2015

Calcule du rectangle d'air minimum et du cercle de diamètre minimum

CONCEPTION ET PRATIQUE DE L'ALGORITHMIQUE

ALEXANDRE ANNIC

Table des matières

Introduction

Ce rapport présente le résultat du projet de CPA. Il aborde le problème du rectangle d'air minimum et celui du cercle de diamètre minimum contenant un ensemble de points dans un plan en deux dimensions. Les domaines concernés par ces problématiques sont nombreux, notamment dans la géométrie algorithmique afin de détecter les collisions entre les objets.

Pour résoudre ces problèmes, deux algorithmes sont étudiés: l'algorithme Toussaint pour déterminer le rectangle d'air minimum et l'algorithme Ritter pour résoudre le cercle de diamètre minimum.

L'implémentation des algorithmes est réalisée en Java. La fonction implémentant l'algorithme Toussant est présente dans la classe `toussaint.Main` et celle implémentant l'algorithme Ritter est située dans la classe `ritter.Main`.

Problème du rectangle minimum

Principe

Le problème du rectangle minimum consiste à déterminer un rectangle d'air minimum contenant tous les points présent dans le plan.

Lemme 1 : Les segments du rectangle d'air minimum ne passent que par des points de l'enveloppe convexe.

Le lemme 1 permet d'effectuer un premier tri en supprimant tous les points ne faisant pas parti de l'enveloppe convexe. On diminue ainsi considérablement le nombre de points à évaluer. Les algorithmes permettant de déterminer l'enveloppe convexe sont nombreux et souvent performant, avec une complexité généralement en $O(n \log n)$. L'algorithme utilisé dans cette étude est la marche de Jarvis. Sa complexité est en $O(nh)$, où h représente le nombre de sommets de l'enveloppe convexe.

Lemme 2 : Le rectangle d'air minimum contenant tous les points d'un polygone convexe a un côté colinéaire avec l'un des côtés de ce polygone.

Le lemme 2 permet de grandement faciliter le calcul du rectangle d'air minimum. En effet, connaissant cette propriété, on sait qu'il existe autant de rectangles candidats que de côtés de l'enveloppe convexe. Un algorithme naïf consiste alors à calculer, pour chaque côté de l'enveloppe convexe, le rectangle de taille minimum colinéaire avec ce côté. La construction d'un tel rectangle est un processus ayant une complexité en $O(n)$. Appliqué à tous les côtés de l'enveloppe convexe, on obtient un algorithme de complexité temporelle quadratique.

Une solution plus efficace consiste à utiliser le principe d'un pied à coulisse qui pivote jusqu'à couvrir tous les côtés du polygone. La mise à jour du rectangle après rotation est une

opération s'effectuant en temps constant. L'algorithme complet est donc de complexité $O(n)$ où n est le nombre de côtés de l'enveloppe convexe.

Les deux figures ci-dessous détaillent la procédure effectuée. Dans un premier temps l'algorithme construit quatre droites formant un rectangle à partir des extrémités du polygone (figure 1). Il cherche ensuite le côté du polygone formant l'angle minimum avec l'un de ces quatre droites.

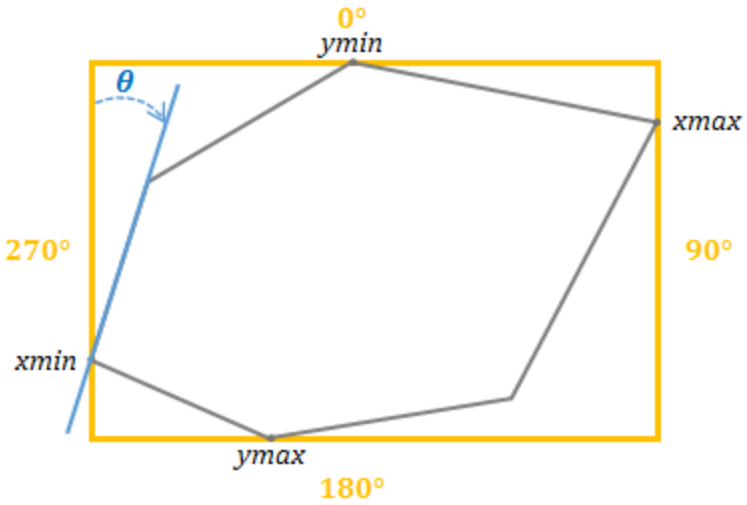

Figure 1 – Etat initial du rectangle

Il fait alors pivoter les quatre droites en ajustant leur position de manière à couvrir tous les points de l'enveloppe convexe (figure 2). Lorsque les droites ont effectuées un pivot d'au moins 90%, l'algorithme s'arrête et retourne le rectangle de taille minimum enregistré.

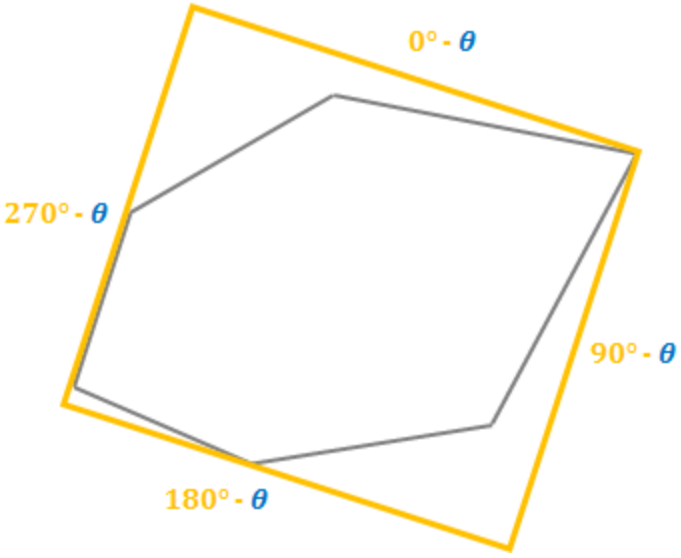

Figure 2 - Etat du rectangle après avoir pivoté

La procédure peut être résumée par les 7 étapes suivantes :

1. Calculer l'enveloppe convexe C de l'ensemble de points donnés.
2. Calculer les quatre extrémités de C : $xmin, xmax, ymin$ et $ymax$.
3. Construire quatre droites passant par un des points précédemment calculés de manière à former un rectangle.
4. Si au moins une droite est colinéaire avec un côté du polygone : calculer et sauvegarder l'air du rectangle en tant qu'air minimum.
5. Faire pivoter les droites jusqu'à ce que l'une d'entre elles soit colinéaire avec un des côtés du polygone.
6. Mettre à jour l'air minimum du rectangle si nécessaire
7. Répéter depuis l'étape 5 jusqu'à ce que les droites aient pivoté jusqu'à 90 degrés.

Complexité de l'algorithme Toussaint

La figure 3 montre le temps d'exécution de l'algorithme toussant pour un nombre de points donnés. L'axe des abscisses représente le temps en millisecondes et l'axe des ordonnés représente le nombre de points.

Les tests ont été effectués à partir de la base de test *VAROUMAS*. Cette base de test contient 1663 fichiers contenant chacun 254 instance de points soit un total de 422 402 points. Pour améliorer la précision, les tests ont été répétés dix fois chacun. La valeur du temps d'exécution est la moyenne de ces dix valeurs.

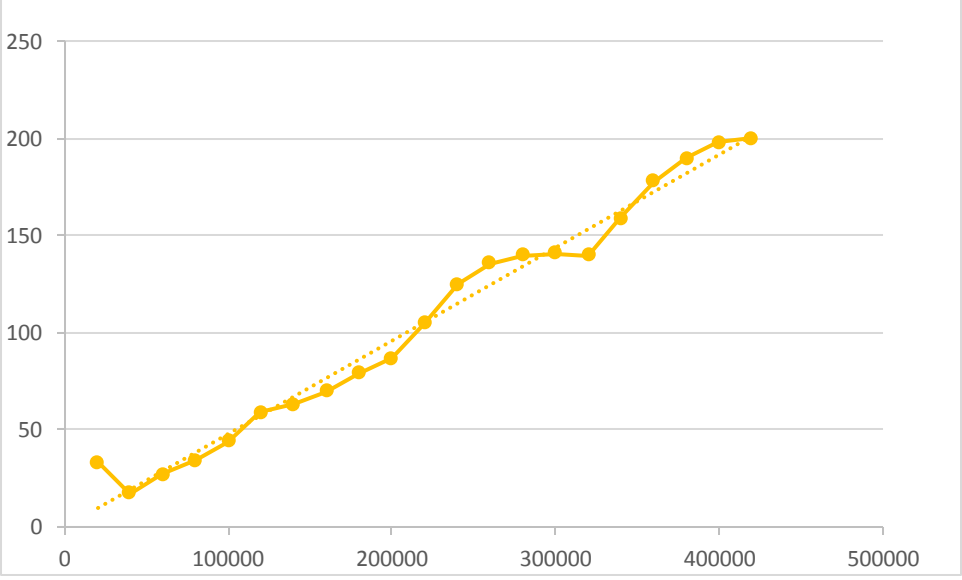

Figure 3 - Complexité de l'algorithme toussaint

La tendance de la courbe est linéaire, ce qui est en accord avec une complexité en $O(n)$. Le temps moyen d'exécution de l'algorithme pour les cas considérés est de 0,1 seconde ce qui est très rapide étant donné la quantité de points considérés.

Problème du cercle minimum

Principe

Le problème du cercle minimum consiste à déterminer, à partir d'un ensemble de points dans le plan, un cercle de rayon minimum contenant tous les points de l'ensemble.

Lemme 1 : Si le cercle dont le diamètre est formé par deux points de la liste couvre tous les points de la liste, alors ce cercle est un cercle couvrant de rayon minimum.

Lemme 2 : Si le cercle circonscrit au triangle formé par trois points non-colinéaires couvre tous les points de la liste, alors ce cercle est un cercle couvrant de rayon minimum.

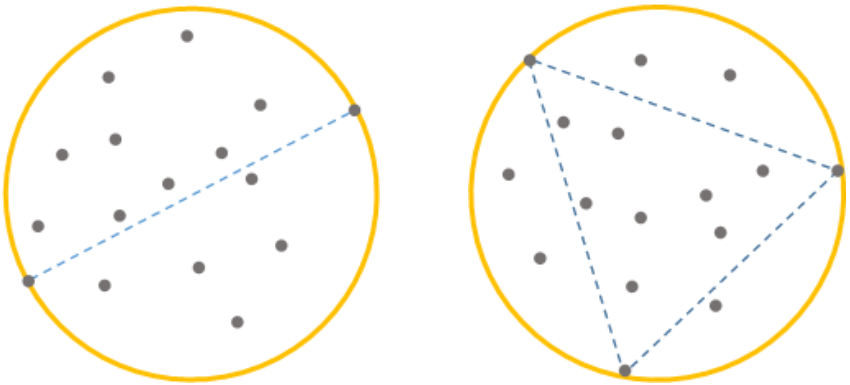

Figure 4 - Cercles minimums d'un ensemble de points

La figure 4 présente les deux cas de figure décrits par les lemmes 1 et 2. Dans chacun de ces cas, il est facile de voir qu'il est impossible de réduire la taille du cercle sans en exclure au moins un point.

Connaissant ces propriétés, il est aisé de concevoir un algorithme naïf pour résoudre le problème du cercle minimum.

Procedure CalculerCercleMinimum

$P \leftarrow$ ensemble de points
pour tout p de P
    pour tout q de P
        $c \leftarrow$ cercle de cntre $\frac{p+q}{2}$ de diamètre $|pq|$
        si c couvre tous les points de P
            retourner c
pour tout p de P
    pour tout q de P
        pour tout r de P
            $c \leftarrow$ cercle circonscrit de p, q et r

si c couvre tous les points de P
retourner c

Cet algorithme, simple à mettre en œuvre, a le défaut d'être très peu performant. En effet, il réalise trois boucles imbriquées sur l'ensemble des points à l'issue desquelles il effectue à nouveau une boucle pour vérifier que le cercle contient tous les points. Il a donc une complexité en $O(n^4)$.

L'algorithme *Ritter* est un algorithme probabiliste. Il calcule une approximation du cercle souvent plus grande que le résultat optimal. Il possède les avantages d'être rapide et facile à implémenter. Le diamètre du cercle retourné est habituellement entre 5% et 20% plus grand que le résultat optimal.

Le principe de cet algorithme est simple. On part d'un cercle de taille raisonnable et tant qu'un point est en dehors de ce cercle, on déplace le cercle jusqu'à celui-ci tout en augmentant son diamètre. L'agrandissement du cercle se fait en fonction du point considéré et de son diamètre initiale.

Procedure CalculeCercleMinimum

$Points \leftarrow$ ensemble de points
$dummy \leftarrow$ point quelconque de $Points$
$P \leftarrow$ point le plus éloigné de $dummy$ dans $Points$
$Q \leftarrow$ point le plus éloigné de P dans $Points$
$C \leftarrow$ centre du segment $[P, Q]$
$CP \leftarrow$ cercle de centre C et de rayon CP

tant que $Points$ n'est pas vide
$Points \leftarrow$ points de $Points$ qui ne sont pas dans le cercle CP
si $Points$ n'est pas vide
$S \leftarrow$ point quelconque de $Points$
$T \leftarrow$ point le plus éloigné de la droite passant par S et C coupant le cercle CP
$C \leftarrow$ centre du segment $[S, T]$

Il existe de nombreux autres algorithmes permettant de résoudre le problème du cercle de diamètre minimum de façon optimal et avec de bonnes performances. Notamment Algorithme de Shamos et Hoey de complexité $O(n \log n)$ ou encore l'algorithme de Megiddo en $O(n)$. Ils sont cependant plus difficiles à implémenter.

Complexité de l'algorithme Ritter

De la même manière que pour l'algorithme Toussaint, les tests ont été effectués à partir de la base de test *VAROUMAS*. Etant donné le temps d'exécution de l'algorithme Ritter sur un nombre aussi important de points, les tests ont été répétés seulement deux fois.

La figure 5 montre le temps d'exécution de l'algorithme toussant pour un nombre de points donné. L'axe des abscisses représente le temps en millisecondes et l'axe des ordonnés représente le nombre de points présents dans le plan.

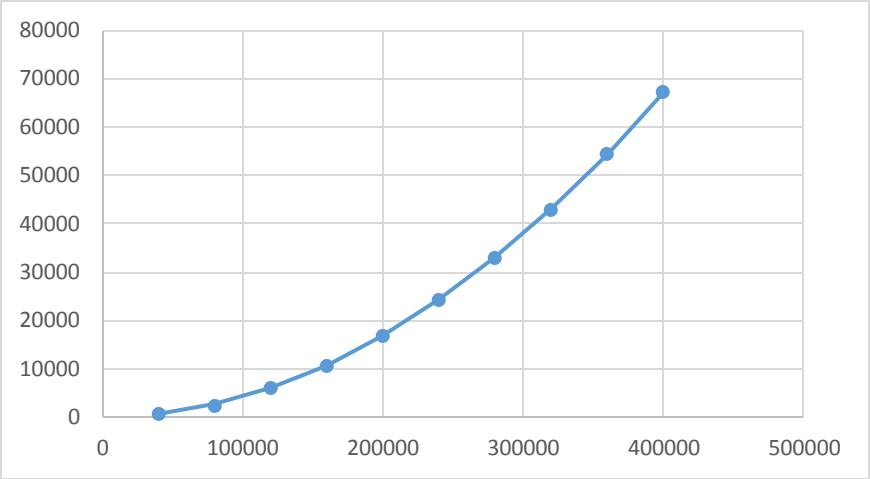

Figure 5 - Temps d'exécution de l'algorithme Ritter

La tendance de la courbe est légèrement exponentielle ce qui désigne une complexité en $O(n \log n)$.

Le temps moyen d'exécution de l'algorithme pour tous les cas testé est d'environ 26 secondes ce qui s'avère être très long par rapport à l'algorithme Toussaint.

Qualité des algorithmes

Les résultats obtenus lors de l'analyse de complexité des deux algorithmes concluent que l'algorithme Toussant permet d'obtenir un résultat beaucoup plus rapidement que l'algorithme Ritter quel que soit la taille du jeu de données.

Il est cependant intéressant de comparer la qualité des conteneurs fournis par chacun des algorithmes. Le tableau ci-dessous donne la proportion de l'augmentation de l'air des conteneurs retournés par les algorithmes par rapport à l'air des enveloppes convexes.

Valeurs	Minimale	Moyenne	Maximale
Algorithme Ritter	7,5 %	20,4 %	51 %
Algorithme Toussaint	21 %	25 %	32 %

Bien que l'algorithme Ritter retourne un cercle souvent plus grand que le cercle minimum, il constitue en moyenne un meilleur conteneur que le rectangle fournis par l'algorithme toussaint. Cependant, comparé aux temps d'exécution des deux algorithmes (0,1 seconde pour Toussaint, 26 secondes en moyenne pour Ritter), un gain sur la qualité en tant que conteneur de moins de 5% est très faible.

Conclusion

Bien que l'algorithme Ritter soit probabilisme, il s'avère être moins performant en terme de temps d'exécution et de complexité asymptotique que l'algorithme Toussaint. Cependant, même approximé, le cercle fournis par l'algorithme Ritter constitue un meilleur conteneur que le rectangle fournis par l'algorithme Toussaint.

Dans cette étude, la qualité en tant que conteneur n'est pas primordiale, puisque l'un des deux algorithmes étudiés est un algorithme probabiliste. Dans ce cas-là, la différence entre les qualités des conteneurs apparait comme négligeable face à la différence entre les temps d'exécution. L'algorithme Toussaint semble constituer un meilleur choix que l'algorithme Ritter.

Références

- https://geidav.wordpress.com/2014/01/23/computing-oriented-minimum-bounding-boxes-in-2d/
- http://fr.wikipedia.org/wiki/Problème_du_cercle_minimum
- http://en.wikipedia.org/wiki/Bounding_sphere
- http://fr.wikipedia.org/wiki/Marche_de_Jarvis
- http://www.mlahanas.de/CompGeom/opt_bbox.htm
- http://web.cs.swarthmore.edu/~adanner/cs97/s08/pdf/calipers.pdf