Rapport de projet

L'Algorithme Toussaint et Ritter pour le problème du conteneur minimum.

ROZOVYK Artemiy

Analyse et expérimentation.

Master 1 Sciences et Technologies du Logiciel

27 Décembre 2019

Résumé

Dans ce document on va décrire la démarche suivie en vue de résoudre le problème du rectangle minimum, ainsi que de cercle minimum couvrant un ensemble de points dans $\mathbb{R}^2$.
L'implémentation des deux algorithmes résolvant ces problèmes, notamment le *rectangle minimum* de G. T. Toussaint [1] ainsi que *l'approximation de cercle minimum* de J. Ritter [2] est présenté. Finalement, un ensemble d'expérimentations mettant en valeur la qualité de ces algorithmes est mis en place.

Chapitre 1

Algorithme de Toussaint

1.1 Introduction

Étant donné un ensemble fini de points P dans $\mathbb{R}^2$, on aimerait obtenir un rectangle d'aire minimum contenant ces points.

Tout d'abord on s'intéresse à l'enveloppe convexe qui représente un polygone d'aire minimum contenant P. Ensuite on l'utilise pour concevoir un premier algorithme naïf. On poursuit par l'implémentation d'algorithmes de Toussaint. Finalement, on compare les avantages et les inconvénients de cet algorithme et on cherche les alternatives.

1.2 Analyse des algorithmes existants

Il est facile d'imaginer que le rectangle d'aire minimum doit avoir au moins un de ses côtés adjacent avec le côté de l'enveloppe convexe.

Théorème 1 *Le rectangle d'aire minimum contenant l'enveloppe convexe possède un côté colinéaire avec l'un des côtés de l'enveloppe. [4]*

1.2.1 Soluction naive

Cela nous mène vers un premier algorithme naif [5] :

```
1  In: List<Points> P - nuage de points
2  Out: List<Points> R - [A; B; C; D] coins du rectangle
3  begin
4  a = b = c = d = infini
5  env = envelope convexe de P
6    pour chaque cote pq de env:
7         s = coin de env a dist. max de pq
8         D = droite passant par s, orthogonale a pq
9         t,u = coins de env a dist. max de D,
10              dans les deux plans de definis par D
11        G = droite passant par t, orthogonale a pq
12        L = droite passant par u, orthogonale a pq
13        F = droite passant par s, parallele a pq
14        a',b',c',d' = intersection de G,L,F et pq
15          si l'aire de [a'; b'; c'; d'] est plus petit que
16             [a ;b ;c ;d]:
17             a,b,c,d = a',b',c',d'
18 retourner [a; b; c; d]
19 end
```

Cet algorithme de force brute utilise le Théorème 1 de manière extrême : il trouve tous les rectangles adjacents à chaque côté de l'enveloppe. Sa complexité est $O(n^2)$ car dans la boucle de la ligne 6, on cherche 3 points s, t, u de distance maximale par rapport à une droite, ce qui nous donne une complexité de $O(n * 3n) = O(n^2)$.

1.2.2 Toussaint

L'algorithme de Toussaint se base sur l'idée de *pied à coulisse*, ou plus formellement sur la notion de paires antipodales, utilisé par Shamos dans sa thèse [3] :

Définition 1 *Une paire de points* ***(p,q)*** *d'un polygone convexe, est telle qu'il existent deux droites parallèles passant par* ***p*** *et par* ***q*** *et tous les autres points se trouvent entre ces deux droites.*

Shamos utilise cette notion pour calculer le diamètre de l'enveloppe convexe, i.e. les deux points les plus éloignés, cela en en $O(n)$. Toussaint généralise cette idée : il propose un algorithme qui utilise plusieurs pieds à coulisse afin de résoudre le problème du rectangle minimum :

```
In: List<Points> P - nuage de points.
                    Supposition: pas de doublons consecutifs.
Out: List<Points> R - [A; B; C; D] coins du rectangle
begin
e,w,n,s = points le plus a l'est,
    le plus a l'ouest,
    le plus a nord et le plus au sud
env = envelope convexe de P
R = droite parallele a l'axe des abscisses passant par e
L = droite parallele a l'axe des abscisses passant par w
T = droite parallele a l'axe des ordonnees passant par n
B = droite parallele a l'axe des ordonnees passant par s
U, V, K, M = points d'intersection des droites R, L, T, B
faire
    aE = l'angle entre R et le point suivant de e dans P
    aW = l'angle entre L et le point suivant de w dans P
    aN = l'angle entre T et le point suivant de n dans P
    aS = l'angle entre B et le point suivant de s dans P
    aMin=min(aE,aW,aN,aS)
    point qui fait l'angle minimal devient son suivant dans P
    pivoter les droites U, V, K, M de aMin
    U', V', K', M' = intersection des droites R, L, T, B
    si aire de rectangle [U'; V'; K'; M']<celui de [U; V; K; M]:
        U, V, K, M = U', V', K', M'
tant que indice de e != indice initial de e
retourner [U; V; K; M]
end
```

Cet algorithme calcule le rectangle minimal en $O(n)$, étant donnée une enveloppe convexe qui peut également être calculé en $O(n)$.

1.2.3 Disscussion

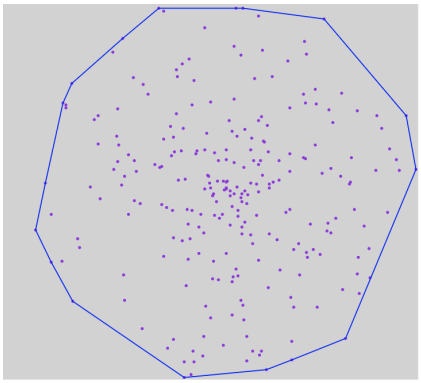
Malgré sa complexité théorique qui résulte en $O(n)$, ainsi que la facilité d'implémentation, l'algorithme de Toussaint utilise la fonction *arctan* [1] afin de calculer l'angle extérieur de chaque côté du polygone qui est une opération très coûteuse ce qui peut augmenter la complexité du calcul effectif jusqu'à $O(n^2)$ [6] Il existent d'autres algorithmes calculant le rectangle minimal en $O(n)$, notamment l'algorithme présenté par Lennert D. Den Boer en 2016 qui utilise la notion de *3P : Projection paramétrique perpendiculaire* [6] qui

1. Ainsi que la fonction racine carés qui emploie un algorithme d'approximation relativement lent

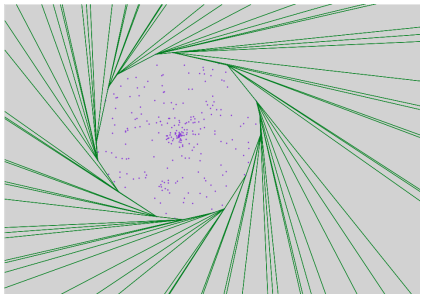
est efficace pour un petit nombre de points et qui tend également vers $O(n^2)$ mais uniquement pour un très grand nombre de points. Un autre algorithme en $O(n)$ utilisé la structure appelé étoile, a été présenté par Toussaint en 1980. [9]

1.3 Tests

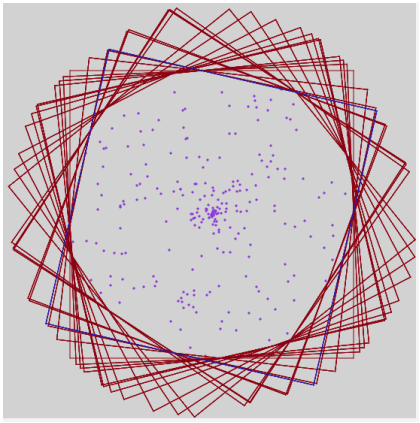
Afin de tester l'algorithme de Toussaint, une interface graphique en Java a été développée, notamment avec JavaFX. Voici quelques résultats :



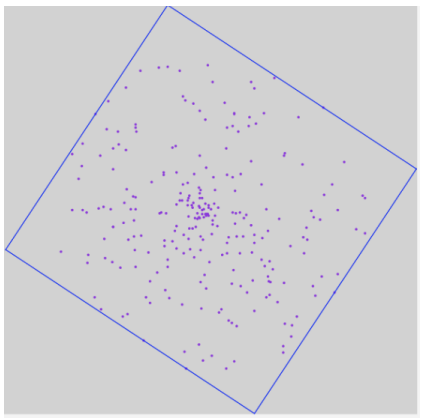
L'enveloppe convexe.



Les paires antipodales.



Les rectangles couvrants.



Le rectangle minimum.

1.3.1 Testbeds

Affin de tester la qualité de l'algorithme suivant la formule :

$$\frac{aire\ de\ rectangle\ min}{aire\ de\ l'env.\ convexe} - 100\%$$

on a utilisé le *testbed* de `Varoumas` : un ensemble de fichiers de points vivifiant :

1. Le nombre de fichiers de teste est supérieur à 1664
2. Le nombre de points dans chaque fichier est au moins 256

1.3.2 Tests de performance

Ayant calculé l'enveloppe convexe [2] et le rectangle minimum de Toussaint pour chaque instance de *testbed* de `Varoumas` on a obtenu les résultats suivants :

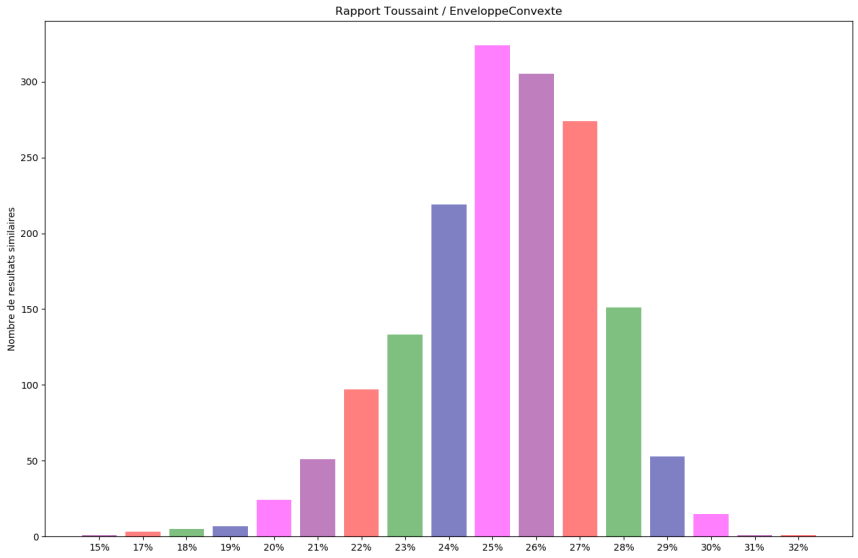


FIGURE 1.3: Statistique de qualité Toussaint.

2. Grâce à l'algorithme de Graham ($O(n\log(n))$ mais $O(n)$ dans notre cas car on a utilisé l'heuristique de Toussaint - le tri pixel en $O(n)$.

1.3.3 Discussion

Ainsi, comme on peut observer sur la figure 1.3 les rectangles minimaux ont un aire supérieur de 25.18% en *moyenne* par rapport aux enveloppes convexes. L'Écart type étant 2.20.

1.4 Conclusion

Malgré la lourdeur de calcul due à l'utilisation des fonction trigonométriques pour un petit nombre de points, l'algorithme de Toussaint se montre relativement bien pour un grand nombre de points.

Chapitre 2

Algorithme de Ritter

2.1 Introduction

Étant donné un ensemble fini de points P dans $\mathbb{R}^2$, on cherche à obtenir un cercle minimum contenant tout point de l'ensemble. Ce problème s'avère très important dans des situations suivantes

1. *Facility location* : trouver le point optimal d'une installation, par exemple un bureau de poste / urgences.
2. Détection de collisions
3. L'application militaire : si on considère que les points ce sont des cibles à détruire, le centre de cercle minimum est l'endroit idéal pour y placer la bombe. Le rayon du cercle sert à déterminer la quantité d'explosif nécessaire.
4. Le problème corollaire de sphère minimum est utilisé dans le partitionnement de données.

2.2 Analyse des algorithmes existants

2.2.1 Algorithme en $O(n^2)$

1. Faire un cercle C centre en c, suffisamment large pour contenir tous les points
2. Réduire le cercle en trouvant le point A le plus éloigné et en faisant passer le cercle par ce point.

3. Si le cercle passe par au moins 2 points passer à l'étape suivant. Sinon réduire le cercle en avançant le centre vers A.
4. Si le cercle contient un intervalle d'arc sans point qui est plus grand que la moitié du tour complet, le cercle peut être réduit. Soit D et E deux points aux extrémités de cet intervalle. En gardant D et E sur le cercle, réduire le diamètre de cercle jusqu'à se trouver dans un des cas :
   — Si le diamètre est la longueur |DE| on s'arrête
   — Sinon, le cercle touche un autre point, F. Vérifier s'il existe toujours un arc plus grand que la moitié du tour complet n'ayant pas de points.
      — si il y en a pas - on a fini.
      — sinon, revenir en point 4, répéter avec les points extérieurs de l'arc.

Les étapes 1,2,3 s'exécutent en temps linéaire, mais dans le 4. on cherche un nouveau point F, ce qui peut se reproduire n-2 fois. D'où la complexité totale $O(n^2)$. Cet algorithme a été proposé par Elzinga and Hearn [7] en 1972.

2.2.2 Algorithme d'approximation de Ritter

Cet algorithme permet de trouver un cercle *"quasi-optimal"* en $O(n)$. Suivant J. Ritter[2] lui même les cercles obtenus sont environ 5% plus grands que les cercles optimaux. Pour trouver l'approximation de cercle minimum on procède ainsi :

1. Prendre au hasard un point **R** parmi tous les points.
2. Trouver un point **P** le plus éloigné de R.
3. Trouver un point **Q** le plus éloigné de P
4. Faire un cercle **C**, centré au milieu de segment **PQ** et de rayon $\frac{|PQ|}{2}$.
5. Retirer **P** et **Q** de l'ensemble des points.
6. Tant qu'il reste encore des points, considérer un point **S** quelconque.
7. Si **S** est dans le cercle centré en **C** retirer **S** de l'ensemble de points. Répéter le 6.
8. Sinon, considérer la droite **SC**, elle couple le cercle en deux points, soit **T** le point le plus éloigné de **S**.
9. Soit **C'** centre de segment **ST**.

10. Remplacer le cercle en cours par cercle centré en **C'**, avec rayon |**C'T**|
11. Répéter l'étape 6, jusqu'à ce qu'il n'y ait plus de points.

2.2.3 D'autres algorithmes

En 1983 Nimrod Megiddo, a démontré qu'il existe un algorithme de cercle minimum qui s'exécute en $O(n)$ [8], il utilise la technique de "élaguer-et-chercher" [1]. Dans cet algorithme un travail linéaire est fait afin de réduire la taille de l'entrée par une quantité f. Cela permet de réduire le temps d'exécution total à $O(n)*(1+(1-f)+(1-f)2+...)$, cette suite géométrique s'additionne en un nombre constant ce qui donne un temps total $O(n)$. Ainsi, l'algorithme de Megiddo enlève au moins $\frac{n}{16}$ points, ce qui donne un temps d'exécution proportionnel à $(n + \frac{15n}{16} + \frac{225n}{256} + ...) = 16n$

2.3 Tests

Après avoir implémenté l'algorithme de Ritter dans notre application JavaFX, on a obtenu les résultats suivants :

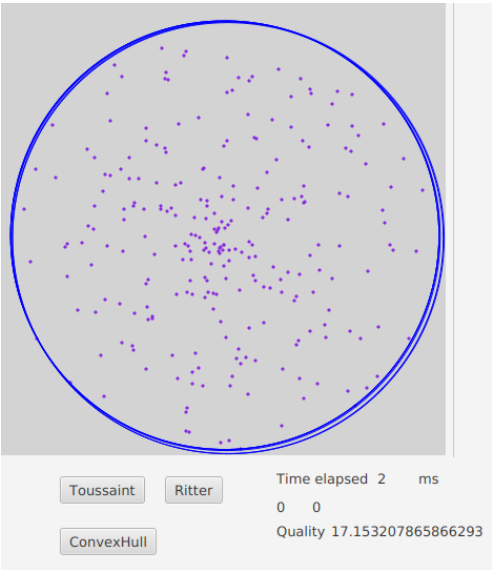


FIGURE 2.1: Plusieurs instances d'approximation de cercle minimal.

1. angl. Prune-and-serach

2.3.1 Tests de performance

Voici les résultats du même procédé qu'en Tests de performance, cette fois-ci pour le cercle minimum.

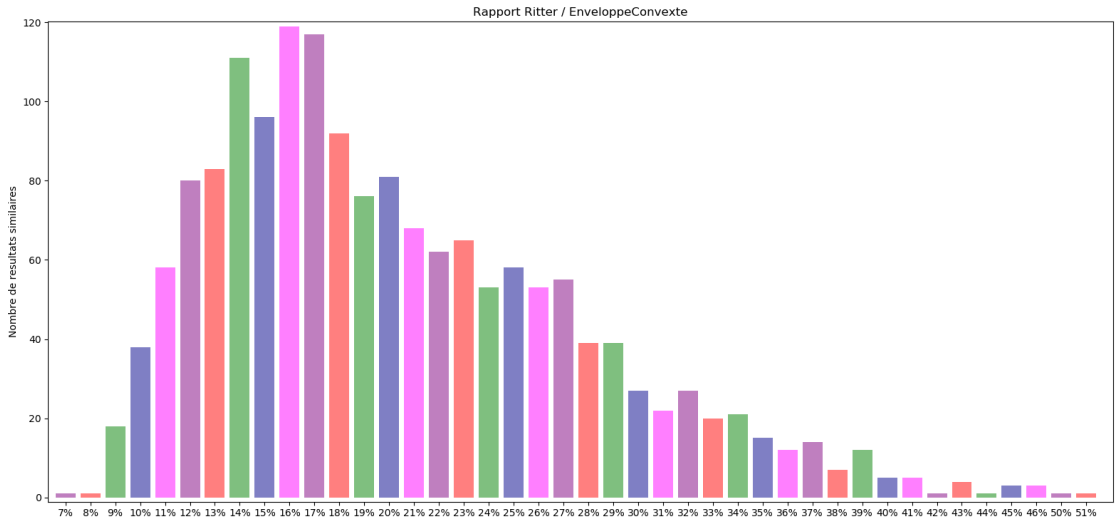


FIGURE 2.2: Statistique de qualité Toussaint.

2.3.2 Discussion

La figure 2.2 nous montre que les cercles trouvés par l'algorithme de Ritter sont en moyenne 20.44% plus grands que l'enveloppe convexe, ce qui est 4.74% mieux que les résultats des `RectangleMin` de Toussaint. Néanmoins on note également que les résultats de Ritter ont un écart type de 7.43 ce qui montre qu'il peut y avoir plus de cas extrêmes dans le cas d'algorithme de Ritter par rapport à celui de Toussaint, dont les airs de rectangles sont moins dispersés par rapport à la moyenne.

Chapitre 3

Conclusion

Quand il s'agit de trouver le conteneur minimal, les algorithmes Ritter et Toussaint donnent des résultats très similaires, l'approximation de Ritter étant légèrement meilleure que le rectangle minimum. En matière de complexité, les deux algorithmes ont un temps d'exécution total $O(n)$. Pourtant, comme on a mentionné en Algorithme d'approximation de Ritter, le cercle trouvé n'est pas le minium exact. Cela étant dit, il existe des algorithmes qui trouvent le cercle min en $O(n)$, notamment celui de Megiddo[8] il en est de même pour l'algorithme de rectangle minimum qui peut être résolu en utilisant la structure appelée *étoile* [9], qui est également en $O(n)$. La différence entre les deux algorithmes qu'on vient de citer et ceux de Ritter et Toussaint c'est leur implémentation relativement facile. Donc pour le rectangle minimum, Toussaint est une solution efficace et pratique. Quant à Ritter il est facile à implémenter et est satisfaisant pour un grand nombre de cas, mais uniquement quand on peut accepter un résultat approximatif.

Bibliographie

[1] Toussaint, G. T (1983). *"Solving geometric problems with the rotating calipers"*.
Proceedings of MELECON '83, Athens.

[2] Ritter, Jack (1990). *"An efficient bounding sphere"*, in Glassner, Andrew S. (ed.), Graphics Gems, San Diego, CA, US : Academic Press Professional, Inc., pp. 301–303, ISBN 0-12-286166-3

[3] M.I. Shamos, *"Computational geometry"*, Ph.D. thesis, Yale University, 1978.

[4] H. Freeman and R. Shapira. Determining the minimum-area encasing rectangle for an arbitrary closed curve. In Communications of the ACM, volume 18, pages 409–413, New York, NY, July 1975.

[5] Bùi Xuân Bình Minh (2019)
https://www-apr.lip6.fr/~buixuan/files/algav2019/algav2019_cours7_notes.pdf

[6] Lennert D. Den Boer (June 2016) *A Fast Algorithm for Generating a Minimal Bounding Rectangle* Fracture Network Modeling

[7] Elzinga, J. ; Hearn, D. W. (1972), *"Geometrical solutions for some minimax location problems"*, Transportation Science, 6 (4) : 379–394,

[8] Megiddo, Nimrod (1983), *"Linear-time algorithms for linear programming in R3 and related problems"*, SIAM Journal on Computing, 12 (4) : 759–776, doi :10.1137/0212052, MR 0721011.

[9] G.T. Toussaint, "Pattern recognition and geometrical complexity", Proc. Fifth Interna- tional Conference on Pattern Recognition, Miami Beach, December 1980, pp. 1324- 1347.