stored in $Cost_{2,1}$. The cost depends on μ^* and is shown in Figure 1 (top middle). $Set_{2,1}$ is set to $[-0.5, 0.5]$.

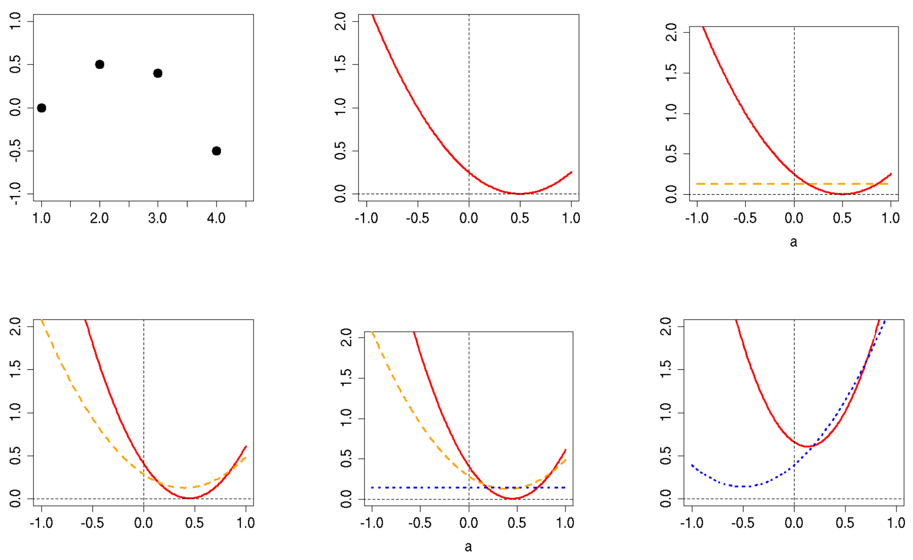

Figure 1: Four points signal (top left); Successive stored candidate functions $h_{2,.,.}$ as a function of μ^*. $h_{2,2,1}$ (top middle), $h_{2,2,1}$ $(-)$ and $h_{2,2,2}$ $(- \ -)$ (top right), $h_{2,3,1}$ $(-)$ and $h_{2,3,2}$ $(- \ -)$ (bottom left), $h_{2,3,1}$ $(-)$, $h_{2,3,2}$ $(- \ -)$ and $h_{2,3,3}$ $(\cdots)$ (bottom middle), $h_{2,4,1}$ $(-)$ and $h_{2,4,3}$ $(\cdots)$ (bottom right).

Step 2 If we go up to point $t = 3$ an other solution is a break just after 2. Its cost is $h_{2,2,2}(\mu^*) = C_{1,2}$ it is stored in object $Cost_{2,1}$ and is compared to $h_{2,2,1}$ (see Figure 1 top right) too retrieve $Set_{2,1} = [0.5 - \frac{1}{2\sqrt{2}}, 0.5]$ and $Set_{2,2} = [-0.5, 0.5 - \frac{1}{2\sqrt{2}}]$. Then we use $(0.4 - \mu^*)^3$ to update $Cost_{2,1}$ and $Cost_{2,2}$ to retrieve $h_{2,3,1}$ and $h_{2,3,2}$ (see Figure 1 bottom left).

Step 3 If we go up to point $t = 4$ an other solution is a break just after 3. Its cost is $h_{2,3,3}(\mu^*) = C_{1,3}$, it is stored in object $Cost_{2,1}$ and then compared to $h_{2,3,1}$ and $h_{2,3,2}$ to retrieve $Set_{2,1} = [\frac{9}{20} - \frac{3}{20}, 0.5]$, $Set_{2,2} = \emptyset$ and $Set_{2,3} = [-0.5, \frac{9}{20} - \frac{3}{20}]$. It can be seen that $h_{2,3,2}$ (see Figure 1 bottom middle) is beaten for any μ^* thus the pruned DP algorithm discard $h_{2,3,2}$. Thus we already know that the best segmentation will not have a break between 2 and 3. Then we use $(-0.5 - \mu^*)^2$ to update $Cost_{2,3}$ and $Cost_{2,1}$ to retrieve $h_{2,4,1}$ and $h_{2,4,3}$ (see Figure 1 bottom right). We see that the best segmentation is obtain for $\mu^* = -0.5$ and a break after point 3.

5 Worst case complexity

Here, we demonstrate that, if the γ function is convex the complexity of our proposed algorithm is bounded in $O(Kn^2)$ time and if the cost function can be written using a base the complexity is bounded in $O(Kn)$ space.

For a given number of segments k, at step i of the algorithm, at most i change-point candidates need to be updated. For each candidate change-point t the algorithm will:

- update the cost function $Cost_{k,t}$;

- compute its minimum value ;
- compute the roots of $Cost_{k,t}(\mu) = C_{k-1,i}$ to get $I = \{\mu | Cost_{k,t}(\mu) \leq C_{k-1,i}\}$;
- update the winning intervals $Set_{k,t}$.

All these steps are in $O(1)$ except the update of intervals. Theorem A.3, demonstrated in the appendix, show that if there are i candidates then there are at most $2i - 1$ intervals. Thus, the time complexity is bounded by $\sum_{i=1}^{n}(O(3i) + O(2i - 1)) \sim O(n^2)$. As this should be done for all $k \leq K$ we retrieve a worst case complexity of $O(Kn^2)$.

Assuming that the cost function can be stored efficiently using a base of ℓ functions, the algorithm will store the optimal change-point for each t in $O(Kn)$, the candidate cost function for each t in $O(\ell n)$ and the set of intervals in $O(2n-1)$. Thus if ℓ is small compared to K, we retrieve an $O(Kn)$ space complexity.

For $\gamma(Y_t, \mu) = (Y_t - \mu)^2$ the worst case complexity, in $O(Kn^2)$, is achieved for the sequence $Y_i = i$. If the signal is constant, the complexity is in $O(Kn)$ as the algorithm will store only one candidate change-point at each step. Thus it can be hoped that for more or less piecewise constant signal the complexity will be closer to $O(Kn)$ than to $O(Kn^2)$.

6 Empirical complexity

In this section, we assessed the efficiency of the pruned DP algorithm on simulated data set and real data set for $\gamma(Y_t, \mu) = (Y_t - \mu)^2$. The algorithm was implemented in C++ and was run on a 3.16GHz Intel(R) Xeon X5460.

6.1 Simulated data

We simulated sequences using a constant, sinusoid or rectangular signal. For the sinusoid and rectangular waves we consider various amplitudes and frequencies. We considered a Gaussian noise of variance 1, a uniform noise of variance 1, a chi-square noise of variance 1 and a Cauchy noise.

We first compare the pruned DP to the classical DP algorithm for relatively small sample sizes, $n \leq 16.10^3$ and checked that the two algorithms retrieve the same result. Then we evaluate the run-time of the pruned DP algorithm to process large signals, $n = 10^6$. Finally, we compute at each step of the algorithm the maximum number of intervals stored.

The comparaison with the classical DP is summarized in Figure 2. The mean number of intervals stored by the pruned DP algorithm is summarized in Figure 3 left and middle. The empirical run-time for large signals of 10^6 points in 50 segments was around 80-250 seconds. The shorter run-times were achieved for a cauchy noise with a run-time close to 80 seconds. The longer run-time were achieved for sine wave with a run-times around $240 - 250$ seconds. Non sine wave signal had a run-time around 100-120 seconds. Mean run-times for the different tested signals will be summarized in table 1 (supplementary materials). Moreover the codes used for these simulations will be provided in supplementary materials.

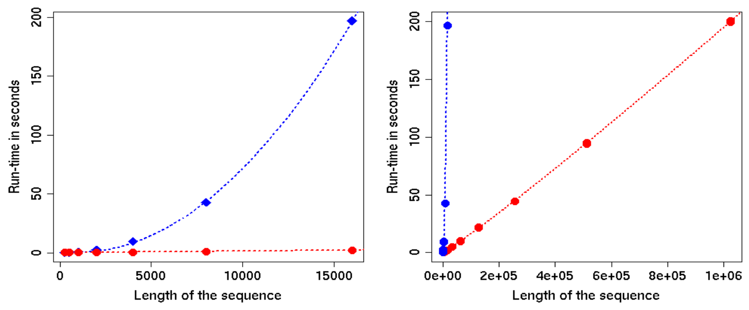

Figure 2: Mean run-time of the classical DP (♦) and pruned DP algorithm (●) for simulated constant signal with a normal noise. For sequences up to 16.10^3 points (left) and for sequences up to 10^6 points (right)

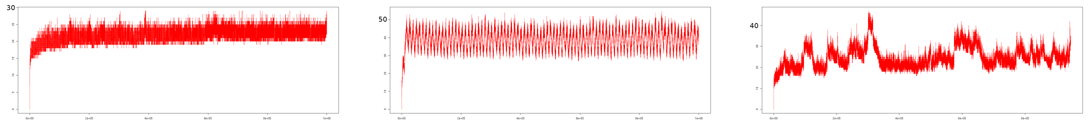

Figure 3: Maximum number of intervals stored by the pruned DP algorithm at each step of the algorithm for $k = 2$. For a 100 simulated constant signal of 10^6 points with a normal noise of variance 1(left), for a 100 simulated sine wave signal of 10^6 points with a normal noise of variance 1(middle), for 19 CNV profiles of 10^6 points (right).

6.2 Real data

We download the publicly available $GSE17359$ project from the GEO database. This data set is composed of 18 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 data a multiple change-point detection procedure based on the quadratic loss is often used ([12]). For each SNP array experiment there are two signals of a million points (CNV and SNP) that were analyzed separately. The mean run-time were respectively of 101 (CNV) and 103 (SNP) seconds. The maximum number of intervals stored by the pruned DP algorithm is summarized in Figure 3 right.

Overall it can be seen that the pruned DP algorithm recovers the optimal solution in a drastically reduced amount of time compared to the classical DP algorithm. This can be explain by the very small number of candidates stored by the algorithm (see Figure 3). More importantly it is able to retrieve this solution in a reasonable amount of time even for large signals and can be used in practise for example to analyse large SNP profiles.

References

[1] J. Bai and P. Perron. Computation and analysis of multiple structural change models. *J. Appl. Econ.*, 18:1–22, 2003.

[2] Jushan Bai and Pierre Perron. Estimating and testing linear models with multiple structural changes. *Econometrica*, 66(1):47–78, 1998.

[3] M. Basseville and N. Nikiforov. *The detection of abrupt changes.* 1993.

[4] Erez Ben-Yaacov and Yonina C. Eldar. A fast and flexible method for the segmentation of aCGH data. *Bioinformatics*, 24(16):i139–145, August 2008.

[5] Nicolas Chopin. Dynamic detection of change points in long time series. *Annals of the Institute of Statistical Mathematics*, 59(2):349–366, 2007.

[6] Paul Fearnhead and Zhen Liu. On-line inference for multiple change points problems. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY B*, 69:589–605, 2007.

[7] S. Gey and E. Lebarbier. Using cart to detect multiple change points in the mean for large samples. Technical report, Research Report n°12 SSB group, 2008.

[8] S.B. Guthery. Partition regression. *Journal ofthe American Statistical. Association*, 69:945–947, 1974.

[9] Y. Guédon. Explorating the segmentation space for the assessment of multiple change-points models. Technical report, Preprint INRIA n°6619, 2008.

[10] Z. Harchaoui and C. Lévy-Leduc. Catching change-points with lasso. In *NIPS 2009*, 2009.

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

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

[13] R. Tibshirani and P. Wang. Spatial smoothing and hot spot detection for cgh data using the fused lasso. *Biostatistics.*, pages 1–12, 2007.

A Lemma and Proofs

In this section we demonstrate that if there are t candidate change-points, then there are at most $2t-1$ intervals. This can be demonstrated for a relatively large class of convex functions.

A.1 $\mathcal{B}$ functions

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

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

where all $u_{B,t}$ are real numbers and all $f_{B,j}$ are convex functions of μ. Note that $\mathcal{B}_n \subset \mathcal{B}_{n+1}$. Let $\mathcal{B} = \bigcup \mathcal{B}_n$.

Definition A.1.2 *For any B in $\mathcal{B}_n$ and A a subset of $[\![1,n]\!]$ we define the function B_A as*

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

Proposition A.1 $B_A \in \mathcal{B}_{card(A)}$

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

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

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

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

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

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

Proposition A.2 *If B is $\mathcal{B}$ then there exists a k-decomposition of B.*

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

Theorem A.3 *For all $B \in \mathcal{B}$, we have $O(B) \leq 2 \times \mathcal{R}(B) - 1$.*

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

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

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

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

Let $\mathcal{I} = \bigcup_{j\in[\![1,k]\!]} I_j$ be the smallest set of intervals such that:

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

Let A_k be the subset of $[\![1,n]\!]$ defined as $\{i|\ \exists\ x\ \in\ I_k\quad C_{\{i\}}(x)\ <\ u_{B,n+1}\}$. As $\mathcal{R}(B) = n+1$ and as for all i in $[\![1,n]\!]$ $C_{\{i\}}$ is convex, 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, $O(C_{A_k}) \leq 2\times card(A_k) - 1$.

Overall, for any B, we have:

$$O(B) \leq\ \sum_{j=1}^{k} O(C_{A_k}) + (k+1) \leq\ 2\sum_{j=1}^{k} card(A_k) - k + (k+1) \leq 2n + 1.$$

■