Le tampon mélangeur

Olivier Devillers, Philippe Guigue

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Le tampon mélangeur

Olivier Devillers — Philippe Guigue

N° 3988

Août 2000

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--3988--FR

INRIA
SOPHIA ANTIPOLIS

Le tampon mélangeur

Olivier Devillers , Philippe Guigue

Thème 2 — Génie logiciel
et calcul symbolique
Projet Prisme

Rapport de recherche n° 3988 — Août 2000 — 38 pages

Résumé : Les méthodes de randomisation ont eu un grand succès en géométrie algorithmique car elles conduisent fréquemment à des algorithmes plus simples à programmer et parfois plus efficaces que leurs équivalents déterministes.

Un algorithme est randomisé lorsque celui-ci effectue des choix aléatoires pour parvenir à ses fins, pour la catégorie des algorithmes incrémentaux à laquelle nous nous intéressons, le hasard intervient, par exemple, dans l'ordre d'insertion des données.

L'analyse de tels algorithmes est alors faite en moyenne sur les différents ordres possibles. Cependant, choisir un ordre d'insertion aléatoire nécessite d'attendre l'ensemble des données avant de les insérer, ce qui fait que l'algorithme n'est plus *en ligne*.

Une solution possible consiste, cependant, à intercaler un tampon mélangeur de taille k entre le processus qui fournit les données et l'entrée de l'algorithme afin d'effectuer un mélange local de l'ordre initial nous permettant ainsi d'introduire assez de randomisation pour nous garantir une amélioration de la complexité moyenne d'algorithmes en ligne dépendant de l'ordre d'insertion.

Ainsi après avoir illustré ce type de technique sur le problème du tri, nous donnons quelques résultats obtenus pour des problèmes plus géométriques tels que la triangulation de Delaunay ou le cloisonnement vertical de segments pour lesquels il existe des algorithmes incrémentaux. Typiquement, pour un algorithme randomisé en $O(n \log n)$ ou $O(n)$, le cas le pire est au moins quadratique et l'utilisation des stratégies proposées permet d'obtenir une complexité $O(\frac{n^2 \log k}{k})$.

Mots-clés : Complexité, Algorithmes randomisés, Algorithmes en ligne, Géométrie algorithmique

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : 04 92 38 77 77 - International : +33 4 92 38 77 77 — Fax : 04 92 38 77 65 - International : +33 4 92 38 77 65

The shuffling buffer

Abstract: Randomization techniques have encountered a large succes in computational geometry since they often lead to conceptually simple algorithms, which often yield a better performance than their deterministic counterparts.

An algorithm is randomized when it has to make choices between several possibilities to reach its goal, for the class of incremental algorithms we will give a particular interest, randomness appears, for example, on the order of insertion used for the data.

Complexity analysis of such algorithms is then done by averaging over all possible orders. However, in order to choose a random order of insertion we have to wait for all the data before starting the insertions and the algorithm is no longer *on-line*.

A possible solution consists in a shuffling buffer that comes in between the process providing the input data and the algorithm. This local shuffling on the adversary order introduces enough randomness to guarantee some improvement of the expected complexity of order dependant on-line algorithms.

We will start with presenting this technique on the sorting problem, following this, we turn our attention to more geometric problems such as constructing Delaunay triangulation or determining all intersection pairs among a set of line segments in the plane for which there exists incremental algorithms. Typically, for a randomized algorithm with running time $O(n \log n)$ or $O(n)$, the worst case is at least quadratic and the use of the suggested strategies permits to obtain a $O(\frac{n^2 \log k}{k})$ complexity.

Key-words: Computational and structural complexity, Randomized algorithms, On-line algorithms, Computational geometry

1 Introduction

Les algorithmes classiques de géométrie algorithmique sont généralement complexes et difficiles à mettre en œuvre. Une des réponses possible consiste à utiliser des algorithmes plus simples, dont la complexité n'est pas optimale dans tous les cas, mais seulement en moyenne sur certains choix aléatoires fait par l'algorithme: les algorithmes randomisés [16].

Contrairement à un algorithme classique qui déduit un résultat à partir de certaines données de départ en suivant un chemin bien déterminé, l'algorithme randomisé a le choix entre plusieurs chemins pour parvenir à ses fins. Il est important de noter que seul le déroulement de l'algorithme est aléatoire et que le résultat est parfaitement déterminé [15, 13, 4, 1].

L'intervention du hasard dans ce type de méthodes réside dans la manière de choisir le chemin que va finalement suivre l'algorithme. L'analyse de l'algorithme est alors faite en moyenne sur les différents chemins possibles.

La résolution incrémentale d'un problème consiste à construire, dans une étape initiale, la solution correspondant à un petit sous ensemble de l'ensemble des données, puis à traiter une par une les autres données, en mettant à jour la solution courante. Ainsi, si l'on considère que l'ordre dans lequel les données sont fournies fait partie du problème alors de tels algorithmes n'ont rien de randomisé, par contre si l'on considère que les données sont fournies en bloc, sans ordre apparent, l'algorithme se voit obligé de choisir entre une multitude de chemins correspondant aux différents ordres possibles sur les données.

Un moyen largement utilisé consiste alors à faire une analyse randomisée de tels algorithmes incrémentaux, c'est à dire que l'on suppose l'ordre d'insertion choisi aléatoirement parmi tous les ordres possibles [10]. Aucune hypothèse n'est faite sur la répartition des données. Ainsi, les résultats obtenus restent valides pour n'importe quel ensemble de données en entrée si celui-ci est inséré selon un ordre aléatoire.

Cependant, nous nous intéressons plus particulièrement aux algorithmes incrémentaux fonctionnant en ligne, c'est-à-dire capable de maintenir la solution d'un problème, lors de l'introduction successive des données, sans avoir une connaissance *a priori* de l'ensemble des données à traiter. Ce type d'algorithmes trouvent de nombreuses applications pratiques en reconstruction 3D ou en compression géométrique, par exemple, où l'on doit traiter les données selon un ordre plus ou moins structuré que l'on ne contrôle pas (donné par un capteur laser ou par la technique de compression).

Il est clair que choisir un ordre aléatoire sur n données nécessite de connaître toutes ces données dès le début de l'algorithme ce qui fait que l'algorithme n'est plus *en ligne*. Nous allons donc nous intéresser à une solution intermédiaire qui n'utilisera ni l'ordre initial des données qui peut être mauvais, ni un ordre totalement randomisé qui nécessite d'attendre la totalité des données mais un mélange local de l'ordre donné par un adversaire. Ce mélange local nous permet d'introduire assez de randomisation pour nous garantir une amélioration de la complexité sans avoir à attendre de connaître l'ensemble des données.

Le chapitre 2 introduit le modèle du tampon mélangeur et présente différentes stratégies envisageables et les notations utilisées par la suite. L'analyse de ce type de technique est illustrée, dans un premier temps, dans le chapitre 3 par l'exemple simple du tri. Les chapitres 4 et 5 traitent de problèmes plus géométriques comme la construction incrémentale de la triangulation de Delaunay ou du cloisonnement vertical de segments. Enfin, quelques calculs complémentaires sont donnés en annexe (chapitre 7).

2 Le modèle du tampon mélangeur

Cette section expose le modèle de tampon mélangeur étudié par la suite. La technique adoptée consiste simplement à intercaler un tableau de taille k entre le processus qui fournit les données et l'entrée de l'algorithme. Cette solution nous permet de brasser localement l'ordre initial des données sans avoir à attendre l'ensemble des données.

2.1 Définitions, Notations

Les objets fournis par l'adversaire sont les éléments d'un univers $\mathcal{O}$ et constituent les données du problème.

L'entrée de l'algorithme sera un sous-ensemble fini $\mathcal{A}=\{a_i\}_{i\in\{1..n\}}$ de $\mathcal{O}$, a_i représentant le i^{eme} objet fourni par l'adversaire.

Le j^{eme} objet traité par l'algorithme sera noté x_j.

On note par $\mathcal{T}_i = \{x_j\}_{1\leq j\leq i}$ l'ensemble des objets présents dans le résultat courant après la i^{eme} insertion.

Dans la suite, $\mathcal{B}^k$ désigne un tampon mélangeur de taille k.

Enfin, l'ensemble des objets présents dans un tampon de taille k juste avant la i^{eme} insertion dans l'arbre est un sous ensemble de $\mathcal{A}$ et est noté $\mathcal{B}^{\mathrm{k}}_i$.

2.2 Première stratégie

Une première stratégie possible consiste à ranger les k premières données dans le tampon pour l'initialiser, puis à fournir ensuite ces k données à notre algorithme en vidant le tampon selon un ordre aléatoire (c'est-à-dire en leur appliquant une permutation aléatoire parmi $k!$), on renouvelle ensuite la procédure en plaçant les k données suivantes dans le tampon.

Avec les notations définies, on a ainsi:
$\mathcal{B}^{\mathrm{k}}_{pk+1} = \{a_{pk+1},\ldots,a_{(p+1)k}\}$, $\mathcal{B}^{\mathrm{k}}_{pk+2} = \{a_{pk+1},\ldots,a_{(p+1)k}\}\setminus\{x_{pk+1}\},\ldots,$
$\mathcal{B}^{\mathrm{k}}_{pk+k} = \{a_{pk+1},\ldots,a_{(p+1)k}\}\setminus\{x_{pk+1},\ldots,x_{pk+(k-1)}\}\quad \forall\ 0\leq p<\lfloor\frac{n}{k}\rfloor$

On remarque que pour un ordre imposé par l'adversaire $[a_1,\ldots,a_n]$ il existe $(k!)^{\lfloor\frac{n}{k}\rfloor}(n-\lfloor\frac{n}{k}\rfloor k)!$ ordres possibles $\mathcal{L}=[x_1,\ldots,x_n]$ pour notre algorithme.

2.3 Deuxième stratégie

Une deuxième stratégie peut consister à ranger les k premières données dans le tampon pour l'initialiser. A chaque étape on choisit ensuite aléatoirement une donnée dans le tampon pour l'insérer dans notre algorithme, mais cette fois au lieu de vider complètement le tampon, on place après chaque tirage la donnée suivante fournie par l'adversaire. A la fin le tampon est vidé selon un ordre aléatoire.

Avec les notations définies, on a cette fois:
$\mathcal{B}^{\mathrm{k}}_1 = \{a_1,\ldots,a_k\}$, $\mathcal{B}^{\mathrm{k}}_2 = \{a_1,\ldots,a_{k+1}\}\setminus\{x_1\},\ldots,$
$\mathcal{B}^{\mathrm{k}}_p = \{a_1,\ldots,a_{k+p-1}\}\setminus\{x_1,\ldots,x_{p-1}\}$, etc...

Pour un ordre imposé par l'adversaire $[a_1,\ldots,a_n]$ et pour cette nouvelle stratégie, il existe cette fois $(k)^{n-k+1}(k-1)!$ ordres possibles $\mathcal{L}=[x_1,\ldots,x_n]$ pour notre algorithme. On remarque que pour k fixé et pour un même ordre initial sur les données, cette deuxième stratégie génère plus de permutations sur les données que la première.

Il est clair que pour les deux stratégies, si $k=n$ on obtient un ordre aléatoire pour notre algorithme et que si $k=1$ l'ordre est totalement imposé par l'extérieur. L'étude va porter sur les petites valeurs de k c'est-à-dire $1<k<<n$.

Afin d'illustrer les principes généraux de ce type d'algorithmes incrémentaux et de cette nouvelle technique de randomisation, passons tout de suite à l'analyse du problème fondamental du tri de n nombres.

3 Un exemple simple: le tri

3.1 Un algorithme incrémental

Le principe de l'algorithme est le suivant: on insère une à une l'ensemble des n données dans un arbre binaire de recherche classique (sans rééquilibrage). Les noeuds de l'arbre correspondent à des intervalles et les deux fils d'un noeud à une partition de cet intervalle en deux sous intervalles. Lorsque un nouveau nombre est inséré, il est localisé dans cet arbre binaire et la feuille qui le contient (I dans l'exemple Figure 1) devient un noeud interne, son intervalle est alors coupé en deux morceaux par rapport au nouveau nombre inséré (I_1 et I_2 Figure 1).

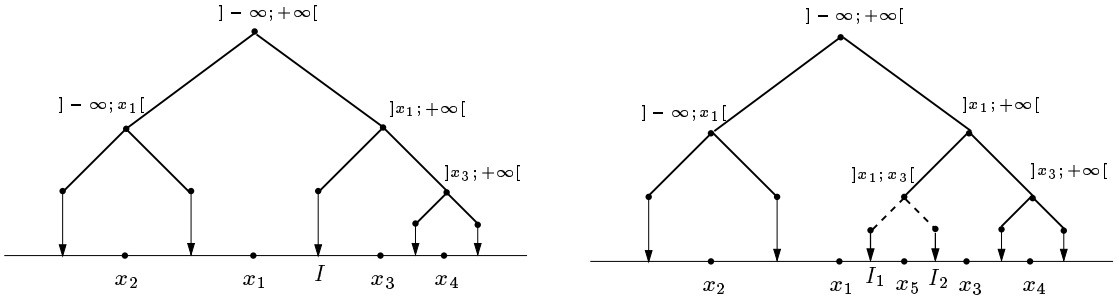


FIG. 1 – *Insertion de x_5 dans l'arbre binaire*

Analyser un algorithme revient à prévoir les ressources nécessaires à cet algorithme, le plus souvent la ressource pertinente mesurée est le temps de calcul. Notre analyse portera ici sur le nombre de comparaisons effectuées.

Il est clair que pour l'algorithme de tri présenté le temps de calcul nécessaire dépend de l'ordre d'insertion des données en entrée.

3.2 Analyse de l'insertion dans un arbre binaire dans le pire des cas

Commençons par considérer une version déterministe de cet algorithme, c'est-à-dire pour lequel l'ordre d'insertion des données dans l'arbre binaire est totalement imposé par l'adversaire.

Un utilisateur peut systématiquement employer l'algorithme sur des mauvais cas ne vérifiant pas l'hypothèse d'uniformité des différentes permutations possibles sur les données. C'est la situation d'un jeu qui oppose le programmeur et un utilisateur malicieux de l'algorithme. Le programmeur cherche à minimiser le temps d'exécution et l'utilisateur à le maximiser. Il se trouve alors qu'une analyse de complexité classique, c'est-à-dire dans le pire des cas, nous donne des résultats assez décevants. Ce type d'analyse nous donne, en effet, une borne supérieure du temps d'exécution; sa connaissance nous assure que l'algorithme ne prendra jamais plus de temps.

Theoreme 3.1 *La complexité dans le pire des cas de l'algorithme de tri est* $\Theta(n^2)$.

Preuve: Pour l'algorithme de tri présenté, le cas le pire sur les données se produit lorsque celles-ci sont fournies déjà triées (selon un ordre croissant par exemple). L'insertion des données selon cet ordre produit un arbre *dégénéré* (en forme de peigne); l'insertion d'un nouveau nombre nécessite alors la comparaison de celui-ci avec l'ensemble des nombres déjà insérés dans l'arbre.

Pour notre analyse, nous supposons connues $\mathcal{L} = [x_1,...,x_n]$ la liste des données à trier dans l'ordre de leur insertion et $\mathcal{L}_o = [b_1,...,b_n]$ le résultat correspondant, c'est-à-dire la liste des données triées.

On peut facilement montrer qu'il existe une unique insertion pour laquelle l'intervalle, correspondant à une feuille de l'arbre, et contenant à la fois b_i et b_j, est partitionné de telle manière à ce qu'aucun des deux nouveaux intervalles ne contienne b_i et b_j à la fois. Cette étape particulière ne peut se produire que lors de l'insertion d'un des $j-i+1$ nombres $b_i,b_{i+1},\ldots,b_{j-1},b_j$. On ne compare alors b_i et b_j que lorsque le nombre inséré à cette étape est b_i ou b_j. L'ordre croissant de l'adversaire nous impose d'insérer b_i avant l'un des $j-i$ nombres $b_{i+1},b_{i+2},\ldots,b_{j-1},b_j$, ainsi on compare b_i et b_j pour tout couple i,j.

Autrement dit, le nombre total de comparaisons effectuées est alors:

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} 1 = \sum_{i=1}^{n-1}(n-i) = \frac{n(n-1)}{2} = \frac{n^2}{2} - \frac{n}{2} = \Theta(n^2)$$

□

L'analyse dans le pire des cas ne permet donc pas de trouver pour notre problème de tri incrémental une borne plus fine que la borne quadratique *pessimiste* $\frac{1}{2}n^2$ qui soit valable pour toute séquence de taille n.

3.3 Analyse de l'insertion dans un arbre binaire selon un ordre aléatoire

Ce *mauvais* comportement de l'algorithme peut toutefois être évité en introduisant du hasard dans son déroulement. Pour cela, on décide d'insérer les nombres non pas dans un ordre défini imposé par un adversaire mais dans un ordre aléatoire. L'aspect randomisé est donc simplement assuré par le choix d'un ordre aléatoire pour l'examen des données.

D'un point de vue pratique, une technique utilisée consiste à brasser l'ensemble des données au début de l'algorithme en leur appliquant une permutation aléatoire (les $n!$ permutations doivent être équiprobables), l'ordre d'insertion des données n'est plus fixé et donc déterministe mais aléatoire parmi les $n!$ ordres possibles (cf. FIG 2).

Les techniques de randomisation assurent que l'utilisateur malicieux, que l'on appelle l'adversaire, ne peut pas gagner: comme les conditions d'uniformité sont effectivement réalisées (et pas seulement supposées), les cas extrêmes sont garantis rares, inconditionnellement.

L'analyse randomisée prouve alors que l'arbre est équilibré en moyenne (En fait, ce modèle d'arbre binaire est strictement équivalent au *Quicksort*, l'ordre d'insertion des n nombres dans l'arbre binaire correspond au choix de n pivots dans *Quicksort*, si x_i et x_j $(i < j)$ doivent être comparés, ils le seront lors de l'insertion de x_j dans le cas du tri par insertion dans l'arbre binaire et lors du choix de x_i comme pivot dans le cas du *Quicksort*).

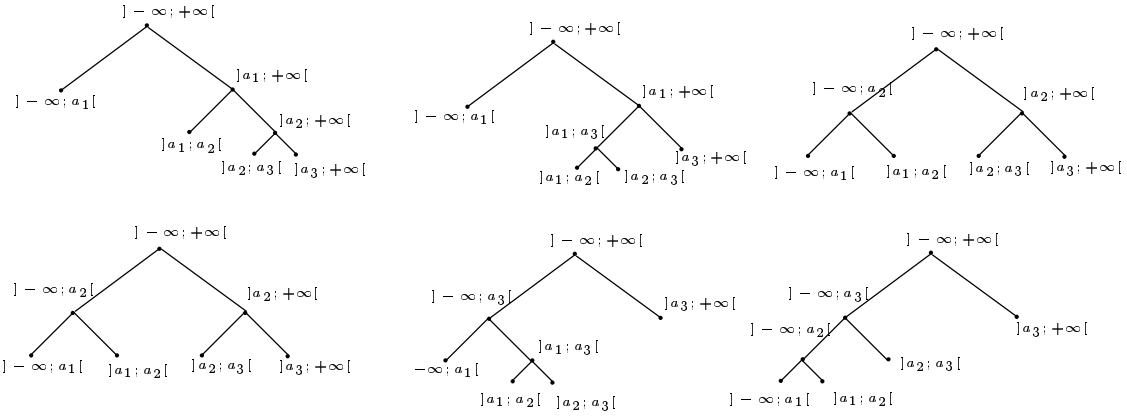


FIG. 2 – *Dans le cas où $n = 3$, il existe $3! = 6$ permutations équiprobables. Les arbres correspondant aux différents ordres d'insertion possibles sur les nombres $a_1 = 1$,$a_2 3$ et $a_3 = 5$ sont dessinés.*

Il est important de noter que cette modification n'améliore pas le temps d'exécution dans le pire des cas de l'algorithme mais il rend le temps d'exécution indépendant de l'ordre d'entrée ainsi *aucune entrée particulière ne provoque un comportement dans le pire des cas.*

Pour ce type d'algorithmes randomisés où l'hypothèse d'équiprobabilité de toutes les entrées d'une taille donnée est pertinente on s'intéresse alors au temps d'exécution moyen de l'algorithme sur l'ensemble des tirages aléatoires possibles. On obtient ainsi une borne plus fine correspondant à un temps d'exécution moyen.

Theoreme 3.2 *La complexité moyenne de l'algorithme de tri est $\Theta(n \log n)$.*

Preuve: Bien que l'algorithme de tri proposé soit simple, il peut s'avérer difficile d'analyser proprement celui-ci si l'on ne dispose pas d'une technique adéquate.

Plutôt que d'effectuer une analyse du cas général, nous allons effectuer une analyse en arrière (*backwards analysis*) [18] qui utilisera une variable aléatoire binaire $\{X_{ij}\}$ particulière appelée *variable indicatrice* définie par:

$$\{X_{ij}\} = \begin{cases} 1 & \text{si les nombres } b_i \text{ et } b_j \text{ sont comparés au cours} \\ & \text{de l'exécution de l'algorithme sur la liste } \mathcal{L} \\ 0 & \text{sinon} \end{cases}$$

On désire alors trouver une borne supérieure de $E[X]$, où X est une variable aléatoire prenant comme valeur le nombre total de comparaisons effectuées lors du tri selon un ordre d'insertion donné $\mathcal{L}$. Il existe une réalisation de cette variable aléatoire pour chacun des ordres d'insertion possibles pour notre algorithme.

Notons que $X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$.

En utilisant la linéarité de l'espérance mathématique on obtient:

$$\begin{aligned} E[X] &= E[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (0.Proba(X_{ij}=0) + 1.Proba(X_{ij}=1)) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Proba(X_{ij}=1) \end{aligned}$$

Il nous reste donc à calculer $Proba(X_{ij}=1)$. Nous supposerons pour la suite que $j > i$ pour tout couple (i,j).

Nous avons vu, lors de l'analyse précédente, qu'il existe, durant le tri des n nombres, une insertion unique telle que deux nombres distincts b_i et b_j qui se trouvaient jusqu'à présent dans un même intervalle correspondant à une feuille de l'arbre binaire, appartiennent désormais chacun à une feuille différente. La feuille contenant à la fois b_i et b_j avant cette insertion contient obligatoirement l'intervalle $]b_i,b_j[$, le nombre inséré doit donc faire parti des $j-i+1$ nombres possibles $b_i,b_{i+1},\ldots,b_{j-1},b_j$ afin que b_i et b_j n'appartiennent plus au même intervalle à l'étape suivante. Si b_i et b_j sont comparés, cela ne peut se passer que lors de cette insertion et lorsque le nombre inséré à cette étape est un des deux nombres b_i et b_j parmi les $j-i+1$ nombres possibles, c'est-à-dire:

$$Proba(X_{ij}=1) = \frac{2}{j-i+1}$$

On obtient alors une borne supérieure du nombre total de comparaisons effectuées à partir du calcul suivant:

$$\begin{aligned} E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Proba(X_{ij}=1) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\ &= 2\sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{1}{k} = 2\sum_{i=2}^{n} (H_i - 1) \end{aligned}$$

où $H_k = \sum_{i=1}^{k} \frac{1}{i}$ désigne le k^{eme} nombre harmonique que l'on peut encadrer par

$$\log k \leq H_k \leq 1 + \log k$$

$$\text{d'où} \qquad E[X] \leq 2\sum_{i=1}^{n} \log i = 2\log(n!) \leq 2n\log n = \Theta(n\log n) \qquad \text{puisque } n! \leq n^n$$

□

Il s'avère donc que le nombre *moyen* de comparaisons nécessaires au tri de n nombres avec cette version randomisée de l'algorithme est $\Theta(n\log n)$, complexité optimale du problème de tri, et cela quelque soit l'ordre des données fourni par l'adversaire (même pour des données déjà triées).

D'un point de vue pratique, il est donc intéressant, si l'on en a la possibilité, de randomiser les données (c'est-à-dire de leur appliquer une permutation aléatoire). Cependant on ne peut obtenir ainsi que des algorithmes statiques étant donné que l'on est obligé d'attendre l'ensemble des nombres avant de les insérer dans l'arbre binaire. Le caractère *en ligne* des algorithmes auxquels on s'intéresse nous impose de considérer les données dans l'ordre dans lequel elles se présentent.

Nous allons dans la suite proposer une solution permettant d'améliorer d'un facteur dépendant de la taille du tampon utilisé la complexité obtenue lors d'une analyse dans le pire des cas de notre algorithme incrémental en ligne de tri.

3.4 Analyse première stratégie

Nous allons voir, dans cette section, comment l'analyse en arrière nous permet d'analyser simplement l'algorithme de tri utilisant un tampon mélangeur avec la première stratégie.

Theoreme 3.3 *La complexité moyenne de l'algorithme de tri utilisant le tampon mélangeur et la première stratégie est* $\Theta(\frac{n^2 \log k}{k})$.

Preuve: On remarque tout d'abord que l'on retrouve pour les k premières données insérées un arbre équilibré en moyenne c'est-à-dire un coût moyen de $2k \log k$ comparaisons.

En effet, analysons le coût du p^{eme} nombre ($p \leq k$) dans notre arbre binaire de recherche.

On rappelle que l'insertion du i^{eme} nombre crée deux nouvelles feuilles dans l'arbre et que le nombre total de feuilles à l'étape i est de $i+1$. Le nombre de feuilles d'un arbre binaire est, en effet, exactement égal au nombre de noeuds internes de cet arbre plus un. Soit e le nombre de feuilles et i le nombre de noeuds internes de l'arbre. Chaque noeud interne possède deux fils, il existe donc $2i$ "branches" dans l'arbre, de même, chaque noeud interne est fils d'un autre noeud à l'exception de la racine d'où $2i = i + e - 1$ c'est-à-dire $i = e - 1$. De plus, juste après l'insertion du i^{eme} nombre, il existe une seule feuille contenant x_p dans l'arbre.

L'hypothèse randomisée sur les k premières données nous permet alors de dire que la feuille contenant x_p à la i^{eme} étape a été créée à l'étape i avec la probabilité $\frac{2}{i+1}$.

On obtient facilement le coût d'insertion du p^{eme} ($p \leq k$) nombre en sommant sur l'étape de création de ces feuilles, d'où le coût d'insertion de x_p:

$$\sum_{1 \leq i < p} \frac{2}{i+1} = 2(H_p - 1)$$

Soit $b_1,\ldots,b_k$ la liste ordonnée des k premiers nombres insérés dans l'arbre, on cherche maintenant à calculer le coût moyen de localisation d'un nombre x_p, $p > k$ dans un tel sous-arbre en fonction de j où j est tel que $b_j < x_p < b_{j+1}$ (i.e $j = Card\{x_i < x_p; i \leq k, p > k\}$) (cf FIG 3).

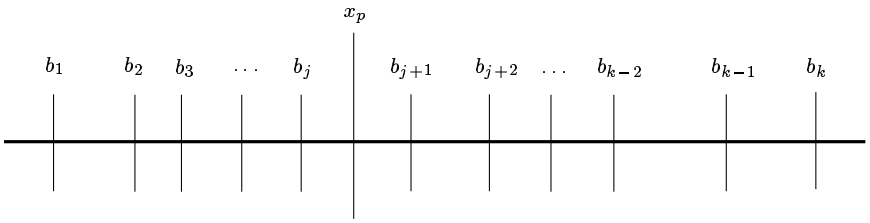


FIG. 3 – *Ordre quelconque*

La probabilité de comparer x_p avec $b_{i\{i \leq j\}}$, l'un des j nombres lui étant inférieur à cette étape, est égale à la probabilité d'insérer b_i avant l'ensemble des $j - i$ nombres compris entre b_i et x_p c'est-à-dire $\frac{1}{j-i+1}$. De même, la probabilité de comparer x_p avec $b_{s\{s > j\}}$, l'un des $k - j$ nombres lui étant supérieur à cette étape, est égale à la probabilité d'insérer b_s avant l'ensemble des $s - (j+1)$ nombres compris entre x_p et b_s et vaut donc $\frac{1}{s-j}$.

Le coût moyen de localisation de x_p dans l'arbre $\mathcal{T}_k$ est alors obtenu en sommant ces probabilités sur l'ensemble des k nombres $\{x_i\}_{i \leq k}$ appartenant à celui-ci, d'où:

$$E[\text{coût de localisation dans } \mathcal{T}_k] = \sum_{i=1}^{j} \frac{1}{j-i+1} + \sum_{s=j+1}^{k} \frac{1}{s-j} = H_j + H_{k-j}$$

Par convexité de la fonction $f(j) = H_j + H_{k-j}$ sur l'intervalle $0,\ldots,k$, on obtient:

$$\max_j(H_j + H_{k-j}) = H_{\lfloor \frac{k}{2} \rfloor} + H_{\lceil \frac{k}{2} \rceil}$$

Le coût moyen de localisation dans l'arbre $\mathcal{T}_k$ est donc maximal lorsque le nombre à localiser appartient à l'intervalle $[\alpha_k,\beta_k]$ où α_k est le $\lfloor \frac{k}{2} \rfloor^{eme}$ nombre dans la liste ordonnée des k éléments de l'arbre $\mathcal{T}_k$ et β_k le $1 + \lfloor \frac{k}{2} \rfloor^{eme}$.

Considérons maintenant le coût des comparaisons de x_p avec x_j ($j \in [qk+1,(q+1)k]$ avec $0 \leq q \leq \lfloor \frac{p}{k} \rfloor - 1$). Soit $[\alpha,\beta]$ la feuille de l'arbre $\mathcal{T}_{qk}$ contenant x_p et soit $m = |[\alpha,\beta] \cap \{x_{qk+1},\ldots,x_{(q+1)k}\}|$, le coût de localisation de x_p dans le sous-arbre de racine $[\alpha,\beta]$ de $\mathcal{T}_{(q+1)k}$ vaut en moyenne $H_j + H_{m-j}$ (où m désigne ici $Card\{x_i \in [\alpha,\beta]; qk+1 \leq i \leq (q+1)k\}$) et est donc maximal pour $m = k$ c'est-à-dire lorsque l'ensemble des k nombres issus de la $q+1$ série d'insertions sont tous contenus dans la même feuille $[\alpha,\beta]$ de $\mathcal{T}_{qk}$.

Dans le pire des cas sur les données, la q^{eme} série de k insertions crée donc un sous-arbre de racine l'intervalle $[\alpha_{(q-1)k},\beta_{(q-1)k}]$ (intervalle le plus haut en moyenne dans l'arbre $\mathcal{T}_{(q-1)k}$).

Un tel ordre sur les données est obtenu, par exemple, avec l'ordre alterné, décrit figure 4, pour lequel tout intervalle défini par a_{m-1} et a_m contient l'ensemble des nombres $\{a_j\}_{j>m}$ fournis après a_m par l'adversaire, tout nombre x_p ($p > qk$) est dans ce cas supérieur à $\lceil \frac{qk}{2} \rceil$ éléments et est inférieur à $\lfloor \frac{qk}{2} \rfloor$ éléments des nombres présents dans l'arbre $\mathcal{T}_{qk}$.

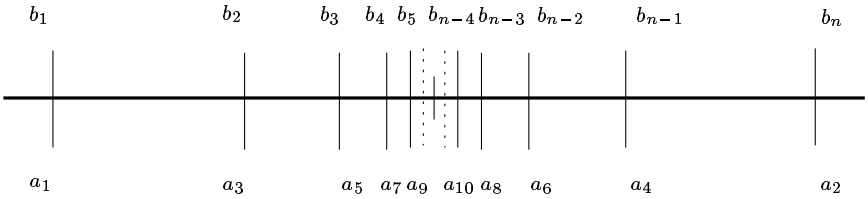


FIG. 4 – *Ordre alterné généré par la suite* $a_i = (-1)^i \frac{1}{\lceil \frac{i}{2} \rceil}$

On calcule alors une borne du coût moyen d'insertion du p^{eme} nombre dans la structure en sommant le coût moyen maximal de localisation de x_p pour chacun des $\lfloor \frac{(p-1)}{k} \rfloor$ sous-arbres construits, et le coût d'insertion de x_p dans le sous-arbre en cours de construction et pour lequel x_p est le $(p - \lfloor \frac{p-1}{k} \rfloor k)^{eme}$ nombre à être inséré:

$$E[\text{coût d'insertion de } x_p \text{ dans } \mathcal{T}_{p-1}] \leq \left\lfloor \frac{p-1}{k} \right\rfloor \left(H_{\lceil \frac{k}{2} \rceil} + H_{\lfloor \frac{k}{2} \rfloor} \right) + 2(H_{p-\lfloor \frac{p-1}{k} \rfloor k} - 1)$$

Le coût moyen du tri de n nombres est alors obtenu à partir du calcul suivant:

$$\begin{aligned}
&= \sum_{p=1}^{n} \left(\left\lfloor \frac{p-1}{k} \right\rfloor \left(H_{\lceil \frac{k}{2} \rceil} + H_{\lfloor \frac{k}{2} \rfloor} \right) + 2(H_{p-\lfloor \frac{p-1}{k} \rfloor k} - 1) \right) \\
&= \left(H_{\lceil \frac{k}{2} \rceil} + H_{\lfloor \frac{k}{2} \rfloor} \right) \sum_{p=1}^{n} \lfloor \frac{p-1}{k} \rfloor + 2 \sum_{p=1}^{n} (H_{p-\lfloor \frac{p-1}{k} \rfloor k} - 1) \\
&\leq \frac{H_{\lceil \frac{k}{2} \rceil} + H_{\lfloor \frac{k}{2} \rfloor}}{2k} n^2 + \frac{H_{\lceil \frac{k}{2} \rceil} + H_{\lfloor \frac{k}{2} \rfloor}}{2} n + 2(n+k)(H_k - 1)
\end{aligned}$$

soit

$$E[X] \leq n \left(\frac{H_{\lceil \frac{k}{2} \rceil} + H_{\lfloor \frac{k}{2} \rfloor}}{2k} n + \frac{H_{\lceil \frac{k}{2} \rceil} + H_{\lfloor \frac{k}{2} \rfloor}}{2} + 2(H_k - 1) \right) + 2k(H_k - 1)$$

en utilisant les majorations

$$2\log(k+1) > H_{\lceil \frac{k}{2} \rceil} + H_{\lfloor \frac{k}{2} \rfloor} \quad \forall k > 1$$

et

$$2\log k > H_{\lceil \frac{k}{2} \rceil} + H_{\lfloor \frac{k}{2} \rfloor} \quad \forall k > 8$$

on obtient:

$$E[X] \leq n\left(\frac{\log(k+1)}{k}n + 3\log(k+1)\right) + 2k\log k \quad k > 0$$

et

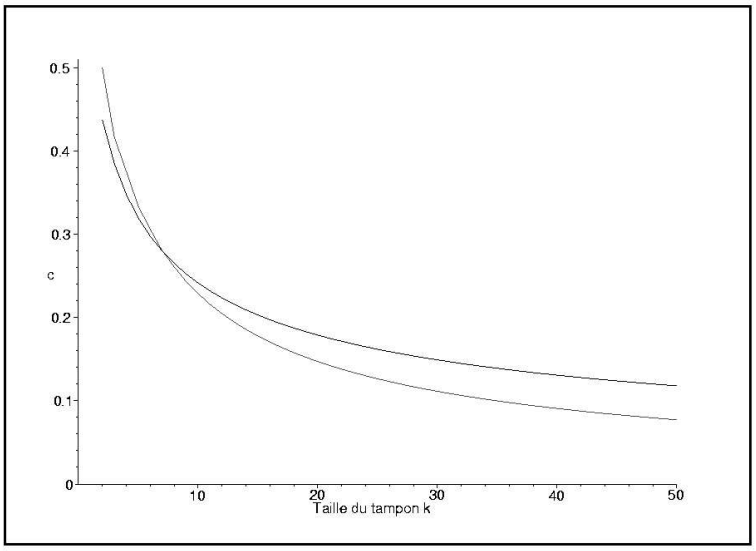
$$E[X] \leq n\left(\frac{\log k}{k}n + 3\log k\right) + 2k\log k \quad k > 8$$

soit une complexité en $\Theta(\frac{n^2 \log k}{k})$. □

Pour des valeurs de k petites devant n, la valeur de la constante facteur du terme quadratique obtenue dans le pire des cas en 3.2 $c_{TRI} = \frac{1}{2}$ est améliorée d'un facteur constant fonction de k, à savoir, $c_{TRI1}(k) = \frac{\log(k+1)}{k}(1 + \Theta(\frac{k}{n}))$ (cf. FIG 5).

Lorsque la taille du tampon s'approche de n, nombre total de données, on retrouve la complexité moyenne optimale $\Theta(n \log n)$ obtenue dans le cas randomisé.

FIG. 5 – *Valeur de la constante* $c_{TRI1}(k) = \frac{H_{\lceil \frac{k}{2} \rceil} + H_{\lfloor \frac{k}{2} \rfloor}}{2k}$ *du terme quadratique* $c_{TRI1}(k)n^2$ *en fonction de* k *pour la première stratégie et pour* $k << n$.



3.5 Analyse deuxième stratégie

Il devient difficile avec cette seconde stratégie de prouver que l'ordre alterné constitue, comme pour la première stratégie, l'ordre le plus défavorable que l'on puisse rencontrer sur les données pour le tri de n nombres. Le dénombrement et le calcul des différentes probabilités, qui dépendent de i,j,k et n, rendent, en effet, rapidement difficiles les calculs nécessaires. On propose néanmoins en annexe une étude (cf. 7.1.1 page 32).

Bien que l'on puisse aboutir à un résultat calculable sous forme de somme, on ne peut toutefois pousser les calculs aussi loin que pour la stratégie précédente et obtenir une borne sous une forme agréable de la complexité du problème. Afin d'illustrer les difficultés rencontrées lors de l'analyse de cette nouvelle stratégie, nous proposons l'étude d'un exemple particulier où $k = 3$ et pour un ordre initial alterné en annexe 7.1.3.

A partir de $k = 4$ le calcul exact de $E[X]$ nécessite le calcul d'un grand nombre de cas particuliers. On peut malgré tout obtenir une borne inférieure et une borne supérieure du nombre moyen de comparaisons lorsque $k << n$ en utilisant, de la même manière que pour l'étude du cas $k = 3$, un encadrement de l'ensemble des probabilités différentes du cas général.

On mène l'analyse en distinguant parmi les $\frac{n(n-1)}{2}$ couples (i,j) $(i < j)$, les cas où:

- i ou j correspond à une des $k-1$ premières données fournis par l'adversaire traitées de façon particulière, pour chacune de ces $k-1$ valeurs il existe moins de n couples (cf. équation 21 page 36, pour le cas $k = 3$ donné en exemple)
- (i,j) est tel que $n-(k-2) < j+i \leq n+k$, il existe, dans ce cas, moins de $\frac{n}{2}$ couples pour chacune des $2(k-1)$ valeurs (cf. équations 18 et 19 page 35 et 35)
- (i,j) est tel que $1 \leq j-i < k$, il existe, dans ce cas, moins de n couples (i,j) pour chacune des $k-1$ valeurs (cf. équations 17 et 20 page 35 et 35)

On peut donc dénombrer plus de

$$\frac{n(n-1)}{2} - \sum_{i=1}^{3(k-1)} n = \frac{n^2+5n-6kn}{2} \tag{1}$$

couples (i,j) représentant le cas général (cf.22 page 36).

Le cas général correspond au cas le plus fréquent lorsque $k << n$ et associe à un couple (i,j) la plus faible probabilité de comparer b_i et b_j. La valeur p_o de cette probabilité est donnée par la formule suivante:

$$Proba(X_{ij} = 1) = 1 - \sum_{i=2}^{2(k-1)} \left(\frac{\lceil \frac{i-1}{2} \rceil}{k} \prod_{j=1}^{i} \frac{k - \lceil \frac{j}{2} \rceil}{k} \right)$$

(Il existe au moins $k-1$ nombres compris entre b_i et b_j parmi l'ensemble des nombres fournis par l'adversaire entre b_i et b_j. La probabilité de ne pas comparer b_i et b_j entre eux est égale à la probabilité de choisir un des nombres contenu dans $[b_i,b_j]$ avant l'un de ces deux nombres. L'ensemble des nombres contenus dans cet intervalle sont fournis par l'adversaire tout les deux coups après l'insertion dans le tampon du premier des deux nombres b_i et b_j. On peut donc choisir un de ces nombres de la 3^{eme} à la $2k-1^{eme}$ étape. Il existe exactement $\lceil \frac{i-1}{2} \rceil$ nombres contenu dans l'intervalle $[b_i,b_j]$ i étapes après l'insertion du premier des deux nombres b_i, b_j dans le tampon.)

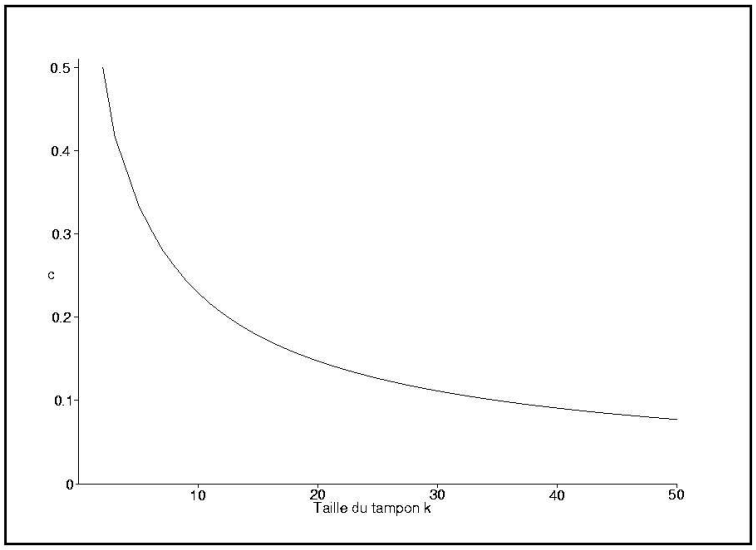
On obtient un encadrement des valeurs de l'ensemble des probabilités différentes du cas général, notées p_i, en minorant celles-ci par la valeur de p_o (dont on a vu qu'elle était minimale) et en les majorant par 1, soit:

$$p_o \leq p_i \leq 1$$

On obtient alors les bornes inférieure et supérieure suivantes pour le nombre moyen de comparaisons nécessaires:

$$\begin{aligned} p_o \frac{n(n-1)}{2} \leq E[X] &\leq p_o(\frac{n^2+5n-6kn}{2}) + 3(k-1)n \\ &\leq \frac{p_o}{2}n^2 + 3k(1-p_o)n \end{aligned}$$

FIG. 6 – *Valeur de la constante $c_{TRI2}(k) = \frac{p_o}{2}$ du terme quadratique $c_{TRI2}(k)n^2$ en fonction de k pour la deuxième stratégie et pour $k << n$.*



D'où, dans le cadre de la deuxième stratégie:

$$\frac{p_o}{2}n^2 - \frac{p_o}{2}n \leq E[X] \leq \frac{p_o}{2}n^2 + 3k(1-p_o)n$$

soit

$$\frac{p_o}{2} - \frac{p_o}{2n} \leq c_{TRI2}(k) \leq \frac{p_o}{2} - \frac{(1-p_o)3k}{n}$$

Lorsque k reste très petit devant n, c'est-à-dire lorsque le rapport $\frac{k}{n}$ tend vers 0, on obtient pour la constante $c_{TRI2}(k)$ du terme quadratique les bornes suivantes:

$$\frac{p_o}{2} - \frac{p_o}{2n} \leq c_{TRI2}(k) \leq \frac{p_o}{2}$$

Les bornes supérieure et inférieure ont alors une différence petite asymptotiquement ($\lim_{n\to\infty} \frac{p_o}{2n} = 0$) et on obtient donc une bonne estimation de la constante $c_{TRI2}(k)$ pour $k << n$ et $n \to \infty$ (cf. FIG 6) par:

$$\frac{1}{2} - \sum_{i=2}^{2(k-1)} \left(\frac{\lceil \frac{i-1}{2} \rceil}{2k} \prod_{j=1}^{i} \frac{k - \lceil \frac{j}{2} \rceil}{k} \right)$$

3.6 Borne inférieure

L'ordre alterné sur les données étant plus défavorable que l'ordre croissant pour l'ensemble des stratégies, une borne inférieure sur l'ordre croissant est donc *a fortiori* une borne inférieure sur l'ordre alterné. Ainsi, afin de calculer le coût minimum du tri de n nombres quelque soit la stratégie utilisée, nous allons calculer un borne inférieure lorsque l'ordre initial des données est $a_i = i$.

Pour chacune des stratégies présentées, on remarque que le plus grand nombre présent dans l'arbre à une étape p est supérieur ou égal au nombre fourni par l'adversaire $k-1$ étapes auparavant soit a_p.

Dans le meilleur des cas, on choisit le plus grand nombre appartenant au tampon toutes les k étapes (c'est-à-dire à la qk^{eme} étape $1 \leq q \leq \lfloor \frac{n}{k} \rfloor$) et l'on choisit pour les $k-1$ insertions suivantes l'ensemble des nombres appartenant au tampon qui lui sont inférieurs. L'ensemble des nombres restant à insérer dans l'arbre après cette série de k

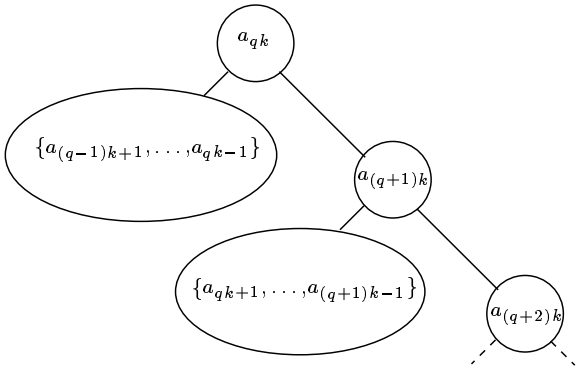


FIG. 7 – *La localisation de x_p dans l'arbre vaut au moins une comparaison pour chacun des paquets de k nombres insérés avant x_p*

insertions sont alors strictement supérieurs au plus grand nombre présent dans l'arbre à cette étape (cf. FIG 7). Le nombre de comparaisons nécessaires à l'insertion de x_p dans $\mathcal{T}_{p-1}$ est donc au moins égal aux nombres de paquets de k nombres insérés avant x_p soit $\frac{p-1}{k}$ plus le nombre de comparaisons nécessaires à la localisation de x_p dans le sous-arbre auquel il appartient et pour lequel il est le $(p - \lfloor \frac{p-1}{k} \rfloor k - 1)^{eme}$ nombre inséré.

Le nombre moyen de comparaisons nécessaires à la localisation de x_p dans $\mathcal{T}_{p-1}$ est donc supérieur ou égal à:

$$\text{E[Coût d'insertion de } x_p] \geq \left\lceil \frac{p-1}{k} \right\rceil$$

On obtient une borne inférieure du nombre moyen de comparaisons nécessaires pour la construction de l'arbre en sommant sur les différentes insertions:

$$\text{E[Coût du tri de } n \text{ nombres}] \geq \sum_{j=1}^{n} \left(\left\lceil \frac{j-1}{k} \right\rceil \right) \geq \frac{n^2}{2k}$$

Quelque soit la stratégie utilisée on a donc $c_{TRI}(k) \geq \frac{1}{2k}$.

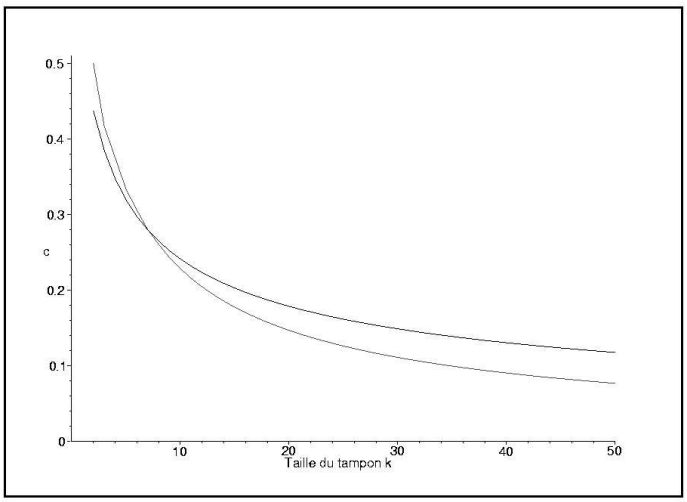
3.7 Comparaison des résultats

On a prouvé que l'ordre alterné était le pire des cas sur les données pour la première stratégie mais on ne possède malheureusement pas de preuve pour dire que cet ordre constitue aussi le pire des cas pour la deuxième (on peut toutefois montrer que l'ordre alterné est plus défavorable que l'ordre trié avec cette stratégie cf. en annexe 7.1.1 et 7.1.2).

Étant donné que la deuxième stratégie nous permet d'obtenir un nombre plus important d'ordres sur les données à partir d'un ordre imposé par l'adversaire, on pourrait penser que celle-ci est meilleure pour toutes valeurs de k. Cependant, de manière assez surprenante, on remarque qu'à partir de $k = 7$, la première stratégie se révèle être plus efficace que la deuxième pour l'ordre alterné (cf. FIG 8 ou l'on compare les deux stratégies pour un ordre initial alterné).

Toutefois, puisque chacune des deux stratégies ne génère pas les mêmes permutations sur les données, il est toujours possible de trouver un ordre initial sur les données qui favorise l'une ou l'autre des deux stratégies. On note alors que ces deux stratégies peuvent toujours être comparées pour un ordre fixé sur les données.

FIG. 8 – $c_{TRI1}(k)$ *et* $c_{TRI2}(k)$ *lorsque* $k << n$.



Afin de tirer parti au mieux de l'utilisation du tampon mélangeur, on essaiera, en pratique, de choisir une taille de tampon maximale pour les contraintes en vigueur, en général, la vitesse avec laquelle sont fournies les données.

Quelques problèmes géométriques

Nous allons, dans la suite, donner quelques exemples d'algorithmes incrémentaux appliqués à des problèmes de calculs géométriques.

La géométrie algorithmique est une vaste discipline, et puisque notre objectif est de nous concentrer sur la randomisation, nous posons, pour la suite, les hypothèses simplificatrices suivantes afin d'éviter de nous perdre dans des détails relatifs à l'implémentation de tels algorithmes.

Pas de cas dégénérés. Nous supposons que les données en entrée de nos algorithmes sont en *position générale*. Ainsi, quatre sites de E^2 ne seront jamais cocycliques pour le problème de la triangulation de Delaunay (cette restriction nous permet de garantir que le diagramme de Delaunay est bien une triangulation). De même, on supposera qu'aucun segment d'entrée n'est vertical, que trois segments d'entrée ne peuvent pas se croiser au même point, enfin, que les abscisses des extrémités et des points d'intersection des segments seront toutes distinctes en ce qui concerne le problème du cloisonnement vertical.

L'absence de cas dégénéré nous permet de nous passer des traitements relatifs aux cas particuliers et d'obtenir ainsi des algorithmes plus simples.

Arithmétique réelle exacte. Dans la pratique la précision numérique des ordinateurs est limitée. Cependant, afin d'échapper aux calculs relatifs à l'analyse numérique, nous supposerons, pour la suite, que l'on dispose d'une arithmétique réelle exacte.

Ces deux restrictions sont communes dans la littérature traitant de géométrie algorithmique. L'hypothèse de non dégénérescence n'est pas une restriction trop sévère. Il est, en effet, assez facile soit de modifier l'algorithme afin que celui-ci gère les entrées dégénérées soit de ramener les données en position dégénérée en position générale par des méthodes de perturbation standard. La restriction concernant l'arithmétique réelle exacte est plus sérieuse.

Cependant, supprimer ce type d'hypothèses simplificatrices et gérer les conditions aux limites est souvent la partie la plus difficile de la programmation des algorithmes de géométrie algorithmique, et de leur validation.

Espace à 2 dimensions. Dans la suite, nous étudierons deux problèmes de géométrie algorithmique en dimension deux. En se restreignant à deux dimension, on peut déjà voir un bon échantillon des techniques utilisées dans le domaine, de plus, ces problèmes se généralisent dans des espaces de dimension trois et au-delà [5].

4 Triangulation de Delaunay

Notre premier exemple géométrique est la construction incrémentale de la triangulation de Delaunay d'un ensemble de n sites du plan.

4.1 Définition et résultats fondamentaux

Calculer le diagramme de Voronoï d'un ensemble de n sites et son dual, la triangulation de Delaunay, est un problème classique de la géométrie algorithmique et constitue une des structures les plus populaires dans ce domaine de par ses nombreuses applications (modèles de terrains, maillages, reconstruction de formes ...).

4.1.1 Diagramme de Voronoï

Le diagramme de Voronoï d'un ensemble $\mathcal{S}$ de n points, appelés *sites*, appartenant à E^d, espace Euclidien de dimension d, est une structure géométrique permettant de résoudre des requêtes de proximité, c'est-à-dire déterminer le plus proche voisin d'une requête donnée.

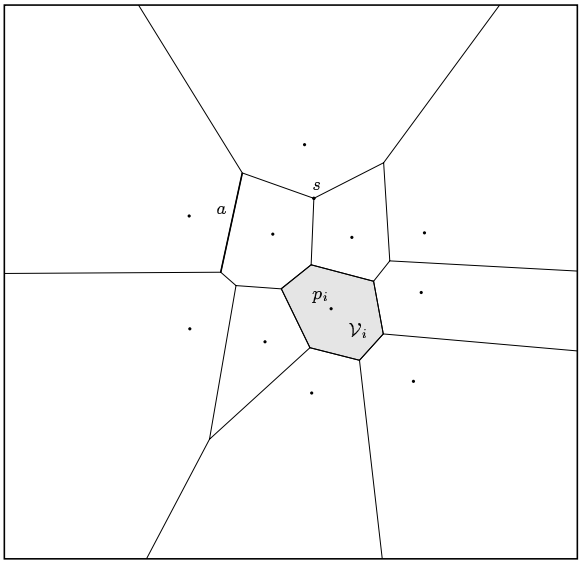


FIG. 9 – *Exemple de diagramme de Voronoï d=2, p_i est un site de Voronoi, $\mathcal{V}_i$ est un polygone de Voronoi, a est une arête de Voronoi, s est un sommet de Voronoi*

On appelle polygone de Voronoï $\mathcal{V}_i$ du point $p_i \in \mathcal{S}$ l'ensemble des points de E^d plus proche de p_i que d'un autre point de $\mathcal{S}$ soit:

$$\mathcal{V}_i = \{q; \forall j \neq i | qp_i| \leq |qp_j|\};$$

dans le cas le plus simple, $\mathcal{S} = \{p_i\}_{1 \leq i \leq n}$ est un ensemble de points du plan, q est également un point du plan, et plus proche fait référence à la distance euclidienne usuelle. On peut aussi définir $\mathcal{V}_i$ par:

$$\mathcal{V}_i = \cap_{q \in \mathcal{S} \setminus \{p_i\}} \mathcal{H}(p_i,q)$$

où $\mathcal{H}(p_i,q)$ est le demi-espace délimité par l'hyperplan, bissection de p et q, contenant p.

Le diagramme de Voronoï est donc un partitionnement de l'espace E^d formé par les polyèdres de Voronoï de l'ensemble des sites (voir FIG 9).

4.1.2 Triangulation de Delaunay

On appelle triangulation de $\mathcal{S}$ un graphe planaire et maximal sur S, c'est-à-dire un ensemble d'arêtes reliant des points de $\mathcal{S}$ sans intersections (deux arêtes ne se coupent pas) et tel que l'on ne peut rajouter d'arête.

La triangulation de Delaunay est le graphe dual du diagramme de Voronoï: deux sites de $\mathcal{S}$ sont reliés par une arête de la triangulation si et seulement si leurs polygones respectifs sont adjacents. L'ensemble des arêtes ainsi formées est appelé triangulation de Delaunay de $\mathcal{S}$ et notée $\mathcal{DT}(\mathcal{S})$, un exemple est donné figure 10.

La triangulation de Delaunay est l'unique triangulation tel que tout cercle circonscrit à un triangle de Delaunay ne contient aucun site de $\mathcal{S}$.

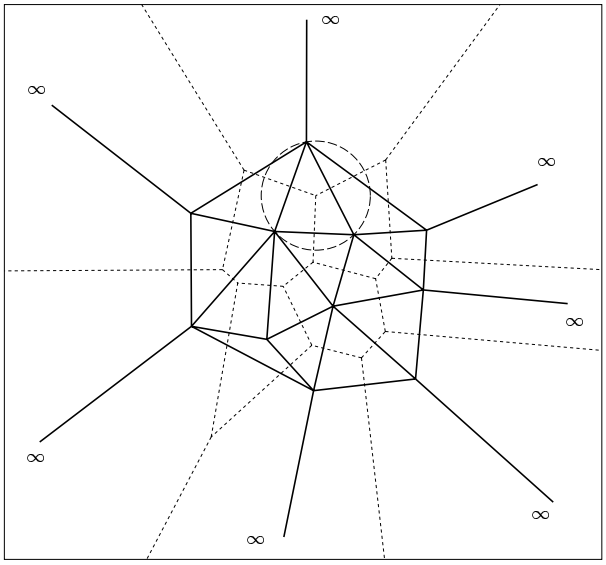


FIG. 10 – *Diagramme de Voronoï et triangulation de Delaunay associée $d = 2$. Un cercle de Delaunay est tracé*

On dit qu'un point p est en conflit avec un triangle $p_x p_y p_z$ (p_x,p_y,$p_z \in \mathcal{S}$) s'il est contenu à l'intérieur de son cercle circonscrit. Le nombres de sites en conflit avec un triangle peut prendre des valeurs allant de 0 à $n-3$, les triangles sans conflit étant précisément les triangles de Delaunay de $DT(\mathcal{S})$.

4.1.3 Propriétés

Dans une triangulation, toutes les cellules sont des triangles, on peut en déduire que

$$2a = 3c$$

où, a désigne le nombre d'arêtes et c le nombre de triangles de la triangulation.

De la relation d'Euler appliquée à la triangulation de Delaunay dans l'espace

$$c - a + n = 2$$

on déduit que le nombre d'arêtes de Delaunay est égal à $3n-6$ où n est le nombre de sites triangulés.

On rappelle que le degré d'un sommet de la triangulation est le nombre d'arêtes incidentes en ce point ou de manière équivalente le nombre de triangles incidents à ce sommet. Chaque arête étant incidente à 2 sommets, on obtient:

$$\sum_{s \in \mathcal{S}} \text{degré}(s) = 2(3n-6) < 6n$$

La définition habituelle de la triangulation d'un ensemble de points spécifie que l'intérieur de l'enveloppe convexe de l'ensemble des points est divisé en un ensemble de triangles qui ne se chevauchent pas, ayant pour sommets les points de l'ensemble.

Ainsi, les faces de la subdivision sont des triangles, exceptée la face non bornée qui a autant de sommets qu'il y a de points sur l'enveloppe convexe. Toutefois, pour plusieurs applications telles que la construction incrémentale de la triangulation de Delaunay, il est pratique de trianguler également la face non bornée. De cette manière, le cas des frontières de l'enveloppe convexe est plus facile à traiter. On ajoute donc un point supplémentaire à la triangulation, le point à l'infini. Tous les sommets de l'enveloppe convexe sont incidents à ce point. A chaque arête de l'enveloppe convexe est associée un triangle infini (cf. FIG 10).

Dans ce cas, il est aisé de remarquer que l'ensemble des n sites de $\mathcal{S}$ ont un degré supérieur ou égal à 3 dans $DT(S)$. La somme des degrés de chaque site à trianguler est donc minoré par:

$$\sum_{s \in \mathcal{S}} \text{degré}(s) \geq 3n$$

4.2 Un algorithme incrémental

Un algorithme de triangulation incrémental consiste à ajouter les points un par un en mettant à jour la triangulation à chaque nouveau point.

D'après la propriété vue précédemment, on sait qu'un triangle est de Delaunay si et seulement si son cercle circonscrit est vide. Lors de l'insertion du site M, les triangles de $\mathcal{DT}(\mathcal{S})$ dont le cercle circonscrit ne contient pas M seront toujours vide si on considère l'ensemble $\mathcal{S} \cup \{M\}$ et font donc partie de $\mathcal{DT}$ $(\mathcal{S} \cup \{M\})$. De même un triangle de $\mathcal{DT}$ $(\mathcal{S} \cup \{M\})$ n'ayant pas M comme sommet faisait déjà partie de $\mathcal{DT}(\mathcal{S})$ puisque son cercle circonscrit ne contient pas de point de $\mathcal{S} \cup \{M\}$ donc *a fortiori* pas de point de $\mathcal{S}$. La triangulation de $\mathcal{DT}$ $(\mathcal{S} \cup \{M\})$ se déduit donc de $\mathcal{DT}(\mathcal{S})$ en supprimant tous les triangles dont le cercle circonscrit contient M et qui ne peuvent donc plus être des triangles de Delaunay, et en retriangulant cette zone depuis M. On a comme corollaire que la zone détruite est étoilée par rapport à M (cf. Figure 11).

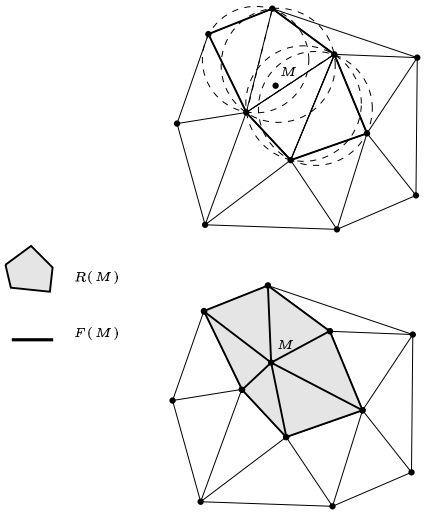


FIG. 11 – *Soit M un site introduit dans la triangulation. Tout les triangles en conflit avec M ne peuvent plus être des triangles de Delaunay. L'union de ces triangles est une région connexe $R(M)$. Les nouveaux triangles sont obtenus en joignant M aux arêtes de $F(M)$*

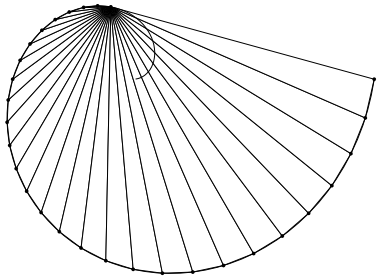


FIG. 12 – *Le pire des cas sur les données se présente lorsque les sites sont placés sur une courbe de courbure croissante (morceau de parabole ou de spirale logarithmique) et lorsque ceux-ci sont fournis dans le sens de la courbure.*

4.3 Analyse de la construction incrémentale de la triangulation de Delaunay

L'insertion d'un nouveau site dans notre algorithme se déroule clairement en deux phases, la première consiste à déterminer la cavité associée au nouveau site dans la triangulation courante, la deuxième consiste à construire les nouveaux triangles induits par le nouveau site.

La complexité de l'algorithme est donc égale à la somme du *coût de localisation* de chaque site dans la triangulation courante et du *coût d'insertion*. Cette section s'intéresse à l'analyse de la complexité de l'algorithme de triangulation en termes de coût d'insertion uniquement, c'est-à-dire le nombre de triangles construits au cours du processus incrémental. Ce choix se justifie par le fait que le coût d'insertion/construction est indépendant de l'implémentation de l'algorithme de triangulation, ce qui n'est pas le cas du coût de localisation qui dépend fortement de la structure de données choisie comme nous le verrons en 4.4.

Le coût d'insertion de chaque étape est proportionnel au nombre de triangles ajoutés à la triangulation courante à cette étape. On obtient alors le coût d'insertion de l'ensemble des sites en sommant le nombre de triangles construits au cours des triangulations successives du processus incrémental.

4.3.1 Analyse dans le pire des cas

Si l'ordre d'insertion des sites est imposé par l'adversaire, on peut montrer qu'il existe des ensembles de points et des ordres sur les insertions tels que $\Theta(n^2)$ triangles apparaîtront comme des triangles de Delaunay au cours de la construction incrémentale.

Theoreme 4.1 *Le nombre de triangles de Delaunay crées dans le pire des cas lors d'une construction incrémentale est* $\Theta(n^2)$.

Preuve: Pour prouver cela, considérons l'exemple de sites placés sur la parabole (cf. Figure 12 et 13), l'insertion du i^{eme} $(3 \leq i \leq n)$ site nécessite la reconstruction de la totalité de la triangulation, c'est-à-dire la création de $i-1$ arêtes (un arête est crée entre a_i et l'ensemble des sites $\{a_{1 \leq j < i}\}$ déjà insérés) et donc de $i-2$ nouveaux triangles, on construit donc

$$\sum_{i=3}^{n}(i-2) = \frac{(n-2)(n-1)}{2} = \frac{n^2-3n+2}{2} = \Theta(n^2)$$

triangles dans le pire des cas au cours de l'algorithme incrémental de triangulation des n sites. □

Un résultat connu pour cette méthode incrémentale de triangulation est que la *randomisation* des données conduit à des cavités de taille constante en moyenne (indépendante du nombre de points) et ainsi minimise globalement le nombre d'opération à effectuer.

4.3.2 Borne supérieure première stratégie

Theoreme 4.2 *Le nombre moyen de triangles de Delaunay crées lors d'une construction incrémentale utilisant le tampon mélangeur et la première stratégie est* $\Theta(\frac{n^2 \log k}{k})$.

Preuve: Supposons que l'on ait un ensemble de m sites, $\mathcal{V}$, pour lesquels on a déjà construit la triangulation et un autre ensemble, $\mathcal{W}$ de j sites tels que $m + j = p$. On désire insérer les sites de $\mathcal{W}$ un à un selon un ordre aléatoire en maintenant la triangulation de Delaunay de l'ensemble ainsi construit jusqu'à l'obtention de la triangulation de $\mathcal{V} \cup \mathcal{W}$.

Il est assez facile de voir que le coût d'insertion d'un tel algorithme est proportionnel au nombre de triangles construits au cours du processus incrémental. Quel est alors le nombre moyen de triangles construits? Il semble difficile de répondre à cette question si l'on fixe notre attention sur un site particulier x_p.

On montre ici que, grâce à l'analyse en arrière qui consiste à exprimer le coût d'insertion de x_p, dernier site inséré dans l'algorithme, non pas en fonction de $DT(\mathcal{V}')$, triangulation courante juste avant l'insertion de x_p, mais en fonction de la triangulation finale $DT(\mathcal{V} \cap \mathcal{W})$, on peut obtenir une borne supérieure du coût d'insertion de la triangulation incrémentale de S comprenant n sites et ce sans aucune hypothèse sur la répartition des données.

On remarque que le nombre de triangles construits à une étape de l'algorithme est exactement égal au degré du site inséré à cette étape (cf. FIG. 11).

Le nombre de triangles construits au cours de l'insertion de x_p dans la triangulation courante est donc égal au degré de x_p dans $DT(\mathcal{V} \cup \mathcal{W})$.

Si x_p est choisi de manière aléatoire dans $\mathcal{W}$, alors le degré moyen de x_p dans $DT(\mathcal{V} \cup \mathcal{W})$ est égal à la somme des degrés des j sites de $\mathcal{W}$ divisée par j.

On obtient, ainsi, le nombre de triangles construits lors de l'insertion du dernier site choisi parmi les j sites de $\mathcal{W}$ en calculant la moyenne des degrés des j sites appartenant à $\mathcal{W}$, soit:

$$\begin{aligned} E[c(x_p)] &= \frac{1}{j} \sum_{s \in \mathcal{W}} \text{degré}(s, DT(\mathcal{V} \cup \mathcal{W})) \\ &= \frac{1}{j} \left(\sum_{s \in \mathcal{V} \cup \mathcal{W}} \text{degré}(s, DT(\mathcal{V} \cup \mathcal{W})) - \sum_{s \in \mathcal{V}} \text{degré}(s, DT(\mathcal{V} \cup \mathcal{W})) \right) \end{aligned} \tag{2}$$

où, $c(x_p)$ désigne le coût d'insertion de x_p dans la triangulation $DT(\mathcal{V} \cup \mathcal{W} \setminus \{x_p\})$.

En utilisant, les résultats obtenus en 4.1.3, on arrive à la majoration suivante:

$$E[c(x_p)] \leq \frac{1}{j}(6p - 3(p - j)) \leq \frac{3(p + j)}{j}$$

L'analyse se mène ensuite comme si l'on déroulait l'algorithme à l'envers en commençant par $DT(\mathcal{V} \cup \mathcal{W})$ puis en répétant successivement la suppression des j sites de $\mathcal{W}$ de manière aléatoire pour un coût proportionnel à leur degré.

Si on considère $\mathcal{V}$ les qk premiers nombres et $\mathcal{W}$ les i premiers nombres choisis dans le $q + 1^{eme}$ tampon, on a:

$$E[c(x_{qk+j})] = \frac{3(qk + 2j)}{j}$$

Le coût moyen d'insertion d'un paquet de k sites est alors obtenu en sommant le coût moyen d'insertion des k sites présent dans le tampon, soit:

$$\sum_{j=1}^{k} \frac{3(qk+2j)}{j} = 3(pkH_k + 2k)$$

On obtient une borne supérieure du coût moyen d'insertion de l'ensemble des n sites de la triangulation, noté $E[Y]$, en sommant la valeur obtenue ci-dessus sur l'ensemble des paquets, soit:

$$E[Y] \leq \sum_{q=0}^{\lfloor \frac{n}{k} \rfloor} 3(qkH_k + 2k) = 3\frac{H_k}{2k}n^2 + \frac{3H_k n}{2} + 6n$$

Le problème de la triangulation de Delaunay de n sites a donc au plus un coût moyen d'insertion de:

$$\begin{aligned} E[Y] &\leq \frac{3H_k}{2k}n^2 + \frac{3H_k n}{2} + 6n \\ &= n\left(\frac{3H_k}{2k}n + \frac{3H_k}{2} + 6\right) \end{aligned}$$

d'où

$$E[Y] \leq n\left(\frac{3(1+\log k)}{2k}n + \frac{3\log k}{2} + \frac{15}{2}\right)$$

c'est-à-dire une complexité en $\Theta(\frac{n^2 \log k}{k})$ si l'on utilise la première stratégie et cela quelque soit la répartition des sites dans le plan. □

4.3.3 Analyse première stratégie pour des points sur la parabole

Il est possible d'affiner l'analyse pour le cas où les sites sont plaçés sur la parabole (cf. Figure 12).

Theoreme 4.3 *Dans le cas où les points sont situés sur la parabole, le nombre moyen de triangles de Delaunay crées lors d'une construction incrémentale utilisant le tampon mélangeur et la première stratégie est* $\Theta(\frac{n^2 \log k}{k})$.

Preuve: En reprenant l'exemple de la parabole, soit $\mathcal{V}$ l'ensemble des points $\{a_l\}_{1 \leq l \leq pk}$ sur la parabole, et soit $\mathcal{W}$ l'ensemble des sites $\{a_{pk+j}\}_{1 \leq j \leq k}$ du $(p+1)^{eme}$ paquet à insérer; si l'on insère les sites de $\mathcal{W}$ un à un selon un ordre quelconque, on doit relier à chacun des sites de $\mathcal{V}$ le nouveau site inséré chaque fois que celui-ci se trouve être le site le plus à gauche de l'ensemble courant des sites déjà présents dans la triangulation (cf. figure 13).

La probabilité pour que $a_{pk+j} = x_{pk+i}$ soit le site le plus à gauche du sous-ensemble courant à l'étape $pk+i$ est égale à $\frac{(j-1)!(k-i)!}{(j-i)!k!}$ $(i \leq j)$. Le $pk+j^{eme}$ site fourni par l'adversaire est, en effet, le i^{eme} site choisi parmi les k sites du p^{eme} paquet et est le site le plus à gauche du sous-ensemble courant si les $(i-1)$ premiers sites insérés avant a_{pk+j} sont choisis parmi les $j-1$ sites, appartenant au p^{eme} tampon, plus à droite que a_{pk+j}. Il existe donc $C_{j-1}^{i-1}(i-1)!(k-j)!$ ordres favorables parmi $k!$ soit:

$$\frac{C_{j-1}^{i-1}(i-1)!(k-i)!}{k!} = \frac{(j-1)!(k-i)!}{(j-i)!k!}$$

Si a_{pk+j} est le i^{eme} site choisi parmi les k sites du p^{eme} paquet alors son insertion nécessite la construction de $pk+i+c$ nouveaux triangles.

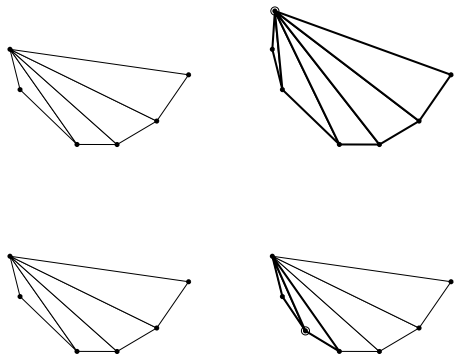


FIG. 13 – *Insertion d'un site, en haut: le site inséré est le site le plus à gauche du sous-ensemble courant, on doit retrianguler entièrement l'ensemble des sites, en bas, le site n'est pas le site le plus à gauche, l'insertion du nouveau site nécessite seulement la création de 2 nouveaux triangles*

La probabilité pour que a_{pk+j} ne soit pas le site le plus à gauche du sous-ensemble courant est donc égale à $1 - \sum_{i=1}^{j} \frac{(j-1)!(k-i)!}{(j-i)!k!} = 1 - \frac{1}{k-j+1}$
Dans ce cas, l'insertion de a_{pk+j} nécessite au plus la création de $d = 2$ nouveaux triangles.

On obtient alors une borne supérieure de $E[Y]$, nombre moyen de triangles construits au cours du processus incrémental de triangulation de n sites, en sommant sur le nombre de paquets de k sites la valeur suivante:

$$\begin{aligned}
& \sum_{j=1}^{k} \left(\sum_{i=1}^{j} \left(\frac{(k-i)!(j-1)!}{k!(j-i)!}(pk+i+c) \right) + d(1 - \frac{1}{k-j+1}) \right) \\
= & \sum_{j=1}^{k} \left(\frac{pk}{k-j+1} + \frac{c}{k-j+1} + \frac{k+1}{(k-j+2)(k-j+1)} + d - \frac{d}{k-j+1} \right) \\
= & \; pkH_k + cH_k + k + dk - dH_k \\
= & \; H_k(pk+c-d) + (d+1)k
\end{aligned}$$

soit à partir du calcul suivant:

$$\begin{aligned}
E[Y] &= \sum_{p=0}^{\lfloor \frac{n}{k} \rfloor} (H_k(pk+c-d) + (d+1)k) \\
&= kH_k \sum_{p=0}^{\lfloor \frac{n}{k} \rfloor} p + \sum_{p=0}^{\lfloor \frac{n}{k} \rfloor} ((c-d)H_k + (d+1)k) \\
&\leq kH_k \frac{n}{2k}(\frac{n}{k}+1) + (\frac{n}{k}+1)\left((c-d)H_k + (d+1)k\right) \\
&= \frac{H_k}{2k}n^2 + \left(\frac{H_k}{2} + \frac{H_k}{k}(c-d) + (d+1) \right) n + ((c-d)H_k + (d+1)k)
\end{aligned}$$

On en déduit la borne supérieure suivante:

$$E[Y] \leq n\left(\frac{H_k}{2k}n + \left(\frac{H_k}{2}+3\right) + 3\frac{k}{n}\right)$$

c'est-à-dire $E[Y] = O(\frac{n^2 \log k}{k})$ puisque

$$\boxed{E[Y] \leq n\left(\frac{1+\log k}{2k}n + \left(\frac{1+\log k}{2}+3\right) + 3\frac{k}{n}\right)}$$

□

4.4 Localisation

Comme nous l'avons vu précédemment, à chaque étape notre algorithme doit localiser les éléments de la cavité associée au nouveau site inséré.

Le choix d'une bonne structure de données est important pour l'efficacité des algorithmes de construction du diagramme de Voronoï/Delaunay. Il existe dans la littérature une grande variété des méthodes de localisation, on peut citer, par exemple, l'Arbre de Delaunay [2, 6] et la méthode Guibas-Knuth-Sharir [9] qui utilisent l'histoire des insertions pour un coût total de localisation $O(n \log n)$ si les points sont insérés selon un ordre aléatoire, la structure proposée par Mulmuley [15] qui utilisent des tirages à *pile ou face* à chaque niveau de la structure pour un même coût total, la Hiérarchie de Delaunay [12] ou encore [8, 11, 14].

L'Arbre de Delaunay conserve toutes les triangulations apparaissant à un moment donné de l'algorithme dans un graphe qui mémorise à chaque insertion d'un site la modification locale apportée à la triangulation. Il se présente donc comme la hiérarchie de la construction de la triangulation finale en établissant des relations entre les triangles des triangulations de Delaunay successives. Le but de la construction d'un tel graphe est de permettre la localisation rapide des éléments de la cavité associé au point en cours d'insertion.

L'Arbre de Delaunay nous fournit une structure dans laquelle le nouveau site peut être localisé en parcourant tous les triangles le contenant dans l'ordre chronologique de leur création.

Quel est le nombre de triangles devant être parcourus pour chaque site à insérer lors de cette phase de localisation?

Si l'on considère que l'ordre dans lequel sont insérées les données est complètement aléatoire [3], le nombre moyen de triangles construits à l'étape i qui sont en conflit avec le n^{eme} site inséré est inférieur à $\frac{6}{i}$ (Le degré moyen d'un site de la triangulation est inférieur à 6 [17], il existe donc au plus en moyenne 4 triangles en conflit avec le dernier site inséré, ces triangles étant définis par 6 sommets, un de ces triangles est crée à l'étape i si l'un des six sites les définissant est inséré précisément à la i^{eme} étape, c'est-à-dire avec la probabilité $\frac{6}{i}$).

Sous l'hypothèse randomisée, le coût moyen de localisation de l'ensemble des n sites dans l'Arbre de Delaunay vaut donc:

$$\sum_{p=1}^{n}\sum_{i=1}^{p}\frac{6}{i} = O(n \log n)$$

Si l'on considère maintenant l'ordre le pire sur les données (parabole), le dernier site inséré est en conflit à chaque étape avec l'ensemble des triangles construits. Le nombre de triangles en conflit avec x_n à la i^{eme} étape est donc égal à i.

Le coût de localisation pour l'ensemble des n sites vaut donc:

$$\sum_{p=1}^{n}\sum_{i=1}^{p} i = \sum_{p=1}^{n}\frac{p(p+1)}{2} = O(n^3)$$

En utilisant un tampon mélangeur avec la première stratégie ce coût devient:

$$\begin{aligned}\sum_{p=0}^{\lfloor \frac{n}{k} \rfloor} \sum_{i=1}^{k} \sum_{j=1}^{pk+i} \frac{3j}{i} &= \frac{3}{2} \sum_{p=0}^{\lfloor \frac{n}{k} \rfloor} \sum_{i=1}^{k} \frac{(pk+i)(pk+i+1)}{i} \\ &= \sum_{p=0}^{\lfloor \frac{n}{k} \rfloor} \left(3p^2k^2H_k + 3pk^2 + 3pkH_k + \frac{3k}{2} + \frac{3k(k+1)}{4} \right) \\ &= O(\frac{H_k n^3}{k})\end{aligned}$$

(le choix du i^{eme} site du tampon à insérer s'effectuant maintenant parmi les i données présentes dans le tampon courant.)

Le coût de localisation est donc amélioré par le tampon mélangeur d'un facteur $\frac{H_k}{k}$ puisque l'on passe d'un coût (désastreux) $O(n^3)$ à un coût $O(\frac{n^3 H_k}{k})$.

Avec des méthodes plus sophistiquées on peut cependant obtenir de meilleurs coûts.

La Hiérarchie de Delaunay nous permet, par exemple, d'avoir un coût de localisation $O(n^2 \log n)$ dans le pire des cas et $O(\log n \frac{n^2 \log k}{k})$ avec la première stratégie.

4.5 Résultats

Pour les mêmes raisons que pour le problème du tri, le calcul du coût moyen d'insertion de l'algorithme de triangulation utilisant la deuxième stratégie ne permet d'obtenir, après un certain nombre de calculs, que des majorations *grossières* et ne permet pas de pousser les calculs aussi loin que pour la première stratégie.

Nous avons effectué des tests expérimentaux sur des données réelles fournies par un capteur laser 3D. L'algorithme utilisé implémente la Hiérarchie de Delaunay [7].

FIG. 14 – *Tétrahedrisation de Delaunay de 145300 points (prothèse dentaire).*

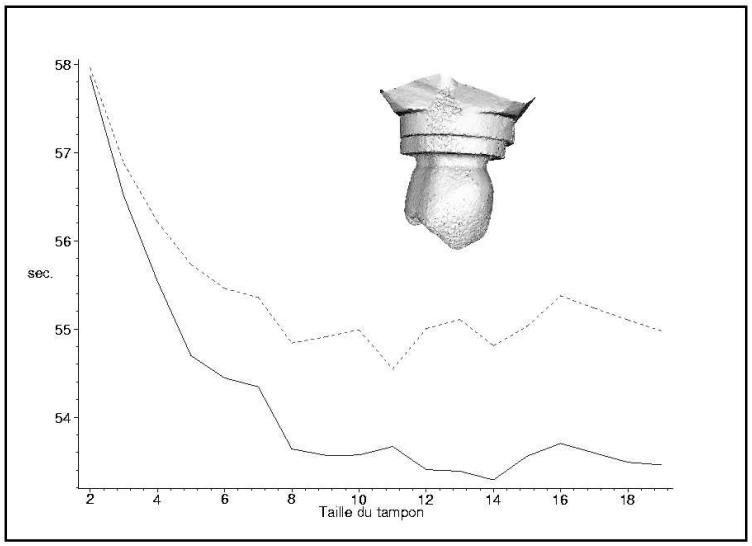
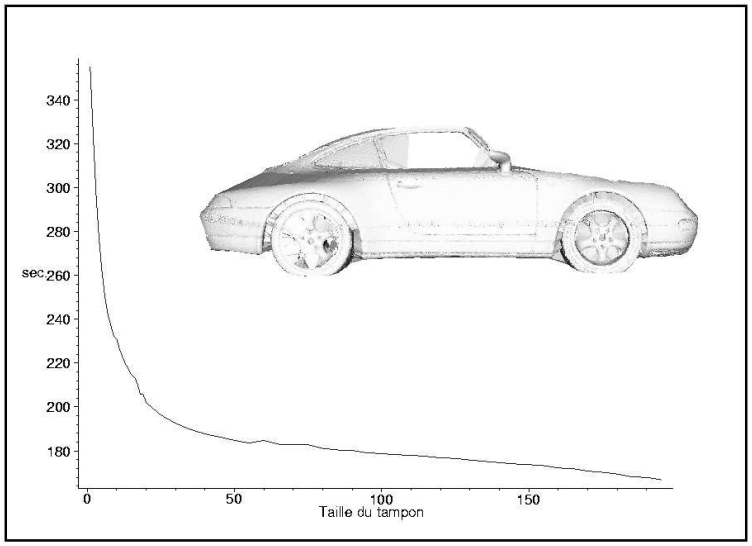


FIG. 15 – *Tétrahedrisation de Delaunay de 407500 points (voiture).*



5 Cloisonnement de segments

Cette section présente un algorithme incrémental de construction du cloisonnement vertical d'un ensemble de segments du plan. Ce type de construction permet entre autres de déterminer tous les points d'intersection d'un ensemble $\mathcal{S}$ de segments de droite dans le plan.

Construire le *cloisonnement* d'un ensemble de segments c'est subdiviser chacune des régions définies comme les parties connexes de $E^2 \setminus \mathcal{S}$ en régions élémentaires trapézoïdales. Cette décomposition du plan peut être considérée comme le prototype de toute une classe de décomposition géométriques analogues, appelées aussi *cloisonnements.*

5.1 Un algorithme incrémental

L'algorithme présenté ici calcule en ligne les intersections d'un ensemble $\mathcal{S}$ de segments en maintenant, chaque fois qu'un nouveau segment est introduit, le cloisonnement vertical du sous-ensemble courant, $\mathcal{R} \subseteq \mathcal{S}$.

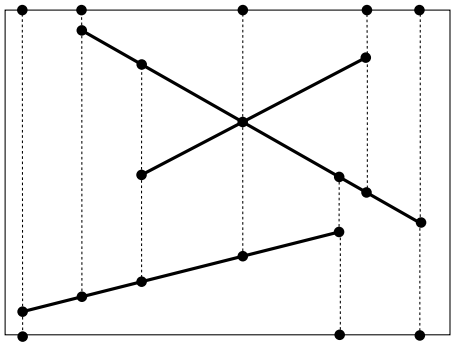


FIG. 16 – *Le cloisonnement vertical d'un ensemble de trois segments du plan*

Le *cloisonnement vertical* $Cl(\mathcal{S})$ du plan pour un ensemble de segments $\mathcal{S}$ peut être défini comme une carte planaire obtenue en traçant, à partir de chaque extrémité et de chaque intersection des segments, deux segments verticaux (vers le haut et vers le bas) que l'on étend jusqu'à ce qu'ils rencontrent un segment de $\mathcal{S}$ (voir figure 16) ou un segment de la boite englobant l'ensemble des segments de $\mathcal{S}$.

Chaque région de la carte ainsi définie a la forme d'un trapèze dont les côtés parallèles sont verticaux. Certains trapèzes sont dégénérés en un triangle (avec un côté vertical).

On peut montrer facilement que $Cl(\mathcal{S})$ contient exactement $3(n+a)+1$ trapèzes, où n est le nombre de segments et a le nombre de points d'intersection des segments de $\mathcal{S}$.

L'étape courante traite un nouveau segment s de $\mathcal{S} \setminus R$ et met à jour le cloisonnement de la manière suivante: chacun des trapèzes en conflit avec s, c'est-à-dire intersectés par s, est divisé en au plus quatre *sous-régions* par le segment s (les cloisons des extrémités de s et les cloisons des points d'intersection de s avec les autres segments de R), de plus, toute cloison intersectée par le segment s doit être raccourcie: la portion de cette cloison qui ne contient plus d'extrémité de segments ni de point d'intersection disparaît et les sous-régions qui partagent une telle portion de cloison sont réunies en un seul trapèze du cloisonnement de $R \cup \{s\}$ (voir Figure 17).

Comme pour l'algorithme de triangulation, la complexité totale de l'algorithme est égale à la somme du coût de localisation et du coût d'insertion dans le cloisonnement courant de chacun des segments à traiter. Il est assez facile de voir que le coût d'insertion d'un segment s est proportionnel au nombre de trapèzes de la décomposition courante en conflit avec s. Le coût de localisation, quant à lui, dépend très fortement de l'algorithme.

Nous nous intéressons, à nouveau, à l'analyse de la construction incrémentale du cloisonnement de segments en termes de coût d'insertion uniquement.

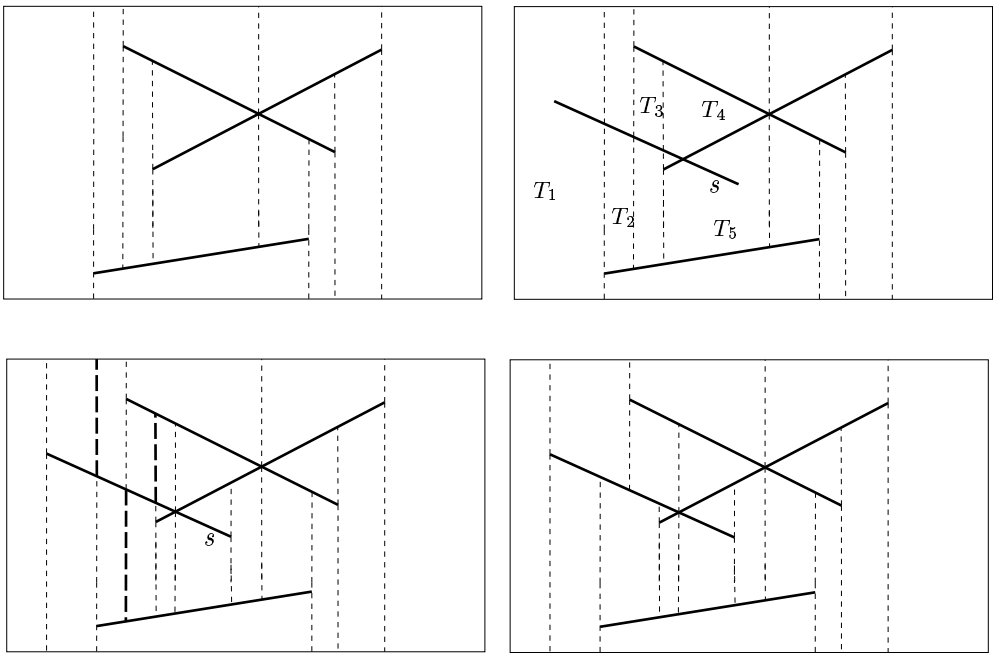


FIG. 17 – *Construction incrémentale du cloisonnement d'un ensemble de segments du plan: en haut à droite, le cloisonnement avant l'introduction du segment s, en haut à gauche, les trapèzes en conflit avec le segment s, en bas à droite, la division de ces trapèzes en sous-régions et, en trait pointillés gras, les portions de cloison à effacer, en bas à droite, le cloisonnement après l'introduction de s*

5.2 Analyse de la construction incrémentale du cloisonnement de segments

Le coût d'insertion d'un segment s dans le cloisonnement courant $Cl(\mathcal{R})$ $\mathcal{R} \subseteq \mathcal{S}$ est égal au nombre de trapèzes apparaissant dans $Cl(\mathcal{R} \cup \{s\})$ qui n'existaient pas dans $Cl(\mathcal{R})$.

Lors de l'introduction d'un nouveau segment s, celui-ci crée 4 nouveaux trapèzes (un nouveau trapèze ayant s comme plancher, un ayant s comme plafond et un trapèze pour chacune des deux nouvelles cloisons verticales passant par les extrémités de s). Pour chaque intersection entre s et un segment appartenant au sous-ensemble courant on crée aussi 4 nouveaux trapèzes (deux nouveaux trapèzes pour chaque cloison verticale passant par un point d'intersection et deux trapèzes n'ayant qu'une seule cloison verticale). Enfin, on crée un nouveau trapèze pour chacune des cloisons verticales intersectées par s.

Le nombre de nouveaux trapèzes construits lors de l'introduction de s est donc égal à:

$$4 + 4a_s + l_s$$

où a_s désigne le nombre d'intersection entre s et l'ensemble des segments présents dans le cloisonnement courant juste avant l'insertion de s, et l_s le nombre de cloisons intersectées par s lors de son introduction dans le cloisonnement courant.

5.2.1 Analyse dans le pire des cas

Theoreme 5.1 *Le nombre de trapèzes crées dans le pire des cas lors d'une construction incrémentale du cloisonnement de segments est* $\Theta(n^2\alpha(n))$.

Preuve: Pour un ordre sur les données totalement déterminé par l'adversaire on obtient alors le coût d'insertion suivant:

$$\text{Coût d'insertion du } p^{eme} \text{ segment} = 4 + 4a_{x_p} + l_{x_p}$$

En remarquant que la taille d'une enveloppe inférieure de n segments est égale à $n\alpha(n)$ [19] où $\alpha(n)$ désigne l'inverse de la fonction d'Ackermann, dont la croissance est extrêmement lente[1] et qu'il existe exactement $2(p-1)+a_{p-1}$ cloisons verticales dans le cloisonnement courant juste avant l'introduction du p^{eme} segment, où a_{p-1} désigne cette fois le nombre d'intersection entre les $p-1$ segments déjà insérés, on majore ce coût par:

$$\text{Coût d'insertion du } p^{eme} \text{ segment} \leq 4 + 4a_{x_p} + 2(p-1) + min(p\alpha(p), a_{p-1})$$

Le coût total d'insertion des n segments est alors obtenu en sommant cette valeur sur l'ensemble des n segments à traiter, soit:

$$\begin{aligned} \text{Coût d'insertion total} &\leq \sum_{i=1}^{n}(4 + 4a_{x_i} + 2(i-1) + min(i\alpha(i), a_{i-1})) \\ &\leq 4n + 4a + n(n-1) + min(\frac{n^2}{2}\alpha(n), an) \\ &\leq n^2 + 3n + 4a + min(\frac{n^2}{2}\alpha(n), an) \end{aligned} \quad (3)$$

Le coût d'insertion des n segments est donc presque quadratique ($\Theta(n^2\alpha(n)+a)$) dans le pire des cas. □

5.2.2 Analyse randomisée

Theoreme 5.2 *Le nombre moyen de trapèzes crées lors d'une construction incrémentale du cloisonnement de segments est* $\Theta(n+a)$.

Preuve: Une nouvelle fois, nous allons utiliser l'analyse en arrière afin d'effectuer une analyse randomisée de l'algorithme.

L'idée principale est d'exprimer le coût d'insertion du r^{eme} segment x_r dans $Cl_S(\mathcal{R}')$ en fonction du graphe résultat $Cl_S(\mathcal{R})$ où $\mathcal{R} = \mathcal{R}' \cup \{s\}$ et non de $Cl_S(\mathcal{R}')$.

On en déduit que si s est choisi de manière aléatoire à partir des r segments de $\mathcal{R}$, alors le coût moyen d'insertion de s dans $Cl_S(\mathcal{R}')$ vaut:

$$\begin{aligned} \text{Coût d'insertion du } r^{eme} \text{ segment} &= \frac{1}{r}\sum_{s\in\mathcal{R}}(4 + 4a_s + l_s) \\ &= 4 + 2.4.\frac{a_{\mathcal{R}}}{r} + \sum_{s\in\mathcal{R}}\frac{l_s}{r} \end{aligned}$$

où a_s désigne le nombre d'intersection entre s et l'ensemble des $r-1$ segments présents dans le cloisonnement courant et l_s le nombre de cloisons intersectées par s lors de son introduction dans le cloisonnement des $r-1$ premiers segments.

Il nous reste donc à calculer la valeur moyenne de $a_{\mathcal{R}}$, nombre de point d'intersection moyen entre les segments du r-échantillon $\mathcal{R}$, en fonction de a, nombre total d'intersections dans l'ensemble $\mathcal{S}$.

1. La fonction $\alpha(n)$ a une valeur inférieure à 4 pour toutes les valeurs de n envisageables en pratique.

Si $\mathcal{R}$ est un échantillon aléatoire de cardinal r de $\mathcal{S}$, l'espérance mathématique du nombre d'intersections $a_{\mathcal{R}}$ est:

$$\frac{a.r.(r-1)}{n(n-1)}$$

En effet, un point d'intersection I entre deux segments de $\mathcal{S}$ est un point d'intersection entre deux segments de $\mathcal{R}$ si et seulement si les deux segments de $\mathcal{S}$ qui s'intersectent au point I appartiennent à $\mathcal{R}$, ce qui arrive avec la probabilité $\frac{C_{n-2}^{r-2}}{C_n^r} = \frac{r(r-1)}{n(n-1)}$.

Le cloisonnement des r segments de l'échantillon aléatoire construit $2r + a_{\mathcal{R}}$ cloisons verticales, chacune d'entre elles est coupée au maximum une fois par un segment de $\mathcal{R}$, on obtient donc:

$$\frac{1}{r}\sum_{s\in\mathcal{R}} l_s \quad \leq \quad \frac{1}{r}(2r + a_{\mathcal{R}}) = 2 + \frac{a(r-1)}{n(n-1)}$$

Ainsi le coût moyen d'insertion du r^{eme} segment dans $Cl_S(\mathcal{R})$ est borné par:

$$4 + 8\frac{a(r-1)}{n(n-1)} + 2 + \frac{a(r-1)}{n(n-1)}$$

Il suffit alors de sommer cette expression pour $1 \leq r \leq n$, c'est-à-dire l'ensemble des n segments de $\mathcal{S}$ à traiter, pour obtenir le coût moyen d'insertion de l'algorithme, soit:

$$\text{Coût d'insertion moyen} \leq \sum_{r=1}^{n}\left(6 + 9\frac{a(r-1)}{n(n-1)}\right) = 6n + \frac{9a}{2} = \Theta(n+a) \tag{4}$$

Le coût d'insertion total calculé en moyenne sur tous les ordres d'insertion possibles est donc clairement linéaire par rapport au nombre de segments et d'intersections. □

5.2.3 Analyse première stratégie

Theoreme 5.3 *Le nombre moyen de trapèzes crées lors d'une construction incrémentale du cloisonnement de segments utilisant le tampon mélangeur et la première stratégie est* $\Theta(n^2\alpha(n)\frac{\log k}{k})$.

Preuve: L'analyse de l'algorithme utilisant la première stratégie se mène en considérant le coût d'insertion dans le cloisonnement courant du dernier des k segments appartenant au tampon courant $\mathcal{B}^k$, soit:

$$\text{Coût d'insertion moyen de } x_n = \frac{1}{k}\sum_{s\in\mathcal{B}^k}(4 + 4a_s + l_s)$$

$$\leq \quad 4 + \frac{4a_p}{k} + \frac{1}{k}\sum_{s\in\mathcal{B}^k} l_s$$

$$\leq \quad 4 + \frac{4}{k}a_p + \frac{1}{k}\sum_{s\in\mathcal{S}} l_s \leq 4 + \frac{4}{k}a_p + \frac{2n + min(a, n\alpha(n))}{k} \tag{5}$$

où a_p désigne le nombre d'intersections sur les segments du p^{eme} tampon
$(a_p = \sum_{s\in\mathcal{B}^k_{(p-1)k+1}} a_s = \sum_{s\in\mathcal{B}^k_{(p-1)k+1}} |s \cap \mathcal{S}|)$.

On obtient une borne supérieure pour le coût moyen d'insertion du $(pk+i)^{eme}$ segment $(1 \leq i \leq k)$ inséré dans le cloisonnement, en considérant, comme dans l'analyse de l'algorithme de triangulation, l'ensemble des segments

insérés après celui-ci fixés, le $(pk+i)^{eme}$ segment est alors choisi parmi les i segments restant. Il suffit, ensuite, de sommer ce coût pour $1 \leq i \leq k$, c'est-à-dire l'ensemble des k segments appartenant au tampon, pour obtenir le coût moyen d'insertion du p^{eme} tampon, soit:

$$\begin{aligned} \text{Coût d'insertion du } p^{eme} \text{ tampon} &\leq \sum_{i=1}^{k} \left(4 + \frac{4}{k}a_p + \frac{2(pk+i) + min(a,n\alpha(n))}{i}\right) \\ &\leq 4k + 4a_p + 2pkH_k + 2k + min(a,n\alpha(n))H_k \end{aligned}$$

En sommant sur le nombre de paquets de k sites, c'est-à-dire pour $0 \leq p \leq \lfloor \frac{n}{k} \rfloor$, et en remarquant que $\sum_{p=0}^{\lfloor \frac{n}{k} \rfloor} a_p \leq 2a$, on obtient enfin une borne supérieure sur le nombre moyen de trapèzes construits au cours du processus incrémental traitant n segments:

$$\text{Coût d'insertion moyen} \leq \sum_{p=0}^{\lfloor \frac{n}{k} \rfloor} (4k + 4a_p + 2pkH_k + 2k + min(a,n\alpha(n))H_k) \tag{6}$$

$$E[Z] \leq \frac{H_k}{k}n^2 + 8a + 6(n+k) + min(a,n\alpha(n))H_k(\frac{n}{k}+1) + nH_k$$

□

Le nombre moyen de trapèzes construits au cours du processus incrémental utilisant la première stratégie, $E[Z]$, est donc presque quadratique $\Theta(\frac{n^2\alpha(n)\log k}{k} + a)$ par rapport au nombre de segments à traiter comme pour le résultat obtenu dans le pire des cas. On remarque que l'on a, toutefois, réduit du facteur constant $\frac{H_k}{k}$ la constante du terme quadratique par rapport au résultat précédemment obtenu (cf .3) pour des valeurs de $k << n$.

Pour $k = n$, on trouve un coût d'insertion en $\Theta(n\log(n) + a)$ qui ne correspond pas exactement au résultat obtenu sous l'hypothèse randomisée en (cf. 4). Le terme en $O(n\log n)$ au lieu du terme linéaire $O(n)$ provient du fait que l'on a supposé dans les calculs que l'on avait plusieurs tampons c'est-à-dire $(k < n)$ et disparaît dans l'équation (6) si $p = 0$.

6 Conclusion

Les différentes stratégies présentées utilisant la technique du tampon mélangeur nous permettent d'obtenir une bonne amélioration de la complexité des algorithmes incrémentaux en ligne analysés (tri, triangulation de Delaunay, cloisonnement de segments). Pour l'ensemble de ces problèmes, la première des deux stratégies se révèle être la plus efficace. Nous arrivons à gagner un facteur $\frac{H_k}{k}$, fonction de la taille du tampon, sur leur complexité; pour des valeurs de k petites devant n, on garde un complexité quadratique mais on réduit la constante cachée dans l'expression $O(n^2)$.

On rappelle que pour les algorithmes géométriques analysés, le calcul de la complexité s'effectue en terme de coût d'insertion uniquement sans considérer le coût de localisation qui dépend trop fortement de la structure de donnée utilisée.

Le principe d'un tampon mélangeur s'intercalant simplement entre le processus qui fournit les données et l'entrée de l'algorithme a l'avantage de ne nécessiter aucune modification de l'algorithme utilisé. Toutefois, si l'ordre des données est complètement fixé, il peut être préférable, plutôt que d'utiliser cette technique, d'effectuer une modification de l'algorithme afin de tirer parti de cette connaissance de l'ordre et minimiser la complexité de l'algorithme. Dans la pratique, cependant, comme par exemple en reconstruction 3D ou en compression géométrique, où l'ordre

sur les données en entrée est donné par des capteurs laser ou par la technique de compression utilisé, on obtient un ordre d'insertion des données plus ou moins structuré en paquets de proximité, c'est-à-dire, ni randomisé, ni complètement ordonné, pour lequel il est avantageux d'utiliser la structure de tampon mélangeur.

Après avoir analysé le fonctionnement de cette technique de randomisation sur quelques problèmes caractéristiques, on pourrait être tenté de faire rentrer ceux-ci dans un formalisme plus général. Cependant, cette approche pose un problème majeur: avec un tel formalisme, on perd en précision ce que l'on gagne en généralité car les contraintes ou particularités de chaque problèmes ne peuvent être prises en compte (une contrainte typique au problème d'intersection de segments, par exemple, et que l'on ne peut avoir plus de $n-1$ intersections pour un segment donné); si l'on ne veut pas perdre en précision et tenir compte de toutes ces particularités, on obtient en contre parti des notations trop artificielles.

On peut malgré tout affirmer que de nombreux problèmes répondent au schéma suivant: si la taille du résultat d'un problème donné est $O(n)$ alors une analyse randomisée, c'est-à-dire en moyenne sur l'ensemble des ordres d'insertion possibles, nous donne une complexité $O(n \log n)$, l'analyse dans le pire des cas nous donne une complexité $O(n^2)$ qui devient avec l'utilisation de la première stratégie proposée $O(n^2 \frac{log k}{k})$ en moyenne.

7 Annexe

7.1 Stratégie 2

Le l^{eme} objet a_l fourni par l'adversaire est le i^{eme} objet x_i inséré dans l'arbre avec la probabilité:

$$Proba(x_i = a_l) = \left(\frac{k-1}{k}\right)^{i-1} \frac{1}{k} \quad si \quad l < k \quad et \quad i \leq n-(k-1) \tag{7}$$

(l'objet n'est pas choisi de la 1^{ere} à la $i-1^{eme}$ étape, et est choisi avec la probabilité $\frac{1}{k}$ à l'étape i)

$$Proba(x_i = a_l) = 0 \quad si \quad l > i+(k-1) \tag{8}$$

(l'objet ne peut être choisi car il n'a pas encore été place dans le tampon)

$$Proba(x_i = a_l) = \frac{(k-1)^{i-l+(k-1)}}{k^{i-l+k}} \quad si \quad l \leq i+(k-1) \quad et \quad i \leq n-(k-1) \tag{9}$$

(l'objet n'est pas choisi de la $l-(k-1)^{eme}$ à la $i-1^{eme}$ étape)

$$Proba(x_i = a_l) = \left(\frac{k-1}{k}\right)^{n-l+1} \frac{1}{k-1} \quad si \quad l < i+(k-1) \quad et \quad n \geq i > n-(k-1) \tag{10}$$

(l'objet n'est pas choisi de la $l-(k-1)^{eme}$ à la $n-(k-1)^{eme}$ étape avec la probabilité $\frac{k-1}{k}$ et n'est pas choisi de la $n-k+2$ à la $i-1^{eme}$ étape avec la probabilité $\frac{k-1-j}{k-j}$)

Enfin, le cas le moins probable:

$$Proba(x_i = a_l) = \left(\frac{k-1}{k}\right)^{n-(k-1)} \frac{1}{k-1} \quad si \quad l < k \quad et \quad i > n-(k-1) \tag{11}$$

(l'objet n'est pas choisi de la 1^{ere} à la $n-(k-1)^{eme}$ étape avec la probabilité $\frac{k-1}{k}$ et n'est pas choisi de la $(n-k+2)^{eme}$ à la $(i-1)^{eme}$ avec la probabilité $\frac{k-j-1}{k-j}$)

7.1.1 Tri: exemple $k = 2$, ordre croissant

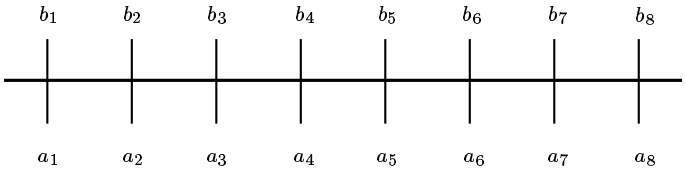


FIG. 18 – *Ordre croissant*

Supposons, dans un premier temps, que l'adversaire nous donne les données selon un ordre croissant (cf. figure 18).

En remarquant que

$$Proba(X_{ij} = 1) = 1 - Proba(X_{ij} = 0) \tag{12}$$

où $Proba(X_{ij} = 0)$ est la probabilité de ne pas comparer b_i et b_j (c'est-à-dire la probabilité d'insérer un nombre compris entre b_i et b_j avant b_i ou b_j) et pour un tampon mélangeur de taille $k = 2$ on obtient:

$Proba(X_{ij} = 1) = 1$ si $j = i + 1$
(il n'existe aucun nombre entre b_i et b_j, la probabilité d'insérer un nombre compris entre b_i et b_j est nulle.)

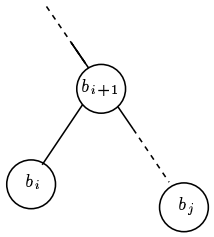


FIG. 19 – *Cas* $j > i + 1$

$Proba(X_{ij} = 1) = 1 - \frac{1}{4}$ si $j > i + 1$
(la probabilité de choisir b_{i+1} avant b_i est égale à la probabilité de ne pas choisir b_i lors de son insertion dans le tampon et de choisir b_{i+1} à l'étape suivante c'est-à-dire $\frac{1}{2}.\frac{1}{2}$ (cf. figure 19)).

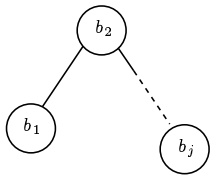


FIG. 20 – *Cas* $i = 1$ *et* $j > 2$

Le premier nombre b_1 inséré dans le tampon est traité de manière particulière, en effet:
$Proba(X_{1j} = 1) = 1 - \frac{1}{2}$ si $j > 2$
(avant la première insertion b_1 et b_2 appartiennent au tampon on choisit b_2 avec la probabilité $\frac{1}{2}$ (cf. figure 20)).

En remarquant que $X = \sum_{j=3}^{n} X_{1j} + \sum_{i=1}^{n-1} X_{ii+1} + \sum_{i=2}^{n-2} \sum_{j=i+2}^{n} X_{ij}$

Le nombre moyen de comparaisons effectuées est alors:

$$\begin{aligned} E[X] &= \sum_{j=3}^{n} Proba(X_{1j} = 1) + \sum_{i=1}^{n-1} Proba(X_{ii+1} = 1) + \sum_{i=2}^{n-2} \sum_{j=i+2}^{n} Proba(X_{ij} = 1) \\ &= \sum_{j=3}^{n} \frac{1}{2} + \sum_{i=1}^{n-1} 1 + \sum_{i=2}^{n-2} \sum_{j=i+2}^{n} \frac{3}{4} = \frac{(n-2)}{2} + (n-1) + \sum_{i=2}^{n-2} \frac{3(n-i-1)}{4} \\ &= \frac{(n-2)}{2} + (n-1) + \frac{3(n-3)(n-2)}{8} = \frac{3n^2}{8} - \frac{3n}{8} + \frac{1}{4} \end{aligned}$$

On réduit dans ce cas la constante cachée c de $\frac{3}{4}$ par rapport au nombre de comparaisons dans le pire des cas.

7.1.2 Tri: exemple $k = 2$, ordre alterné

L'ordre croissant imposé par l'adversaire n'est cependant pas le cas le plus défavorable pour notre problème. Considérons, en effet, l'ordre alterné décrit figure 4, on obtient alors:

$$Proba(X_{ij} = 1) = 1 \quad si \quad j = i + 1 \tag{13}$$

(il n'existe aucun nombre entre b_i et b_j, la probabilité d'insérer un nombre compris entre b_i et b_j est nulle).

$$Proba(X_{ij} = 1) = 1 \quad si \quad j > i + 1 \quad avec \quad i + j = n + 1 \quad ou \quad i + j = n + 2 \tag{14}$$

(b_i et b_j se succèdent dans l'ordre de l'adversaire, on ne peut choisir un nombre compris entre b_i et b_j avant b_i ou b_j).

$$Proba(X_{ij} = 1) = 1 - \frac{1}{8} \quad si \quad j > i + 1 \quad avec \quad i + j \neq n + 1 \quad et \quad i + j \neq n + 2 \tag{15}$$

(la probabilité de choisir b_{i+1} avant b_i est égale à la probabilité de ne pas choisir b_i et de choisir b_{i+1} lors de son insertion c'est-à-dire $\frac{1}{2}.\frac{1}{2}.\frac{1}{2}$ puisque b_{i+1} est inséré deux coups après b_i).

Le 1^{er} nombre b_1 inséré dans le tampon est traité de manière particulière, en effet:

$$Proba(X_{1j} = 1) = 1 - \frac{1}{4} \quad si \quad 2 < j < n \tag{16}$$

(avant la première insertion b_1 et b_n appartiennent au tampon, on choisit b_3 avant b_1 avec la probabilité $\frac{1}{4}$).

En remarquant qu'il existe respectivement $(n-1)$, $(n-2)$ et $(n-3)$ couples (i,j) vérifiant (13) (14) et (16). Le nombre moyen de comparaisons effectuées est alors:

$$\begin{aligned} E[X] &= \frac{(n-3)(n-4)}{2}(1 - \frac{1}{8}) + (n-1) + (n-2) + (n-3)(1 - \frac{1}{4}) \\ &= \frac{7}{16}(n-3)(n-4) + 2n - 3 + \frac{3}{4}(n-3) \\ &= \frac{7}{16}n^2 - \frac{5}{16}n \end{aligned}$$

La constante cachée c est seulement réduite de $\frac{7}{8}$, l'ordre alterné est donc plus défavorable que l'ordre trié avec cette stratégie et pour un tampon de taille $k = 2$.

7.1.3 Tri: exemple $k = 3$, ordre alterné

Parmi l'ensemble des $\frac{n(n-1)}{2}$ couples (i,j), on obtient selon les couples (i,j), les probabilités $Proba(X_{ij} = 1)$ suivantes:

- il existe $(n-1)$ couples (i,j) tels que $j = i + 1$.
  Dans ce cas, il n'existe aucun nombre compris entre b_i et b_j, la probabilité d'insérer un nombre compris entre

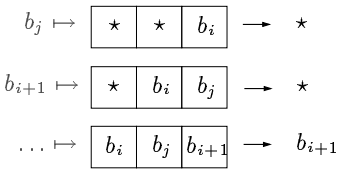


FIG. 21 – *Cas $j+i=n+1$, on choisit b_{i+1} avant b_i et b_j avec la probabilité $\frac{2.1.1}{3.3.3}$. Les $\star$ désignent les nombres fournis par l'adversaire non compris entre b_i et b_j.*

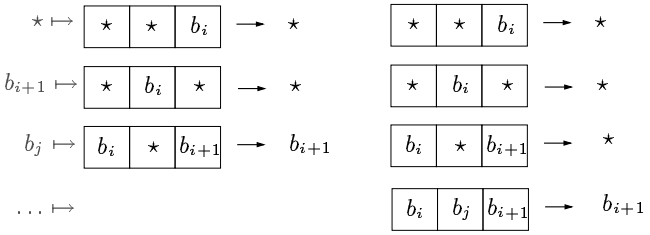


FIG. 22 – *Cas $j+i=n$*

b_i et b_j avant b_i et b_j est nulle.

$$Proba(X_{ij}=1)=1 \quad j=i+1 \tag{17}$$

- il existe moins de n couples tels que $j+i=n+1$ ou $j+i=n+2$.
  Pour le cas $j+i=n+1$ (resp. $j+i=n+2$), l'ensemble des nombres fournis par l'adversaire après b_i et b_j appartiennent à l'intervalle $[b_i,b_j]$, de plus, b_i et b_j sont consécutifs selon l'ordre de l'adversaire.
  La probabilité de ne pas comparer b_i et b_j revient à calculer la probabilité de ne pas choisir b_i (resp. b_j) lors de son insertion dans le tampon, de ne choisir ni b_i ni b_j le coup d'après et enfin de choisir b_{i+1} (resp. b_{j-1}) (cf. figure 21).

$$Proba(X_{ij}=1)=1-\frac{2}{3}\frac{1}{3}\frac{1}{3}=\frac{25}{27} \quad j+i=n+1 \quad ou \quad j+i=n+2 \tag{18}$$

- il existe moins de n couples (i,j) tels que $j+i=n+3$ et $j+i=n$.
  Pour le cas $j+i=n$ (resp. $j+i=n+3$), b_j (resp. b_i) est fourni trois coups après b_i (resp. b_j). Un seul nombre, b_{i+1} (resp. b_{j-1}), fourni entre b_i et b_j appartient à $[b_i,b_j]$, celui-ci est fourni par l'adversaire exactement deux coups après b_i (resp. b_j) et une étape avant b_j (resp. b_i) (cf. 22).

$$Proba(X_{ij}=1)=1-(\frac{2}{3}\frac{2}{3}\frac{1}{3}+\frac{2}{3}\frac{2}{3}\frac{1}{3}\frac{1}{3})=\frac{65}{81} \quad j+i=n+3 \quad ou \quad j+i=n \tag{19}$$

- il existe moins de n couples tels que $j=i+2$ avec $i,j<\lceil\frac{n+1}{2}\rceil$ ou $i,j>\lceil\frac{n+1}{2}\rceil$.
  Dans le cas $i,j<\lceil\frac{n+1}{2}\rceil$ (resp. $i,j>\lceil\frac{n+1}{2}\rceil$), b_j (resp. b_i) est fourni quatre coups après b_i (resp. b_j).
  Parmi les trois nombres fournis par l'adversaire entre b_i et b_j, un seul appartient à $[b_i,b_j]$ et est fourni exactement deux coups après b_i (cf. 23).

$$Proba(X_{ij}=1)=1-(\frac{2}{3}\frac{2}{3}\frac{1}{3}+\frac{2}{3}\frac{2}{3}\frac{1}{3}\frac{1}{3}+\frac{2}{3}\frac{2}{3}\frac{1}{3}\frac{1}{3}\frac{1}{3})=\frac{191}{243} \quad j=i+2 \tag{20}$$

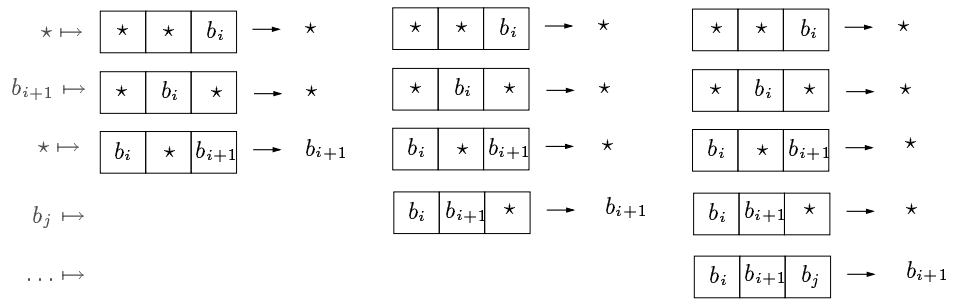


FIG. 23 – *Cas* $j = i + 2$ *avec* $i,j < \lceil \frac{n+1}{2} \rceil$ *ou* $i,j > \lceil \frac{n+1}{2} \rceil$

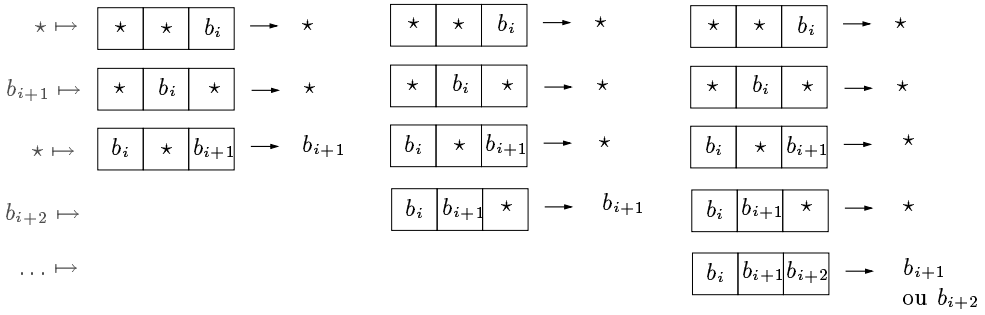


FIG. 24 – *Cas général*

- Les deux premiers nombres ($a_1 = b_1$ et $a_2 = b_n$) fournis par l'adversaire sont traités de manière particulière, il existe moins de n couples tels que $i = 1$ avec $Proba(X_{1j}) \leq 1$ et moins de n couples tels que $j = n$ avec:

$$Proba(X_{in}) \leq 1 \tag{21}$$

- Pour tout les autres couples (i,j), il existe au moins deux nombres compris entre b_i et b_j parmi ceux fournis par l'adversaire entre b_i et b_j. Après l'insertion du premier des deux nombres b_i et b_j fourni par l'adversaire dans le tampon, les nombres compris entre b_i et b_j sont fournis tout les deux coups (cf. 24).

$$Proba(X_{ij} = 1) = 1 - (\frac{2}{3}\frac{2}{3}\frac{1}{3} + \frac{2}{3}\frac{2}{3}\frac{1}{3}\frac{1}{3} + \frac{2}{3}\frac{2}{3}\frac{1}{3}\frac{1}{3}\frac{2}{3}) = \frac{187}{243} \quad j > i + 2 \tag{22}$$

En majorant simplement l'ensemble des probabilités associées aux cas différents du cas général par 1, on obtient la borne supérieure suivante pour le nombre moyen de comparaisons effectuées:

$$\begin{aligned} E[X] &\leq (\frac{n(n-1)}{2} - 6n)\frac{187}{243} + 6n \\ &= \frac{187}{486}n^2 - \frac{485}{486}n \end{aligned}$$

Références

[1] J.-D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec. Applications of random sampling to on-line algorithms in computational geometry. *Discrete Comput. Geom.*, 8:51–71, 1992.

[2] Jean-Daniel Boissonnat and Monique Teillaud. A hierarchical representation of objects: The Delaunay tree. In *Proc. 2nd Annu. ACM Sympos. Comput. Geom.*, pages 260–268, 1986.

[3] Jean-Daniel Boissonnat and Monique Teillaud. On the randomized construction of the Delaunay tree. *Theoret. Comput. Sci.*, 112:339–354, 1993.

[4] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[5] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[6] O. Devillers. Robust and efficient implementation of the Delaunay tree. Rapport de recherche 1619, INRIA, 1992.

[7] O. Devillers. Improved incremental randomized Delaunay triangulation. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 106–115, 1998.

[8] P. J. Green and R. R. Sibson. Computing Dirichlet tessellations in the plane. *Comput. J.*, 21:168–173, 1978.

[9] Leonidas J. Guibas, D. E. Knuth, and Micha Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7:381–413, 1992.

[10] R. Karp. An introduction to randomized algorithms. *Discrete Appl. Math.*, 34:165–201, 1991.

[11] Rolf Klein, Kurt Mehlhorn, and Stefan Meiser. Randomized incremental construction of abstract Voronoi diagrams. *Comput. Geom. Theory Appl.*, 3(3):157–184, 1993.

[12] J.-M. Moreau. Hierarchical Delaunay triangulation. In *Proc. 6th Canad. Conf. Comput. Geom.*, pages 165–170, 1994.

[13] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, 1995.

[14] Ernst P. Mücke, Isaac Saias, and Binhai Zhu. Fast randomized point location without preprocessing in two- and three-dimensional Delaunay triangulations. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 274–283, 1996.

[15] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[16] K. Mulmuley and O. Schwarzkopf. Randomized algorithms. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 34, pages 633–652. CRC Press LLC, Boca Raton, FL, 1997.

[17] M. S. Paterson and F. F. Yao. On nearest-neighbor graphs. In *Proc. 19th Internat. Colloq. Automata Lang. Program.*, volume 623 of *Lecture Notes Comput. Sci.*, pages 416–426. Springer-Verlag, 1992.

[18] R. Seidel. Backwards analysis of randomized geometric algorithms. Report TR-92-014, Computer Science Division, University of California, Berkeley, February 1992.

[19] A. Wiernik and Micha Sharir. Planar realizations of nonlinear Davenport-Schinzel sequences by segments. *Discrete Comput. Geom.*, 3:15–47, 1988.

Table des matières

1 Introduction **3**

2 Le modèle du tampon mélangeur **3**
2.1 Définitions, Notations . . . 3
2.2 Première stratégie . . . 4
2.3 Deuxième stratégie . . . 4

3 Un exemple simple: le tri **4**
3.1 Un algorithme incrémental . . . 4
3.2 Analyse de l'insertion dans un arbre binaire dans le pire des cas . . . 5
3.3 Analyse de l'insertion dans un arbre binaire selon un ordre aléatoire . . . 6
3.4 Analyse première stratégie . . . 8
3.5 Analyse deuxième stratégie . . . 10
3.6 Borne inférieure . . . 12
3.7 Comparaison des résultats . . . 13

4 Triangulation de Delaunay **16**
4.1 Définition et résultats fondamentaux . . . 16
4.1.1 Diagramme de Voronoï . . . 16
4.1.2 Triangulation de Delaunay . . . 17
4.1.3 Propriétés . . . 17
4.2 Un algorithme incrémental . . . 18
4.3 Analyse de la construction incrémentale de la triangulation de Delaunay . . . 19
4.3.1 Analyse dans le pire des cas . . . 19
4.3.2 Borne supérieure première stratégie . . . 20
4.3.3 Analyse première stratégie pour des points sur la parabole . . . 21
4.4 Localisation . . . 23
4.5 Résultats . . . 24

5 Cloisonnement de segments **26**
5.1 Un algorithme incrémental . . . 26
5.2 Analyse de la construction incrémentale du cloisonnement de segments . . . 27
5.2.1 Analyse dans le pire des cas . . . 27
5.2.2 Analyse randomisée . . . 28
5.2.3 Analyse première stratégie . . . 29

6 Conclusion **30**

7 Annexe **32**
7.1 Stratégie 2 . . . 32
7.1.1 Tri: exemple $k = 2$, ordre croissant . . . 32
7.1.2 Tri: exemple $k = 2$, ordre alterné . . . 34
7.1.3 Tri: exemple $k = 3$, ordre alterné . . . 34

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399