

Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych
Instytut Informatyki

Rok akademicki 2013/2014

Praca dyplomowa magisterska

Adam Stelmaszczyk

**DE/mid – nowy wariant algorytmu
ewolucji różnicowej wykorzystujący
punkt środkowy populacji**

Opiekun pracy:
prof. nzw. dr hab. Jarosław Arabas

Ocena

.....
Podpis Przewodniczącego
Komisji Egzaminu Dyplomowego



Specjalność: Inżynieria Systemów Informacyjnych

Data urodzenia: 12 stycznia 1990 r.

Data rozpoczęcia studiów: 1 października 2009 r.

Życiorys

Urodziłem się 12 stycznia 1990 roku w Piotrkowie Trybunalskim. Ukończył Szkołę Podstawową nr 33, Gimnazjum nr 11 oraz Liceum Ogólnokształcące im. Jana Kochanowskiego w Warszawie, gdzie uczęszczałem do klasy o profilu matematyczno-fizyczno-informatycznym. W październiku 2009 roku rozpoczęłem studia na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej na kierunku Informatyka. W lutym 2013 roku uzyskałem tytuł inżyniera.

.....
podpis studenta

Egzamin dyplomowy

Złożył egzamin dyplomowy w dn.

Z wynikiem

Ogólny wynik studiów

Dodatkowe wnioski i uwagi Komisji

Streszczenie

W pracy przeanalizowano działanie znanych wariantów algorytmu ewolucji różnicowej – DE/rand oraz DE/best. W celu poprawienia wyników zaproponowano nowy wariant nazwany DE/mid. Żeby porównanie jakości algorytmów było sprawiedliwe, każdemu z wariantów ustalono taki sam zasięg mutacji poprzez dobranie odpowiedniego współczynnika skalującego. Na podstawie przeprowadzonych testów stwierdzono poprawę jakości uzyskiwanych wyników na 6 z 7 wielowymiarowych funkcji testowych z zestawu BBOB 2013.

Słowa kluczowe: *ewolucja różnicowa, algorytm ewolucyjny, genetyczny, optymalizacja globalna*

Abstract

Title: *DE/mid – new variant of differential evolution algorithm using the midpoint of the population*

This thesis analyzes behaviour of known differential evolution variants – DE/rand and DE/best. In order to achieve better performance, new variant named DE/mid has been introduced. To ensure a fair comparison between algorithms, mutation of every algorithm variant had exactly the same range. Tests indicated improvement of results quality on 6 out of 7 high-dimensional testing functions from BBOB 2013 testbed.

Key words: *differential evolution, evolutionary, genetic algorithm, global optimization*

Spis treści

1. Wstęp	1
1.1. Cel pracy	2
1.2. Układ pracy	2
1.3. Słownik pojęć i oznaczeń	2
2. Ewolucja różnicowa	4
2.1. Algorytm ewolucyjny	4
2.2. Algorytm ewolucji różnicowej	5
2.2.1. Selekcja	6
2.2.2. Mutacja	7
2.2.3. Krzyżowanie	7
2.2.4. Inne modyfikacje ewolucji różnicowej	8
3. Algorytm DE/mid – nowy wariant ewolucji różnicowej	9
3.1. Analiza macierzy kowariancji dla podstawowych wariantów DE	9
3.1.1. DE/rand	9
3.1.2. DE/rand/ ∞	11
3.1.3. DE/best	11
3.1.4. DE/best/ ∞	12
3.2. Analiza rozkładu mutantów dla podstawowych wariantów DE	13
3.2.1. Symetria rozkładu mutantów	13
3.2.2. Wpływ liczby par wektorów różnicowych k na rozkład mutantów	15
3.3. Algorytm DE/mid – nowy wariant ewolucji różnicowej	18
3.3.1. Schemat algorytmu DE/mid/ k	18
3.3.2. Analiza macierzy kowariancji generowanych mutantów	19
3.4. Współczynniki skalujące	20
4. Metodyka testowania	21
4.1. Zestaw funkcji testowych BBOB 2013	21
4.1.1. Funkcja numer 15 – Rastrigina	23
4.1.2. Funkcja numer 16 – Weierstrassa	23
4.1.3. Funkcja numer 19 – Griewanka-Rosenbrocka	24
4.1.4. Funkcja numer 20 – Schwefela	25
4.1.5. Funkcja numer 21 – góryki gaussowskie Gallaghera 101-me	26
4.1.6. Funkcja numer 22 – góryki gaussowskie Gallaghera 21-hi	27
4.1.7. Funkcja numer 24 – Lunačka	28
4.2. Porównywanie wyników	29
4.2.1. Testy istotności statystycznej	29
4.2.2. Dystrybuanty empiryczne	31
4.2.3. Procent osobników poza zbiorem dopuszczalnym	31
5. Wyniki eksperymentów	32

5.1.	Parametry eksperymentów	32
5.2.	Implementacja	32
5.3.	Wnioski	33
5.3.1.	Wpływ liczby wymiarów na jakość rozwiązań	35
5.3.2.	Wpływ optymalizowanych funkcji na wyniki porównań	36
5.3.3.	Wpływ liczby wymiarów na liczbę osobników poza zbiorem dopuszczalnym	37
5.3.4.	Wpływ liczby osobników poza zbiorem dopuszczalnym na jakość rozwiązań	38
5.3.5.	Wpływ liczby par wektów różnicowych k na jakość wyników	39
5.3.6.	Porównanie z algorytmami biorącymi udział w konkursie BBOB 2013	39
6.	Podsumowanie	43
A.	Załączniki	45
B.	Pełne wyniki eksperymentów	46
B.1.	Wyniki uzyskane dla 10 wymiarów	46
B.2.	Wyniki uzyskane dla 20 wymiarów	52
B.3.	Wyniki uzyskane dla 40 wymiarów	57
B.4.	Wyniki uzyskane dla 80 wymiarów	63
	Bibliografia	69

1. Wstęp

Optymalizacja globalna polega na znalezieniu ekstremum globalnego (minimum bądź maksimum) pewnej funkcji, zwanej funkcją celu. Funkcja ta jest „czarną skrzynką”, tzn. nie można zakładać znajomości tej funkcji podczas projektowania algorytmu optymalizacji. W związku z tym analityczne wyznaczenie ekstremów nie jest możliwe. W przypadku skomplikowanych funkcji znalezienie dokładnego optimum może być bardzo czasochłonne, a w praktyce wręcz niewykonalne. Wówczas celem jest znalezienie możliwie najlepszego rozwiązania suboptimalnego. Algorytm optymalizujący wybiera pewne rozwiązanie, następnie poddaje to rozwiązanie ocenie przez funkcję oceniającą, w wyniku czego otrzymuje liczbową wartość (jakość, dobroć) podanego rozwiązania. Proces optymalizacji przebiega w sposób iteracyjny, tzn. nowe rozwiązania powstają w wyniku przetworzenia poprzednich.

Przykładowo, „czarną skrzynką” może być program (symulator), który, po podaniu parametrów na wejście, przeprowadza jeden eksperyment i zwraca np. liczbę watów wygenerowanych przez panele słoneczne. Parametrami wejściowymi mogą być kąty nachylenia i prędkości obrotu paneli. Panele te umieszczone są na sztucznym satelicie, który krąży po orbicie ziemskiej. Przez to niektóre panele rzucają cień na inne, jeśli są źle ustalone. Panele również nie mogą być zbyt długo wystawione na działanie promieni słonecznych, ponieważ może dojść do ich uszkodzenia. To wprowadza kłopotliwe ograniczenia w przestrzeni przeszukiwań. Symulator przeprowadza kosztowne obliczenia, żeby zwrócić całkowitą liczbę watów wygenerowanych przez panele w ciągu pełnego obiegu satelity wokół Ziemi. Zadanie optymalizacji polega w tym przypadku na doborze takich parametrów wejściowych, które powodują wygenerowanie jak największej liczby watów przez wszystkie panele.

Ewolucja różnicowa (ang. differential evolution, DE) jest algorytmem przeznaczonym do optymalizacji globalnej w przestrzeniach ciągłych [12]. Algorytm ten może mieć wiele postaci (wariantów), w zależności od używanych operatorów genetycznych – selekcji, mutacji oraz krzyżowania. Ewolucja różnicowa jest atrakcyjnym algorytmem ze względu na nieskomplikowaną definicję i małą liczbę parametrów. Jednocześnie jej

efektywność jest porównywalna do innych metod optymalizacji startujących w konkursach typu BBOB [6] czy CEC [13].

1.1. Cel pracy

Celem pracy była poprawa efektywności algorytmu ewolucji różnicowej. Wiązało się to ze szczegółową analizą i przetestowaniem istniejących wariantów algorytmu, a następnie zapronowaniem lepszej metody. Skoncentrowano się na odpowiednim doborze operatora selekcji. Nowy wariant ewolucji różnicowej, nazwany DE/mid, został porównany do dwóch znanych odmian – DE/rand oraz DE/best. Wyniki eksperymentów wskazały na znaczną przewagę DE/mid na 6 z 7 wielowymiarowych funkcji testowych.

1.2. Układ pracy

Rozdział pierwszy stanowi wprowadzenie do tematu pracy, definiując jej cel oraz podstawowe pojęcia. W rozdziale drugim wyjaśniono podstawy działania algorytmów ewolucyjnych. Następnie opisano schemat ewolucji różnicowej, a także trzy operatory genetyczne – selekcję, mutację oraz krzyżowanie. W rozdziale trzecim zaprezentowano nowy wariant ewolucji różnicowej nazwanej DE/mid. W części tej dokonano również analizy macierzy kowariancji rozkładu mutantów dla różnych wariantów ewolucji różnicowej. Na tej podstawie wyprowadzono wzory na współczynniki skalujące – istotny parametr algorytmu ewolucji różnicowej mający bezpośredni wpływ na zasięg mutacji. W rozdziale czwartym opisano metodykę testowania, jakich dokładnie funkcji testowych używano. Wyjaśniono również w jaki sposób testy zostały zaimplementowane oraz jak została sprawdzona ich poprawność. Na zakończenie rozdziału opisano metodę porównywania wyników eksperymentów – zwrócono uwagę na testy istotności statystycznej, dystrybuanty empiryczne oraz liczbę osobników poza zbiorem dopuszczalnym. W rozdziale piątym zaprezentowano w sposób zbiorczy wyniki wszystkich eksperymentów, przedstawiono również wnioski z nich płynące. Pracę kończy podsumowanie oraz wskazanie możliwych kierunków dalszego rozwoju.

1.3. Słownik pojęć i oznaczeń

Osobnik, punkt, wektor, rozwiązanie – podstawowy element przestrzeni przeszukiwań, przetwarzany przez algorytm ewolucji różnicowej

Populacja – zbiór osobników, przetwarzany w pojedynczej iteracji algorytmu

DE (ang. Differential Evolution) – ewolucja różnicowa

DE/rand – klasyczny wariant DE wybierający losowy punkt w selekcji

DE/best – klasyczny wariant DE wybierający najlepszy punkt w selekcji

DE/mid – zaproponowany w pracy wariant DE wybierający punkt środkowy w selekcji

DE/X/k/Z – skrócony zapis nazwy wariantu DE, X oznacza rodzaj selekcji (np. rand, best, mid), k oznacza liczbę wektorów różnicowych stosowanych w mutacji (domyślnie 1), zaś Z to rodzaj krzyżowania (domyślnie bin – dwumianowe).

BBOB (ang. Black-Box Optimization Benchmarking) – nazwa warsztatów poświęconych testowaniu algorytmów optymalizacji. Warsztaty te odbywały się w latach 2009, 2010, 2012 oraz 2013. Zmieniały się algorytmy startujące w konkursie, nie zmieniał się sposób ich testowania (opisany w rozdziale 4.1).

k – liczba wektorów różnicowych używanych w mutacji

μ – liczba osobników w populacji

D – liczba cech osobnika, również wymiar optymalizowanej funkcji

P^t – populacja o numerze t

P_i^t – osobnik o indeksie i w populacji P^t

O^t – populacja potomków u numerze t

O_i^t – osobnik o indeksie i w populacji potomków O^t

v_i – punkt wybrany w selekcji w iteracji i

u_i – mutant w iteracji i

o_i – potomek po krzyżowaniu w iteracji i

F (ang. Factor) – współczynnik skalujący, parametr ewolucji różnicowej sterujący zasięgiem mutacji

CR (ang. Crossover Rate) – prawdopodobieństwo krzyżowania, parametr algorytmów ewolucyjnych sterujący częstością krzyżowań

f – funkcja celu, przystosowania, oceny, optymalizowana funkcja, funkcja testowa

f_{opt} – optymalna wartość f , tzn. minimum podczas minimalizacji lub maksimum podczas maksymalizacji

$C[u]$ – macierz kowariancji rozkładu zmiennej losowej u

2. Ewolucja różnicowa

2.1. Algorytm ewolucyjny

Algorytmy ewolucyjne to rodzaj algorytmów, w których wykorzystywane są operatory genetyczne (selekcja, mutacja, krzyżowanie) na zbiorach punktów w przestrzeni przeszukiwań nazywanych populacjami [1]. Celem działania algorytmu jest wyznaczenie ekstremum globalnego (minimum bądź maksimum) optymalizowanej funkcji. Zasadę działania typowego algorytmu ewolucyjnego przedstawiono na listingu 1.

Algorytm 1 Algorytm ewolucyjny

```
1: Inicjalizacja parametrów  $CR, \mu$ 
2: Inicjalizacja populacji początkowej  $P^1 \leftarrow \{P_1^1, P_2^1, \dots, P_\mu^1\}$ 
3:  $t \leftarrow 1$ 
4: while kryterium stopu nie zostało spełnione do
5:   for all  $i \in \{1, 2, \dots, \mu\}$  do
6:     if  $\mathcal{U}(0, 1) < CR$  then
7:        $O_i^t \leftarrow \text{mutacja(krzyżowanie(selekcia}(P(t)), selekcja(P(t))))$ 
8:     else
9:        $O_i^t \leftarrow \text{mutacja(selekcia}(P(t)))$ 
10:    end if
11:   end for
12:    $P^{t+1} \leftarrow \text{sukcesja}(P^t, O^t)$ 
13:    $t \leftarrow t + 1$ 
14: end while
```

$\mathcal{U}(0, 1)$ to realizacja zmiennej losowej o rozkładzie jednostajnym na przedziale od 0 do 1. Algorytm ewolucyjny w podstawowej postaci ma trzy parametry wejściowe. Są nimi: CR – prawdopodobieństwo krzyżowania, μ – liczba osobników w populacji oraz F – współczynnik skalujący wykorzystywany w mutacji. Po nadaniu wartości początkowych parametrom, następuje inicjalizacja populacji początkowej. Jeśli zupełnie nie wiadomo, gdzie może znajdować się poszukiwane optimum, populację początkową generuje się w sposób losowy z rozkładem jednostajnym w zbiorze dopuszczalnym.

Po inicjalizacji populacji początkowej następuje główna pętla programu. Wykonywana jest ona dopóty, dopóki nie zajdzie tzw. kryterium stopu. Kryterium stopu

może być np. maksymalna liczba iteracji lub wywołań funkcji oceny. Algorytm można zatrzymać również wtedy, gdy znalezione rozwiązanie jest zadowalające.

W głównej pętli programu przetwarzana jest bieżąca populacja. Wykonywana jest kolejna pętla, tym razem iterująca kolejno po wszystkich osobnikach (rozwiązaniach) z bieżącej populacji. W niej losowana jest liczba od 0 do 1 i jeśli jest ona mniejsza od ustalonego CR – dokonywana jest selekcja dwóch punktów, ich krzyżowanie, a następnie mutacja. W przeciwnym przypadku krzyżowanie jest pomijane. Po wygenerowaniu populacji potomków O^t następuje sukcesja, czyli wybór punktów, które przechodzą do następnej populacji P^{t+1} . Zwiększany jest licznik populacji (czasu) t , sprawdzane jest kryterium stopu i jeśli nie jest spełnione – wykonywana jest kolejna pętla.

Kolejnym problemem jest dobór parametrów dla algorytmu ewolucyjnego. Istnieją pewne porady, np. stosowana w tej pracy reguła $\mu = 10D$, jednak w ogólnym przypadku nie wiadomo jaki rozmiar populacji czy jakie prawdopodobieństwo krzyżowania CR sprawdzi się najlepiej. Duży problem również stanowi dobór parametrów dla mutacji. Z pomocą przychodzą metody samoadaptacji, które same dostrajają parametry algorytmu w zależności od stanu w jakim się znajdują, np. w zależności od zawartości populacji [7]. Ewolucja różnicowa radzi sobie z problemem doboru parametrów właśnie dzięki pewnemu rodzajowi samoadaptacji.

2.2. Algorytm ewolucji różnicowej

Ewolucja różnicowa jest algorytmem ewolucyjnym służącym do optymalizacji w przestrzeniach ciągłych zaproponowanym w 1995 roku. [12] Sposób działania algorytmu przedstawiono na listingu 2.

Algorytm 2 Ewolucja różnicowa

```

1: Inicjalizacja parametrów  $CR, F, \mu$ 
2: Inicjalizacja populacji początkowej  $P^1 \leftarrow \{P_1^1, P_2^1, \dots, P_\mu^1\}$ 
3:  $t \leftarrow 1$ 
4: while kryterium stopu nie zostało spełnione do
5:   for all  $i \in \{1, 2, \dots, \mu\}$  do
6:      $v_i \leftarrow$  selekcja( $P_i^t$ )
7:      $u_i \leftarrow$  mutacja( $v_i, P_i^t, F$ )
8:      $o_i \leftarrow$  krzyżowanie( $u_i, P_i^t, CR$ )
9:     if  $f(o_i) \leq f(P_i^t)$  then
10:       $P_i^{t+1} \leftarrow o_i$ 
11:    else
12:       $P_i^{t+1} \leftarrow P_i^t$ 
13:    end if
14:   end for
15: end while
16: return  $\arg \min_i f(P_i^t)$ 

```

Bardzo istotną różnicą pomiędzy ewolucją różnicową a typowym algorytmem ewolucyjnym jest sposób mutacji. W ewolucji różnicowej punkty są od siebie odejmowane, w wyniku czego powstają wektory różnicowe. Następnie mutowane punkty są przesuwane o te wektory. Natomiast w typowym algorytmie ewolucyjnym odejmowanie nie występuje – mutacja polega na dodawaniu losowego wektora do mutowanych punktów. Kolejność wykonywania operatorów mutacji oraz krzyżowania w ewolucji różnicowej jest również inna niż w typowym algorytmie ewolucyjnym. W algorytmie ewolucyjnym najpierw wykonywane jest krzyżowanie, potem mutacja. W ewolucji różnicowej – na odwrót. Kolejną różnicą jest konkretyzacja operatora sukcesji. W ewolucji różnicowej, po wygenerowaniu potomka o_i jest on od razu porównywany ze swoim rodzicem. Jeśli okaże się lepszy – zastępuje go w kolejnej populacji. Powoduje to silny nacisk selektywny – dalej przechodzą jedynie dzieci lepiej przystosowane od swoich rodziców.

W tej pracy stosowany jest zapis nazwy algorytmu w postaci DE/X/k/Z, gdzie X oznacza rodzaj selekcji (np. rand, best, mid), k oznacza liczbę wektorów różnicowych stosowanych w mutacji (domyślnie 1), zaś Z to rodzaj krzyżowania (domyślnie bin – dwumianowe).

2.2.1. Selekcja

Operator selekcji na wejściu otrzymuje bieżącą populację i zwraca jedno rozwiązanie. W tej pracy zwrócono szczególną uwagę na właściwy dobór tego operatora. Skrót DE/rand oznacza ewolucję różnicową z losowym typem selekcji, tzn. wybierany jest

jeden losowy punkt z całej populacji zgodnie z rozkładem jednostajnym: W wariantie DE/best wybierany jest punkt najlepszy. Zaproponowany w tej pracy DE/mid wybiera punkt środkowy populacji. DE/rand-to-best stanowi połączenie DE/rand oraz DE/best – punkt wybierany w selekcji v_i to liniowa kombinacja punktu najlepszego x_* oraz losowego x_{i_1} z wagami λ oraz $(1 - \lambda)$:

$$v_i = \lambda x_* + (1 - \lambda)x_{i_1}$$

gdzie $\lambda \in (0, 1)$. Wariant DE/current-to-best jest podobny DE/rand-to-best, z tym, że zamiast punktu losowego występuje punkt bieżący x_i . Natomiast w DE/current-to-rand punktem wybieranym w selekcji jest liniowa kombinacja punktu bieżącego x_i oraz losowego x_{i_1} [16].

2.2.2. Mutacja

Operator mutacji na wejściu otrzymuje jedno rozwiązanie i zwraca również jedno. Jako trzeci człon skróconej nazwy algorytmu podawana jest liczba wektorów różnicowych używanych w mutacji – k . Np. DE/rand/1 oznacza losową selekcję z jednym wektorem różnicowym w mutacji. Ten podstawowy wariant algorytmu nazywany jest również „klasyczną” ewolucją różnicową.

2.2.3. Krzyżowanie

Operator krzyżowania na wejściu otrzymuje dwa rozwiązania rodzicielskie, a zwraca jedno potomne. W ewolucji różnicowej jednym z rodziców jest mutant. Często wykorzystywany rodzajem krzyżowania jest krzyżowanie wymieniające, w którym część rozwiązania jest brana od jednego z rodziców, a pozostała część od drugiego. Krzyżowanie wymieniające ma dwie odmiany – bin (ang. binomial – dwumianowe) oraz exp (ang. exponential – wykładnicze). Sposób działania krzyżowania dwumianowego przedstawiono na listingu 3.

Algorytm 3 Krzyżowanie dwumianowe (bin)

```

1: Wejście: dwa rozwiązania rodzicielskie  $x_1, x_2$ 
2:  $o \leftarrow x_1$ 
3: for all  $d \in \{1, 2, \dots, D\}$  do
4:   if  $\mathcal{U}(0, 1) < CR$  then
5:      $o[d] \leftarrow x_2[d]$ 
6:   end if
7: end for
8: return  $o$ 
```

$\mathcal{U}(0, 1)$ to realizacja zmiennej losowej o rozkładzie jednostajnym na przedziale od 0 do 1. CR (ang. Crossover Rate) to sparametryzowane prawdopodobieństwo krzyżowania. Sposób działania krzyżowania wykładowicze przedstawiono na listingu 4.

Algorytm 4 Krzyżowanie wykładowicze (exp)

```

1: Wejście: dwa rozwiązania rodzicielskie  $x_1, x_2$ 
2:  $o \leftarrow x_1$ 
3:  $d \leftarrow 1$ 
4: while  $d \leq D$  and  $\mathcal{U}(0, 1) < CR$  do
5:    $o[d] \leftarrow x_2[d]$ 
6:    $d \leftarrow d + 1$ 
7: end while
8: return  $o$ 
```

W tej pracy w każdym z wariantów ewolucji różnicowej stosowano krzyżowanie dwumianowe (bin).

2.2.4. Inne modyfikacje ewolucji różnicowej

Można wprowadzić olbrzymią liczbę modyfikacji do podstawowego algorytmu ewolucji różnicowej. Oprócz zmian w podstawowych operatorach genetycznych, możliwe jest również dodawanie reguł adaptacyjnych. Przykładami mogą być algorytmy SaDE oraz jDE wykorzystujące metody losowego wyboru wartości parametrów [7]. Istnieją również podejścia hybrydowe, np. w algorytmach JADE [18] oraz DEPSO [19] wykorzystano elementy znane w algorytmach optymalizacji rojem cząstek (ang. PSO – Particle Swarm Optimization).

3. Algorytm DE/mid – nowy wariant ewolucji różnicowej

W tym rozdziale przeanalizowano rozkłady mutantów różnych wariantów ewolucji różnicowej. Dla ustalenia tego samego zasięgu mutacji kluczowe było wyprowadzenie odpowiednich współczynników skalujących dla każdego z wariantów DE. Następnie przedstawiono nowy wariant ewolucji różnicowej nazwanej DE/mid.

3.1. Analiza macierzy kowariancji dla podstawowych wariantów DE

W poniższych podrozdziałach przedstawiono operatory mutacji dla DE/rand/k, DE/best/k, DE/mid/k oraz wyprowadzono wzory na współczynniki skalujące F . We wzorach, dla zwiększenia czytelności, pomijano indeks górnny t dla populacji, np. stosowano zapis P_{i_1} zamiast $P_{i_1}^t$.

3.1.1. DE/rand

W DE/rand/1, i -ty mutant w populacji P o μ osobnikach powstaje w następujący sposób [2]:

$$u_i = P_{i_1} + F(P_{i_2} - P_{i_3}) \quad (3.1)$$

i_1, i_2, i_3 to indeksy wylosowane zgodnie z rozkładem jednostajnym ze zbioru $\{1, 2, \dots, \mu\}$. Zatem $P_{i_1}, P_{i_2}, P_{i_3}$ to rozwiązania wylosowane zgodnie z rozkładem jednostajnym z populacji P . $F \in \mathbb{R}_+$ to współczynnik skalujący zasięg mutacji.

DE/rand/1 można uogólnić na DE/rand/k, w którym dodawanych jest $k \in \mathbb{N}$ wektorów różnicowych:

$$u'_i = P_{i_1} + F_k \sum_{j=1}^k (P_{i_{2j}} - P_{i_{2j+1}}) \quad (3.2)$$

Podobnie jak poprzednio, $i_1, i_2, \dots, i_{2k+1}$ to indeksy wylosowane zgodnie z rozkładem jednostajnym ze zbioru $\{1, 2, \dots, \mu\}$. Zatem $P_{i_1}, P_{i_2}, \dots, P_{2k+1}$ to rozwiązania wylosowane zgodnie z rozkładem jednostajnym z populacji P . $F_k \in \mathbb{R}_+$ to współczynnik skalujący dla DE/rand/k.

Żeby zasięg mutacji nie zmieniał się wraz ze wzrostem k , macierz kowariancji mutanta u'_i musi być taka sama jak macierz kowariancji mutanta u_i . Można to osiągnąć tak dobierając F_k , aby było spełnione równanie:

$$\mathbf{C}[u_i] = \mathbf{C}[u'_i] \quad (3.3)$$

$\mathbf{C}[u_i]$ jest macierzą kowariancji rozkładu zmiennej losowej u_i . Rozwijając lewą stronę równania (3.3) i korzystając z tego, że osobniki są niezależne od siebie:

$$\mathbf{C}[u_i] \stackrel{(3.1)}{=} \mathbf{C}[P_{i_1} + F(P_{i_2} - P_{i_3})] = \mathbf{C}[P_{i_1}] + F^2(\mathbf{C}[P_{i_2}] + \mathbf{C}[P_{i_3}])$$

Każdy osobnik ma taki sam rozkład prawdopodobieństwa. Macierz kowariancji punktów losowanych z P ma teorytyczną wartość $\mathbf{C}[P]$, równą empirycznej macierzy kowariancji populacji P . Zatem dla DE/rand/1 macierz kowariancji mutanta wynosi:

$$\mathbf{C}[u_i] = \mathbf{C}[P] + F^2(\mathbf{C}[P] + \mathbf{C}[P]) = \mathbf{C}[P](2F^2 + 1) \quad (3.4)$$

Rozwijając prawą stronę równania (3.3), w przypadku DE/rand/k macierz kowariancji mutanta wynosi:

$$\begin{aligned} \mathbf{C}[u'_i] &\stackrel{(3.2)}{=} \mathbf{C}[P_{i_1} + F_k \sum_{j=1}^k (P_{i_{2j}} - P_{i_{2j+1}})] = \mathbf{C}[P_{i_1}] + F_k^2 \mathbf{C}[\sum_{j=1}^k (P_{i_{2j}} - P_{i_{2j+1}})] \\ &= \mathbf{C}[P_{i_1}] + F_k^2 \mathbf{C}[\sum_{j=2}^{2k+1} P_{i_j}] = \mathbf{C}[P](2kF_k^2 + 1) \end{aligned} \quad (3.5)$$

Przeryównując (3.4) oraz (3.5):

$$\mathbf{C}[P](2F^2 + 1) = \mathbf{C}[P](2kF_k^2 + 1)$$

Przy założeniu, że $\mathbf{C}[P] \neq \mathbf{0}$:

$$F^2 = kF_k^2$$

Obie strony są nieujemne, więc ostatecznie:

$$F_k = \frac{F}{\sqrt{k}}$$

Zatem, dla zachowania zasięgu mutacji, współczynnik skalowania dla DE/rand/k powinien być \sqrt{k} razy mniejszy od współczynnika skalowania dla DE/rand/1. Po wstawieniu powyższego wyniku do (3.2), wzór na mutanta dla DE/rand/k można zapisać następująco:

$$u'_i = P_{i_1} + \frac{F}{\sqrt{k}} \sum_{j=1}^k (P_{i_{2j}} - P_{i_{2j+1}})$$

3.1.2. DE/rand/ ∞

Wariant DE/rand/ ∞ został zaproponowany w [2] i nie jest wariantem klasycznym. Zgodnie z centralnym twierdzeniem granicznym, część $\frac{1}{\sqrt{k}} \sum_{j=1}^k (P_{i_{2j}} - P_{i_{2j+1}})$ zbiega według rozkładu do $\mathcal{N}(0, C[P])$ gdy $k \rightarrow \infty$. Dzięki temu, równanie mutanta DE/rand/ ∞ można zapisać jako:

$$u_i^\infty = P_{i_1} + F_\infty \cdot v_\infty$$

gdzie $v_\infty \sim \mathcal{N}(0, C[P])$. Współczynnik F_∞ zapewniający równość macierzy kowariancji dla wariantów DE/rand/1 oraz DE/rand/ ∞ można wyznaczyć następująco:

$$\begin{aligned} C[u_i^\infty] &= C[u_i] \\ C[P_{i_1} + F_\infty \cdot v_\infty] &= C[P](2F^2 + 1) \\ C[P] + C[F_\infty \cdot v_\infty] &= C[P](2F^2 + 1) \\ C[F_\infty \cdot v_\infty] &= 2F^2 C[P] \\ F_\infty^2 C[P] &= 2F^2 C[P] \\ F_\infty^2 &= 2F^2 \\ F_\infty &= \sqrt{2}F \end{aligned}$$

3.1.3. DE/best

W DE/best/k, i -ty mutant w populacji P o μ osobnikach powstaje w następujący sposób:

$$u_i^* = P_* + F_* \sum_{j=1}^k (P_{i_{2j}} - P_{i_{2j+1}}) \quad (3.6)$$

gdzie P_* to najlepszy osobnik w populacji P . Formalnie:

$$P_* = \arg \min_i f(P_i) \quad (3.7)$$

Podobnie jak w przypadku DE/rand/k, $i_2, i_3, \dots, i_{2k+1}$ to indeksy wylosowane zgodnie z rozkładem jednostajnym ze zbioru $\{1, 2, \dots, \mu\}$. $F_* \in \mathbb{R}_+$ to współczynnik skalujący dla DE/best/k.

Żeby zasięg mutacji nie zmieniał się wraz ze wzrostem k , macierz kowariancji mutantu u_i^* musi być taka sama jak macierz kowariancji mutantu u_i . Można to osiągnąć tak dobierając F_k , aby było spełnione równanie:

$$C[u_i] = C[u_i^*] \quad (3.8)$$

Rozwijając prawą stronę równania (3.8), macierz kowariancji mutanta dla DE/best/k wynosi:

$$\begin{aligned} \mathbf{C}[u_i^*] &\stackrel{(3.6)}{=} \mathbf{C}[P_* + F_* \sum_{j=1}^k (P_{i_{2j}} - P_{i_{2j+1}})] = \mathbf{C}[P_*] + F_*^2 \mathbf{C}\left[\sum_{j=1}^k (P_{i_{2j}} - P_{i_{2j+1}})\right] \\ &= \frac{1}{\mu} \mathbf{C}[P] + F_*^2 \mathbf{C}\left[\sum_{j=2}^{2k+1} P_{i_j}\right] = \mathbf{C}[P]\left(2kF_*^2 + \frac{1}{\mu}\right) \end{aligned} \quad (3.9)$$

Przyciągając (3.4) oraz (3.9):

$$\mathbf{C}[P](2F^2 + 1) = \mathbf{C}[P]\left(2kF_*^2 + \frac{1}{\mu}\right)$$

Przy założeniu, że $\mathbf{C}[P] \neq \mathbf{0}$:

$$\begin{aligned} 2F^2 + 1 &= 2kF_*^2 + \frac{1}{\mu} \\ F_*^2 &= \frac{2F^2 + 1 - \frac{1}{\mu}}{2k} \end{aligned}$$

Obie strony są nieujemne, więc ostatecznie:

$$F_* = \sqrt{\frac{2F^2 + 1 - \frac{1}{\mu}}{2k}}$$

3.1.4. DE/best/ ∞

Przechodząc z k do nieskończoności, równanie mutanta DE/best/ ∞ można zapisać jako:

$$u_i^{\infty*} = P_* + F_{\infty*} \cdot v_{\infty*}$$

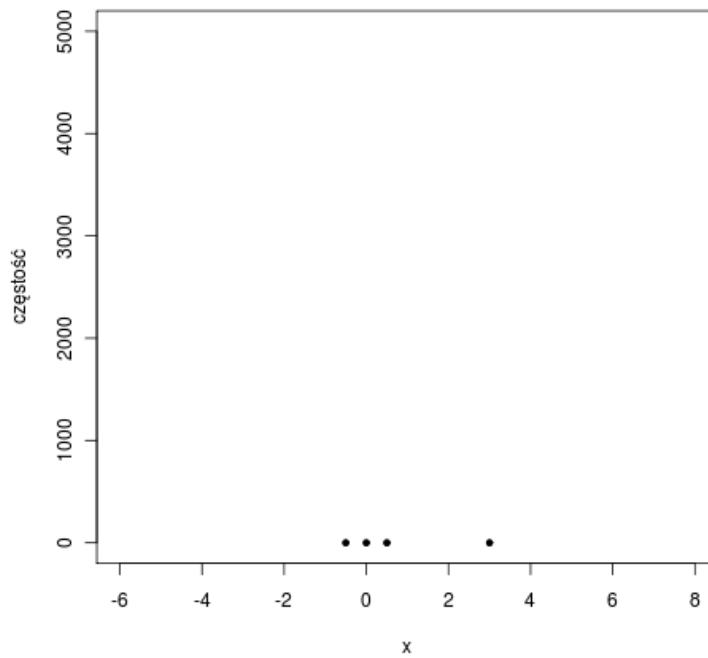
gdzie $v_{\infty*} \sim \mathcal{N}(0, \mathbf{C}[P])$. Współczynnik skalujący $F_{\infty*}$ zapewniający równość macierzy kowariancji dla wariantów DE/rand/1 oraz DE/best/ ∞ można wyznaczyć następująco:

$$\begin{aligned} \mathbf{C}[u_i^{\infty*}] &= \mathbf{C}[u_i] \\ \mathbf{C}[P_* + F_{\infty*} \cdot v_{\infty*}] &= \mathbf{C}[P](2F^2 + 1) \\ \mathbf{C}[P_*] + \mathbf{C}[F_{\infty*} \cdot v_{\infty*}] &= \mathbf{C}[P](2F^2 + 1) \\ \frac{\mathbf{C}[P]}{\mu} + F_{\infty*}^2 \mathbf{C}[P] &= \mathbf{C}[P](2F^2 + 1) \\ F_{\infty*} &= \sqrt{2F^2 + 1 - \frac{1}{\mu}} \end{aligned}$$

3.2. Analiza rozkładu mutantów dla podstawowych wariantów DE

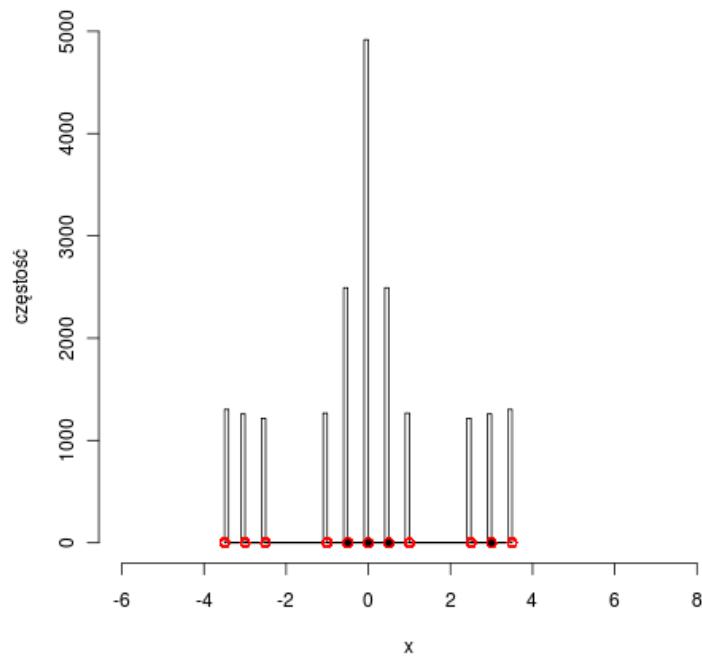
3.2.1. Symetria rozkładu mutantów

W tym podrozdziale przyjęto przestrzeń o jednym wymiarze x . Populację początkową przedstawiono na rysunku 3.1. Są nią 4 punkty o wartościach x równych kolejno $-0.5, 0, 0.5, 3$. Punkty te dobrano specjalnie w taki sposób, aby można było zaobserwować asymetrię rozkładu mutantów dla DE/rand/1 – trzy pierwsze punkty znajdują się w grupie po lewej stronie, po prawej natomiast znajduje się tylko jeden punkt. Współczynnik skalujący F wynosił 1, dla uproszczenia. Dla DE/best/1 przyjęto, że najlepszym punktem był punkt o $x = 0$.

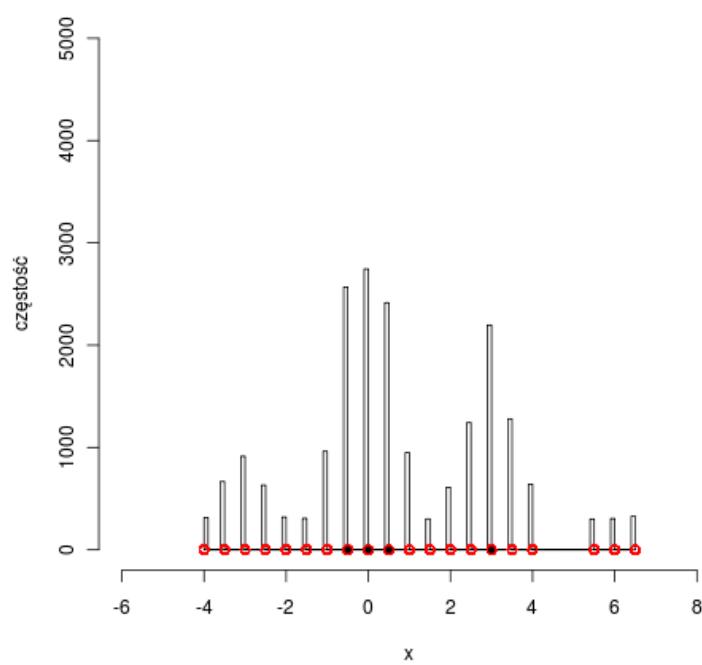


Rysunek 3.1. 6 punktów populacji początkowej

Rysunek 3.2 przedstawia histogram częstości występowania mutantów (zaznaczonych czerwonymi okrągami) dla DE/best/1. Rysunek 3.3 przedstawia histogram częstości występowania mutantów dla DE/rand/1. Dla każdego z algorytmów wygenerowano 20 tysięcy mutantów.



Rysunek 3.2. Histogram mutantów dla DE/best/1



Rysunek 3.3. Histogram mutantów dla DE/rand/1

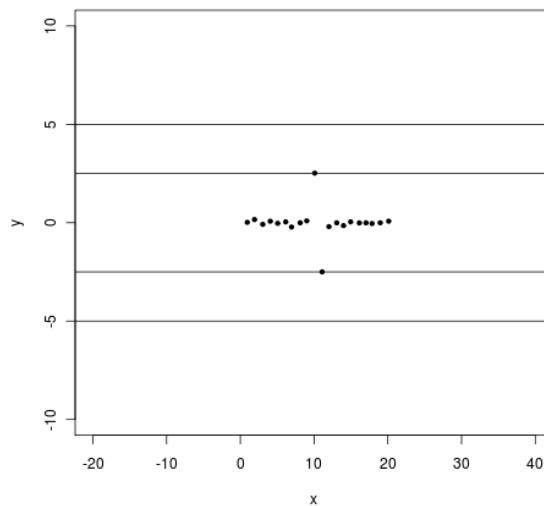
Wektor różnicowy dla DE/rand/1 oraz DE/best/1 ma postać $w = \sum_{j=1}^k (P_{i_{2j}} - P_{i_{2j+1}})$. Rozkład wartości wektora w jest symetryczny względem 0 i dąży do rozkładu Gaussa, gdy k dąży do nieskończoności.

W przypadku DE/best/1 w selekcji wybierany był zawsze jeden ustalony punkt (najlepszy), a więc histogram (empiryczny rozkład) mutantów był symetryczny względem tego punktu.

Natomiast w przypadku DE/rand/1 w selekcji wybierany był punkt losowy. To sprawia, że wektory w są dodawane przeważnie do osobników o współrzędnej x bliskiej 0 (są 3 takie punkty). Z tego powodu histogram mutantów nie posiada osi symetrii i zupełnie różni się od tego uzyskanego dla DE/best/1.

3.2.2. Wpływ liczby par wektorów różnicowych k na rozkład mutantów

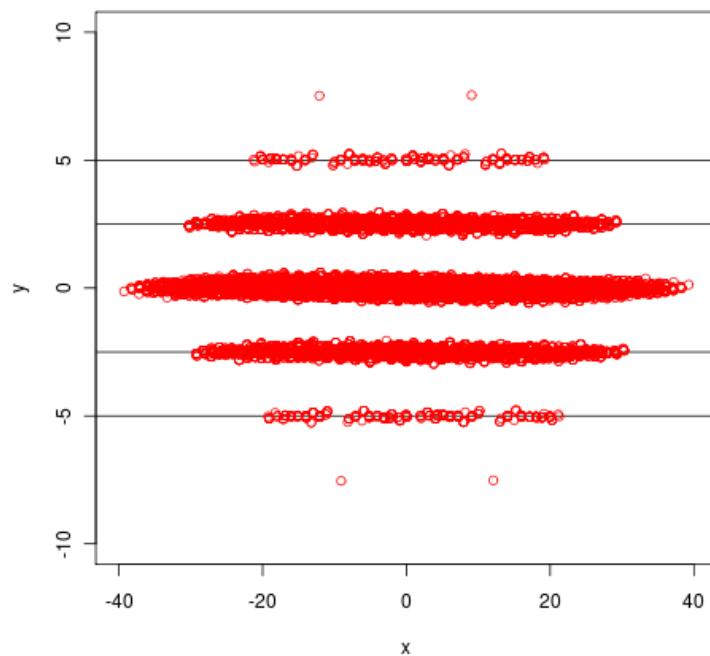
Mutanty, które sięgają na tyle daleko, że umożliwiają wyjście z lokalnego optimum, są bardzo cenne. Ciągłe próbkowanie okolicznych, już zbadanych obszarów, na ogół nie przynosi korzyści. W tym podrozdziale rozważana jest przestrzeń dwuwymiarowa. Populacja początkowa przedstawiona na rysunku 3.4 zawierała 20 punktów, przy czym 18 z nich leżało wokół osi $y = 0$. Jedynie 2 punkty wprowadzały większą różnorodność w wymiarze y : jeden o $y = 2.5$ oraz jeden o $y = -2.5$.



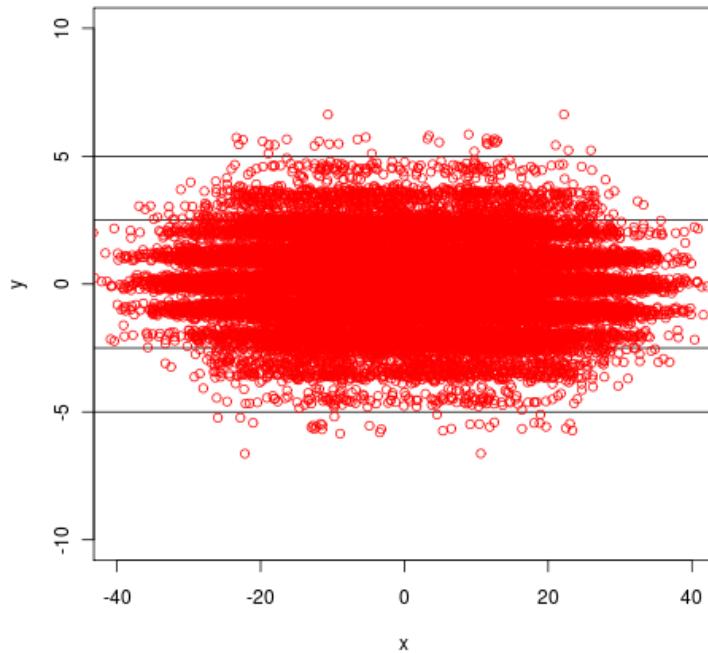
Rysunek 3.4. 20 punktów populacji początkowej

Rysunek 3.5 przedstawia 40 tysięcy mutantów wygenerowanych zgodnie ze schematem DE/rand/1. Rysunek 3.6 przedstawia 40 tysięcy mutantów wygenerowanych zgodnie

ze schematem DE/rand/6. Współczynnik skalujący F dla DE/rand/1 wynosił 1, dla DE/rand/6 (zgodnie z tabelą 3.1) $\frac{1}{\sqrt{6}}$.



Rysunek 3.5. 40 tysięcy mutantów dla DE/rand/1



Rysunek 3.6. 40 tysięcy mutantów dla DE/rand/6

Rozkład mutantów dla $k = 1$ nie przypomina rozkładu normalnego, natomiast dla $k = 6$ już upodabnia się do niego.

Liczba mutantów, dla których $|y| > 5$, znacznie się różniła w przypadku obu porównywanych schematów mutacji. Dla DE/rand/1 było to 352 mutantów, natomiast dla DE/rand/6 jedynie 64.

Ponadto, podczas mutacji DE/rand/1 pojawiły się 2 bardzo oddalone punkty o $y > 7.5$. Źeby tak się zdarzyło, najpierw wylosowany musi zostać osobnik P_{i_1} o $y = 2.5$. Prawdopodobieństwo takiego zdarzenia wynosi $\frac{1}{20}$. Następnie musi zostać wylosowany wektor różnicowy o długości 5. Tylko jedna para punktów dawała taki wektor – pierwszy punkt o $y = 2.5$, drugi punkt o $y = -2.5$. Prawdopodobieństwo wylosowania takiej pary wynosiło $\frac{1}{400}$. Zatem łączne prawdopodobieństwo powstania mutanta o $y = 7.5$ dla DE/rand/1 wynosiło $\frac{1}{8000}$.

Dla DE/rand/6 prawdopodobieństwo powstania mutanta o $y > 7.5$ jest mniejsze. W celu oszacowania, przyjęto dla uproszczenia, że rozkład prawdopodobieństwa mutantów dla DE/rand/6 jest wystarczająco zbliżony do DE/rand/ ∞ . Zgodnie z opisem w rozdziale 3.1.4, na początku wylosowany musi zostać osobnik P_{i_1} o

$y = 2.5$. Następnie musi zostać wygenerowana zmienna losowa v_∞ o rozkładzie normalnym, której wartość w wymiarze y przekroczy $5\sqrt{6} \approx 12.25$. Macierz kowariancji populacji początkowej $C[P]$ wynosiła $\begin{bmatrix} 35.1798210 & -0.2902915 \\ -0.2902915 & 0.6719438 \end{bmatrix}$. Wektor średnich $m = \begin{bmatrix} 10.463749675 & 0.004673205 \end{bmatrix}$. Prawdopodobieństwo wylosowania zmiennej o $y > 5\sqrt{6}$ wynosi $1 - F(\infty, 5\sqrt{6}) \approx 0.1$. F to dystrybuanta wielowymiarowego rozkładu $\mathcal{N}(m, C[P])$. Gdyby pierwszym wylosowanym punktem P_{i_1} nie był ten o $y = 2.5$, tylko np. któryś z punktów leżących blisko osi $y = 0$, wówczas mutacja DE/rand/6 musiałaby wygenerować zmienną o wartości y większej niż $7.5\sqrt{6} \approx 18.37$. Prawdopodobieństwo takiego zdarzenia jest bliskie zeru.

3.3. Algorytm DE/mid – nowy wariant ewolucji różnicowej

W poniższym podrozdziale przedstawiono nowy wariant ewolucji różnicowej wybierający w selekcji punkt środkowy populacji – DE/mid. Wybór punktu środkowego jest uzasadniony wynikiem pracy Zhang'a oraz Sandersona, w której dowodzi się, że środek populacji w DE ma zbieżność w sensie wartości oczekiwanej do wierzchołka paraboli optymalizowanej funkcji [17]. Punkt środkowy jest również wykorzystywany w bardzo efektywnym algorytmie CMA-ES (ang. Covariance Matrix Adaptation Evolution Strategy) [8]. Warianty tego algorytmu od lat zajmują najwyższe miejsca w konkursach typu BBOB [6] czy CEC [13].

3.3.1. Schemat algorytmu DE/mid/k

W DE/mid/k mutant powstaje w następujący sposób:

$$u_i^{\text{mid}} = m + F_{\text{mid}} \sum_{j=1}^k (P_{i_{2j}} - P_{i_{2j+1}}) \quad (3.10)$$

gdzie m to punkt środkowy populacji:

$$m = \frac{1}{\mu} \sum_{j=1}^n P_j \quad (3.11)$$

$F_{\text{mid}} \in \mathbb{R}_+$ jest współczynnikiem skalującym dla DE/mid/k, analogicznym do F w przypadku DE/rand/1 opisanego w rozdziale 3.1.1.

¹ Wartość ta jest tak mała, że w reprezentacji liczb podwójnej precyzji użytych przy wywołaniu funkcji `pmvnorm` języka R została uznana za 0.

3.3.2. Analiza macierzy kowariancji generowanych mutantów

Żeby macierz kowariancji populacji w DE/mid/k była taka sama jak w DE/rand/1, macierz kowariancji mutanta u_i^{mid} musi być taka sama jak macierz kowariancji mutanta u_i . Można to osiągnąć tak dobierając F_{mid} , żeby było spełnione równanie:

$$\mathbf{C}[u_i] = \mathbf{C}[u_i^{\text{mid}}] \quad (3.12)$$

Rozwijając prawą stronę równania (3.12):

$$\begin{aligned} \mathbf{C}[u_i^{\text{mid}}] &\stackrel{(3.10)}{=} \mathbf{C}[m + F_{\text{mid}} \sum_{j=1}^k (P_{i_{2j}} - P_{i_{2j+1}})] \stackrel{(3.11)}{=} \mathbf{C}\left[\frac{1}{\mu} \sum_{j=1}^{\mu} P_j\right] + F_{\text{mid}}^2 \mathbf{C}\left[\sum_{j=1}^k (P_{i_{2j}} - P_{i_{2j+1}})\right] \\ &= \frac{1}{\mu^2} \mu \mathbf{C}[P] + F_{\text{mid}}^2 \mathbf{C}\left[\sum_{j=2}^{2k+1} P\right] = \mathbf{C}[P](2kF_{\text{mid}}^2 + \frac{1}{\mu}) \end{aligned} \quad (3.13)$$

Przyrównując (3.4) do (3.13):

$$\mathbf{C}[P](2F^2 + 1) = \mathbf{C}[P](2kF_{\text{mid}}^2 + \frac{1}{\mu})$$

Przy założeniu, że $\mathbf{C}[P] \neq \mathbf{0}$:

$$\begin{aligned} 2F^2 + 1 &= 2kF_{\text{mid}}^2 + \frac{1}{\mu} \\ F_{\text{mid}}^2 &= \frac{2F^2 + 1 - \frac{1}{\mu}}{2k} \end{aligned}$$

Obie strony są nieujemne, więc:

$$F_{\text{mid}} = \sqrt{\frac{2F^2 + 1 - \frac{1}{\mu}}{2k}} \quad (3.14)$$

Zakładając dostatecznie duże wartości μ , można przyjąć $\frac{1}{\mu} \approx 0$. Wówczas $F_{\text{mid}} \approx \sqrt{\frac{2F^2 + 1}{2k}}$. Przykładowo, współczynnik skalujący dla DE/mid/1 równoważny DE/rand/1 przy $F = 0.9$ na mocy zależności (3.14) wynosi $F_{\text{mid}} \approx 1.14$.

Przechodząc z k do nieskończoności, równanie mutanta DE/mid/ ∞ można zapisać w następujący sposób:

$$u_i^{\infty^{\text{mid}}} = m + F_{\infty^{\text{mid}}} \cdot v_{\infty^{\text{mid}}}$$

Gdzie współczynnik skalujący $F_{\infty_{\text{mid}}}$ jest równy $\sqrt{2F^2 + 1 - \frac{1}{\mu}}$, ponieważ:

$$\begin{aligned} C[u_i^{\infty^{\text{mid}}}] &= C[u_i] \\ C[m + F_{\infty_{\text{mid}}} \cdot v_{\infty_{\text{mid}}}] &= C[P](2F^2 + 1) \\ \frac{C[P]}{\mu} + F_{\infty_{\text{mid}}}^2 C[P] &= C[P](2F^2 + 1) \\ F_{\infty_{\text{mid}}} &= \sqrt{2F^2 + 1 - \frac{1}{\mu}} \end{aligned}$$

3.4. Współczynniki skalujące

Tabela 3.1 podsumowuje wyprowadzone współczynniki skalujące dla wszystkich wariantów algorytmów analizowanych w tej pracy. Użycie tych współczynników gwarantuje, że macierz kowariancji mutantów dla dowolnego wariantu ewolucji różnicowej będzie równa tej macierzy dla algorytmu DE/rand/1 ze współczynnikiem skalującym F .

Współczynniki skalujące dla DE/best są identyczne jak dla DE/mid, ponieważ w obu tych wariantach, w iteracji dla bieżącej populacji, zawsze mutowany jest ten sam, ustalony punkt. W DE/rand punkt mutowany jest za każdym razem losowany, przez co jest bardziej zmienny (ma większą normę macierzy kowariancji). Z tego powodu warianty DE/best oraz DE/mid potrzebują większych współczynników skalujących niż DE/rand.

Algorytm	Współczynnik skalający równoważny F z DE/rand/1
DE/rand/k	$\sqrt{\frac{2F^2}{2k}} = \frac{F}{\sqrt{k}}$
DE/rand/ ∞	$\sqrt{2F^2} = \sqrt{2}F$
DE/best/k	$\sqrt{\frac{2F^2+1-\frac{1}{\mu}}{2k}}$
DE/best/ ∞	$\sqrt{2F^2 + 1 - \frac{1}{\mu}}$
DE/mid/k	$\sqrt{\frac{2F^2+1-\frac{1}{\mu}}{2k}}$
DE/mid/ ∞	$\sqrt{2F^2 + 1 - \frac{1}{\mu}}$

Tabela 3.1. Współczynniki skalujące dla algorytmów analizowanych w tej pracy gwarantujące równość macierzy kowariancji mutantów dla DE/rand/1 ze współczynnikiem skalującym F

4. Metodyka testowania

Twierdzenie No Free Lunch dla optymalizacji głosi, że nie istnieje najlepszy uniwersalny algorytm dla wszystkich zadań [15]. Niezależnie od miary jakości algorytmu optymalizacyjnego, po uśrednieniu dla wszystkich zadań optymalizacyjnych, dowolne dwa różne algorytmy będą osiągały taką samą jakość wyników. Innymi słowy, nie czyniąc żadnych założeń na temat natury optymalizowanej funkcji celu f , nigdy nie będziemy w stanie wykazać wyższości dowolnego algorytmu ewolucyjnego nad np. błędzeniem losowym. Zatem należy przyjąć pewne założenia na temat optymalizowanych funkcji, żeby móc wskazać algorytm najlepszy (dla danego zbioru funkcji). W idealnym przypadku, funkcje testowe odpowiadałyby tym spotykanym w rzeczywistych zadaniach optymalizacji. Nie wiadomo jednak jakie rodziny funkcji występują najczęściej w realnych problemach. Dlatego w tej pracy przyjęto zbiór 7 funkcji wybranych z zestawu BBOB 2013 [4], szczegółowo opisanych w poniższym rozdziale.

4.1. Zestaw funkcji testowych BBOB 2013

BBOB (ang. Black-Box Optimization Benchmarking) to nazwa warsztatów poświęconych testowaniu algorytmów optymalizacji [6]. Warsztaty te odbywały się w latach 2009, 2010, 2012 oraz 2013. Zmieniały się algorytmy startujące w konkursie, nie zmieniał się sposób ich testowania. Podobnie funkcjonują warsztaty CEC (ang. Commission of the European Communities) [13].

W tej pracy wykorzystano funkcje z zestawu BBOB 2013. Wszystkie funkcje z tego zestawu są zdefiniowane w całej przestrzeni \mathbb{R}^D , ale zbiór dopuszczalny został zawężony do $[-5; 5]^D$. Wszystkie poszukiwane minima leżą w tym obszarze. Na rozwiązania spoza zbioru dopuszczalnego nakładana jest zewnętrzna funkcja kary zdefiniowana w [6] następująco:

$$f_{\text{pen}} : \mathbb{R}^D \rightarrow \mathbb{R}, \mathbf{x} \mapsto \sum_{i=1}^D \max(0, |x_i| - 5)^2$$

Zestaw BBOB 2013 składa się z 54 funkcji, które dzielą się na dwie zasadnicze klasy:

- 24 funkcje bez szumów o numerach od 1 do 24 [4].
- 30 funkcji z szumami o numerach od 101 do 130 [5].

Funkcje bez szumów zwracają dokładną wartość funkcji celu w punkcie x . Funkcje z szumami wartość tą dodatkowo modyfikują używając losowego szumu. W tej pracy skoncentrowano się na funkcjach bez szumów. Można je podzielić ze względu na ich właściwości:

- Funkcje separowalne (numery od 1 do 5).
- Funkcje dobrze uwarunkowane numerycznie (od 6 do 9).
- Funkcje jednomodalne (z jednym ekstremum lokalnym), źle uwarunkowane numerycznie (od 10 do 14).
- Funkcje wielomodalne o regularnej strukturze (od 15 do 19).
- Funkcje wielomodalne o nieregularnej strukturze (od 20 do 24).

Zgodnie z procedurą BBOB 2013 [6], celem algorytmu optymalizacji jest wygenerowanie rozwiązania o wartości funkcji celu mniejszej bądź równej od $f_{\text{opt}} + 10^{-8}$, gdzie f_{opt} to minimum optymalizowanej funkcji. W wielu przypadkach znalezienie takiego rozwiązania trwałoby zbyt długo. Dlatego zastosowano kryterium stopu, jakim jest ograniczenie na maksymalną liczbę wywołań funkcji oceny (FEs , ang. Function Evaluations), wynoszącą 10^5D . Rozmiar populacji μ dla każdego algorytmu wynosił $10D$. Początkowa populacja była inicjalizowana zgodnie z rozkładem jednostajnym w zbiorze dopuszczalnym tzn. w hipersześcianie $[-5; 5]^D$. Punkty, które w czasie optymalizacji wyjdą poza zbiór dopuszczalny, nie są naprawiane. Zewnętrzna funkcja kary zapobiega znacznemu oddalaniu się punktów od interesującego obszaru.

Ze względu na losowość procesu optymalizacji, na każdej funkcji algorytm jest uruchamiany 15 razy. Z każdego uruchomienia zapisywany jest najlepszy wynik będący wartością funkcji dla najlepszego osobnika pomniejszoną o f_{opt} .

Na funkcjach separowalnych, dobrze uwarunkowanych numerycznie lub jednomodalnych ewolucja różnicowa znajdowała rozwiązanie bliskie optymalnemu na tyle często, że kłopotliwe było porównywanie jej wariantów. Różnice pomiędzy wariantami algorytmów udało się uchwycić przy użyciu trudniejszych funkcji. Z tego powodu wybrano 7 funkcji wielomodalnych o numerach 15, 16, 19, 20, 21, 22, 24. Trzy z nich mają regularną strukturę (15, 16, 19), cztery nieregularną (20, 21, 22, 24). Wszystkie są niesymetryczne. Poniżej przedstawiono je dokładniej. Na trójwymiarowych wykresach wartości funkcji f maleją ku górze, żeby minima globalne funkcji były dobrze widoczne. Dlatego minimalizacja oznacza „wchodzenie pod góre”.

4.1.1. Funkcja numer 15 – Rastrigina

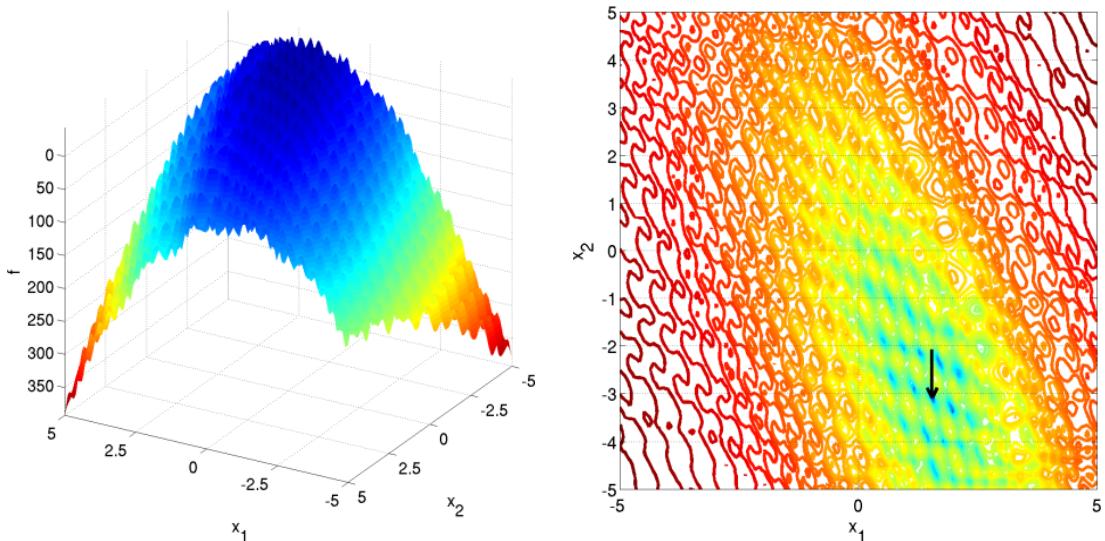
Ekstrema w oryginalnej funkcji Rastrigina są rozmiieszczone regularnie i symetryczne. Dzięki dwóm transformacjom T_{asy}^β oraz T_{osz} zaburzono symetrię oraz regularność. Właściwości funkcji numer 15:

- Około $10D$ minimów lokalnych.
- Globalnie duże zmiany wartości, lokalnie - małe.
- Separowalna liniowo.

$$f_{15}(\mathbf{x}) = 10 \left(D - \sum_{i=1}^D \cos(2\pi z_i) \right) + \|\mathbf{z}\|^2 + f_{\text{opt}}$$

$$\mathbf{z} = \mathbf{R} \boldsymbol{\Lambda}^{10} \mathbf{Q} T_{\text{asy}}^{0.2}(T_{\text{osz}}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}})))$$

\mathbf{Q} i \mathbf{R} to ortogonalne macierze obrotu. $\boldsymbol{\Lambda}^\alpha$ to D -wymiarowa macierz diagonalna, w której i -ty element na przekątnej jest równy $\alpha^{\frac{1}{2} \frac{i-1}{D-1}}$ dla $i = 1, \dots, D$.



Rysunek 4.1. Dwuwymiarowa funkcja Rastrigina, strzałka wskazuje minimum [4]

4.1.2. Funkcja numer 16 – Weierstrassa

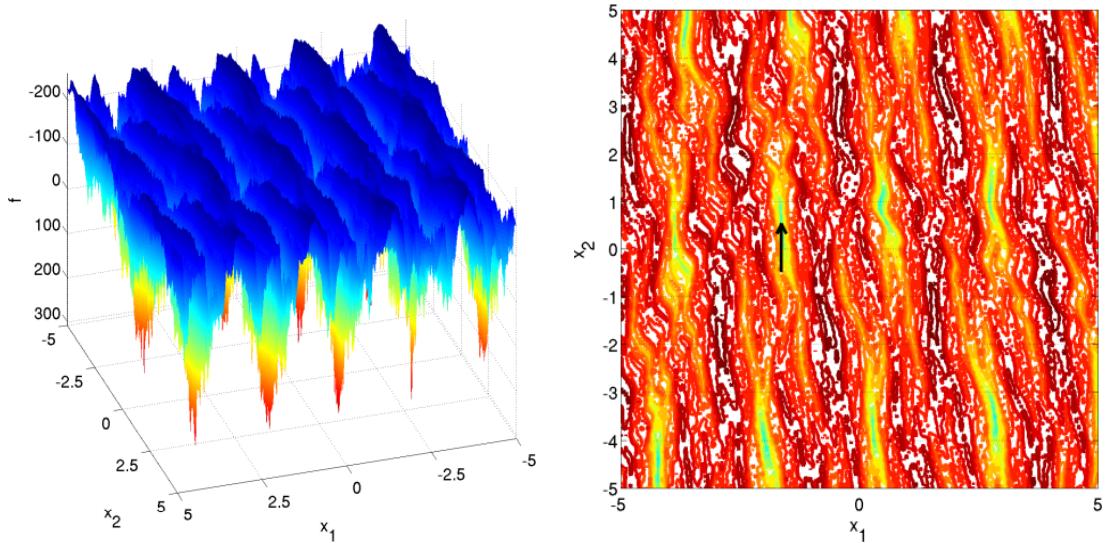
Funkcja 16 jest obrócona w stosunku do oryginalnej funkcji Weierstrassa – pierwszego opublikowanego przykładu rzeczywistej funkcji ciągłej, nieróżniczkowalnej w żadnym punkcie [14]. Posiada powtarzalny, ale bardzo „wyboisty” przebieg oraz więcej niż jedno optimum globalne. Właściwości:

- Globalnie regularna, lokalnie nieregularna.
- Brak unikalnego optimum globalnego.

$$f_{16}(\mathbf{x}) = 10 \left(\frac{1}{D} \sum_{i=1}^D \sum_{k=0}^{11} 1/2^k \cos(2\pi 3^k(z_i + 1/2)) - f_0 \right)^3 + \frac{10}{D} f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}}$$

$$\mathbf{z} = \mathbf{R} \boldsymbol{\Lambda}^{1/100} \mathbf{Q} T_{\text{osz}}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}}))$$

$$f_0 = \sum_{k=0}^{11} 1/2^k \cos(2\pi 3^k 1/2)$$



Rysunek 4.2. Dwuwymiarowa funkcja Weierstrassa, strzałka wskazuje minimum [4]

4.1.3. Funkcja numer 19 – Griewanka-Rosenbrocka

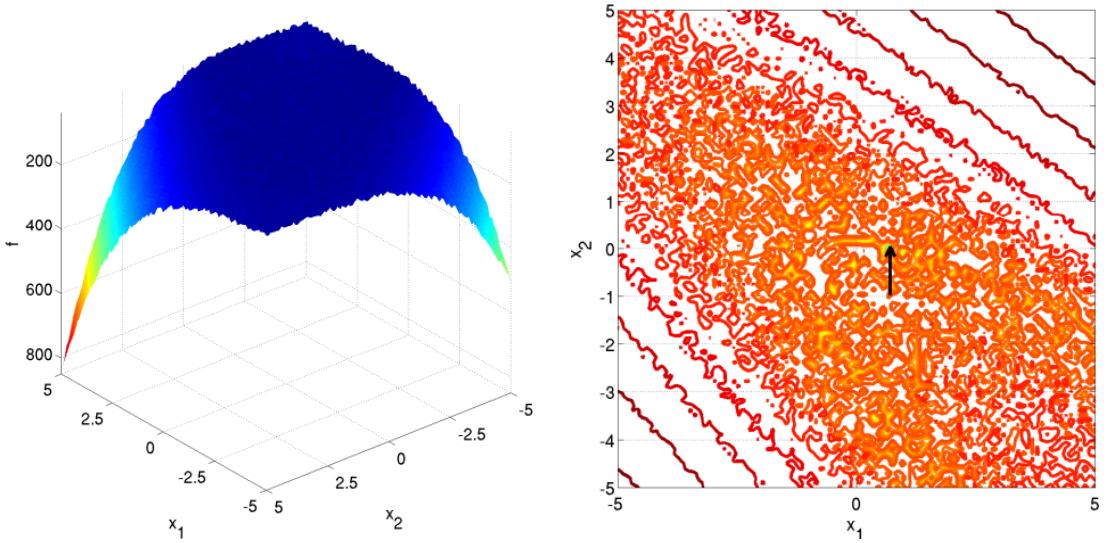
Funkcja 19 to złożenie funkcji Griewanka oraz Rosenbrocka, o ogromnej liczbie ekstremów lokalnych.

$$f_{19}(\mathbf{x}) = \frac{10}{D-1} \sum_{i=1}^{D-1} \left(\frac{s_i}{4000} - \cos(s_i) \right) + 10 + f_{\text{opt}}$$

$$\mathbf{z} = \max \left(1, \frac{\sqrt{d}}{8} \right) \mathbf{R} \mathbf{x} + 0.5$$

$$\bigwedge_{i \in 1, \dots, D} s_i = 100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2$$

$$\mathbf{z}^{\text{opt}} = \mathbf{1}$$



Rysunek 4.3. Dwuwymiarowa funkcja numer 19, strzałka wskazuje minimum [4]

4.1.4. Funkcja numer 20 – Schwefela

$\mathbf{1}_-^+$ oznacza wektor długości D zawierający 1 oraz -1 losowane z równym prawdopodobieństwem. W funkcji Schwefela najlepsze 2^D minimów lokalnych jest położonych stosunkowo blisko narożników atrakcyjnej hiperpłaszczyzny. Właściwości:

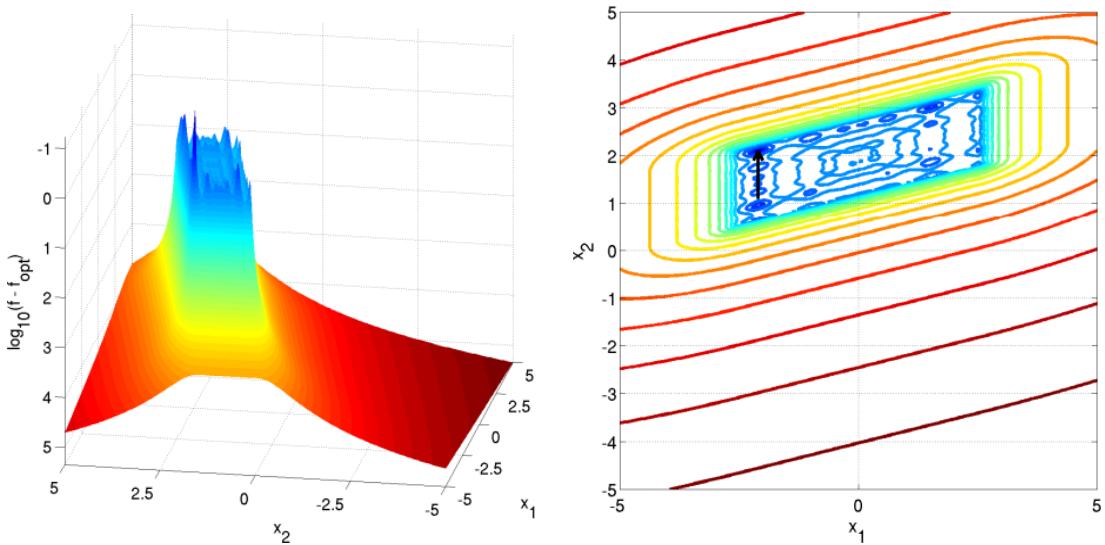
- Częściowo separowalna.
- Charakterystyczna, obrócona struktura.
- Atrakcyjne obszary przeszukiwań znajdują się w narożnikach hiperpłaszczyzny.

$$f_{20}(\mathbf{x}) = -\frac{1}{D} \sum_{i=1}^D z_i \sin \left(\sqrt{|z_i|} \right) + 4.189828872724339 + 100 f_{\text{pen}} \left(\frac{\mathbf{z}}{100} \right) + f_{\text{opt}}$$

$$\hat{\mathbf{x}} = 2 \cdot \mathbf{1}_-^+ \otimes \mathbf{x}$$

$$\bigwedge_{i \in 1, \dots, D} \hat{z}_1 = \hat{x}_1, \hat{z}_{i+1} = \hat{x}_{i+1} + 0.25(\hat{x}_i - x_i^{\text{opt}})$$

$$\mathbf{x}^{\text{opt}} = \frac{4.2096874633}{2} \mathbf{1}_-^+$$



Rysunek 4.4. Dwuwymiarowa funkcja Schwefela, strzałka wskazuje minimum [4]

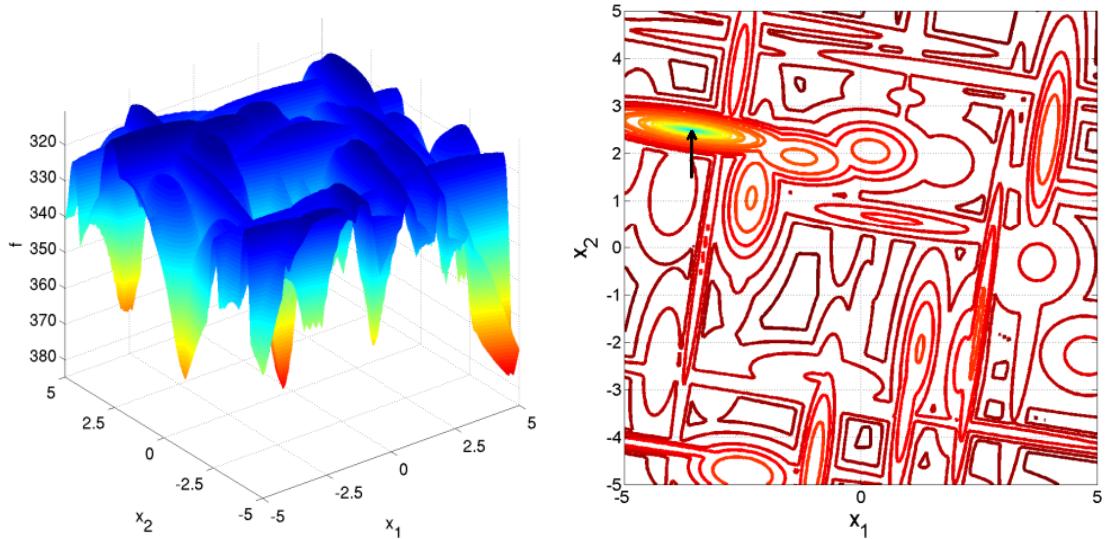
4.1.5. Funkcja numer 21 – górnki gaussowskie Gallaghera 101-me

Funkcja posiada 101 ekstremów lokalnych, których położenie oraz wielkość są losowe i niezależne od siebie. Analizowanie wyników na tej funkcji pomaga odpowiedź na pytanie, czy przeszukiwanie jest efektywne, gdy funkcja celu nie ma żadnej globalnej struktury.

$$f_{21}(\mathbf{x}) = T_{\text{osz}} \left(10 - \max_{i=1}^{101} w_i \exp \left(-\frac{1}{2D} (\mathbf{x} - \mathbf{y}_i)^T \mathbf{R}^T \mathbf{C}_i \mathbf{R} (\mathbf{x} - \mathbf{y}_i) \right) \right)^2 + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}}$$

$$w_i = \begin{cases} 1.1 + 8 \cdot \frac{i-2}{99} & \text{dla } i = 2, \dots, 101 \\ 10 & \text{dla } i = 1 \end{cases}$$

\mathbf{C}_i jest zdefiniowane w [4].



Rysunek 4.5. Dwuwymiarowa funkcja numer 21, strzałka wskazuje minimum [4]

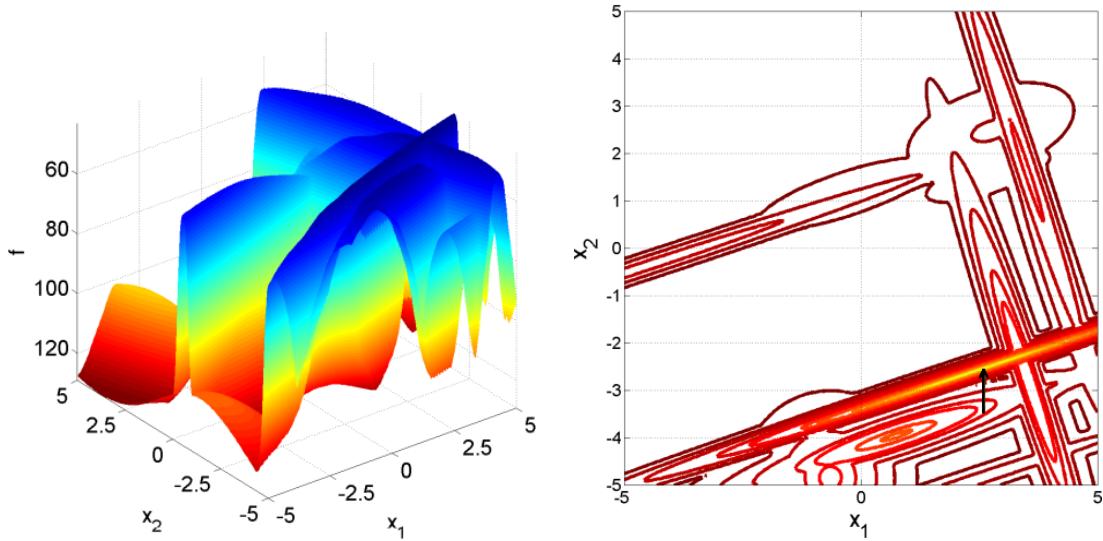
4.1.6. Funkcja numer 22 – górniki gaussowskie Gallaghera 21-hi

Funkcja posiada 21 ekstremów lokalnych, których położenie oraz wielkość są losowe i niezależne od siebie. Analizowanie wyników na tej funkcji pomaga odpowiedź na pytanie, jak wysoki wskaźnik uwarunkowania numerycznego wpływa na efektywność przeszukiwania, w porównaniu do funkcji 21.

$$f_{22}(\mathbf{x}) = T_{\text{osz}} \left(10 - \max_{i=1}^{21} w_i \exp \left(-\frac{1}{2D} (\mathbf{x} - \mathbf{y}_i)^T \mathbf{R}^T \mathbf{C}_i \mathbf{R} (\mathbf{x} - \mathbf{y}_i) \right) \right)^2 + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}}$$

$$w_i = \begin{cases} 1.1 + 8 \cdot \frac{i-2}{19} & \text{dla } i = 2, \dots, 21 \\ 10 & \text{dla } i = 1 \end{cases}$$

\mathbf{C}_i zostało zdefiniowane w [4].



Rysunek 4.6. Dwuwymiarowa funkcja numer 22, strzałka wskazuje minimum [4]

4.1.7. Funkcja numer 24 – Lunačka

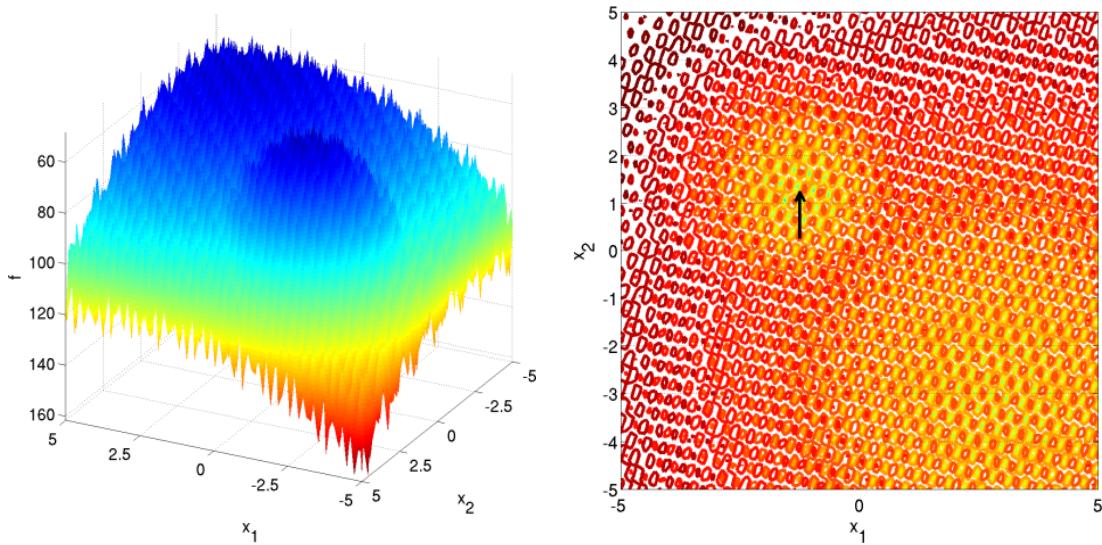
Funkcja Lunačka jest również nazywana podwójnym Rastriginem (bi-Rastrigin). Funkcja ta ma dużą liczbę ekstremów lokalnych. Na jej dwuwymiarowym wykresie można zauważać dwa charakterystyczne zbiory przyciągania minimum lokalnego – „górnki”. Żeby znaleźć optimum, algorytm optymalizacji musi najpierw poprawnie wybrać „górkę”, a następnie dokładnie przeszukać wielomodalny obszar wewnątrz jej. Funkcja została skonstruowana w taki sposób, aby zmylić niektóre algorytmy ewolucyjne z dużym rozmiarem populacji. Obszar przyciągania zawierający zwodnicze minimum lokalne stanowi około 70% całej przestrzeni przeszukiwań. Analiza wyników na tej funkcji pomaga odpowiedzieć na pytanie, czy przeszukiwanie może mieć charakter lokalny w skali globalnej oraz charakter globalny w skali lokalnej.

$$f_{24}(\mathbf{x}) = \min \left(\sum_{i=1}^D (\hat{x}_i - \mu_0)^2, dD + s \sum_{i=1}^D (\hat{x}_i - \mu_1)^2 \right) + 10 \left(D - \sum_{i=1}^D \cos(2\pi z_i) \right) + 10^4 f_{\text{pen}}(\mathbf{x})$$

$$\hat{\mathbf{x}} = 2 \operatorname{sign}(\mathbf{x}^{\text{opt}}) \otimes \mathbf{x}, \mathbf{x}^{\text{opt}} = \mu_0 \mathbf{1}_-^+$$

$$\mathbf{z} = \mathbf{Q} \Lambda^{100} \mathbf{R} (\hat{\mathbf{x}} - \mu_0 \mathbf{1})$$

$$\mu_0 = 2.5, \mu_1 = -\sqrt{\frac{\mu_0^2 - d}{s}}, s = 1 - \frac{1}{2\sqrt{D+20} - 8.2}, d = 1$$



Rysunek 4.7. Dwuwymiarowa funkcja Lunačka, strzałka wskazuje minimum [4]

4.2. Porównywanie wyników

Duża liczba wariantów algorytmów wymaga systematycznego i wiarygodnego sposobu porównywania i prezentacji wyników. Testy istotności statystycznej pozwalają w szybki sposób rozstrzygnąć, czy wyniki osiągane przez dwa algorytmy są od siebie różne w statystycznie istotny sposób. Statystycznie istotny, to znaczy, czy obserwowane różnice wyników nie są jedynice wynikiem losowości związanej z niedeterministycznym charakterem porównywanych algorytmów. Dystrybuenty empiryczne pozwalają zorientować się na ile wyniki algorytmów różnią się od siebie. Procent osobników, które znalazły się poza przestrzenią przeszukiwań, pozwala ocenić, czy algorytm dotarł do krawędzi zbioru dopuszczalnego, czy pozostawał cały czas w jego wnętrzu.

4.2.1. Testy istotności statystycznej

W tej pracy jako podstawowy sposób porównywania wyników został wykorzystany test Wilcoxona dla par obserwacji. Zaproponowany pierwotnie jako test przesunięcia dla dwóch równolicznych próbek przez Franka Wilcoxona w 1945, uogólniony następnie przez Manna i Whitneya w 1947 dla przypadku różnolicznych próbek [10]. Test Wilcoxona to nieparametryczna alternatywa dla testu t-Studenta w przypadku dwóch równolicznych próbek. Test t-Studenta sprawdza hipotezę zerową o równejś średnich arytmetycznych w odpowiadających im populacjach, natomiast test Wilcoxona weryfikuje równość median. Średnia jest wrażliwa na wartości odstające, natomiast mediana nie. Ponadto, test t-Studenta jest parametryczny, to znaczy zakłada pewien rozkład

wartości z badanej próby. Test Wilcooxona jest nieparametryczny, to znaczy, nie zakłada postaci rozkładu badanych wartości. Dlatego został wybrany.

W teście Wilcooxona porównywane jest n par obserwacji, pochodzących z dwóch zbiorów. Pierwszy zbiór odpowiada algorytmowi A i zawiera liczby x_1, x_2, \dots, x_n . Drugi zbiór odpowiada algorytmowi B i zawiera liczby y_1, y_2, \dots, y_n . Liczby w zbiorach oznaczają różnice wartości funkcji celu najlepszego punktu oraz minimum globalnego optymalizowanej funkcji. W tej pracy $n = 15$, ponieważ liczba niezależnych uruchomień algorytmu wynosiła 15. Zostały spełnione wszystkie założenia dla testu Wilcooxona:

1. Wartości x_i i y_i są parowane w sposób losowo niezależny od siebie.
2. Wartości x_i i y_i pochodzą z populacji o rozkładzie ciągłym.
3. Wartości x_i i y_i można porównywać ze sobą, jednoznacznie stwierdzając, która jest większa, mniejsza, bądź równa.

Testowana hipoteza zerowa H_0 brzmi: „różnica pomiędzy medianami rozkładów generujących zbiory A i B wynosi zero”. Hipoteza alternatywna H_1 brzmi: „różnica pomiędzy medianami rozkładów generujących zbiory A i B jest różna od zera”. Test Wilcooxona przebiega następująco:

1. Obliczenie różnic $d_i = y_i - x_i$.
2. Usunięcie par, dla których $d_i = 0$. $N \leq n$ oznacza liczbę pozostałych par.
3. Posortowanie rosnąco wartości bezwględnych różnic $|d_i|$.
4. Rangowanie posortowanego zbioru rangami R_i , poczynając od rangi równej 1.
5. Obliczenie statystyki $W = \left| \sum_{i=1}^N \text{sgn}(d_i)R_i \right|$.
6. Rozkład W zbiega do rozkładu normalnego wraz ze wzrostem N . Obliczane jest $p = \frac{W-0.5}{\sigma_W}$, gdzie $\sigma_W = \sqrt{\frac{N(N+1)(2N+1)}{6}}$.
7. Jeśli $z > z_{\text{critical}}$, hipoteza H_0 jest odrzucana, w przeciwnym przypadku – przyjmowana. Wartości z_{critical} zależą od przyjętego poziomu ufności. W tej pracy poziom ufności wynosił 0.05, co odpowiada $z_{\text{critical}} = 1.96$ [9].

Jeśli test nie pozwalał na odrzucenie hipotezy zerowej, uznawano, że wyniki porównywanych algorytmów nie są istotnie różne od siebie. Jeśli test odrzucał hipotezę zerową, uznawano, że algorytmy A i B dają różne wyniki. Wówczas porównywane były mediany obu zbiorów. Algorytm, którego mediana wyników była niższa, uznawany był za lepszy.

4.2.2. Dystrybuanty empiryczne

W celu porównania wyników osiąganych przez algorytmy, wykreślono dystrybuanty empiryczne najlepszych wyników z każdego uruchomienia dla wszystkich algorytmów na jednej funkcji. Oś x wykresu odpowiada wartościom najlepszych wyników osiąganych przez algorytmy, tzn. różnicom wartości funkcji celu najlepszego punktu wyznaczonego w pojedynczym uruchomieniu oraz minimum globalnego optymalizowanej funkcji. Oś y odpowiada natomiast estymowanemu prawdopodobieństwu, z jakim algorytm osiąga dany wynik. Dystrybuanty empiryczne pozwalają również określić, jak bliskie minimum było najlepsze znalezione przez algorytm rozwiązanie.

4.2.3. Procent osobników poza zbiorem dopuszczalnym

W każdym uruchomieniu zliczana była liczba osobników, których jakakolwiek współrzędna wykraczała poza zbiór dopuszczalny zdefiniowany jako $[-5; 5]^D$. Następnie obliczany był stosunek liczby osobników spoza zbioru dopuszczalnego do liczby wszystkich wygenerowanych osobników. Jeśli algorytm nie znajdował minimum, wówczas liczba wszystkich generowanych osobników była równa maksymalnej liczbie obliczeń wartości funkcji celu $FEs = 10^5 D$. Dla jednego algorytmu i jednej funkcji, procent osobników poza zbiorem dopuszczalnym uśredniano z 15 niezależnych uruchomień.

5. Wyniki eksperymentów

W tym rozdziale przedstawiono najważniejsze wyniki eksperymentów opatrzone komentarzem. Wszystkie wykresy oraz dane tabelaryczne znajdują się w dodatku B.

5.1. Parametry eksperymentów

Dla funkcji o małej liczbie wymiarów trudno dostrzec różnicę pomiędzy algorytmami. Im wyższy wymiar przestrzeni przeszukiwań, tym trudniejszy problem i dłuższy czas obliczeń, ale różnice pomiędzy efektywnością badanych algorytmów stają się bardziej widoczne. Dlatego liczba wymiarów D w testach wynosiła 10, 20, 40 oraz 80.

Współczynnik skalujący algorytmu DE/rand/1 wynosił $F = 0.9$. Dla pozostałych algorytmów współczynnik skalujący był obliczany zgodnie z tabelą 3.1. W każdym z algorytmów zostało wykorzystane krzyżowanie dwumianowe (opisane w rozdziale 2.2.3) ze współczynnikiem krzyżowania $CR = 0.9$ [11].

5.2. Implementacja

Zestaw BBOB 2013 dostarcza implementację wszystkich funkcji testowych w języku C. Funkcje te obliczane są miliardy razy podczas eksperymentów, dlatego ważne jest, żeby wykonywały się jak najszybciej. Procedura testująca oraz badane algorytmy zostały zaimplementowane w Javie. Procedura testująca wywołuje funkcje napisane w języku C dzięki Java Native Interface (JNI). Do automatyzacji testów oraz przetwarzania wyników zostały napisane skrypty w sh (języku powłoki Bourne'a) oraz języku R. Do przeprowadzenia testów istotności statystycznej Wilcooxona użyto funkcji `wilcox.test` z pakietu `stats` języka R. Dystrybuanty empiryczne zostały obliczone dzięki funkcji `ecdf` (ang. empirical cumulative distribution function), pochodzącej z tego samego pakietu.

Poprawność implementacji algorytmów w Javie została zweryfikowana dzięki testom macierzy kowariancji populacji. Zbadano wartości macierzy po pierwszej mutacji, uśrednione z 10000 niezależnych uruchomień. Średnie wartości macierzy kowariancji były bardzo zbliżone dla wszystkich badanych algorytmów, zgodnie z założeniami

teoretycznymi poczynionymi w rozdziale 3. Uśrednioną macierz kowariancji populacji przedstawia tabela 5.1.

21.36	0.02	0.01	-0.04	0.06	-0.01	-0.03	-0.02	-0.01	0.03
0.02	21.37	0.00	0.01	-0.01	0.02	0.03	0.02	0.03	0.00
0.01	0.00	21.40	-0.01	-0.03	-0.03	-0.09	0.04	0.01	-0.02
-0.04	0.01	-0.01	21.37	0.02	0.00	0.02	0.07	0.03	-0.03
0.06	-0.01	-0.03	0.02	21.43	-0.00	0.02	0.01	-0.01	0.01
-0.01	0.02	-0.03	0.00	-0.00	21.41	-0.00	-0.01	0.02	-0.01
-0.03	0.03	-0.09	0.02	0.02	-0.00	21.38	0.04	0.05	-0.01
-0.02	0.02	0.04	0.07	0.01	-0.01	0.04	21.40	-0.05	0.04
-0.01	0.03	0.01	0.03	-0.01	0.02	0.05	-0.05	21.39	0.02
0.03	0.00	-0.02	-0.03	0.01	-0.01	-0.01	0.04	0.02	21.37

Tabela 5.1. Uśredniona macierz kowariancji dla 10-wymiarowej funkcji numer 15 i algorytmu DE/rand/1.

Wszystkie funkcje użyte do testów są nieseparowalne liniowo. Dlatego wartości macierzy kowariancji populacji poza główną przekątną są bliskie 0. Oznacza to, iż korelacja liniowa Pearsona pomiędzy wartościami różnych cech praktycznie nie występuje, wymiary są niezależne od siebie. Z kolei wartość bliska 21.4 powtarzająca się na głównej przekątnej oznacza taką samą wariancję wartości w każdym z wymiarów, tzn. dla każdej cechy osobnika. Przedstawiona macierz kowariancji jest praktycznie taka sama dla wszystkich algorytmów, dzięki odpowiednim współczynnikom skalującym F wyprowadzonym w rozdziale 3 oraz identycznym populacjom początkowym. Dzięki niemal identycznej macierzy kowariancji algorytmy mają taki sam zasięg mutacji i wyniki przez nie zwracane zależą tylko od badanego sposobu selekcji. Wartość bliską 21.4 można uzasadnić dzięki wyprowadzonej zależności (3.4), wiążącej macierz kowariancji mutanta $C[u_i]$ z macierzą kowariancji początkowej populacji $C[P]$. Populacja początkowa była losowana zgodnie z rozkładem jednostajnym na przedziale $[-5; 5]^{10}$, zatem na głównej przekątnej $C[P]$ znajduje się liczba $\frac{10^2}{12} \approx 8.3$, zgodnie ze wzorem na wariancję rozkładu jednostajnego $\frac{(b-a)^2}{12}$. F było równe 0.9, zatem $C[u_i] = C[P](2 \cdot 0.9^2 + 1) = C[P] \cdot 2.62 \approx \Lambda_{10} \cdot 8.3 \cdot 2.62 \approx \Lambda_{10} \cdot 21.7$, gdzie Λ_{10} to 10-wymiarowa macierz diagonalna.

5.3. Wnioski

W notacji stosowanej w tabelach brak statystycznej różnicy oznaczono znakiem „.”. Jeśli test istotności (opisany w rozdziale 4.2.1) odrzucał hipotezę zerową, uznawano, że algorytmy A i B dają różne wyniki. Wówczas porównywane były mediany obu zbiorów.

Algorytm, którego mediana wyników była niższa, uznawany był za lepszy i oznaczany znakiem „+”. Algorytm istotnie gorszy był oznaczany znakiem „-”.

W tabeli 5.2 przedstawiono porównanie wariantu DE/mid/1 do pozostałych, w 80 wymiarach. DE/mid/1 dawał lepsze rezultaty niemal dla wszystkich przypadków, jedynie na funkcji 16 zremisował z DE/rand/1 oraz uległ DE/best/1.

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/1	+	.	+	+	+	+	+
DE/rand/2	+	+	+	+	+	+	+
DE/rand/6	+	+	+	+	+	+	+
DE/rand/ ∞	+	+	+	+	+	+	+
DE/best/1	+	-	+	+	+	+	+
DE/best/2	+	+	+	+	+	+	+
DE/best/6	+	+	+	+	+	+	+
DE/best/ ∞	+	+	+	+	+	+	+
DE/mid/2	+	+	+	+	+	+	+
DE/mid/6	+	+	+	+	+	+	+
DE/mid/ ∞	+	+	+	+	+	+	+

Tabela 5.2. Porównanie DE/mid/1 do reszty wariantów DE w 80 wymiarach

Istnieje jednak nieskończenie wiele funkcji dla których DE/mid/1 nie jest najlepszym wyborem, np. rodzina funkcji podobnych do funkcji numer 16 – Weierstrassa. Na takich funkcjach lepiej sprawuje się DE/best/1.

Przetestowano również wariant algorytmu, w którym dla każdej populacji obliczano punkt środkowy, a następnie wywoływano dla niego funkcję oceny, w nadziei znalezienia najlepszego rozwiązania. Punkt ten nie był brany jako punkt do populacji. Wyniki otrzymywane po takiej modyfikacji nie różniły się w sposób statystycznie istotny w porównaniu z oryginalnym podejściem.

Biorąc pod uwagę najwyższy zbadany wymiar i wszystkie 7 funkcji testowych, końcowy ranking efektywności algorytmów, opracowany na podstawie porównań testów Wilcooxona dla wyników symulacji na 7 wybranych funkcjach, przedstawia się następująco:

1. DE/mid/1
2. DE/rand/1
3. DE/best/1
4. DE/mid/2
5. DE/rand/2

6. DE/best/2
7. DE/mid/6, DE/mid/ ∞ , DE/rand/6, DE/rand/ ∞ , DE/best/6, DE/best/ ∞

5.3.1. Wpływ liczby wymiarów na jakość rozwiązań

Wraz ze wzrostem liczby wymiarów, algorytm DE/mid/1 sprawuje się coraz lepiej w porównaniu do pozostałych wariantów. W 80 wymiarach jednoznacznie wygrywa ze wszystkimi pozostałymi algorytmami na 6 z 7 funkcji testowych. W 40 wymiarach prowadzenie DE/mid/1 wygląda podobnie. Jedynie różnica w stosunku do DE/rand/1 jest mniejsza, z którym to DE/mid/1 wygrywa na 3 funkcjach, remisuje również na 3, a na 1 przegrywa. W 20 wymiarach zajmuje drugie miejsce, ustępując DE/rand/1. W 10 wymiarach było najwięcej remisów, DE/mid/1 był zawsze najlepszy na funkcjach numer 15, 19 oraz 24.

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/1	+	.	+	-	.	.	+
DE/rand/2	+	+	+	+	+	+	+
DE/rand/6	+	+	+	+	+	+	+
DE/rand/ ∞	+	+	+	+	+	+	+
DE/best/1	+	-	+	+	+	+	+
DE/best/2	+	+	+	+	+	+	+
DE/best/6	+	+	+	+	+	+	+
DE/best/ ∞	+	+	+	+	+	+	+
DE/mid/2	+	+	+	+	+	+	+
DE/mid/6	+	+	+	+	+	+	+
DE/mid/ ∞	+	+	+	+	+	+	+

Tabela 5.3. Porównanie DE/mid/1 do reszty wariantów DE w 40 wymiarach

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/1	.	-	.	-	.	.	-
DE/rand/2	+	+	+	.	.	.	+
DE/rand/6	+	+	+	+	+	.	+
DE/rand/ ∞	+	+	+	.	+	.	+
DE/best/1	+	-	+	-	+	+	+
DE/best/2	+	+	+	+	+	+	+
DE/best/6	+	+	+	+	+	+	+
DE/best/ ∞	+	+	+	+	+	+	+
DE/mid/2	+	+	+	.	+	.	+
DE/mid/6	+	+	+	.	+	.	+
DE/mid/ ∞	+	+	+	.	+	.	+

Tabela 5.4. Porównanie DE/mid/1 do reszty wariantów DE w 20 wymiarach

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/1	+	-	+	-	.	.	+
DE/rand/2	+	+	+	-	.	.	+
DE/rand/6	+	+	+	.	-	.	+
DE/rand/ ∞	+	.	+	.	.	.	+
DE/best/1	+	.	+	-	+	+	+
DE/best/2	+	.	+	-	.	.	+
DE/best/6	+	+	+	+	.	.	+
DE/best/ ∞	+	+	+	+	+	.	+
DE/mid/2	+	+	+	.	.	.	+
DE/mid/6	+	+	+	.	.	.	+
DE/mid/ ∞	+	+	+	.	.	.	+

Tabela 5.5. Porównanie DE/mid/1 do reszty wariantów DE w 10 wymiarach

5.3.2. Wpływ optymalizowanych funkcji na wyniki porównania

Na funkcji 21, w 10 oraz 20 wymiarach, najlepsze wyniki dawał DE/rand/1. W 40 wymiarach wygrał ze wszystkimi algorytmami, oprócz DE/mid/1, z którym remisował. W 80 wymiarach jednak DE/mid/1 okazał się bezkonkurencyjny. DE/mid/1 miał zawsze niższy % osobników poza zbiorem dopuszczalnym niż DE/rand/1. Mimo, że funkcja 21 nie ma żadnej globalnej struktury, dla DE/mid/1 nie stanowiło to przeszkody.

Na funkcji 22 w niższych wymiarach sytuacja była remisowa. W 40 i 80 wymiarach relacja pomiędzy DE/mid/1 oraz DE/rand/1 była taka sama jak w przypadku funkcji 21 – w 40 wymiarach remis, natomiast w 80 zwycięstwo DE/mid/1. % osobników DE/mid/1 poza zbiorem dopuszczalnym cały czas utrzymywał się na niskim poziomie, niemal zawsze najniższym w porównaniu do innych algorytmów. W porównaniu do funkcji 21, wysoki wskaźnik uwarunkowania powodował niewielki spadek efektywności przeszukiwania wśród badanych wariantów. Ich kolejność w rankingu jakości zwracanych rozwiązań nie zmieniła się.

Na funkcji 24 DE/mid/1 przegrał tylko w wymiarze 20 z DE/rand/1, wszystkie pozostałe porównania wygrał. To pokazuje, że przeszukiwanie w DE/mid/1 może mieć charakter lokalny w skali globalnej oraz charakter globalny w skali lokalnej. Wszystkie algorytmy na funkcji 24 miały niski % osobników poza zbiorem dopuszczalnym, bliski zeru.

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/1	10	.	10	80	80	80	10
DE/rand/2	10	10	10	40	40	40	10
DE/rand/6	10	10	10	20	20	40	10
DE/rand/ ∞	10	20	10	40	20	40	10
DE/best/1	10	–	10	40	40	10	10
DE/best/2	10	20	10	20	20	20	10
DE/best/6	10	10	10	10	20	20	10
DE/best/ ∞	10	10	10	10	10	20	10
DE/mid/2	10	10	10	40	20	40	10
DE/mid/6	10	10	10	40	20	40	10
DE/mid/ ∞	10	10	10	40	20	40	10

Tabela 5.6. Liczba wymiarów od której DE/mid/1 jest lepszy od porównywanego wariantu

5.3.3. Wpływ liczby wymiarów na liczbę osobników poza zbiorem dopuszczalnym

Wraz ze wzrostem liczby wymiarów, początkowo odsetek osobników (punktów) poza zbiorem dopuszczalnym również wzrastał. W wyższych wymiarach jednak nie zawsze było to zasadą. Wraz z dalszym wzrostem liczby wymiarów, liczba osobników poza zbiorem dopuszczalnym w kilku przypadkach nie zmieniła się, a nawet trochę spadła. To zjawisko jest związane z problemem „wchodzenia w narożniki”, który zaczyna się tym bardziej nasilać, im wyższy wymiar. Problem ten polega na tym, że im wyższy wymiar, tym prawdopodobieństwo wylosowania punktu blisko każdego z ograniczeń, jest coraz mniejsze. Przykładowo, w dwóch wymiarach, niech definicją „wylosowania punktu blisko każdego z ograniczeń” będzie wylosowanie punktu w polu pozostałym po wycięciu okręgu wpisanego w kwadrat. W tym obszarze może znajdować się poszukiwane optimum globalne. Stosunek pola przy narożnikach do całego kwadratu wynosi $\frac{\pi r^2}{(2r)^2} \approx 0.79$. W trzech wymiarach stosunek ten spada do $\frac{4}{3}\frac{\pi r^3}{(2r)^3} \approx 0.52$. W dziesięciu – do około 0.08. Przy liczbie wymiarów dążącej do nieskończoności, stosunek ten spada do zera.

D	Numer funkcji testowej						
	15	16	19	20	21	22	24
10	1	30	7	0	1	1	0
20	12	38	2	2	4	5	0
40	31	42	3	6	17	16	0
80	40	39	10	20	16	17	1

Tabela 5.7. % osobników poza zbiorem dopuszczalnym dla DE/rand/1, w rozbiciu na funkcje i wymiary

D	Numer funkcji testowej						
	15	16	19	20	21	22	24
10	8	18	5	0	0	3	0
20	8	21	1	0	1	2	0
40	24	19	0	1	4	11	0
80	22	20	1	2	11	10	0

Tabela 5.8. % osobników poza zbiorem dopuszczalnym dla DE/best/1, w rozbiciu na funkcje i wymiary

D	Numer funkcji testowej						
	15	16	19	20	21	22	24
10	0	27	0	0	1	1	0
20	3	42	0	0	0	0	0
40	7	45	0	0	1	1	0
80	14	46	1	1	2	3	0

Tabela 5.9. % osobników poza zbiorem dopuszczalnym dla DE/mid/1, w rozbiciu na funkcje i wymiary

5.3.4. Wpływ liczby osobników poza zbiorem dopuszczalnym na jakość rozwiązań

Im algorytm miał mniejszą liczbę osobników poza zbiorem dopuszczalnym, tym zazwyczaj lepsze osiągał wyniki. DE/mid/1 na 6 funkcjach, na których zwyciężył, zawsze miał najmniej osobników poza zbiorem dopuszczalnym. W przypadku funkcji 16 – jedynej na której przegrał – nie miał najmniejszej liczby osobników poza zbiorem dopuszczalnym. Najmniej wówczas miał DE/best/1 i to właśnie ten wariant zwyciężył w wysokich wymiarach. Istnieje zatem korelacja pomiędzy jakością rozwiązań a zdolnością algorytmu do utrzymywania się w zbiorze dopuszczalnym. Potwierdza to obserwacja przedstawiona w pracy [3], w której porównano różne sposoby naprawiania rozwiązań i zauważono, że im mniej osobników jest naprawianych, tym algorytm otrzymuje lepsze wyniki.

W czasie badań prowadzonych w ramach tej pracy, liczba osobników poza zbiorem dopuszczalnym dochodziła nawet do 47% (DE/mid/2 na 20-wymiarowej funkcji 16). To pokazuje, jak ważny jest sposób radzenia sobie z takimi osobnikami. W eksperymentach nie naprawiano osobników, stosowano natomiast zewnętrzną funkcję kary opisaną w rozdziale 4.1, zgodnie z wytycznymi BBOB 2013. Wyniki na poziomie 47% pokazują, że funkcja kary nie jest skuteczną metodą. Prawdopodobnie efektywność wszystkich algorytmów można znacząco poprawić stosując sposoby naprawiania punktów opisane w [3], np. ponowne próbkowanie (resampling).

5.3.5. Wpływ liczby par wektorów różnicowych k na jakość wyników

Algorytmy dla $k = 6$ zachowywały się praktycznie identycznie jak dla $k = \infty$. Jest to wynik zgodny z rozważaniami teoretycznymi przedstawionymi w [2]. W przypadku $k = 6$, w mutacji, sumujemy aż 12 wektorów różnicowych, czyli 12 niezależnych zmiennych losowych. Na mocy centralnego twierdzenia granicznego, im większe k , tym rozkład sumy coraz bardziej przypomina wielowymiarowy rozkład normalny, a zatem przypadek $k = \infty$. Zostało to dokładnie opisane w rozdziale 3.1.4. Wydajne losowanie zmiennych o rozkładzie wielowymiarowym normalnym, o zadanej macierzy kowariancji, było zadaniem trudnym do implementacji w Javie. Dla każdego populacji, najpierw trzeba było wyznaczyć jej macierz kowariancji, a następnie macierz tę trzeba było rozłożyć. Macierz kowariancji często nie była dodatnio określona, dlatego dekompozycja Choleskiego nie mogła zostać użyta. Z tego powodu wykorzystano rozkład macierzy na wektory i wartości własne – metodę ogólniejszą, jednak bardziej kosztowną obliczeniowo. Operator mutacji w DE/mid/ ∞ wykonuje się o wiele wolniej od operatora mutacji w DE/mid/6, a ponieważ mutacja jest powtarzana miliony razy, ogólna wydajność DE/mid/ ∞ prezentuje się zdecydowanie najgorzej na tle pozostałych algorytmów. Z powodu braku różnic w jakości otrzymywanych rozwiązań, algorytmy z $k = 6$ są lepszym wyborem z praktycznego punktu widzenia.

Natomiast ogólnie najlepszą wartością dla k było 1. Im wyższe k , tym algorytm spisywał się gorzej. Relacja pomiędzy DE/mid, DE/rand oraz DE/best, dla większości funkcji pozostawała taka sama, niezależnie od k , tzn. DE/mid był zazwyczaj najlepszy, DE/rand drugi, zaś DE/best najgorszy dla ustalonego k .

5.3.6. Porównanie z algorytmami biorącymi udział w konkursie BBOB 2013

W tabeli 5.10 porównano DE/mid/1 z algorytmami startującymi w konkursie BBOB 2013 w 20 wymiarach. Wyniki dla 20 wymiarów i niższych dostarczyli wszyscy

członkowie konkursu. Dla 40 wymiarów – nieliczni, ponieważ było to opcjonalne. Dla 80 wymiarów – nikt, ponieważ nie było to wymaganiem. DE/mid/1 okazał się gorszy od 17 z 26 algorytmów z BBOB 2013 dla 20 wymiarów. W małej liczbie wymiarów DE/rand/1 był lepszy od DE/mid/1, dlatego DE/rand/1 również porównano do algorytmów z BBOB 2013. Wyniki przedstawiono w tabeli 5.11. DE/rand/1 był gorszy od 9 z 26 algorytmów.

Algorytm	Numer funkcji testowej							Suma
	15	16	19	20	21	22	24	
BIPOP-aCMA-STEP	-	-	-	-	-	-	-	-7
BIPOP-saACM-k	-	-	-	-	-	-	-	-7
CGA-grid100	+	-	+	-	+	+	+	3
CGA-grid16	+	-	+	-	+	+	+	3
CMAES_Hutter	+	-	+	-	+	.	+	2
DE_Pal	+	.	+	-	.	.	+	2
fmincon	+	-	-	-	-	-	+	-3
fminunc	+	+	-	-	-	-	+	-1
GA-100	.	-	.	-	+	.	-	-2
HCMA	-	-	-	-	-	-	-	-7
HILL	+	-	+	-	+	.	+	2
HMLSL	.	-	-	-	.	.	-	-4
IP	-	-	-	-	.	.	-	-5
IP-10DDr	-	-	-	-	-	.	-	-6
IP-500	-	-	-	-	.	.	-	-5
MEMPSODE	.	-	-	-	-	-	-	-6
MLSL	+	-	-	-	-	-	+	-3
OQNLP	+	-	-	-	.	.	-	-3
P-DCN	+	-	+	-	+	.	+	2
P-zero	+	-	+	-	+	.	+	2
PRCGA	.	-	-	-	+	.	-	-3
Simplex	+	-	-	-	.	.	+	-1
tany	-	-	-	-	-	.	-	-6
texp	-	-	-	-	-	-	-	-7
U-DCN	+	-	.	-	+	+	+	2
U-zero	+	-	.	-	+	+	+	2
Suma	6	-23	-9	-26	0	-4	0	-56

Tabela 5.10. Porównanie DE/mid/1 do 26 algorytmów z BBOB 2013 w 20 wymiarach

Algorytm	Numer funkcji testowej							Suma
	15	16	19	20	21	22	24	
BIPOP-aCMA-STEP	-	-	-	-	-	-	-	-7
BIPOP-saACM-k	-	-	-	-	-	-	-	-7
CGA-grid100	+	+	+	+	+	+	+	7
CGA-grid16	+	+	.	+	+	+	+	6
CMAES_Hutter	+	+	+	+	+	.	+	6
DE_Pal	+	+	+	.	.	.	+	4
fmincon	+	+	-	+	-	-	+	1
fminunc	+	+	-	+	-	-	+	1
GA-100	+	+	-	+	+	.	.	3
HCMA	-	-	-	-	-	-	-	-7
HILL	+	+	.	+	+	.	+	5
HMLSL	+	+	-	1
IP	-	-	-	-	.	.	-	-5
IP-10DDr	-	-	-	-	-	.	-	-6
IP-500	-	-	-	+	.	.	-	-3
MEMPSODE	-	.	-	.	-	-	-	-5
MLSL	+	+	-	+	-	-	+	1
OQNLP	+	+	-	+	.	.	.	2
P-DCN	+	+	+	+	+	+	+	7
P-zero	+	+	+	+	+	.	+	6
PRCGA	+	+	-	+	+	.	-	2
Simplex	+	+	-	+	.	.	+	3
tany	-	-	-	-	-	.	-	-6
texp	-	-	-	.	-	-	-	-6
U-DCN	+	+	-	+	+	+	+	5
U-zero	+	+	.	+	+	+	+	6
Suma	8	9	-13	10	0	-3	3	14

Tabela 5.11. Porównanie DE/rand/1 do 26 algorytmów z BBOB 2013 w 20 wymiarach

14 autorów startujących w konkursie BBOB 2013 zamieściło wyniki działania swoich algorytmów dla 40 wymiarów. Zarówno DE/mid/1 (tabela 5.12) jak i DE/rand/1 (tabela 5.13) w 40 wymiarach spisał się gorzej od 10 z 14 algorytmów z BBOB.

Algorytm	Numer funkcji testowej							Suma
	15	16	19	20	21	22	24	
IPOP-aCMA-STEP	-	-	-	-	-	-	-	-7
BIPOP-saACM-k	-	-	-	-	-	-	-	-7
CGA-grid100	+	-	+	-	+	.	+	2
CGA-grid16	+	-	+	-	.	.	+	1
CMAES	.	-	+	-	+	.	+	1
GA-100	.	-	-	-	.	.	-	-4
HILL	+	-	+	-	.	.	+	1
IP	-	-	-	-	.	-	-	-6
IP-10Dr	-	-	-	-	.	-	-	-6
IP-500	-	-	-	-	.	.	-	-5
MEMPSODE	.	-	-	-	-	-	-	-6
PRCGA	.	-	-	-	.	.	-	-4
tany	-	-	-	-	-	-	-	-7
texp	-	-	-	-	-	-	-	-7
Suma	-4	-14	-6	-14	-3	-7	-6	-54

Tabela 5.12. Porównanie DE/mid/1 do 14 algorytmów z BBOB 2013 w 40 wymiarach

Algorytm	Numer funkcji testowej							Suma
	15	16	19	20	21	22	24	
BIPOP-aCMA-STEP	-	-	-	-	-	-	-	-7
BIPOP-saACM-k	-	-	-	-	-	.	-	-6
CGA-grid100	+	-	+	-	+	.	+	2
CGA-grid16	+	-	+	-	.	+	+	2
CMAES	.	-	+	-	+	.	+	1
GA-100	-	-	-	-	.	.	-	-5
HILL	+	-	+	-	.	+	+	2
IP	-	-	-	-	.	.	-	-5
IP-10Dr	-	-	-	-	-	.	-	-6
IP-500	-	-	-	-	.	.	-	-5
MEMPSODE	-	-	-	-	-	-	-	-7
PRCGA	-	-	-	-	.	.	-	-5
tany	-	-	-	-	-	.	-	-6
texp	-	-	-	-	-	.	-	-6
Suma	-7	-14	-6	-14	-4	0	-6	-51

Tabela 5.13. Porównanie DE/rand/1 do 14 algorytmów z BBOB 2013 w 40 wymiarach

Na niekorzyść DE/mid/1 działała nie tylko niska liczba wymiarów, ale również brak dostrojenia parametrów takich jak μ , F , CR do optymalizowanych funkcji. Pozostałe algorytmy startujące w BBOB 2013 miały dostrojone parametry.

6. Podsumowanie

Cel pracy został osiągnięty – poprawiono efektywność algorytmu ewolucji różnicowej dla przyjętych funkcji proponując nowy wariant algorytmu nazwany DE/mid. Eksperymenty oraz przeprowadzone testy istotności statystycznej wykazały znaczną przewagę DE/mid/1 nad innymi wariantami ewolucji różnicowej na 6 z 7 wielowymiarowych funkcji testowych, opisanych w rozdziale 4.1. Przewaga DE/mid/1 jest szczególnie widoczna przy wysokiej liczbie wymiarów przestrzeni przeszukiwań. Ponadto, szczegółowo przeanalizowano, zaimplementowano i przetestowano również dwa inne, popularne warianty – DE/rand oraz DE/best, co pozwoliło wyciągnąć praktyczne wnioski, np. na temat doboru liczby wektorów różnicowych k . Okazało się, że dla każdego z trzech testowanych wariantów ewolucji różnicowej, zawsze najlepiej sprawdzało się ustawienie $k = 1$.

DE/mid sprawdzał się tym lepiej w porównaniu do reszty algorytmów, im bardziej złożony był problem, tzn. im wyższy wymiar optymalizowanej funkcji. Dla małej liczby wymiarów różnice jakości osiąganych rozwiązań zacierają się. Jednak istnieją funkcje, dla których DE/mid nie jest najlepszym wyborem. Taką funkcją okazała się np. funkcja numer 16 z zestawu testowego – funkcja Weierstrassa. Zatem, jeśli optymalizowana funkcja celu przypomina funkcję Weierstrassa, to warto wówczas użyć algorytmu DE/best/1.

Zauważono również korelację pomiędzy liczbą osobników poza zbiorem dopuszczalnym a jakością zwracanych rozwiązań. Algorytmy, które najlepiej utrzymywały się w zbiorze dopuszczalnym, dawały przeważnie najlepsze wyniki.

W tej pracy skoncentrowano się na modyfikacji operatora selekcji, natomiast operator krzyżowania dla każdego algorytmu był taki sam, używano krzyżowania dwumianowego. Prawdopodobnie dokładne przeanalizowanie wpływu krzyżowania i zaproponowanie własnych modyfikacji pozwoliłoby na dalsze ulepszenia algorytmów ewolucji różnicowej.

Ze względu na ograniczenia czasowe eksperymenty były przeprowadzane na ograniczonym zbiorze funkcji testowych, w ograniczonej liczbie wymiarów. Przetestowanie algorytmów ewolucji różnicowej na pewnych rodzinach funkcji, posiadających wspólne

cechy, być może pozwoliłoby zaobserwować wpływ cech optymalizowanej funkcji na zachowanie algorytmu. Otwartym i bardzo ważnym pytaniem pozostaje również, jak w ogóle powinny wyglądać funkcje testowe, tak aby były jak najbardziej zbliżone do praktycznych zadań optymalizacji globalnej.

Obecnie testowanie przy użyciu BBOB 2013 czy innych zestawów, np. CEC 2005 [13], jest bardzo czasochłonne. Autorzy algorytmów sami muszą wygenerować i opracować wszystkie wyniki. Nikt również nie weryfikuje procedury testowej stosowanej przez autorów ani poprawności dostarczanych przez nich wyników. Możliwe byłoby stworzenie serwisu, na który autorzy mogliby wysyłać kod źródłowy algorytmów optymalizacji. Wówczas wszystkie obliczenia byłby wykonywane w taki sam sposób dla każdego uczestnika konkursu. W taki sam sposób przetwarzane i prezentowane byłby wyniki w postaci tabel, wykresów itp. Możliwe byłoby prowadzenie bieżących rankingów – konkurs mógłby trwać cały czas. Dodatkowo, obliczenia mogłyby zostać znacznie przyspieszone dzięki chmurze obliczeniowej.

Obiecującym kierunkiem rozwoju jest również adoptowanie parametrów algorytmu w czasie jego wykonywania, czyli tzw. samoadaptacja [7]. W momencie, gdy różnorodność populacji algorytmu ewolucji różnicowej spadnie poniżej pewnego poziomu, algorytm przestaje badać nowe obszary. Wówczas opłacalne byłoby zwiększenie różnorodności populacji, np. poprzez zwiększenie zasięgu mutacji.

A. Załączniki

Do pracy została dołączona płyta CD. Zawiera one katalogi kod/ oraz praca/, w których znajduje się pełny kod źródłowy oraz ta praca w formie elektronicznej.

B. Pełne wyniki eksperymentów

Poniżej zamieszczono pełne wyniki testów istotności statystycznej (opisanych w rozdziale 4.2.1) oraz wszystkie dystrybuanty empiryczne (opisane w rozdziale 4.2.2) dla 10, 20, 40 oraz 80 wymiarów. Każda kolejna tabela zawiera porównanie algorytmu odniesienia (np. DE/rand/1 w tabeli B.1) do pozostałych wariantów ewolucji różnicowej. W notacji stosowanej w tabelach brak statystycznej istotności pomiędzy różnicami median wyników oznaczano znakiem „.”. Jeśli test istotności odrzucał hipotezę zerową, uznawano, że algorytmy A i B dają różne wyniki i różnica ta jest statystycznie istotna. Wówczas porównywane były mediany obu zbiorów. Jeśli mediana wyników algorytmu odniesienia była mniejsza niż algorytmu użytego w porównaniu, oznaczano to znakiem „+”, w przeciwnym przypadku znakiem „-”.

B.1. Wyniki uzyskane dla 10 wymiarów

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/2	+	+	.	+	.	.	.
DE/rand/6	+	+	.	+	.	.	.
DE/rand/ ∞	+	+	.	+	.	.	.
DE/best/1	+	+	+	+	+	+	+
DE/best/2	+	+	.	+	.	.	.
DE/best/6	+	+	+	+	.	.	+
DE/best/ ∞	+	+	+	+	+	.	+
DE/mid/1	-	+	-	+	.	.	-
DE/mid/2	+	+	.	+	+	.	.
DE/mid/6	+	+	.	+	.	.	.
DE/mid/ ∞	+	+	.	+	.	.	.

Tabela B.1. Porównanie DE/rand/1 do reszty algorytmów w 10 wymiarach

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/1	-	-	-	-	-	-	-
DE/rand/2	-	.	-	+	-	-	-
DE/rand/6	-	.	-	+	-	-	-
DE/rand/ ∞	-	.	-	+	-	-	-
DE/best/2	-	-	-	-	-	-	-
DE/best/6	.	+	-	+	-	-	-
DE/best/ ∞	.	+	-	+	.	-	-
DE/mid/1	-	.	-	+	-	-	-
DE/mid/2	-	.	-	+	.	-	-
DE/mid/6	-	.	-	+	-	-	-
DE/mid/ ∞	-	.	-	+	-	-	-

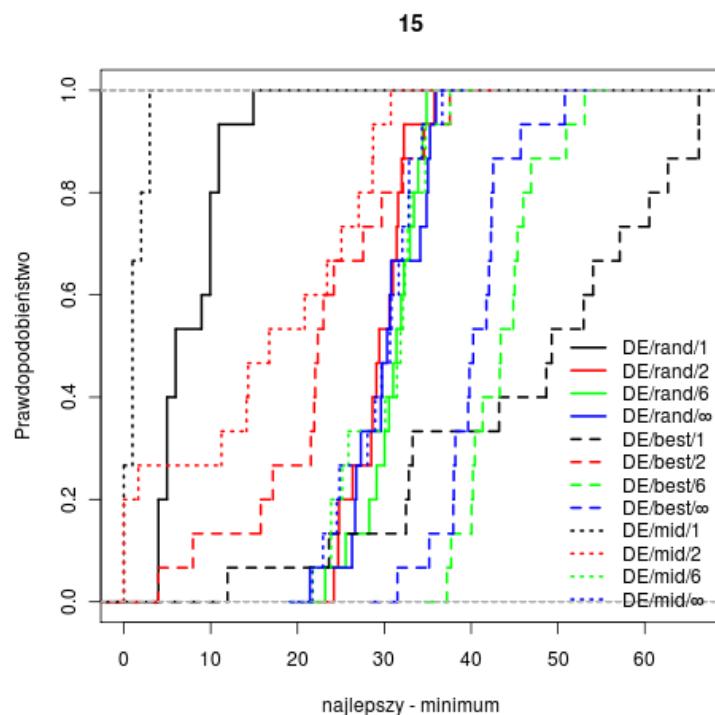
Tabela B.2. Porównanie DE/best/1 do reszty algorytmów w 10 wymiarach

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/1	+	-	+	-	.	.	+
DE/rand/2	+	+	+	-	.	.	+
DE/rand/6	+	+	+	.	-	.	+
DE/rand/ ∞	+	.	+	.	.	.	+
DE/best/1	+	.	+	-	+	+	+
DE/best/2	+	.	+	-	.	.	+
DE/best/6	+	+	+	+	.	.	+
DE/best/ ∞	+	+	+	+	+	.	+
DE/mid/2	+	+	+	.	.	.	+
DE/mid/6	+	+	+	.	.	.	+
DE/mid/ ∞	+	+	+	.	.	.	+

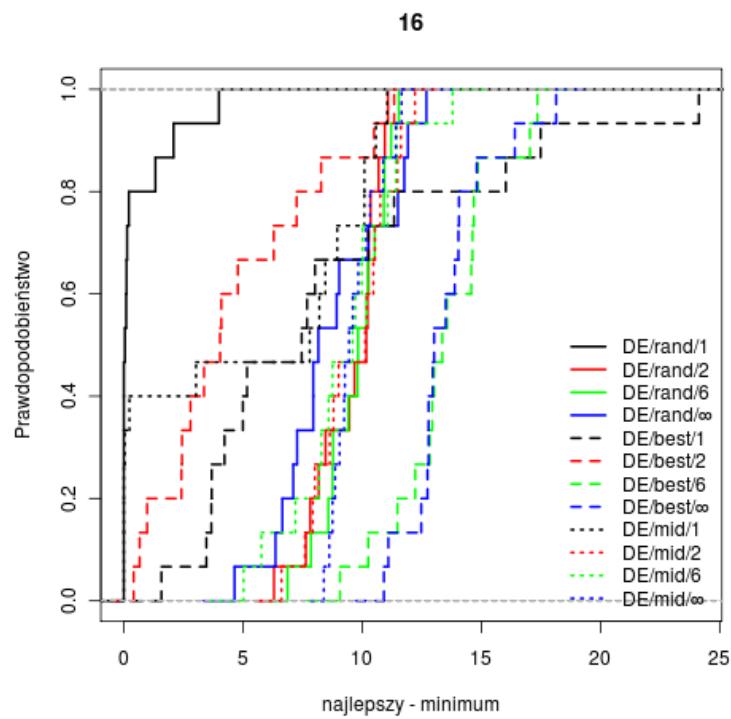
Tabela B.3. Porównanie DE/mid/1 do reszty algorytmów w 10 wymiarach

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/1	1	30	7	0	1	1	0
DE/rand/2	8	43	9	0	5	3	0
DE/rand/6	9	43	8	0	9	5	0
DE/rand/ ∞	9	43	10	0	10	7	0
DE/best/1	8	18	5	0	0	3	0
DE/best/2	6	19	6	0	2	2	0
DE/best/6	7	22	7	0	7	7	0
DE/best/ ∞	7	22	7	0	10	7	0
DE/mid/1	0	27	0	0	1	1	0
DE/mid/2	5	42	10	0	12	16	0
DE/mid/6	8	43	11	0	9	6	0
DE/mid/ ∞	9	43	11	0	11	6	0
średnia	6	33	8	0	6	5	0

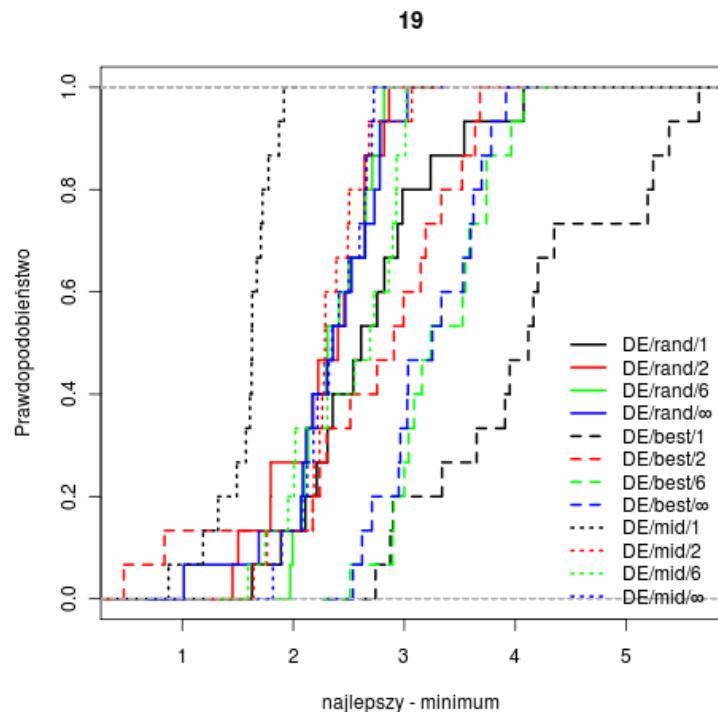
Tabela B.4. Średni procent osobników poza zbiorem dopuszczalnym w 10 wymiarach



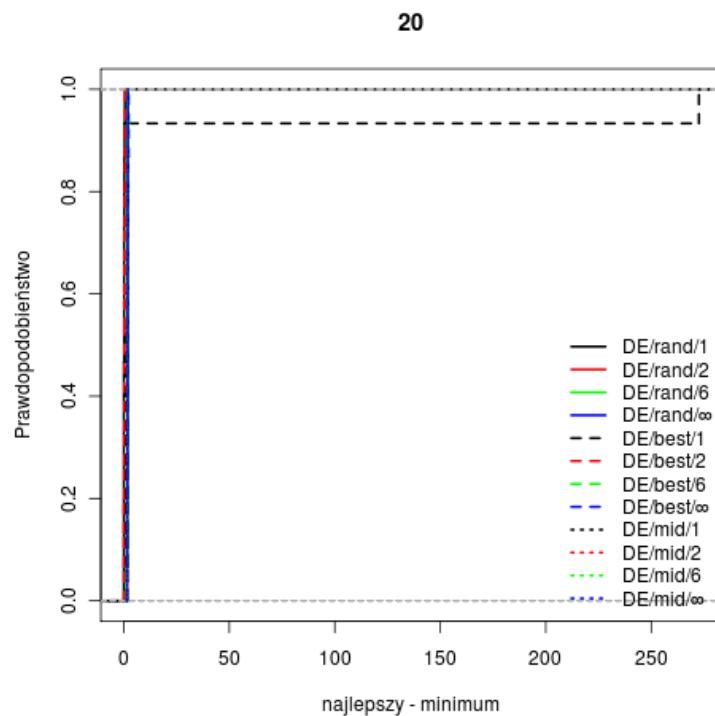
Rysunek B.1. Dystrybuanty empiryczne dla 10-wymiarowej funkcji numer 15



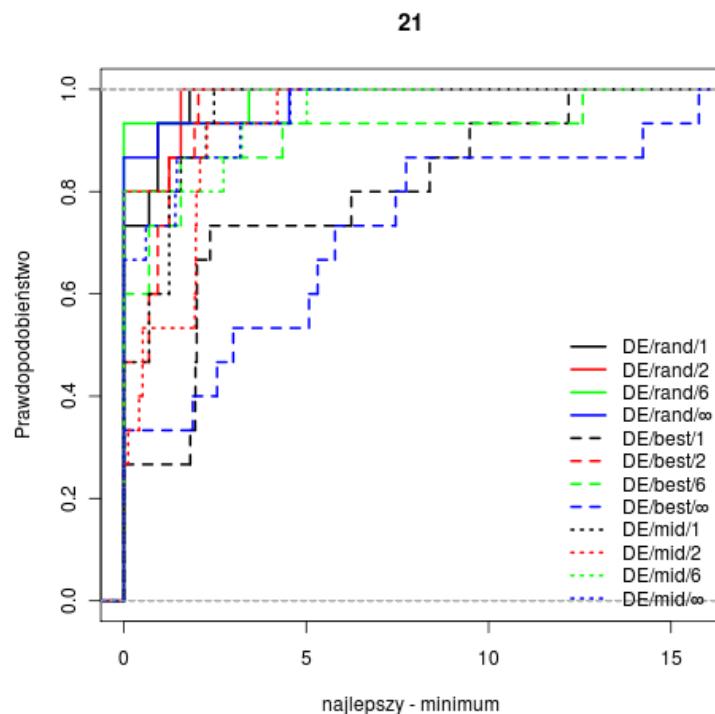
Rysunek B.2. Dystrybuanty empiryczne dla 10-wymiarowej funkcji numer 16



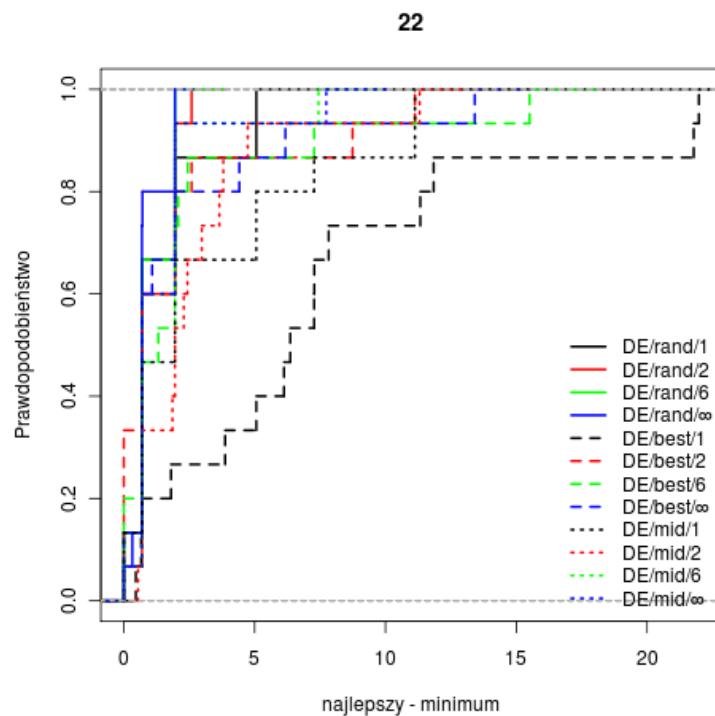
Rysunek B.3. Dystrybuanty empiryczne dla 10-wymiarowej funkcji numer 19



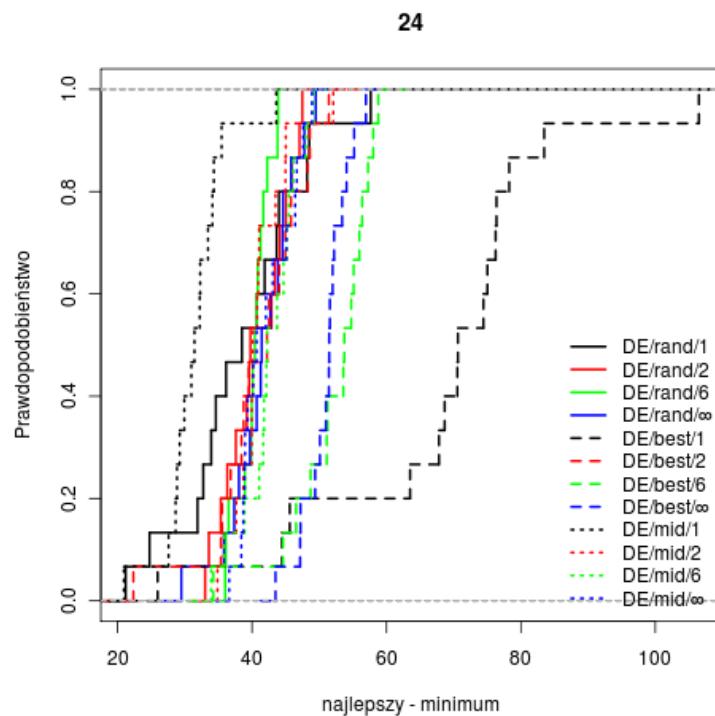
Rysunek B.4. Dystrybuanty empiryczne dla 10-wymiarowej funkcji numer 20



Rysunek B.5. Dystrybuanty empiryczne dla 10-wymiarowej funkcji numer 21



Rysunek B.6. Dystrybuanty empiryczne dla 10-wymiarowej funkcji numer 22



Rysunek B.7. Dystrybuanty empiryczne dla 10-wymiarowej funkcji numer 24

B.2. Wyniki uzyskane dla 20 wymiarów

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/2	+	+	+	+	.	.	+
DE/rand/6	+	+	+	+	+	.	+
DE/rand/ ∞	+	+	+	+	+	.	+
DE/best/1	+	+	+	+	+	+	+
DE/best/2	+	+	+	+	+	+	+
DE/best/6	+	+	+	+	+	+	+
DE/best/ ∞	+	+	+	+	+	+	+
DE/mid/1	.	+	.	+	.	.	+
DE/mid/2	+	+	+	+	+	.	+
DE/mid/6	+	+	+	+	+	.	+
DE/mid/ ∞	+	+	+	+	+	.	+

Tabela B.5. Porównanie DE/rand/1 do reszty algorytmów w 20 wymiarach

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/1	-	-	-	-	-	-	-
DE/rand/2	.	+	.	+	-	-	-
DE/rand/6	+	+	.	+	.	-	-
DE/rand/ ∞	+	+	.	+	.	-	-
DE/best/2	+	+	+	+	.	.	+
DE/best/6	+	+	+	+	+	+	+
DE/best/ ∞	+	+	+	+	+	+	+
DE/mid/1	-	+	-	+	-	-	-
DE/mid/2	-	+	-	+	.	-	-
DE/mid/6	.	+	.	+	.	-	-
DE/mid/ ∞	.	+	.	+	.	-	-

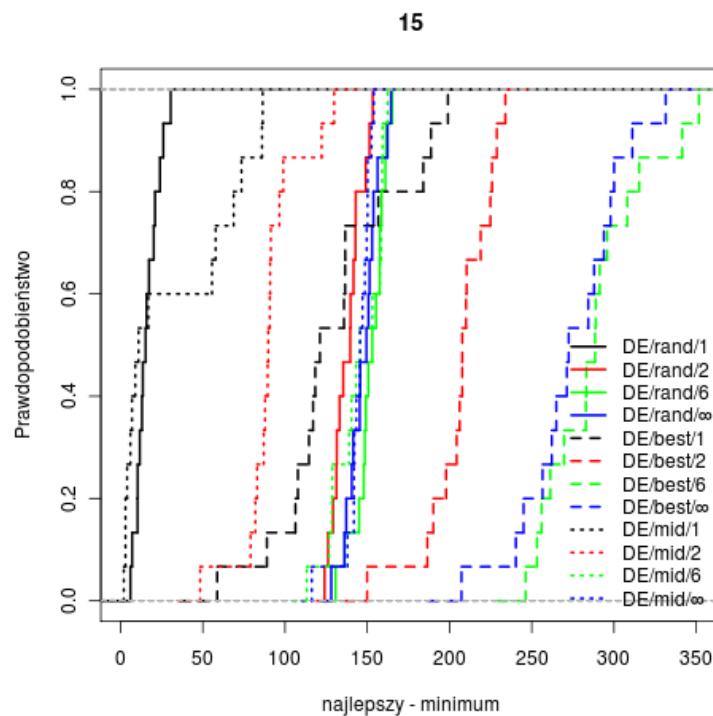
Tabela B.6. Porównanie DE/best/1 do reszty algorytmów w 20 wymiarach

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/1	.	-	.	-	.	.	-
DE/rand/2	+	+	+	.	.	.	+
DE/rand/6	+	+	+	+	+	.	+
DE/rand/ ∞	+	+	+	.	+	.	+
DE/best/1	+	-	+	-	+	+	+
DE/best/2	+	+	+	+	+	+	+
DE/best/6	+	+	+	+	+	+	+
DE/best/ ∞	+	+	+	+	+	+	+
DE/mid/2	+	+	+	.	+	.	+
DE/mid/6	+	+	+	.	+	.	+
DE/mid/ ∞	+	+	+	.	+	.	+

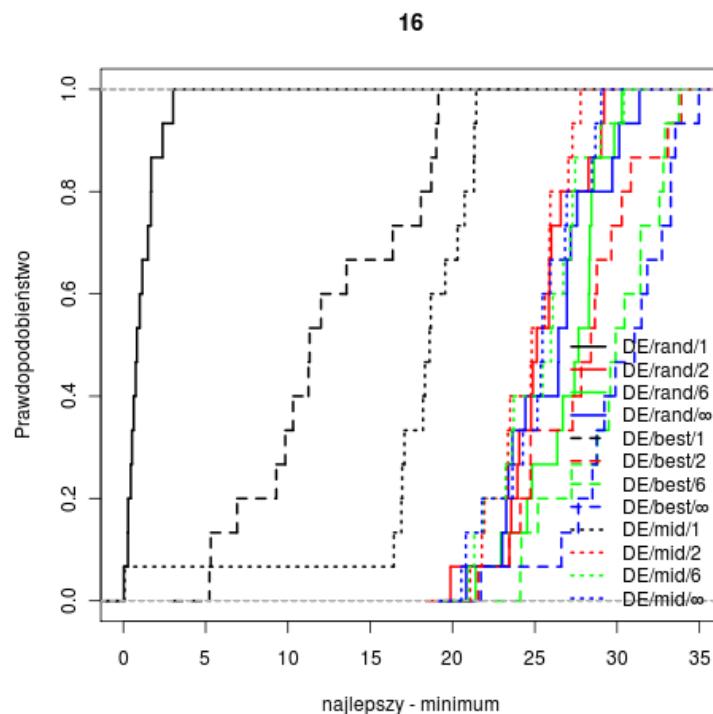
Tabela B.7. Porównanie DE/mid/1 do reszty algorytmów w 20 wymiarach

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/1	12	38	2	2	4	5	0
DE/rand/2	30	47	13	5	23	20	0
DE/rand/6	33	47	16	5	31	27	0
DE/rand/ ∞	34	47	18	5	33	28	0
DE/best/1	8	21	1	0	1	2	0
DE/best/2	21	22	11	8	15	16	0
DE/best/6	23	21	15	13	15	15	1
DE/best/ ∞	23	21	14	12	15	16	1
DE/mid/1	3	42	0	0	0	0	0
DE/mid/2	23	47	9	3	25	18	0
DE/mid/6	32	47	18	5	31	26	0
DE/mid/ ∞	32	47	17	4	32	25	0
średnia	23	37	11	5	19	17	0

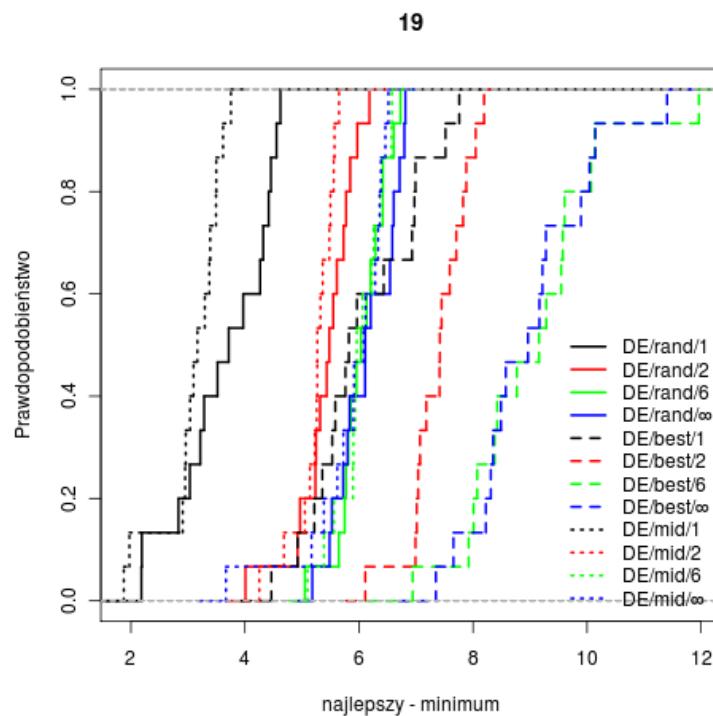
Tabela B.8. Średni % osobników poza zbiorem dopuszczalnym w 20 wymiarach



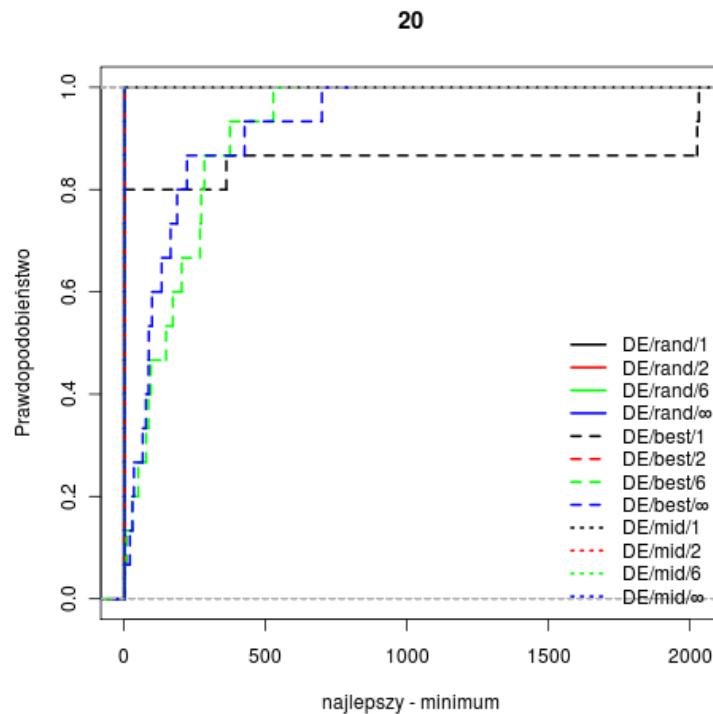
Rysunek B.8. Dystrybuanty empiryczne dla 20-wymiarowej funkcji numer 15



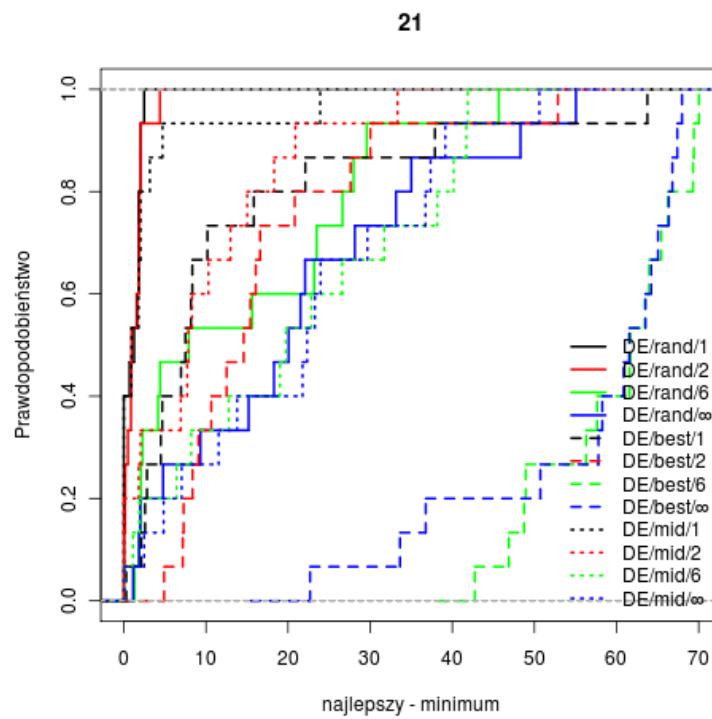
Rysunek B.9. Dystrybuanty empiryczne dla 20-wymiarowej funkcji numer 16



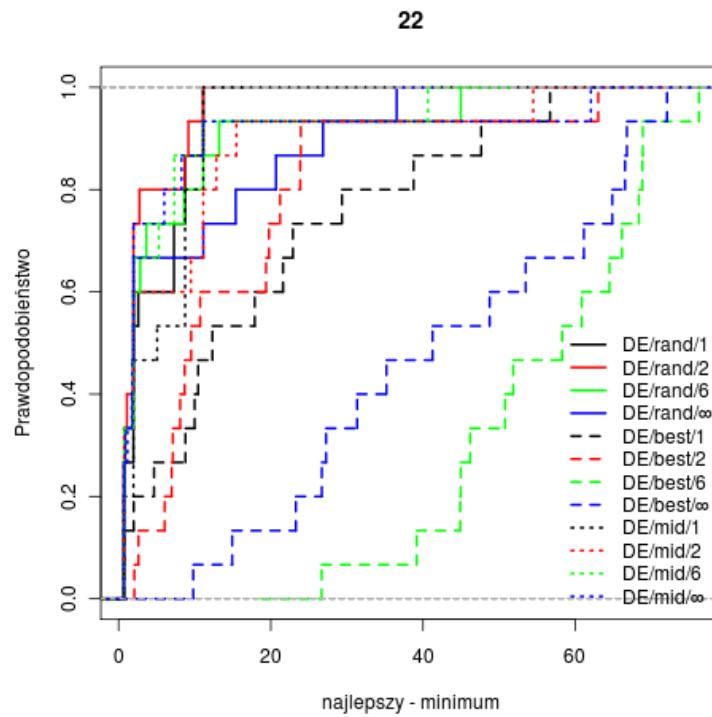
Rysunek B.10. Dystrybuanty empiryczne dla 20-wymiarowej funkcji numer 19



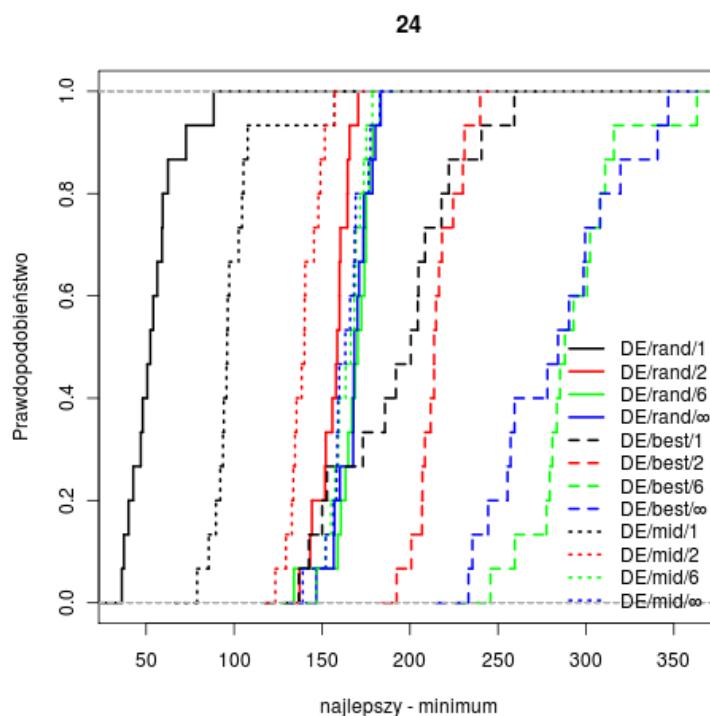
Rysunek B.11. Dystrybuanty empiryczne dla 20-wymiarowej funkcji numer 20



Rysunek B.12. Dystrybuanty empiryczne dla 20-wymiarowej funkcji numer 21



Rysunek B.13. Dystrybuanty empiryczne dla 20-wymiarowej funkcji numer 22



Rysunek B.14. Dystrybuanty empiryczne dla 20-wymiarowej funkcji numer 24

B.3. Wyniki uzyskane dla 40 wymiarów

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/2	+	+	+	+	+	+	+
DE/rand/6	+	+	+	+	+	+	+
DE/rand/ ∞	+	+	+	+	+	+	+
DE/best/1	+	-	+	+	+	+	+
DE/best/2	+	+	+	+	+	+	+
DE/best/6	+	+	+	+	+	+	+
DE/best/ ∞	+	+	+	+	+	+	+
DE/mid/1	-	.	-	+	.	.	-
DE/mid/2	+	+	+	+	+	+	+
DE/mid/6	+	+	+	+	+	+	+
DE/mid/ ∞	+	+	+	+	+	+	+

Tabela B.9. Porównanie DE/rand/1 do reszty algorytmów w 40 wymiarach

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/1	-	+	-	-	-	-	-
DE/rand/2	+	+	+	+	+	+	+
DE/rand/6	+	+	+	+	+	+	+
DE/rand/ ∞	+	+	+	+	+	+	+
DE/best/2	+	+	+	+	+	+	+
DE/best/6	+	+	+	+	+	+	+
DE/best/ ∞	+	+	+	+	+	+	+
DE/mid/1	-	+	-	-	-	-	-
DE/mid/2	-	+	.	-	.	.	-
DE/mid/6	+	+	+	+	+	+	+
DE/mid/ ∞	+	+	+	+	+	+	+

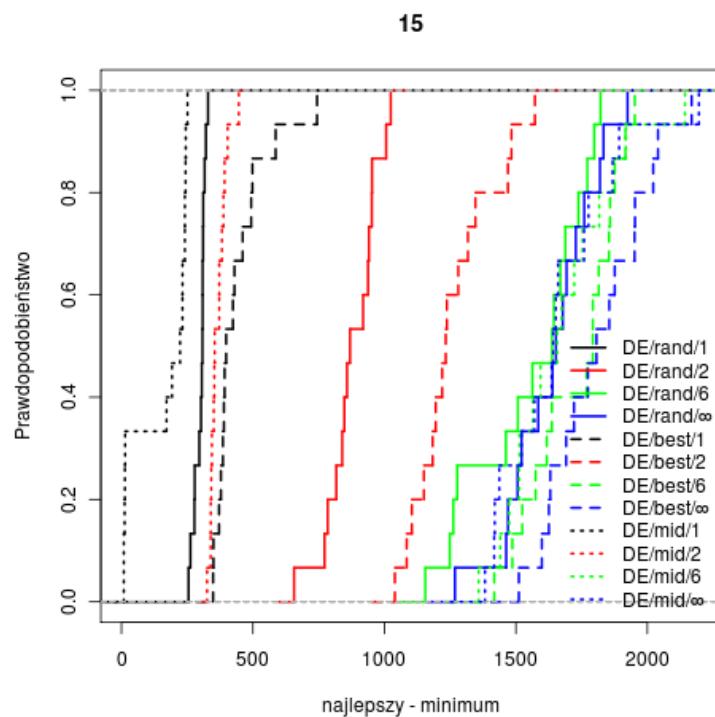
Tabela B.10. Porównanie DE/best/1 do reszty algorytmów w 40 wymiarach

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/1	+	.	+	-	.	.	+
DE/rand/2	+	+	+	+	+	+	+
DE/rand/6	+	+	+	+	+	+	+
DE/rand/ ∞	+	+	+	+	+	+	+
DE/best/1	+	-	+	+	+	+	+
DE/best/2	+	+	+	+	+	+	+
DE/best/6	+	+	+	+	+	+	+
DE/best/ ∞	+	+	+	+	+	+	+
DE/mid/2	+	+	+	+	+	+	+
DE/mid/6	+	+	+	+	+	+	+
DE/mid/ ∞	+	+	+	+	+	+	+

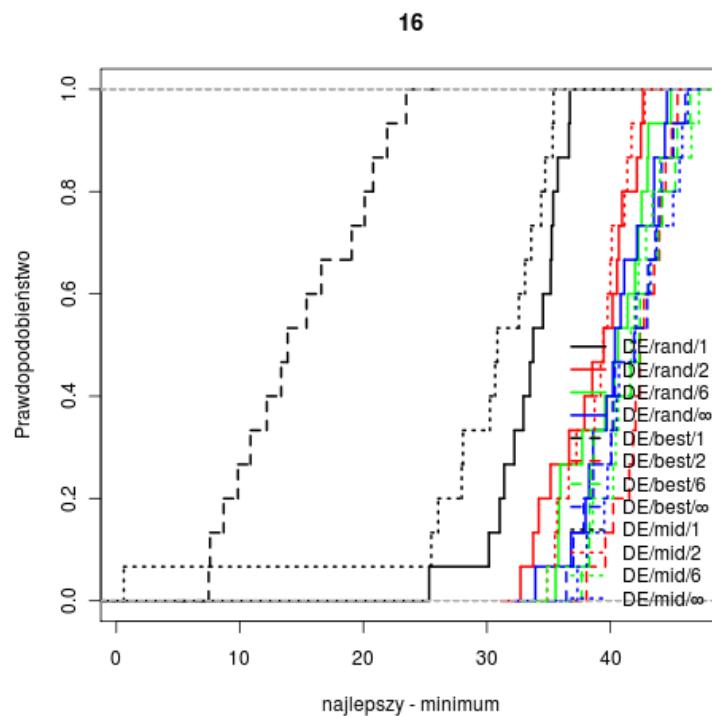
Tabela B.11. Porównanie DE/mid/1 do reszty algorytmów w 40 wymiarach

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/1	31	42	3	6	17	16	0
DE/rand/2	47	42	35	45	14	12	2
DE/rand/6	44	41	35	43	4	3	0
DE/rand/ ∞	43	40	33	42	3	2	0
DE/best/1	24	19	0	1	4	11	0
DE/best/2	20	18	10	19	0	0	0
DE/best/6	17	16	6	15	0	0	0
DE/best/ ∞	16	16	6	15	0	0	0
DE/mid/1	7	45	0	0	1	1	0
DE/mid/2	46	43	17	26	26	25	2
DE/mid/6	44	40	35	43	4	3	0
DE/mid/ ∞	43	38	35	44	5	3	0
średnia	32	33	18	25	7	6	0

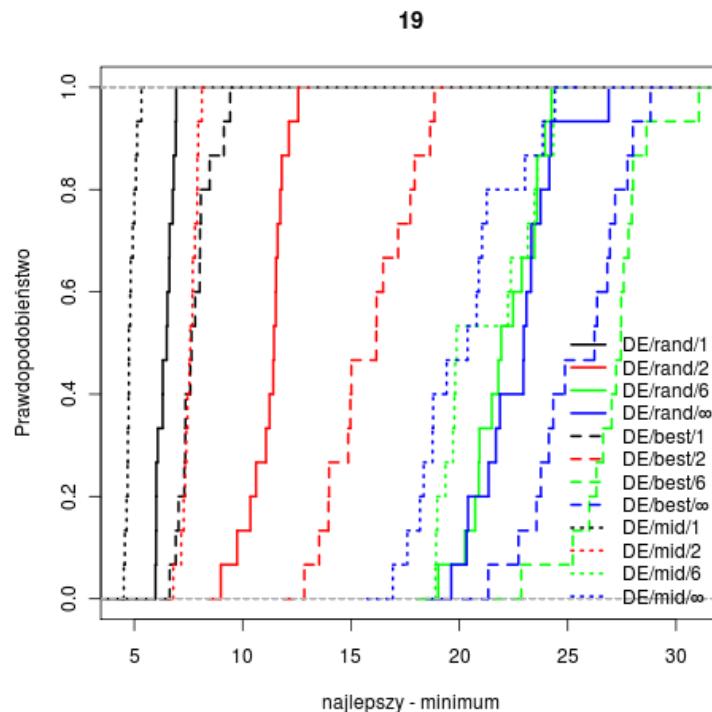
Tabela B.12. Średni % osobników poza zbiorem dopuszczalnym w 40 wymiarach



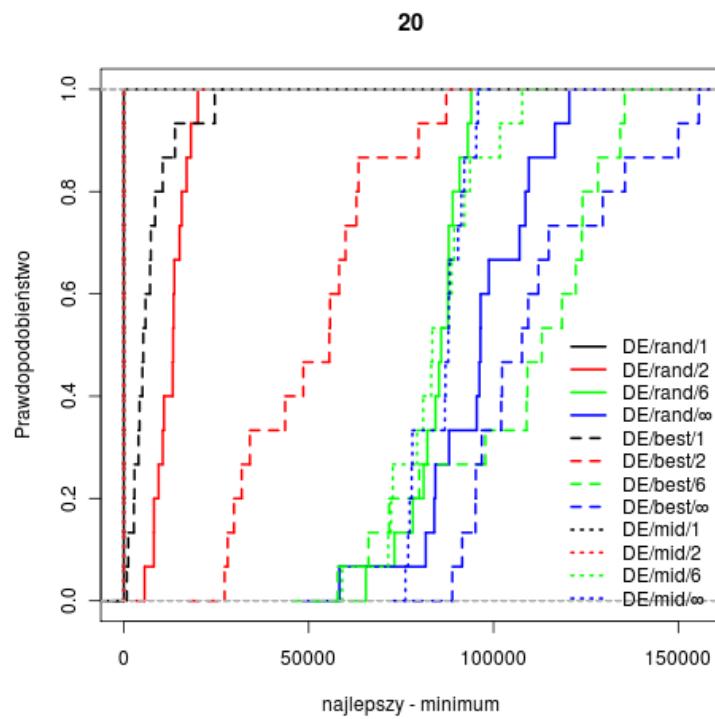
Rysunek B.15. Dystrybuanty empiryczne dla 40-wymiarowej funkcji numer 15



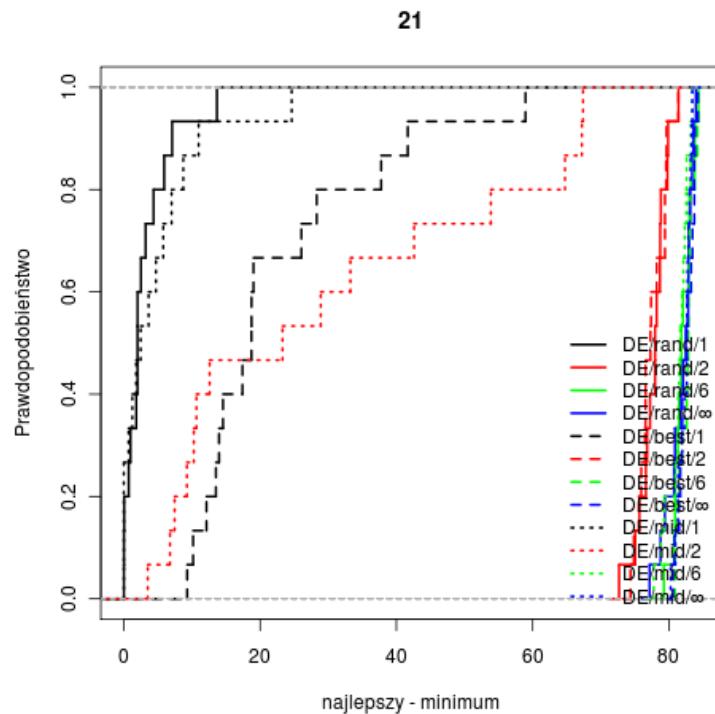
Rysunek B.16. Dystrybuanty empiryczne dla 40-wymiarowej funkcji numer 16



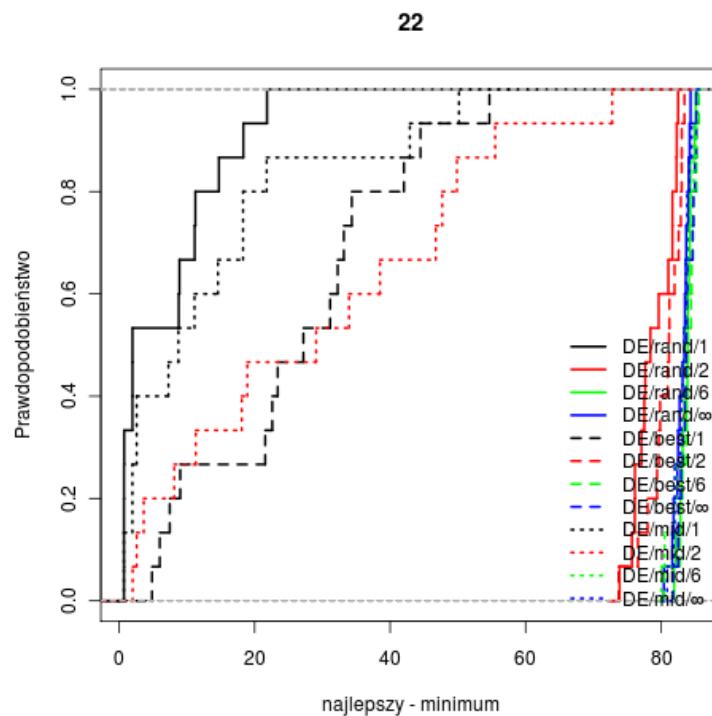
Rysunek B.17. Dystrybuanty empiryczne dla 40-wymiarowej funkcji numer 19



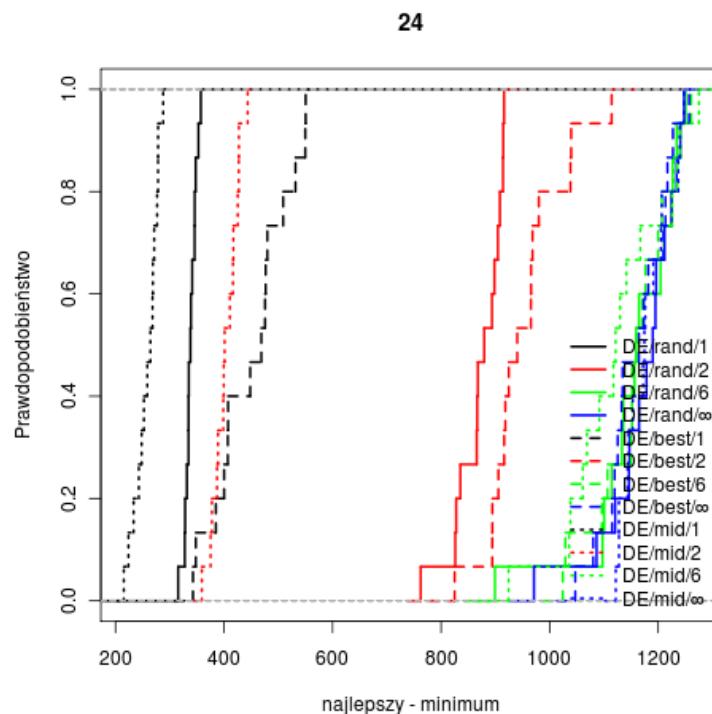
Rysunek B.18. Dystrybuanty empiryczne dla 40-wymiarowej funkcji numer 20



Rysunek B.19. Dystrybuanty empiryczne dla 40-wymiarowej funkcji numer 21



Rysunek B.20. Dystrybuanty empiryczne dla 40-wymiarowej funkcji numer 22



Rysunek B.21. Dystrybuanty empiryczne dla 40-wymiarowej funkcji numer 24

B.4. Wyniki uzyskane dla 80 wymiarów

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/2	+	+	+	+	+	+	+
DE/rand/6	+	+	+	+	+	+	+
DE/rand/ ∞	+	+	+	+	+	+	+
DE/best/1	+	-	+	+	+	.	+
DE/best/2	+	+	+	+	+	+	+
DE/best/6	+	+	+	+	+	+	+
DE/best/ ∞	+	+	+	+	+	+	+
DE/mid/1	-	.	-	-	-	-	-
DE/mid/2	+	+	+	+	+	+	+
DE/mid/6	+	+	+	+	+	+	+
DE/mid/ ∞	+	+	+	+	+	+	+

Tabela B.13. Porównanie DE/rand/1 do reszty algorytmów w 80 wymiarach

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/1	-	+	-	-	-	.	-
DE/rand/2	+	+	+	+	+	+	+
DE/rand/6	+	+	+	+	+	+	+
DE/rand/ ∞	+	+	+	+	+	+	+
DE/best/2	+	+	+	+	+	+	+
DE/best/6	+	+	+	+	+	+	+
DE/best/ ∞	+	+	+	+	+	+	+
DE/mid/1	-	+	-	-	-	-	-
DE/mid/2	+	+	-	-	+	+	+
DE/mid/6	+	+	+	+	+	+	+
DE/mid/ ∞	+	+	+	+	+	+	+

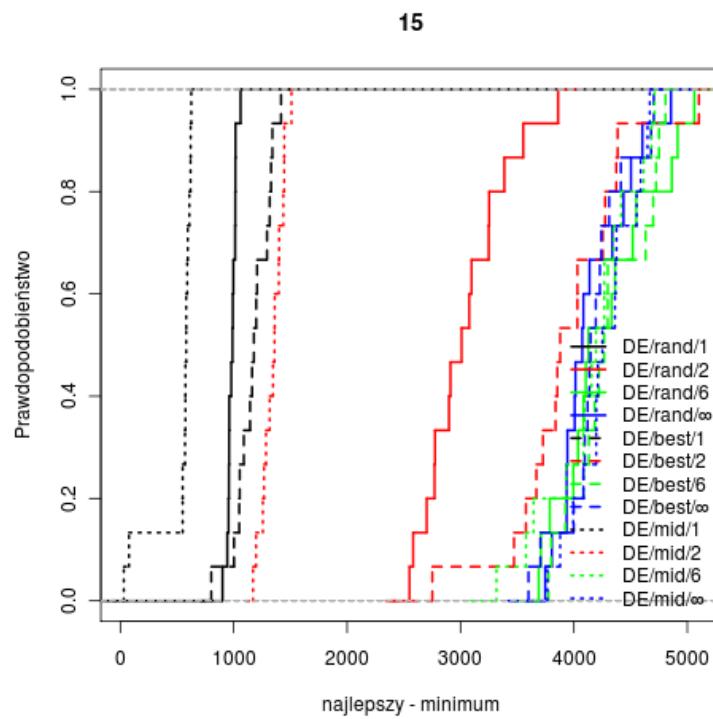
Tabela B.14. Porównanie DE/best/1 do reszty algorytmów w 80 wymiarach

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/1	+	.	+	+	+	+	+
DE/rand/2	+	+	+	+	+	+	+
DE/rand/6	+	+	+	+	+	+	+
DE/rand/ ∞	+	+	+	+	+	+	+
DE/best/1	+	-	+	+	+	+	+
DE/best/2	+	+	+	+	+	+	+
DE/best/6	+	+	+	+	+	+	+
DE/best/ ∞	+	+	+	+	+	+	+
DE/mid/2	+	+	+	+	+	+	+
DE/mid/6	+	+	+	+	+	+	+
DE/mid/ ∞	+	+	+	+	+	+	+

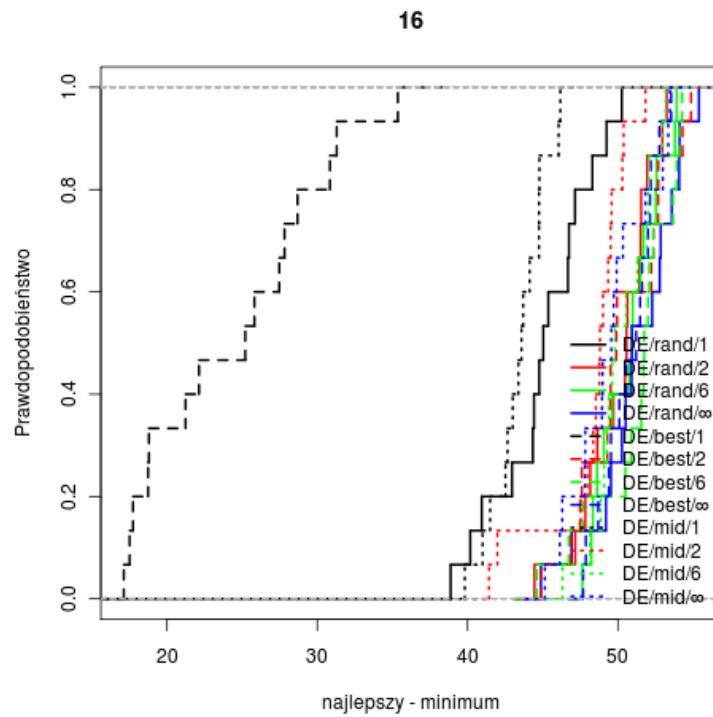
Tabela B.15. Porównanie DE/mid/1 do reszty algorytmów w 80 wymiarach

Algorytm	Numer funkcji testowej						
	15	16	19	20	21	22	24
DE/rand/1	40	39	10	20	16	17	1
DE/rand/2	39	36	18	32	0	0	0
DE/rand/6	31	32	4	21	0	0	0
DE/rand/ ∞	30	30	3	19	0	0	0
DE/best/1	22	20	1	2	11	10	0
DE/best/2	12	14	1	8	0	0	0
DE/best/6	8	11	0	4	0	0	0
DE/best/ ∞	8	9	0	4	0	0	0
DE/mid/1	14	46	1	1	2	3	0
DE/mid/2	45	39	30	40	1	0	0
DE/mid/6	32	30	4	22	0	0	0
DE/mid/ ∞	28	26	3	19	0	0	0
średnia	26	28	6	16	3	3	0

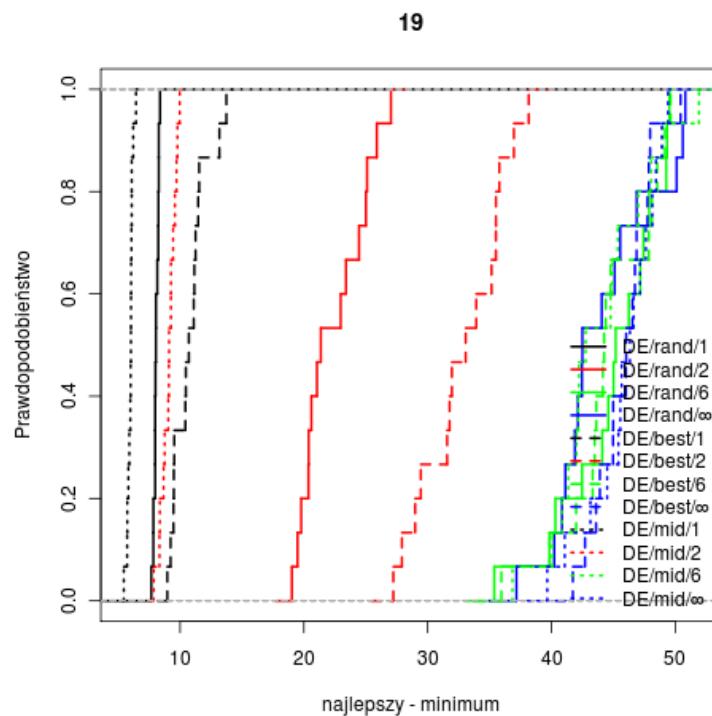
Tabela B.16. Średni % osobników poza zbiorem dopuszczalnym w 80 wymiarach



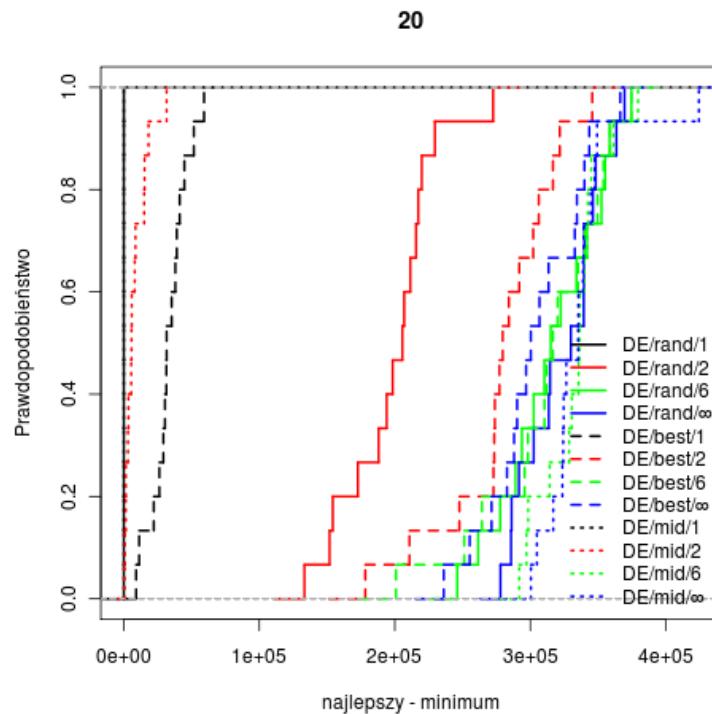
Rysunek B.22. Dystrybuanty empiryczne dla 80-wymiarowej funkcji numer 15



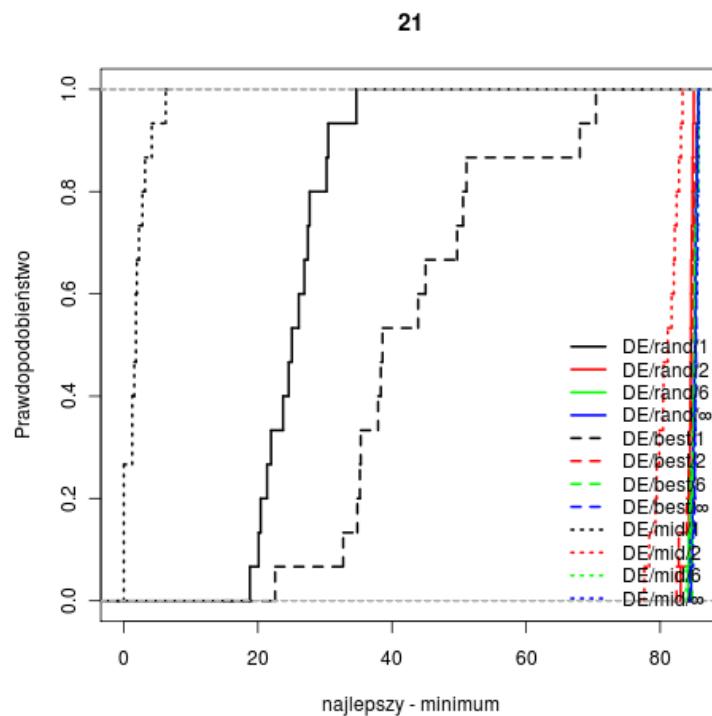
Rysunek B.23. Dystrybuanty empiryczne dla 80-wymiarowej funkcji numer 16



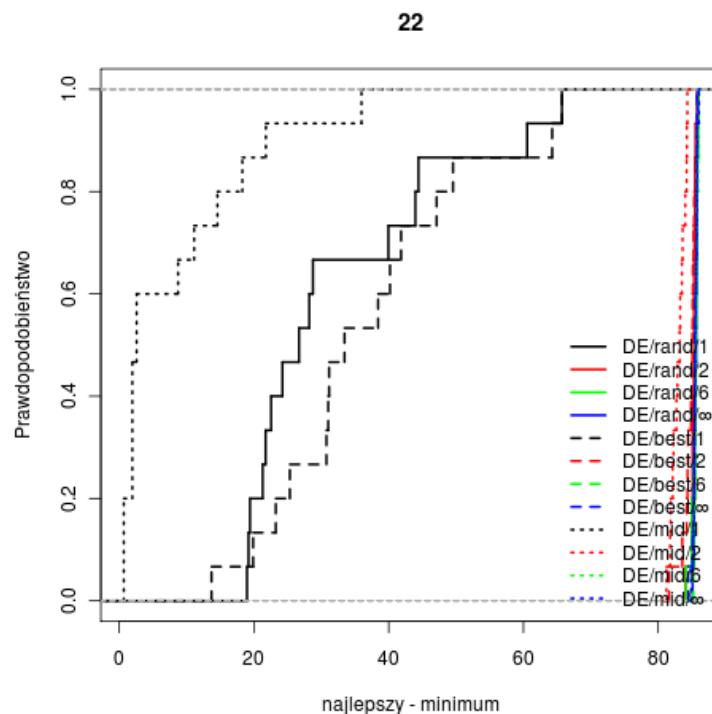
Rysunek B.24. Dystrybuanty empiryczne dla 80-wymiarowej funkcji numer 19



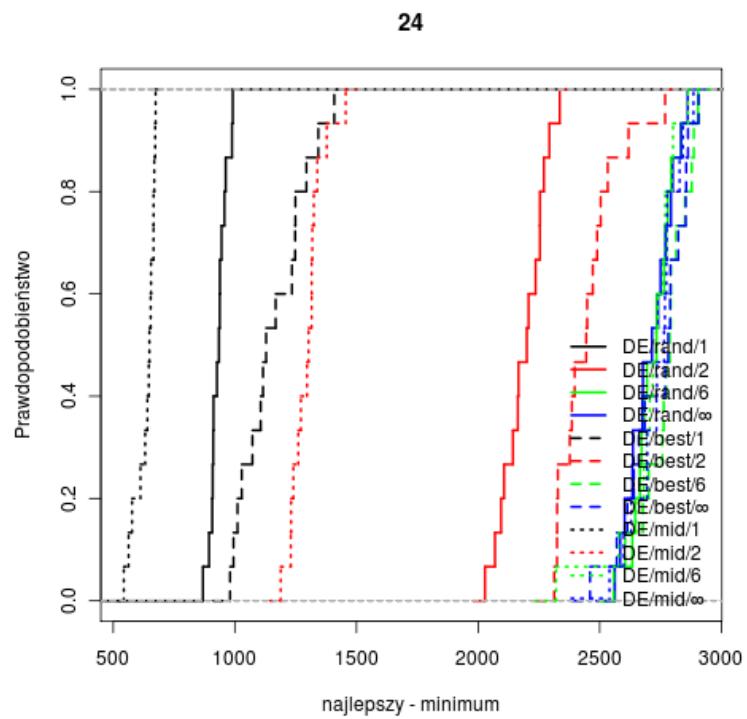
Rysunek B.25. Dystrybuanty empiryczne dla 80-wymiarowej funkcji numer 20



Rysunek B.26. Dystrybuanty empiryczne dla 80-wymiarowej funkcji numer 21



Rysunek B.27. Dystrybuanty empiryczne dla 80-wymiarowej funkcji numer 22



Rysunek B.28. Dystrybuanty empiryczne dla 80-wymiarowej funkcji numer 24

Bibliografia

- [1] Jarosław Arabas. *Wykłady z algorytmów ewolucyjnych*. WNT, Warszawa, 2001.
- [2] Jarosław Arabas, Karol Opara. *Decomposition and Metaoptimization of Mutation Operator in Differential Evolution. Swarm and Evolutionary Computation*, (LNCS 7269), 2012.
- [3] Jarosław Arabas, Adam Szczepankiewicz, Tomasz Wroniak. *Experimental Comparison of Methods to Handle Boundary Constraints in Differential Evolution. Proceedings of the 11th International Conference on Parallel Problem Solving from Nature: Part II*, PPSN'10, strony 411–420, Berlin, Heidelberg, 2010. Springer-Verlag.
- [4] Anne Auger, Steffen Finck, Nikolaus Hansen, Raymond Ros. *Real-Parameter Black-Box Optimization Benchmarking 2013: Presentation of the Noiseless Functions*. 2013. <http://coco.lri.fr/downloads/download13.09/bbobdocfunctions.pdf>.
- [5] Anne Auger, Steffen Finck, Nikolaus Hansen, Raymond Ros. *Real-Parameter Black-Box Optimization Benchmarking 2013: Presentation of the Noisy Functions*. 2013. <http://coco.lri.fr/downloads/download13.09/bbobdocfunctions.pdf>.
- [6] Anne Auger, Steffen Finck, Nikolaus Hansen, Raymond Ros. *Real-Parameter Black-Box Optimization Benchmarking: Experimental Setup*. 2013. <http://coco.lri.fr/downloads/download13.09/bbobdocexperiment.pdf>.
- [7] Janez Brest, Saso Greiner, Borko Boskovic, Marjan Mernik, Viljem Zumer. *Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. Transaction on Evolutionary Computation*, wolumen 10, strony 646–657. IEEE Press, 2006.
- [8] Nikolaus Hansen, Andreas Ostermeier. *Completely Derandomized Self-Adaptation in Evolution Strategies. Evolutionary Computation*, wolumen 9, strony 159–195. MIT Press, 2001.
- [9] Richard Lowry. *Concepts & Applications of Inferential Statistics*. <http://vassarstats.net/textbook/ch12a.html>.
- [10] Henry Mann, Donald Whitney. *On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. Annals of Mathematical Statistics*, wolumen 18, strony 50–60, 1947.
- [11] Jani Ronkkonen, Saku Kukkonen, Kenneth Price. *Real-parameter optimization with differential evolution. Evolutionary Computation, 2005. The 2005 IEEE Congress on*, wolumen 1, strony 506–513, 2005.
- [12] Rainer Storn, Kenneth Price. *Differential Evolution – A simple and efficient adaptive scheme for global optimization over continuous spaces*. 1995.
- [13] Ponnuthurai Suganthan, Nikolaus Hansen, J. Liang, K. Deb, Y. Chen, Anne Auger, S. Tiwari. *Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization*. 2005.
- [14] Tomasz Szarek. *O wymiarze wykresu funkcji nigdzie nieróżniczkowalnej. Wiadomości Matematyczne*, wolumen 42, Poznań, 2006. Polskie Towarzystwo Matematyczne.
- [15] David H. Wolpert, William G. Macready. *No free lunch theorems for optimization. IEEE Transactions On Evolutionary Computation*, wolumen 1, strony 67–82, 1997.

-
- [16] Daniela Zaharie. *Differential Evolution: from Theoretical Analysis to Practical Insights.* 2012.
 - [17] Jingqiao Zhang, Arthur Sanderson. *An approximate gaussian model of Differential Evolution with spherical fitness functions.* IEEE Congress on Evolutionary Computation, strony 2220–2228. IEEE, 2007.
 - [18] Jingqiao Zhang, Arthur Sanderson. *JADE: Adaptive Differential Evolution With Optional External Archive.* *Evolutionary Computation, IEEE Transactions on*, wolumen 13, strony 945–958, 2009.
 - [19] Wen-Jun Zhang, Xiao-Feng Xie. *DEPSO: hybrid particle swarm with differential evolution operator.* *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, wolumen 4, strony 3816–3821 vol.4, 2003.