L'algorithme Welzl résolvant le problème du cercle minimum

Conception et Pratique de l'Algorithmique

Rédigé par :

ZHANG Zimeng

Année universitaire 2021/2022

TABLE DES MATIÈRES

Abstract

Cet article présente plusieurs algorithmes différents tels que l'algorithme naif, l'algorithme Welzl, etc. pour résoudre le problème du cercle minimal, comparera les performances des deux algorithmes, et analysera les raisons de la différence de performances et les points à améliorer sous plusieurs angles différents

Key words — cercle minimum; algorithme Welzl; computational geometry; analysis of algorithms

1 Introduction

Le problème du cercle minimum a été initialement proposé par le mathématicien anglais James Joseph Sylvester en 1857, il est une illustration du problème d'emplacement d'installations dans lequel l'emplacement d'une nouvelle installation doit être choisi pour desservir plusieurs clients, en minimisant la distance la plus éloignée que tout client doit parcourir depuis la nouvelle installation. Il existe de nombreux domaines de notre vie où ce problème a une application pratique, par exemple dans le domaine des soins de santé où l'inaccessibilité des installations médicales peut entraîner une morbidité et une mortalité accrues. Un autre exemple est le problème de l'élimination des déchets dans les villes, où le fait de ne pas localiser les sites d'enfouissement au bon endroit pour toucher davantage de personnes peut entraîner une augmentation de la production de déchets et des coûts de gestion des déchets.

Cet article présentera d'abord brièvement le concept de cercle minimal et les théorèmes mathématiques pertinents, puis présentera et comparera deux algorithmes pour résoudre le problème du cercle minimal, analysera les résultats et les performances des deux algorithmes, et conclura par un résumé et des points à améliorer.

Problème du cercle minimum

Le recouvrement minimal de cercle est un problème algorithmique en mathématiques qui étudie comment trouver le plus petit cercle qui peut recouvrir un groupe de points dans le plan. Une généralisation de ce problème aux espaces à n dimensions en général est le problème des sphères minimales englobantes, c'est-à-dire trouver la plus petite sphère qui peut couvrir un certain ensemble de points dans un espace à n dimensions. [2]

La recherche du plus petit cercle nécessite les deux faits de base suivants:

1) Le cercle de couverture minimale est unique.

2) Étant donné un ensemble de points, il y a au plus trois points sur le cercle de couverture minimale. Si trois points se trouvent sur le cercle de couverture minimale, alors le cercle de couverture minimale est le cercle extérieur de ces trois points ; si seulement deux points se trouvent sur le cercle de couverture minimale, alors le cercle de couverture minimale est le cercle ayant pour diamètre le segment de droite entre ces deux points.

Concepts mathématiques

Comme il est nécessaire de calculer un cercle composé de trois points dans l'algorithme, certains concepts pertinents seront brièvement présentés ici.

1. Pour déterminer si un point est contenu dans un cercle, il suffit de calculer la distance entre le centre du cercle et le point et de la comparer au rayon du cercle. Si la distance est inférieure au rayon, le point est à l'intérieur du cercle.

2. Tout d'abord, pour que trois points sont cocycliques, il faut que ces trois points ne sont pas conlinéaires, et nous utiliserons le produit vectoriel pour faire cette détermination. Supposons qu'il y ait trois points A, B et C qui peuvent être représentés par le vecteur $\vec{AB}$ *et* $\vec{AC}$. Si $\vec{AB}$ * $\vec{AC}$!= 0, alors nous savons que les trois points A,B,C ne sont pas colinéaires.

3. Ayant déterminé que les trois points ne sont pas colinéaires, nous devons calculer le cercle formé par les trois points. Nous définissons d'abord le centre du cercle(x_0, y_0), la rayon

r, puis le cercle est$(x - x_0)^2 + (y - y_0)^2 = r^2$. Les coordonnées des trois points sont (x_1, y_1),(x_2, y_2) et (x_3, y_3).Le cercle formé par ces trois points est

$$\begin{cases} (x_1 - x_0)^2 + (y_1 - y_0)^2 = r^2 \\ (x_2 - x_0)^2 + (y_3 - y_0)^2 = r^2 \\ (x_3 - x_0)^2 + (y_3 - y_0)^2 = r^2 \end{cases}$$

Après simplification, nous obtenons les coordonnées du centre du cercle et du rayon.

2 Algorithme

2.1 Agorithme Naïf

2.1.1 Principe

Cet algorithme est la méthode la plus simple mais aussi la moins efficace pour résoudre le problème du cercle minimum.

Tout d'abord, l'ensemble des points est parcouru deux par deux pour produire un cercle dont le diamètre est la distance entre les deux points et sur lequel les deux points sont situés, puis on détermine si tous les points de l'ensemble sont à l'intérieur du cercle. Si un cercle contenant tous les points peut être trouvé dans cette étape, le calcul s'arrête.

Si un cercle minimum contenant tous les points n'est pas trouvé à partir du processus précédent, nous continuons la recherche. Dans cette étape, nous parcourons l'ensemble des points pour trouver trois points et nous les parcourons en boucle pour déterminer si le cercle composé de ces trois points contient tous les points. Dans le pire des cas, la complexité de cet algorithme est de O(n^4).

2.1.2 Pseudo Code

Algorithm 1 Naïf

Données : ArrayList<Point> $InputPoints$

Résultat : Circle

Circle $circle$;

pour *Points p in* $InputPoints$ **faire**

  pour *Points q in* $InputPoints$ **faire**

    Point $centre$ ←`MiddleOfLine(`*p,q*`)`;

    Double $rayon$ ←`distance(`*p,centre*`)`;

    $circle$ ← `Circle(`*centre,rayon*`)`;

    si $circle$ *contains* $InputPoints$ **alors**

      retourner $circle$;

    fin

  fin

fin

pour *Points p in* $InputPoints$ **faire**

  pour *Points q in* $InputPoints$ **faire**

    pour *Points r in* $InputPoints$ **faire**

      si `CrossProduct(`*p,q,r*`)` *== 0* **alors**

        continue;

      fin

      Circle $temp$ ← `Circonscrit(`*p,q,r*`)`;

      si $temp$ *contains inputPoints AND* $temp$ *plus petit que* $circle$ **alors**

        $circle$ ← $temp$;

      fin

    fin

  fin

fin

retourner $circle$;

2.2 Algorithme Welzl

2.2.1 Principe

Saisissez d'abord deux ensembles de points P et Q, P contenant tous les points et Q une liste vide. Si P est vide ou s'il y a trois points dans Q, nous créons un cercle constitué des trois points dans Q (les trois points déterminent le cercle unique), si cette condition n'est pas remplie, nous prenons

récursivement un point à la fois dans P jusqu'à ce que P soit vide et déterminons à chaque fois si le point appartient au cercle, s'il ne l'appartient pas, nous l'ajoutons à l'ensemble R. [3]

2.2.2 Pseudo Code

Algorithm 2 Welzl

Données : ArrayList<Point> $pointsListP$

Une liste contient tous les points **Données :** ArrayList<Point> $pointsListR$

Une liste vide

Résultat : Circle

Circle $circle$;

si $pointsListP$ *is empty OR* `Size(`$pointsListR$`)` == *3* **alors**
  retourner $circle \leftarrow$ `trivial(`$pointsListR$`);`

sinon
  Prendre un point aléatoire dans la liste et le retirer de la liste
  Point $temp \leftarrow$ `random(`$pointsListP$`);`
  $circle \leftarrow$ `welzl(`$pointsListP$,$pointsListR$`);`
  si `isNotIn(`$circle$,$temp$`)` **alors**
    add $temp$ to $pointsListR$;
    $circle \leftarrow$ `welzl(`$pointsListP$,$pointsListR$`);`
  fin

fin

retourner $circle$;

3 Programmation

Nous utilisons Java pour implémenter les deux algorithmes et visualisons les résultats pour comparer les performances des deux algorithmes.

3.1 Le grahpe

D'après les deux figures ci-dessous, il est non seulement clair que le cercle minimal résultant peut contenir tous les points, mais aussi que le cercle minimal calculé par les deux algorithmes se chevauche lorsqu'on passe d'une vue à l'autre, ce qui prouve que le cercle minimal est unique.

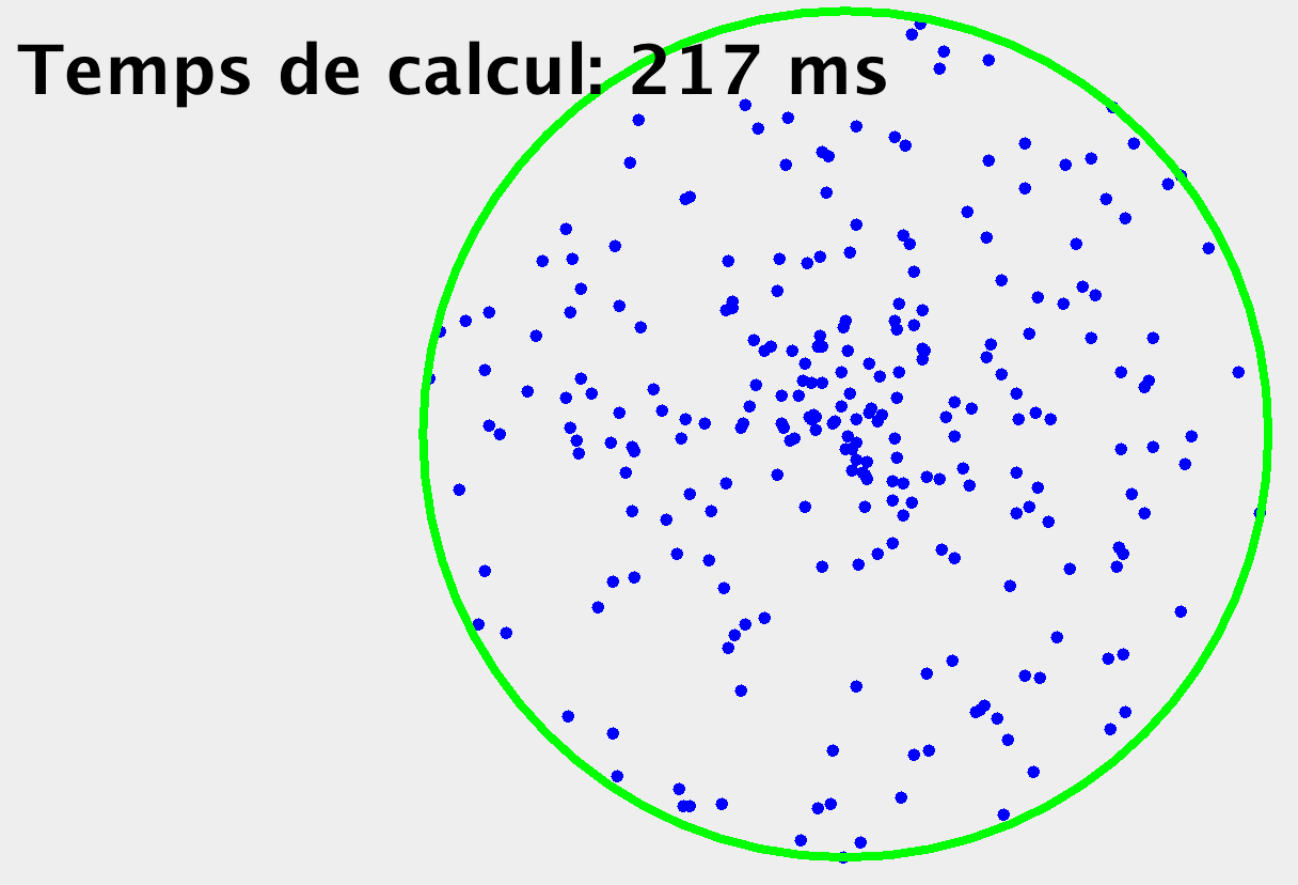

Figure 1: Cercle d'algorithme naïf

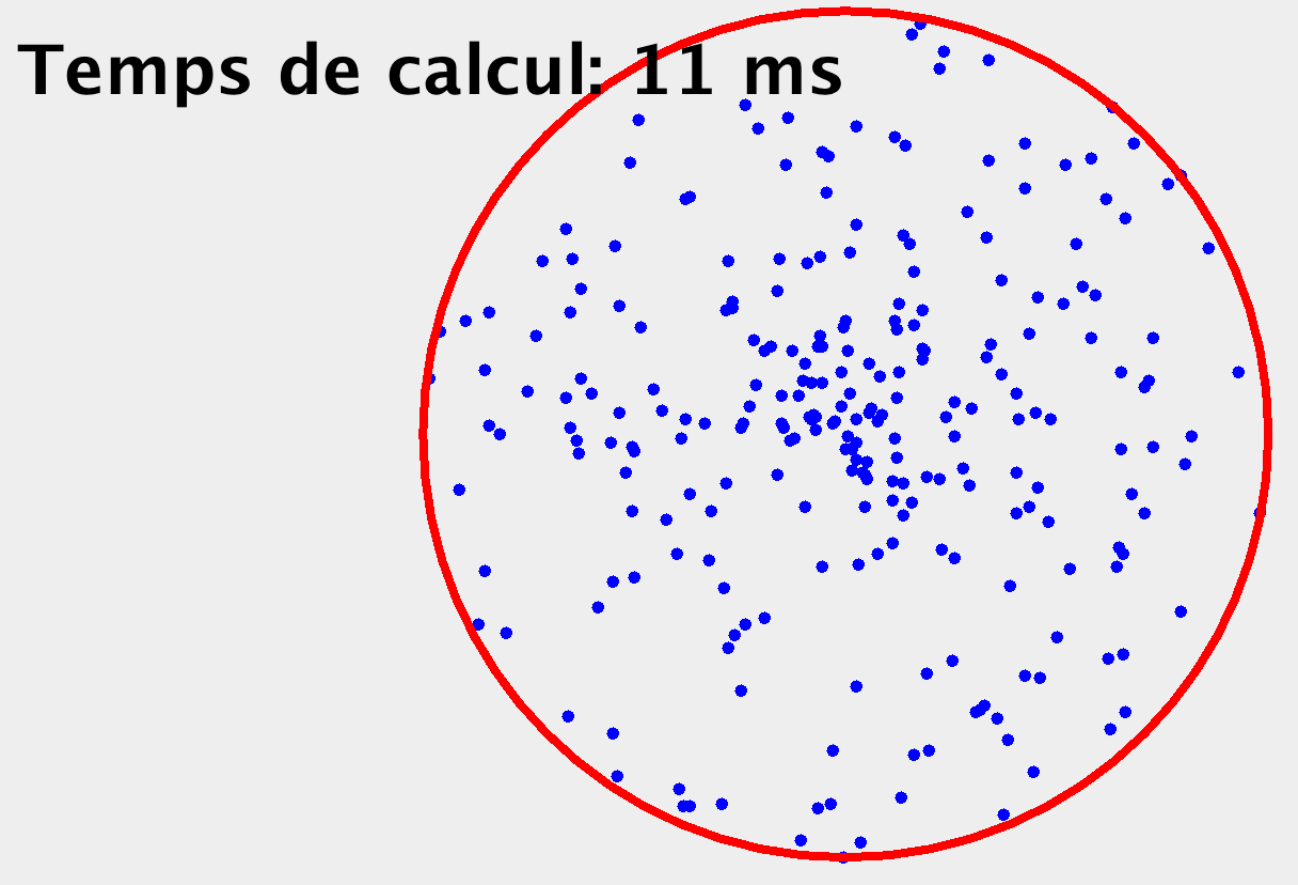

Figure 2: Cercle d'algorithme Welzl

4 Experimental results

Les fichiers source pour nos expérimentations proviens de VAROUMAS [1]. Après avoir effectué 1663 tests de taille 256, nous avons calculé le temps moyen et l'écart type des deux algorithmes. Nous pouvons voir sur le graphique que l'algorithme de Welzl est plus performant que celui de Naif, Welzl prenant 0,62 ms pour calculer contre 71,71 ms pour Naif, et d'après les résultats de l'écart type Les performances de Welzl sont également plus stables. Nous avons écrit les résultats dans un fichier à l'aide de FileWriter, puis nous les avons importés dans Excel pour

établir les tableaux et les diagrammes.

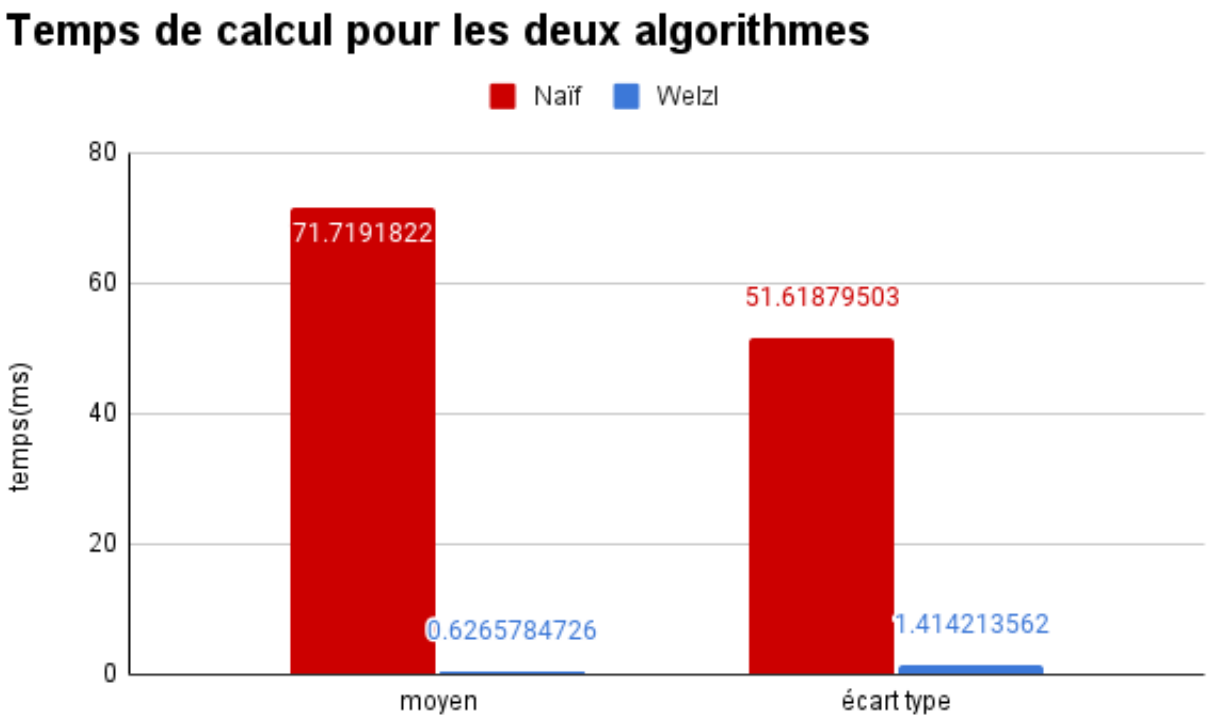

Figure 3: Diagrammes de temps moyenne et écart type de deux algorithemes

5 Discussion

Nous sommes tous satisfaits des résultats de l'expérience. Le calcul théorique de l'algorithme welzl est moins complexe que celui de l'algorithme naif, les résultats confirment donc pleinement cette déduction, mais il y a encore quelques domaines où des améliorations pourraient être apportées.

Par exemple, lors de l'implémentation de l'algorithme en java, nous avons utilisé une liste de tableaux pour stocker les points du graphe. Pour améliorer l'algorithme, nous aurions peut-être pu utiliser un hashset pour le stockage des données, mais malheureusement, en raison des contraintes

de temps, nous n'avons pas tenté cela. De plus, nos expériences ont été réalisées sur une base bidimensionnelle et il aurait été plus intéressant d'explorer l'application de l'algorithme de welzl à des graphes tridimensionnels.
En ce qui concerne les résultats de cercle minimum générés par les deux algorithmes pour le même fichier d'ensembles de points, nous constatons qu'il existe des cas où les deux cercles ne coïncident pas exactement. Puisque nous savons que le cercle minimum calculé par l'algorithme naïf est unique et précis, nous pouvons constater que le cercle calculé par l'algorithme de welzl n'est pas toujours précis, mais les résultats expérimentaux montrent que dans la plupart des cas, les deux cercles coïncident exactement.

6 Conclusion

Théoriquement, la complexité de l'algorithme naif dans le pire cas est $O(n^4)$ et la complexité de l'algorithme Welzl est $O(n)$. Les expérimentations ci-dessus montrent également que les performances de Welzl sont bien meilleures que celles de Naif. Cependant, les fonctions récursives utilisées par Welzl font que le processus de calcul nécessite un grand nombre d'appels de fonction, avec parfois des problèmes de stack overflow, il faut donc aussi éviter d'utiliser Welzl pour des fichiers trop volumineux. Mais malgré cela, l'algorithme de Welzl est l'un des algorithmes les plus efficaces pour traiter le problème du cercle minimum.

References

[1] Varoumas.

[2] Donald W. Hearn D. Jack Elzinga. The minimum covering sphere problem. 1972.

[3] Emo Welzl. Smallest enclosing disks (balls and ellipsoids). 1991.