Projekt zaliczeniowy

Piotr Śliperski & Karol Więckowiak

2023-01-20

Wstępne założenia

Wybraliśmy dwa algorytmy spośród zaproponowanych:

- **Poszukiwanie przypadkowe (Pure Random Search)**
- **Algorytm Genetyczny (GA)**

Oraz dwie funkcje z pakietu *smoof*:

- *Funkcja Rosenbrocka*

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} 100 \cdot (\mathbf{x}_{i+1} - \mathbf{x}_i^2)^2 + (1 - \mathbf{x}_i)^2.$$

The domain is given by the constraints $\mathbf{x}_i \in [-30, 30], i = 1, \ldots, n.$

- *Funkcja Schwefela*

$$f(\mathbf{x}) = \sum_{i=1}^{n} -\mathbf{x}_i \sin\left(\sqrt{(|\mathbf{x}_i|)}\right)$$

with $\mathbf{x}_i \in [-500, 500], i = 1, \ldots, n.$

Ustawiamy seed, żeby wyniki były powtarzalne dla kolejnych uruchomień.

```
set.seed(2137)
```

Definiujemy również funkcje dla zadanych wymiarów (*n= 2, n=10, n=20)*

```
library(smoof)
library(GA)
library(ecr)
rosenbrock2d <- makeRosenbrockFunction(2)
rosenbrock10d <- makeRosenbrockFunction(10)
rosenbrock20d <- makeRosenbrockFunction(20)

schwefel2d <- makeSchwefelFunction(2)
schwefel10d <- makeSchwefelFunction(10)
schwefel20d <- makeSchwefelFunction(20)
```

Definiowanie losowych punktów

Funkcja zwracająca losowy punkt dla **rozkładu jednostajnego**. Na podstawie dobranych wyżej funkcji zakładam że dziedziny są symetryczne

```
getRandomPointInUD <- function(dimensions, domain){
  rndPoint <- replicate(n=dimensions, runif(1,-domain,domain))
  return(rndPoint)
}
```

Szukanie minimalnej wartości funkcji I

Zgodnie z dokumentacją dla porównania PRS-GA określamy budżet obliczeniowy jako 1000 wywołań. Czyli losujemy 1000 punktów i szukamy najmniejszej wartości.

Definiujemy funkcję odpowiedzialną za szukanie minimum wartości funkcji podanej jako argument. W ten sposób unikniemy powtarzania kodu

```
getSmallestVal <- function(numberOfExec,givenFunc, pointsGenerator, dimensions, domain){
  generatedPoints <- replicate(numberOfExec, pointsGenerator(dimensions, domain))
  pointsMat <- matrix(generatedPoints, nrow = numberOfExec)
  res <- apply(pointsMat, 1, givenFunc)
  return(min(res))
}
```

Dla PRS

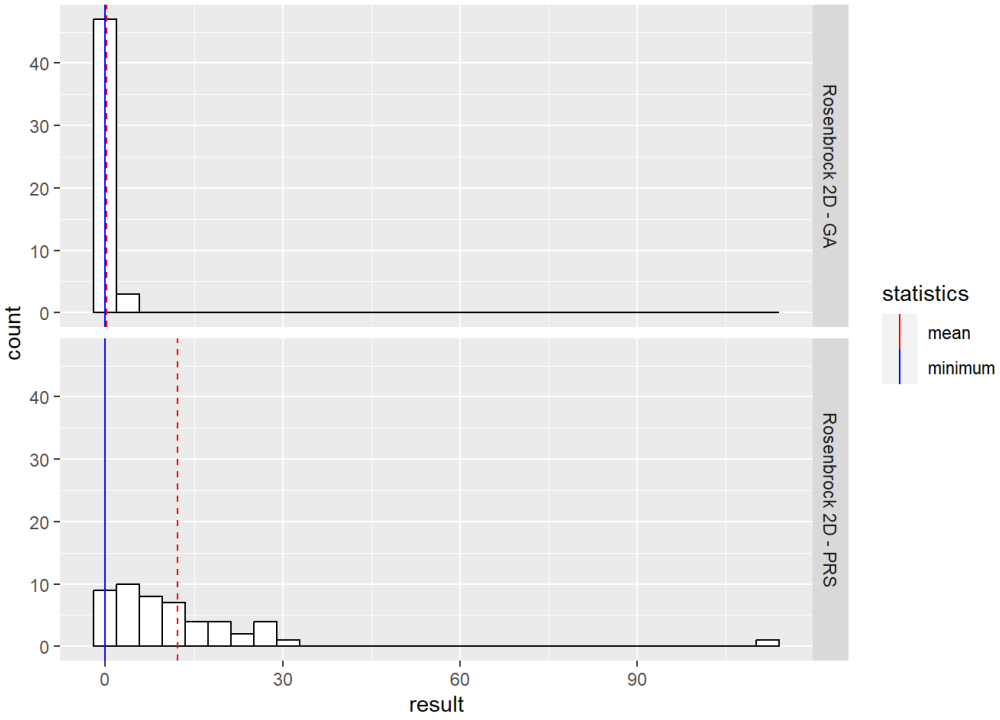
```
Rosenbrock2dResultsPRS <- replicate(50, getSmallestVal(1000, rosenbrock2d, getRandomPointInUD,2,30))
Rosenbrock10dResultsPRS <- replicate(50, getSmallestVal(1000, rosenbrock10d, getRandomPointInUD,10,30))
Rosenbrock20dResultsPRS <- replicate(50, getSmallestVal(1000, rosenbrock20d, getRandomPointInUD,20,30))

Schwefel2dResultsPRS <- replicate(50, getSmallestVal(1000, schwefel2d, getRandomPointInUD, 2, 500))
Schwefel10dResultsPRS <- replicate(50, getSmallestVal(1000, schwefel10d, getRandomPointInUD, 10, 500))
Schwefel20dResultsPRS <- replicate(50, getSmallestVal(1000, schwefel20d, getRandomPointInUD, 20, 500))



MeanRosenbrock2dPRS <- mean(Rosenbrock2dResultsPRS)
MeanRosenbrock10dPRS <- mean(Rosenbrock10dResultsPRS)
MeanRosenbrock20dPRS <- mean(Rosenbrock20dResultsPRS)

MeanSchwefel2dPRS <- mean(Schwefel2dResultsPRS)
MeanSchwefel10dPRS <- mean(Schwefel10dResultsPRS)
MeanSchwefel20dPRS <- mean(Schwefel20dResultsPRS)
```

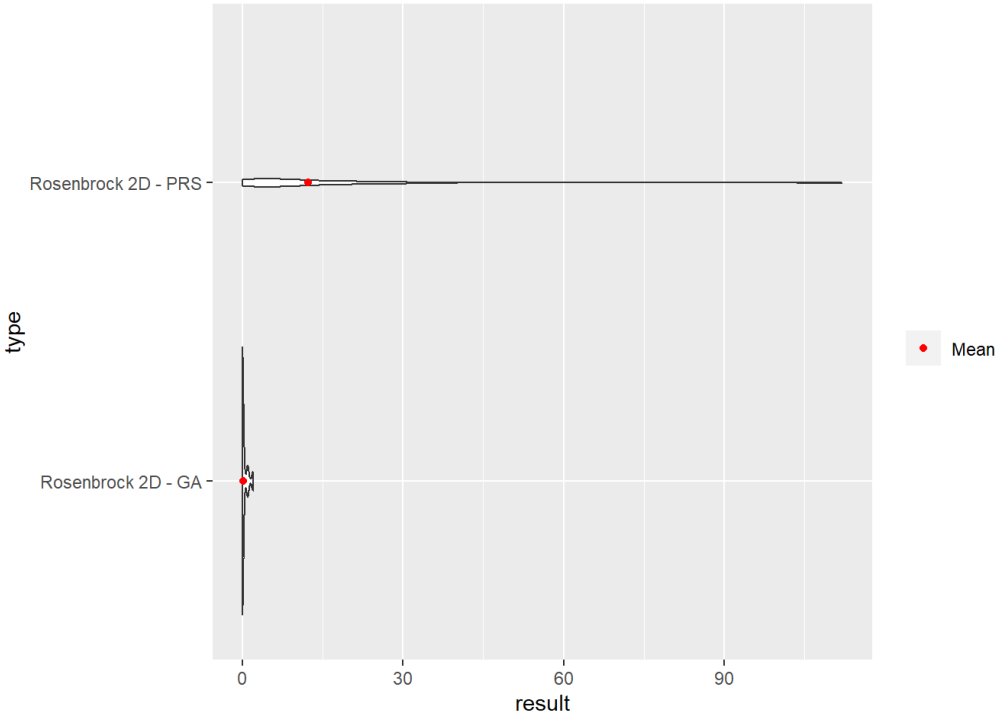
Dla GA

```
maxEvlas <- list(stopOnEvals(1000))
muVar <- 50L
lambdaVar <- 25L

lower <- replicate(2,-30)
upper <- replicate(2,30)
Rosenbrock2dResultsGA <- replicate(50,ecr(rosenbrock2d, n.dim = 2L, lower = lower, upper = upper,
          minimize=TRUE,
          representation = "float", mu = muVar, lambda = lambdaVar,
          terminators = maxEvlas,
          mutator = setup(mutGauss, lower = lower, upper = upper))$best.y)

lower <- replicate(10,-30)
upper <- replicate(10,30)
Rosenbrock10dResultsGA <- replicate(50,ecr(rosenbrock10d, n.dim = 10L, lower = lower, upper = upper,
                                           minimize=TRUE,
          representation = "float", mu = muVar, lambda = lambdaVar,
          terminators = maxEvlas,
          mutator = setup(mutGauss, lower = lower, upper = upper))$best.y)

lower <- replicate(20,-30)
upper <- replicate(20,30)
Rosenbrock20dResultsGA <- replicate(50,ecr(rosenbrock20d, n.dim = 20L, lower = lower, upper = upper,
                                           minimize=TRUE,
          representation = "float", mu = muVar, lambda = lambdaVar,
          terminators = maxEvlas,
          mutator = setup(mutGauss, lower = lower, upper = upper))$best.y)


lower <- replicate(2,-500)
upper <- replicate(2,500)
Schwefel2dResultsGA <- replicate(50,ecr(schwefel2d,n.dim = 2L, lower = lower, upper = upper,
                                        minimize=TRUE,
          representation = "float", mu = muVar, lambda = lambdaVar,
          terminators = maxEvlas,
          mutator = setup(mutGauss, lower = lower, upper = upper))$best.y)

lower <- replicate(10,-500)
upper <- replicate(10,500)
Schwefel10dResultsGA <- replicate(50,ecr(schwefel10d, n.dim = 10L, lower = lower, upper = upper,
                                         minimize=TRUE,
          representation = "float", mu = muVar, lambda = lambdaVar,
          terminators = maxEvlas,
          mutator = setup(mutGauss, lower = lower, upper = upper))$best.y)

lower <- replicate(20,-500)
upper <- replicate(20,500)
Schwefel20dResultsGA <- replicate(50,ecr(schwefel20d, n.dim = 20L, lower = lower, upper = upper,
                                         minimize=TRUE,
          representation = "float", mu = muVar, lambda = lambdaVar,
          terminators = maxEvlas,
          mutator = setup(mutGauss, lower = lower, upper = upper))$best.y)



MeanRosenbrock2dGA <- mean(Rosenbrock2dResultsGA)
MeanRosenbrock10dGA <- mean(Rosenbrock10dResultsGA)
MeanRosenbrock20dGA <- mean(Rosenbrock20dResultsGA)
MeanSchwefel2dGA <- mean(Schwefel2dResultsGA)
MeanSchwefel10dGA <- mean(Schwefel10dResultsGA)
MeanSchwefel20dGA <- mean(Schwefel20dResultsGA)
```

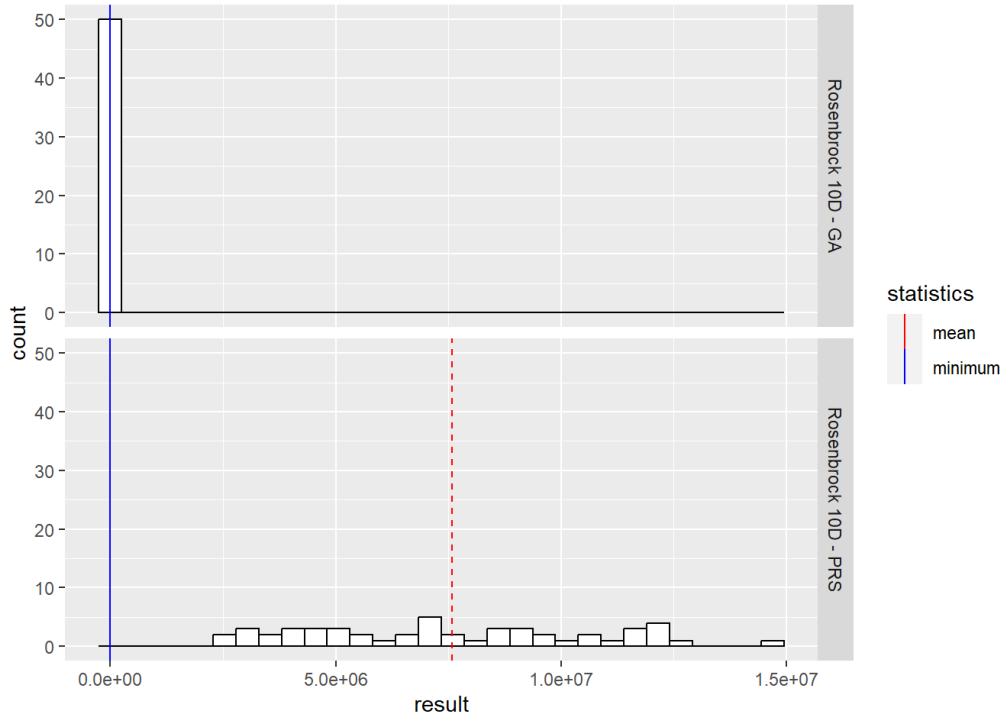
Wizualizacja otrzymanych wyników

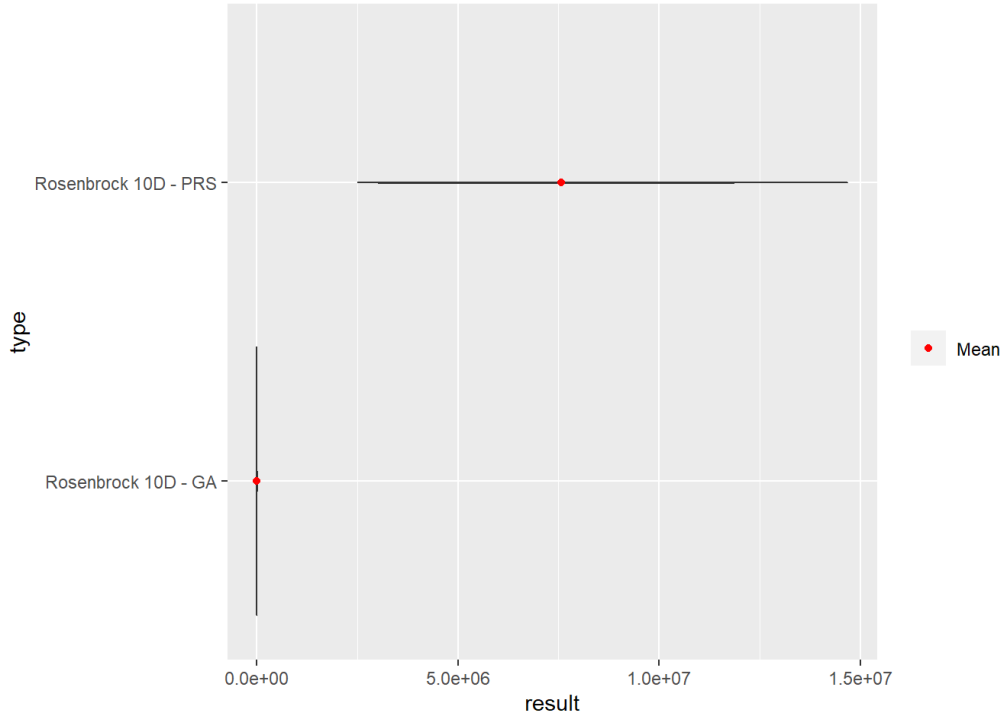
Funkcja Rosenbrocka 2D



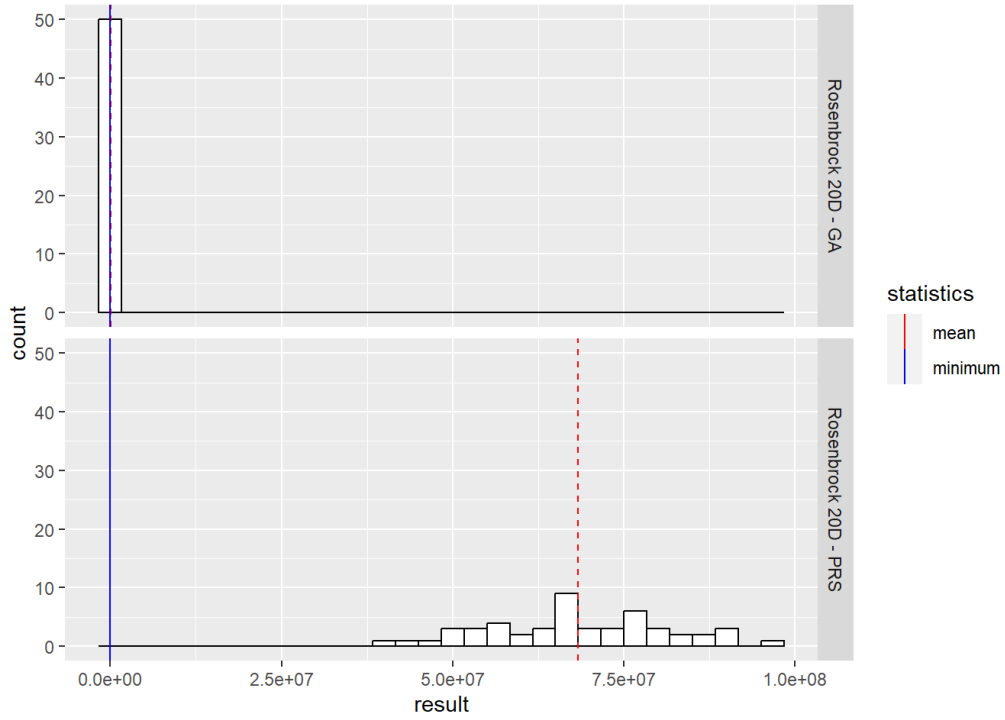


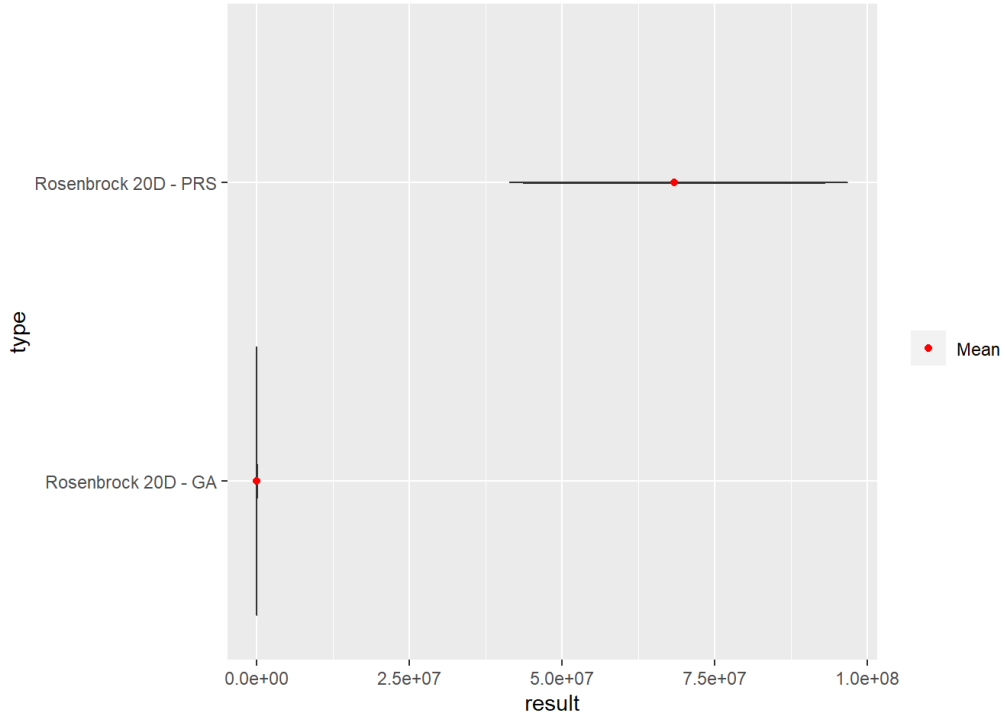
Funkcja Rosenbrocka 10D





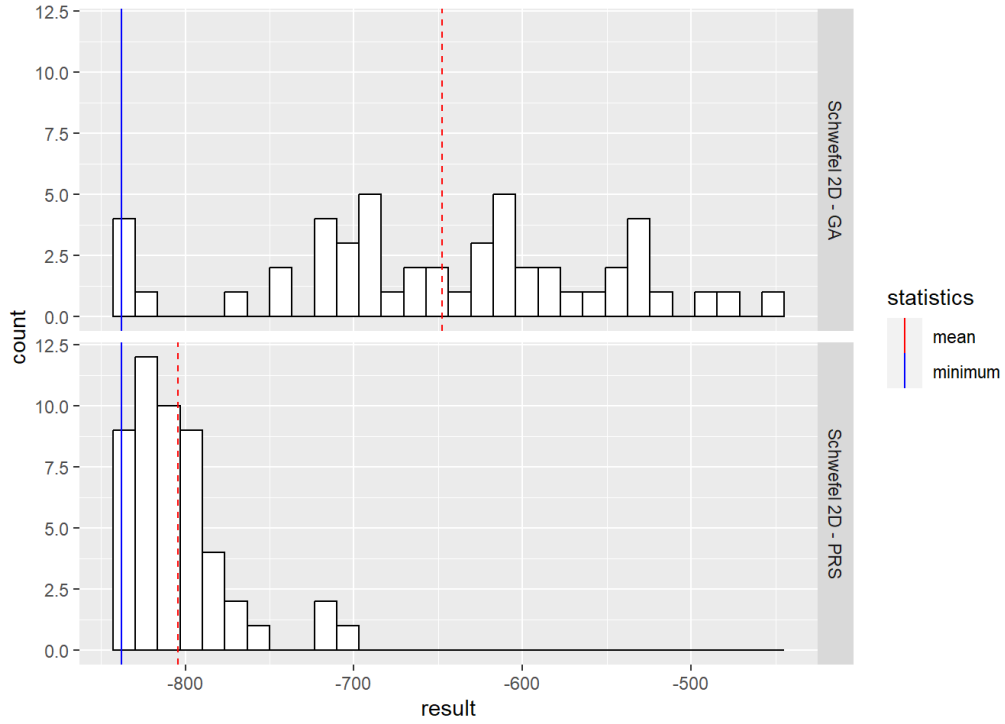
Funkcja Rosenbrocka 20D

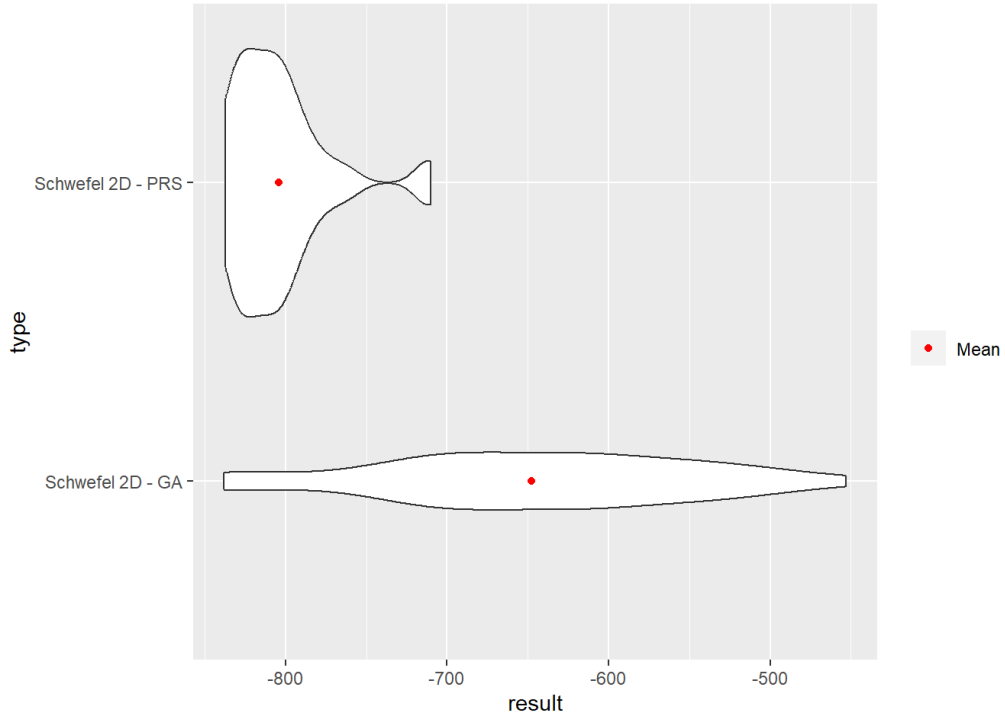




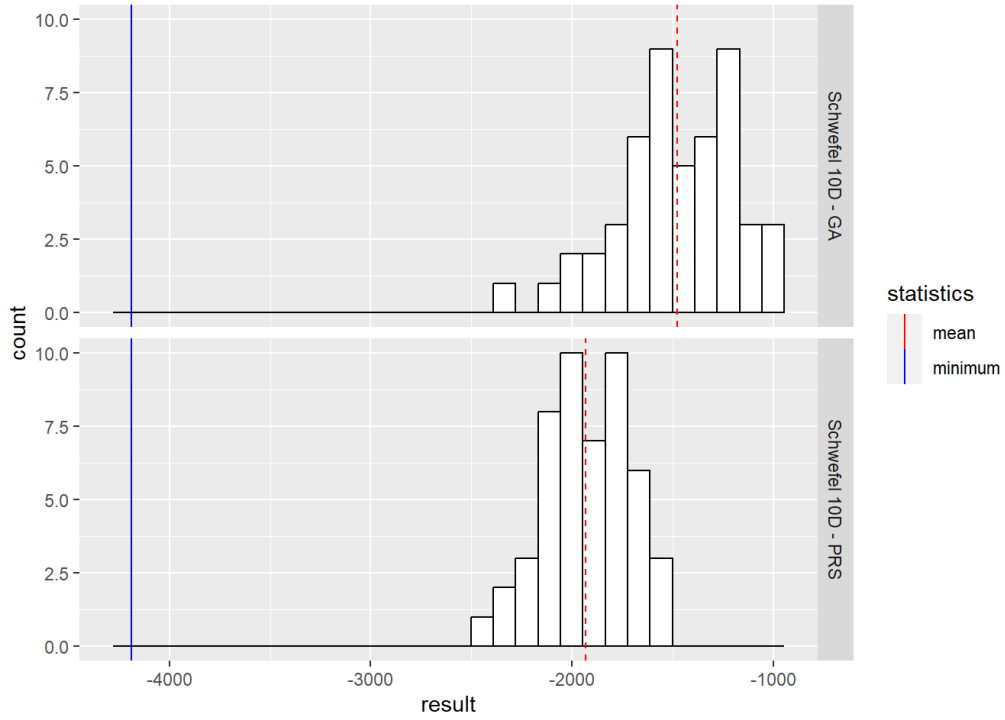
Wykresy skrzypcowe ułożyły się w bardzo dziwne kształty, dla algorytmu genetycznego są pionowe, a dla PRS są pozomie. Widzimy że większość wyników GA była bardzo blisko faktycznego minimum, natomiast wyniki algorytmu PRS były bardzo rozciągnięte po dziedzinie. Jest to spowodowane duża ilością możliwych punktów przez co dla próby 1000 punktów dość ciężko jest “wylosować” właściwe minimum przez PRS, chociaż jak widzimy dla funkcji 2D jest to możliwe.

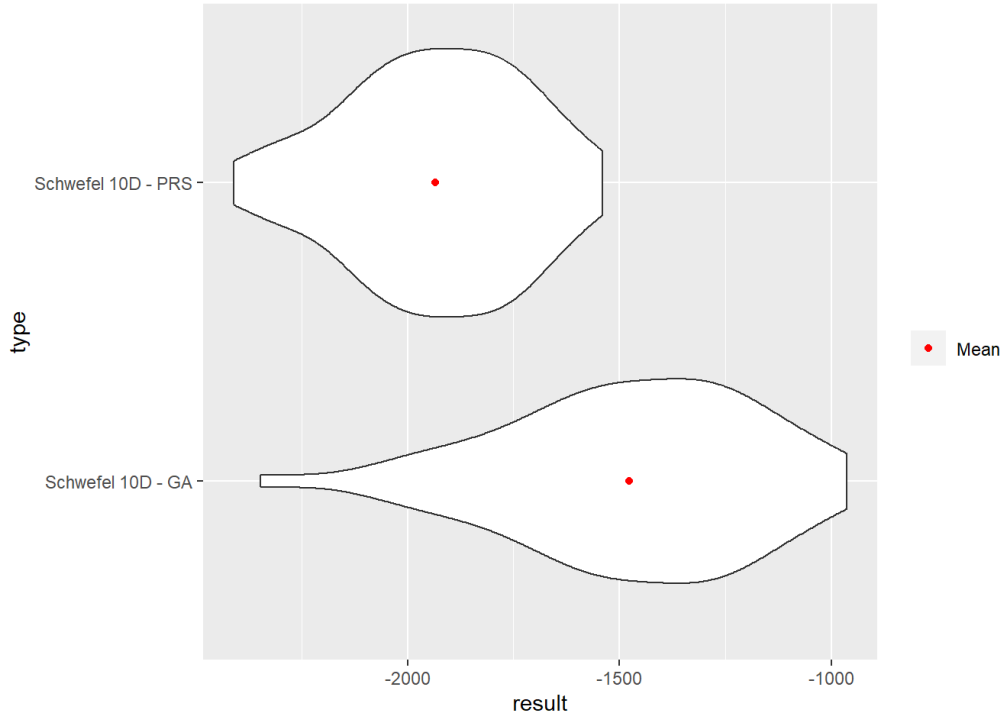
Funkcja Schwefela 2D



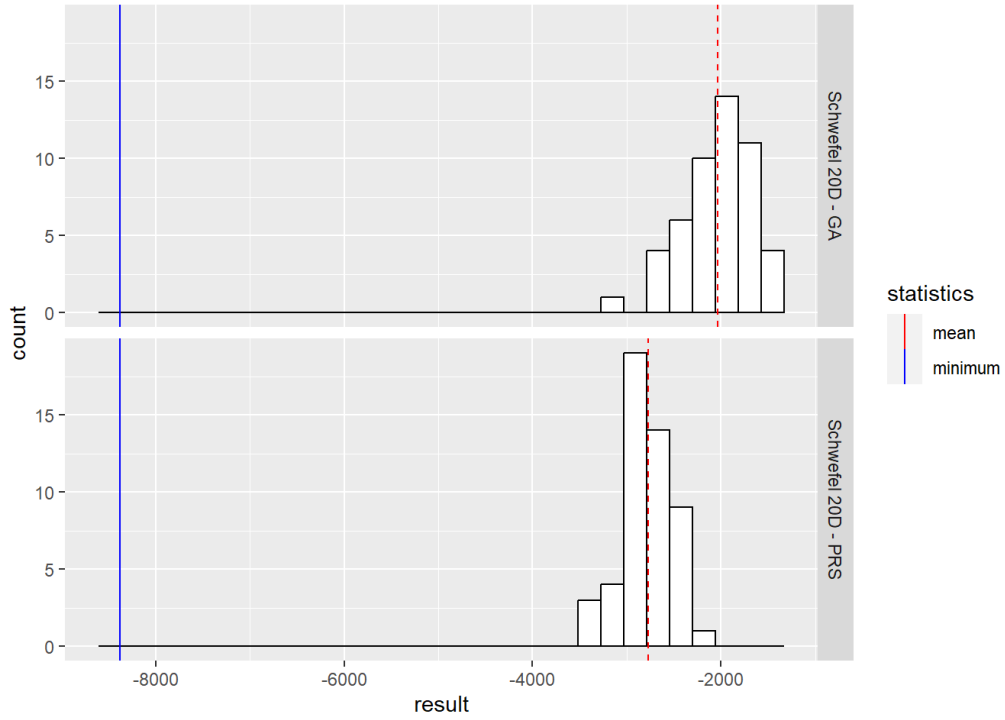


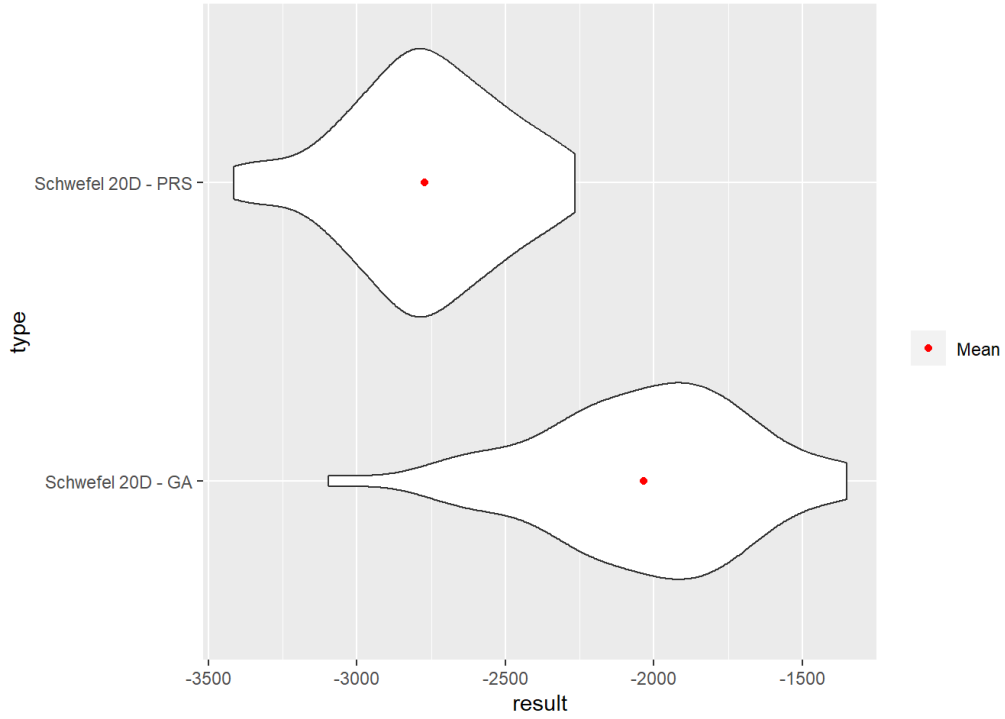
Funkcja Schwefela 10D





Funkcja Schwefela 20D



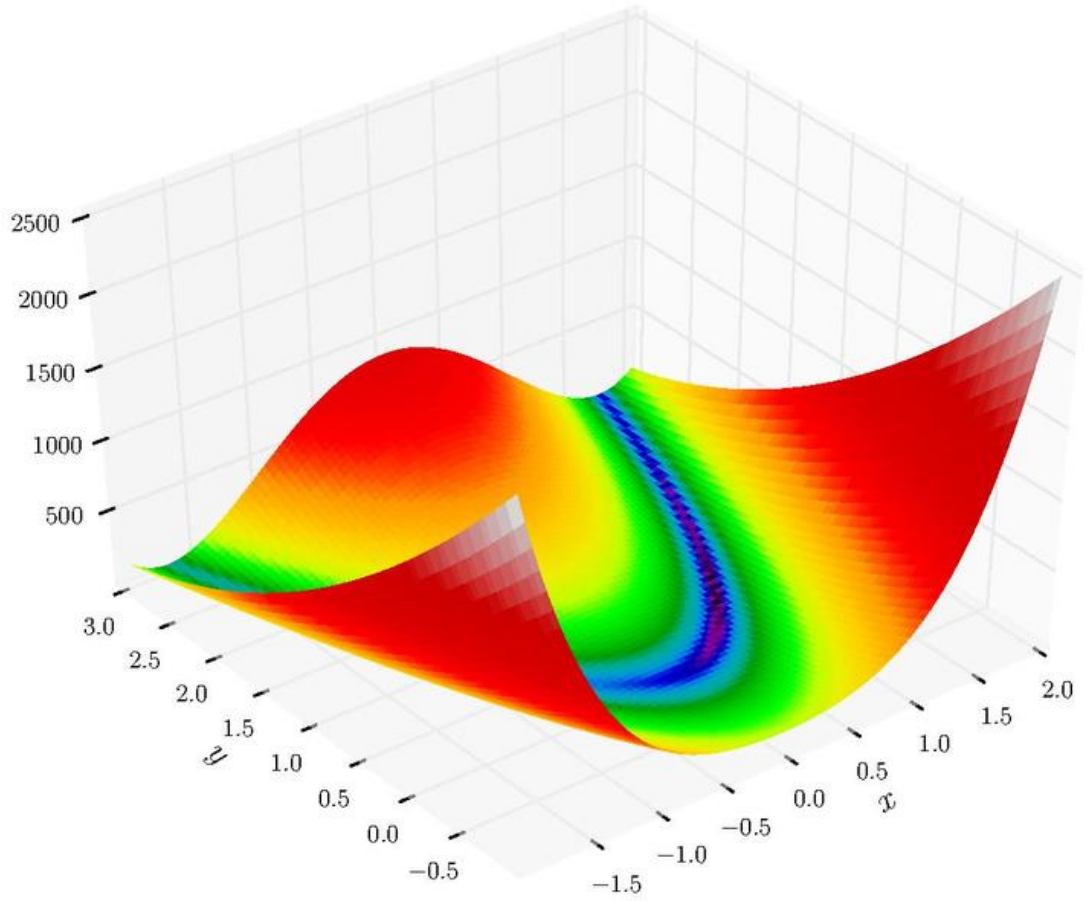


Porównanie otrzymanych wyników

W poniższej tabeli pokazujemy wartość bezwględną różnicy średnich w otrzymanych wynikach

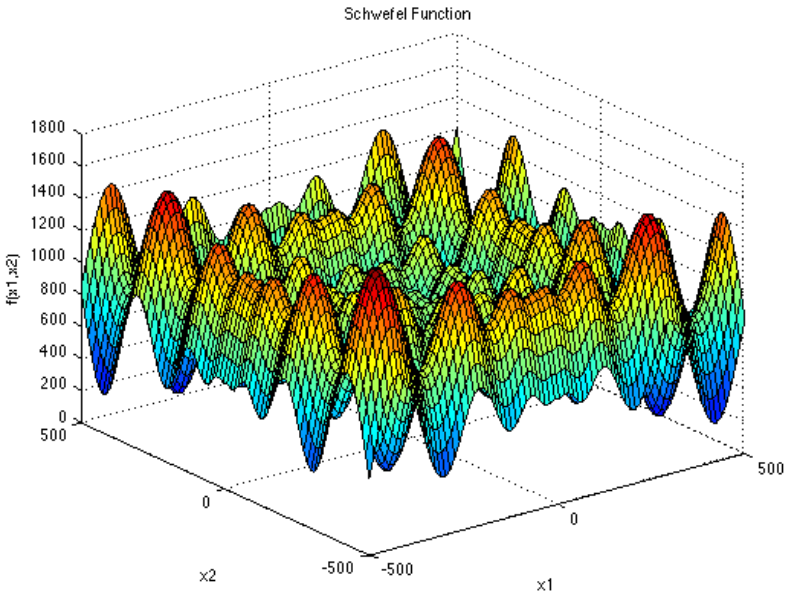
	Rosenbrock2D	Rosenbrock10D	Rosenbrock20D	Schwefel2D	Schwefel10D	Schwefel20D
PRS	12.2465557	7574228.774	68376691.03	-804.3137	-1934.2887	-2771.5036
GA	0.2512967	1547.454	16785.73	-647.5771	-1477.8303	-2035.6735
Difference	11.9952590	7572681.320	68359905.30	156.7366	456.4585	735.8301
Global minimum	0.0000000	0.000	0.00	-837.9658	-4189.8290	-8379.6580

Dla funkcji Rosenbrock, możemy zauważyć że zawsze, dużo lepiej radził sobie GA niż PRS w wyznaczaniu minimum. Jest to spowodowane specyfikacją funkcji Rosenbrock: - *Funkcja Rosenbrocka*



Jak widać nie posiada lokalnych minimów, dzięki czemu dla algorytmu GA łatwo jest ‘iść wzdłuż’ antygradientu żeby znaleźć globalne minimum. Natomiast algorytm PRS dla funkcji 2 zmiennych osiąga dobry wynik, z powodu małej dziedziny co przekłada się na mała liczbę wszystkich możliwych punktów. Dla funkcji 10 i 20 zmiennych można zaobserwować że liczba punktów jest już zbyt duża żeby trafnie znalazł minimum. Możemy również zauważyć po wykresach że dla funkcji Rosenbrock wyniki GA posiadają mniejsze odchylenie, są dość zbliżone do siebie, natomiast dla PRS odchylenie rośnie wzrast ze wzrostem wymiarów z powodu ilości punktów.

Dla funkcji Schwefela sprawa wygląda trochę inaczej. I algorytm PRS i GA radzą sobie podobnie, dla funkcji 2 zmiennych są dośc blisko minimum, natomiast dla funkcji 10 i 20 zmiennych znajdowane minimum odbiega już od rzeczywistego. Zacznijmy od specyfikacji Schwefela: - *Funkcja Schewefla*



Możemy zauważyć że funkcja posiada wiele minimum lokalnych ale istnieje tylko jedno minimum globalne. W związku z tym algorytm genetyczny GA często ‘wpadał’ w lokalne minimum nie potrafiąc z niego wyjść stąd osiągał takie słabe wyniki. Natomiast algorytm PRS losując z dużej dziedziny i dla dużych wymiarów też nie był w stanie osiągać dobrych wyników. Po wykresach możemy zauważyć, że GA posiada tym razem większe odchylenie od PRS.

Analiza statystyczna

Używamy funkcji t.test i konstruujemy 95-procentowe przedziały ufności dla różnic średnich. Dla testów statystycznych za hipotezę zerową przyjmujemy stwierdzenie, że średnie z obu algorytmów są równe. Hipoteza alternatywna to oczywiście stwierdzenie, że są one różne.

	Rosenbrock2D	**Rosenbrock10D**	**Rosenbrock20D**	**Schwefel2D**	**Schwefel10D**	**Schwefel20D**
Conf Int	7.258 - 16.733	6671949.192 - 8473413.448	64718964.485 - 72000846.117	-185.489 - -127.984	-557.688 - -355.229	-860.89 - -610.77
P-value	2.27192006032392e-06	2.16323045683406e-30	1.14152937161658e-59	2.04950137586318e-18	2.31503538266822e-14	2.93660142785995e-20
Mean Of PRS	12.247	7574228.774	68376691.033	-804.314	-1934.289	-2771.504
Mean of GA	0.251	1547.454	16785.732	-647.577	-1477.83	-2035.673
Standard error	2.387	453891.106	1834719.339	14.489	51.011	63.019

Jak widać na powyższym zestawieniu nasze p-wartości są znacznie mniejsze od przyjętego poziomu istotności (0.05). Zatem każdą z postawionych hipotez zerowych (mówiących o równości średnich) musimy odrzucić.

Podsumowanie

Powyższe porównanie pozwoliło nam zaobserwować jak bardzo róznymi algorytmami minimalizacji stochastycznej są algorytmy PRS i GA. W dodatku zaobserwowaliśmy ciekawe zjawisko radzenia sobie algorytmu genetycznego w dwóch odmiennych rodzajach funkcji. Kiedy analizowana funkcja ma wiele minimów lokalnych algorytm genetyczny nie radzi sobie z odnalezieniem minimum globalnego. W takim wypadku lepszym pomysłem będzie użycie algorytmu PRS, który przy odrobinie szczęścia będzie mógł się znaleźć naprawdę niedaleko faktycznego minimum.