R Project

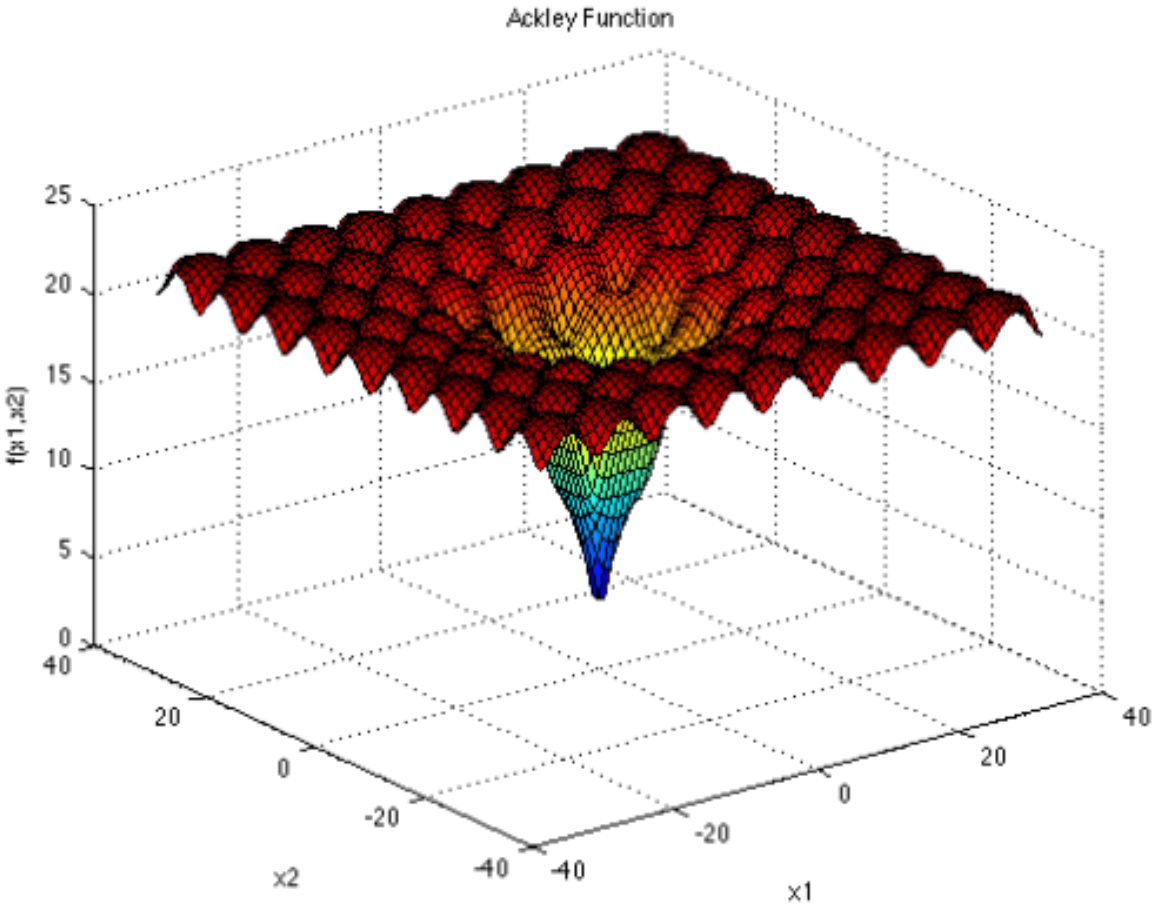
Mikołaj Pajor, Kacper Ćwiertnia

Poniższy projekt przedstawia analizę statystyczną porównania algorytmów minimalizacji stochastycznej. Porównywane przez nas algorytmy to algorytm Poszukiwania przypadkowego (PRS) oraz algorytm wielokrotnego startu (MS), natomiast wybrane funkcje to funkcja Alpine01 oraz funkcja Ackleya

Funkcja Ackleya

Funkcja Ackleya to niewypukła funkcja służąca do testowania wydajności algorytmów optymalizacji, której minimum znajduje się na środku dziedziny. Jest częścią pakietu smooof środowiska R i wyrażona jest poniższym wzorem.

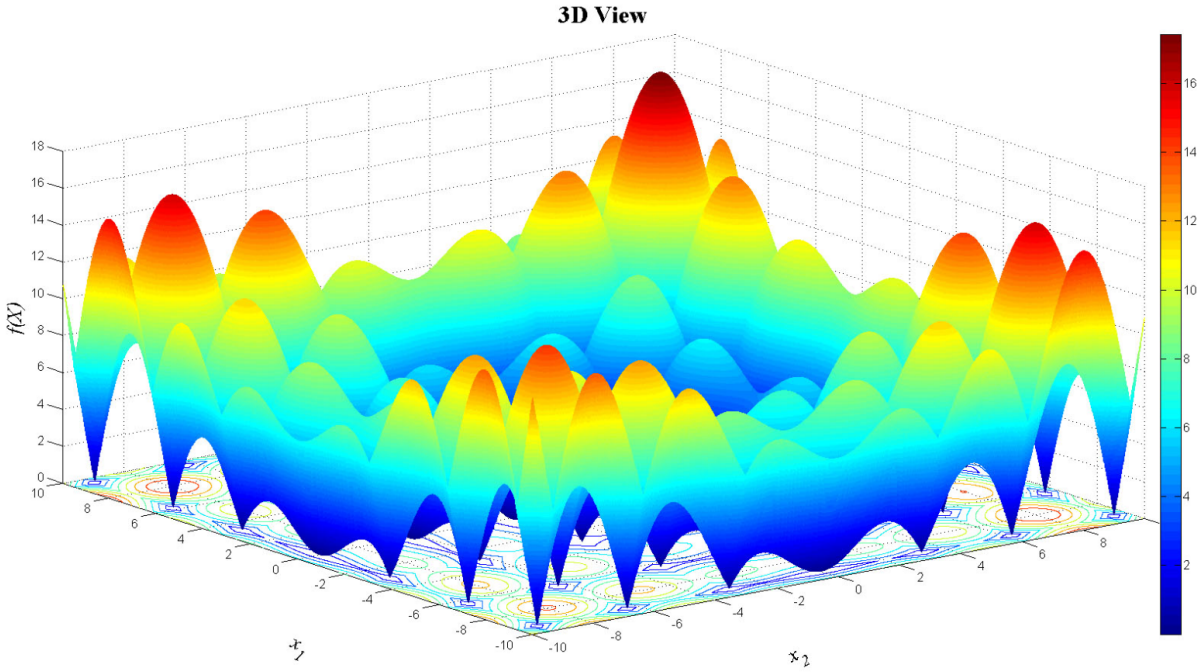
$$f(x) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i)}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$$



Funkcja Alpine01

Alpine01 to inny przykład funkcji z pakietu smoof, której również użyjemy do testowania naszych algorytmów. Od funkcji Ackleya odróżnia ją to, że ma znacząco więcej ekstremów (zarówno minimów jak i maksimów), co możemy zobaczyć na wykresie funkcji Alpine 2D poniżej.

$$f(x) = \sum_{i=1}^{n} |x_i \sin(x_i) + 0.1x_i|$$



Algorytm Poszukiwania Przypadkowego (Pure Random Search)

Poniższy kod przedstawia implementacje algorytmu PRS w języku R polegającego na losowaniu zadanej ilości punktów wylosowanych z rozkładu jednostajnego. Dla każdego z nich oblicza wartość funkcji, za każdym razem porównując z najmniejszą do tej pory wyliczoną wartością (zaczynamy z min_val=inf). Po sprawdzeniu wszystkich punktów algorytm zwraca najmniejszą napotkaną wartość.

```
prs_algorithm <- function(fun, dim, num_of_points){
  f <- fun(dim);
  min_value <- Inf

  for(x in 1:num_of_points){
    point <- runif(dim, getLowerBoxConstraints(f), getUpperBoxConstraints(f))
    value <- f(point)

    if(value < min_value){
      min_value <- value
    }
  }

  return(min_value)
}
```

Metoda wielokrotnego startu (Multi-Start,ms)

Poniższy kod przedstawia implementacje w języku R algorytmu wielokrotnego startu, który podobnie jak algorytm PRS losuje zadaną ilość punktów z rozkładu jednostajnego dla których znajduje minimum. Znajdowanie minimum odbywa się przy wykorzystaniu funkcji optim() z metodą "**L-BFGS-B**". Dodatkowo przy wykorzystanie funkcji optim, określamy budżet wywołań algorytmu PRS.

```
ms_algorithm <- function(fun, dim, num_of_points){
  f <- fun(dim);
  min_value <- Inf
  counter <- 0
  for(x in 1:num_of_points){
    point <- runif(dim, getLowerBoxConstraints(f), getUpperBoxConstraints(f))

    result <- optim(point, f, method = "L-BFGS-B",
                    lower = getLowerBoxConstraints(f), upper = getUpperBoxConstraints(f))
    value <- as.numeric(result$value)
    counter <- counter + as.numeric(result$counts[1])

    if(value < min_value){
      min_value <- value
    }
  }
  return(list(min_value, counter))
}
```

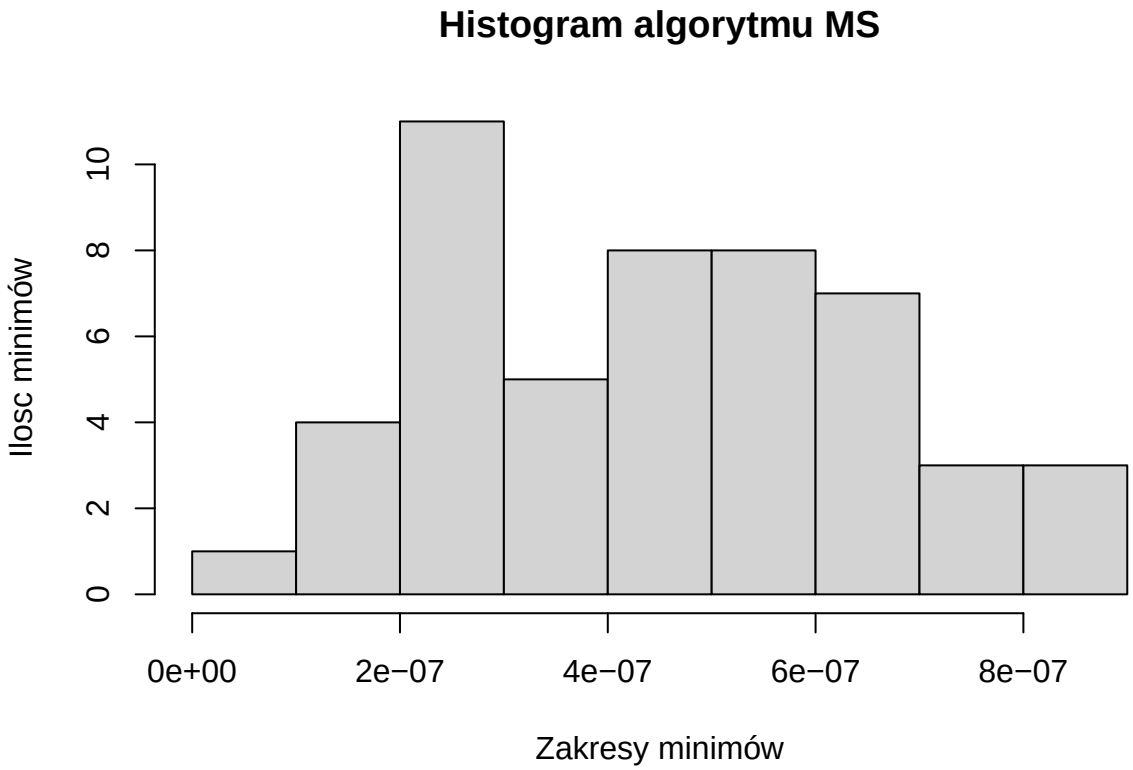
Algorytm porównujący wyniki algorytmów PRS i MS dla zadanej funkcji i wymiaru

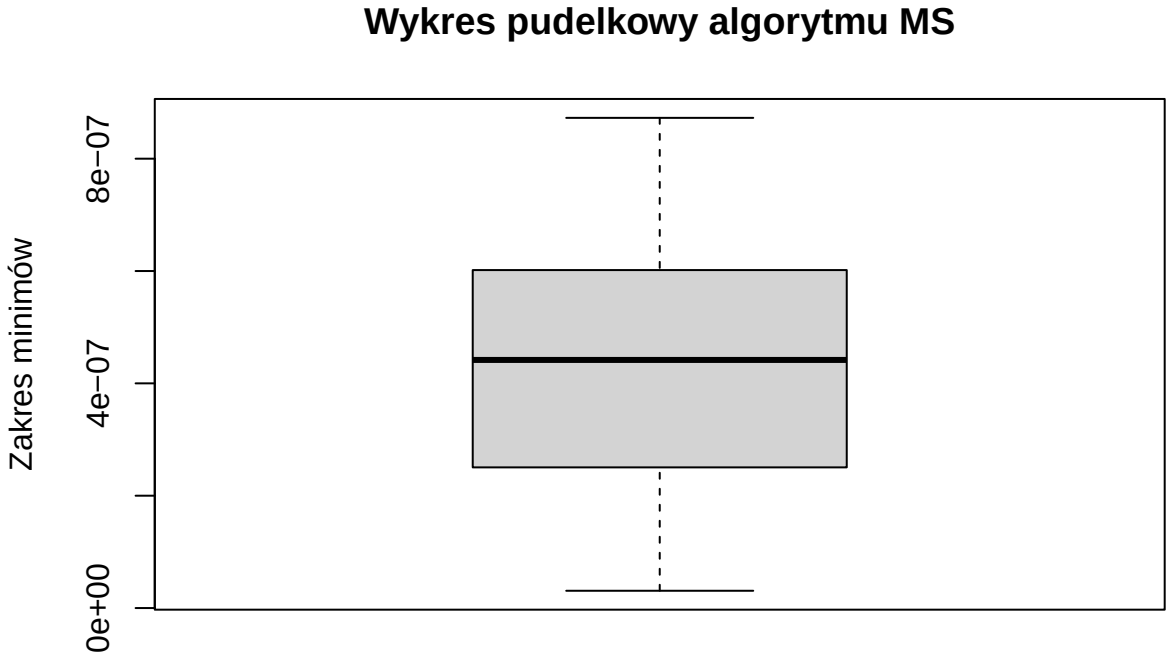
Poniższy kod w pierwszej kolejności wywołuje algorytm MS 50 razy dla 100 punktów, a następnie oblicza średnią wywołań, która jednocześnie jest budżetem algorytmu PRS. W kolejnym kroku wywoływany jest algorytm PRS, również 50 razy dla określonego budżetu. Funkcja porównująca zapisuje wyniki obliczeń algorytmów przeszukujących i zwraca je w celu opracowania danych.

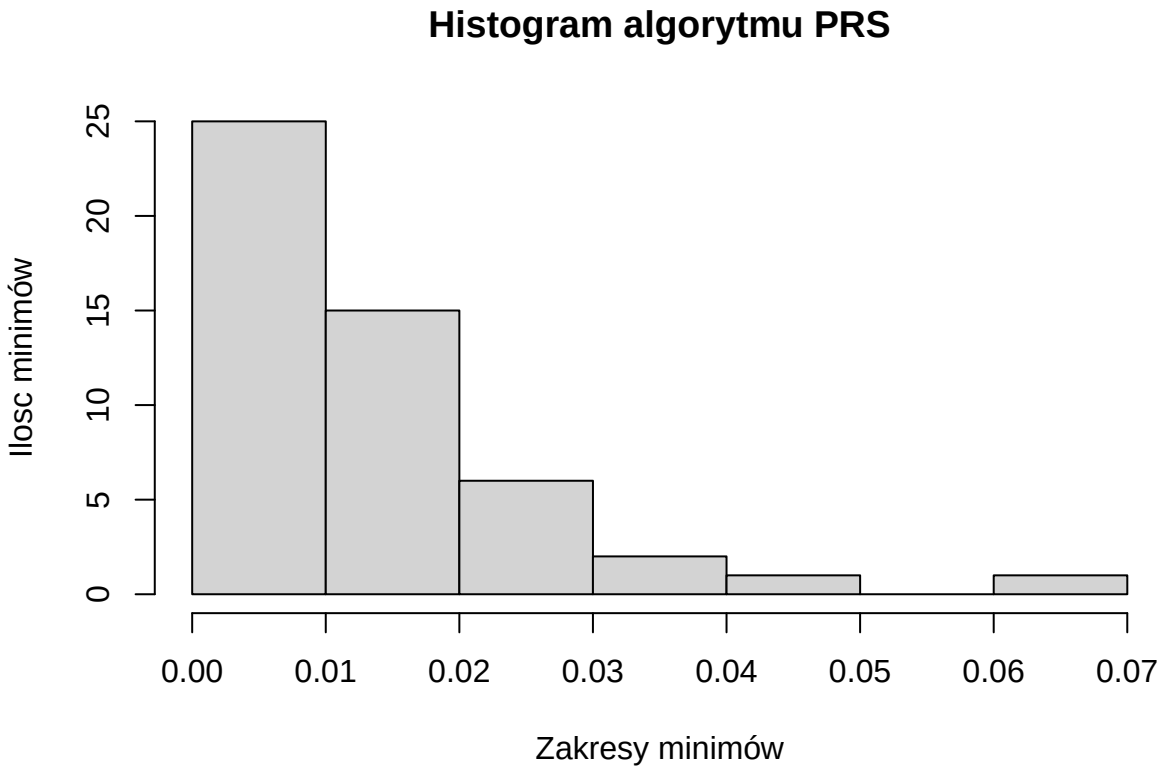
```
compare_algorithms <- function(alg1, alg2, func, dim){
  res_alg1 <- replicate(50, alg1(func, dim, 100))
  alg1_counters <- as.numeric(res_alg1[2,])
  alg1_points <- as.numeric(res_alg1[1,])
  counter <- mean(alg1_counters)
  res_alg2 <- replicate(50, alg2(func, dim, counter))
  return(list(alg1_points,res_alg2))
}
```

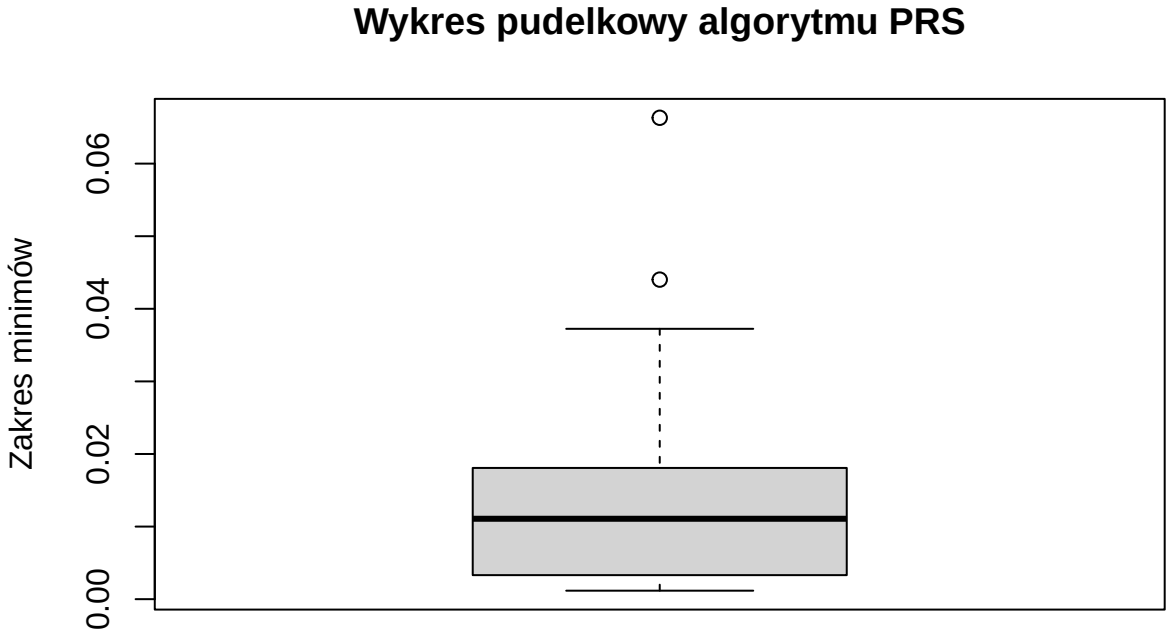
Opracowanie wyników

Porównanie algorytmów dla funkcji Alpine01 i 2 wymiarów









Z wykresów wyników pracy obu algorytmów widać wyraźną przewagą algorytmu MS, którego wyniki były rzędu 10^-4 niższe algorytmu PRS. Dodatkowo MS znajduje minima w znacznie węższym obszarze niż jego konkurent.

Przedział ufności i hipoteza zerowa funkcji Alpine01 2 wymiarów dla algorytmu MS

```
##
##  One Sample t-test
##
## data:  result[[1]]
## t = 14.558, df = 49, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  3.837585e-07 5.066777e-07
## sample estimates:
##    mean of x
## 4.452181e-07
```

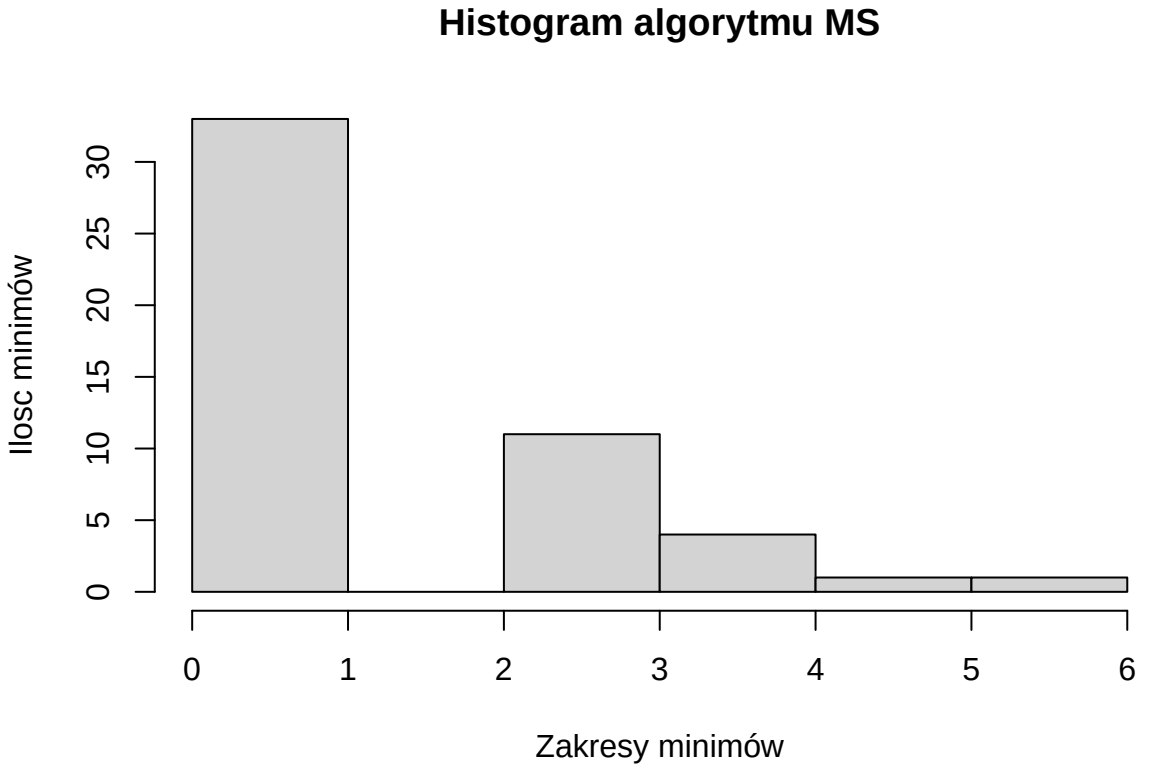
Przedział ufności i hipoteza zerowa funkcji Alpine01 2 wymiarów dla algorytmu PRS

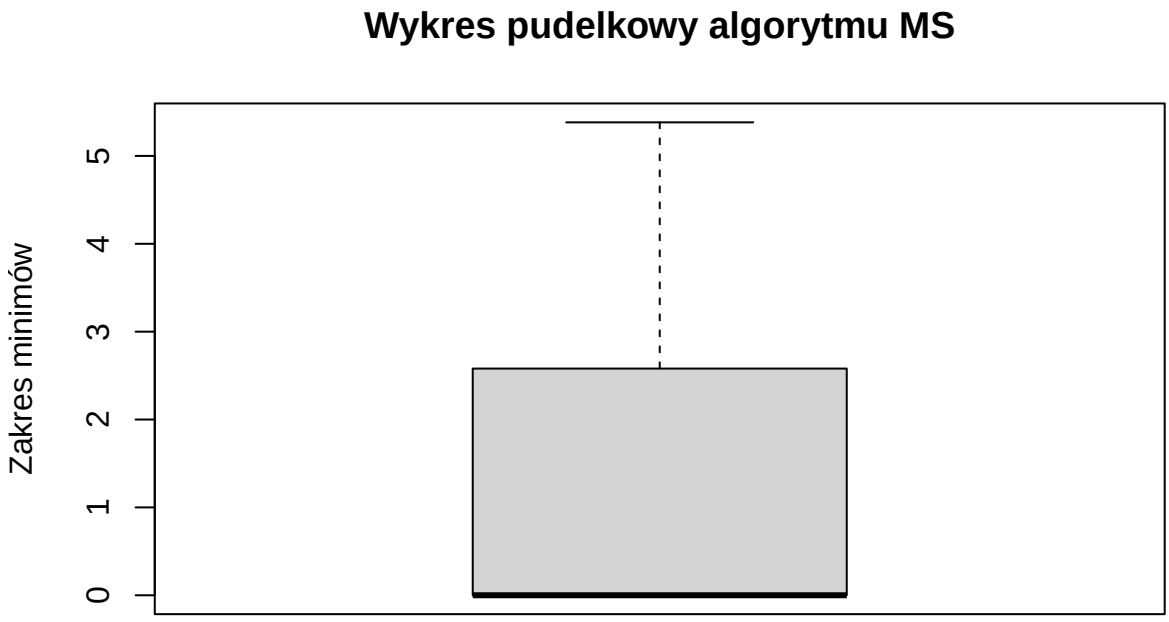
```
##
##  One Sample t-test
##
## data:  result[[2]]
## t = 7.2592, df = 49, p-value = 2.634e-09
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.009419342 0.016630872
## sample estimates:
##  mean of x
## 0.01302511
```

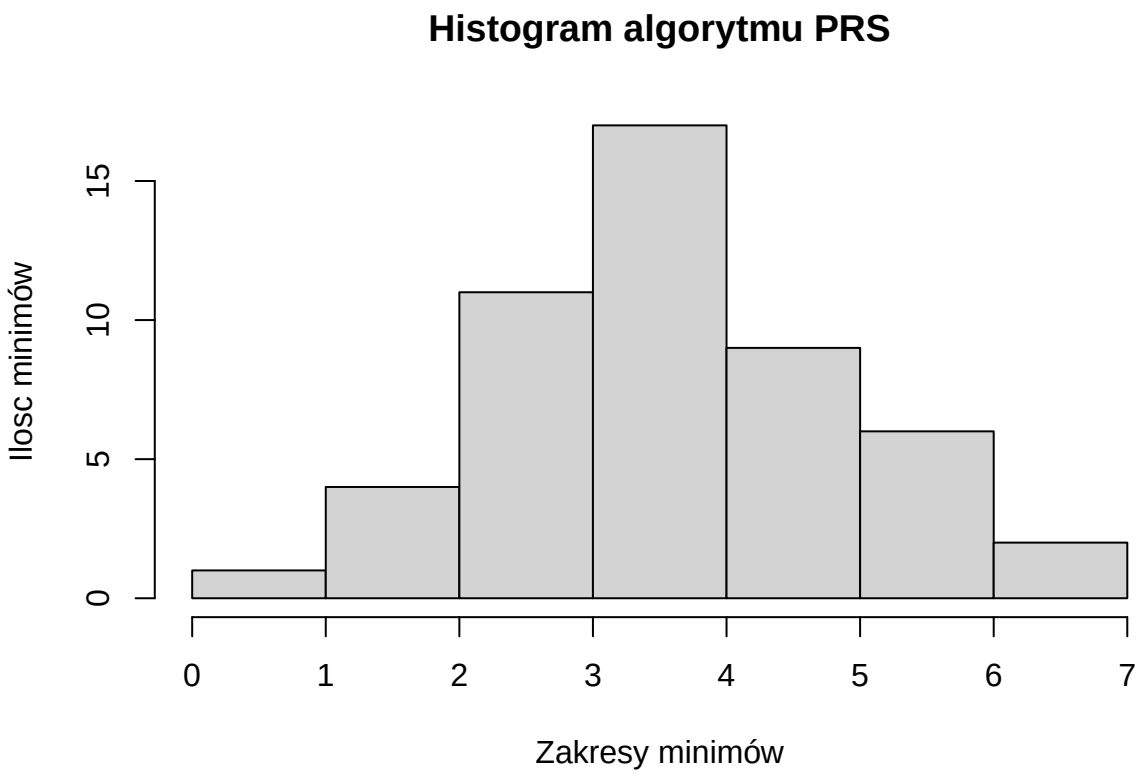
Przedział ufności i hipoteza zerowa funkcji Alpine01 2 wymiarów dla porównania algorytmów MS i PRS

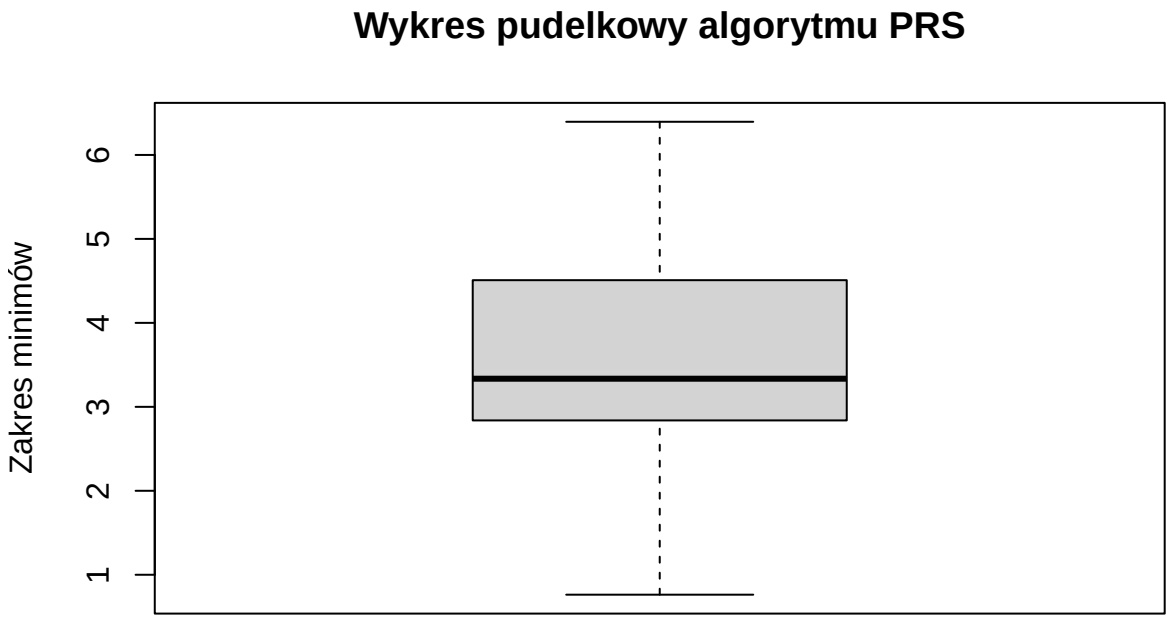
```
##
##  Welch Two Sample t-test
##
## data:  result[[1]] and result[[2]]
## t = -7.2589, df = 49, p-value = 2.637e-09
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.016630426 -0.009418897
## sample estimates:
##    mean of x    mean of y
## 4.452181e-07 1.302511e-02
```

Porównanie wyników algorytmów dla funckji Ackleya i 2 wymiarów









Algorytm MS znalazł mnóstwo minimów w okolicach 0, co zgadza się z wykresem tej funkcji, natomiast, znalazł również dużo punktów odbiegających od 0, spowodowane większą ilością minimów lokalnych utrudniających znalezienie minima globalnego. PRS jednak wypadł dużo gorzej, nie będąc nawet bliskim znalezienia minimów w zerze.

Przedział ufności i hipoteza zerowa funkcji Ackleya 2 wymiarów dla algorytmu MS

```
##
##  One Sample t-test
##
## data:  result2[[1]]
## t = 4.763, df = 49, p-value = 1.735e-05
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.6121114 1.5056062
## sample estimates:
## mean of x
##  1.058859
```

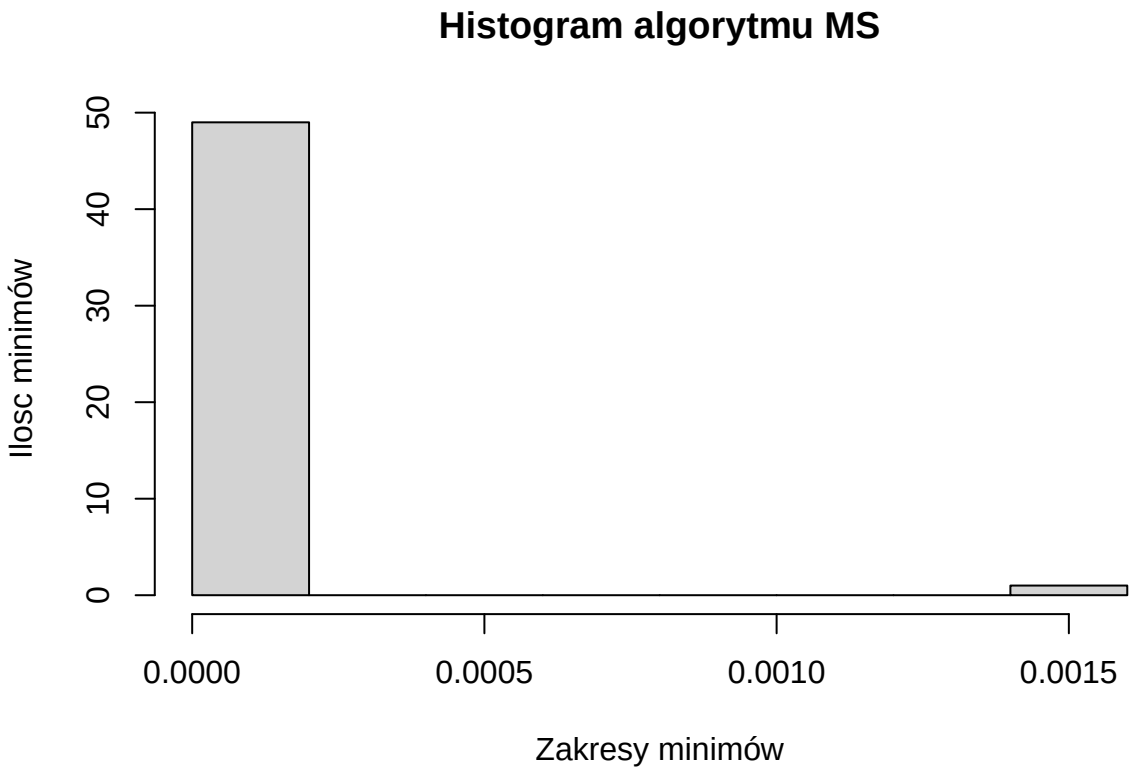
Przedział ufności i hipoteza zerowa funkcji Ackleya 2 wymiarów dla algorytmu PRS

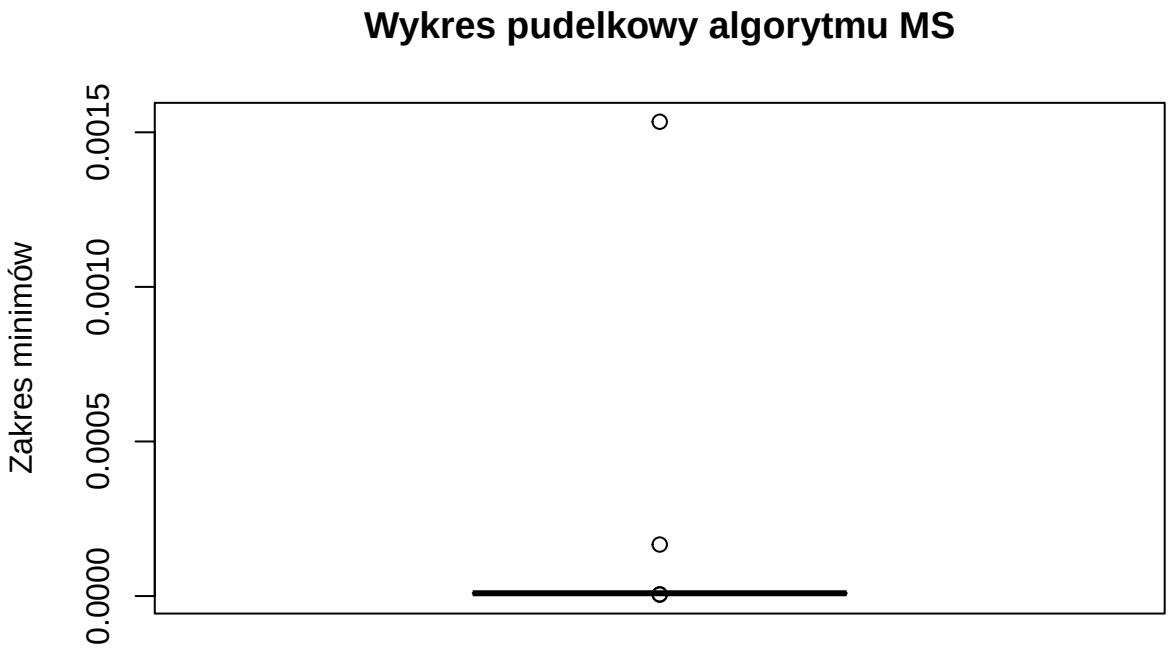
```
##
##  One Sample t-test
##
## data:  result2[[2]]
## t = 20.758, df = 49, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  3.235063 3.928574
## sample estimates:
## mean of x
##  3.581819
```

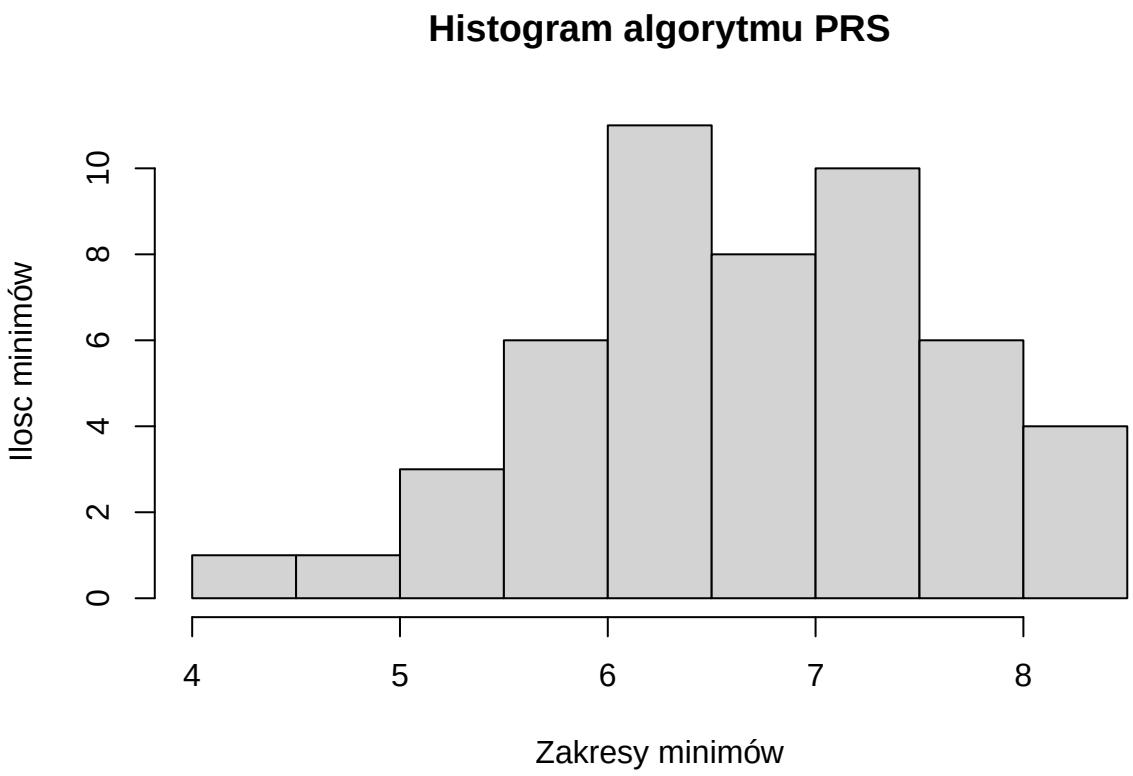
Przedział ufności i hipoteza zerowa funkcji Ackleya 2 wymiarów dla porównania algorytmów MS i PRS

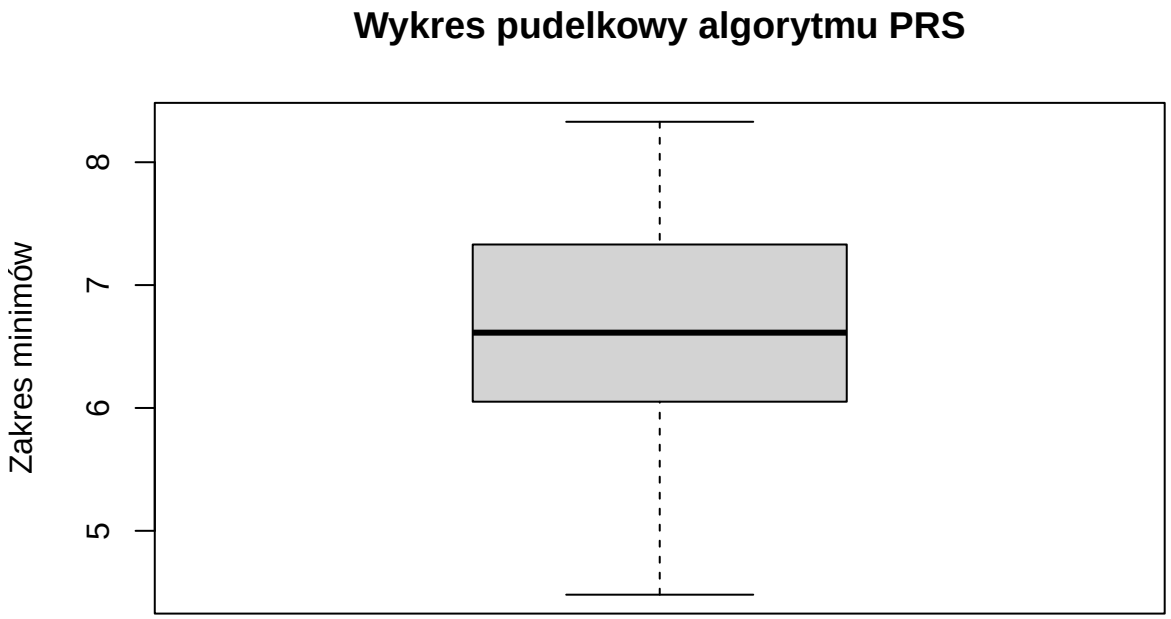
```
##
##  Welch Two Sample t-test
##
## data:  result2[[1]] and result2[[2]]
## t = -8.9652, df = 92.318, p-value = 3.345e-14
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.081852 -1.964067
## sample estimates:
## mean of x mean of y
##  1.058859  3.581819
```

Porównanie algorytmów dla funkcji Alpine01 i 10 wymiarów









Algorytm MS standardowo pokazuje swoją przewagę znajdując znaczącą większość minimów blisko zera, spontanicznie wpadając w minima lokalne oddalone od tego punktu. PRS wciąż wyznacza minima w sporej odległości od tych znalezionych przez MS.

Przedział ufności i hipoteza zerowa funkcji Alpine01 10 wymiarów dla algorytmu MS

```
##
##  One Sample t-test
##
## data:  result3[[1]]
## t = 1.3889, df = 49, p-value = 0.1711
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -1.899554e-05  1.040073e-04
## sample estimates:
##   mean of x
## 4.25059e-05
```

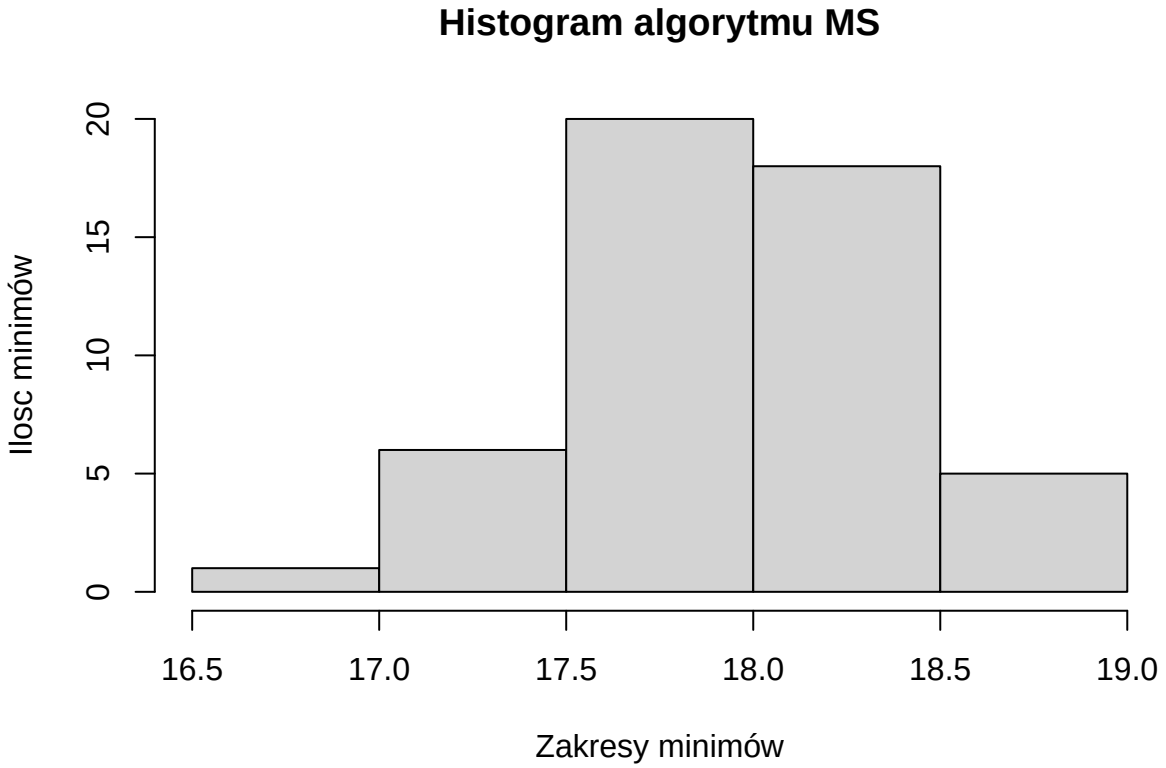
Przedział ufności i hipoteza zerowa funkcji Alpine01 10 wymiarów dla algorytmu PRS

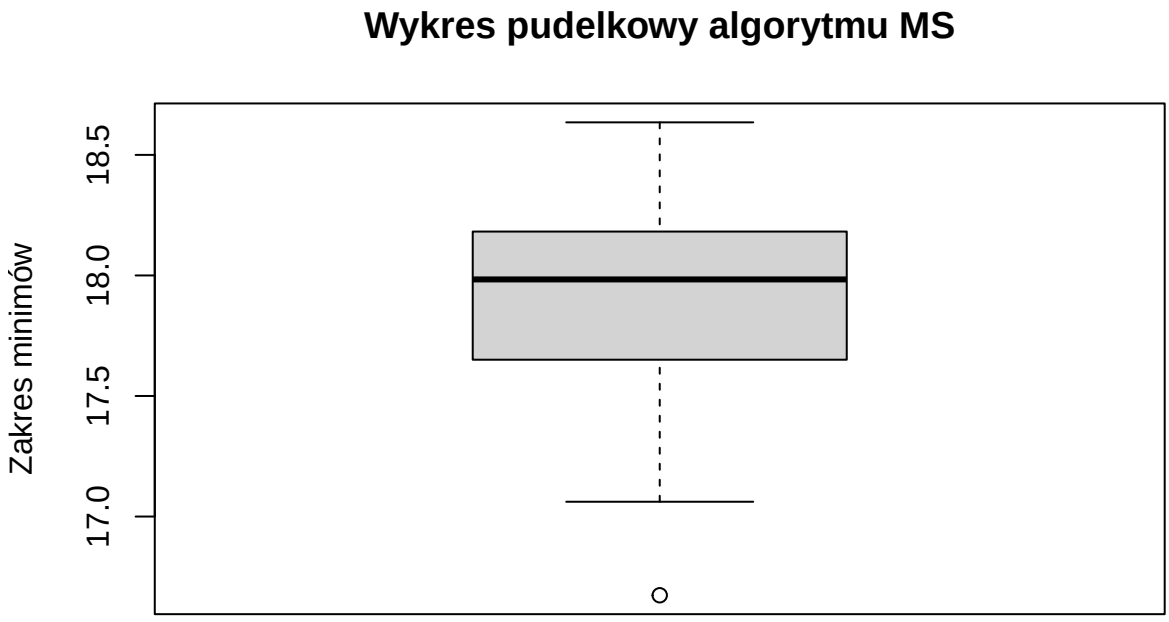
```
##
##  One Sample t-test
##
## data:  result3[[2]]
## t = 51.192, df = 49, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  6.391521 6.913830
## sample estimates:
## mean of x
##  6.652676
```

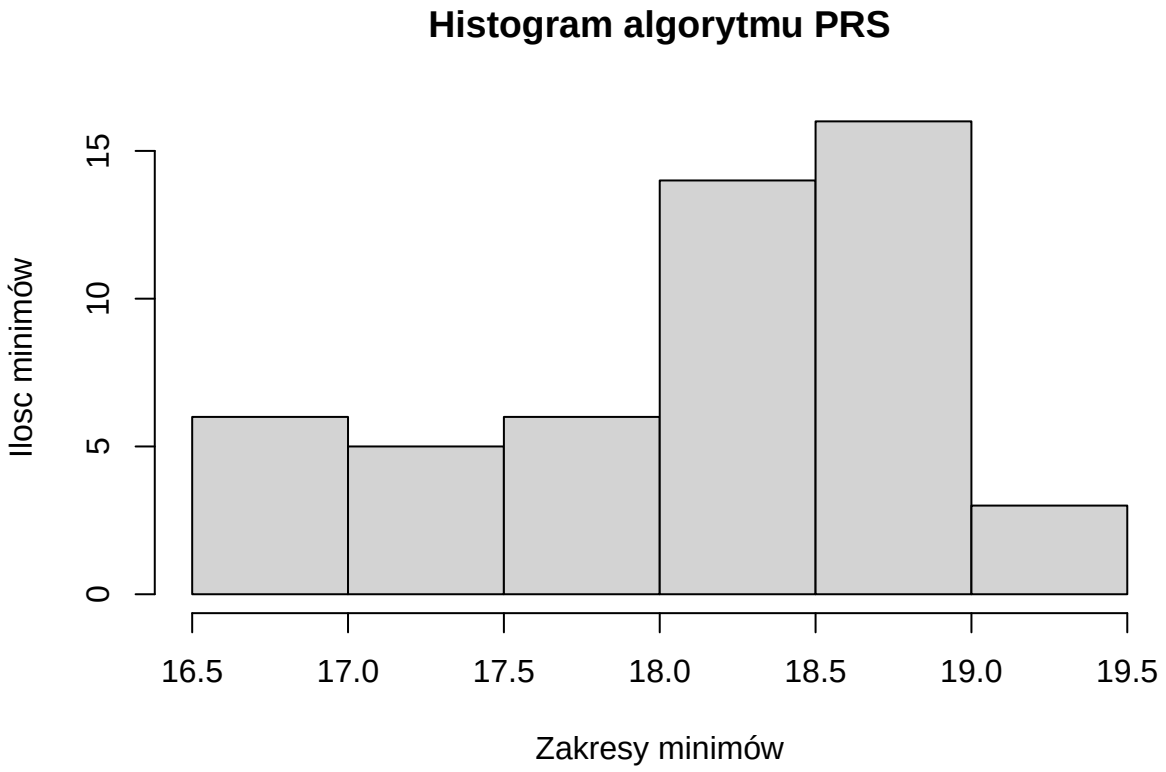
Przedział ufności i hipoteza zerowa funkcji Alpine01 10 wymiarów dla porównania algorytmów MS i PRS

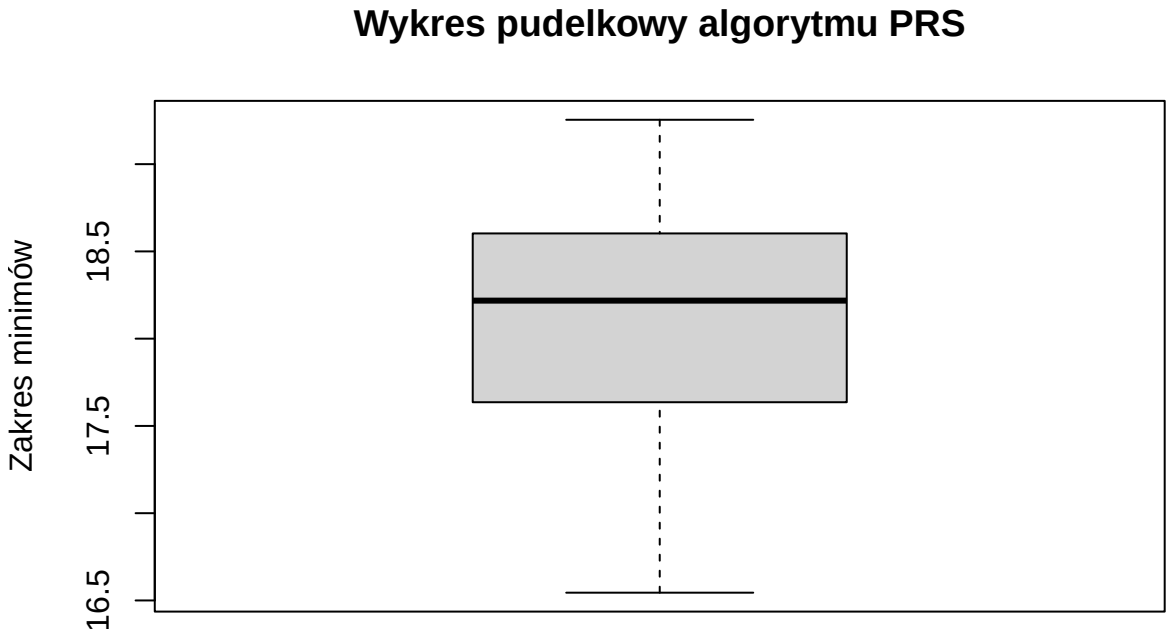
```
##
##  Welch Two Sample t-test
##
## data:  result3[[1]] and result3[[2]]
## t = -51.192, df = 49, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -6.913788 -6.391479
## sample estimates:
##    mean of x    mean of y
## 0.0000425059 6.6526757648
```

Porównanie wyników algorytmów dla funckji Ackleya i 10 wymiarów









Dla funkcji Ackleya w 10 wymiarach napotykamy zaskającą sytuację - oba algorytmy dały bardzo zbliżone wyniki. Rozbieżność między wyznaczonymi przez nie średnimi była mniejsza niż 0.1.

Przedział ufności i hipoteza zerowa funkcji Ackleya 10 wymiarów dla algorytmu MS

```
##
##  One Sample t-test
##
## data:  result4[[1]]
## t = 298.86, df = 49, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  17.80918 18.05030
## sample estimates:
## mean of x
##  17.92974
```

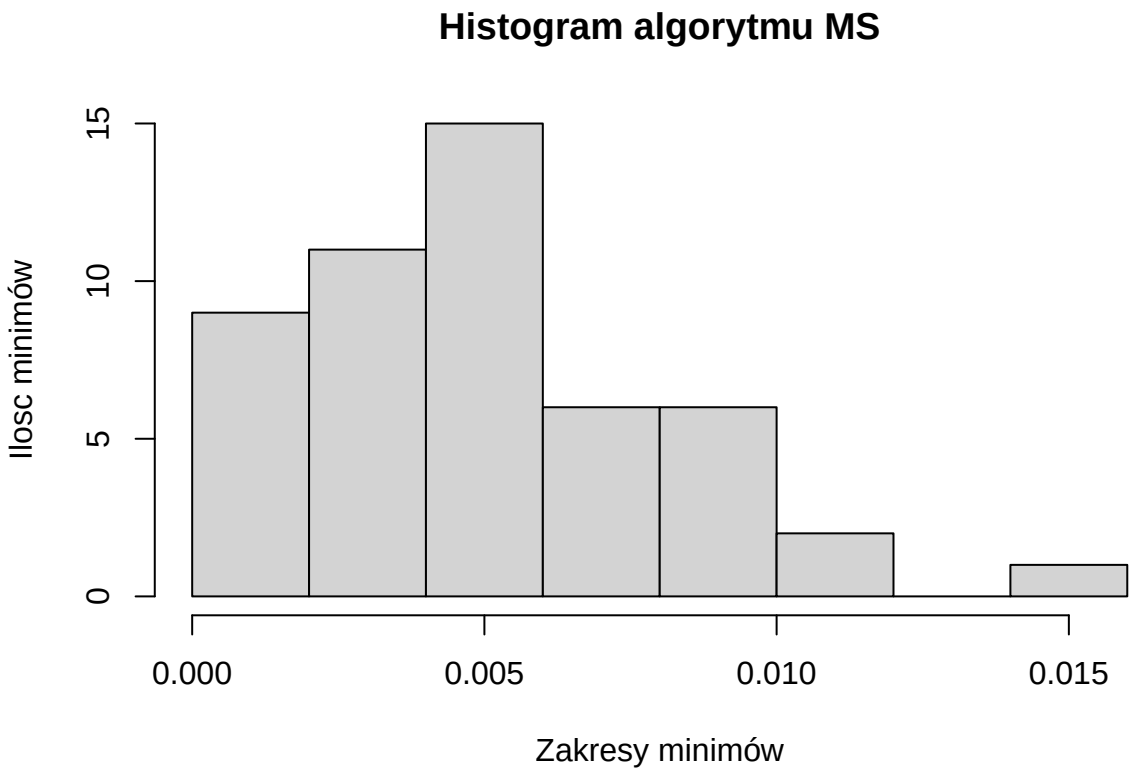
Przedział ufności i hipoteza zerowa funkcji Ackleya 10 wymiarów dla algorytmu PRS

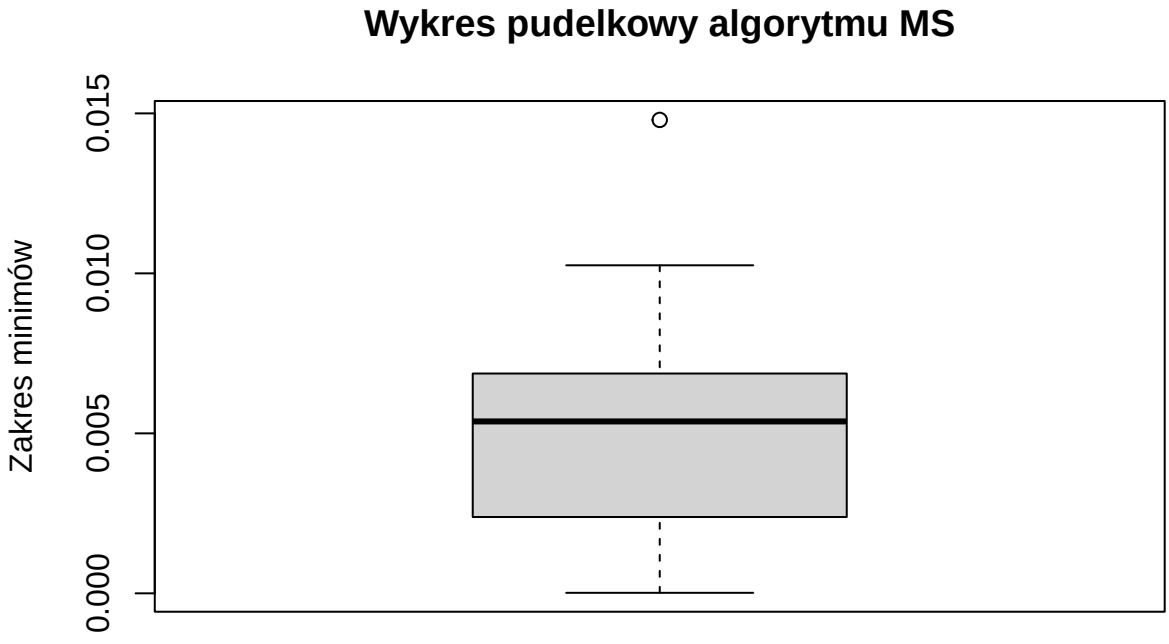
```
##
##  One Sample t-test
##
## data:  result4[[2]]
## t = 177.78, df = 49, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  17.88217 18.29106
## sample estimates:
## mean of x
##  18.08661
```

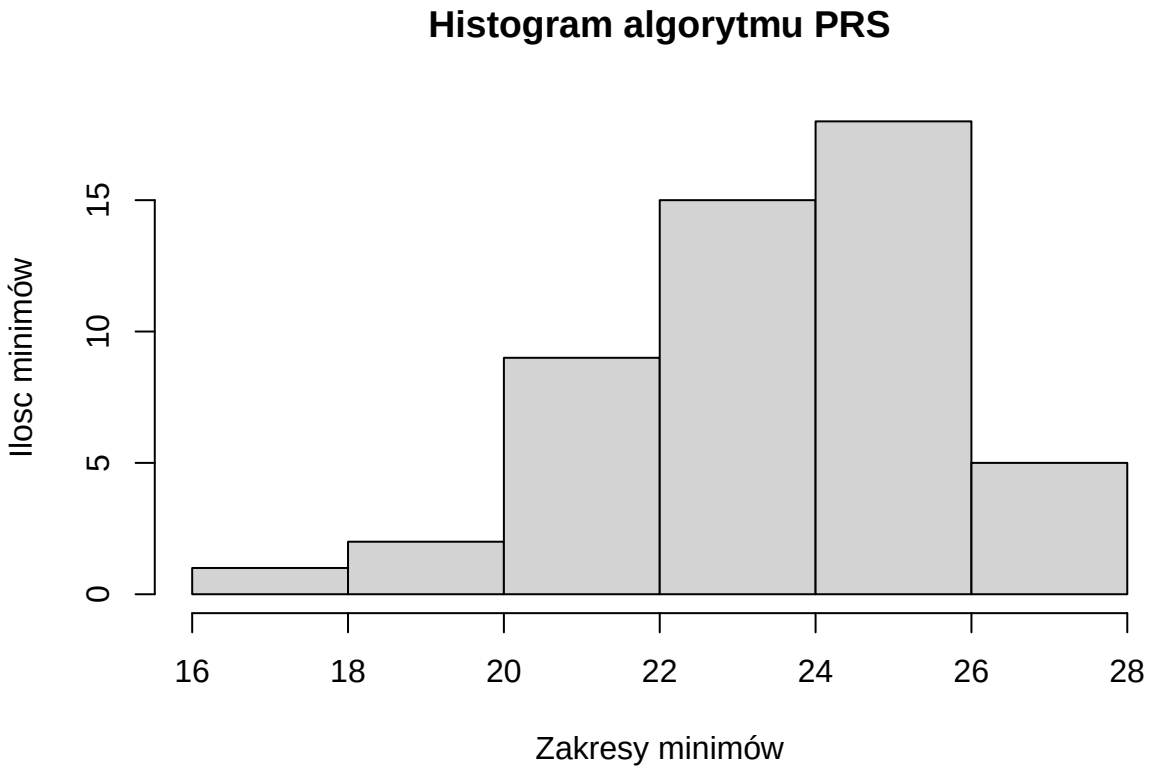
Przedział ufności i hipoteza zerowa funkcji Ackleya 10 wymiarów dla porównania algorytmów MS i PRS

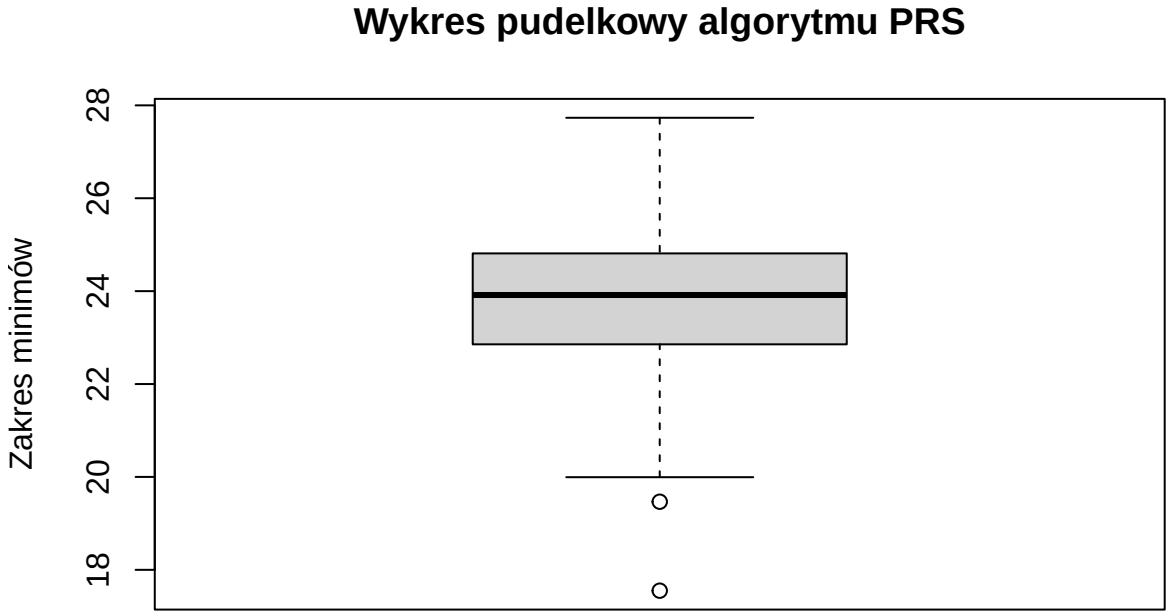
```
##
##  Welch Two Sample t-test
##
## data:  result4[[1]] and result4[[2]]
## t = -1.3282, df = 79.402, p-value = 0.1879
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.39194189  0.07819677
## sample estimates:
## mean of x mean of y
##  17.92974  18.08661
```

Porównanie algorytmów dla funkcji Alpine01 i 20 wymiarów









Dla funkcji Alpine01 w 20 wymiarach, funkcja MS znajduje minima blisko 0. PRS ucieka do wartości blisko 25, które co więcej rozłożone są na znacznie większym obszarze.

Przedział ufności i hipoteza zerowa funkcji Alpine01 20 wymiarów dla algorytmu MS

```
##
##  One Sample t-test
##
## data:  result5[[1]]
## t = 10.905, df = 49, p-value = 1.05e-14
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.003997834 0.005804180
## sample estimates:
##   mean of x
## 0.004901007
```

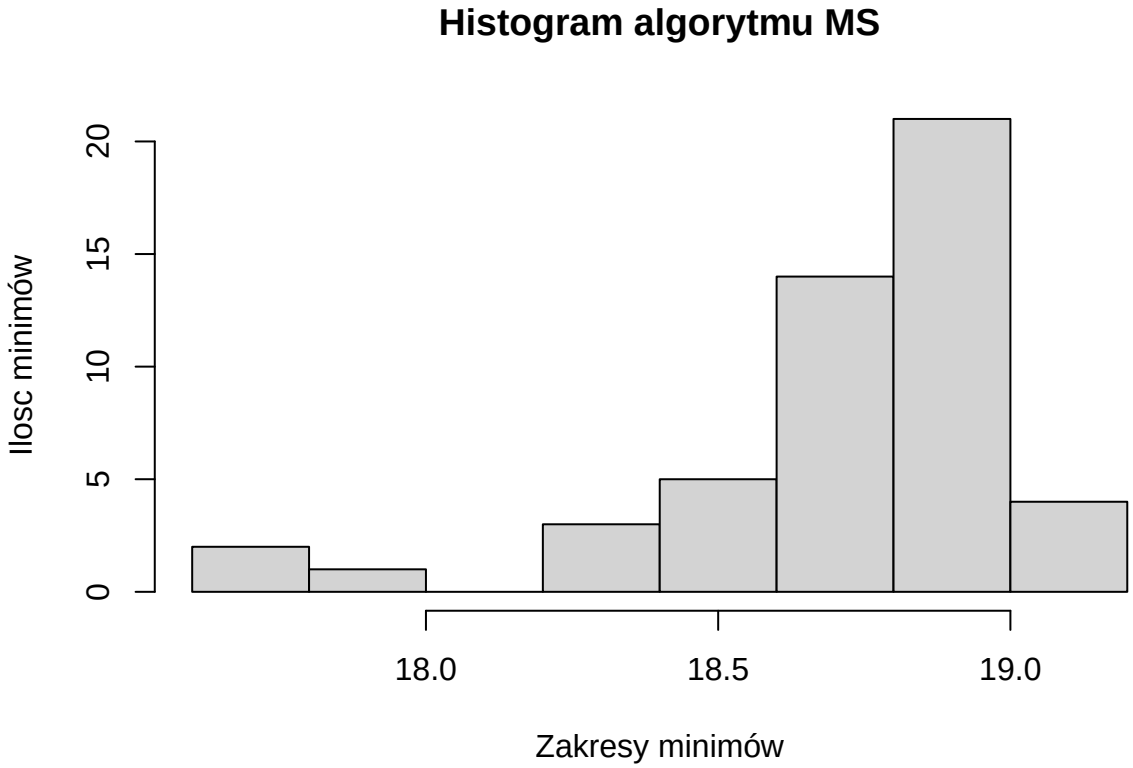
Przedział ufności i hipoteza zerowa funkcji Alpine01 20 wymiarów dla algorytmu PRS

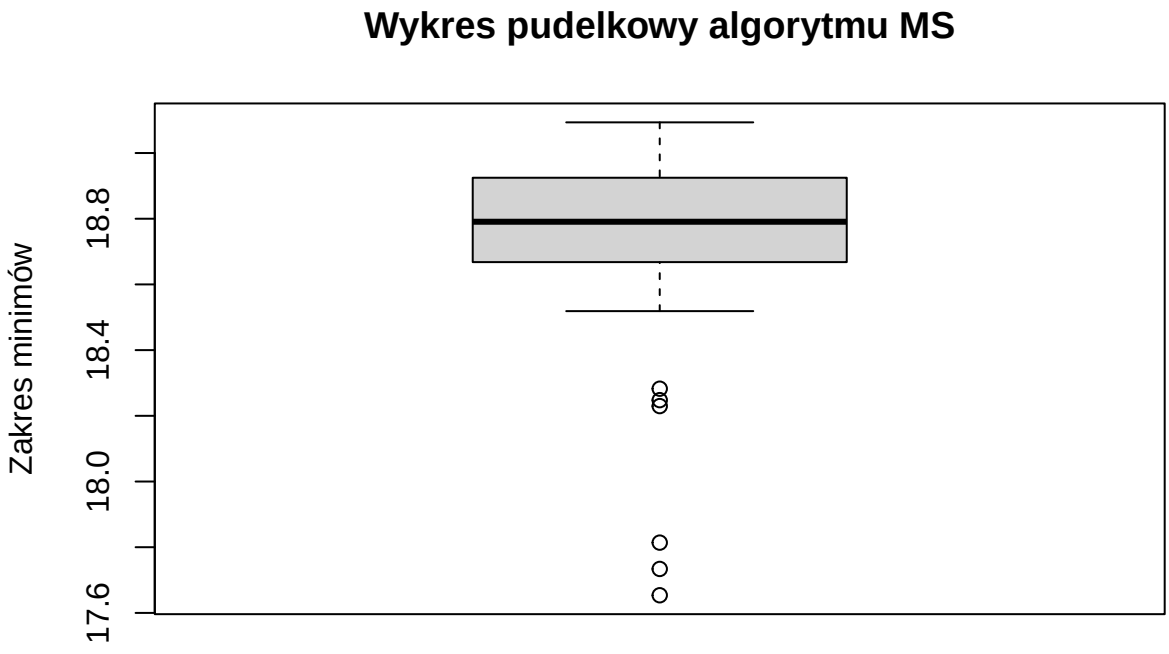
```
##
##  One Sample t-test
##
## data:  result5[[2]]
## t = 81.841, df = 49, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  23.02682 24.18612
## sample estimates:
## mean of x
##  23.60647
```

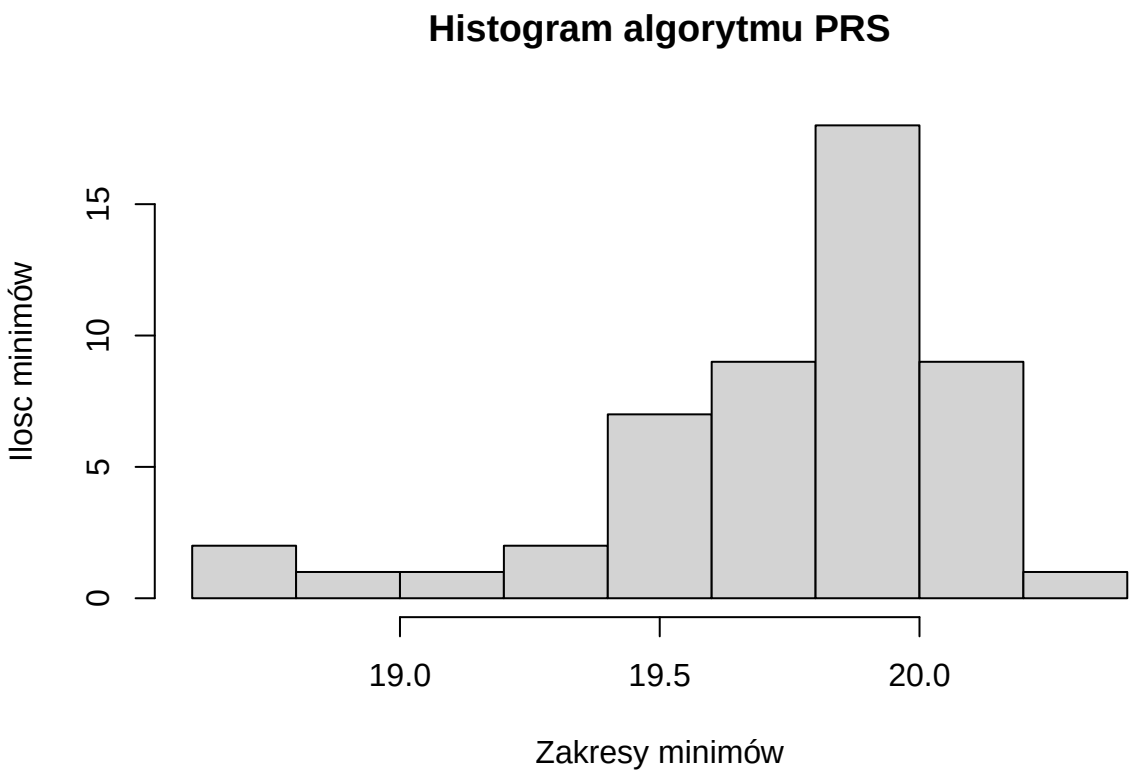
Przedział ufności i hipoteza zerowa funkcji Alpine01 20 wymiarów dla porównania algorytmów MS i PRS

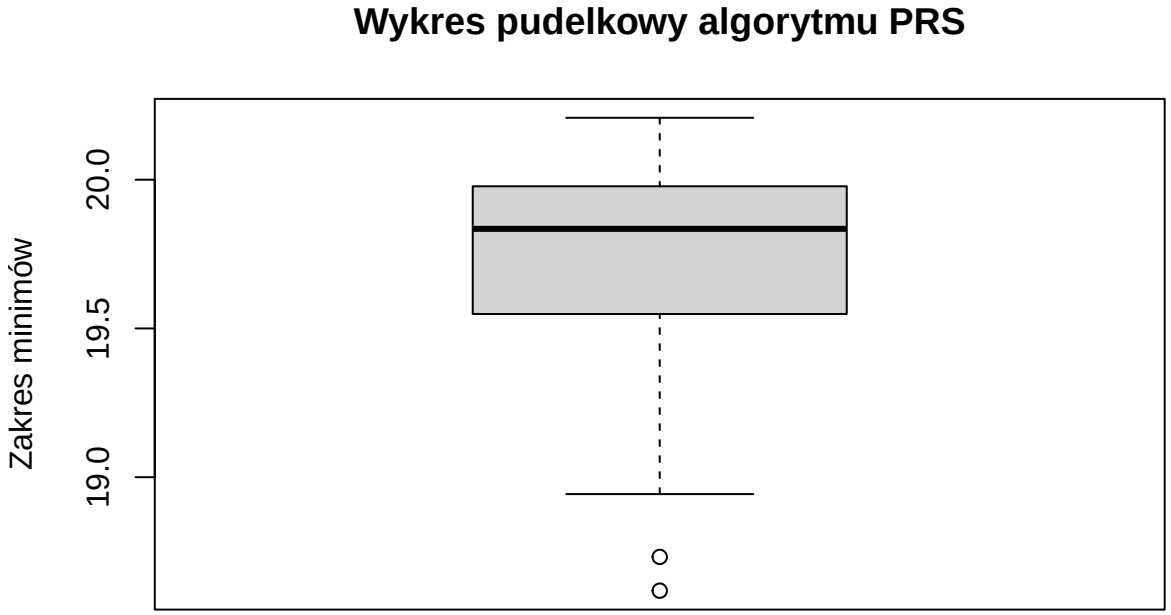
```
##
##  Welch Two Sample t-test
##
## data:  result5[[1]] and result5[[2]]
## t = -81.824, df = 49, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -24.18122 -23.02192
## sample estimates:
##    mean of x    mean of y
##  0.004901007 23.606471067
```

Porównanie wyników algorytmów dla funckji Ackleya i 20 wymiarów









Dla funkcji Ackleya 20 wymiarów ponownie obserwujemy zbliżanie się wyników obu algorytmów. Mimo to, MS wypada lepiej, wyznaczając minima na węższym obszarze niż algorytm PRS.

Przedział ufności i hipoteza zerowa funkcji Ackleya 20 wymiarów dla algorytmu MS

```
##
##  One Sample t-test
##
## data:  result6[[1]]
## t = 415.01, df = 49, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  18.62998 18.81128
## sample estimates:
## mean of x
##  18.72063
```

Przedział ufności i hipoteza zerowa funkcji Ackleya 20 wymiarów dla algorytmu PRS

```
##
##  One Sample t-test
##
## data:  result6[[2]]
## t = 399.79, df = 49, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  19.64003 19.83847
## sample estimates:
## mean of x
##  19.73925
```

Przedział ufności i hipoteza zerowa funkcji Ackleya 20 wymiarów dla porównania algorytmów MS i PRS

```
##
##  Welch Two Sample t-test
##
## data:  result6[[1]] and result6[[2]]
## t = -15.231, df = 97.211, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.1513515 -0.8858892
## sample estimates:
## mean of x mean of y
##  18.72063  19.73925
```

Podsumowanie

Z wyników obliczeń wyraźnie widać przewagę algorytmu MS, który dzięki wykorzystaniu funkcji optim z metodą L-BFGS-B, znajduje minima znacznie skuteczniej, ponieważ nie losuje za każdym razem punktu do sprawdzenia, ale posługuje się gradientem funkcji, dzięki czemu lepiej przewiduje, w którą stronę powinien się kierować.