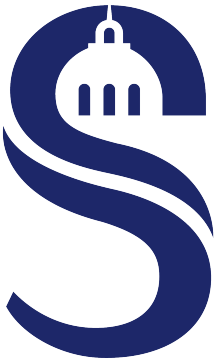

SORBONNE UNIVERSITÉ

Master Informatique

Rapport de recherche
L'algorithme Welzl resolvant le problème du cercle minimum

Adan Bougherara

Remis le 20 03 2022

Calculer le cercle minimum d'un ensemble de points

Résumé : Etant donné un ensemble fini de points, déterminer le cercle minimum à l'ensemble P consiste a trouver le cercle de plus petit rayon couvrant tous les points de P. L'algorithme de Welzl semble être une option intéressante pour ce problème. Il s'agit d'analyser la performance en temps de calcul de l'algorithme Welzl sous son aspect récursif comme itératif en le comparant à un algorithme naïf.

Mots-clés : Cercle minimum, algorithme Welzl, dérécursivation, performance

1 Introduction

Le problème du cercle couvrant minimum est un problème qui est notamment lié à la détection de collision lors de modélisations ou de jeux vidéos en 2D.
Une approche consiste à représenter un objet par un ensemble de point et de le modéliser alors par un cercle couvrant de taille minimum afin de rendre la détection de collision plus simple par la suite. C'est alors que le problème du cercle couvrant minimum entre en jeu.
Nous allons dans ce rapport expliciter puis comparer deux algorithmes permettant de trouver un cercle couvrant minimum d'un ensemble de points dans un plan 2D. Le premier est un algorithme dit naïf et le second est l'algorithme de Welzl, que nous allons voir sous sa forme récursive puis sous sa forme itérative pour éviter tout dépassement de pile.

2 Notations et définitions

2.1 Structures utilisées

Un point est représenté par la classe java.awt.Point, il est donc défini par des coordonnées entières x et y.

Un cercle est représenté par la classe src.structure.Circle, il est donc défini par un point qui est son centre et par un entier naturel qui est son rayon.

2.2 Notations

On note $P(x, y)$ les coordonnées d'un point du plan avec $(x, y) \in \mathbb{N}^2$
On note $C(p; r)$ avec p un point du plan et $r \in \mathbb{N}$ le cercle de centre p et de rayon r.
Soit c un cercle, on note son centre $c.centre$ et son rayon $c.rayon$
On note P l'ensemble des points à couvrir et $n \in \mathbb{N}$ sa taille.
On note $\|AB\|$ la distance euclidienne entre les points A et B

2.3 Définitions

La distance euclidienne : La distance entre deux points $X_1 = P(x_1, y_1)$ et $X_2 = P(x_2, y_2)$ est notée $\|X_1X_2\|$ et vaut :

$$\|X_1X_2\| = \sqrt{(x_2 - y_2)^2 + (y_2 - y_1)^2}$$

Cercle circonscrit : Soient A, B et C trois points. Le cercle circonscrit est le cercle qui passe par tous les sommets du triangle ABC.

Lemme 1 : Si un cercle de diamètre égal à la distance de deux points de la liste couvre tout autre point de la liste, alors ce cercle est un cercle couvrant de rayon minimum

Lemme 2 : En 2D, il existe un unique cercle passant par 3 points non-colinéaires.

Lemme 3 : Le cercle circonscrit à un triangle ABC non plat est le cercle couvrant minimum de l'ensemble $\{A, B, C\}$. *(Immédiat en utilisant le Lemme 2)*

3 L'algorithme naïf

3.1 Principe

Cet algorithme permet d'approcher le problème du cercle couvrant minimum d'une première façon. Il peut être découpé en deux phases.

Premièrement, il s'agit de parcourir tous les points de P deux à deux, notés respectivement $A = (x_A, y_A)$ et $B = (x_B, y_B)$, et de construire leur cercle minimum associé $c = C(\ P(\frac{x_A + x_B}{2},\ \frac{y_A + y_B}{2})\ ;\ \frac{\|AB\|}{2})$.
Si c couvre tous les points de P, d'après le *Lemme 1*, c est alors le cercle couvrant de P de rayon minimum.

Si le cercle minimum n'a pas été trouvé lors de la première phase, on passe à la deuxième.
La deuxième étape consiste à parcourir tous les points de P trois à trois, que l'on note respectivement A, B et C, et à construire le cercle circonscrit c associé si les points ne sont pas colinéaires.
D'après le *Lemme 3*, c est le cercle couvrant minimum de $\{A,\ B,\ C\}$. Par conséquent, si c couvre également tous les autres points de P, alors c est le cercle couvrant minimum de P.

3.2 Exemple d'une solution trouvée avec un cercle circonscrit

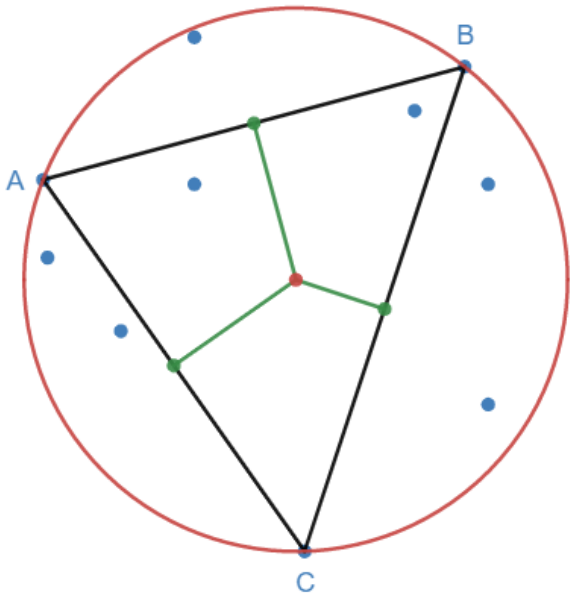

FIGURE 1 – Cercle circonscrit de ABC couvrant tout P

3.3 Pseudo-code

Algorithm 1: Algorithme naïf

```
Data: Une liste de points Points
Result: Le cercle couvrant minimum de Points
forall p dans Points do
    forall q dans Points do
        c ← C( ((x_p+x_q)/2, (y_p+y_q)/2), ||PQ||/2)
        if c couvre Points then
            return c
        end
    end
end
res ← cercle de rayon infini
forall p dans Points do
    forall q dans Points do
        forall r dans Points do
            c ← cercle circonscrit du triangle PQR
            if c couvre Points et que c.rayon < res.rayon then
                res ← c
            end
        end
    end
end
return res
```

3.4 Complexité

Nous analysons ici la complexité en temps dans le pire cas. La première phase comprend un double parcours des points de P pour construire les cercles ainsi qu'un parcourt des points pour vérifier s'ils sont dans le cercle. Nous avons donc 3 parcours des points de P imbriqués, ce qui donne une complexité au pire cas en $\mathcal{O}(n^3)$.

La seconde phase comprend un triple parcours des points de P pour construire les cercles circonscrits ainsi qu'un parcourt des points de P pour vérifier s'ils sont dans le cercle. Nous avons donc 4 parcours des points de P imbriqués, ce qui donne une complexité au pire cas en $\mathcal{O}(n^4)$.

Ainsi la complexité en temps de l'algorithme naïf est au pire cas vaut $\mathcal{O}(n^3) + \mathcal{O}(n^4) = \mathcal{O}(n^4)$

4 L'algorithme Welzl récursif

4.1 Principe

L'algorithme de Welzl est un algorithme récursif incrémental qui résout le problème du cercle couvrant minimum en construisant des cercles minimum de sous ensembles de P jusqu'à couvrir tous les points de P. L'algorithme utilise P ainsi qu'un ensemble R représentant les points à la frontière du cercle. A chaque itération, l'algorithme sélectionne un point p dans P de manière aléatoire et uniforme, et trouve récursivement le cercle minimal c contenant les points de P $-$ {p} (P décroît donc à chaque itération hors cas d'arrêt). Deux cas surviennent alors :

c couvre p : Le cercle n'a pas besoin d'être agrandi. On retourne donc c

c ne couvre pas p : On recalcule le cercle par un appel récursif en ajoutant p à R

L'algorithme de Welzl possède deux cas d'arrêt :

Si |P| = 0 :
Trois comportements possibles selon la valeur de |R| :

Si |R| = 0 ou |R| = 1 :
On construit c avec c.rayon = 0

Si |R| = 2 :
On construit c avec les deux points de R avec c.centre le milieu du segment des 2 points et c.rayon la moitié de la distance entre les deux points

Si |R| = 3 :
On construit c comme étant le cercle circonscrit des 3 points de R.

Si |R| = 3 :
On recalcule le cercle par un appel récursif en ajoutant p à R

4.2 Pseudo-code

Algorithm 2: Algorithme récursif Welzl

```
1  Function minidisk(P) :
2  |  return b_minidisk (P, ∅)
3
4  Function b_minidisk(P, R) :
5  |  if P = ∅ ou |R| = 3 then
6  |  |  return circle(R, |R|)
7  |  end
8  |  on prend p ∈ P aléatoirement
9  |  res ← b_minidisk(P − {p}, R)
10 |  if p n'est pas dans res then
11 |  |  R ← R + {p}
12 |  |  res ← b_minidisk(P, R)
13 |  |  R ← R − {p}
14 |  |  P ← P + {p}
15 |  end
16 |  return res
17 return
```

4.3 Complexité

Nous analysons ici la complexité en temps dans le pire cas. A chaque appel à **b_minidisk** hors cas d'arrêt, P perd un élément. Dans le pire des cas, c'est le cas d'arrêt P = ∅ qui sera déclenché une fois P vide. De plus, deux appels récursifs sont effectués. Ainsi, la complexité en temps dans le pire cas s'évalue en $2 * \mathcal{O}(n) = \mathcal{O}(n)$ L'algorithme de Welzl permet donc de résoudre le problème du cercle minimum en un temps linéaire.

5 L'algorithme Welzl itératif

5.1 Principe

L'algorithme de Welzl sous forme itérative reprend le fonctionnement de la forme récursive. L'objectif est de remplacer les appels récursifs par des itérations au sein d'un même environnement de fonction et de par conséquent n'avoir qu'un appel de fonction pour alléger grandement la pile. Pour ce faire, on utilise un tableau de booléen qui remplace le rôle de la pile (implanté par un BitSet dans le code

source). Le tableau stocke pour chaque "appel" un booléen indiquant si le point p est dans le cercle res ou pas. On dispose également d'un booléen calling qui indique si on fait le premier appel récursif. De plus, on dispose de deux compteurs point_cpt et r_count qui permettent respectivement de se situer dans les itérations et de connaître l'état de la frontière du cercle.

5.2 Pseudo-code

Algorithm 3: Algorithme itératif Welzl

```
1  Function iterative_welzl(P) :
2      P ← shuffle(P)
3      point_cpt ← 0
4      r_count ← 0
5      r ← tab [3]                    /*Tableau de points*/
6      second_call ← tab [ |P| ] /*Tableau de booléen*/
7      calling ← TRUE
8      res ← NULL
9      do
10         if calling then
11             if point_cpt = 0 || r_count = 3 then
12                 res ← circle(r, r_count)
13                 point_cpt ← point_cpt + 1
14                 calling ← FALSE
15             else
16                 point_cpt ← point_cpt − 1
17                 calling ← TRUE
18             end
19         else
20             if NOT second_call [point_cpt] then
21                 p ← P [point_cpt − 1]
22                 if p est dans res then
23                     point_cpt ← point_cpt + 1
24                     calling ← FALSE
25                 else
26                     r_count ← r_count + 1
27                     r[r_count] ← p
28                     point_cpt ← point_cpt − 1
29                     second_call[point_cpt] = TRUE
30                 end
31             else
32                 second_call [point_cpt] = FALSE
33                 /*On remet le point en tête de P*/
34                 P ← {P [point_cpt − 1]} + ( P − {P [point_cpt - 1]} )
35                 r_count ← r_count − 1
36                 point_cpt ← point_cpt + 1
37                 calling ← FALSE
38             end
39         end
40     while  point_cpt ≤ |P|
```

5.3 Complexité

La dérécursification n'a pas changé la complexité de l'algorithme de Welzl. Donc l'algorithme de Welzl itératif a une complexité en temps dans le pire cas qui s'évalue en $\mathcal{O}(n)$

6 Comparaison des performances

Les tests ont été réalisés à partir de la base de test de Varoumas. Cette base de tests comporte 1664 tests contenant chacun 256 points.
Il faut noter que puisque tous les tests n'ont que 256 points, cette base de tests ne permet pas de montrer qu'en réalité, la pile peut saturer assez vite lors de l'utilisation de l'algorithme de Welzl récursif.

6.1 Correction de l'algorithme Welzl

Sur les 1664 tests effectués, l'algorithme de Welzl récursif a trouvé un résultat identique à l'algorithme naïf 1653 fois avec une marge de $\pm$ 5 pour la taille du rayon.

Avec ces mêmes paramètres, Welzl itératif a 1654 tests ayant le même résultat que l'algorithme naïf.

6.2 Performance en temps

La *figure 2* ci-dessous montre que l'algorithme naïf est environ deux fois plus lent que l'algorithme de Welzl. Par ailleurs les résultats entre l'algorithme récursif et itératif de Welzl sont très similaire.
Cela se justifie par le fait que l'algorithme de Welzl a une complexité linéaire $\mathcal{O}(n)$ contrairement à l'algorithme naïf qui est en $\mathcal{O}(n^4)$.

On peut également noter qu'avec l'algorithme naïf, les temps d'exécutions des tests sont plus dispersés qu'avec Welzl. Cela s'explique par le fait qu'avec l'algorithme naïf il peut y avoir une grande différence de temps entre un test ou la première double boucle suffit à déterminer le cercle minimum et entre un autre test ou il faut parcourir intégralement la première double boucle puis la triple.
Concernant l'algorithme de Welzl, mélanger la liste des points au départ permet d'ajouter une part de hasard lors du choix de p à chaque itération ce qui a pour effet de minimiser l'écart type sur de grands échantillons.

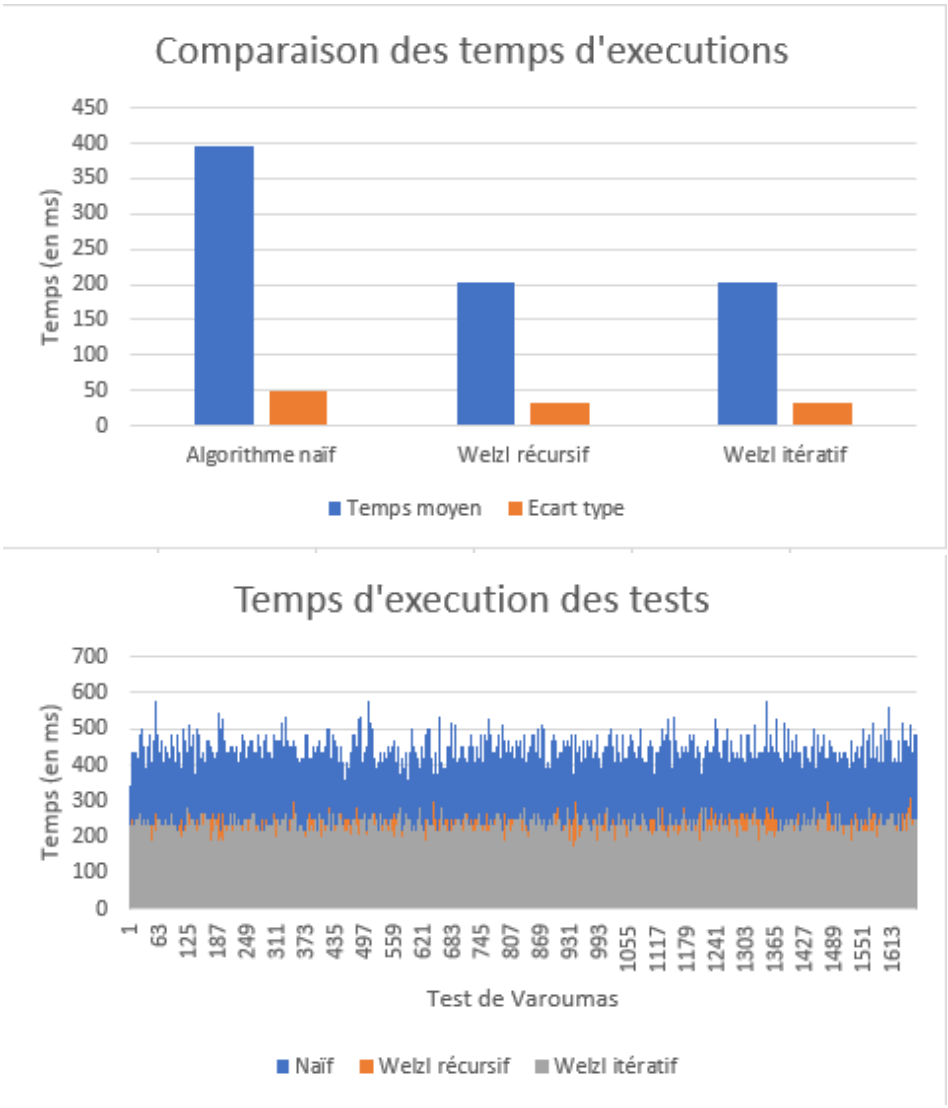

FIGURE 2 – Diagramme en bâton et diagramme de fréquence

7 Conclusion

Pour conclure, l'algorithme naïf est une solution simple qui permet de résoudre le problème du cercle minimum. Bien que sa complexité soit très insatisfaisante et qu'on peut le qualifier d'algorithme très lent, il nous permet dans ce rapport d'avoir un point de comparaison pour l'algorithme de Welzl. L'algorithme de Welzl récursif permet quant à lui de résoudre le problème du cercle minimum en temps linéaire en obtenant un résultat similaire à l'algorithme naïf dans la grande majorité du temps.
Toutefois, dès que la taille de l'ensemble des points à couvrir devient trop importante, il y a un dépassement de pile et l'algorithme ne fonctionne plus.
L'algorithme de Welzl sous sa forme dérécursifiée résout cette contrainte, puisqu'un seul appel est effectué, tout en préservant des performances en temps et en correction par rapport à l'algorithme naïf similaire à sa forme récursive.

Ainsi, si l'objectif est de trouver à coup sûr le cercle couvrant minimum d'un ensemble de point, l'algorithme naïf semble être la meilleure option. Si on peut se permettre une petite marge d'erreur, l'algorithme Welzl itératif est sans hésiter le meilleur choix.