Algorytmy Geometryczne Lab2 - Otoczka wypukła

Nieć Witold

Spis treści

1. Dane Podstawowe

Nazwisko, Imię: Nieć, Witold
<removed>

2. Dane Techniczne

Język: Python 3.12.0
Procesor: Intel core i7-8750H
System operacyjny: Windows 10
RAM: 8GB
Środowisko: Jupyter Notebook

3. Realizacja ćwiczenia

3.1. Testowane Algorytmy

3.1.1. Algorytm Grahama

Działanie algorytmu Grahama rozpoczyna się od znalezienia punktu o najmniejszej współrzędnej y. W przypadku, gdy kilka punktów ma identyczną wartość y, wybierany jest ten o najmniejszej współrzędnej x. Punkt ten jest oznaczany jako punkt główny. Algorytm działa poprzez posortowanie punktów według kąta(jeśli kąt jest taki sam to punkty sortowane są według odległości), jaki tworzy prosta przechodząca przez punkt główny oraz rozważany punkt z dodatnią półosią osi OX. Następnie punkty są kolejno dodawane na stos otoczki wypukłej w kolejności wyznaczonej przez sortowanie. Jeżeli dodawany punkt leży na lewo od prostej wyznaczanej przez dwa ostatnie punkty ze stosu otoczki wypukłej, usuwamy po kolei punkty ze stosu, aż otoczka ponownie stanie się wypukła. Proces ten trwa do momentu przeanalizowania wszystkich punktów. Wynikiem algorytmu jest lista punktów tworzących otoczkę wypukłą.

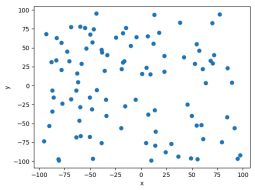
3.1.2. Algorytm Jarvisa

Podobnie jak dla algorytmu Grahama, proces ponownie rozpoczyna się od znalezienia punktu o najmniejszej współrzędnej y, w przypadku takich samych wartości y wybierany jest ten o niższej współrzędnej x. Algorytm Jarvisa polega na iteracyjnym wyznaczaniu kolejnych punktów otoczki wypukłej. Każdy nowy punkt wybierany jest jako ten, który tworzy najmniejszy kąt względem ostatniej krawędzi otoczki (jeśli kąt dla dwóch punktów jest taki sam to wybieramy ten punkt który jest najbardziej oddalony od ostatniego punktu otoczki). Algorytm rozpoczyna się od znalezienia punktu o najmniejszej współrzędnej y (a w przypadku remisu - najmniejszej współrzędnej x). Następnie wybierany jest pierwszy punkt pod kątem względem dodatniej półosi OX. Proces powtarza się do momentu, gdy kolejnym wybranym punktem stanie się punkt początkowy, co oznacza zamknięcie otoczki i zakończenie algorytmu.

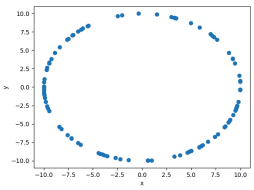
3.2. Rozważane zbiory punktów

- Zbiór A - zawierający 100 losowo wygenerowanych punktów o współrzędnych z przedziału $[-100, 100]$,
- Zbiór B - zawierający 100 losowo wygenerowanych punktów leżących na okręgu o środku (0,0) i promieniu R=10,
- Zbiór C - zawierający 100 losowo wygenerowanych punktów leżących na bokach prostokąta o wierzchołkach (-10, 10), (-10,-10), (10,-10), (10,10),
- Zbiór D - zawierający wierzchołki kwadratu (0, 0), (10, 0), (10, 10), (0, 10) oraz punkty wygenerowane losowo w sposób następujący: po 25 punktów na dwóch bokach kwadratu leżących na osiach i po 20 punktów na przekątnych kwadratu.

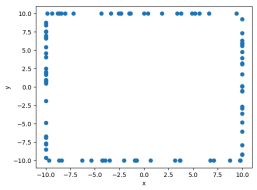
Do generowania punktów użyto biblioteki „random” oraz funkcji „random.uniform()”



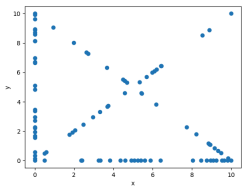
Rys.1.1: zbiór A



Rys. 1.2: zbiór B



Rys. 1.3: zbiór C



Rys. 1.4: zbiór D

3.3. Sposób klasyfikacji punktów

Do sortowania po kącie czy znajdowania punktu tworzącego najmniejszy kąt wykorzystano wyznacznik macierzy 2x2 własnej implementacji z poprzedniego laboratorium. W algorytmie Grahama stosowany jest on jako klucz porównania dla dwóch punktów przy sortowaniu. Dla algorytmu Jarvisa wyznacznik wykorzystywany jest przy znajdowaniu punktu, dla którego kąt względem ostatniej krawędzi otoczki jest najmniejszy, stosując wyznacznik również jako klucz porównań przy iteracji przez wszystkie punkty.

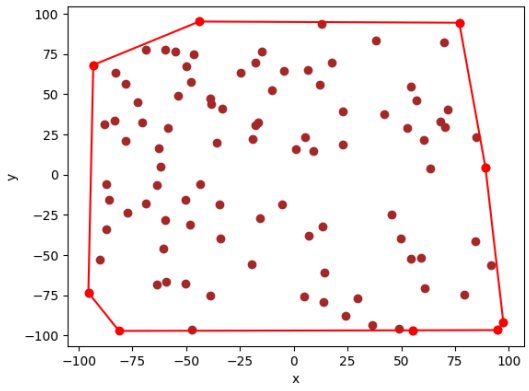
4. Analiza działania algorytmów dla podstawowych zbiorów

4.1. Algorytm Grahama

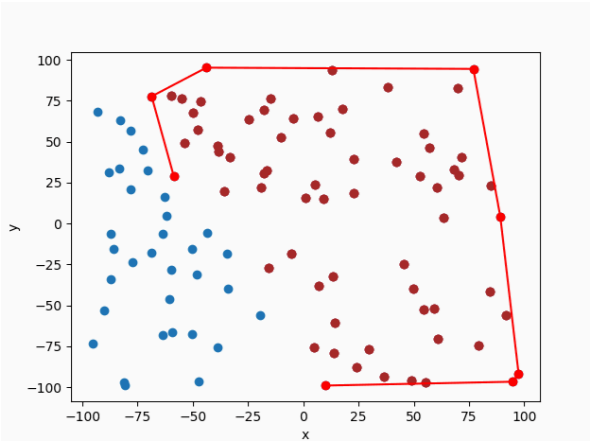
Na wykresach przedstawiających działanie algorytmu Grahama zastosowano następującą konfigurację oznaczeń:

- kolor niebieski - punkty nieprzetworzone przez algorytm
- kolor brązowy - punkty przetworzone przez algorytm które nie należą do otoczki
- kolor czerwony - punkty przetworzone przez algorytm które należą do otoczki oraz krawędzie otoczki

4.1.1. Zbiór A



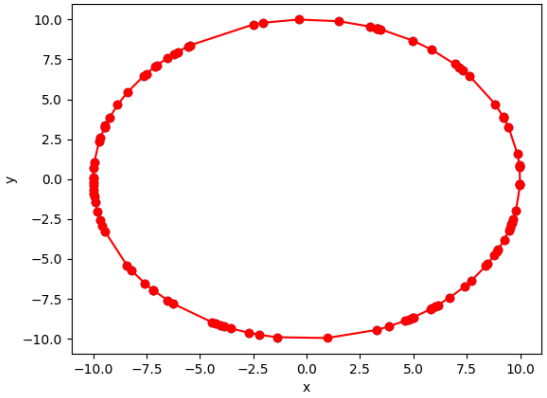
Rys 2.1 - Działania algorytmu Grahama zbiór A



Rys 2.2 - W trakcie działania algorytmu Grahama zbiór A

Dla zbioru A algorytm Grahama wydaje się działać poprawnie. Po posortowaniu wystarczy raz liniowo przejść przez pozostałe wierzchołki aby uzyskać oczekiwany wynik.

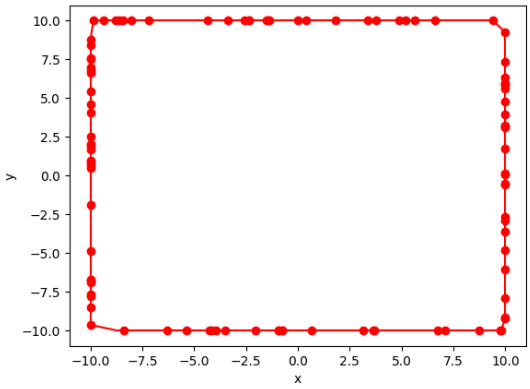
4.1.2. Zbiór B



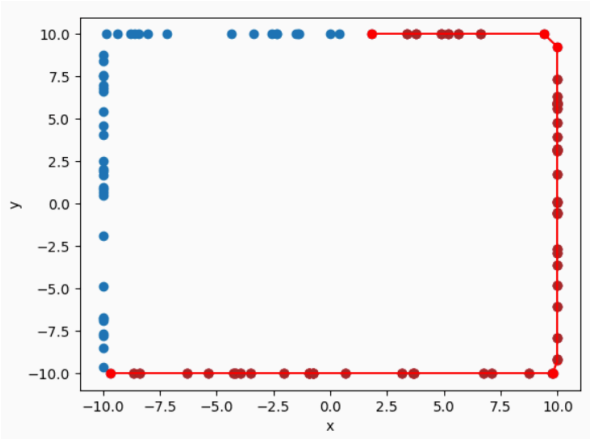
Rys 3 - Efekt działania algorytmu Grahama zbiór B

W tym przypadku gdy z założenia wszystkie punkty należą do otoczki, sortowanie po kącie wydaje się być wydajnym rozwiązaniem. Konieczne przetworzenie wszystkich punktów nie szkodzi znacznie szybkości algorytmu ponieważ żadna krawędź dodana do stosu nie będzie usuwana.

4.1.3. Zbiór C



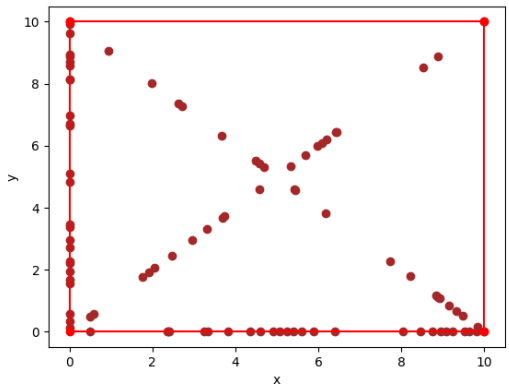
Rys 4.1 - Efekt działania algorytmu Grahama zbiór C



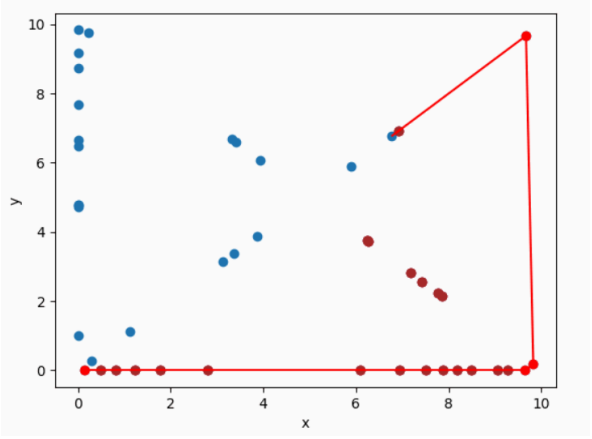
Rys 4.2 - W trakcie działania algorytmu Grahama zbiór C

Natomiast w przypadku gdy wszystkie punkty należą do otoczki oraz dzielą się na grupy leżące na tej samej prostej, wydajność algorytmu nie jest już taka zachwycająca, choć nadal niezła. Ponieważ w wyniku działania algorytmu chcemy uzyskać otoczkę wypukłą ale z pominięciem punktów współliniowych, musimy je odrzucić. Cechą algorytmu Grahama jest iteracja po wszystkich punktach. Zanim algorytm znajdzie rozwiązanie będzie musiał dodać i odrzucić znaczną większość punktów ze stosu.

4.1.4. Zbiór D



Rys 5.1 - Efekt działania algorytmu Grahama zbiór D



Rys 5.2 - Efekt działania algorytmu Grahama zbiór D

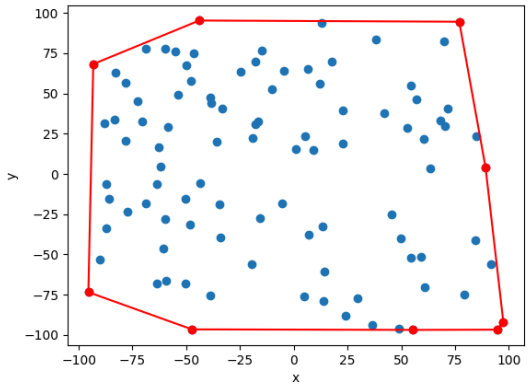
Mała niedoskonałość z poprzedniego przykładu jest jeszcze lepiej uwydatniona dla zbioru D. Teraz wymóg iteracji przez wszystkie punkty zmusza nas do przetworzenia wszystkich punktów, w szczególności tych na przekątnych kwadratu, które nie mają szans na bycie częścią otoczki.

4.2. Algorytm Jarvisa

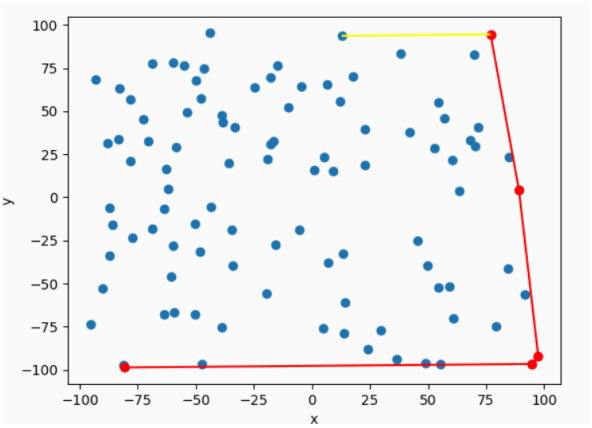
Na wykresach przedstawiających działanie algorytmu Grahama zastosowano następującą konfigurację oznaczeń:

- kolor niebieski - punkty nienależące do otoczki
- kolor czerwony - punkty należące do otoczki i krawędzie otoczki
- kolor żółty - tymczasowy wierzchołek leżący pod najmniejszym kątem względem ostatniej krawędzi otoczki, potencjalnie nowa krawędź otoczki.

4.2.1. Zbiór A



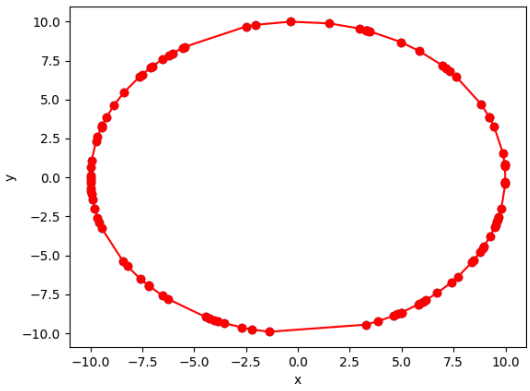
Rys 6.1 - Efekt działania algorytmu Jarvisa zbiór A



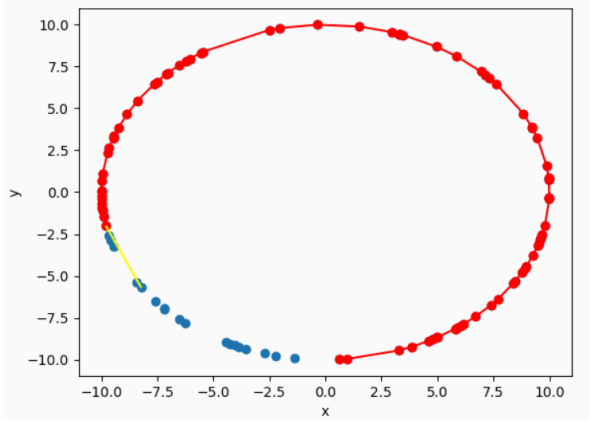
Rys 6.2 - W trakcie działania algorytmu Jarvisa zbiór A

Zaletą algorytmu Jarvisa w stosunku do algorytmu Grahama jest możliwość uzyskania złożoności. W przypadku A natomiast liczba wierzchołków należących do otoczki jest porównywalna z log(n), zatem oba algorytmy będą działać podobnie.

4.2.2. Zbiór B



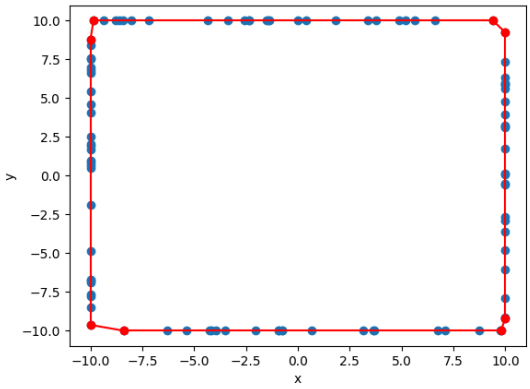
Rys 7.1 - Efekt działania algorytmu Jarvisa zbiór B



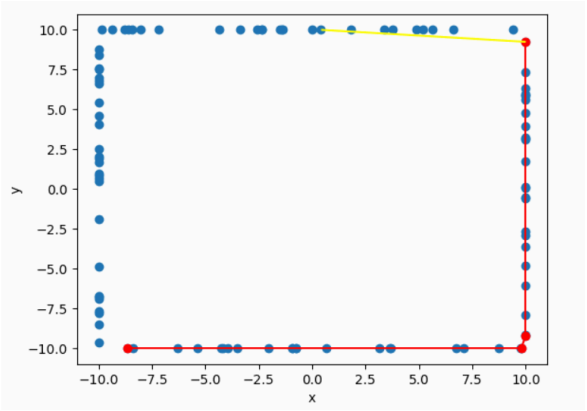
Rys 7.2 - W trakcie działania algorytmu Jarvisa zbiór B

W przypadku zbioru B gdy wszystkie punkty należeć będą do otoczki złożoność algorytmu Jarvisa wynosić będzie O(n^2) co nie jest optymalne w zestawieniu z algorytmem Grahama którego złożoność wynosi stale O(n log n).

4.2.3. Zbiór C



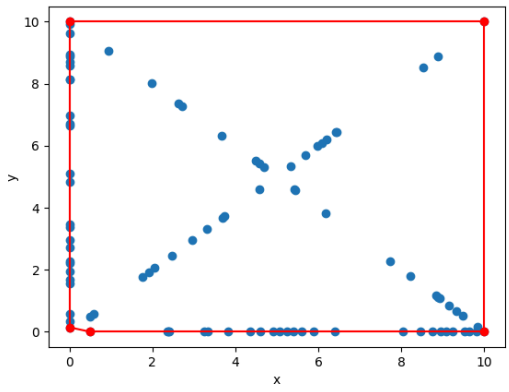
Rys 8.1 - Efekt działania algorytmu Jarvisa zbiór C



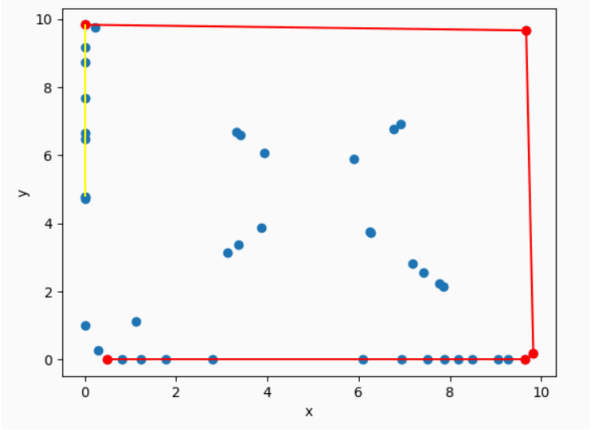
Rys 8.2 - W trakcie działania algorytmu Jarvisa zbiór C

Zbiór C natomiast przez współliniowość wierzchołków, otoczka redukuje się do maksymalnie 8 wierzchołków. Dzięki temu złożoność Jarvisa dla zbioru C jest pseudo liniowa.

4.2.4. Zbiór D



Rys 9.1 - Efekt działania algorytmu Jarvisa zbiór D



Rys 9.2 - W trakcie działania algorytmu Jarvisa zbiór D

Dla zbioru D podobnie jak dla zbioru C dzięki małej liczbie punktów należących do otoczki udaje się uzyskać złożoność liniową. Ponieważ szukamy punktu którego prosta tworzy najmniejszy kąt tyle razy ile wynosi liczność otoczki, redukujemy liczbę iteracji po wszystkich wierzchołkach do k.

5. Analiza dla zbiorów zmodyfikowanych i porównanie wydajności

Głównym czynnikiem rozróżniającym algorytmy Grahama i Jarvisa jest złożoność obliczeniowa. Dla Grahama jest to stałe O($n \log(n)$), gdzie n - liczba punktów w zbiorze. Dla Jarvisa złożoność zależy od liczby punktów należących do otoczki: O(nk), n - liczba punktów w zbiorze

Tabela 1: Porównanie złożoności algorytmów

	Graham	Jarvis
złożoność	$n \log n$	nk
złożoność optymistyczna	$n \log n$	$\sim n$
złożoność pesymistyczna	$n \log n$	n^2

5.1. Zbiór A

Tabela 2: Porównanie wydajności czasowej algorytmów dla zbioru A

Liczba punktów	Czas dla Algorytmu Grahama	Czas dla algorytmu Jarvisa
1000	0.0080 s	0.0189 s
10000	0.0988 s	0.0828 s
100000	0.9161 s	0.8733 s
1000000	11.8352 s	15.0499 s
1000000	168.8834 s	141.7804 s

Dla zbioru A, lepiej radzi sobie algorytm Jarvisa ale wyniki są bardzo zbliżone. Dzieje się tak ponieważ liczba punktów otoczki jest porównywalna z $\log n$.

5.2. Zbiór B

Tabela 3: Porównanie wydajności czasowej algorytmów dla zbioru B

Liczba punktów	Czas dla Algorytmu Grahama	Czas dla algorytmu Jarvisa
50	0.0000 s	0.0010 s
250	0.0010 s	0.0249 s
1250	0.0070 s	0.5157 s
6250	0.0359 s	0.0359 s
31250	0.2470 s	140.8679 s

Dla zbioru B zdecydowanie lepszą wydajnością cechuje się Algorytm Grahama. Ze względu na to że wszystkie punkty należące do zbioru należą do otoczki złożoność Jarvisa zamienia się w O(n^2), podczas gdy Graham utrzymuje stałą złożoność O($n \cdot \log(n)$). Wyniki w tabeli ukazują tą dysproporcję w złożoności czasowej algorytmu. Wraz ze wzrostem liczby punktów wyższość Grahama nad Jarvisem jest coraz większa.

5.3. Zbiór C

Tabela 4: Porównanie wydajności czasowej algorytmów dla zbioru C

Liczba punktów	Czas dla Algorytmu Grahama	Czas dla algorytmu Jarvisa
100	0.0004 s	0.0010 s
1000	0.0070 s	0.0041 s
10000	0.0848 s	0.0339 s
100000	0.8838 s	0.3401 s
1000000	11.9684 s	3.9027 s

Mimo iż dla zbioru C wszystkie punkty znajdują się na krawędziach otoczki, to przez to że są one współliniowe liczba punktów otoczki redukuje się do co najwyżej 8. Dzięki temu złożoność asymptotyczna dla algorytmu Jarvisa wynosi O(n), co w porównaniu ze złożonością Grahama - (n log n), daje lepsze wyniki dla takiego zbioru punktów.

5.4. Zbiór D

Tabela 5: Porównanie wydajności czasowej algorytmów dla zbioru D

Liczba punktów	Czas dla Algorytmu Grahama	Czas dla algorytmu Jarvisa
250, 100	0.0050 s	0.0010 s
2500, 10000	0.0728 s	0.0180 s
25000, 100000	0.7141 s	0.1725 s
250000, 1000000	9.5642 s	2.2576 s
2500000, 10000000	145.0205 s	21.1210 s

Jak widać w Tabeli 5. dla zbioru D, Algorytm Jarvisa uzyskuje znacznie lepsze czasy działania do algorytmu Grahama. Dzięki temu że, przy nieznacząco małej liczbie wierzchołków należących do otoczki złożoność Jarvisa jest liniowa. Wraz ze wzrostem liczby punktów w zbiorze obserwujemy dominację złożoności liniowej nad O(n log(n)).

6. Wnioski

Algorytm Grahama cechuje złożoność O(nlogn), co czyni go bardziej wydajnym w przypadku dużych zbiorów punktów o zróżnicowanej strukturze.
Algorytm Jarvisa, ze złożonością O(nk) (gdzie k to liczba punktów otoczki), jest bardziej efektywny, gdy liczba punktów tworzących otoczkę jest stosunkowo mała w porównaniu do całkowitej liczby punktów.
Oba algorytmy mają swoje zalety i wady w zależności od wybranego zbioru punktów.
Algorytm Grahama może być preferowany w ogólnych przypadkach, gdzie złożoność danych jest wysoka. Algorytm Jarvisa jest lepszy w szczególnych przypadkach, gdy dane są zorganizowane w sposób, który minimalizuje liczbę punktów otoczki.