SCIENCES SORBONNE UNIVERSITÉ

CPA

Problème du Cercle Minimum en Géométrie

Malek BOUZARKOUNA
28706508
12 Mars 2024

Superviseur académique :
Binh-Minh BUI-XUAN

Master STL 2023/2024

Table des matières

Résumé

Etant donné un ensemble fini de point de $\mathbb{R}^2$, on cherche montrer l'éfficacité de l'algorithme de Welzl comparer à un algorithme naïf. Bien que l'algorithme de Welzl commet des erreurs dans ces cercles minimums, il le résout dans la majorité des cas dans un temps linéaire.

1 Introduction

Le problème du cercle minimum ou "Smallest enclosing disks" est un problème classique en géométrie algorithmique. Le problème du cercle minimum est assez simple en apparence : il s'agit de trouver le plus petit cercle englobant tous les points d'un ensemble bidimensionnel, avec un rayon minimal.

Malgré sa simplicité apparente, résoudre ce problème de manière efficace est une tâche bien plus ardue. Des approches ont été réalisées dans la littérature pour résoudre ce problème, que ce soit avec des algorithmes randomisés tels que "Linear programming and convex hulls made easy" de R. Seidel ou encore avec des algorithmes déterministes tels que "Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems" de N. Megiddo.

Dans le cadre de notre étude, nous proposons une structure de données : une liste de n points cartésiens dans $\mathbb{R}^2$. Avec cette liste de points, nous nous concentrons sur le développement et la comparaison d'algorithmes. Nous explorons des approches naïves avec une complexité élevée en $O(n^4)$ car elles examinent toutes les combinaisons possibles pour trouver le cercle minimal, ainsi que des méthodes plus avancées comme l'algorithme de Welzl. Ces types d'algorithmes réduisent considérablement la complexité et accélèrent la résolution du problème, utilisant des stratégies telles que la randomisation pour atteindre une efficacité accrue, souvent avec une complexité temporelle moyenne en $O(n)$.

Dans les sections suivantes, nous approfondirons notre étude en examinant les algorithmes célèbres. La section 3, "Analyse et Présentation Théorique des Algorithmes Connus", se concentrera sur l'analyse des méthodes établies dans la littérature. Nous critiquerons des algorithmes existant dans la Littérature, et de leurs méthodes, discutant de manière concise de leurs forces et faiblesses.

2 Définition, notations et préliminaires géométrique

Pour commencer on se propose de définir quelques notions géométrique utile à notre sujet et qui seront utilisé dans les algorithmes.

Definition 2.1 (Cercle Minimum) *Le cercle minimum, consiste à trouver le cercle le plus petit englobant un ensemble borné de points du plan euclidien. [11]*

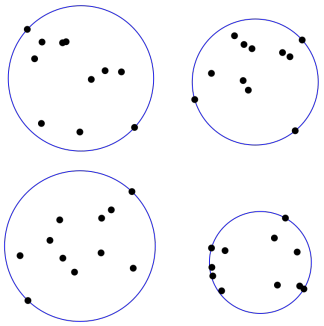

FIGURE 1 – Illustration de la définition 2.2.

On note notre liste de points n par une liste de points cartésiens, où chaque point est représenté par ses coordonnées (x, y). On notera donc la distance entre deux points du plan p_1 et p_2 par la distance euclidienne : $\|\mathbf{p}_1 - \mathbf{p}_2\|$.[4]

Definition 2.2 (Distance euclidienne) *Considérons deux points p et p' de coordonnées respectives (x, y) et (x', y'). Leur distance euclidienne est donnée par la formule :*

$$\|p - p'\| = \sqrt{(x - x')^2 + (y - y')^2}$$

Definition 2.3 (Cercle circonscrit) *En géométrie du triangle, le cercle circonscrit à un triangle non plat est l'unique cercle passant par ses trois sommets.[2]*

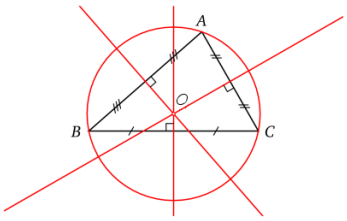

FIGURE 2 – Illustration de la définition 2.3.

Definition 2.4 (Enveloppe convexe) *L'enveloppe convexe d'un ensemble de points est le plus petit ensemble convexe qui les contient tous. C'est un polyèdre dont les sommets sont des points de l'ensemble. Le calcul de l'enveloppe convexe consiste à calculer une représentation compacte de l'enveloppe, le plus souvent les sommets de celle-ci.[1]*

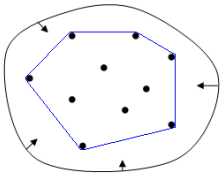

FIGURE 3 – Illustration de la définition 2.4.

Definition 2.5 (Diagramme de Voronoï) *L'enveloppe convexe d'un ensemble de points est le plus petit ensemble convexe qui les contient tous. C'est un polyèdre dont les sommets sont des points de l'ensemble. Le calcul de l'enveloppe convexe consiste à calculer une représentation compacte de l'enveloppe, le plus souvent les sommets de celle-ci.[3]*

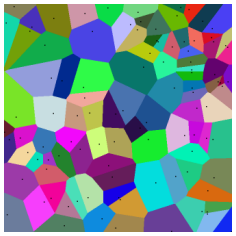

FIGURE 4 – Illustration de la définition 2.5.

3 Analyse et Présentation Théorique des Algorithmes Connus

Dans la littérature, plusieurs approches ont été abordées pour résoudre le problème du cercle minimum, en utilisant des théorèmes géométriques, ainsi que des approches déterministes et randomisées. Dans cette section, nous allons brièvement analyser des algorithmes connus.

3.1 Algorithmes Randomisés

3.1.1 R. Seidel - "Linear programming and convex hulls made easy"

L'algorithme de Seidel [9] utilise la construction d'enveloppes convexes. Cette approche plus directe et moins complexe, a rendu le problème du cercle minimum plus abordable permetant à d'autre informaticien dont Emo Welzl de s'inspirer de ses travaux. L'algorithme s'appuie sur un procédé de randomisation pour la résolution du problème. Il adopte une méthode incrémentale, c'est à dire en traitant les points un par un et en mettant à jour la solution optimale.

L'efficacité de l'algorithme de Seidel repose principalement sur sa stratégie de sélection aléatoire des points. Cette technique évite les scénarios défavorables qui peuvent entraver les performances des méthodes traditionnelles. Grâce à cette randomisation, l'algorithme parvient à réduire de manière significative sa complexité temporelle moyenne, souvent évaluée à $O(n)$ au pire cas, ce qui le rend particulièrement bien adapté aux ensembles de données volumineux.

3.2 Algorithmes Déterministes

3.2.1 N. Megiddo - "Linear-time algorithms for linear programming in R 3 and related problems"

L'algorithme de Megiddo[7], fondamental dans la théorie de la programmation linéaire pour $\mathbb{R}^2$ et $\mathbb{R}^3$, opère avec une complexité en $O(n)$. Cette méthode innovante repose sur une stratégie d'élagage où, à chaque étape, les contraintes non pertinentes sont éliminées, réduisant ainsi progressivement l'espace de recherche jusqu'à ne retenir que 3 points.. Contrairement aux méthodes antérieures qui pouvaient présenter une complexité temporelle exponentielle en fonction du nombre de contraintes, l'approche de Megiddo permet d'atteindre la solution optimale de manière plus directe et efficiente. Grâce à son processus itératif qui s'appuie sur des principes géométriques fondamentaux, comme l'utilisation de médiatrices pour éliminer les points superflus, l'algorithme converge rapidement vers la solution sans être entravé par le nombre de données.

3.2.2 S. Skyum - "A simple algorithm for computing the smallest circle"

L'algorithme de Skyum [10], dont le pseudocode est présenté dans l'Algorithme 1, est une méthode itérative dont la force réside dans l'utilisation d'un ensemble de points pour construire un diagramme de Voronoï. Conçu pour traiter efficacement les ensembles de points formant des polygones convexes, cet algorithme se distingue par son approche qui consiste à réduire progressivement l'ensemble de points en évaluant les relations géométriques entre eux. À chaque itération, il examine les points un par un, déterminant le cercle englobant minimum en fonction des propriétés telles que le rayon et l'angle formé par des triplets de points. Cette méthode permet à l'algorithme d'avoir une complixité pire cas en $O(nlogn)$ le rendant très compétitif par rapport aux autres méthodes de construction de diagrammes de Voronoï.

L'un des avantage de l'algorithme est qu'il gére efficacement les situations où les points peuvent être à une distance immensément grande. Utilisant des médiatrices et des angles clés, il maintient la précision du diagramme de Voronoï, même dans des configurations géométriques complexes. Cet avantage assure que l'algorithme reste robuste et précis pour une variété de configurations de points.

Algorithm 1 Algorithme de Skyum

for all $p \in S$ **do**
  add $v(p)$ to K ;
end for
if $n > 2$ **then**
  repeat
    find p maximizing radius(before(p), p, next(p)), angle(before(p), p, next(p)) ;
    $q \leftarrow$ before(p) ;
    $c \leftarrow$ centre(q, p, next(p)) ;
    add c to K ;
    add $(c, v(p))$ and $(c, v(q))$ to E ;
    $v(q) \leftarrow c$;
    next(q) $\leftarrow$ next(p) ;
    before(next(q)) $\leftarrow q$;
    $n \leftarrow n - 1$;
  until $n = 2$
  add $(v(q), v(\text{next}(q)))$ to E ;
else
  if $n = 2$ **then**
    $S \leftarrow \{p_1, p_2\}$;
    add $(v(p_1), v(p_2))$ to E ;
  end if
end if=0

4 Algorithme Naïf

La première approche pour le problème du cercle minimum est l'algorithme naïf, dont le pseudo-code est présenté ci-dessous dans l'Algorithme 2. Le principe est simple :

Premier cas : on examine tous les couples de points afin de trouver une paire p_1, p_2 qui détermine un cercle. Le cercle est défini avec un diamètre $\|\mathbf{p}_1 - \mathbf{p}_2\|$ et un centre $\frac{p_1+p_2}{2}$. Si ce cercle englobe tous les autres points, alors l'algorithme le retourne.

Deuxième cas : si aucun cercle satisfaisant n'est trouvé dans le premier cas, l'algorithme prend 3 points pour déterminer le cercle circonscrit le plus petit englobant tous les points.

Algorithm 2 Algorithme naif du cercle minimum

0: *resultat* $\leftarrow$ cercle de rayon infini
0: **for all** p dans *Points* **do**
0: **for all** q dans *Points* **do**
0: $c \leftarrow$ cercle avec centre $\frac{p+q}{2}$ et de diamètre $||pq||$
0: **if** c couvre tous les points dans *Points* **then**
0: **return** c
0: **end if**
0: **end for**
0: **end for**
0: **for all** p dans *Points* **do**
0: **for all** q dans *Points* **do**
0: **for all** r dans *Points* **do**
0: $c \leftarrow$ cercle circonscrit de p, q et r
0: **if** c couvre *Points* et c est plus petit que *resultat* **then**
0: *resultat* $\leftarrow c$
0: **end if**
0: **end for**
0: **end for**
0: **end for**
0: **return** *resultat* =0

Le cercle trouvé est bien le cercle minimum avec le rayon possible, cette affirmation est possible en ce reposant sur ces deux lemmes :

Lemme 1 *Si un cercle de diamètre égale 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 et un seul cercle passant par 3 points non-colinéaires.*

En terme de complexité, cet algorithme dans le pire cas est en $O(n^4)$. En effet cela est dû au deuxième cas avec le cercle circonscrit nous avons pour la liste de points une triple boucle imbriquée ce qui nous fait du $O(n^3)$. Et un dernier facteur n se rajoute à la complexité lorsqu'on regarde si le cercle circonscrit couvre bien tous les points ce qui nous $O(n^4)$.

Théorème 1 *le problème du cercle minimum peut être résolu en temps $O(n^4)$*

5 Algorithme de Welzl

Le problème du cercle minimum avait été abordé par Emo Welzl [6] un informaticien connu pour ses recherches en géométrie computationnelle et actuellement professeur à l'Institut d'informatique théorique de l'ETH Zurich en Suisse. En 1991 il publie un article intitulé "Smallest enclosing disks (balls and ellipsoids)" où il introduit l'algorithme de Welzl se basant sur l'algorithme de Raimund Seidel. L'algorithme de Welzl est un algorithme récursif, il résout le problème du cercle minimum en $O(n)$.

L'algorithme de Welzl [12] dont le pseudo-code est présenté ci-dessous dans l'Algorithme 3. Le principe est le suivant : Tout d'abord, on sélectionne trois points aléatoires pour former un triangle. Il crée son cercle circoncrit et vérifie si ce cercle englobe tous les autres points, alors l'algorithme le retourne.

Il vérifie par la suite si tous les autres points sont à l'intérieur du cercle circonscrit du triangle. Si c'est le cas, l'algorithme retourne ce cercle comme le cercle minimum. Sinon, il sélectionne un point à l'extérieur du cercle et répète le processus avec ce nouveau point ajouté au triangle.

Algorithm 3 Algorithme de Welzl

```
0: function B_MINIDISK(P, R) {returns b_md(P, R)}
0:   if P = ∅ or |R| = 3 then
0:     D ← b_md(P, R)
0:   else
0:     choose a random p ∈ P
0:     D ← B_MINIDISK(P − {p}, R)
0:     if D defined and p ∉ D then
0:       D ← B_MINIDISK(P − {p}, R ∪ {p})
0:     end if
0:   end if
0:   return D
0: end function
0: function WELZL(P) {returns md(P)}
0:   return B_MINIDISK(P, ∅)
0: end function=0
```

Cette approche récursive associée à une approche probabiliste en choisissant un point au hasard et en le retirant temporairement de l'ensemble permet à Welzl de prouver que son algorithme résout le problème du cercle minimum avec une complexité linéaire en O(n), ce qui en fait théoriquement une solution plus efficace que l'algorithme naïf pour résoudre le problème du cercle minimum. Mais pour s'en assurer, il faut passer à l'expérimentation.

Théorème 2 *Welzl calcule le cercle minimum d'un ensemble de n points dans le plan en un temps moyen attendu en $O(n)$.*

6 Résultats Expérimentaux

Dans cette section, nous présentons les résultats de nos expérimentations. Cette étape est cruciale, car elle nous permettra de mettre à l'épreuve les algorithmes étudiés dans les sections 4 et 5. Les critères pour évaluer sont le temps d'exécution moyen des algorithmes, la stabilité, la robustesse et la précision. Ces tests, sauf mention contraire, seront réalisés avec le benchmark Varoumas qui represente 1663 fichiers de tests avec 256 points par fichier et, pour ce qui est des conditions des expérimentations, les tests ont été effectués sur un processeur AMD Ryzen 7 5800X 8-Core Processor cadencé à 3,50 GHz.
Tous les fichiers de tests seront mis en annexe du projet dans le dossier *CSV*.

6.1 Gain De Temps

Dans cette expérience, nous avons mesuré le temps d'exécution en ms de l'algorithmes naïf et celui de Welzl sur 1663 fichiers et avons calculé la moyenne de toutes les mesures. La figure 5 illustre les résultats de ce test.

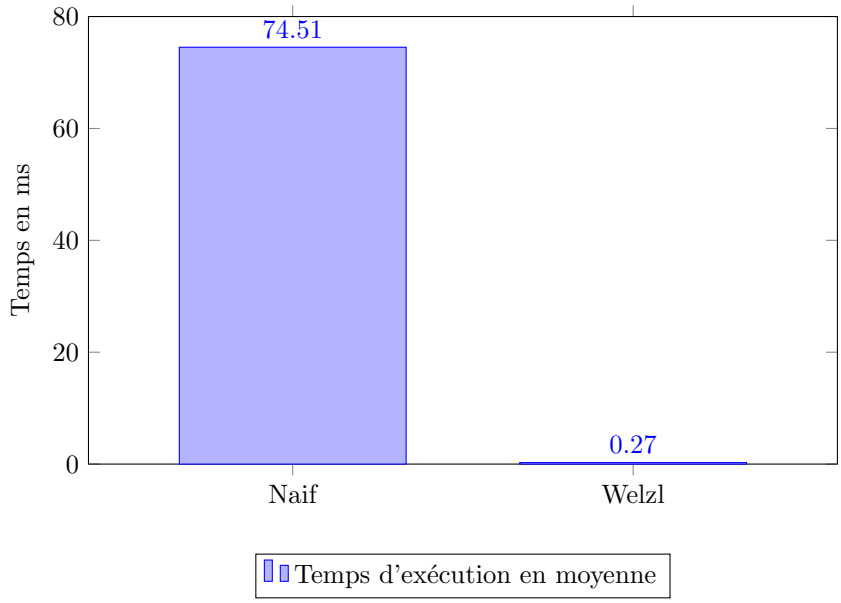

FIGURE 5 – Diagramme en bâtons représentant le gain de temps de l'implémentation de l'algorithme de Welzl par rapport à celle de l'algorithme naïf

6.1.1 Analyse des résulats

Ce diagramme en bâtons permet de constater l'algorithme naïf à une éxecution moyenne de 74.5 ms et que l'algorithme de Welzl à une éxecution moyenne de 0.27 ms sur le benchmark de Varoumas.
On a donc sur 256 points l'algorithme de Welzl serait en moyenne

$$\frac{74.5}{0.27} \approx 276$$

fois plus rapide que l'algorithme naif.

6.1.2 Conclusion de l'experience

Ce résultat est cohérent quand on sait que Welzl résout le problème du cercle en temps linéaire et que l'algorithme naïf le résout en temps $O(n^4)$. Avec l'algorithme de Welzl sur 256 points, on gagne donc un facteur d'environ 276 en moyenne par rapport à l'algorithme naïf.

6.2 Stabilité

Dans cette expérience, nous avons mesuré le temps d'exécution en ms de l'algorithme naïf et celui de Welzl sur 1663 fichiers dans un second temps nous avons calculé sa moyenne empirique et son écart type pour réaliser une une loi normale tronquée à $[0,+\infty[$ pour examiner la répartition des résultats.

Pour l'algorithme naïf, on a calculé une espérance de 47.51 ms et un écart type de 15 ms ce qui nous donne la gaussienne suivante :

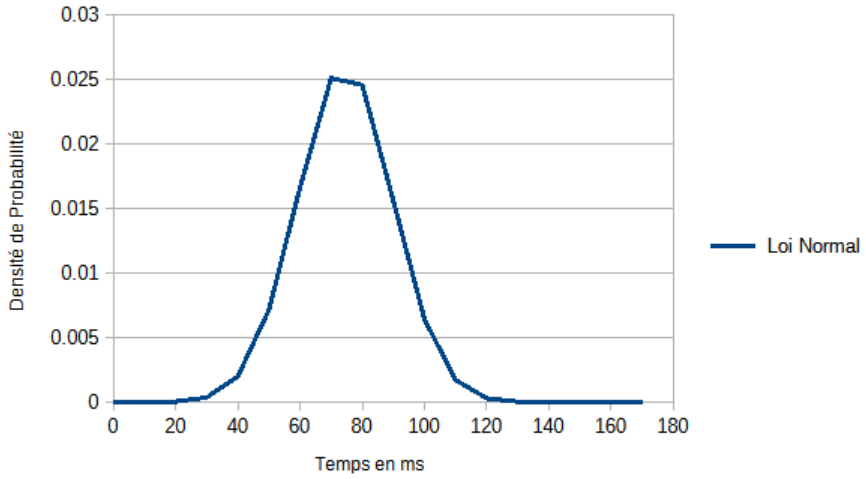

FIGURE 6 – Densité du temps d'exécution pour l'algorithme naïf

6.2.1 Analyse des résulats pour l'algorithme naïf

La courbe gaussienne indique que la majorité des temps d'exécution sont concentrés autour de la moyenne, qui est de 47.51 ms, avec une dispersion caractérisée par un écart-type de 15 ms. Pour interpreter les résultats ont utilises, la règle des trois sigma [8] qui nous dis que

$$P(\mu - 3\sigma \leq X \leq \mu + 3\sigma) >= 0.98$$

tel que μ est la moyenne, σ l'écart-type et X une valeur aléatoire continue qui représente le temps d'exécution. Ce qui signifie que 98% des temps d'exécution se situent entre $\mu - 3\sigma$ et $\mu + 3\sigma$.
on trouve donc sur la borne supérieur $\mu + 3\sigma$ = 47.51 + 3×15 =47.51 + 45 = 92.51ms, et pour la borne inférieure $\mu - 3\sigma$ = 47.51 - 3×15 =47.51 - 45 = 2.51ms.

Bien que l'algorithme affiche un temps d'exécution moyen raisonnable, l'analyse de l'application numérique couplé avec un écart-type relativement large, implique une dispersion considérable des temps d'exécution autour de la moyenne et suggère une certaine imprévisibilité des performances. Dans un contexte d'utilisation pratique, cette imprévisibilité pourrait affecter la fiabilité et la performance de l'algorithme.

Pour l'algorithme de Welzl, nous avons calculé une espérance de 0.27 ms et un écart-type de 0.17 ms, ce qui nous donne la courbe gaussienne suivante :

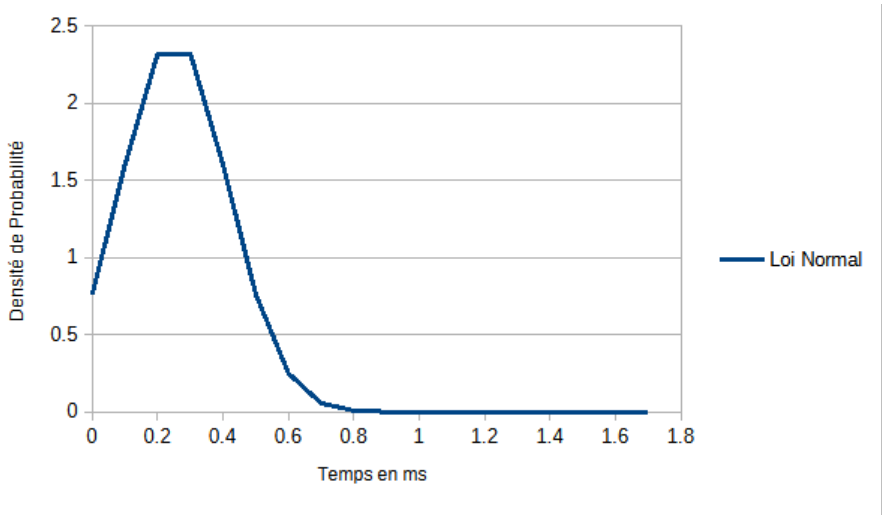

FIGURE 7 – Densité du temps d'exécution pour l'algorithme Welzl

6.2.2 Analyse des résulats pour l'algorithme Welzl

La courbe n'est pas exactement une gaussienne car les temps ne peuvent pas être négatifs ni nuls et donc les valeurs sur $[\mu - 3\sigma, 0]$ n'existent pas Ainsi, nous nous concentrons uniquement sur les valeurs dans l'intervalle $]0, \mu + 3\sigma]$.

En appliquant le même procedé, la figure 7 révèle que la majorité des temps d'exécution de l'algorithme de Welzl sont concentrés autour d'une moyenne de $\mu = 0.27\,\text{ms}$, avec une dispersion relativement faible, illustrée par un écart-type de $\sigma = 0.17\,\text{ms}$. En appliquant la règle des trois sigmas, mentionnée précédemment on trouve sur la borne supérieur $\mu + 3\sigma = 0.27{+}3{\times}0.17$ $=0.27{+}0.51 = 0.78\text{ms}$, et pour la borne inférieure une valeur très proche de 0.

Ces faibles résultats associé à un faible écart-type indique que les temps d'exécution de l'algorithme de Welzl fluctuent peu autour de la moyenne, ce qui témoigne d'une stabilité remarquable des performances. Dans un contexte d'utilisation concrète, cette régularité se traduit par une fiabilité et des performances optimales de l'algorithme.

6.2.3 Conclusion de l'experience

Bien que l'algorithme naïf puisse sembler stable à première vue, les analyses approfondies révèlent une réalité différente. Comme démontré précédemment, cet algorithme présente une volatilité considérable dans ses performances. Cette instabilité se manifeste par des variations importantes dans les temps d'exécution, ce qui peut entraîner une fiabilité moindre dans des situations où des temps de réponse constants sont essentiels. Ces fluctuations peuvent être particulièrement problématiques dans des applications en temps réel ou là où la précision et la prévisibilité sont cruciales.

À l'opposé, l'algorithme de Welzl démontre une grande stabilité. Les faibles variations dans les temps d'exécution, indiquées par un écart-type relativement mineur, suggèrent une performance plus prévisible et fiable. Cette stabilité est un atout majeur, surtout dans des contextes où des résultats rapides et constants sont nécessaires. En somme, l'algorithme de Welzl offre non seulement une efficacité accrue mais aussi une fiabilité supérieure par rapport à l'approche naïve.

6.3 Correction

Cette partie est dédiée à l'expérimentation de la validité de l'algorithme de Welzl. Nous supposons que l'algorithme naïf est valide pour notre référence grâce au lemme 2.
Nous calculerons le rayon pour chaque fichier du benchmark de Varoumas avec chaque algorithme et comparerons les résultats.

Pour ce test, nous observons la différence de rayon calculée par les algorithmes, ce qui nous donne cette distribution. Remarque : nous avons opté pour une représentation en distribution, en raison du grand nombre de valeurs extrêmes qui rendent d'autres représentations, telles qu'un diagramme en boîte ou une régression des moindres carrés, totalement illisibles.

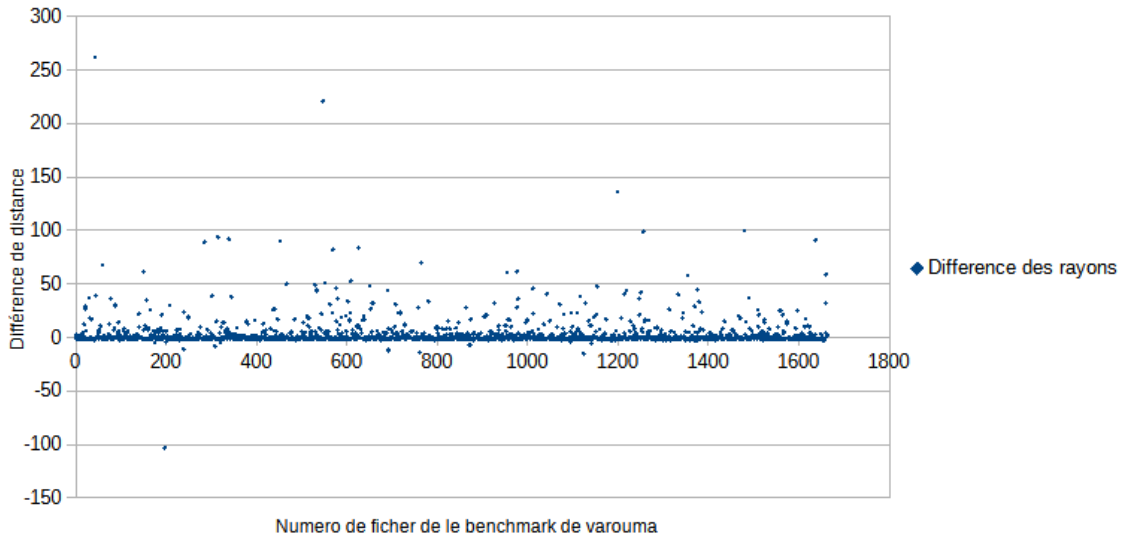

FIGURE 8 – Distribution des Écarts de Rayons entre Algorithmes sur le Benchmark Varoumas

6.3.1 Analyse des résulats

Dans la figure 8, nous pouvons observer que la majorité des points se regroupe autour du zéro, ce qui indique que, dans la plupart des cas, l'algorithme de Welzl fournit un résultat correct ou presque correct. Néanmoins, il apparaît également que certains points présentent une différence de rayon significative, révélant ainsi les limites de l'algorithme de Welzl dans certaines configurations de points.

6.3.2 Conclusion de l'experience

Ces variations mentionnées dans la partie analyse peuvent soulever des doutes quant à la validité de l'algorithme de Welzl et, par conséquent, à sa fiabilité en cas d'utilisation concrète. En effet, un trop grand nombre de valeurs extrêmes existe. Cependant, dans la mesure où, dans la majorité des cas, l'algorithme de Welzl donne des résultats acceptables, il reviendra à l'utilisateur de décider du compromis entre fiabilité et performance en fonction des usages.

6.4 Scalabilité

Dans cette section, les tests ont été effectués avec un benchmark personnalisé. Ce benchmark a été développé à l'aide d'un script Python, dont le code figure en annexe. L'objectif de ce script est d'augmenter progressivement le nombre de points par fichier selon la formule suivante : nombre de points = 50 + (numéro du fichier * 10). L'objectif de cette section est d'évaluer la robustesse de nos algorithmes face à l'augmentation progressive du nombre de points.

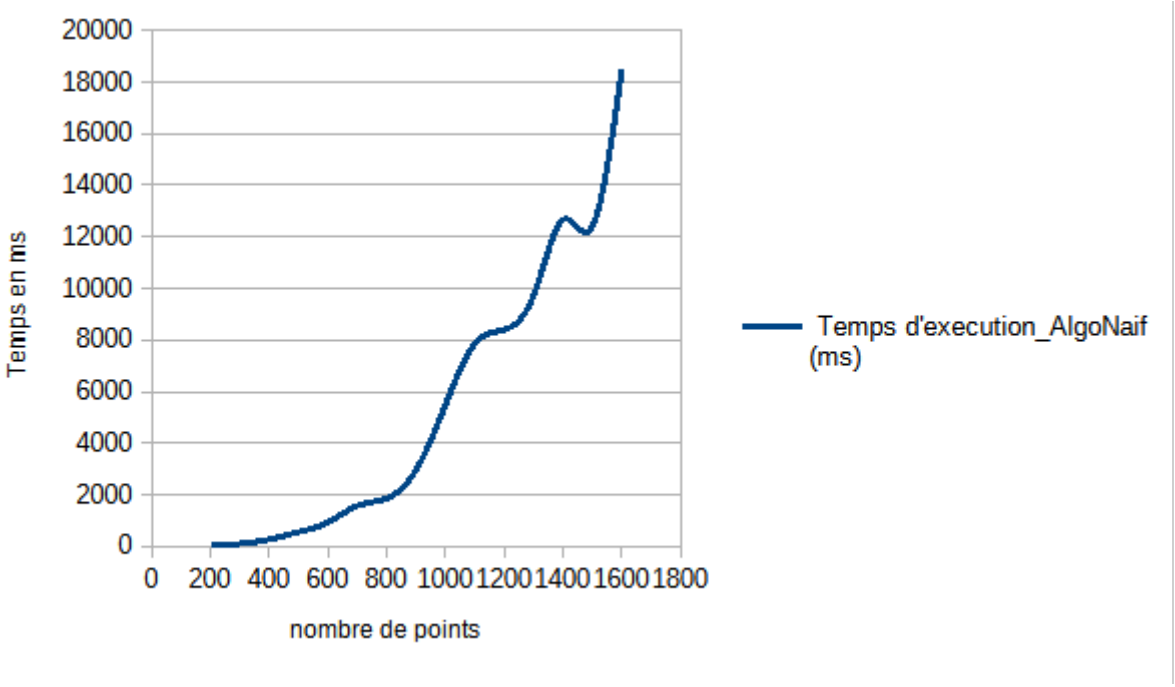

FIGURE 9 – Temps d'exécution de l'algorithme Naïf en fonction du nombre de points

6.4.1 Analyse des résulats pour l'algorithme naïf

Dans la figure 9, nous observons que le temps d'exécution de l'algorithme naïf augmente de manière exponentielle avec l'augmentation du nombre de points. Cette tendance confirme la complexité $O(n^4)$ de l'algorithme. Nous avons limité nos tests à 1800 points, car au-delà de cette valeur, le temps d'exécution de l'algorithme atteint des durées de l'ordre de la minute.

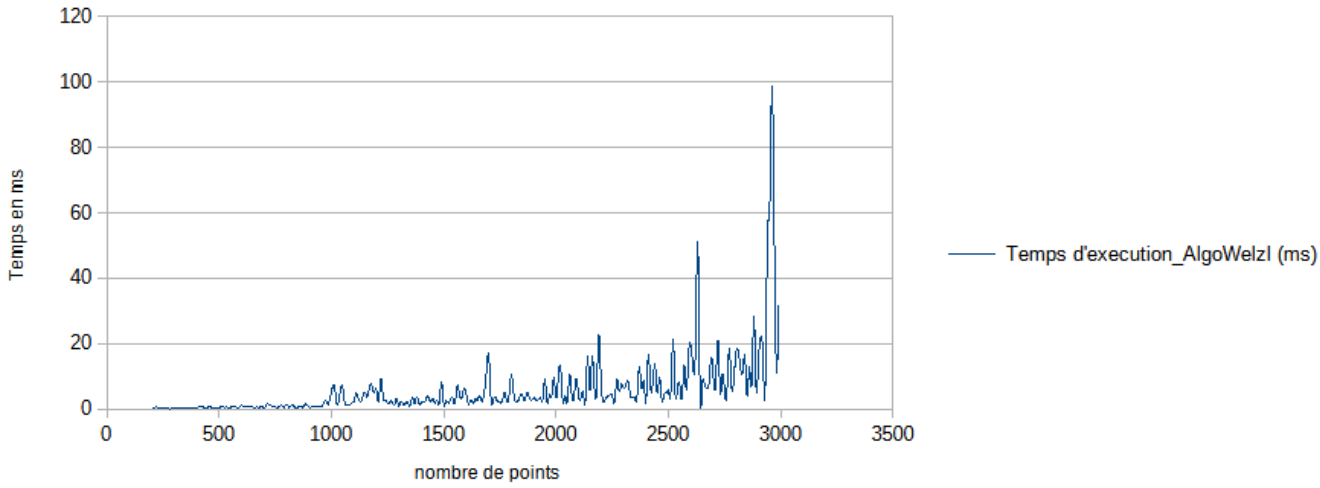

FIGURE 10 – Temps d'exécution de l'algorithme Welzl en fonction du nombre de points

6.4.2 Analyse des résulats pour l'algorithme Welzl

Dans la figure 10, nous observons que le temps d'exécution de l'algorithme de Welzl est linéaire et ne croit que légerement avec l'augmentation du nombre de points. Cette tendance confirme la complexité $O(n)$ de l'algorithme. Nous avons limité nos tests à 3000 points, car au-delà d'environ 3800 points, nous avons rencontré les limites de la version récursive de l'algorithme, menant à un dépassement de capacité de la pile (stackoverflow). Des solutions pour résoudre ce problème ont été trouvés par la communauté informatique tel que de le rendre itératif. sont disponibles en ligne, notamment en Python sur des forums comme Stack Overflow.[5]

6.4.3 Conclusion de l'experience

L'analyse de la performance sur un jeu de données croissant des deux algortihmes, révèle des tendances qui correspondent aux complexités théoriques attendues de chaque algorithme. Cependant l'algorithme naïf est inutilisable avec une liste de point trop grande, et l'implémentation récurive de l'algorithme de Welzl atteint une limite avec un grand nombre de point du à un dépassement de capacité de la pile. Cette contrainte technique suggère une modification vers une approche itérative.

7 Discussion

Nous avons étudié des algorithmes pour le problème du cercle minimum, au travers d'algorithmes randomisés et déterministes. Parmi ces dernières, les travaux de Seidel, Dyer et Frieze, ainsi que ceux de Clarkson, Megiddo et Skyum, ont été examinés. Cette analyse nous a permis d'apprécier la diversité et l'ingéniosité des méthodes utilisées dans la résolution de ce problème.

Dans notre étude de l'algorithme naïf et de celui de Welzl, nous avons constaté des différences significatives en termes d'efficacité. En effet Welzl, avec sa complexité linéaire et sa facilité d'implémentation, surpasse nettement l'algorithme naïf en termes de temps d'exécution moyen, de stabilité et de scalabilité.

Cependant, malgré sa supériorité globale, l'algorithme de Welzl a révélé certaines imprécisions concernant le cercle minimum. De plus, il présente un problème de débordement de pile (stack overflow) qui pourrait remettre en question sa pertinence. Toutefois, dans la majorité des cas, l'écart de précision est acceptable selon l'usage de l'utilisateur et en modifiant l'algorithme pour le rendre itératif, il est possible de pallier le problème de stack overflow.

En conclusion, ce rapport contribue à une meilleure compréhension des algorithmes existants pour résoudre le problème du cercle minimum et souligne l'intérêt de développer des approches combinant la précision des algorithmes déterministes à l'efficacité des méthodes randomisées, surtout pour traiter des ensembles de données de plus en plus volumineux et complexes.

En ce qui concerne des notions similaire en 3D ce problème devient alors de trouver la sphère minimum, ce qui implique des adapations dans les algorithmes déjà connu ce qui peut acroite la compléxité. De plus la visualisation et l'analyse des résultats sont plus difficiles en 3D. Cela peut nécessiter des outils de visualisation avancés pour interpréter correctement les données et les résultats.

Références

[1] *Calcul de l'enveloppe convexe*. fr. Page Version ID : 212522853. Fév. 2024. URL : https://fr.wikipedia.org/w/index.php?title=Calcul_de_l%27enveloppe_convexe&oldid=212522853 (visité le 17/03/2024).

[2] *Cercle circonscrit à un triangle*. fr. Page Version ID : 211191147. Jan. 2024. URL : https://fr.wikipedia.org/w/index.php?title=Cercle_circonscrit_%C3%A0_un_triangle&oldid=211191147 (visité le 16/03/2024).

[3] *Diagramme de Voronoï*. fr. Page Version ID : 208865937. Oct. 2023. URL : https://fr.wikipedia.org/w/index.php?title=Diagramme_de_Vorono%C3%AF&oldid=208865937 (visité le 17/03/2024).

[4] *Distance*. URL : https://www.bibmath.net/dico/index.php?action=affiche&quoi=./d/distance.html (visité le 16/03/2024).

[5] David Eisenstat. *Answer to "Iterative version of Welzl's algorithm"*. Oct. 2021. URL : https://stackoverflow.com/a/69430104 (visité le 16/03/2024).

[6] *Emo Welzl*. en. Page Version ID : 1212966461. Mars 2024. URL : https://en.wikipedia.org/w/index.php?title=Emo_Welzl&oldid=1212966461 (visité le 14/03/2024).

[7] Nimrod Megiddo. *Linear-Time Algorithms for Linear Programming in $R^3 $ and Related Problems*. en. DOI : 10.1137/0212052. URL : https://epubs-siam-org.accesdistant.sorbonne-universite.fr/doi/epdf/10.1137/0212052 (visité le 12/03/2024).

[8] *Règle 68-95-99,7*. fr. Page Version ID : 213268727. Mars 2024. URL : https://fr.wikipedia.org/w/index.php?title=R%C3%A8gle_68-95-99,7&oldid=213268727 (visité le 17/03/2024).

[9] Raimund Seidel. "Small-dimensional linear programming and convex hulls made easy". en. In : *Discrete & Computational Geometry* 6.3 (sept. 1991), p. 423-434. ISSN : 1432-0444. DOI : 10.1007/BF02574699. URL : https://doi.org/10.1007/BF02574699 (visité le 11/03/2024).

[10] Sven Skyum. "A simple algorithm for computing the smallest enclosing circle". In : *Information Processing Letters* 37.3 (fév. 1991), p. 121-125. ISSN : 0020-0190. DOI : 10.1016/0020-0190(91)90030-L. URL : https://www.sciencedirect.com/science/article/pii/002001909190030L (visité le 12/03/2024).

[11] James Joseph Sylvester. *Problème du cercle minimum*. fr. Page Version ID : 208545755. Oct. 2023. URL : https://fr.wikipedia.org/w/index.php?title=Probl%C3%A8me_du_cercle_minimum&oldid=208545755 (visité le 16/03/2024).

[12] Emo Welzl. "Smallest enclosing disks (balls and ellipsoids)". en. In : *New Results and New Trends in Computer Science*. Sous la dir. d'Hermann Maurer. Berlin, Heidelberg : Springer, 1991, p. 359-370. ISBN : 978-3-540-46457-0. DOI : 10.1007/BFb0038202.