Wyznaczanie minimalnego okręgu i prostokąta zawierającego chmurę punktów na płaszczyźnie

Aga Patro

Opis problemu

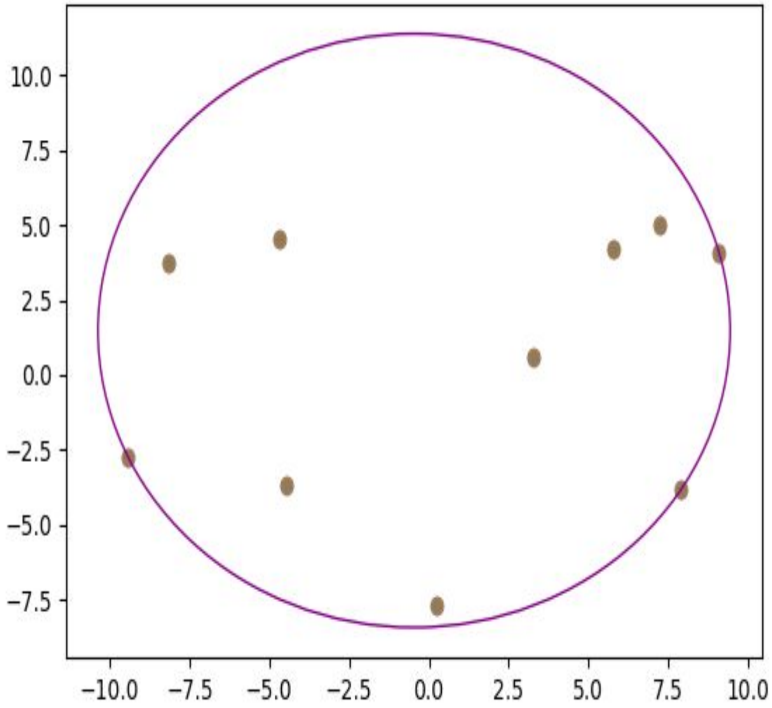
Zadawana jest chmura punktów na płaszczyźnie dwuwymiarowej. Program ma wyznaczać:

– minimalny okrąg zawierający tę chmurę,

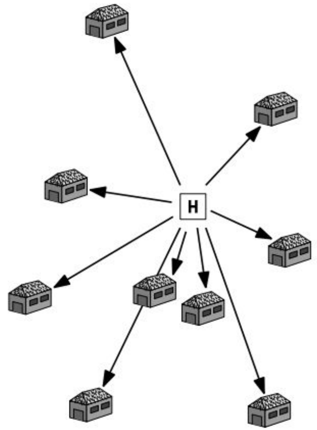
– prostokąt o minimalnym polu powierzchni zawierający tę chmurę.

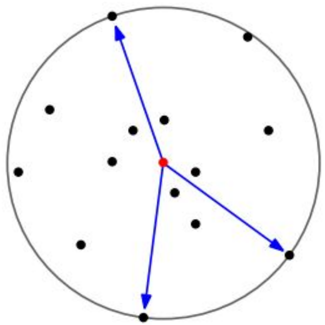
– prostokąt o minimalnym obwodzie zawierający tę chmurę.

Wyznaczanie minimalnego okręgu - zastosowanie w praktyce

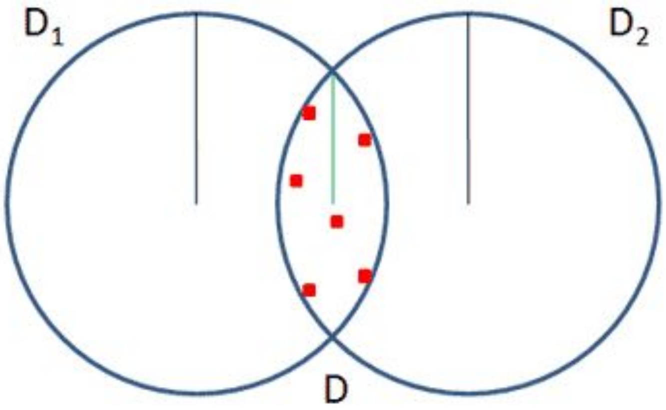


- lokalizacja wspólnego obiektu - np. znalezienie najmniejszego okalającego okręgu ułatwia nam umiejscowienie np. szpitala lub anteny
- w wojsku ten problem jest znany jako problem bomby - jeśli potraktujemy punkty jako cele na mapie to środek będzie dobrym miejscem do zrzucenia bomby
- w zbiorach danych punkty na granicy okręgu okalającego są często w pewnym sensie odstające od zbioru, przez co są odrzucane - wykorzystuje się to w statystyce aby oszacowania były bardziej niezawodne





Algorytm Welzla



Jest to algorytm rekurencyjny. Na wejściu przyjmuje listę punktów. Punkty muszą być unil

By lepiej zrozumieć, załóżmy, że znamy już najmniejszy okrąg (NO) D dla n-1 punktów.

Teraz są dwa przypadki dla n-tego punktu:

1) Pn leży wewnątrz D. Więc nic się nie zmienia - odpowiedź to D
2) Pn nie leży wewnątrz D. Musimy wyznaczyć nowy NO, z tym że musi on leżeć na granicy D, nazwijmy to BD. Musimy więc wyznaczyć NO D' dla p1,...,pn-1 z pn na granicy D'.

To jest główna idea. Ta właściwość wraz z trzema następującymi twierdzeniami pozwala nam obliczyć taki najmniejszy otaczający dysk w sposób iteracyjny:

Niech P znowu będzie zbiorem n punktów, P nie jest puste i p jest punktem w PR. R jest również zbiorem punktów, w rzeczywistości są to punkty na granicy okręgu. Następnie, lemat mówi :

1. Jeśli istnieje koło zawierające P i ma wszystkie punkty z R na jego brzegu, to D(P,R) jest wyznaczone jednoznacznie.

2. Jeżeli p nie leży w D(P - {p},R), to p leży na granicy D(P,R), pod warunkiem, że istnieje, czyli: D(P,R) = D(P - {p}, R u {p}).

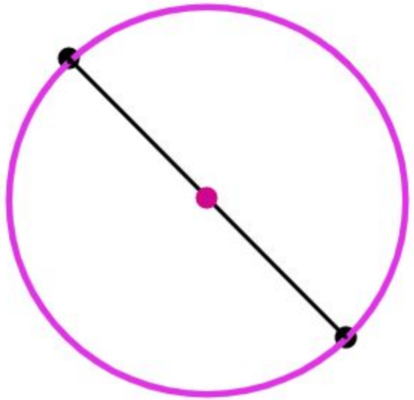
3. Jeżeli D(P,R) istnieje, to istnieje zbiór S złożony z większości max{0,3 - |R|} punktów w P takich, że D(P,R) = D(S,R). Oznacza to, że P jest określone przez co najwyżej 3   punkty w P, które leżą na granicy D(P).

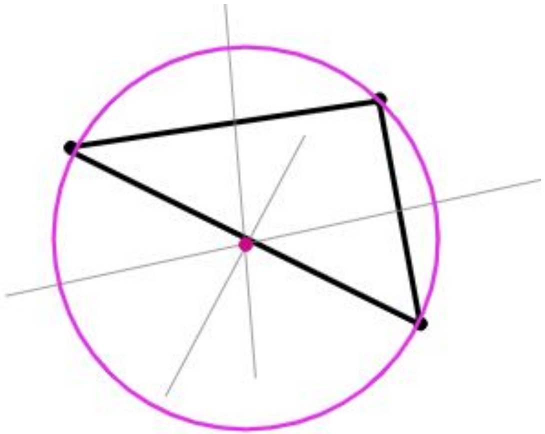
Algorytm działa w czasie O(n)

Algorytm zawiera następujące kroki:

1. Losowo wybieramy jeden punkt z P i rekurencyjnie znajduje najmniejszy okrąg zawierający P - {p}, czyli wszystkie pozostałe punkty oprócz w P oprócz p.
2. Jeśli zwrócony okrąg zawiera również p, to jest to minimalny okrąg dla całego P i jest zwracany.
3. Jeśli zwrócony okrąg nie zawiera p, to punkt p musi leżeć na granicy okręgu wynikowego, w zbiorze R (które znajdują się na brzegu okręgu)
4. Rekurencja kończy się, gdy P jest puste a rozwiązanie można znaleźć za pomocą punktów w R:

- dla 0 lub 1 -> nie ma takiego okręgu lub ma środek w tym jedynym punkcie
- dla 2 -> minimalny okrąg ma środek w punkcie środkowym pomiędzy punktami, a jego promień to odległość od środka do danych punktów
- dla 3 -> jest to okrąg opisany na trójkącie





Bibliografia

https://en.wikipedia.org/wiki/Smallest-circle_problem

https://www.geeksforgeeks.org/minimum-enclosing-circle-using-welzls-algorithm

https://www.gamedev.net/tutorials/programming/graphics/welzl-r2484/

http://www.sunshine2k.de/coding/java/Welzl/Welzl.html

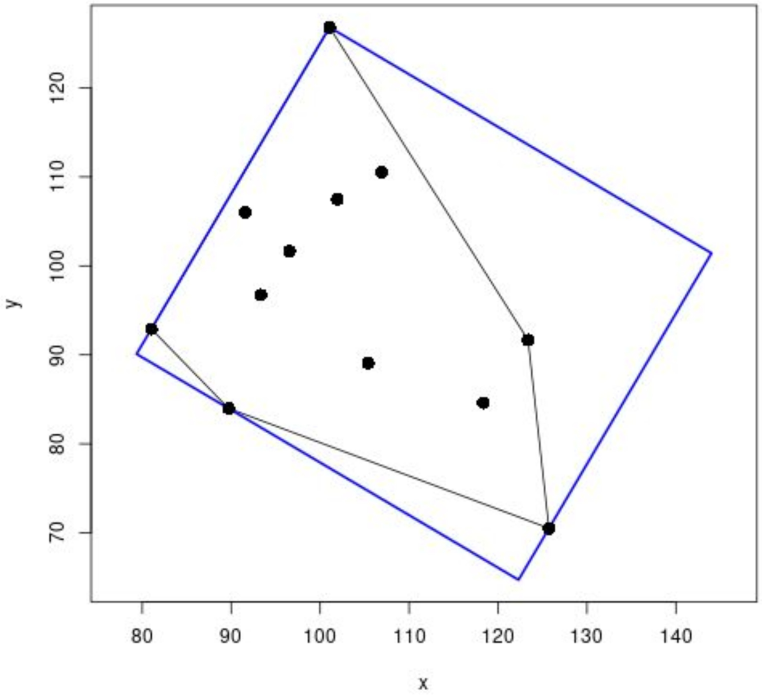
https://github.com/uhuaha/smallestCircle

https://www.cs.mcgill.ca/~cs507/projects/1998/jacob/problem.html

https://www.nayuki.io/res/smallest-enclosing-circle/computational-geometry-lecture-6.pdf

de Berg Mark, "Computational Geometry - Algorithms and Applications", edycja trzecia

Wyznaczanie minimalnych prostokątów - zastosowanie w praktyce



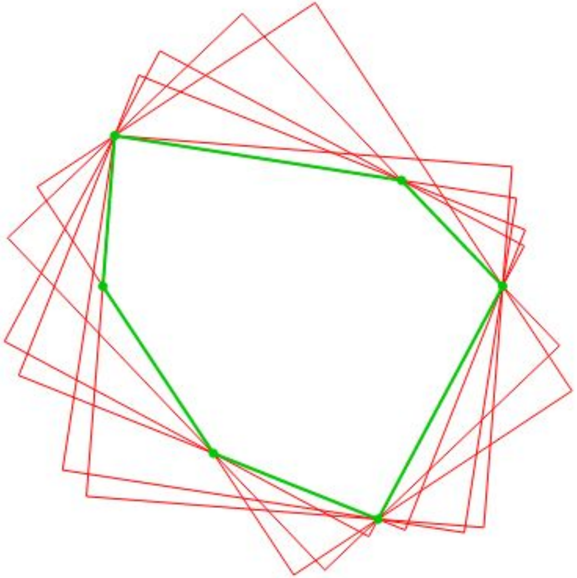
- lokalizacja obiektów na potrzeby jazdy pojazdami autonomicznymi - ramki ograniczające są zwykle używane w szkoleniu modeli wizyjnych samojezdnych samochodów
- wykrywanie przedmiotów w pomieszczeniach
- w grach wideo minimalne pola okalające mogą być używane do określania, czy dwa obiekty kolidują, sprawdzając, czy ich pola obwiedni przecinają się

Algorytm oparty o rotating_calipers

Aby rozwiązać ten problem, wyznaczyłam wszystkie okalające prostokąty, a następnie z nich wyznaczyłam interesujące mnie minimalne prostokąty (o najmniejszym polu i najmniejszym obwodzie)

Algorytm zawiera w sobie następujące kroki:

1. Wyznacz otoczkę wypukłą
2. Skonstruuj dwie pionowe linie w punktach x_min i x_max oraz dwie poziome linie w punktach y_min i y_max
3. Obracaj liniami aż jedna pokryje się z bokiem otoczki wypukłej. Dodaj prostokąt do listy wszystkich prostokątów.
4. Powtarzaj punkt 3 aż wszystkie boki otoczki wypukłej zostaną 'pokryte' przez rotują
5. Z otrzymanej listy prostokątów wyciągnij ten który się interesuje tj. z najmniejszym obwodem lub najmniejszym polem.



Bibliografia

https://en.wikipedia.org/wiki/Rotating_calipers

https://geidav.wordpress.com/tag/rotating-calipers/

https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.ConvexHull.html

https://chadrick-kwag.net/python-implementation-of-rotating-caliper-algorithm/

https://www.statology.org/matplotlib-rectangle/

https://matplotlib.org/stable/api/_as_gen/matplotlib.patches.Rectangle.html

http://dwoll.de/rexrepos/posts/diagBounding.html

https://hypersense.subex.com/aiglossary/bounding-box/

https://openai.com/blog/chatgpt/