Projet de Conception Pratique Algorithmique 2016

Tomohiro ISHIWATA

25 février 2016

Calculer le cercle couvrant minimum d'un ensemble de points

Résumé : Etant donné un ensemble fini $\mathcal{P}$ de points de $\mathbb{R}^d$, le cercle couvrant minimum de $\mathcal{P}$ est le cercle minimum qui contient tous les points de $\mathcal{P}$.
On propose deux algorithmes pour calculer le cercle minimum d'un ensemble fini de points.
Le premier est un algorithme naïf, qui s'avère efficace dans le seul cas où on trouve le cercle minimum avec 2 points, mais extrêmement lent dès lors où il faut calculer le cercle circonscrit entre 3 points distincts.
Le deuxième est l'algorithme de Welzl, qui possède l'énorme avantage d'avoir une complexité linéaire dans tous les cas.

Mots-clés : Géométrie algorithmique, cercle minimum

INTRODUCTION

Etant donné un ensemble $\mathcal{P}$ de points de $\mathbb{R}^d$, le cercle couvrant minimum de $\mathcal{P}$ est le cercle de diamètre le plus petit contenant tous les points de $\mathcal{P}$.
En géométrie algorithmique, il est d'usage de commencer par trouver un algorithme naïf, puis de chercher ensuite une optimisation dans la résolution du problème. C'est la raison pour laquelle nous analyserons l'algorithme naïf dans un premier temps, afin d'identifier les cas qui ralentissent l'algorithme, puis nous allons présenter l'algorithme de Welzl qui permet la résolution du problème en temps linéaire.

Nous travaillerons sur un ensemble fini de points dans $\mathbb{R}^2$ pour des raisons de simplicité, mais il existe un problème similaire en $\mathbb{R}^3$ appelé le calcul de la sphère minimum englobante On parle d'hypersphere englobante pour des problèmes en dimension d (d>3).

1 Definitions et notations

Nous utiliserons les notations suivantes :
— - soit $\mathcal{P}$ l'ensemble de points de $\mathbb{R}^2$, et n le nombre de points dans $\mathcal{P}$;
— - soit $\mathcal{C}$ le cercle couvrant minimum de l'ensemble de points $\mathcal{P}$;
— - soit $\delta(A, B)$ la distance euclidienne entre le point A et le point B et $\delta^2(A, B)$ cette même distance euclidienne au carré ;

Voici les différents concepts mathématiques dont nous aurons besoin au sein de ce projet.

Definition 1
Un cercle, défini par son centre c ainsi que son rayon r, est dit couvrant minimum pour un ensemble de points $\mathcal{P}$ si $\forall p \in \mathcal{P}$, $\delta(p, c) \leqslant r$.

Definition 2
Un point p de $\mathcal{P}$ est sur le cercle lorsque $\delta(p, c) = r$.

Definition 3
Le cercle circonscrit à 3 points est défini par le cercle qui passe par les 3 sommets de ce triangle.

Definition 4
Le centre du cercle circonscrit se définit comme le point d'intersection des médiatrices d'un triangle.

C'est grâce à la **Definition 4** que nous allons calculer les coordonnées du centre du cercle circonscrit à 3 points. Nous faisons appel à un système à 2 équations correspondant à 2 des 3 médiatrices d'un triangle choisies aléatoirement.

Soit $E1$ l'équation de la 1ière médiatrice, et $E2$ l'équation de la 2ième médiatrice.

$$(E1) : ax_1 + by_1 = c$$
$$(E2) : dx_2 + ey_2 = f$$

Comme le centre du cercle circonscrit est l'intersection des médiatrices, ses coordonnées doivent respecter les 2 équations ci-dessus.
Nous obtenons donc le système d'équation suivant :

$$\begin{cases} ax + by = c & (1) \\ dx + ey = f & (2) \end{cases}$$

En multipliant (1) par d et (2) par a, nous obtenons :

$$\begin{cases} adx + bdy = cd & (1) \\ dax + eay = fa & (2) \end{cases}$$

Avec $(1) - (2)$, on obtient :

$$\begin{aligned} (adx + bdy) - (dax - eay) &= cd - fa \\ y(bd - ea) &= cd - fa \\ y &= \frac{cd-fa}{bd-ea} \end{aligned}$$

De la même manière, en multipliant (1) par e et (2) par b, nous obtenons :

$$\begin{cases} aex + bey = ce & (1) \\ dbx + eby = fb & (2) \end{cases}$$

Avec $(1) - (2)$, on obtient :

$$\begin{aligned} (aex + bey) - (dbx - eby) &= ce - fb \\ x(ae - db) &= ce - fb \\ x &= \frac{ce-fb}{ae-db} \end{aligned}$$

Soit 3 points A, B, C distincts formant un triangle quelconque.
Soit I le milieu du segment $[AB]$ et J le milieu du segment $[BC]$.
Supposons que (1) soit l'équation de la médiatrice de $[AB]$ et (2) soit l'équation de la médiatrice de $[BC]$.
Alors, nous obtenons :

$$\begin{aligned} a &= B.x - A.x \\ b &= B.y - A.y \\ c &= I.x(B.x - A.x) + I.y(B.y - A.y) \\ d &= C.x - B.x \\ e &= C.y - B.y \\ f &= J.x(C.x - B.x) + J.y(C.y - B.y) \end{aligned}$$

Dans le cas particulier où $ae - db$ (respectivement $bd - ea$) vaut 0, alors nous renvoyons une coordonnée nulle car cela veut dire que nous avons au moins 2 points avec les mêmes coordonnées.
Même si mathématiquement parlant, cela n'a pas forcément de sens de renvoyer une coordonnée nulle, dans notre cas, nous nous en accommoderons, étant donné que nous supposons que chaque point possède des coordonnées uniques.

En utilisant ces notations, nous pouvons poser les deux lemmes suivants :

Lemma 1.1. *$\forall$ p,$q \in \mathcal{P}$, si le cercle de diamètre $d = \delta(p, q)$ couvre tout autre point de $\mathcal{P}$, alors ce cercle est un cercle couvrant de rayon minimum.*

Lemma 1.2. *En dimension 2, il existe un unique cercle passant par 3 points non-colinéaires. On parle de cercle circonscrit au triangle formé par ces 3 points.*

2 Calcul naïf du cercle couvrant minimum

L'algorithme naïf est basé sur l'application directe des deux lemmes précédents. Nous pouvons séparer cet algorithme en deux phases :

— - la première consiste à essayer de trouver un cercle à l'aide de 2 points qui contiendrait donc tous les autres points de $\mathcal{P}$.
Si un tel cercle existe, alors c'est le cercle couvrant minimum, et l'algorithme retourne ce cercle.

— - la seconde phase n'intervient que dans le cas où la recherche du cercle couvrant minimum avec 2 points échoue. Dans ce cas, l'algorithme va chercher le cercle couvrant minimum à l'aide de 3 points p, q, r en calculant le cercle circonscrit à ces 3 points dont le centre c est tel que $\delta(c,p) = \delta(c,q) = \delta(c,r)$.

Algorithm 1 Algorithme naïf du calcul du cercle couvrant minimum

INPUT : $\mathcal{P}$: la liste des points dans le plan
OUTPUT : $\mathcal{C}$: le cercle couvrant minimum de l'ensemble $\mathcal{P}$

1: **for** $p \in \mathcal{P}$ **do**
2: **for** $q \in \mathcal{P}$, $p \neq q$ **do**
3: $\mathcal{C} \leftarrow$ cercle de centre $c = (\frac{p.x+q.x}{2}; \frac{p.y+q.y}{2})$ et de rayon $r = \frac{\delta(p,q)}{2}$
4: **if** $\mathcal{C}$ contient tous les points de $\mathcal{P}$ **then**
5: **return** $\mathcal{C}$
6: **end if**
7: **end for**
8: **end for**
9: $\mathcal{C}_{min} \leftarrow$ le cercle circonscrit de 3 points de rayon minimum
10: $r_{min} \leftarrow$ le rayon du cercle $\mathcal{C}_{min}$
11: **for** $p \in \mathcal{P}$ **do**
12: **for** $q \in \mathcal{P}$, $p \neq q$ **do**
13: **for** $r \in \mathcal{P}$, $p \neq r$, $q \neq r$ **do**
14: $\mathcal{C} \leftarrow$ cercle de centre $c = (\frac{p.x+q.x}{2}; \frac{p.y+q.y}{2})$ et de rayon $r = \frac{\delta(p,q)}{2}$
15: **if** $\mathcal{C}$ contient tous les points de $\mathcal{P}$ **then**
16: **if** $\mathcal{C}.r < r_{min}$ **then**
17: $\mathcal{C}_{min} \leftarrow \mathcal{C}$
18: $r_{min} \leftarrow \mathcal{C}.r$
19: **end if**
20: **end if**
21: **end for**
22: **end for**
23: **end for**
24: **return** $\mathcal{C}$

Pour tester si le cercle contient tous les points d'un ensemble donné, nous faisons appel à la fonction "Contient" ci-dessous.

L'algorithme 1 cherche, dans un premier temps, toutes les paires de points possibles de manière à ce qu'elles puissent former le diamètre du cercle : il y a donc en tout $\frac{n(n-1)}{2}$ combinaisons possibles.
Puis, si cette recherche échoue, on considère tous les triplets de points possibles de

Algorithm 2 Fonction Contient

INPUT : $\mathcal{C}$: un cercle quelconque de centre c et de rayon r, $\mathcal{P}$: la liste des points dans le plan
OUTPUT : TRUE or FALSE

```
1: for p ∈ P do
2:    if δ(p, c) > r then
3:       return FALSE
4:    end if
5: end for
6: return TRUE
```

manière à calculer le cercle circonscrit à ces 3 points : il y a donc en tout $\frac{n(n-1)(n-2)}{6}$ combinaisons possibles.

La fonction "Contient" utilisé dans l'algorithme 1 aux lignes 4 et 15 parcourt toute la liste si tous les points sont contenus dans le cercle.

Ainsi, la compléxité de l'algorithme naïf est en $\mathcal{O}(n^4)$, ce qui en fait un algorithme très peu performant.

3 Calcul du cercle couvrant minimum - Algorithme de Welzl

En 1991, Welzl a proposé un algorithme qui a l'avantage d'avoir une complexité en $\mathcal{O}(n)$. D'après le **lemme 1.2**, il existe un unique cercle passant par 3 points non-colinéraires.

Nous pouvons donc ne considérer que les cas où nous avons 1, 2 ou 3 points situés exactement sur le cercle, car dans tous les autres cas, le cercle circonscrit à 3 points suffit pour trouver le cercle passant par tous les autres points.
C'est la raison pour laquelle nous utilisons un tableau de 3 points que nous appelons *boundary*, qui correspond aux points situés sur le cercle, c'est-à-dire les points dont la distance $\delta(x, c) = \mathcal{C}.rayon$ pour un cercle $\mathcal{C}$ donné de centre c et de rayon *rayon*, et x appartenant au tableau *boundary*.

Il s'agit d'un algorithme récursif dont le pseudo-code est présenté ci-dessous.

3.1 Principe de l'algorithme

Notre algorithme consiste à construire un cercle couvrant à chaque appel récursif en fonction du nombre de points stockés dans le tableau *boundary*, puis à regarder s'il existe des points qui ne sont pas contenus dans ce cercle en considérant tous les points $\mathcal{P}$ jusqu'à l'indice courant.
Si un tel point existe, alors on l'ajoute dans le tableau *boundary* puis on rappelle la fonction de manière récursive, en augmentant la taille du tableau *boundary* de 1.
Sinon, on ne fait rien car cela veut dire que le point considéré est déjà situé dans le cercle.

A chaque appel à la fonction récursive *b_minidisk()*, nous construisons un cercle passant par 1, 2 ou 3 points grâce à la fonction *buildMinidisk()*. Lors du parcours de

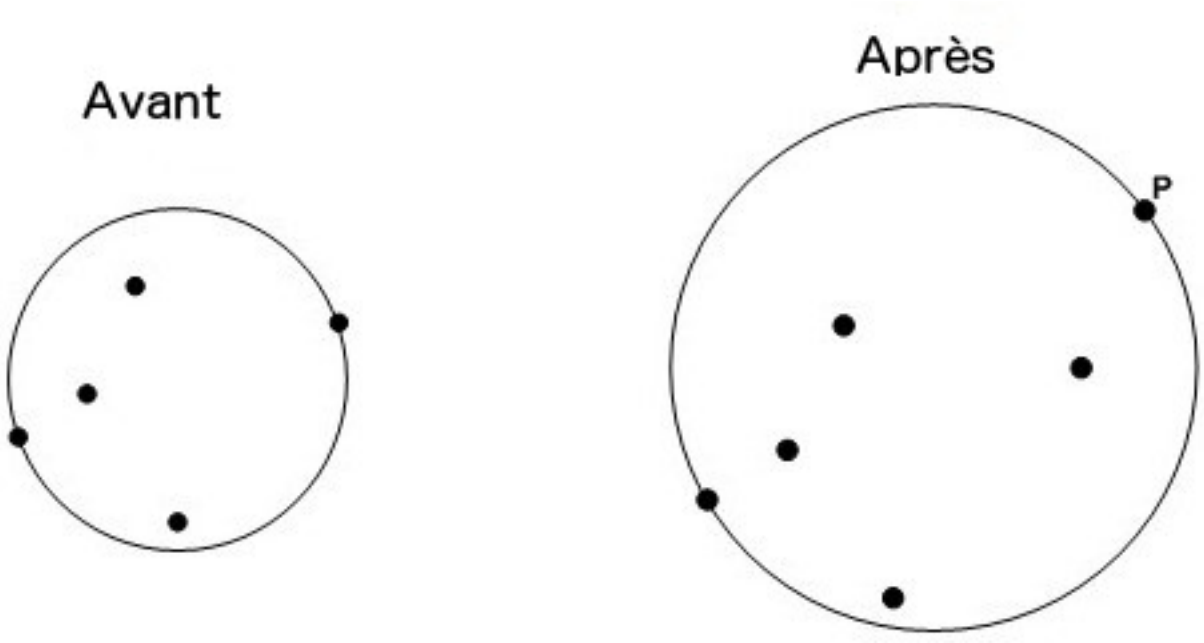

FIGURE 1 – Mise à jour du cercle couvrant minimum courant à l'itération i

Nous pouvons observer sur l'image suivante que lorsque l'on traite un point qui est situé à l'extérieur du cercle courant, alors le cercle couvrant est mis à jour, et contient nécessairement tous les points que l'on a déjà traité.

l'ensemble de points $\mathcal{P}$, nous regardons tous les points qu'on a déjà traité, jusqu'à l'index du point courant.
Ainsi, tous les points déjà traités sont nécessairement situés dans ou sur le cercle courant car ce même cercle a été créé à l'aide du tableau *boundary* qu'on met à jour avant chaque appel récursif.
De ce fait, à première vue on pourrait penser qu'il peut y avoir dans le pire des cas un appel récursif pour chaque élément de la liste de points $\mathcal{P}$, et ce $n(n-1)...1 = n!$ fois.
Or l'appel récursif ne se déclenche que lorsque l'on trouve un point qui n'est pas contenu dans le cercle courant, qui est censé contenir tous les points jusqu'à l'indice courant dans la liste $\mathcal{P}$ (qui correspond donc à l'argument *index* dans l'algorithme).

La FIGURE 1 illustre à un instant donné la mise à jour du cercle couvrant minimum. Donc nous pouvons en conclure que l'algorithme possède bien une complexité en $\mathcal{O}(n)$.

Algorithm 3 Algorithme de Welzl

INPUT : $\mathcal{P}$: la liste des points dans le plan
OUTPUT : $\mathcal{C}$: le cercle couvrant minimum de l'ensemble $\mathcal{P}$

1: $\mathcal{C} \leftarrow b\_minidisk(\mathcal{P}, \mathcal{P}.size, \mathcal{L} = [\,], 0)$
2: **return** $\mathcal{C}$

Algorithm 4 b_minidisk()

INPUT :
$\mathcal{P}$: la liste des points dans le plan,
index : l'index jusqu'où on effectue la récursion, $\mathcal{P}$,
boundary : le tableau des points situés sur le cercle,
b : le nombre de points situés sur le cercle
OUTPUT : $\mathcal{C}$ le cercle couvrant minimum de l'ensemble $\mathcal{P}$

1: $\mathcal{C} \leftarrow buildMinidisk(boundary, b)$
2: **for** $i = 0$ **to** $i = index$-1 **do**
3: $p \leftarrow \mathcal{P}[i]$
4: **if** $\mathcal{C}$ ne contient pas p **and** $b < 3$ **then**
5: $boundary[b] = p$
6: $\mathcal{C} \leftarrow b\_minidisk(\mathcal{P}, i, boundary, b{+}1)$
7: **end if**
8: **end for**
9: **return** $\mathcal{C}$

Algorithm 5 buildMinidisk()

INPUT :
boundary : le tableau des points situés sur le cercle,
b : le nombre de points situés sur le cercle
OUTPUT : $\mathcal{C}$ le cercle couvrant minimum des points dans *boundary*

1: $\mathcal{C} \leftarrow$ cercle de centre $(0;0)$ et de rayon 0
2: **if** $b = 1$ **then**
3: $\mathcal{C} \leftarrow$ cercle de centre $boundary[0]$ et de rayon 0
4: **else if** $b = 2$ **then**
5: $\mathcal{C} \leftarrow$ cercle de centre $c = (\frac{p.x+q.x}{2}; \frac{p.y+q.y}{2})$ et de rayon $r = \frac{\delta(p,q)}{2}$
6: **else if** $b = 3$ **then**
7: $\mathcal{C} \leftarrow$ cercle circonscrit aux points $boundary[0]$, $boundary[1]$, $boundary[2]$
8: **end if**

4 Résultats expérimentaux

Une base de test nous a été fournie pour pouvoir tester nos algorithmes : elle contient 1664 instances de test, chacune d'entre elle contenant 256 lignes, et chaque ligne représentant les coordonnées d'un point dans un plan en 2D.

Le tableau en annexe montre les résultats expérimentaux des deux algorithmes présentés, le naïf et celui de Welzl, pour les 100 premières instances de test.
La colonne de gauche concerne les résultats pour l'algorithme naïf, et la colonne de droite ceux de l'algorithme de Welzl.
La colonne rayon représente le rayon du cercle couvrant minimum de l'instance n°i, arrondi à l'entier supérieur.
Nous observons que la différence entre les rayons trouvés avec l'algorithme naïf et ceux trouvés avec l'algorithme de Welzl est nulle pour les 100 premières instances.
Le fichier tableFormat.txt fourni avec le code source montre que l'on retrouve la même caractéristique sur les autres instances.

D'autre part, nous constatons également que la comparaison des coordonnées du centre de chaque cercle couvrant minimum de chaque instance est identique pour les 2 algorithmes (voir.

Nous pouvons donc conclure que nos 2 algorithmes trouvent le même résultat, à savoir le même cercle couvrant minimum pour toutes les instances données de la base de test.

5 Discussions

Les résultats expérimentaux nous permettent de constater une énorme différence en terme de temps d'exécution entre l'algorithme naïf et l'algorithme de Welzl, ce qui est logique lorsque l'on compare les complexités de chacun de ces algorithmes ($\mathcal{O}(n^4)$ vs. $\mathcal{O}(n)$).

Il peut arriver que l'algorithme naïf soit aussi efficace au niveau du temps d'exécution que l'algorithme de Welzl, lorsque le cercle couvrant minimum est trouvé directement à l'aide de 2 points. Dans cette configuration, ce cercle est retourné au pire cas en temps polynomial $\mathcal{O}(n^2)$, et donc face à une instance qui ne contient que 256 points, le temps d'exécution avec les machines d'aujourd'hui diffère de très peu.

Cependant, en considérant des instances avec un nombre de points plus élevé, la différence entre les temps d'exécution risque dans ce cas de se faire sentir. Il serait alors plus judicieux d'utiliser un algorithme comme celui de Welzl.

Avec l'algorithme de Welzl, la meilleure situation correspondrait au fait que le cercle couvrant minimum soit trouvé directement à l'aide des 2 premiers points de la liste $\mathcal{P}$, auquel cas il n'y aurait aucun appel récursif. Au contraire, la pire configuration serait qu'on ferait un appel récursif à chaque parcourt de l'ensemble $\mathcal{P}$, ce qui permet de trouver le cercle en faisant au pire n appel récursif.

Il s'agit donc d'un excellent résultat, et c'est, probablement à l'heure actuelle, la meilleure technique connue pour résoudre le problème du cercle couvrant minimum. Pour faire mieux, il faudrait trouver un algorithme qui trouve un tel cercle en temps logarithmique, ce qui implique de ne considérer qu'un ensemble qui diminue à chaque itération,

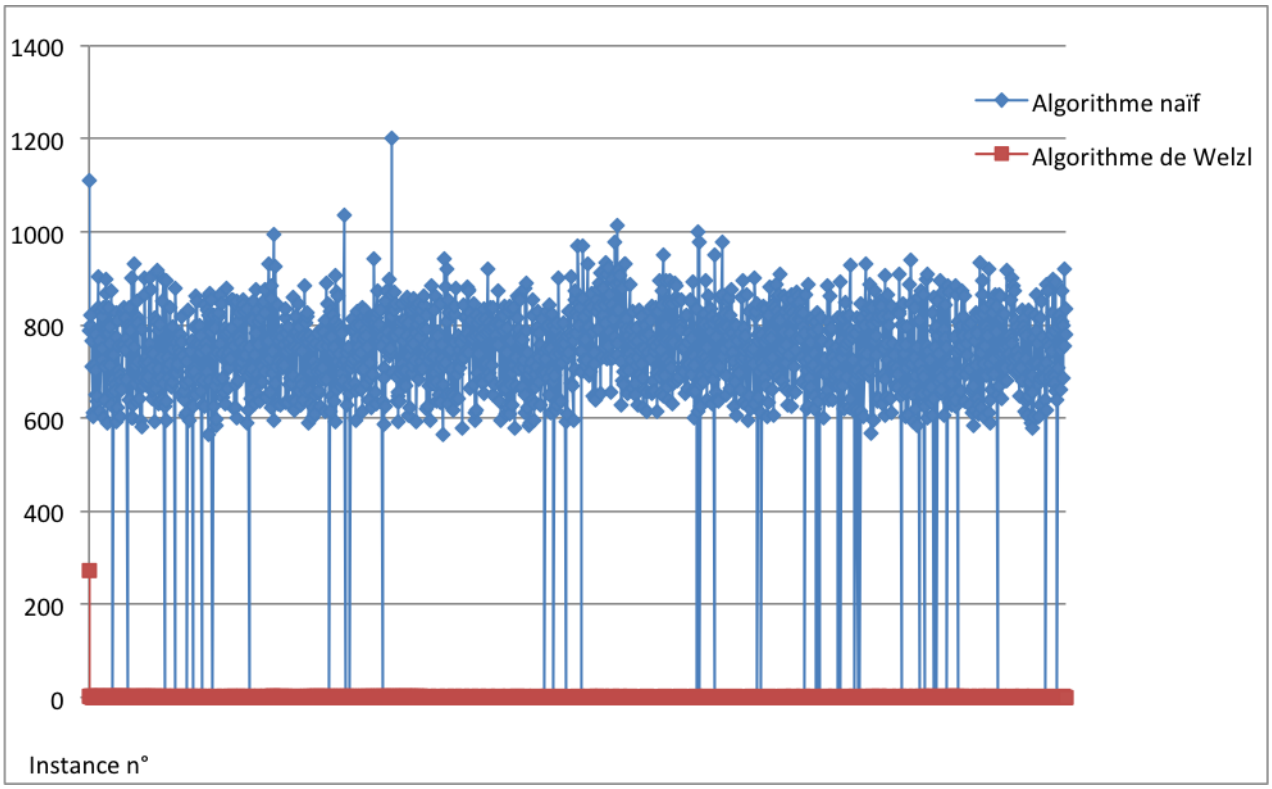

FIGURE 2 – Comparaison du temps d'exécution entre l'algorithme naïf (en bleu) et l'algorithme de Welzl (en rouge)

sans avoir à parcourir toute la liste, ce qui relève donc de l'impossible.

6 Conclusion

Ce projet avait pour but de comparer à quel point l'efficacité en terme de temps d'exécution entre l'algorithme naïf et celui de Welzl pouvait varier, tout en ayant la certitude d'obtenir les mêmes résultats dans les 2 cas.

Nous avons, d'une part, l'algorithme naïf dont la complexité est en $\mathcal{O}(n^4)$, ce qui est passablement horrible, et d'autre part, l'algorithme de Welzl qui possède lui une complexité linéaire en seulement $\mathcal{O}(n)$.

Nous pouvons constater que notre algorithme de Welzl rend effectivement les mêmes résultats que ceux de l'algorithme naïf, ce qui permet d'en valider son implémentation.

L'efficacité des 2 algorithmes a été discutée dans la section précédente (**Discussions**), et il apparaît clair qu'aujourd'hui, un algorithme tel que celui de Welzl, ayant une complexité linéaire, est sans doute le plus efficace en terme de temps d'exécution pour résoudre le problème du cercle couvrant minimum.

	Algorithme Naïf		Algorithme de Welzl	
instance n°	Temps d'exécution (ms)	rayon	Temps d'exécution (ms)	rayon
1	1111	197	272	197
2	788	197	1	197
3	799	198	1	198
4	820	199	1	199
5	765	197	1	197
6	711	198	0	198
7	611	198	1	198
8	604	199	0	199
9	786	197	0	197
10	712	198	0	198
11	701	197	1	197
12	780	196	0	196
13	649	196	0	196
14	636	199	1	199
15	646	198	0	198
16	903	199	0	199
17	836	197	1	197
18	766	197	0	197
19	615	198	2	198
20	728	196	0	196
21	830	197	1	197
22	724	198	0	198
23	748	195	0	195
24	819	199	1	199
25	837	199	0	199
26	600	197	0	197
27	711	198	1	198
28	639	199	0	199
29	897	197	1	197
30	868	197	0	197
31	793	198	0	198
32	590	198	1	198
33	781	197	0	197
34	836	196	0	196
35	712	199	1	199
36	759	197	0	197
37	795	198	1	198
38	628	198	0	198
39	873	199	0	199
40	656	198	1	198
41	0	198	0	198
42	736	197	1	197
43	683	196	0	196
44	671	199	1	199
45	658	196	0	196
46	774	193	1	193
47	827	198	0	198
48	592	199	0	199
49	809	196	1	196
50	623	199	0	199

	Algorithme Naïf		Algorithme de Welzl	
instance n°	Temps d'exécution (ms)	rayon	Temps d'exécution (ms)	rayon
51	702	197	1	197
52	755	200	0	200
53	602	198	0	198
54	692	200	1	200
55	681	199	2	199
56	661	198	1	198
57	756	196	0	196
58	825	199	0	199
59	831	197	1	197
60	836	198	0	198
61	758	196	0	196
62	814	196	0	196
63	747	199	1	199
64	771	199	0	199
65	684	196	0	196
66	0	198	1	198
67	798	192	0	192
68	638	197	0	197
69	838	197	0	197
70	760	198	1	198
71	796	198	0	198
72	787	200	0	200
73	599	197	0	197
74	900	199	1	199
75	601	194	0	194
76	629	198	0	198
77	931	199	1	199
78	753	198	0	198
79	676	200	0	200
80	714	198	1	198
81	635	198	0	198
82	850	197	0	197
83	814	194	0	194
84	722	195	0	195
85	803	198	0	198
86	682	198	1	198
87	643	198	0	198
88	594	198	0	198
89	679	195	0	195
90	580	199	1	199
91	790	198	0	198
92	668	194	0	194
93	686	195	1	195
94	720	198	1	198
95	865	199	0	199
96	902	197	0	197
97	677	199	0	199
98	728	199	1	199
99	679	199	0	199
100	734	194	0	194

TABLE 1 – Tableau des 100 premières instances

instance n°	Naïf	Welzl
1	[398 ;301]	[398 ;301]
2	[398 ;301]	[398 ;301]
3	[400 ;299]	[400 ;299]
4	[397 ;301]	[397 ;301]
5	[396 ;299]	[396 ;299]
6	[403 ;298]	[403 ;298]
7	[401 ;298]	[401 ;298]
8	[399 ;299]	[399 ;299]
9	[400 ;299]	[400 ;299]
10	[398 ;300]	[398 ;300]
11	[396 ;298]	[396 ;298]
12	[400 ;304]	[400 ;304]
13	[397 ;300]	[397 ;300]
14	[395 ;301]	[395 ;301]
15	[400 ;301]	[400 ;301]
16	[398 ;303]	[398 ;303]
17	[401 ;300]	[401 ;300]
18	[398 ;306]	[398 ;306]
19	[399 ;300]	[399 ;300]
20	[399 ;296]	[399 ;296]
21	[397 ;299]	[397 ;299]
22	[400 ;301]	[400 ;301]
23	[394 ;303]	[394 ;303]
24	[398 ;299]	[398 ;299]
25	[401 ;300]	[401 ;300]
26	[402 ;296]	[402 ;296]
27	[399 ;302]	[399 ;302]
28	[400 ;299]	[400 ;299]
29	[398 ;295]	[398 ;295]
30	[398 ;300]	[398 ;300]
31	[397 ;301]	[397 ;301]
32	[399 ;301]	[399 ;301]
33	[400 ;297]	[400 ;297]
34	[399 ;296]	[399 ;296]
35	[397 ;302]	[397 ;302]
36	[400 ;301]	[400 ;301]
37	[398 ;300]	[398 ;300]
38	[398 ;299]	[398 ;299]
39	[398 ;300]	[398 ;300]
40	[399 ;299]	[399 ;299]
41	[397 ;299]	[397 ;299]
42	[397 ;297]	[397 ;297]
43	[401 ;303]	[401 ;303]
44	[401 ;300]	[401 ;300]
45	[394 ;302]	[394 ;302]
46	[391 ;310]	[391 ;310]
47	[397 ;298]	[397 ;298]
48	[400 ;300]	[400 ;300]
49	[395 ;300]	[395 ;300]
50	[398 ;298]	[398 ;298]

instance n°	Naïf	Welzl
51	[400 ;297]	[400 ;297]
52	[399 ;300]	[399 ;300]
53	[399 ;298]	[399 ;298]
54	[400 ;299]	[400 ;299]
55	[398 ;299]	[398 ;299]
56	[401 ;299]	[401 ;299]
57	[401 ;301]	[401 ;301]
58	[396 ;299]	[396 ;299]
59	[402 ;303]	[402 ;303]
60	[398 ;300]	[398 ;300]
61	[397 ;303]	[397 ;303]
62	[400 ;300]	[400 ;300]
63	[395 ;302]	[395 ;302]
64	[399 ;298]	[399 ;298]
65	[402 ;300]	[402 ;300]
66	[401 ;299]	[401 ;299]
67	[398 ;305]	[398 ;305]
68	[399 ;297]	[399 ;297]
69	[395 ;299]	[395 ;299]
70	[396 ;299]	[396 ;299]
71	[400 ;300]	[400 ;300]
72	[399 ;299]	[399 ;299]
73	[402 ;299]	[402 ;299]
74	[399 ;299]	[399 ;299]
75	[396 ;300]	[396 ;300]
76	[396 ;300]	[396 ;300]
77	[397 ;300]	[397 ;300]
78	[399 ;300]	[399 ;300]
79	[399 ;300]	[399 ;300]
80	[401 ;301]	[401 ;301]
81	[399 ;301]	[399 ;301]
82	[393 ;302]	[393 ;302]
83	[400 ;295]	[400 ;295]
84	[398 ;297]	[398 ;297]
85	[396 ;301]	[396 ;301]
86	[400 ;300]	[400 ;300]
87	[403 ;299]	[403 ;299]
88	[401 ;300]	[401 ;300]
89	[403 ;302]	[403 ;302]
90	[399 ;299]	[399 ;299]
91	[402 ;300]	[402 ;300]
92	[402 ;298]	[402 ;298]
93	[400 ;300]	[400 ;300]
94	[398 ;299]	[398 ;299]
95	[400 ;300]	[400 ;300]
96	[399 ;302]	[399 ;302]
97	[401 ;300]	[401 ;300]
98	[400 ;300]	[400 ;300]
99	[400 ;300]	[400 ;300]
100	[397 ;297]	[397 ;297]

TABLE 2 – Tableau des 100 premières instances correspondant aux coordonnées du centre du cercle couvrant minimum pour chaque instance