Équilibrage d'un arbre pondéré : le problème Balanced Mobile

Yassine Hamoudi
encadré par Sophie Laplante et Roberto Mantaci

1er juin - 21 août, 2015

English version: http://perso.ens-lyon.fr/yassine.hamoudi/files/BalancedMobiles.pdf

Table des matières

1 Introduction

Ce rapport de stage présente l'étude d'un nouveau problème algorithmique qui, malgré la simplicité de son énoncé, ne paraît pas avoir été étudié dans la littérature scientifique existante. Il consiste à placer n poids $p_1, p_2, \dots p_n$ aux feuilles d'un arbre binaire de sorte à minimiser son "déséquilibre", dans le sens que nous allons définir par la suite. Nous exposons ci-dessous les définitions préalables à notre travail.

Définition 1. Un *mobile* est un arbre binaire dont chaque feuille comporte un poids (entier positif non nul).

Si M est un mobile, on notera P_M la somme des poids placés sur ses feuilles. On définit alors son déséquilibre comme suit.

Définition 2. Soit M un mobile. Le *déséquilibre (global)* Δ_M de M est :
- si M a une seule feuille, $\Delta_M = 0$
- sinon, notons respectivement M_G et M_D les sous-arbres gauche et droit de M. On a alors $\Delta_M = \Delta_{M_G} + \Delta_{M_D} + |P_{M_G} - P_{M_D}|$.

On définit également pour chaque nœud interne ν de M le *déséquilibre local* $\delta(\nu)$ comme la différence (en valeur absolue) des poids des sous-arbres gauche et droit de ν. La valeur de Δ_M est ainsi la somme des déséquilibres locaux.

Problème 1. Le problème Balanced Mobile (abrégé BM) consiste à construire pour tout ensemble de poids $\{p_1, \dots, p_n\}$ un mobile de déséquilibre minimum.

Pour un même ensemble de poids, il peut exister plusieurs mobiles de déséquilibre minimum. Ceux-ci sont qualifiés de *mobiles optimaux*.

La Figure 1 représente deux mobiles construits à partir des poids $\{2, 3, 5, 7\}$. A côté de chaque nœud interne figure son déséquilibre local (on conservera cette convention par la suite). Le déséquilibre global du mobile de gauche est 8, celui de droite est 4. Le mobile de droite est optimal : on ne peut pas obtenir de déséquilibre inférieur à 4 avec les poids $\{2, 3, 5, 7\}$.

Figure 1 – Deux exemples de mobiles construits sur les poids $\{2, 3, 5, 7\}$.

Les motivations ayant conduit à l'étude de ce problème sont exposées dans la partie 2. Différents algorithmes répondant au problème BM seront présentés dans la partie 3. Nous restreindrons par la suite notre étude à deux situations particulières : le cas des poids tous égaux (partie 4) et les mobiles construits sur des puissances de deux (partie 5).

2 Motivations et état de l'art

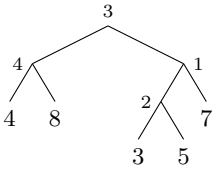
Bien qu'il n'y ait pas d'équivalent au problème Balanced Mobile dans la littérature scientifique existante, nous avons pu isoler deux problèmes qui présentent des similarités avec notre travail. Nous les exposons ci-dessous.

2.1 Multiplications chaînées de matrices

Notre intérêt pour le problème Balanced Mobile vient de l'étude préalable de la variante suivante.

Problème 2. Le problème Ordered Balanced Mobile$(p_1, \dots, p_n)$ (abrégé OBM$(p_1, \dots, p_n)$) consiste à construire sur la séquence de poids $p_1, \dots, p_n$ un mobile de déséquilibre minimum, dont les feuilles comportent de gauche à droite, et dans l'ordre, les poids $p_1, \dots, p_n$.

Exemple 3. Le mobile suivant, de déséquilibre 10, est solution au problème OBM$(4, 8, 3, 5, 7)$:



Le problème OBM$(p_1, \ldots, p_n)$ se résout facilement par programmation dynamique, en $O(n^3)$. En effet, si on note $\Delta[i, j]$ le déséquilibre optimal de OBM$(p_i, \ldots, p_j)$, on a la relation suivante :

$$\Delta[i,j] = \begin{cases} 0 & \text{si } i = j \\ \min\limits_{i \leq k < j} \left\{ \Delta[i,k] + \Delta[k+1,j] + \left| \sum\limits_{l=i}^{k} p_l - \sum\limits_{l=k+1}^{j} p_l \right| \right\} & \text{si } i < j \end{cases}$$

On constate qu'OBM est très similaire au problème Matrix Chain Multiplication.

Problème 3. Soit $A_1, \ldots, A_n$ une séquence de matrices telle que pour tout i, la matrice A_i est de dimension $d_{i-1} \times d_i$. Le problème Matrix Chain Multiplication consiste à trouver un parenthésage qui minimise le coût de calcul du produit $A_1 \cdot A_2 \cdots A_n$.

Si on note $m[i, j]$ le coût optimal pour calculer $A_i \cdots A_j$, l'algorithme de programmation dynamique présenté dans [God73] est identique à celui de OBM et utilise la formule suivante :

$$m[i,j] = \begin{cases} 0 & \text{si } i = j \\ \min\limits_{i \leq k < j} \{ m[i,k] + m[k+1,j] + d_{i-1} d_k d_j \} & \text{si } i < j \end{cases}$$

2.2 Codage de Huffman

Le codage de Huffman, détaillé dans [Huf52], répond à la question algorithmique suivante.

Problème 4. Soit $a_1, \ldots, a_n$ un ensemble de lettres. On note p_i le nombre d'occurrences de a_i (dans un texte donné). Comment encoder chaque lettre a_i par un mot $m_i \in \{0, 1\}^+$ tel que :

— le code est préfixe : aucun mot m_i n'a pour préfixe un autre mot m_j

— le coût $\sum\limits_{i=1}^{n} p_i \cdot |m_i|$ est minimal.

L'algorithme de Huffman commence par associer à chaque a_i un mobile constitué d'une unique feuille de poids p_i (il y a initialement n mobiles). Puis, tant qu'il existe au moins deux mobiles distincts, les deux mobiles M_1 et M_2 de plus petit poids sont regroupés en un unique mobile de poids $P_{M_1} + P_{M_2}$ (une racine est ajoutée, qui pointe vers M_1 et M_2).

Ensuite, chaque arête de l'unique mobile final M est étiquetée 0 si elle descend à gauche et 1 si elle descend à droite. La lettre a_i est alors encodée par le mot constitué des étiquettes successives du chemin menant de la racine de M à la feuille de poids p_i.

Cet algorithme construit un code préfixe optimal de coût $\sum\limits_{i=1}^{n} p_i \cdot depth(M, p_i)$ (où $depth(M, p_i)$ est la profondeur de la feuille de poids p_i dans M). La Figure 2 illustre son fonctionnement sur un exemple.

lettre a_i	a	b	c	d
occurrence p_i	8	1	7	4
encodage m_i	1	010	00	011

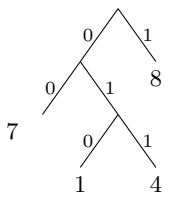


Figure 2 – Exemple de codage de Huffman.

Le codage de Huffman met ainsi en lumière une question équivalente au Problème 4, et qui présente une certaine similarité avec le problème Balanced Mobile :

Problème 5. Étant donné un ensemble de poids $p_1, \ldots, p_n$, comment construire un mobile M sur ces poids qui minimise $\sum_{i=1}^{n} p_i \cdot depth(M, p_i)$?

Le codage de Huffman répond de manière optimale à cette question. L'algorithme SMALLEST que nous allons présenter dans la partie 3.1 réutilise la première partie du codage (construction du mobile M) pour l'appliquer à notre problème.

Il existe également une généralisation du Problème 4, dans laquelle l'alphabet d'encodage Σ peut contenir des lettres de longueur variable. On note $l(x)$ la longueur d'une lettre $x \in \Sigma$ et $l(m)$ la somme des longueurs des lettres du mot $m \in \Sigma^+$. Le problème consiste alors à associer à chaque lettre a_i un mot $m_i \in \Sigma^+$ de sorte à obtenir un code préfixe de coût $\sum_{i=1}^{n} p_i \cdot l(m_i)$ minimal. Ce problème a été résolu par un programme linéaire en nombres entiers présenté dans [Kar61]. Il existe également un PTAS exposé dans [GKY02]. Cependant, malgré plus de soixante ans de travail sur cette question, on ne dispose pas d'algorithme polynomial ou de preuve de NP-complétude.

2.3 Définitions alternatives

Les différents problèmes exposés précédemment peuvent être regroupés dans un cadre commun. En effet, étudions la question suivante.

Problème 6. Soit n éléments $x_1, \ldots, x_n$ d'un ensemble muni d'une opération *associative* $\times$. Notons $c(x, y)$ le coût du calcul $x \times y$. Comment calculer $x_1 \times x_2 \times \cdots \times x_n$ en minimisant le coût total ?

ORDERED BALANCED MOBILE et MATRIX CHAIN MULTIPLICATION correspondent à des restrictions de ce problème. Dans le premier cas, les éléments sont les poids $p_1, \ldots, p_n$ et l'opération $\times$ est l'addition, de coût $c(x, y) = |x - y|$. Dans le second cas, les éléments sont les matrices et c est le coût classique de la multiplication entre matrices.

Il est également intéressant de généraliser la question précédente aux opérations commutatives.

Problème 7. Soit n éléments $x_1, \ldots, x_n$ d'un ensemble muni d'une opération *associative* et *commutative* $\times$. Notons $c(x, y)$ le coût du calcul $x \times y$. Comment calculer $x_1 \times x_2 \times \cdots \times x_n$ en minimisant le coût total ?

Dans ce problème, lorsque l'opération $\times$ est l'addition entre entiers de coût $c(x, y) = x + y$, on obtient le Problème 5. Si le coût est remplacé par $c(x, y) = |x - y|$, on retrouve le problème BALANCED MOBILE.

3 Étude du cas général

Lorsque les poids $p_1, \ldots, p_n$ sont quelconques, il est possible de construire un mobile optimal en temps $2^{O(n)}$. Pour ce faire, il suffit de calculer par programmation dynamique tous les mobiles optimaux associés aux 2^n sous-ensembles de $\{p_1, \ldots, p_n\}$.

Nous présentons ici quelques alternatives à cet algorithme. Bien que nous n'ayons pas trouvé de procédure plus efficace dans le cas général, un algorithme glouton en $O(n \log n)$ permet de résoudre le problème lorsque le déséquilibre optimal est au plus un.

3.1 L'algorithme SMALLEST

La recherche d'une solution au problème BM s'oriente naturellement dans un premier temps vers un algorithme glouton. L'algorithme SMALLEST présenté ci-dessous s'inspire du codage de Huffman, et va être étudié tout au long du rapport.

Algorithme 1 : Algorithme SMALLEST

Entrées : poids $p_1 \leq \cdots \leq p_n$
Sorties : déséquilibre SMALLEST$(p_1, \ldots, p_n)$
si $n = 2$ **alors**
  Renvoyer $|p_2 - p_1|$
sinon
  Renvoyer $|p_2 - p_1| +$ SMALLEST(SORT$(p_1 + p_2, p_3, \ldots, p_n)$)

On note que dans l'algorithme précédent $p_1 + p_2$ n'est pas forcément le plus petit des $n-1$ poids $p_1 + p_2, p_3, \ldots, p_n$. Ces poids doivent donc être réordonnés en ordre croissant. En pratique, on peut utiliser un tas pour extraire à chaque fois les deux plus petits éléments, puis réinsérer leur somme. L'algorithme SMALLEST fonctionne ainsi en $O(n \log n)$.

Par ailleurs, l'algorithme précédent ne construit pas un mobile mais renvoie un déséquilibre possible avec les poids $p_1, \ldots, p_n$. Il est cependant facile de modifier l'algorithme pour construire au fur et à mesure le mobile associé au résultat retourné. On confondra donc par la suite le déséquilibre renvoyé par l'algorithme SMALLEST et le mobile correspondant.

Définition 4. On dit qu'un mobile est *construit selon l'algorithme* SMALLEST s'il existe un déroulement de l'algorithme conduisant à ce mobile.

Il arrive parfois que plusieurs mobiles (forcément de déséquilibres égaux) puissent être construits selon l'algorithme SMALLEST sur un même ensemble de poids. La Figure 3 illustre cette situation.

FIGURE 3 – Deux mobiles construits selon l'algorithme SMALLEST avec les poids $\{2, 2, 3, 4\}$.

L'algorithme SMALLEST ne produit pas systématiquement des mobiles optimaux. Il ne s'agit pas pas non plus un algorithme d'approximation. En effet, la Figure 4 démontre que pour l'ensemble de poids $\{2, 2, 3, 3, 5, 5, 10, 10, \cdots, 5 \cdot 2^k, 5 \cdot 2^k\}$ l'algorithme SMALLEST aboutit à un déséquilibre $4 + 2k$ alors qu'il existe un mobile de déséquilibre deux.

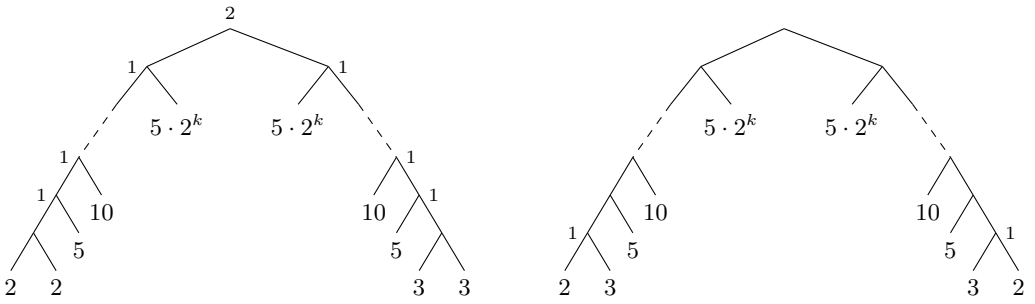


FIGURE 4 – A gauche, un mobile de déséquilibre $4 + 2k$ construit selon l'algorithme SMALLEST. A droite, un mobile optimal de déséquilibre 2.

On peut toutefois caractériser un certains nombres de cas dans lesquels l'algorithme SMALLEST produit bien des mobiles optimaux. Ceci va faire l'objet des parties 3.2, 4.2 et 5.

3.2 Mobiles de déséquilibre au plus un

Lorsque le déséquilibre minimum est égal au plus à un, l'algorithme SMALLEST parvient à construire un mobile optimal. On démontre ce résultat en deux temps.

Théorème 5. L'algorithme SMALLEST construit un mobile de déséquilibre nul si et seulement si le déséquilibre minimum est nul.

Démonstration. Le sens direct de ce théorème est évident. On démontre le sens inverse par récurrence sur n.

Si $n = 2$ il y a un seul mobile possible.

Soit $n > 2$ et $p_1 \leq \cdots \leq p_n$ un ensemble de poids. Supposons qu'il existe un mobile de déséquilibre nul construit sur ces poids. On a forcément $p_1 = p_2$ et on peut supposer que les feuilles de poids p_1 et p_2 sont frères dans un mobile optimal (si p_1 et son frère n'ont pas le même poids alors le déséquilibre global n'est pas nul). Or, la première étape de l'algorithme SMALLEST consiste à regrouper les feuilles de poids p_1 et p_2 puis à effectuer un appel récursif sur $\{p_1 + p_2, \ldots, p_n\}$. Ce dernier ensemble admet un mobile de déséquilibre nul, donc par hypothèse de récurrence SMALLEST parvient à construire un mobile de déséquilibre minimum. □

On énonce ci-dessous une autre caractéristique des mobiles de déséquilibre nul.

Théorème 6. Soit $p_1 \leq \cdots \leq p_n$ un ensemble de poids. Il existe un mobile de déséquilibre nul sur ces poids si et seulement si :

— pour tout i, le rapport p_i/p_1 est une puissance de deux

— $\sum_{i=1}^{n} p_i/p_1$ est une puissance de deux.

Démonstration. On démontre que les deux conditions du théorème sont nécessaires et suffisantes. Cela s'effectue facilement par récurrence sur le nombre n de poids. □

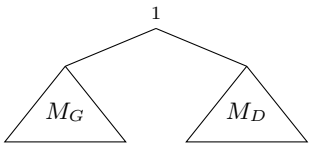
Corollaire 7. Dans un mobile de déséquilibre nul, seuls les plus petits poids peuvent avoir une valeur impaire.

On démontre maintenant que l'algorithme SMALLEST construit également des mobiles optimaux lorsque le déséquilibre minimum est égal à un.

Théorème 8. L'algorithme SMALLEST construit un mobile de déséquilibre un si et seulement si le déséquilibre minimum est égal à un.

Démonstration. On démontre le sens inverse en étudiant la structure des mobiles de déséquilibre un.

1er cas : Considérons tout d'abord un mobile M dont le déséquilibre local de la racine est égal à un, et tous les autres déséquilibres locaux sont nuls. Notons M_G et M_D les sous-arbres gauches et droits de M :



On a $|P_{M_G} - P_{M_D}| = 1$. Par conséquent, P_{M_G} ou P_{M_D} est impair. Supposons, sans perte de généralité, que $P_{M_G} = 2m + 1$ (on a donc $P_{M_D} = 2m$ ou $P_{M_D} = 2m + 2$). Le déséquilibre global Δ_{M_G} de M_G est nul. Or, les mobiles de poids impair et de déséquilibre nul ont une seule feuille. Par conséquent, M_G est constitué d'une unique feuille de poids $2m + 1$.

Par ailleurs, tous les poids de M_D sont inférieur à $2m + 1$ (excepté si $P_{M_D} = 2m + 2$ et M_D a une seule feuille). Ainsi, si on applique l'algorithme SMALLEST sur les poids de M, il va commencer par construire un mobile de déséquilibre nul sur les poids de M_D d'après le Théorème 5. Puis, à la dernière étape, il va créer un déséquilibre local de valeur un en unissant les poids $2m + 1$ et P_{M_D}.

2e cas : On considère maintenant un mobile M de déséquilibre un dont l'unique déséquilibre local non nul ne se trouve pas à la racine. Le sous-arbre M' de M ayant un déséquilibre non nul à sa racine a un poids impair : $P_{M'} = 2m + 1$. Si on remplace dans M le sous-arbre M' par une feuille de poids $2m + 1$, le déséquilibre global devient nul. D'après le Corollaire 7, cela implique que tous les poids de M (en dehors de ceux du sous-arbre M') sont supérieurs à $2m + 1$. Par conséquent, d'après le 1er cas étudié ci-dessus, l'algorithme SMALLEST exécuté sur les poids de M va commencer par construire un mobile de déséquilibre un sur les poids de M'. Puis, d'après le Théorème 5, il n'ajoutera aucun déséquilibre local supplémentaire dans la suite de son exécution. □

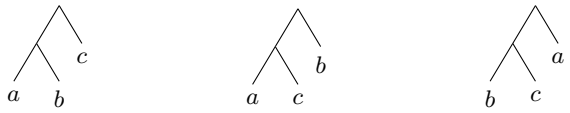
Les deux théorèmes précédents démontrent que l'algorithme SMALLEST parvient à construire des mobiles optimaux lorsque le déséquilibre minimum est au plus un. La Figure 4 témoigne cependant que ce n'est plus le cas à partir du déséquilibre deux. Toutefois, on étudiera dans les parties 4.2 et 5 d'autres situations dans lesquels cet algorithme produit des mobiles optimaux.

3.3 Algorithme SMALLEST relâché

Il est possible de relâcher la procédure de l'algorithme SMALLEST pour construire systématiquement des mobiles optimaux. Cet algorithme, qui fonctionne en temps exponentiel, va reposer sur le lemme suivant.

Lemme 9. Soit $p_1 \leq p_2 \leq \cdots \leq p_n$ un ensemble de poids. Il existe un mobile optimal sur $p_1, \ldots, p_n$ dans lequel le frère de la feuille de poids p_1 est une autre feuille de poids p_i.

Démonstration. Soit $a \leq b \leq c$. Il y a trois mobiles possibles sur ces poids-là :



Or $|c-(a+b)|+(b-a) \leq (c-a)+(a+c-b) \leq (c-b)+(b+c-a)$, donc le mobile de gauche est systématiquement optimal. Par conséquent, on peut pivoter les deux mobiles de droite pour aboutir à celui de gauche, sans augmenter le déséquilibre global.

Ainsi, pour tout mobile M construit sur les poids $p_1 \leq \cdots \leq p_n$, on peut effectuer la rotation précédente pour faire "descendre" le plus petit poids p_1 dans l'arbre jusqu'à ce que son frère soit une feuille. Cette opération n'augmente pas le déséquilibre global. □

Le Lemme 9 nous permet de relâcher modérément la procédure de l'algorithme SMALLEST pour obtenir un algorithme exponentiel capable de déterminer s'il existe un mobile de déséquilibre inférieur à k. En effet, si p_1 est le plus petit poids, il suffit de chercher récursivement pour chaque poids p_i tel que $|p_1 - p_i| \leq k$ s'il existe un mobile de déséquilibre inférieur à $k - |p_1 - p_i|$ sur les poids $(p_1 + p_i, p_2, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n)$. L'algorithme 2 présente cette démarche.

Algorithme 2 : Algorithme SMALLEST-REL

Entrées : $(p_1 \leq \cdots \leq p_n), k$
Sorties : un booléen exprimant l'existence d'un mobile de déséquilibre inférieur à k sur les poids $p_1, \ldots, p_n$

```
1 si n = 2 alors
2 |  Retourner le booléen |p_1 − p_2| ≤ k
3 sinon
4 |  pour chaque i ≥ 2 tels que |p_1 − p_i| ≤ k faire
5 |  |  si SMALLEST-REL(SORT(p_1 + p_i, p_2, ..., p_{i−1}, p_{i+1}, ... p_n), k − |p_1 − p_i|) alors
6 |  |  |  Retourner Oui
7 |  Retourner Non
```

Cet algorithme est particulièrement intéressant lorsque le déséquilibre est faible, comme en témoigne l'analyse de sa complexité.

Théorème 10. L'algorithme SMALLEST-REL s'exécute en temps $O((k+1)\log(n)n^{k+1})$.

Démonstration. On représente $\{p_1, \ldots, p_n\}$ par l'ensemble $\{(q_j, n_j)\}$ où les q_j sont les poids distincts du multiensemble des poids $p_1, \ldots, p_n$ et les n_j sont leurs multiplicités ($\sum n_j = n$).

La ligne 4 de l'algorithme nécessite d'effectuer un seul test pour chaque q_j. Puisque ces derniers sont tous distincts, l'appel récursif ligne 5 est exécuté au plus $k+1$ fois. Chacun de ces appels nécessite au préalable une insertion en $O(\log n)$, comme effectué pour l'algorithme SMALLEST.

Notons alors $T(n,k)$ la complexité de calcul de SMALLEST-REL$((p_1,\dots,p_n),k)$. On a la relation : $T(n,k) \leq \sum_{i=0}^{k} T(n-1,i) + O((k+1)\log n)$ et $T(n,0) = O(n \log n)$. Un raisonnement par récurrence sur cette formule démontre que $T(n,k) = O((k+1)\log(n) n^{k+1})$. □

On peut également calculer le déséquilibre optimal en exécutant l'algorithme pour $k = 1, 2, 4, 8 \dots$ jusqu'à trouver k_0 tel que SMALLEST-REL$((p_1,\dots,p_n),k_0)$ soit faux et SMALLEST-REL$((p_1,\dots,p_n), 2 \cdot k_0)$ soit vrai. On effectue ensuite une recherche dichotomique du déséquilibre minimum dans l'intervalle $[k_0, 2 \cdot k_0]$. On obtient ainsi un mobile de déséquilibre minimum m en temps $O((m+1)\log(m+1)\log(n) n^{2m+1})$.

Un script Python [1] a été implémenté sur la base de cet algorithme. On constate expérimentalement que les temps de calcul deviennent trop importants au-delà d'une dizaine de poids.

3.4 Optimisation linéaire en nombres entiers

On développe ici une autre approche au problème BM. Nous allons concevoir un programme linéaire en nombres entiers dont la solution correspondra à un mobile optimal sur les poids $p_1, \dots, p_n$.

Pour ce faire, on remarque que tous les arbres à n feuilles comportent $n-1$ nœuds. On associe alors à chaque nœud un numéro différent, compris entre 1 et $n-1$. Ces numéros sont attribués par ordre croissant dans l'arbre, de haut en bas, de gauche à droite (niveau par niveau). La Figure 5 donne un exemple de numérotation (l'arbre de gauche a trois nœuds numérotés de 1 à 3).

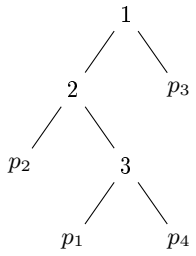
On constate que la racine a systématiquement le numéro 1. Par ailleurs, pour deux nœuds de numéros $u < v$, le nœud u ne peut pas être dans le sous-arbre de racine v.

On définit ensuite pour chaque poids p_i $(1 \leq i \leq n)$ et chaque nœud u $(1 \leq u \leq n-1)$ deux variables entières $0 \leq d_{i,u} \leq 1$ et $0 \leq g_{i,u} \leq 1$. Pour tout mobile construit sur les poids $p_1, \dots, p_n$, on associe l'assignation des variables suivante :

— $d_{i,u} = 1$ si p_i apparaît dans le sous-arbre droit du nœud u et 0 sinon

— $g_{i,u} = 1$ si p_i apparaît dans le sous-arbre gauche du nœud u et 0 sinon.

Le tableau de la Figure 5 donne l'assignation des variables associée au placement des p_i sur l'arbre de gauche.



poids / nœud	1	2	3
p_1	$d_{1,1} = 0$	$d_{1,2} = 1$	$d_{1,3} = 0$
	$g_{1,1} = 1$	$g_{1,2} = 0$	$g_{1,3} = 1$
p_2	$d_{2,1} = 0$	$d_{2,2} = 0$	$d_{2,3} = 0$
	$g_{2,1} = 1$	$g_{2,2} = 1$	$g_{2,3} = 0$
p_3	$d_{3,1} = 1$	$d_{3,2} = 0$	$d_{3,3} = 0$
	$g_{3,1} = 0$	$g_{3,2} = 0$	$g_{3,3} = 0$
p_4	$d_{4,1} = 0$	$d_{4,2} = 1$	$d_{4,3} = 1$
	$g_{4,1} = 1$	$g_{4,2} = 0$	$g_{4,3} = 0$

FIGURE 5 – Exemple de placement des p_i et assignation associée.

Le déséquilibre local du nœud u est ainsi : $\left|\sum_{i=1}^{n} p_i(d_{i,u} - g_{i,u})\right|$. Le déséquilibre global (somme des déséquilibres locaux) est :

$$\sum_{u=1}^{n-1} \left|\sum_{i=1}^{n} p_i(d_{i,u} - g_{i,u})\right|$$

Définition 11. Soit $(d_{i,u}, g_{i,u})_{1 \leq i \leq n, 1 \leq u \leq n-1}$ des variables. Une assignation de ces variables est dite *valide* si elle correspond à l'assignation associée à un mobile construit sur des poids $p_1, \dots, p_n$.

Le problème BM sur les poids $p_1, \dots, p_n$ consiste donc à minimiser $\sum_{u=1}^{n-1} |\sum_{i=1}^{n} p_i(d_{i,u} - g_{i,u})|$ parmi l'ensemble des assignations valides des variables $(d_{i,u}, g_{i,u})_{1 \leq i \leq n, 1 \leq u \leq n-1}$. On cherche alors à caractériser par des contraintes linéaires les assignations valides. On commence par énoncer un

1. http://perso.ens-lyon.fr/yassine.hamoudi/wp-content/uploads/2015/08/smallestRelaxed.py

ensemble de conditions nécessaires et suffisantes qu'elles doivent vérifier, puis nous traduirons ces conditions en un programme linéaire.

Théorème 12. Soit $(d_{i,u}, g_{i,u})_{1\leq i\leq n, 1\leq u\leq n-1}$ des variables binaires. Toute assignation valide de ces variables doit vérifier les conditions suivantes :

Condition 1 : Pour tout poids p_i et pour tout nœud u, on ne peut pas avoir avoir simultanément $d_{i,u} = 1$ et $g_{i,u} = 1$ (la feuille de poids p_i n'est pas à la fois dans le sous-arbre droit et le sous-arbre gauche du nœud u).

Condition 2 : Le nœud de numéro 1 est systématiquement la racine. Par conséquent, quelque soit p_i, il faut $d_{i,1} = 1$ (la feuille de poids p_i est dans le sous-arbre droit de la racine) ou $g_{i,1} = 1$ (la feuille de poids p_i est dans le sous-arbre gauche de la racine).

Condition 3 : Tout nœud u admet au moins une feuille dans son sous-arbre gauche (i.e. il existe i tel que $g_{i,u} = 1$) et une feuille dans son sous-arbre droit (i.e. il existe j tel que $d_{j,u} = 1$).

Condition 4 : Si $d_{i,u} = 1$ et $d_{i,v} = g_{i,v} = 0$ pour tout $v > u$, alors le fils droit du nœud u est une feuille de poids p_i. Par conséquent, aucun autre p_j ne peut être dans le sous-arbre droit du nœud u, i.e. $d_{j,u} = 0$ quelque soit $j \neq i$. De même, si $g_{i,u} = 1$ et $d_{i,v} = g_{i,v} = 0$ pour tout $v > u$ alors $g_{j,u} = 0$ pour tout $j \neq i$.

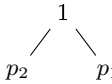
Condition 5 : Si p_i et p_j sont tous deux dans le sous-arbre de racine v (i.e. $d_{i,v} = 1$ ou $g_{i,v} = 1$, et $d_{j,v} = 1$ ou $g_{j,v} = 1$) alors pour tout $u < v$ on a $d_{i,u} = d_{j,u}$ et $g_{i,u} = g_{j,u}$ (du nœud 1 au nœud v, les deux suites sont identiques).

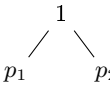
Condition 6 : Supposons que p_i et p_j soient dans le même sous-arbre (gauche ou droit) du nœud u (i.e. $d_{i,u} = d_{j,u} = 1$ ou $g_{i,u} = g_{j,u} = 1$). Notons alors v le plus petit entier strictement supérieur à u tel que $d_{i,v} = 1$ ou $g_{i,v} = 1$ (v existe sinon on contredit la condition 4). Le poids p_j doit également être dans le sous-arbre de racine v, on a donc : $d_{j,v} = 1$ ou $g_{j,v} = 1$. Par ailleurs, il n'y a aucun autre nœud dans l'arbre entre u et v (le nœud v est un fils de u), on a donc aussi $d_{j,w} = g_{j,w} = 0$ pour tout $u < w < v$.

On démontre maintenant par récurrence que ces six conditions sont suffisantes.

Théorème 13. Soit $(d_{i,u}, g_{i,u})_{1\leq i\leq n, 1\leq u\leq n-1}$ des variables binaires. Toute assignation de ces variables vérifiant les six conditions du Théorème 12 est valide.

Démonstration. On considère l'hypothèse de récurrence H_n suivante : toute assignation des variables binaires $(d_{i,u}, g_{i,u})_{1\leq i\leq n, 1\leq u\leq n-1}$ vérifiant les six conditions du Théorème 12 est valide.

Si $n = 2$, il y a 4 variables : $d_{1,1}, g_{1,1}, d_{2,1}$ et $g_{2,1}$. Les conditions 2 et 3 ne permettent que deux assignations possibles. La première correspond à $d_{1,1} = g_{2,1} = 1$ et $g_{1,1} = d_{2,1} = 0$. Cette assignation est valide car elle est associée au mobile suivant : 

. L'autre assignation (symétrique) correspond au mobile : 

.

Soit maintenant $n > 2$. On suppose l'hypothèse H_m vraie pour tout $m < n$. On considère alors une assignation des variables $(d_{i,u}, g_{i,u})_{1\leq i\leq n, 1\leq u\leq n-1}$ vérifiant les conditions du Théorème 12. Montrons que cette assignation est valide.

On partitionne $\{1, \ldots, n\}$ en deux ensembles disjoints : $A = \{i/d_{i,1} = 1\}$ et $B = \{i/g_{i,1} = 1\}$. On a $A \cap B = \varnothing$ d'après la condition 1, et $A \cup B = \{1, \ldots, n\}$ à cause de la condition 2. De plus, d'après la condition 3, chacun de ces ensembles contient au moins un élément. On va raisonner sur l'ensemble A en supposant qu'il comporte au moins deux éléments (sinon on raisonne sur B).

Notons $A' = \{u > 1/\exists i \in A$ tel que $d_{i,u} \neq 0$ ou $g_{i,u} \neq 0\}$. L'ensemble $\Omega = \{(d_{i,u}, g_{i,u})_{i\in A, u\in A'}\}$ vérifie les conditions du Théorème 12 :

- Ω vérifie trivialement les conditions 1, 4, 5 et 6 (car $(d_{i,u}, g_{i,u})_{1\leq i\leq n, 1\leq u\leq n-1}$ les vérifie).
- Si la condition 3 n'est pas vérifiée par Ω, alors il existe $u \in A'$ tel que pour tout $i \in A$, $d_{i,u} = 0$, ou pour tout $i \in A$, $g_{i,u} = 0$. Supposons pour tout $i \in A$, $d_{i,u} = 0$. D'après la condition 3, il existe $j \in B$ et $v \in B'$ tel que $d_{j,v} = 1$. Ainsi, d'après la condition 5, $d_{j,1} = d_{i,1}$ et $g_{j,1} = g_{i,1}$. Or, par définition de A et B, on a $d_{i,1} = 1$ et $d_{j,1} = 0$.
- Notons $u_0 = \min A'$. D'après la condition 6, et par définition de A et A', pour tout $i \in A$, $d_{i,u_0} = 1$ ou $g_{i,u_0} = 1$. La condition 2 est ainsi vérifiée.

Enfin, d'après le Lemme 14 (énoncé et démontré ci-dessous) on a bien $|A'| = |A| - 1$. On peut alors appliquer l'hypothèse de récurrence $H_{|A|}$ sur Ω : toute assignation des variables de Ω vérifiant les six conditions précédentes est valide. On démontre le même résultat pour les variables restantes associées à B et B'. On en conclut que H_n est vraie. □

Lemme 14. Soit $n \geq 2$ et $m \geq 1$ des entiers. Considérons $(d_{i,u}, g_{i,u})_{1 \leq i \leq n, 1 \leq u \leq m}$ des variables binaires. S'il existe une assignation de ces variables vérifiant les six conditions du Théorème 12, alors $m = n - 1$.

Démonstration. On démontre ce résultat par récurrence sur n.

Si $n = 2$, le résultat est trivial.

Soit $n \geq 3$ et $m \geq 1$. Supposons qu'il existe une assignation des variables $(d_{i,u}, g_{i,u})_{1 \leq i \leq n, 1 \leq u \leq m}$ qui vérifie les six conditions du Théorème 12. On reprend les définitions des ensembles A, A', B et B' utilisés dans la preuve du Théorème 13. Le raisonnement suivi dans la preuve du théorème précédent s'applique à nouveau ici et permet de montrer qu'il existe des assignation des variables $(d_{i,u}, g_{i,u})_{i \in A, u \in A'}$ et $(d_{i,u}, g_{i,u})_{i \in B, u \in B'}$ qui vérifient les conditions du Théorème 12. Par hypothèse de récurrence, on a donc $|A'| = |A| - 1$ et $|B'| = |B| - 1$. Or $m = 1 + |A'| + |B'|$ et $n = |A| + |B|$. On obtient bien $m = n - 1$. □

Il reste à traduire les six conditions précédentes par des contraintes linéaires.

Théorème 15. Pour tout poids $p_1, \ldots, p_n$, le programme linéaire (en nombres entiers) suivant calcul le déséquilibre optimal.

$$\texttt{Minimiser} \sum_{u=1}^{n-1} \left| \sum_{i=1}^{n} p_i (d_{i,u} - g_{i,u}) \right| \text{ sujet à :}$$

Contrainte 0 : $\forall i, u, 0 \leq d_{i,u} \leq 1$ et $0 \leq g_{i,u} \leq 1$

Contrainte 1 : $\forall i, u, d_{i,u} + g_{i,u} \leq 1$

Contrainte 2 : $\forall i, d_{i,1} + g_{i,1} = 1$

Contrainte 3 : $\forall u, \begin{cases} \sum_i d_{i,u} > 0 \\ \sum_i g_{i,u} > 0 \end{cases}$

Contrainte 4 : $\forall i \neq j, \forall u, \begin{cases} (1 - d_{i,u}) + \sum_{v>u} (d_{i,v} + g_{i,v}) \geq d_{j,u} \\ (1 - g_{i,u}) + \sum_{v>u} (d_{i,v} + g_{i,v}) \geq g_{j,u} \end{cases}$

Contrainte 5 : $\forall i \neq j, \forall u < v, \begin{cases} d_{i,u} + (d_{i,v} + g_{i,v} + d_{j,v} + g_{j,v}) \leq 2 + d_{j,u} \\ g_{i,u} + (d_{i,v} + g_{i,v} + d_{j,v} + g_{j,v}) \leq 2 + g_{j,u} \end{cases}$

Contrainte 6 : $\forall i \neq j, \forall u < v, \begin{cases} 2 - d_{i,u} - d_{j,u} + \sum_{w=u+1}^{v} (d_{i,w} + g_{i,w}) \geq d_{j,v} + g_{j,v} \\ 2 - g_{i,u} - g_{j,u} + \sum_{w=u+1}^{v} (d_{i,w} + g_{i,w}) \geq d_{j,v} + g_{j,v} \end{cases}$

Démonstration. Montrons par exemple que les contrainte 5 et 6 expriment bien les conditions 5 et 6 du Théorème 12.

Les inégalités de la contrainte 5 sont contraignantes uniquement si $d_{i,v} + g_{i,v} + d_{j,v} + g_{j,v} = 2$ (i.e. p_i et p_j sont tous deux dans le sous-arbre de racine v). Dans ce cas-là, les deux inégalités impliquent $d_{i,u} = d_{j,u}$ et $g_{i,u} = g_{j,u}$ quelque soit $u < v$.

Les inégalités de la contrainte 6 sont contraignantes uniquement si $\sum_{w=u+1}^{v} (d_{i,w} + g_{i,w}) = 0$, et $d_{i,u} = d_{j,u} = 1$ ou $g_{i,u} = g_{j,u} = 1$. Dans ce cas, elles imposent bien $d_{j,v} = g_{j,v} = 0$. □

Remarque 16. Le programme linéaire précédent contient $O(n^2)$ variables et $O(n^4)$ contraintes.

Un script Python [2] permet d'obtenir le programme linéaire au format LPSolve associé aux poids $p_1, \ldots, p_n$. La résolution s'effectue à priori en temps exponentiel puisqu'il s'agit de programmation entière. En pratique, des tests sommaires effectués avec le solveur LPSolve démontre des performances inférieures à celles obtenues avec l'algorithme Smallest relâché de la partie 3.3.

Il est également intéressant de noter que la relaxation réelle du programme précédent admet systématiquement une solution optimale nulle (obtenue en instanciant toutes les variables à 1/2). Par conséquent, il ne semble pas pertinent d'étudier la relaxation réelle pour en déduire des résultats sur la relaxation entière.

2. http://perso.ens-lyon.fr/yassine.hamoudi/wp-content/uploads/2015/08/optimLPSolve.py

4 Cas particulier : les poids tous égaux

On s'intéresse ici à une variante du problème initial, dans laquelle tous les poids ont la même valeur. Nous allons présenter deux algorithmes gloutons calculant des mobiles optimaux et s'exécutant en temps polynomial. Ces deux algorithmes vont également nous conduire à étudier et résoudre une suite récurrente particulière.

Problème 8. On appelle All Equal Balanced Mobile(n, a) (abrégé AEBM(n, a)) le problème consistant à trouver un mobile optimal dont les n feuilles comportent toutes le poids a.

Le lemme suivant nous permet de restreindre notre étude aux feuilles de poids un.

Lemme 17. Soit $\{p_1, \ldots, p_n\}$ un ensemble de poids et k un de leurs diviseurs communs. Le déséquilibre de chaque mobile construit sur ces poids est égal à k fois le déséquilibre obtenu en remplaçant chaque poids p par p/k (et réciproquement, pour tout mobile construit sur $\{p_1/k, \ldots, p_n/k\}$, le déséquilibre est multiplié par k lorsqu'on remplace chaque poids p par $k \cdot p$).

Exemple 18. Le mobile de droite est obtenu en multipliant chaque poids du mobile de gauche par 3. Chaque déséquilibre local est alors également multiplié par 3.

Corollaire 19. Soit $\Delta_{n,a}$ le déséquilibre d'un mobile solution au problème AEBM(n, a). On a $\Delta_{n,a} = a \cdot \Delta_{n,1}$.

On notera par la suite $\Delta_{n,1} = \Delta_n$ le déséquilibre minimum pour placer n poids égaux à un. On peut d'ores et déjà remarquer que pour tout k, on a $\Delta_{2^k} = 0$. En effet, il suffit de placer les poids aux feuilles de l'arbre parfait de profondeur k. La Figure 6 illustre le cas $k = 3$.

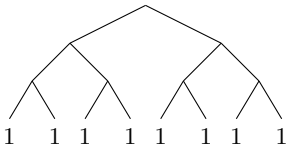


Figure 6 – Équilibrage de 8 poids égaux à un. Le déséquilibre global est nul.

4.1 Algorithme de partition

Le premier algorithme consiste à séparer les n poids en deux ensembles de tailles identiques à une unité près (chaque ensemble définit un sous-arbre), puis à répéter cette opération sur chacun d'entre eux.

On note AEBM-Part(n) le déséquilibre obtenu par cet algorithme. La procédure est détaillée dans l'algorithme 3 ci-dessous.

Algorithme 3 : Algorithme de partition AEBM-Part

Entrées : nombre n de poids (tous égaux à un)
Sorties : déséquilibre AEBM-Part(n)
si $n = 1$ **alors**
  Renvoyer 0
sinon
  si $n = 2m + 1$ **alors**
    Renvoyer $1 +$ AEBM-Part$(m) +$ AEBM-Part$(m + 1)$
  sinon
    Renvoyer $2 \cdot$ AEBM-Part$(n/2)$

La Figure 7 représente un mobile de déséquilibre 5 construit selon cet algorithme.

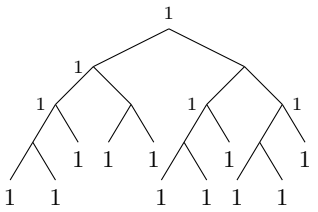


FIGURE 7 – Équilibrage de 11 poids égaux à un selon AEBM-PART.

Il reste à prouver que les déséquilibres obtenus par l'algorithme AEBM-PART sont optimaux.

Lemme 20. Pour tout $n \geq 4$, on AEBM-PART$(n) < n - 2$.

Démonstration. Ce résultat s'obtient facilement par récurrence sur n, à partir de la définition inductive : AEBM-PART$(1) = 0$, AEBM-PART$(2n) = 2 \cdot$AEBM-PART(n) et AEBM-PART$(2n+1) =$ AEBM-PART(n) + AEBM-PART$(n+1) + 1$. □

Théorème 21. L'algorithme AEBM-PART construit des mobiles optimaux lorsque tous les poids sont égaux.

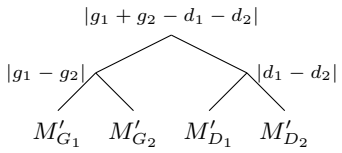
Démonstration. Considérons un mobile optimal M à n feuilles de poids un (ainsi $\Delta_M = \Delta_n$). On donne une *procédure récursive* pour transformer M en un mobile M_0 construit selon l'algorithme AEBM-PART (aucun déséquilibre local ne doit excéder un), tout en conservant $\Delta_M = \Delta_{M_0}$.

Si $n \leq 3$, aucune modification n'est nécessaire. On suppose donc $n \geq 4$. Notons M_G le sous-arbre gauche de M et M_D son sous-arbre droit. On appelle g le nombre de feuilles de M_G et d celui de M_D (on a donc $g + d = n$).

Si $|g - d| \leq 1$, le déséquilibre local à la racine de M est déjà inférieur à un, on applique alors la procédure à M_G et M_D pour obtenir M_0

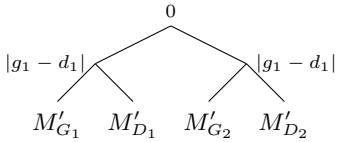
Supposons donc $|g - d| > 1$. Si $g = 1$ ou $d = 1$ alors $\Delta_n \geq n - 2$ (le déséquilibre local à la racine de M est $n - 2$). Or, d'après le Lemme 20, AEBM-PART$(n) < n - 2$. Par conséquent, le cas $g = 1$ ou $d = 1$ est à exclure. On transforme alors M_G et M_D en M'_G et M'_D selon la procédure, puis on note M'_{G_1} et M'_{D_1} les sous-arbres gauches de M'_G et M'_D, et M'_{G_2}, M'_{D_2} leurs sous-arbres droits (ces quatre arbres existent car $g, d \geq 1$).

Soit g_1, g_2, d_1 et d_2 le nombre de feuilles de chacun des quatre arbres précédents (on a $g_1 + g_2 = g$ et $d_1 + d_2 = d$). D'après la procédure, $|g_1 - g_2| \leq 1$ et $|d_1 - d_2| \leq 1$. On a le mobile M' suivant :



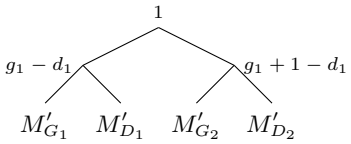
Il y a alors quatre cas possibles (à symétrie près) :

— $g_1 = g_2$ et $d_1 = d_2$. Le déséquilibre global de M' est $\Delta_M = \Delta_{M'} = \Delta_{M'_{G_1}} + \Delta_{M'_{G_2}} + \Delta_{M'_{D_1}} + \Delta_{M'_{D_2}} + |2g_1 - 2d_1|$. On transforme alors M' en M'' de la manière suivante :



Le déséquilibre est : $\Delta_{M''} = \Delta_{M'_{G_1}} + \Delta_{M'_{G_2}} + \Delta_{M'_{D_1}} + \Delta_{M'_{D_2}} + 2\,|g_1 - d_1| = \Delta_M$. La racine de M'' a bien un déséquilibre inférieur à un. On applique ensuite la procédure aux sous-arbres gauche et droit de M'' pour obtenir M_0.

— $g_2 = g_1 + 1$, $d_1 = d_2$ et $g_1 > d_1$. Le déséquilibre de M' est : $\Delta_{M'} = \Delta_{M'_{G_1}} + \Delta_{M'_{G_2}} + \Delta_{M'_{D_1}} + \Delta_{M'_{D_2}} + 1 + (2g_1 + 1 - 2d_1)$. On transforme alors M' en M'' de la manière suivante :



Le déséquilibre est $\Delta_{M''} = \Delta_{M'_{G_1}} + \Delta_{M'_{G_2}} + \Delta_{M'_{D_1}} + \Delta_{M'_{D_2}} + 2 + 2(g_1 - d_1) = \Delta_M$. La racine de M'' a bien un déséquilibre inférieur à un. On applique ensuite la procédure aux sous-arbres gauche et droit de M'' pour obtenir M_0.

— $g_2 = g_1 + 1$, $d_1 = d_2$ et $g_1 < d_1$. On effectue un raisonnement similaire.

— $g_2 = g_1 + 1$, $d_2 = d_1 + 1$ et $g_1 < d_1$. On effectue un raisonnement similaire.

On peut donc transformer récursivement tout mobile optimal M en un mobile M_0 construit selon AEBM-Part et vérifiant $\Delta_M = \Delta_{M_0}$. Par conséquent, l'algorithme AEBM-Part produit bien des mobiles optimaux. □

La preuve précédente nous permet également d'affirmer $\Delta_n = \text{AEBM-Part}(n)$. L'entier Δ_n vérifie donc la même relation de récurrence que AEBM-Part(n).

Théorème 22. On a :

$$\begin{cases} \Delta_1 = 0 \\ \Delta_{2n} = 2\Delta_n \\ \Delta_{2n+1} = 1 + \Delta_n + \Delta_{n+1} \end{cases}$$

4.2 Algorithme Smallest

Afin d'obtenir un second algorithme pour AEBM, on commence par déterminer une expression alternative de Δ_n :

Théorème 23. Soit $|n|_0$ le nombre de 0 dans l'écriture en binaire de n, et $|n|_1$ le nombre de 1. On a :

$$\begin{cases} \Delta_1 = 0 \\ \Delta_{n+1} = \Delta_n + |n|_0 - |n|_1 + 1 \end{cases}$$

Démonstration. Notons $\lambda_n = \Delta_{n+1} - \Delta_n$. On démontre par récurrence sur n que $\lambda_n = |n|_0 - |n|_1 + 1$.

Si $n = 1$ alors $\lambda_1 = \Delta_2 - \Delta_1 = 0 - 0 = |1|_0 - |1|_1 + 1$.

Soit $n > 1$, il y a 2 cas :

— Si $n = 2m$ (n pair) alors :

$$\begin{aligned} \lambda_{2m} &= \Delta_{2m+1} - \Delta_{2m} \\ &= 1 + \Delta_m + \Delta_{m+1} - 2\Delta_m && \text{d'après le Théorème 22} \\ &= 1 + \lambda_m \\ &= 1 + |m|_0 - |m|_1 + 1 && \text{par hypothèse de récurrence} \end{aligned}$$

Or $|n|_0 = |2m|_0 = |m|_0 + 1$ et $|n|_1 = |2m|_1 = |m|_1$. On a donc bien $\lambda_n = \lambda_{2m} = |2m|_0 - |2m|_1 + 1 = |n|_0 - |n|_1 + 1$.

— Le cas $n = 2m + 1$ (n impair) peut être prouvé avec un raisonnement similaire.

□

La relation de récurrence précédente va permettre de démontrer que l'algorithme Smallest produit des mobiles optimaux lorsque tous les poids sont égaux. On caractérise au préalable certains mobiles construits selon cet algorithme.

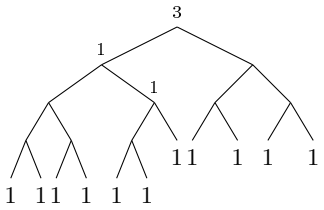
Lemme 24. Soit $2^k + m$ poids tous égaux à un (avec $m < 2^k$). Le mobile obtenu selon la procédure suivante peut être construit par l'algorithme Smallest :

— considérer l'arbre binaire parfait à 2^k feuilles

— remplacer les m feuilles les plus à gauches par des arbres binaires à deux feuilles

— placer les poids sur les $2^k + m$ feuilles de cet arbre.

Exemple 25. Lorsqu'il y a 11 éléments à placer, la procédure du Lemme 24 construit le mobile suivant :



Théorème 26. L'algorithme SMALLEST construit des mobiles optimaux lorsque tous les poids sont égaux.

Démonstration. On utilise les mobiles décrits dans le Lemme 24. Notons AEBM-SMALLEST(n) le déséquilibre obtenu par l'algorithme SMALLEST lorsque tous les poids sont égaux, et montrons qu'il vérifie la relation de récurrence du Théorème 23.

Si n est une puissance de deux, on a bien AEBM-SMALLEST$(n) = 0$.

Soit $n = 2^k + m$ $(0 \leq m < 2^k)$. On considère le mobile à n feuilles construit selon le Lemme 24, et on remplace la feuille à hauteur k la plus à gauche par un arbre binaire à deux feuilles (pour obtenir le mobile à $n+1$ feuilles du Lemme 24). En dehors du chemin menant de cette feuille à la racine, les déséquilibres locaux sont inchangés. En chaque nœud de ce chemin :

— si la feuille remplacée se trouve dans le sous-arbre droit du nœud, alors le déséquilibre local diminue de 1

— si la feuille est dans le sous-arbre gauche, il augmente de 1.

Les valeurs +1 et -1 de la Figure 8 illustrent l'évolution des déséquilibres locaux lorsque l'on passe de 9 à 10 feuilles, puis de 10 à 11.

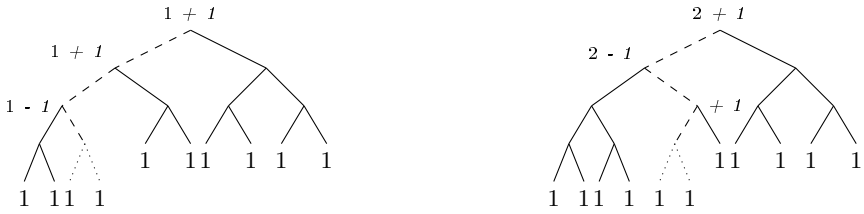


FIGURE 8 – Évolution du déséquilibre lors du passage de 9 à 10 feuilles, puis de 10 à 11 feuilles.

Autrement dit, sur le chemin étudié, il faut ajouter +1 au déséquilibre global AEBM-SMALLEST(n) lorsque l'on va à gauche et -1 lorsque l'on va à droite pour obtenir AEBM-SMALLEST$(n+1)$. Si on numérote sur k bits les nœuds du niveau k (le nœud le plus à gauche est numéroté par k zéros, celui le plus à droite par k uns), alors le déséquilibre à ajouter est égal au nombre de zéros moins le nombre de uns dans l'écriture en binaire de m sur k bits.

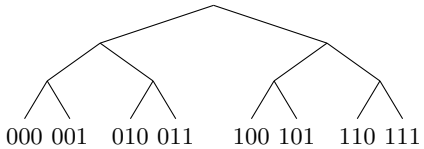


FIGURE 9 – Numérotation sur 3 bits des nœuds du niveau 3. Lorsque l'on passe de 9 à 10 feuilles le déséquilibre augmente de $+1 = |001|_0 - |001|_1$ puisque l'on dédouble le nœud 001.

Le déséquilibre à ajouter à AEBM-SMALLEST(n) pour obtenir AEBM-SMALLEST$(n+1)$ est ainsi $\left|2^k+m\right|_0 - \left|2^k+m\right|_1 + 1$ (le +1 permet d'éliminer le bit de poids fort de 2^k+m, et conserver l'écriture en binaire sur k bits de m).

On a donc bien AEBM-SMALLEST$(n+1)$ = AEBM-SMALLEST$(n) + |n|_0 - |n|_1 + 1$. On en déduit AEBM-SMALLEST$(n) = \Delta_n$, l'algorithme SMALLEST est optimal lorsque tous les poids sont égaux. □

4.3 Étude du déséquilibre Δ_n

Les Théorèmes 22 et 23 fournissent deux définitions récursives différentes du déséquilibre Δ_n :

$$\begin{cases} \Delta_1 = 0 \\ \Delta_{2n} = 2\Delta_n \\ \Delta_{2n+1} = 1 + \Delta_n + \Delta_{n+1} \end{cases} \quad \text{et} \quad \begin{cases} \Delta_1 = 0 \\ \Delta_{n+1} = \Delta_n + |n|_0 - |n|_1 + 1 \end{cases}$$

On obtient alors facilement les inégalités suivantes sur Δ_n :

Lemme 27. On a :

- Pour tout entiers a et b, $\Delta_{a+b} \leq \Delta_a + \Delta_b + |a-b|$
- Pour tout n, $\begin{cases} \Delta_n \leq \frac{1}{2}(n-1) \text{ si } n \text{ est impair} \\ \Delta_n \leq \frac{1}{2}(n-2) \text{ si } n \text{ est pair} \end{cases}$
- Pour tout $2^k \leq n < 2^{k+1}, |\Delta_{n+1} - \Delta_n| \leq k$

La Figure 10 montre l'évolution de Δ_n pour $n \in [\![1, 64]\!]$. On constate que cette courbe définit plusieurs zones, délimitées par les puissances de deux (pour lesquelles le déséquilibre est nul). En effet, un raisonnement par récurrence démontre que dans l'intervalle $[2^k, 2^{k+1}]$:

- dans $[2^k, 2^k + 2^{k-2}]$: la courbe suit une phase transitoire depuis $\Delta_{2^k} = 0$ jusqu'à $\Delta_{2^k+2^{k-2}} = 2^{k-1}$.
- dans $[2^k + 2^{k-2}, 2^k + 2^{k-1}]$: la courbe reproduit l'évolution qu'elle a eu dans l'intervalle $[2^{k-2}, 2^{k-1}]$, translatée vers le haut de 2^{k-1}.
- dans $[2^k + 2^{k-1}, 2^{k+1}]$, la courbe reproduit symétriquement l'évolution qu'elle a eu dans l'intervalle $[2^k, 2^k + 2^{k-1}]$.

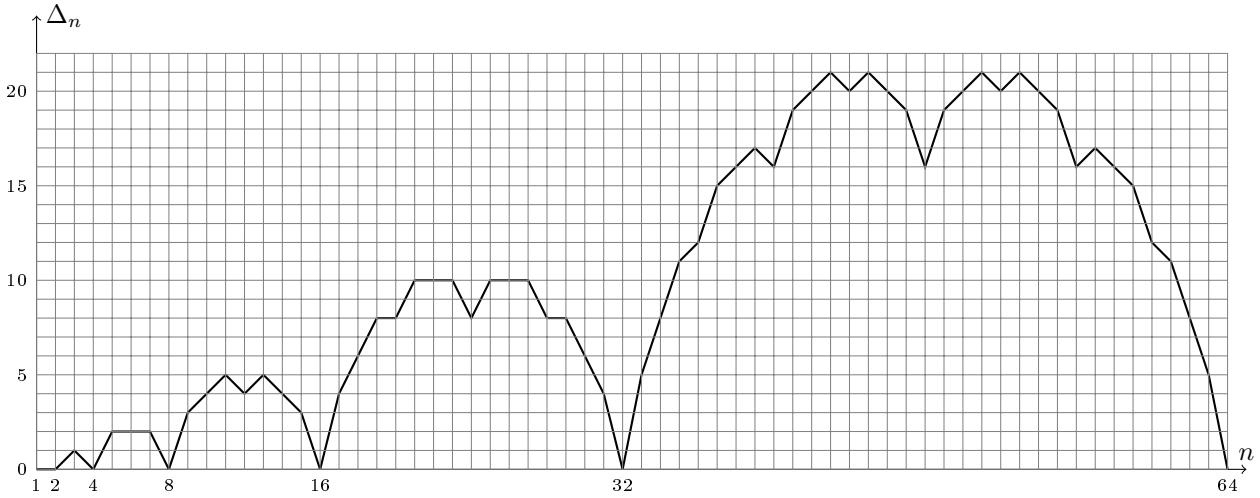


FIGURE 10 – Évolution du déséquilibre Δ_n.

Les symétries précédentes permettent également de caractériser les maximums locaux de la courbe dans chaque intervalle $[2^k, 2^{k+1}]$.

Lemme 28. Pour tout nombre $k \geq 2$ pair, dans l'intervalle $[2^k, 2^{k+1}]$ la courbe atteint son maximum en $3 \cdot 3^{k/2-1}$ points. Ces maximums correspondent aux "plateaux" que l'on observe sur la Figure 10 :  .

Lemme 29. Pour tout nombre $k \geq 1$ impair, dans l'intervalle $[2^k, 2^{k+1}]$ la courbe atteint son maximum en $2^{(k-1)/2}$ points. Ces maximums correspondent aux "pics" que l'on observe sur la Figure 10 :  .

Enfin, on énonce deux expressions alternatives de Δ_n. Le Théorème 23 donne cette première égalité :

$$\Delta_n = n - 1 + \sum_{i=1}^{n-1} |i|_0 - |i|_1$$

Le théorème suivant énonce une expression plus élégante.

Théorème 30. Si $b_k b_{k-1} \dots b_0$ est l'expression en binaire de l'entier n, alors :

$$\Delta_n = 2 \cdot (n \mod 2^k) + \sum_{i=0}^{k-1} (-1)^{b_i} \cdot (n \mod 2^{i+1})$$

Démonstration. On démontre ce résultat par récurrence sur n en utilisant le Théorème 22.

On constate que l'expression est correcte pour $n \leq 8$.

Soit $b_k b_{k-1} \dots b_0$ l'expression en binaire d'un entier $n > 8$, il y a deux cas possibles :

— Si $n = 2m$ (n pair) alors $b_0 = 0$. D'après le Théorème 22, on a $\Delta_n = 2 \cdot \Delta_m$. Donc, par hypothèse de récurrence :

$$\begin{aligned}
\Delta_n &= 2 \cdot \left(2 \cdot (m \mod 2^{k-1}) + \sum_{i=0}^{k-2} (-1)^{b_{i+1}} \cdot (m \mod 2^{i+1}) \right) \\
&= 4 \cdot b_{k-1} \dots b_1 + 2 \cdot \sum_{i=0}^{k-2} (-1)^{b_{i+1}} \cdot b_{i+1} \dots b_1 \\
&= 2 \cdot b_{k-1} \dots b_1 0 + \sum_{i=0}^{k-2} (-1)^{b_{i+1}} \cdot b_{i+1} \dots b_1 0 \\
&= 2 \cdot (n \mod 2^k) + \sum_{i=0}^{k-2} (-1)^{b_{i+1}} \cdot (n \mod 2^{i+2}) \\
&= 2 \cdot (n \mod 2^k) + \sum_{i=0}^{k-1} (-1)^{b_i} \cdot (n \mod 2^{i+1})
\end{aligned}$$

— Si $n = 2m + 1$ (n impair). Ce cas est d'avantage calculatoire. La preuve peut être consultée en ligne [3].

□

5 Cas particulier : les puissances de deux

Il a été démontré précédemment que l'algorithme SMALLEST construit des mobiles optimaux lorsque le déséquilibre minimum est au plus un (cf partie 3.2) ou lorsque tous les poids sont égaux (cf partie 4). Cela se produit également dans un cas beaucoup plus général, lorsque les poids sont des puissances de deux.

Théorème 31. L'algorithme SMALLEST construit des mobiles optimaux lorsque tous les poids sont des puissances de deux.

Le Lemme 17 permet d'étendre la portée de ce résultat, et d'englober ainsi complètement le Théorème 26 dans l'énoncé suivant.

Théorème 32. Soit $\{p_1, \dots, p_n\}$ un ensemble de poids. S'il existe un entier p tels que $p_1/q, \dots, p_n/q$ sont des puissances de deux, alors l'algorithme SMALLEST construit un mobile optimal sur les poids précédents.

3. http://perso.ens-lyon.fr/yassine.hamoudi/wp-content/uploads/2015/08/Delta.pdf

Le Théorème 31 est démontré dans la suite de cette partie. La Figure 11 donne quelques exemples de poids auxquels s'appliquent les Théorèmes 31 et 32.

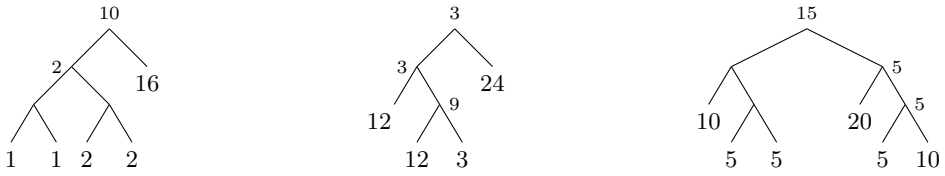


FIGURE 11 – Mobiles construits selon l'algorithme SMALLEST et optimaux d'après les Théorèmes 31 et 32.

Par ailleurs, la Figure 12 démontre que ce résultat n'est plus valide si on utilise des puissances de trois par exemple.

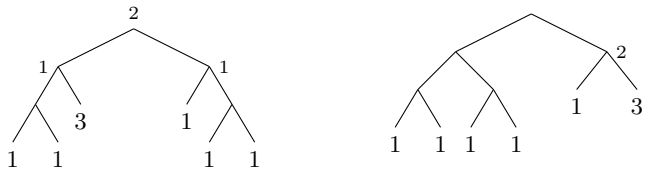


FIGURE 12 – A gauche, un mobile de déséquilibre 4 construit selon l'algorithme SMALLEST. A droite, un mobile optimal construit sur les mêmes poids.

5.1 Les mobiles témoins

La preuve utilise des mobiles particuliers, définis ci-dessous.

Définition 33. On appelle *mobile témoin* tout mobile optimal M construit sur des puissances de deux et vérifiant les conditions suivantes :
- M n'est pas construit selon l'algorithme SMALLEST
- le déséquilibre de M est *strictement inférieur* à celui obtenu par l'algorithme SMALLEST sur les mêmes poids.

Le Théorème 31 affirme qu'il n'existe pas de mobile témoin. Afin de démontrer ce résultat, menons un raisonnement par l'absurde et supposons le contraire.

Pour tout mobile témoin A, on note $\max A$ le plus grand poids porté par une feuille de A. Soit alors m_0 le *plus petit* entier parmi l'ensemble des $\max A$.

On choisit un mobile témoin particulier, noté M par la suite, vérifiant les hypothèses suivantes :
- $\max M = m_0$
- M a le plus petit nombre de feuilles parmi l'ensemble des mobiles témoins A tels que $\max A = m_0$.

Soit M_G et M_D les sous-arbres gauche et droit de M. On peut effectuer les hypothèses suivantes sur ceux-ci.

Lemme 34. On a $m_0 \geq 2$.

Démonstration. D'après le Théorème 26, il n'existe pas de mobile témoin dont tous les poids sont égaux à un. □

Lemme 35. On peut supposer que M_G et M_D sont construits selon l'algorithme SMALLEST.

Démonstration. On a $\max M_G \leq m_0$ et M_G comporte strictement moins de feuilles que M. Par conséquent, M_G n'est pas un mobile témoin (sinon il contredit les hypothèses de minimalité faites sur M). Ainsi, si on remplace M_G par un mobile construit selon l'algorithme SMALLEST sur les mêmes poids, on n'augmente pas le déséquilibre global et M vérifie toujours les hypothèses de minimalité.

Il en va de même pour M_D. □

Lemme 36. M comporte au moins une feuille de poids un.

Démonstration. Si ce n'est pas le cas, on peut diviser tous les poids de M par deux et obtenir un mobile témoin M' tel que $\max M' \leq m_0/2$ d'après le Lemme 17. □

Lemme 37. M_G et M_D comportent au plus une feuille de poids un chacun.

Démonstration. Supposons qu'il y a au moins deux feuilles de poids un dans M_G (le raisonnement est identique pour M_D). D'après le Lemme 35, M_G est construit selon l'algorithme SMALLEST. Il y a donc au moins deux feuilles de poids un qui sont frères dans M_G. Notons M' le mobile M dans lequel deux feuilles frères et de poids un sont remplacées par une unique feuille de poids deux.

Il est clair que M' est également un mobile témoin. Par ailleurs, d'après le Lemme 34, l'opération précédente n'a pas augmenté le maximum : $\max M = \max M' = m_0$. Or M' comporte une feuille de moins que M, on contredit donc les hypothèses de minimalité faites sur M. □

Lemme 38. Si M_G (resp. M_D) a une feuille de poids un, alors M_G (resp. M_D) a au moins deux feuilles.

Démonstration. Supposons qu'il y a une unique feuille dans M_G, de poids un. Notons $p_1 \leq \cdots \leq p_n$ les poids de M (on a $p_1 = 1$). Le déséquilibre de M est alors $\Delta_M = \left(\sum_{i=2}^{n} p_i - 1\right) + \Delta_{M_D}$.

Soit M' le mobile construit à partir de M de la manière suivante :
— prendre le mobile M_D
— remplacer la feuille de poids p_2 par un arbre binaire à deux feuilles
— placer sur ces deux feuilles les poids p_1 et p_2

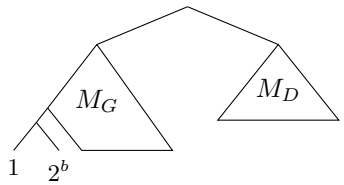
Cette opération crée un déséquilibre local $p_2 - 1$ et augmente chaque déséquilibre local de M_D d'au plus un. Or il y a $n-2$ nœuds dans M_D. On a donc : $\Delta_{M'} \leq (p_2 - 1) + \Delta_{M_D} + n - 2$.

D'après le Lemme 37, il y a au plus deux poids égaux à un dans M. Pour $n \geq 3$ on a donc $(p_2 - 1) + n - 2 < \sum_{i=2}^{n} p_i - 1$. Ainsi, $\Delta_{M'} < \Delta_M$. Or, M est un mobile témoin, donc il est optimal. L'hypothèse de départ est donc fausse. □

Les Lemmes 36 et 37 laissent alors deux possibilités : soit M comporte une unique feuille de poids un, soit il y en a exactement une dans chaque sous-arbre M_G et M_D. On étudie chacune de ces deux possibilités dans les deux sections suivantes.

5.2 Une unique feuille de poids un

Supposons que M contienne une unique feuille de poids un, présente dans M_G par exemple. D'après les Lemmes 35 et 38, le mobile M est de la forme suivante :



Notons alors M^+ le mobile obtenu à partir de M en remplaçant la feuille de poids un par une feuille de poids deux, et M^- en enlevant la feuille de poids un (la feuille de poids 2^b remonte donc dans l'arbre).

FIGURE 13 – A gauche, M^+. A droite, M^-.

Lemme 39. M^+ et M^- ne sont pas des mobiles témoins.

Démonstration. Si M^+ est un mobile témoin, on peut diviser tous ses poids par deux et obtenir un mobile témoin M'^+ tel que $\max M'^+ \leq m_0/2$ d'après le Lemme 17. Ceci contredit les hypothèses de minimalité faites sur M.

Il en va de même pour M^-. □

Lemme 40. On a : $\Delta_M = \frac{1}{2}\Delta_{M^+} + \frac{1}{2}\Delta_{M^-} + 2^{b-1}$.

Démonstration. Notons δ les déséquilibres locaux de M, δ^+ ceux de M^+ et δ^- ceux de M^-.

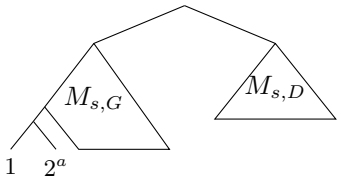
En dehors du chemin menant de la feuille de poids un à la racine, les déséquilibres locaux en chaque nœud ν sont égaux : $\delta(\nu) = \delta^+(\nu) = \delta^-(\nu)$. On a donc $\delta(\nu) = \frac{1}{2}\delta^+(\nu) + \frac{1}{2}\delta^-(\nu)$.

Pour tout nœud ν du chemin, le sous-arbre contenant la feuille de poids un a un poids impair, alors que l'autre sous-arbre a un poids pair. Par conséquent $\delta(\nu) > 0$. Ceci implique $\delta^+(\nu) = \delta(\nu)+1$ et $\delta^-(\nu) = \delta(\nu) - 1$, ou l'inverse ($\delta^+(\nu) = \delta(\nu) - 1$ et $\delta^-(\nu) = \delta(\nu) + 1$). On a donc forcément $\delta(\nu) = \frac{1}{2}\delta^+(\nu) + \frac{1}{2}\delta^-(\nu)$.

Il y a un cas particulier pour le père ν_0 de la feuille de poids un. Puisque ce nœud disparaît dans M^- il faut ajouter le déséquilibre $\delta^-(\nu_0)$ manquant. On a ainsi $\delta(\nu_0) = \frac{1}{2}\delta^+(\nu_0) + \frac{1}{2} \cdot 2^b$.

On somme finalement l'ensemble des déséquilibres locaux. On obtient bien $\Delta_M = \frac{1}{2}\Delta_{M^+} + \frac{1}{2}\Delta_{M^-} + 2^{b-1}$. □

On définit maintenant le mobile M_s construit par l'algorithme Smallest sur les mêmes poids que M. Le mobile M_s est de la forme suivante :



On note à nouveau M_s^+ et M_s^- les mobiles obtenus à partir de M_s en remplaçant la feuille de poids un par une feuille de poids deux, ou en l'enlevant.

Lemme 41. M_s^+ et M_s^- sont construits selon l'algorithme Smallest.

Démonstration. Notons $p_1 < p_2 \leq \cdots \leq p_n$ les poids de M_s (on sait que $p_1 = 1$ et tous les poids sont des puissances de deux). Lorsque l'algorithme Smallest compare deux sommes de poids P_1 et P_2, le résultat de la comparaison est identique si la valeur de p_1 est changée en 0 ou 2. En effet, supposons par exemple que P_1 contienne le poids p_1. Puisque tous les autres poids sont des puissances de deux supérieures à deux, la valeur de P_1 est impaire et celle de P_2 est paire. Autrement dit, P_1 et P_2 ne sont pas égaux. Ainsi, $P_1 \leq P_2 \Leftrightarrow P_1 \pm 1 \leq P_2$ et $P_2 \leq P_1 \Leftrightarrow P_2 \leq P_1 \pm 1$.

Par conséquent, les résultats des comparaisons effectuées par l'algorithme Smallest pour construire le mobile M_s ne sont pas modifiés si la valeur de p_1 est changée en 0 ou 2 (dans le premier cas le poids p_1 est retiré). Les mobiles M_s^+ et M_s^- sont donc bien construits selon selon l'algorithme Smallest. □

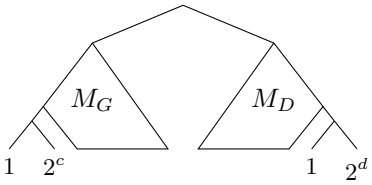
Lemme 42. On a $\Delta_{M_s} = \frac{1}{2}\Delta_{M_s^+} + \frac{1}{2}\Delta_{M_s^-} + 2^{a-1}$.

On peut maintenant conclure le cas où M contient une unique feuille de poids un. D'après le Lemme 39, M^+ et M^- ne sont pas des mobiles témoins. Or, selon le Lemme 41, M_s^+ et M_s^- sont construits par l'algorithme Smallest. Donc $\Delta_{M_s^+} \leq \Delta_{M^+}$ et $\Delta_{M_s^-} \leq \Delta_{M^-}$. De plus, par définition de l'algorithme Smallest, 2^a est le deuxième plus petit poids du mobile M, donc $2^a \leq 2^b$. D'après les Lemmes 40 et 42, on a donc $\Delta_{M_s} = \frac{1}{2}\Delta_{M_s^+} + \frac{1}{2}\Delta_{M_s^-} + 2^{a-1} \leq \frac{1}{2}\Delta_{M^+} + \frac{1}{2}\Delta_{M^-} + 2^{b-1} = \Delta_M$.

Or, par définition d'un mobile témoin, $\Delta_{M_s} > \Delta_M$. On aboutit donc à une absurdité.

5.3 Deux feuilles de poids un

Supposons que M contienne exactement une feuille de poids un dans chaque sous-arbre M_G et M_D. Le mobile M est de la forme suivante :



Soit M^+ le mobile obtenu à partir de M en remplaçant la feuille de poids un dans M_G par une feuille de poids deux, et en enlevant celle de M_D. Soit M^- le mobile obtenu avec l'opération inverse.

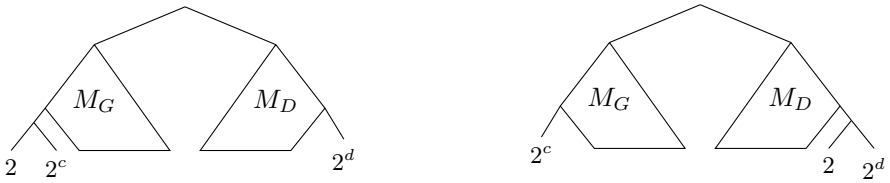


FIGURE 14 – A gauche, M^+. A droite, M^-.

On retrouve plusieurs lemmes similaires à ceux de la partie précédente.

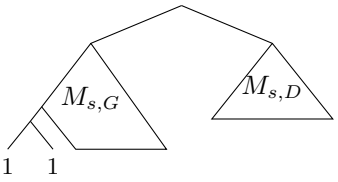
Lemme 43. M^+ et M^- ne sont pas des mobiles témoins.

Lemme 44. On a : $\Delta_M \geq \frac{1}{2}\Delta_{M^+} + \frac{1}{2}\Delta_{M^-} + 2^{c-1} + 2^{d-1} - 2$.

Démonstration. On étudie les déséquilibres locaux, comme dans le Lemme 40.

La racine ν_0 de M est ici un autre cas particulier. On sait uniquement que $\delta^+(\nu_0), \delta^-(\nu_0) \leq \delta(\nu_0) + 2$. Par conséquent, $\delta(\nu_0) \geq \frac{1}{2}\delta^+(\nu_0) + \frac{1}{2}\delta^+(\nu_0) - 2$. □

On définit maintenant le mobile M_s produit par l'algorithme SMALLEST sur les mêmes poids que M. Le mobile M_s est de la forme suivante :



On note M'_s le mobile obtenu à partir de M_s en remplaçant les deux frères de poids un par une unique feuille de poids deux (les mobiles M'_s, M^+ et M^- ont donc les mêmes feuilles).

Lemme 45. M'_s est construit selon l'algorithme SMALLEST.

Lemme 46. On a $\Delta_{M_s} = \Delta_{M'_s}$.

On peut maintenant conclure le cas où M a une feuille de poids un dans chaque sous-arbre M_G et M_D. En effet, M^+ et M^- ne sont pas des mobiles témoins, ainsi $\Delta_{M'_s} \leq \Delta_{M^+}$ et $\Delta_{M'_s} \leq \Delta_{M^-}$. Selon les Lemmes 44 et 46, on a donc $\Delta_{M_s} = \frac{1}{2}\Delta_{M'_s} + \frac{1}{2}\Delta_{M'_s} \leq \frac{1}{2}\Delta_{M^+} + \frac{1}{2}\Delta_{M^-} + 2^{b-1} + 2^{c-1} - 2 \leq \Delta_M$.

Or, par définition d'un mobile témoin, $\Delta_{M_s} > \Delta_M$. On aboutit donc à une absurdité. Ceci conclut la preuve du Théorème 31.

6 Conclusion

Nous avons exposé dans ce rapport de stage plusieurs résultats concernant le problème Balanced Mobile. Deux algorithmes exponentiels, dont un programme linéaire en nombres entiers, ont été présentés. Notre étude s'est portée tout particulièrement sur l'algorithme glouton Smallest qui s'exécute en $O(n \log n)$. Bien que cet algorithme ne construit pas systématiquement des mobiles optimaux, nous avons pu caractériser plusieurs situations où c'était tout de même le cas.

De nombreuses questions restent toutefois en suspens. En particulier, nous ne savons pas si le problème Balanced Mobile est NP-complet. Nous avons essayé, sans succès, des réductions depuis 2-Partition et 3-SAT. Cependant, les nombreux cas que nous avons pu couvrir avec l'algorithme Smallest laissent ouverte la possibilité d'avoir un algorithme polynomial. Nous pensons notamment que l'étude des techniques développées dans la preuve du Théorème 31 présente un grand intérêt dans la compréhension du problème et devrait être poursuivie d'avantage.

Enfin, il serait également intéressant d'étudier d'autres variantes au problème Balanced Mobile. Par exemple, étant donné un arbre binaire T à n feuilles et des poids $p_1, \ldots, p_n$, on peut rechercher la répartition des poids sur les feuilles de T qui minimise le déséquilibre global. Nous n'avons pas trouvé de réponse particulière à cette question.

Références

[GKY02] Mordecai J. Golin, Claire Kenyon, and Neal E. Young. Huffman coding with unequal letter costs. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 785–791, New York, NY, USA, 2002. ACM.

[God73] Sadashiva S. Godbole. On efficient computation of matrix chain products. *IEEE Trans. Comput.*, 22(9) :864–866, September 1973.

[Huf52] D.A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9) :1098–1101, Sept 1952.

[Kar61] Richard M. Karp. Minimum-redundancy coding for the discrete noiseless channel. *Information Theory, IRE Transactions on*, 7(1) :27–38, January 1961.

Annexes

A Remarques générales sur le stage

Ce rapport a été réalisé suite à un stage de douze semaines à Paris, au sein du Laboratoire d'Informatique Algorithmique : Fondements et Applications (LIAFA). Ce travail conclue l'année de M1 et donnera lieu à une présentation orale.

Lors de la première partie du stage, nous avons cherché à rapprocher Balanced Mobile d'autres problèmes préexistants dans la littérature scientifique (notamment les codes préfixes). Nous avons également essayé de montrer que BM est NP-complet, ou de trouver un algorithme polynomial. Dans le même temps, nous avons débuté l'étude de l'algorithme Smallest, et nous avons obtenu rapidement quelques résultats le concernant (notamment les parties 3.2 et 3.3 du rapport).

Par la suite, nous avons recentré notre travail sur des instances particulières du problème BM. Nous avons démontré de nombreux résultats concernant le problème All Equal Balanced Mobile et la fonction Δ_n liée. La première preuve du Théorème 26 que nous avions trouvée utilisait plusieurs techniques reprises par la suite pour démontrer le Théorème 31.

Enfin, nous avons établi dans le dernier mois de stage le programme linéaire présenté partie 3.4 ainsi que les théorèmes exposés dans la partie 5. Nous n'avions malheureusement plus le temps pour travailler d'avantage sur ces résultats (simplifier, analyser et évaluer le programme linéaire, généraliser les preuves à d'autres situations...).

J'ai trouvé très intéressant d'étudier un problème algorithmique nouveau, et d'établir les premiers résultats le concernant. J'ai notamment été surpris par la diversité des recherches que nous avons pu mener sur un problème aussi simple en apparence. J'ai également pu assister à de nombreux séminaires, à Paris et particulièrement au LIAFA, variés et très enrichissants.

Pour finir, je remercie chaleureusement Mme Sophie Laplante et M. Roberto Mantaci pour m'avoir accueilli et encadré tout au long du stage. J'ai beaucoup apprécié leurs nombreux conseils et leur grande disponibilité.