Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Instytut Informatyki

Rozprawa doktorska

Metoda optymalizacji inspirowana algorytmami ewolucji różnicowej oraz CMA-ES

mgr inż. Dariusz Jagodziński

Numer albumu: 5598

promotor

dr hab. inż. Jarosław Arabas, prof. ucz.

WARSZAWA 2023

Streszczenie

Niniejsza rozprawa prezentuje metodę optymalizacji bazującą na generowaniu kolejnych populacji punktów w taki sposób, że ich wektor średni oraz macierz kowariancji są definiowane za pomocą formuł analogicznych do ewolucyjnej strategii adaptacji macierzy kowariancji (CMA-ES). W przeciwieństwie do metody CMA-ES, która generuje nowe punkty przy użyciu wielowymiarowego rozkładu normalnego, z jawnie określoną macierzą kowariancji, wprowadzona metoda wykorzystuje kombinacje wektorów różnic między punktami archiwalnymi oraz jednowymiarowych wektorów losowych o rozkładzie normalnym o kierunku wzdłuż przeszłych przesunięć punktów środkowych populacji. Został zdefiniowany algorytm różnicowej strategii ewolucyjnej (DES), będący skrzyżowaniem ewolucji różnicowej (DE) i CMA-ES. Algorytm został wszechstronnie przebadany w użyciem benchmarków z rodziny CEC oraz BBOB. Przedstawione w ramach rozprawy wyniki wskazują, że DES jest konkurencyjny wobec CMA-ES w zadaniach optymalizacji zarówno lokalnej, jak i globalnej.

Słowa kluczowe: **Ewolucyjna strategia adaptacji macierzy kowariancji, Ewolucja różnicowa, metody optymalizacji**

Abstract

This thesis presets an optimization method for generating new populations of points such that their mean vector and the covariance matrix are defined by formulas analogous to the Covariance Matrix Adaptation Evolution Strategy (CMA-ES). In contrast to CMA-ES, which generates new individuals using multivariate Gaussian distribution with an explicitly defined covariance matrix, the introduced method uses combinations of difference vectors between archived individuals and univariate Gaussian random vectors along directions of past shifts of the population midpoints. This idea was used to formulate the Differential Evolution Strategy (DES), an algorithm that is a crossover between Differential Evolution (DE) and CMA-ES. The algorithm has been extensively tested using CEC and BBOB benchmarks. The numerical results presented in the thesis indicate that DES is competitive against CMA-ES in performing both local and global optimization.

Keywords: **Covariance Matrix Adaptation Evolution Strategy, Differential Evolution, optimization methods**

Spis treści

Rozdział 1

Wprowadzenie

1.1 Motywacja

Metody optymalizacyjne opisują sposób przechodzenia pomiędzy możliwymi rozwiązaniami w celu uzyskania optymalnego rozwiązania zadanego problemu. Opisywane w tej rozprawie algorytmy będą stosowane do rozwiązywania problemów minimalizacji. Niech punkt $\mathbf{x}$ będzie jednoznacznie identyfikowany poprzez pojedynczy wektor cech długości n (wartość ta zwana jest również wymiarowością problemu i w takim kontekście oznaczana będzie dalej poprzez n).

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]$$

Niech $\mathbb{X}$ będzie przestrzenią przeszukiwań, $\mathbb{F}$ jej podzbiorem $\mathbb{F} \subseteq \mathbb{X}$, a każdy punkt $\mathbf{x}$ posiada wartości wyłącznie ze zbioru $\mathbb{F}$ (zbiór ten zwany będzie dalej również zbiorem dopuszczalnym). Funkcja celu q punktu $\mathbf{x}$ będzie wówczas odwzorowaniem jego cech do współczynnika jakości w przestrzeni liczb rzeczywistych $q : \mathbb{F} \longrightarrow R^1$. Problem optymalizacji z ograniczeniami jest zdefiniowany jako poszukiwanie punktu $\mathbf{x}^*$ takiego że:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{F}} q(\mathbf{x}).$$

Rozwiązywanie zadań maksymalizacji jest możliwe poprzez sformułowanie równoważnego problemu dualnego:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{F}} -q(\mathbf{x}).$$

Zbiór dopuszczalny, jako podzbiór przestrzeni $\mathbb{X}$, spełnia następujące warunki:

$$g_i(\mathbf{x}) \leqslant 0 \qquad i = 1, \ldots, n_g$$

$$h_j(\mathbf{x}) = 0 \qquad j = 1, \ldots, n_h$$

Funkcje g_i oraz h_j opisują odpowiednio nierównościowe oraz równościowe ograniczenia problemu. Ograniczenia kostkowe są szczególnym przypadkiem ograniczeń nierównościowych, w których zakłada się, że przestrzeń przeszukiwań jest n-wymiarowym zbiorem wektorów liczb rzeczywistych, $\mathbb{X} = \mathbb{R}^n$, oraz że dla każdego wymiaru istnieje górna oraz dolna wartość ograniczająca. W takim podejściu, dla ograniczeń kostkowych, zbiór rozwiązań dopuszczalnych jest n-wymiarowym hipersześcianem zdefiniowanym następująco:

$$\mathbb{F} = \{\mathbf{x} : l_i \leqslant x_i \leqslant u_i, \quad i = 1, \ldots, n\},$$

Powyższy opis stanowi definicję problemu optymalizacji z ograniczeniami kostkowymi, której rozwiązania dotyczyć będą dalsze części rozprawy. Z uwagi na wysoką wymiarowość oraz nieliniowy charakter wielu rzeczywistych zadań podlegających optymalizacji, typowym podejściem do rozwiązywania takich problemów jest stosowanie algorytmów aproksymacyjnych. Metody te, w przeciwieństwie do algorytmów dokładnych, służą do znajdowania rozwiązań przybliżonych. Szczególnym przypadkiem tej koncepcji jest podejście heurystyczne. Tym co odróżnia metody heurystyczne od aproksymacyjnych jest to, że metody heurystyczne nie dostarczają informacji o jakości zwracanego rozwiązania, w stosunku do rozwiązania optymalnego. Aproksymacyjne podejście jest więc zazwyczaj stosowane w przypadku problemów algorytmicznie zbadanych, o znanej przybliżonej lub dokładnej wartości rozwiązania optymalnego. W przypadku, gdy problem, dla którego poszukiwane jest rozwiązanie, nie jest poznany, lub gdy pełny algorytm z przyczyn technicznych jest zbyt kosztowny, używa się metod znajdowania rozwiązań przybliżonych. Algorytmy te mogą być heurystykami, specjalizowanymi pod konkretny, rozważany typ zadania i przestrzeń problemu $\mathbb{X}$, bądź metaheurystykami podającymi sposób na utworzenie odpowiedniego algorytmu niezależnie od przestrzeni przeszukiwań czy wiedzy eksperckiej.

Każda z metaheurystyk definiuje swój własny sposób przeszukiwania przestrzeni. Fakt ten może implikować dla różnych metaheurystyk i skończonego budżetu (określonego w postaci czasowej lub liczby obliczeń wartości funkcji celu) różnymi jakościami rozwiązań. Zgodnie z twierdzeniem NFL Davida Wolperta i Williama Macreadyego [60] nie istnieje najlepszy, uniwersalny dla wszystkich zadań, algorytm optymalizacyjny. Niezależnie od przyjętej miary jakości algorytmu optymalizacyjnego, dowolne dwa różne algorytmy

optymalizacyjne zachowują się średnio tak samo, dla wszystkich zadań optymalizacyjnych. Twierdzenie NFL działa również w kontekście parametryzacji jednego algorytmu przeszukiwań. Niezależnie od przyjętej miary jakości algorytmu optymalizacyjnego, dwa dowolne zestawy parametrów dla jednego algorytmu przeszukiwań, zachowują się średnio tak samo, dla wszystkich zadań optymalizacyjnych.

Skoro strategia optymalizacyjna zawsze przewyższająca jakością inne, dla dowolnego problemu, jest nierealizowalna, należy szukać algorytmów, które będą najlepsze w kontekście rozwiązywania problemów, dla których użycie innych niż heurystycznych metod (np. z powodu nieznajomości nawet przybliżonej wartości punktu optymalnego) nie jest w praktyce możliwe. Wyniki konkursów optymalizacyjnych dla problemów z ograniczeniami kostkowymi wskazują, że metaheurystyki populacyjne, bazujące na algorytmach ewolucyjnych z rodzin ewolucji różnicowej i CMA-ES, są liderami na tym polu.

Algorytmy ewolucyjne używają mechanizmów inspirowanych mechanizmami ewolucji biologicznej, takimi jak mutacja czy krzyżowanie, które w swojej idei bazują na przypadkowych zmianach. Generowanie nowych punktów w przestrzeni przeszukiwań jest definiowanie pośrednio poprzez wybór metody próbkowania losowego, czyli poprzez ustalenie rozkładów prawdopodobieństwa dla mechanizmów ewolucyjnych, oraz metod aktualizacji parametrów owych rozkładów. Obecnie zauważyć można dużą popularność metod używających rozkładu normalnego jako funkcji gęstości rozkładu prawdopodobieństwa. Jednym z powodów jest stabilność tego typu rozkładu: suma dwóch zmiennych losowych o rozkładzie normalnym ma dalej rozkład normalny, co pomaga w procesie projektowania i analizowania algorytmu. Najpopularniejszymi metodami ewolucyjnymi, bazującymi na wielowymiarowym rozkładzie normalnym, są algorytmy z rodziny CMA-ES. Ewolucyjna strategia adaptacji macierzy kowariancji używa macierzy kowariancji $\mathbf{C}$, wyestymowanej z aktualnej populacji, w celu aktualizacji parametrów rozkładu normalnego używanego do generowania kolejnych generacji punktów. Ta idea, rozszerzając niejako pojęcie wariancji na wiele wymiarów, implementuje adaptacyjny proces dopasowania parametrów rozkładu, do kształtu optymalizowanej funkcji celu. Zwiększa to prawdopodobieństwo obecności przyszłych, lepszych, punktów w kolejnej generacji. Cecha ta, zwana dopasowaniem do poziomic funkcji celu, jest najbardziej pożądaną właściwością dla wszystkich algorytmów optymalizacji stochastycznej. Idealny algorytm powinien wiązać prawdopodobieństwo obecności każdego punktu w kolejnej populacji, z poziomem jego jakości (wartością

funkcji celu). Implementacja własności dopasowania do poziomic funkcji celu w algorytmie CMA-ES jest obarczona problemem wysokiego poziomu niezbędnego nakładu obliczeniowego. Zadanie pełnej faktoryzacji macierzy kowariancji $\mathbf{C}$ do postaci $\mathbf{A}\mathbf{A}^T = \mathbf{C}$ lub $\mathbf{A}\mathbf{D}\mathbf{A}^T = \mathbf{C}$ posiada nakład obliczeniowy $O(n^3)$. W niektórych implementacjach CMA-ES, faktoryzacja ta jest niezbędna w każdej generacji, by móc utworzyć nową populację zgodnie z wielowymiarowym rozkładem normalnym i macierzą kowariancji $\mathbf{C}$. Duży nakład obliczeniowy metod z rodziny CMA-ES sprawia, że ich stosowanie w problemach o wysokiej wymiarowości jest ograniczone.

Opisane problemy numeryczne wynikają z wysokiego poziomu skomplikowania mechanizmów matematycznych, na których oparta jest strategia ewolucyjna CMA-ES. Równie szeroko stosowaną metaheurystyką populacyjną jest ewolucja różnicowa. Metoda ta implementuje znacząco prostszy mechanizm mutacji, bazujący na idei skalowanego różnicowania punktów w populacji. Klasyczna ewolucja różnicowa, posiadając nakład obliczeniowy liniowo zależny od wymiarowości, może być łatwiej niż CMA-ES wykorzystywana dla problemów o dużej skali trudności. Metoda ta nie posiada jednakże wykazanej wprost cechy dopasowywania się do poziomic funkcji celu, może również wykazywać dużą zależność skuteczności działania od pierwszej populacji lub kształtu funkcji celu. Dla metod rodziny ewolucji różnicowej znany jest problem jej nieodporności na obroty przestrzeni przeszukiwań (w przypadku ewolucji różnicowej z aktywnym mechanizmem krzyżowania).

Obie wiodące metaheurystki populacyjne posiadają cechy, na których opiera się ich skuteczność oraz popularność. Każda z nich posiada jednakże wady, które uniemożliwiają uniwersalne zastosowanie każdej z nich do dowolnego problemu. Wydaje się, że idealnym łącznikiem pomiędzy tymi dwiema rodzinami metaheurystyk byłby algorytm, który implementowałby sposób działania metody CMA-ES i cechę dopasowania do kształtu funkcji celu, używając do tego nieskomplikowanej numerycznie idei mutacji różnicowej. Taka metaheurystyka miałaby wówczas zbiór zalet umożliwiający jej uniwersalne zastosowanie, niezależnie od typu problemu.

1.2 Cel rozprawy

Celem rozprawy jest sformułowanie efektywnego algorytmu bazującego na generowaniu nowych punktów w sposób analogiczny do algorytmu CMA-ES, jednak niewykorzystują-

cego złożonych operacji macierzowych.

Autor zaproponuje metodę, która generuje kolejne populacje z użyciem mutacji różnicowej w taki sposób, że empiryczna macierz kowariancji i wartość średnia populacji będą wyznaczane podobnie jak w algorytmie CMA-ES. W ramach rozprawy przeprowadzona będzie analiza empiryczna skuteczności działania zaproponowanego algorytmu z użyciem zadań zdefiniowanych na potrzeby konkursów optymalizacyjnych. Jakość tego algorytmu zostanie porównana z wynikami wiodących metod biorących udział w tych konkursach, co pozwoli na weryfikację efektywności zaproponowanej metody.

1.3 Struktura rozprawy

Główna treść rozprawy została podzielona na trzy części. W rozdziale 2 zaprezentowany został przegląd metaheurystycznych metod optymalizacji z rodzin CMA-ES i DE. Przedstawione zostały podstawy teoretyczne działania, scharakteryzowane zostały wady oraz zaprezentowane zostały nowoczesne linie rozwojowe tych technik. Rozdział 3 wprowadza definicję algorytmu różnicowej strategii ewolucyjnej (DES) oraz ilustruje jej działanie w kontekście metody CMA-ES oraz zjawiska dopasowywania się rozkładów generowanych punktów do poziomic funkcji celu. Rozdział 4 obrazuje skuteczność działania zaproponowanego przez autora algorytmu DES, w kontekście konkursów optymalizacyjnych, na tle wiodących rozwiązań.

Rozdział 2

Metaheurystyczne metody optymalizacji

Taksonomia metaheurystyk, z uwagi na brak jednoznacznych kryteriów uznania metody za metaheurystyką, jest trudna do ustalenia. Algorytmy te często bazują na analogiach do procesów ze świata rzeczywistego, które można interpretować w kategoriach optymalizacji. Dziedzina pojęć i definicje czerpane są wówczas z domeny wiedzy, z której pochodzi dana metafora. Ten fakt czyni zadanie konceptualizacji domeny metaheurystyk problemem, w którym budowa podstawowego systemu definicji może nie być możliwa do ustalenia. Kategoryzacja metaheurystyk jest możliwa w kontekście pewnego, rozważanego, zbioru rodzin algorytmów.

Istnieje wiele prób podziału metaheurystyk, zgodnie z pewną, zaproponowaną taksonomią, zwykle w oparciu o jedną lub kilka wyróżnionych cech. Blum oraz Roli[9] specyfikują grupę cech, które mogą być kryteriami klasyfikacji metaheurystyk. Są nimi takie właściwości jak: źródło inspiracji metody (inspirowane biologicznie lub nie), sposób przeszukiwania przestrzeni problemu (lokalna bądź globalna), przeszukiwanie z wykorzystaniem populacji bądź pojedynczego punktu, sposób użycia funkcji celu (niezmienność funkcji celu w czasie działania metody) oraz sposób uwzględniania historii (używanie historii przeszukiwanych punktów bądź nie). Zbiór tych cech nie jest jednak wystarczający do szczegółowej klasyfikacji każdej możliwej metaheurystyki, lecz może stanowić pewne kryteria budowy systemu taksonomicznego. Przykładów takich systemów dokonujących kategoryzacji dziedziny metaheurystyk jest tak wiele, że istnieją prace przeglądowe(np.[52]) dokonujące analizy i porównania najpopularniejszych taksonomii, które starają się sklasyfikować

obecnie wiodące rozwiązania rodzin algorytmów.

Wyniki konkursów optymalizacyjnych z ostatnich lat, takich jak BBOB (Black Box Optimization Benchmarking)[21], czy CEC (Congress on Evolutionary Computation)[5] wskazują, że wiodącymi metaheurystykami dla wielowymiarowych problemów optymalizacji ciągłej są metody z rodziny CMA-ES[17] oraz DE[53]. Z uwagi na cel niniejszej rozprawy przedstawiony w Rozdziale 1, poniżej przedstawione zostaną charakterystyki wyłącznie tych dwóch metod.

2.1 Ewolucyjna strategia adaptacji macierzy kowariancji

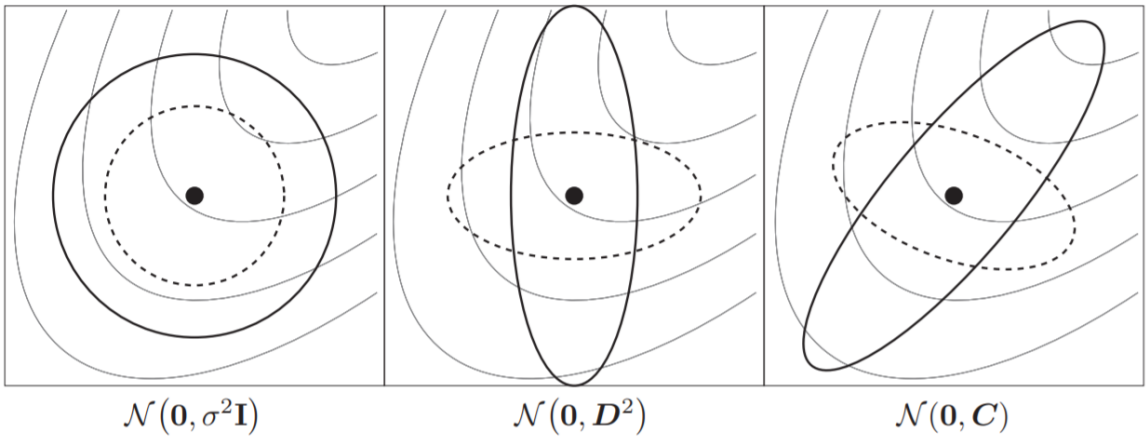
Metoda CMA-ES[17] jest stochastyczną metaheurystyką populacyjną do optymalizacji ciągłej, dla problemów zarówno liniowych, jak i nieliniowych. Jest to algorytm ewolucyjny bazujący na takich mechanizmach ewolucji biologicznej jak mutacja i selekcja. W każdej generacji nowe punkty są generowane z użyciem mutacji o stochastycznych podstawach. Następnie nowa populacja jest poddawana selekcji na podstawie funkcji celu q, w celu wyznaczenia punktów będących rodzicami dla kolejnej generacji. W ten sposób w kolejnych generacjach generowane są punkty o możliwe wysokiej jakości.

W CMA-ES iteracyjne przybliżana jest dodatnio określona macierz kowariancji, która dla wypukłych funkcji kwadratowych jest bliska odwrotności hesjanu funkcji celu q [14]. Metoda CMA-ES jest algorytmem ewolucyjnym do optymalizacji lokalnej[27], która może być jednak stosowana z powodzeniem do optymalizacji globalnej[23] [18]. Metoda jest typowo stosowana do rozwiązywania problemów optymalizacyjnych z ograniczeniami kostkowymi lub bez ograniczeń, w których wymiarowość problemu powinna zawierć się w przedziale $3 \leqslant n \leqslant 100$.

2.1.1 Podstawy teoretyczne

Wielowymiarowy rozkład normalny

Algorytm CMA-ES jest populacyjnym algorytmem ewolucyjnym, który bazuje na stochastycznej generacji punktów zgodnie z wielowymiarowym rozkładem normalnym. Wielowymiarowy rozkład normalny $\mathcal{N}(\mathbf{m}, \mathbf{C})$ jest złożeniem wielu jednowymiarowych rozkładów



Rysunek 2.1: Linia przerywana oraz pogrubiona pokazuje kształt możliwych elipsoid jednakowej gęstości rozkładu normalnego, gdzie $\mathbf{I}$ oznacza macierz jednostkową, $\mathbf{D}$ macierz diagonalną, a $\mathbf{C}$ pełną macierz kowariancji. Linia szara ukazuje poziomice funkcji celu.

Źródło: [16]

normalnych w każdym z wymiarów i posiada jedno-modalną funkcję gęstości o środku w punkcie $\mathbf{m} \in \mathbb{R}^n$. Wielowymiarowy rozkład normalny $\mathcal{N}(\mathbf{m}, \mathbf{C})$ jest jednoznacznie opisany przez wektor wartości oczekiwanej $\mathbf{m}$ oraz dodatnio określoną, symetryczną macierz kowariancji $\mathbf{C} \in \mathbb{R}^{n \times n}$. Macierz kowariancji $\mathbf{C}$ posiada także interpretację geometryczną definiując jednoznacznie hiper-elipsoidę $\{\mathbf{x} \in \mathbb{R}^n | \mathbf{x}^T \mathbf{C}^{-1} \mathbf{x} = 1\}$, będącą poziomicą funkcji gęstości prawdopodobieństwa rozkładu $\mathcal{N}(\mathbf{0}, \mathbf{C})$. Rysunek 2.1 pokazuje kształt elipsoid jednakowego poziomu gęstości prawdopodobieństwa, dla dwuwymiarowego rozkładu normalnego, o macierzach kowariancji zadanych w różny sposób. Tylko pełna (nie diagonalna) macierz kowariancji sprawia, że rozkład (oznaczony na rysunku poprzez $\mathcal{N}(\mathbf{0}, \mathbf{C})$) posiada możliwość dopasowania się do poziomic dowolnej kwadratowej funkcji celu. Macierz kowariancji zadana w postaci macierzy diagonalnej, posiada maksymalną liczbę stopni swobody równą wymiarowości problemu. W przypadku gdy $\mathbf{C} = \sigma^2\ \mathbf{I}$ liczba ta równa jest jeden. Taki rozkład posiada jedynie zdolność do całkowicie symetrycznego przeskalowywania się na skutek zmiany wartości odchylenia standardowego (rozkład jest izotropowy). Gdy $\mathbf{C} = \ \mathbf{D}^{\ 2}$ właściwość dopasowywania się rozkładu jest niezależna dla każdego wymiaru sprawiając, że osie główne elipsoid jednakowej gęstości prawdopodobieństwa mogą być skalowane równolegle do osi każdego z wymiarów problemu. Macierz kowariancji, która nie jest diagonalna, posiada dodatkowe stopnie swobody wynikające z uwzględniania zależności pomiędzy wymiarami problemu, co daje możliwość obrotu elipsoidy jednakowej

gęstości prawdopodobieństwa względem osi problemu.

Realizacja wprost wielowymiarowego rozkładu normalnego o zadanej pełnej macierzy kowariancji jest zadaniem nietrywialnym obliczeniowo. Wzór $\mathcal{N}(\mathbf{m}, \mathbf{C})$ można jednakże przekształcić w następujący sposób:

$$\begin{aligned} \mathcal{N}(\mathbf{m}, \mathbf{C}) &\sim \mathbf{m} + \mathcal{N}(\mathbf{0}, \mathbf{C}) \\ &\sim \mathbf{m} + \mathbf{C}^{\frac{1}{2}}\mathcal{N}(\mathbf{0}, \mathbf{I}) \end{aligned} \tag{2.1}$$

Stosując rozkład macierzy według wartości własnych (rozkład spektralny), macierz C może być przedstawiona w postaci rozkładu $\mathbf{C} = \mathbf{B}\mathbf{D}^2\mathbf{B}^T$ gdzie:

B jest macierzą ortogonalną ($\mathbf{B}^{-1} = \mathbf{B}^T$ oraz $\mathbf{B}\mathbf{B}^T = \mathbf{I}$), w której kolumny są bazami ortonormalnymi wektorów własnych macierzy **C**

D jest macierzą diagonalną, której przekątną tworzą pierwiastki wartości własnych macierzy **C**. $\mathbf{D}^2 = \mathbf{D}\ \mathbf{D} = diag(d_1, \ldots, d_n)^2 = diag(d_1^2, \ldots, d_n^2)$, gdzie d_i^2 jest i-tą wartością własną macierzy **C**

Wzór na dekompozycje spektralną pozwala na obliczenie pierwiastka macierzy $\mathbf{C}^{\frac{1}{2}}$. Stosując serię przekształceń i podstawień[16, Rozdział 0.1] można otrzymać wzór:

$$\mathbf{C}^{\frac{1}{2}} = \mathbf{B}\mathbf{D}\mathbf{B}^T \tag{2.2}$$

B jest macierzą ortogonalną w której kolumny stanowią bazy ortonormalne, więc każda kolumna posiada normę równą $\|B_k\|^2 = 1$, gdzie k to numer kolumny macierzy B. Można więc przyjąć, że

$$\mathbf{B}^T\mathcal{N}(\mathbf{0}, \mathbf{I}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{2.3}$$

Z użyciem (2.1), (2.2) oraz (2.3) można zapisać wzór na $\mathcal{N}(\mathbf{m}, \mathbf{C})$ w postaci:

$$\begin{aligned} \mathcal{N}(\mathbf{m}, \mathbf{C}) &\sim \mathbf{m} + \mathbf{C}^{\frac{1}{2}}\mathcal{N}(\mathbf{0}, \mathbf{I}) \\ &\sim \mathbf{m} + \mathbf{B}\mathbf{D}\mathbf{B}^T\mathcal{N}(\mathbf{0}, \mathbf{I}) \\ &\sim \mathbf{m} + \mathbf{B}\mathbf{D}\mathcal{N}(\mathbf{0}, \mathbf{I}) \end{aligned} \tag{2.4}$$

Algorytm CMA-ES generuje punkty zgodnie z wielowymiarowym rozkładem normalnym $\mathcal{N}(\mathbf{m}, \mathbf{C})$. Wzór (2.4) pozwala uprościć proces generowania rozkładu wykorzystując w tym celu sferyczny (izotropowy) rozkład normalny $\mathcal{N}(\mathbf{0}, \mathbf{I})$, który może być z łatwością numerycznie realizowany na komputerach.

Empiryczna macierz kowariancji

Metoda CMA-ES bazuje na generacji punktów z użyciem wielowymiarowego rozkładu normalnego opartego o macierz kowariancji. Jeżeli macierz kowariancji rozkładu wynosi $\mathbf{C}$, to empiryczna macierz kowariancji zbioru λ niezależnych realizacji tego rozkładu wynosi $\mathbf{C}_e$. W celu wyznaczenia $\mathbf{C}_e$ używana jest wygenerowana z rozkładem normalnym populacja punktów $\mathbf{x}_1, \ldots, \mathbf{x}_\lambda$, o liczebności λ. Empiryczna macierz kowariancji jest wówczas zadana wzorem [16, Rozdział 3.1]:

$$\mathbf{C}_e = \frac{1}{\lambda - 1} \sum_{i=1}^{\lambda} \left(\mathbf{x}_i - \frac{1}{\lambda} \sum_{j=1}^{\lambda} \mathbf{x}_j \right) \left(\mathbf{x}_i - \frac{1}{\lambda} \sum_{j=1}^{\lambda} \mathbf{x}_j \right)^T \tag{2.5}$$

W przypadku jednoczesnego szacowania wariancji i średniej ze skończonej próby, tak jak ma to miejsce we wzorze (2.5), należy stosować poprawkę Bessela aby zredukować obciążenie estymatora. Z tego powodu w mianowniku we wzorze 2.5 występuje wartość $\lambda - 1$. Poprawka ta nie jest potrzebna, jeśli do obliczeń wykorzystuje się wartość oczekiwaną zamiast średniej populacyjnej. Macierz $\mathbf{C}_e$ jest nieobciążonym estymatorem macierzy kowariancji $\mathbf{C}$, przy założeniach niezależności generowania losowych punktów $\mathbf{x}_1, \ldots, \mathbf{x}_\lambda$. Czynnik $\sum_{j=1}^{\lambda} \mathbf{x}_j$ wyznacza średni punkt populacji, realizując empiryczną estymatę wartości oczekiwanej rozkładu. Wartość ta jest jednak znana dla każdej generacji i wynosi $\mathbf{m}$. Wówczas zamiast wzoru (2.5) można użyć zależności:

$$\mathbf{C}_e = \frac{1}{\lambda} \sum_{i=1}^{\lambda} \left(\mathbf{x}_i - \mathbf{m} \right) \left(\mathbf{x}_i - \mathbf{m} \right)^T \tag{2.6}$$

Zastąpienie punktu średniego aktualnie zrealizowanej próby (wygenerowanej populacji), punktem $\mathbf{m}$ będącym wartością oczekiwaną zmiennej losowej dodaje jeden stopień swobody. Wymusza to zmianę normalizacji z $\frac{1}{\lambda-1}$ na $\frac{1}{\lambda}$ w celu zachowania braku obciążenia estymatora macierzy kowariancji. Algorytm CMA-ES posługuje się definicją ważonej macierzy kowariancji [23]:

$$\begin{aligned} \mathbf{C}_\mu &= \sum_{i=1}^{\mu} w_i \left(\mathbf{x}_{i:\lambda} - \mathbf{m} \right) \left(\mathbf{x}_{i:\lambda} - \mathbf{m} \right)^T \\ \sum_{i=1}^{\mu} w_i &= 1, \qquad w_1 \geqslant w_2 \geqslant \cdots \geqslant w_\mu \geqslant 0 \qquad 0 < \mu \leqslant \lambda \end{aligned} \tag{2.7}$$

Zmiana ta ma na celu uwzględnić podczas procesu estymacji macierzy kowariancji jakość poszczególnych punktów, poprzez uszeregowanie punktów zgodnie z ich jakością oraz wprowadzenie wektora wag zależnych od pozycji w uszeregowaniu. Wektor wag daje moż-

liwość określenia poziomu wpływu jakości punktów na modyfikacje empirycznej macierzy kowariancji. Jeśli $\mu = \lambda$ oraz $w_{i=1,\ldots,\mu} = \frac{1}{\mu}$ wówczas $\mathbf{C}_{\mu}^{(g+1)} = \mathbf{C}_{e}^{(g+1)}$.

2.1.2 Opis algorytmu CMA-ES

Opisywana w tym rozdziale wersja algorytmu została wprowadzona przez Hansena i Ostermeiera[17] i będzie dalej nazywana wersją pierwotną lub bazową. Bazowy algorytm CMA-ES zakłada, że adaptacja paramentów rozkładu (nowego punktu środkowego $\mathbf{m}$ oraz nowej macierzy kowariancji $\mathbf{C}$) jest zdefiniowana tylko na podstawie grupy μ punktów charakteryzujących się najlepszymi wartościami funkcji celu, gdy wartości wag są jednakowe ($w_i = \frac{1}{\mu}$). Rysunek 2.2 ukazuje zarys bazowego algorytmu CMA-ES. Metoda opiera się na

1: $\mathbf{p}_c^{(1)} \leftarrow \mathbf{0}$, $\mathbf{p}_\sigma^{(1)} \leftarrow \mathbf{0}$, $\mathbf{C}^{(1)} \leftarrow \mathbf{I}$
2: $t \leftarrow 1$
3: inicjuj($\mathbf{m}^{(1)}, \sigma^{(1)}$)
4: **while** !stop **do**
5: **for** $i = 1$ **to** λ **do**
6: $\mathbf{d}_i^{(t)} \sim N(\mathbf{0}, \mathbf{C}^{(t)})$
7: $\mathbf{x}_i^{(t)} \leftarrow \mathbf{m}^{(t)} + \sigma^{(t)}\mathbf{d}_i^{(t)}$
8: **end for**
9: oceń($X^{(t)}$)
10: $\mathbf{\Delta}^{(t)} \leftarrow \sum_{i=1}^{\mu} w_i \mathbf{d}_i^{(t)}$
11: $\mathbf{m}^{(t+1)} \leftarrow \mathbf{m}^{(t)} + \sigma^{(t)}\mathbf{\Delta}^{(t)}$
12: $\mathbf{p}_\sigma^{(t+1)} \leftarrow (1 - c_\sigma)\mathbf{p}_\sigma^{(t)} + \sqrt{\mu c_\sigma (2 - c_\sigma)} \cdot (\mathbf{C}^{(t)})^{-\frac{1}{2}}\mathbf{\Delta}^{(t)}$
13: $\mathbf{p}_c^{(t+1)} \leftarrow (1 - c_c)\mathbf{p}_c^{(t)} + \sqrt{\mu c_c (2 - c_c)} \cdot \mathbf{\Delta}^{(t)}$
14: $\mathbf{C}^{(t+1)} \leftarrow (1 - c_{cov})\mathbf{C}^{(t)} + c_1\mathbf{C}_1^{(t)} + c_\mu\mathbf{C}_\mu^{(t)}$
15: $\sigma^{(t+1)} \leftarrow \sigma^{(t)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma^{(t+1)}\|}{E\|N(\mathbf{0},\mathbf{I})\|} - 1\right)\right)$
16: $t \leftarrow t + 1$
17: **end while**

Rysunek 2.2: Zarys algorytmu CMA-ES

generacji punktów z użyciem wielowymiarowego rozkładu normalnego, scharakteryzowanego przez punkt środkowy $\mathbf{m}^{(t)}$, macierz kowariancji $\mathbf{C}^{(t)}$ oraz współczynnik kroku $\sigma^{(t)}$, gdzie t jest indeksem oznaczającym numer generacji. Macierz kowariancji pierwszej gene-

racji $\mathbf{C}^{(1)}$ jest inicjowana macierzą jednostkową $\mathbf{I}$, a wartości $\mathbf{m}^{(1)}$ oraz $\sigma^{(1)}$ są parametrami algorytmu, zadawanymi przez użytkownika. Tak scharakteryzowany rozkład generuje w każdej generacji λ punktów z przestrzeni problemu. Następnie wszystkie punkty poddawane są ocenie z użyciem funkcji celu, a zbiór μ najlepszych punktów używany jest do aktualizacji parametrów rozkładu.

Generacja punktów

Generacja nowych punktów w populacji $X^{(t)}$, w metodzie CMA-ES następuje w dwustopniowym procesie. W pierwszej fazie tworzony jest zbiór $D^{(t)} = \{\mathbf{d}_i^{(t)}, i = 1, \ldots, \lambda\}$ będący zbiorem punktów bazowych wygenerowanych z rozkładem normalnym o zerowej wartości średniej i macierzy kowariancji $\mathbf{C}^{(t)}$. Oczekiwana wartość macierzy kowariancji empirycznej tego zbioru wynosi $\mathbf{\Sigma}[\mathbf{D}^{(t)}] = \left(1 - \frac{1}{\lambda}\right) \mathbf{C}^{(t)}$.

W drugiej fazie generowane są punkty $\mathbf{x}_i^{(t)}$ będące punktami populacji t i realizowane poprzez zsumowanie punktu środkowego $\mathbf{m}^{(t)}$ i przeskalowanych parametrem $\sigma^{(t)}$ wektorów $\mathbf{d}_i^{(t)}$. Tak utworzona populacja jest następnie ewaluowana poprzez przyporządkowanie każdemu punktowi jego wartości funkcji celu, a następnie sortowana względem tych wartości. Zbiór μ indeksów z przyporządkowanymi wartościami funkcji celu jest zbiorem μ najlepszych punktów populacji $X^{(t)}$. Implementowana jest tym samym selekcja obcinająca oparta na deterministycznej operacji wyboru $\mu < \lambda$ najlepszych spośród λ punktów.

Zbiór μ najlepszych punktów populacji $X^{(t)}$ wykorzystywany jest do adaptacji parametrów rozkładu normalnego, m.in do obliczenia wartości punktu środkowego kolejnej populacji. W tym celu obliczana jest ważona średnia μ najlepszych punktów aktualnej populacji $X^{(t)}$. Wartość oczekiwana $\mathbf{m}^{(t+1)}$ kolejnej populacji jest sumą wartości oczekiwanej aktualnej populacji oraz obliczonej wartości średniej, przeskalowanej poprzez współczynnik kroku. Zgodnie z [20], metoda będzie działać poprawnie nawet w przypadku, gdy mnożnik σ będzie wartością stałą. Adaptacja parametru kroku $\sigma^{(t)}$ zwiększa jednakże tempo zbieżności algorytmu.

Aktualizacja macierzy kowariancji

CMA-ES po każdej iteracji dokonuje adaptacji macierzy kowariancji za pomocą dwóch różnych macierzy opisanych poniżej. Metoda aktualizacji rank-μ definiuje macierz $\mathbf{C}_\mu^{(t)}$

wyznaczaną na podstawie zbioru $D_\mu^{(t)}$ generacji t.

$$\mathbf{C}_\mu^{(t)} = \sum_{i=1}^{\mu} w_i \mathbf{d}_i^{(t)} \left(\mathbf{d}_i^{(t)}\right)^T \tag{2.8}$$

Wagi w_i spełniają założenia zgodne z wzorem (2.7) i dają możliwość określenia poprawki uwzględniającej poziom jakości poszczególnych punktów. Te same wagi używane są podczas wyznaczania wektora $\mathbf{\Delta}^{(t)}$ będącego ważoną średnią wektorów $\mathbf{d}_i^{(t)}$ (rysunek 2.2, linia 10).

Drugą używaną przez CMA-ES metodą aktualizacji jest rank-1, która definiuje macierz $\mathbf{C}_1$. Opiera się ona na wektorze $\mathbf{p}_c^{(t)}$, który jest rezultatem wygładzania wykładniczego wektorów $\mathbf{\Delta}^{(\tau)}, \tau = 1, \ldots, t$.

$$\mathbf{C}_1^{(t)} = \mathbf{p_c^{(t)}} \left(\mathbf{p_c^{(t)}}\right)^T \tag{2.9}$$

Ostateczną macierzą kowariancji kolejnej generacji $\mathbf{C}^{(t+1)}$ jest ważona suma aktualnej macierzy $\mathbf{C}^{(t)}$ oraz macierzy $\mathbf{C}_\mu^{(t)}$ oraz $\mathbf{C}_1^{(t)}$, zgodnie z zależnością:

$$\begin{aligned} &\mathbf{C}^{(t+1)} = (1 - c_{cov})\mathbf{C}^{(t)} + c_1\mathbf{C}_1^{(t)} + c_\mu\mathbf{C}_\mu^{(t)} \\ &c_1, c_\mu, c_{cov} < 1, \quad c_{cov} = c_1 + c_\mu, \quad c_{cov} \leqslant 1. \end{aligned} \tag{2.10}$$

Macierz kowariancji $\mathbf{C}^{(t)}$ jednoznacznie definiuje rozkład wektorów $\mathbf{d}_i^{(t)}$. Zgodnie z (2.9) czynnik $\mathbf{C}_1^{(t)}$ preferuje generowanie wektorów które są zgodne z kierunkiem wektora skumulowanego przesunięcia punktów środkowych $\mathbf{p}_c^{(t)}$. W efekcie zastosowania wzoru 2.10, gdy punkty środkowe poprzednich populacji przesuwały się w tym samym kierunku, w macierzy $\mathbf{C}^{(t)}$ zostaną wydłużone wektory własne, o kierunku zgodnym z przesunięciem $\mathbf{p}_c^{(t)}$. Innymi słowy metoda CMA-ES będzie przeszukiwać przestrzeń problemu z większym krokiem, zgodnym z kierunkiem skumulowanego przesunięcia punktów środkowych, w celu przyspieszenia procesu optymalizacji i osiągnięcia lokalnego optimum.

Zgodnie z (2.10), macierz kowariancji $\mathbf{C}^{(t)}$ zapamiętuje historię różnorodności wektorów $\mathbf{d}_i^{(t)}$ oraz historię zmian punktów środkowych. Wzór ten przyjmuje formę równania rekurencyjnego implementującego wykładnicze wygładzanie średnich wektorów $\mathbf{d}_i^{(t)}$.

Twierdzenie 2.1.1. *Procedura adaptacji macierzy kowariancji(rysunek 2.2, linia 14)*

może być przedstawiona jako:

$$\begin{aligned}\mathbf{C}^{(t)} =& c_\mu \frac{\mu-1}{\mu}\sum_{\tau=1}^{t}(1-c_{cov})^{t-\tau}\mathbf{S}\left(D_\mu^{(\tau)}\right)\\ &+ c_1\sum_{\tau=1}^{t}(1-c_{cov})^{t-\tau}\mathbf{p_c}^{(\tau)}\left(\mathbf{p_c}^{(\tau)}\right)^T\\ &+ c_\mu\sum_{\tau=1}^{t}(1-c_{cov})^{t-\tau}\boldsymbol{\Delta}^{(\tau)}\left(\boldsymbol{\Delta}^{(\tau)}\right)^T\\ &+ (1-c_{cov})^t\mathbf{I}\end{aligned} \tag{2.11}$$

gdzie $D_\mu^{(\tau)}$ jest zbiorem μ najlepszych punktów bazowych, wybranych z zgodnie z wartościami funkcji celu w generacji τ, a $\mathbf{S}(D_\mu^{(\tau)})$ jest empiryczną macierzą kowariancji zbioru $D_\mu^{(\tau)}$.

Dowód. Procedura adaptacji macierzy kowariancji jest liniowym równaniem rekurencyjnym pierwszego rzędu o niejednorodnej budowie, które definiuje sekwencję macierzy $C^{(1)}, C^{(2)}, \ldots, C^{(t)}$. Wzór jawny ciągu ma postać:

$$\mathbf{C}^{(t+1)} = \sum_{\tau=1}^{t}(1-c_{cov})^{t-\tau}\left(c_1\mathbf{C}_1^{(\tau)} + c_\mu\mathbf{C}_\mu^{(\tau)}\right) + (1-c_{cov})^t\mathbf{I} \tag{2.12}$$

Dalsze przekształcenia wzoru (2.12) ułatwia poniższy lemat.

Lemat 2.1.2. *Macierz rank-μ oraz empiryczna macierz kowariancji najlepszych punktów bazowych, pozostają w następującej relacji:*

$$\mathbf{C}_\mu^{(t)} = \frac{\mu-1}{\mu}\mathbf{S}\left(D_\mu^{(t)}\right) + \boldsymbol{\Delta}^{(t)}\left(\boldsymbol{\Delta}^{(t)}\right)^T \tag{2.13}$$

Aby udowodnić lemat 2.1.2 należy przytoczyć wzór na empiryczną macierz kowariancji $\mathbf{S}\left(D_\mu^{(t)}\right)$:

$$\mathbf{S}\left(D_\mu^{(t)}\right) = \frac{1}{\mu-1}\sum_{i=1}^{\mu}(\mathbf{d}_i^{(t)} - \boldsymbol{\Delta}^{(t)})(\mathbf{d}_i^{(t)} - \boldsymbol{\Delta}^{(t)})^T \tag{2.14}$$

Przekształcając wzór (2.8) definiujący macierz $\mathbf{C}_{\mu}^{(t)}$ i stosując wzór (2.14) można otrzymać:

$$\begin{aligned}
\mathbf{C}_{\mu}^{(t)} =& \frac{1}{\mu}\sum_{i=1}^{\mu}\left(\mathbf{d}_i^{(t)} - \boldsymbol{\Delta}^{(t)} + \boldsymbol{\Delta}^{(t)}\right)\left(\mathbf{d}_i^{(t)} - \boldsymbol{\Delta}^{(t)} + \boldsymbol{\Delta}^{(t)}\right)^T \\
=& \frac{1}{\mu}\sum_{i=1}^{\mu}\left(\mathbf{d}_i^{(t)} - \boldsymbol{\Delta}^{(t)}\right)\left(\mathbf{d}_i - \boldsymbol{\Delta}^{(t)}\right)^T \\
&+ \frac{1}{\mu}\sum_{i=1}^{\mu}\left(\mathbf{d}_i^{(t)} - \boldsymbol{\Delta}^{(t)}\right)\left(\boldsymbol{\Delta}^{(t)}\right)^T \\
&+ \frac{1}{\mu}\sum_{i=1}^{\mu}\boldsymbol{\Delta}^{(t)}\left(\mathbf{d}_i^{(t)} - \boldsymbol{\Delta}^{(t)}\right)^T \\
&+ \frac{1}{\mu}\sum_{i=1}^{\mu}\boldsymbol{\Delta}^{(t)}\left(\boldsymbol{\Delta}^{(t)}\right)^T \\
=& \frac{\mu-1}{\mu}\mathbf{S}(D_{\mu}^{(t)}) + \boldsymbol{\Delta}^{(t)}\left(\boldsymbol{\Delta}^{(t)}\right)^T
\end{aligned} \tag{2.15}$$

Stosując równości (2.9) oraz (2.15) we wzorze (2.12) można otrzymać:

$$\begin{aligned}
\mathbf{C}^{(t)} =& \sum_{\tau=1}^{t}(1-c_{cov})^{t-\tau}c_1\left(\mathbf{p}_c^{(\tau)}\right)\left(\mathbf{p}_c^{(\tau)}\right)^T \\
&+ \sum_{\tau=1}^{t}(1-c_{cov})^{t-\tau}c_{\mu}\frac{\mu-1}{\mu}\mathbf{S}(D_{\mu}^{(\tau)}) \\
&+ \sum_{\tau=1}^{t}(1-c_{cov})^{t-\tau}c_{\mu}\boldsymbol{\Delta}^{(\tau)}\left(\boldsymbol{\Delta}^{(\tau)}\right)^T \\
&+ (1-c_{cov})^t\mathbf{I}
\end{aligned}$$

□

Adaptacja długości kroku

Procedura adaptacji macierzy kowariancji w algorytmie CMA-ES posiada w pewnym stopniu zdolność adaptacji długości kroku. Czynnik $\mathbf{C}_1^{(t)}$ posiada zdolność wydłużania wektorów własnych macierzy $\mathbf{C}^{(t)}$, których kierunek jest zgodny z wektorem skumulowanego przesunięcia punktów środkowych $\mathbf{p}_c^{(t)}$, w stopniu zależnym od wartości wag c_1 i c_{μ}, wektora skumulowanego przesunięcia $\mathbf{p}_c^{(t)}$ oraz rekurencyjnego charakteru aktualizacji macierzy $\mathbf{C}^{(t)}$(2.11) implementującego ideę wygładzania wykładniczego. Dynamika procesu adaptacji macierzy kowariancji jest w algorytmie CMA-ES poprawiana poprzez wprowadzenie dodatkowego czynnika - współczynnika kroku - $\sigma^{(t)}$, który reguluje w sposób adaptacyjny ogólny krok procesu przeszukiwania przestrzeni.

Aktualizacja czynnika $\sigma^{(t)}$ opiera się na obserwowaniu dynamiki zmian punktu środkowego populacji z użyciem wektora $\mathbf{p}_{\sigma}^{(t+1)}$. Definicja tego wektora podobna jest do definicji

wektora $\mathbf{p}_c^{(t+1)}$, a sam mechanizm nazywany jest ścieżką ewolucji. W efekcie, algorytm CMA-ES dokonuje adaptacji dwóch ścieżek ewolucji nazywanych odpowiednio anizotropową $\mathbf{p}_c^{(t+1)}$ oraz izotropową $\mathbf{p}_\sigma^{(t+1)}$ Ścieżka ewolucyjna jest sumą punktów środkowych następujących po sobie generacji. W praktyce, ścieżki ewolucyjne w CMA-ES są wektorami, $\mathbf{p}_c^{(t)}, \mathbf{p}_\sigma^{(t)} \in \mathbb{R}^n$, z zaimplementowanym mechanizmem wykładniczego wygładzania o punkcie startowym $\mathbf{p}_c^{(0)}, \mathbf{p}_\sigma^{(0)} = [0]^n$. Sposób aktualizacji $\mathbf{p}_c^{(t+1)}$ (Rysunek 2.2, linia 13) nadaje tej ścieżce ewolucji charakter anizotropowy, zależny od kierunku fluktuacji punktu środkowego $\mathbf{\Delta}^{(t)}$, względem wektorów własnych macierzy $\mathbf{C}$. Oznacza to, że oczekiwana długość ścieżki $\mathbf{p}_c^{(t+1)}$ jest zależna nie tylko od długości wektorów punktów środkowych $\mathbf{\Delta}^{(1,\ldots,t)}$ ale również od ich kierunku. Aktualizacja wektora $\mathbf{p}_\sigma^{(t+1)}$ dokonuje izotropowego sumowania długości wektorów $\mathbf{d}_i^{(t)}$. We wzorze na aktualizacje izotropowej ścieżki ewolucji (Rysunek 2.2 linie 12) używany jest pierwiastek odwrotności aktualnej macierzy kowariancji $(\mathbf{C}^{(t)})^{-\frac{1}{2}}$ do normalizacji wektorów $\mathbf{\Delta}^{(t)}$. Z wzoru (2.2) możemy otrzymać:

$$(\mathbf{C}^{(t)})^{-\frac{1}{2}} = \mathbf{B}^{(t)} \left(\mathbf{D}^{(t)}\right)^{-1} \left(\mathbf{B}^{(t)}\right)^T \tag{2.16}$$

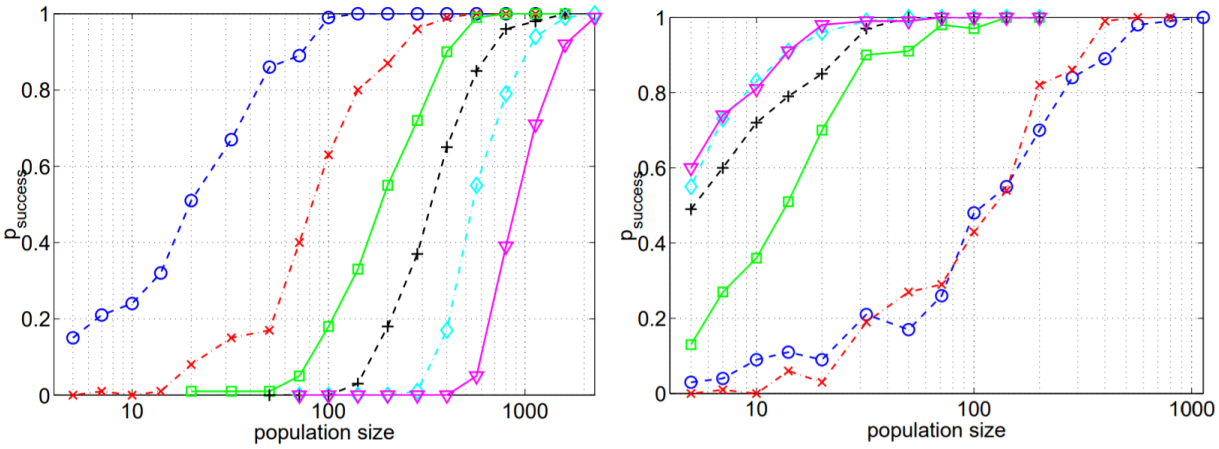
Transformacja $(\mathbf{C}^{(t)})^{-\frac{1}{2}}\mathbf{\Delta}^{(t)}$ dokonuje przeskalowania wektorów punktów środkowych do układu współrzędnych opisanych w $\mathbf{B}^{(t)}$. Czynnik $(\mathbf{C}^{(t)})^{-\frac{1}{2}}$ sprawia, że oczekiwana długość wektora izotropowej ścieżki ewolucji jest niezależna od jego kierunku. Można wykazać, że dla dowolnego ciągu macierzy kowariancji $\mathbf{C}^{(1,\ldots,t)}$, przy założeniu selekcji nieuwzględniającej wartości funkcji celu, $\mathbf{p}_\sigma^{(t+1)} \sim \mathcal{N}(0, \mathbf{I})$ [15]. Wynika z tego, że oczekiwana norma wektora $\mathbf{p}_\sigma^{(t+1)}$ równa jest oczekiwanej normie wektora wygenerowanego zgodnie z rozkładem normalnym standaryzowanym, o jednostkowej macierzy kowariancji. Ostateczny wzór na aktualizację współczynnika kroku $\sigma^{(t+1)}$ (Rysunek 2.2 linie 15) wykorzystuję tą własność, obliczając wartość ilorazu $\frac{||\mathbf{p}_\sigma^{(t+1)}||}{E||\mathcal{N}(\mathbf{0},\mathbf{I})||}$ i dokonując wykładniczego zwiększenia poprzedniej wartości kroku, gdy dłuższe wektory przeżywają selekcję, oraz wykładniczej redukcji w przeciwnym wypadku. Metoda kontrolowania czynnika kroku w algorytmie CMA-ES opiera się na kilku podstawowych obserwacjach:

1. Gdy ścieżka ewolucji jest krótka, świadczy to o tym , że kolejne generacje fluktuują wokół pewnego punktu w przestrzeni problemu. Innymi słowy następujące po sobie wektory $\mathbf{\Delta}^{(t)}$ populacji posiadają ujemnie skorelowane kierunki. Może to świadczyć o przebywaniu w pobliżu punktu lokalnie optymalnego, bądź o zbyt dużym rozproszeniu populacji. W takim wypadku czynnik $\sigma^{(t)}$ powinien być zmniejszony.

2. Ścieżka ewolucji jest długa, gdy kolejne wektory $\mathbf{\Delta}^{(t)}$ są ze sobą dodatnio skorelowane. W takim przypadku zwiększenie wartości $\sigma^{(t)}$ spowoduje zastąpienie kilku mniejszych kroków, w tym samym kierunku jednym, większym, co znacząco przyspieszy proces dochodzenia do rozwiązania optymalnego.

3. Gdy długość ścieżki ewolucji jest porównywalna do średniej długości wektorów $\mathbf{\Delta}^{(t)}$ kolejnych generacji, następują losowe zmiany kierunku ewolucji. W takim przypadku wartość kroku $\sigma^{(t)}$ powinna pozostać niezmieniana.

Wielkość populacji



(a) (b)

Rysunek 2.3: Wykres prawdopodobieństwa osiągnięcia poziomu $f_{stop} = 10^{-10}$ przy różnych wielkościach parametru λ i (a) funkcji Rastrigina oraz (b) funkcji Griewanka, dla wymiarowości problemu $n = 2(- - \mathrm{O} - -)$, $n = 5(- - \mathrm{x} - -)$, $n = 10(\text{—}\square\text{—})$, $n = 20(- - + - -)$, $n = 40(- - \Diamond - -)$, $n = 80(\text{—}\triangledown\text{—})$.

Źródło: [23]

Strategia ewolucyjna CMA-ES posiada szereg wewnętrznych parametrów, takich jak wielkość populacji λ, liczność populacji po selekcji μ, zbiór wag $w_{i=1,\ldots,\mu}$, parametry ścieżek ewolucji c_σ i c_c, adaptacji macierzy kowariancji c_1, c_μ, c_{cov} oraz czynnika kroku d_σ. Każda z tych wartości ma bezpośredni wpływ na różne elementy algorytmu CMA-ES. Poprawny ich dobór gwarantuje działanie metody zgodnie z jej założeniami oraz oddziałuje na tempo zbieżności w kierunku rozwiązania optymalnego. Domyślne wartości wszystkich parametrów zostały opisane w [16, Dodatek A]. Wielkość populacji została zaproponowana na

poziomie $\lambda = 4 + 3ln(n)$.

Dobór zaproponowanej wielkości populacji jest konsekwencją pierwotnych założeń na temat przyszłych zastosowań metody. Oryginalnie CMA-ES miał służyć optymalizacji lokalnej[25] - miał być algorytmem, który będzie się efektywnie zachowywał na funkcjach unimodalnych, w szczególności źle uwarunkowanych (o wysokim wskaźniku uwarunkowania hesjanu). Metoda ta sprawdza się jednakże również w rozwiązywaniu zadań optymalizacji globalnej, osiągając dobre rezultaty w poszukiwaniu optimum funkcji wielomodalnych i zaszumionych[23].

Użycie metody CMA-ES do zadań optymalizacji globalnej może wymagać innego niż domyślny doboru wielkości populacji. Rysunek 2.3 przedstawia wyniki badania, którego celem było zobrazowanie zależności prawdopodobieństwa znalezienia optimum $\mathrm{P}_{success}$, dla dwóch różnych funkcji celu (Rastrigina i Griewanka), od wielkości populacji. Wykresy te zostały wygenerowane z użyciem CMA-ES dla wymiarowości $n = [2, 5, 10, 20, 40, 80]$ oraz domyślnych parametrów, poza wielkością populacji. Dla każdej wymiarowości i liczności populacji ze zbioru $\lambda = [5, 7, 10, 14, 20, 32, \lfloor 32\sqrt{2} \rfloor, \lfloor \lfloor 32\sqrt{2} \rfloor \sqrt{2} \rfloor, \ldots, 1000]$, bazowa metoda CMA-ES została uruchomiona 100 razy. Optymalizacja trwała do osiągnięcia wartości funkcji celu $f_{stop} = 10^{-10}$ lub wyczerpania budżetu 10^7 ewaluacji funkcji celu. Wartość prawdopodobieństwa sukcesu dla zadanej wielkości populacji równa jest liczbie uruchomień które osiągnęły poziom f_{stop}, w stosunku do wszystkich uruchomień (liczby 100).

Wyniki przeprowadzonych eksperymentów pokazują, że prawdopodobieństwo sukcesu $\mathrm{P}_{success}$ dla funkcji wielomodalnych silnie zależy od wielkości populacji. Dla funkcji Rastrigina, na rysunku 2.3(a), widoczny jest sigmoidalny kształt zależności dla wszystkich wymiarowości. Większe wymiarowości potrzebują większej populacji, aby osiągnąć poziom prawdopodobieństwa $\mathrm{P}_{success} = 1$. Co więcej prawdopodobieństwo znalezienia rozwiązania optymalnego jest niskie dla wszystkich wymiarowości i równe jest zero dla najmniejszych wartości λ. Rysunek 2.3(b) dla funkcji Griewanka ukazuje odmienne charakterystyki sugerujące, że niższe wymiarowości potrzebują większych populacji oraz że osiągnięcie poziomu $\mathrm{P}_{success} = 1$ jest możliwe i stosunkowo łatwe (wymaga niskich wartości λ).

Uzyskane wyniki potwierdzają pierwotne założenia metody CMA-ES. Proces optymalizacji funkcji Griewanka przypomina optymalizację unimodalną, w której zależność parametru λ wymaganego do $\mathrm{P}_{success} = 1$ rośnie logarytmicznie wraz ze wzrostem n, do

około $\lambda = 20$. Domyślna wielkość populacji $\lambda = 4 + 3ln(n)$ realizuje te założenia.

Nakład obliczeniowy

Empiryczny pomiar, dla różnych funkcji celu i klasycznego kształtu algorytmu CMA-ES bez modyfikacji, pokazuje kwadratowy lub sub-kwadratowy nakład obliczeniowy dla dużej części testowanego zbioru funkcji [24] [27]. Metoda CMA-ES posiada kilka czynników wpływających na taki poziom nakładu obliczeniowego.

Macierz $\mathbf{C}^{(t)}$ posiada $\frac{n^2+n}{2}$ parametrów, które muszą zostać zaktualizowane w celu wyznaczenia macierzy $\mathbf{C}^{(t+1)}$. Potrzeba więc co najmniej $\frac{n^2+n}{2}$ obliczeń w każdej generacji, tylko w celu wyznaczenia nowej macierzy kowariancji.

Kolejnym problemem jest generowanie punktów z wielowymiarowym rozkładem normalnym o zadanej macierzy kowariancji, które posiada nakład obliczeniowy $\mathcal{O}(n^2)$. Zgodnie z wzorem (2.4) każde generowanie punktu wymaga konieczności przeprowadzenia mnożenia macierzy ortogonalnej $\mathbf{B}$, macierzy diagonalnej $\mathbf{D}$ oraz n-wymiarowego wektora wynikowego $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Taka operacja posiada nakład obliczeniowy co najmniej $\mathcal{O}(n^2)$.

Sposób estymacji macierzy kowariancji wprowadzony w algorytmie CMA-ES (opisany w rozdziale 2.1.2) wykorzystuje macierze $\mathbf{C}_{\mu}^{(t)}$ oraz $\mathbf{C}_{1}^{(t)}$. Zgodnie z wzorem (2.8) obliczenie bieżącej macierzy $\mathbf{C}_{\mu}^{(t)}$ wymaga n^2 operacji dla każdego z μ punktów bazowych. Ze wzoru (2.9) wynika, że do ustalenia nowej macierzy $\mathbf{C}_{1}^{(t)}$ niezbędne jest przeprowadzenie n^2 operacji z wektorem anizotropowej ścieżki ewolucji $\mathbf{p}_{c}^{(t)}$. Estymacja macierzy kowariancji realizowana w algorytmie CMA-ES posiada więc sumaryczny nakład obliczeniowy $\mathcal{O}((\mu + 1)n^2)$.

Spektralna dekompozycja macierzy kowariancji $\mathbf{C}$, opisana w rozdziale 2.1.1, posiada nakład obliczeniowy rzędu $\mathcal{O}(n^3)$. Faktoryzacja ta potrzebna jest w procesie generowania punktów z wielowymiarowym rozkładem normalnym o zadanej macierzy kowariancji. Znalezione z użyciem tej procedury macierze są również wykorzystywane do aktualizacji izotropowej ścieżki ewolucji $\mathbf{p}_{\sigma}^{(t+1)}$ (Rysunek 2.2, linia 13) i parametru kroku. Zgodnie ze wzorem (2.16) macierze $\mathbf{B}$ oraz $\mathbf{D}$ wykorzystywane są do wyznaczenia odwrotności pierwiastka macierzy kowariancji $\mathbf{C}^{-\frac{1}{2}}$.

Istnieją rozwiązania umożliwiające zmniejszenie liczby koniecznych obliczeń w każdym kroku metody CMA-ES, tym samym obniżając nieco nakład obliczeniowy, które zostaną zaprezentowane w kolejnym podrozdziale. Ostatecznie nakład obliczeniowy poje-

dynczej iteracji podstawowego algorytmu CMA-ES wynosi jednakże $\mathcal{O}(n^3)$. Poziom ten jest wynikiem konieczności przeprowadzania w każdej generacji kosztownej obliczeniowo faktoryzacji macierzy kowariancji.

2.1.3 Linie rozwojowe algorytmu CMA-ES

Bazowa wersja algorytmu CMA-ES zmaga się z wieloma problemami uniemożliwiającymi jej szerokie stosowanie. Z uwagi na nakład obliczeniowy pojedynczej iteracji wynoszący $\mathcal{O}(n^3)$, użytkowa wymiarowość optymalizowanego problemu posiada górne ograniczenie w okolicy $n = 100$. Domyślna wielkość populacji jest zestrojona do problemów optymalizacji lokalnej. Problemy optymalizacji globalnej wymagają ustawienia znacznie liczniejszych populacji, co obniża tempo zbieżności poprzez zwiększenie wymaganej liczby ewaluacji funkcji celu w jednej iteracji algorytmu.

Redukcja zapotrzebowania na zasoby

Duży nakład obliczeniowy metody, wynikający głównie z konieczności przeprowadzania faktoryzacji macierzy kowariancji, uniemożliwia skuteczne stosowanie algorytmu dla dużych wymiarowości. Nakład obliczeniowy procesu faktoryzacji macierzy kowariancji z użyciem dekompozycji spektralnej wynosi $\mathcal{O}(n^3)$. Wartość tą można jednak zredukować poprzez faktoryzację macierzy kowariancji $\mathbf{C}$ nie w każdej, lecz np. co dziesiątą generację[27]. Mechanizm ten szczegółowo opisany w [49][Rozdział 1, przypis 3], posiada niewielki wpływ na ogólną jakość optymalizacji. Pomimo tej redukcji, operacje macierzowe używane w algorytmie CMA-ES do generowania wektorów bazowych czy adaptacji kroku σ, są nadal bardzo kosztowne. Istnieje wiele prac naukowych, w których proponowane są rozwiązania zmieniające lub modyfikujące procesy przekształceń macierzowych, w celu dalszego zmniejszenia ogólnego nakładu obliczeniowego algorytmu.

Jednym ze sposobów realizacji uproszczenia procesu estymacji macierzy kowariancji jest przedstawienie jej w postaci sumacyjnej, składającej się z macierzy jednostkowej oraz pewnego modyfikowalnego produktu bieżącej populacji. Poland oraz Zell proponują metodę MVA-ES[47], w której zamiast dokonywać adaptacji, a następnie faktoryzacji, macierzy $\mathbf{C}$, adaptowany jest jeden wektor. W tym celu tworzony jest wektor $\mathbf{v}^{(t+1)}$(nazywany

głównym), zgodnie z zaproponowanymi zależnościami:

$$\begin{aligned}
\mathbf{p}_c^{(t+1)} &= (1-c_m)\mathbf{p}_c^{(t)} + c_m(\mathcal{M} + \mathcal{M}_1 w_v \mathbf{v}^{(t)}) \\
\mathbf{v}^{(t+1)} &= (1-c_v) sign((\mathbf{v}^{(t)})^T, \mathbf{p}_c^{(t)})\mathbf{v}^{(t)} + c_v \mathbf{p}_c^{(t+1)} \\
\mathcal{M} &\leftarrow \mathcal{N}(0, \mathbf{I}), \qquad \mathcal{M}_1 \leftarrow \mathcal{N}(0,1)
\end{aligned} \tag{2.17}$$

Parametry c_v, c_m, w_m, w_v to pewne stałe, zaś $sign(x)$ funkcja zwracającą znak x. Punkty potomne generowane są na podstawie wzoru:

$$\mathbf{x}_i^{(t+1)} \leftarrow \mathbf{x}_i^{(t)} + \sigma(\mathcal{M} + \mathcal{M}_1 w_m \mathbf{v}^{(t)})$$

Taki zapis procedury generacyjnej definiuje macierz kowariancji rozkładu punktów jako sumę macierzy jednostkowej oraz iloczynu diadycznego skumulowanego przesunięcia punktu środkowego. Z uwagi na to, że algorytm opiera się na przetwarzaniu wektorów, a nie macierzy, nakład obliczeniowy został zredukowany do poziomu liniowego $\mathcal{O}(n)$. Metoda MVA-ES uzyskuje podobne wyniki do CMA-ES jednakże jedynie w przypadku, gdy istnieje jeden, preferowany kierunek, w którym następować będzie adaptacja parametrów rozkładu. MVA-ES sprawdzi się więc tylko w optymalizacji funkcji unimodalnych, o wyraźnym kierunku najszybszych spadków wartości celu. Dla funkcji wielomodalnych, wyniki zaproponowanego rozwiązania są znacząco gorsze od rezultatów osiąganych z użyciem bazowego algorytmu CMA-ES.

Podobny do MVA-ES sposób upraszczania wzoru na macierz kowariancji prezentowany jest w algorytmie R1-NES[55]. Autorzy metody proponują przyjęcie parametru generowane punktów z użyciem wzorów:

$$\begin{aligned}
\mathbf{x}_i^{(t+1)} &\leftarrow e^{\kappa}(\mathcal{M} + \mathcal{M}_1 \mathbf{u}^{(t)}) \\
\kappa \in \mathbb{R}: &\quad \kappa \leftarrow \kappa + \eta \nabla_\kappa J
\end{aligned}$$

Definicja $\mathcal{M}$ oraz $\mathcal{M}_1$ jest zgodna z (2.17). Wektor $\mathbf{u}^{(t)}$ podlega adaptacji na podstawie:

$$\mathbf{u}^{(t+1)} \leftarrow \mathbf{u}^{(t)} + \eta \nabla_u J$$

Współczynnik η jest współczynnikiem długości kolejnych kroków w stronę wyznaczoną z użyciem gradientu. Procedura generacyjna metody R1-NES definiuje macierz kowariancji rozkładu jako sumę macierzy jednostkowej oraz iloczynu diadycznego dominującego wektora własnego macierzy, przesuniętego w kierunku zgodnym z gradientem funkcji celu.

Estymacja gradientu posiada liniowy nakład obliczeniowy $\mathcal{O}(n)$. Parametryzacja macierzy kowariancji $\mathbf{C}$ w R1-NES umożliwia tylko jednemu kierunkowi w każdej generacji posiadać duże wartości własne. Rzeczywisty kształt funkcji nie może więc być efektywnie aproksymowany z użyciem zaproponowanej procedury. Rezultaty przeprowadzonych badań pokazują, że metodę R1-NES można skutecznie używać tylko dla konkretnych kształtów funkcji celu.

Innym sposobem redukcji nakładu obliczeniowego jest wyłączenie konieczności przeprowadzania faktoryzacji macierzy kowariancji poprzez pracę na estymacjach sfaktoryzowanych macierzy. Metoda MA-ES[6] proponuje rozwiązanie, w którym generowanie punktów nie wymaga adaptacji izotropowej ścieżki $\mathbf{p}_\sigma$, ani macierzy $\mathbf{C}$. Rozkład generowanych punktów w kolejnych populacjach jest jednocześnie zbliżony do realizowanego w bazowym algorytmie CMA-ES (przy założeniu równej parametryzacji obu ścieżek ewolucji $c_\sigma = c_c$). CMA-ES generuje punkty zgodnie wielowymiarowym rozkładem normalnym $\mathcal{N}(\mathbf{m}, \mathbf{C})$ o zadanej pełnej macierzy kowariancji i wartości oczekiwanej. W praktyce, zgodnie ze wzorem (2.1), jest to realizowane poprzez wygenerowanie punktu z użyciem rozkładu normalnego o jednostkowej macierzy kowariancji, a następnie transformowanie go do wektora bazowego $\mathbf{d}_i^{(t)}$ poprzez wymnożenie przez macierz $\mathbf{M}^{(t)} = (\mathbf{C}^{(t)})^{\frac{1}{2}}$ i dodanie wartości średniej $\mathbf{m}^{(t)}$. Do poprawnego działania nie jest więc niezbędna znajomość macierzy kowariancji $\mathbf{C}^{(t)}$, wystarczające jest posiadanie w kolejnych generacjach informacji na temat jej pierwiastka $(\mathbf{C}^{(t)})^{\frac{1}{2}}$. Autorzy metody MA-ES proponują, aby zamiast obliczać $(\mathbf{C}^{(t)})^{\frac{1}{2}}$ z użyciem rozkładu Cholesky'ego lub dekompozycji spektralnej, dokonywać jej przybliżenia poprzez macierz $\mathbf{M}^{(t)}$. Jest ona opisana w postaci sumy odchyleń macierzy $\mathbf{C}_1$ oraz $\mathbf{C}_\mu$ od macierzy jednostkowej $\mathbf{I}$.

$$\mathbf{M}^{(t+1)} \leftarrow \mathbf{M}^{(t)}[\mathbf{I} + \frac{c_1}{2}[\mathbf{C}_1 - \mathbf{I}] + \frac{c_\mu}{2}[\mathbf{C}_\mu - \mathbf{I}]]$$

Z uwagi na konieczność mnożenia macierzy oraz szereg dodatkowych operacji, nakład obliczeniowy jednej iteracji zaproponowanej metody wynosi $\mathcal{O}(n^3)$. Bazując na teoretycznej analizie, MA-ES pokazuje możliwość całkowitego wyłączenia konieczności stosowania kosztownej faktoryzacji, wprowadzając inny mechanizm, bez znaczących strat dla skuteczności działania metody. Po zastosowaniu metody mnożenia macierzy Coppersmitha-Winograda można uzyskać nakład obliczeniowy algorytmu MA-ES na poziomie $\mathcal{O}(n^{2.37})$. Metoda ta dokonuje, na gruncie teoretycznej analizy, przeformułowania wzoru na generację punktów w algorytmie CMA-ES. Kosztowny proces faktoryzacji macierzy $\mathbf{C}$ został

wyłączony i zastąpiony mechanizmem mnożenia macierzy. Modyfikacja ta nie wpływa jednakże na obniżenie ostatecznego poziomu nakładu obliczeniowego, tylko na zmianę najbardziej istotnego czynnika.

Metoda LM-MA-ES[40] proponuje rozwiązanie, które jest modyfikacją metody MA-ES ukierunkowaną w celu dalszego obniżenia jej nakładu obliczeniowego. Proces generowania punktów jest zmieniany, wyłączając konieczność używania macierzy $\mathbf{M}^{(t)}$. W każdej generacji tworzona jest nowa populacja wektorów bazowych $\mathbf{d}_i^{(t+1)}$ na postawie zmian wprowadzonych w odpowiadających wektorach bazowych $\mathbf{d}_i^{(t+)}$ poprzedniej generacji. W celu wyznaczenia potomka wektora $\mathbf{d}_i^{(t)}$, przeprowadzanych jest iteracyjnie λ modyfikacji poprzedniej wartości, mającej na celu poprawne zamodelowanie odchylenia macierzy kowariancji od macierzy jednostkowej.

$$\begin{aligned} &\textbf{for } j \leftarrow 1, \ldots, min(t, \lambda) \textbf{ do} \\ &\quad \mathbf{d}_i^{(t+1)} \leftarrow (1 - c_{d,j})\mathbf{d}_i^{(t)} + c_{d,j}\mathbf{o}_j^{(t)} \left((\mathbf{o}_j^{(t)})^T \mathbf{d}_i^{(t)} \right) \end{aligned} \tag{2.18}$$

Wartość $c_{d,j}$ opisana jest wzorem, zależnym od numeru iteracji j. Wektor $\mathbf{o}_j^{(t)}$ jest definiowany w taki sposób, aby jego kolejne wartości były analogiczne do wektora $\mathbf{p}_\sigma^{(t)}$ w oryginalnym algorytmie CMA-ES. Realizowane jest to w analogiczny do (2.18) sposób:

$$\begin{aligned} &\textbf{for } i \leftarrow 1, \ldots, \lambda \textbf{ do} \\ &\quad \mathbf{o}_i^{(t+1)} \leftarrow (1 - c_{c,i})\mathbf{o}_i^{(t)} + \sqrt{\mu_w(2 - c_{c,i})} \sum_{j=1}^{\mu} w_j \mathbf{z}_j^{(t)} \end{aligned}$$

Wektory $\mathbf{z}_j^{(t)}$ są realizacjami zmiennej losowej $\mathcal{M}$ (2.17). Parametry $c_{d,j}$ oraz $c_{c,i}$ realizują wykładnicze zanikanie wraz ze wzrostem numeru iteracji. Algorytm LM-MA-ES dokonuje modelowania macierzy $\mathbf{M}^{(t)}$, będącej sumą odchyleń macierzy $\mathbf{C}_1$ oraz $\mathbf{C}_\mu$ od macierzy jednostkowej $\mathbf{I}$. Proces iteracyjnego przybliżania dotyczy wyniku mnożenia $\mathbf{M}^{(t)}$ przez realizacje zmiennej losowej $\mathcal{M}$. Zamiast obliczać macierz $\mathbf{M}^{(t)}$, by następnie poprzez mnożenie z $\mathbf{z}_i^{(t)}$ uzyskać wektory bazowe $\mathbf{d}_i^{(t)}$, metoda LM-MA-ES dokonuje bezpośredniego przybliżenia wyniku tego mnożenia. Z uwagi na wyeliminowanie konieczności mnożenia macierzy w procedurze generacji nowej populacji, ostateczny poziom nakładu obliczeniowego metody wynosi $\mathcal{O}(nlog(n))$. Z uwagi na to, że LM-MA-ES dokonuje przybliżenia wektorów bazowych generowanych przez algorytm MA-ES, finalny kształt macierzy kowariancji generowanych punktów może się różnić od pierwotnych założeń algorytmu CMA-ES. Zaproponowany algorytm nie może być więc rozpatrywany jako rozwiązanie problemu zbyt dużego nakładu obliczeniowego metody CMA-ES dla większych wymiarowości.

Algorytmem dokonującym bardziej radykalnego niż LM-MA-ES uproszczenia pierwotnych założeń CMA-ES, w celu redukcji nakładu obliczeniowego, jest metoda SEP-CMA-ES[49]. Idea metody zakłada narzucenie na macierz kowariancji $\mathbf{C}^{(t)}$ konieczności diagonalności. Adaptacja macierzy kowariancji sprowadza się wówczas do adaptacji odchyleń standardowych w każdym kierunku. Proces generowania punktów z rozkładem $\mathcal{N}(\mathbf{0}, \mathbf{C}^{(t)})$ opiera się w takim przypadku na n niezależnych wariancjach. Gdy macierz $\mathbf{C}^{(t)}$ jest diagonalna, macierz $\mathbf{B} = \mathbf{I}$ dla wszystkich generacji, a więc jest pomijalna. Macierz pierwiastków wartości własnych jest wówczas pierwiastkiem macierzy kowariancji.

$$\mathbf{D} = (\mathbf{C}^{(t)})^{\frac{1}{2}} \tag{2.19}$$

Z uwagi na redukcję stopni swobody macierzy $\mathbf{C}^{(t)}$ (parametrów wymagających aktualizacji) z $\frac{n^2+n}{2}$ do n, jedna iteracja metody posiada liniowy poziom nakładu obliczeniowego $\mathcal{O}(n)$. Algorytm SEP-CMA-ES nie jest odporny na obroty funkcji celu. Wektory własne macierzy kowariancji są równoległe do poszczególnych osi układu współrzędnych. Metoda ta nie dokonuje przybliżenia macierzy kowariancji, lecz jej znacznego uproszczenia. Z równania (2.19) wynika, że $\mathbf{C}^{(t)} = \mathbf{D}^2$, a więc metoda generuje populacje zgodnie z rozkładem $\mathcal{N}(\mathbf{0}, \mathbf{D}^2)$. Zgodnie z rysunkiem 2.1 funkcja gęstości generowanego rozkładu posiada tylko częściową zdolność adaptacji do poziomic funkcji celu. Algorytm SEP-CMA-ES nie może być więc rozpatrywany jako realny krok w kierunku zmniejszenia ogólnego poziomu nakładu obliczeniowego bazowej metody CMA-ES.

Poprawa szybkości zbieżności

Metoda CMA-ES posiada domyślną wielkość populacji, która maksymalizuje tempo zbieżności dla funkcji unimodalnych, w szczególności o źle uwarunkowanym hesjanie. W przypadku problemów wielomodalnych, zwiększanie liczności populacji wpływa na zwiększenie prawdopodobieństwa osiągnięcia optimum (rozdział 2.1.2). Duże populacje sprawiają, że algorytm CMA-ES staje się więc bardziej dostosowany do optymalizacji globalnej. Obarczone jest to kosztem wynikającym ze znaczącego wzrostu liczby ewaluacji funkcji celu w jednej iteracji, powodując ostateczny spadek szybkości zbieżności procesu optymalizacji, w stosunku do liczy wywołań funkcji celu. Istnieje wiele propozycji zarządzania wielkością populacji, by zachowując skuteczność na problemach wielomodalnych, zwiększać szybkość zbieżności procesu optymalizacji.

Autorzy metody IPOP-CMA-ES[3] proponują rozwiązanie wprowadzające mechanizm restartu algorytmu połączony ze zwiększaniem wielkości populacji. Proces ewolucji w tak zaproponowanej metodzie przebiega wieloetapowo, w następujących po sobie sekwencjach wywołań bazowej metody CMA-ES. Każda z kolejnych realizacji posiada zwiększoną populację w stosunku do poprzednika.

$$\begin{aligned} \lambda^{(0)} &\leftarrow 4 + 3ln(\mathrm{n}) \\ \lambda^{(i)} &\leftarrow K\lambda^{(i-1)} \end{aligned} \tag{2.20}$$

Wartość mnożnika K jest parametrem metody, która powinna zawierać się w przedziale $K \in < 1.5, 5 >$. Badania na funkcji Rastrigina pokazują, że metoda osiąga podobnie wysoką skuteczność dla mnożnika z przedziału od 2 do 3. Wynik ten jest powodem sugerowanej przez autorów wartości $K = 2$, w zastosowaniach optymalizacji globalnej metody IPOP-CMA-ES. Proces ewolucji każdego elementu sekwencji trwa aż do osiągnięcia jednego z kilku kryteriów stopu:

1. Restart następuje gdy odchylenie standardowe rozkładu normalnego jest mniejsze od $TolX$ we wszystkich kierunkach oraz wszystkie składowe wektora $\mathbf{p}_c^{(t)}$ są mniejsze od $TolX$. Domyślna wartość wynosi $TolX = 10^{-12}$ [16, Dodatek B.3].

2. Restart następuje gdy najlepszy punkt nie zmienił się przez $10 + \frac{30n}{\lambda}$ generacji, lub zmienił się w stopniu mniejszym niż $TolFun = 10^{-12}$ [16, Dodatek B.3].

3. Restart następuje gdy dodanie wartości odchylenia standardowego $0.1\sigma^{(t)}$ we wiodącym kierunku $\mathbf{C}$, nie zmieni punktu środkowego $\mathbf{\Delta}^{(t)}$.

4. Restart następuje gdy dodanie wartości odchylenia standardowego $0.2\sigma^{(t)}$ w każdym kierunku, nie zmieni punktu środkowego $\mathbf{\Delta}^{(t)}$.

5. Restart następuje gdy wartość uwarunkowania macierzy kowariancji $\mathbf{C}^{(t)}$ przekroczy 10^{14}.

Testy zaproponowanej metody dowodzą, że IPOP-CMA-ES przewyższa skutecznością bazową metodę CMA-ES dla części problemów wielomodalnych, w tym funkcji Rastrigina. Jednocześnie widoczny jest brak istotnych zmian w tempie zbieżności oraz prawdopodobieństwie sukcesu dla funkcji unimodalnych. IPOP-CMA-ES może być traktowany jako uniwersalnie sparametryzowany algorytm CMA-ES dla nieliniowych, niewypukłych i zaszumionych problemów optymalizacji globalnej i lokalnej. Metoda ta nie poprawia jednak

szybkości zbieżności bazowego CMA-ES, wyłącza jedynie konieczność ręcznej zmiany wielkości populacji, w zależności od typu optymalizowanego problemu.

Algorytm BI-POP-CMA-ES[19] wprowadza rozwiązanie mające ograniczać wielkość populacji, zachowując jednocześnie zdolność do jej wzrostu. W tym celu wprowadzone są dwa sposoby zarządzania populacją po restarcie. Pierwszy z nich jest strategią zwiększania populacji, zaproponowaną w metodzie IPOP-CMA-ES. Wielkość populacji w tym podejściu oznaczana jest jako $\lambda_{IP}^{(i)}$ i opisana jest zgodnie z wzorem (2.20), posiadając górne ograniczenie $\lambda_{IP}^{(i)} \leqslant 512\lambda^{(0)}$. Druga strategia stosuje ideę małych populacji zgodnie ze wzorem:

$$\lambda_S^{(i)} \leftarrow \left\lfloor \lambda^{(0)} \left(\frac{1}{2} \frac{\lambda_{IP}^{(a)}}{\lambda^{(0)}} \right)^{U[0,1]^2} \right\rfloor,$$

gdzie $\lambda_{IP}^{(a)}$ oznacza aktualną liczność populacji w strategii zwiększania populacji, a $U[0,1]$ jest realizacją zmiennej losowej o rozkładzie jednostajnym w przedziale $[0,1]$. W efekcie wielkość populacji w tym sposobie jej zarządzania jest zmienną losową mieszczącą się w przedziale:

$$\lambda_S^{(i)} \in \left[\lambda^{(0)}, \frac{\lambda_{IP}^{(a)}}{2} \right].$$

Działanie metody BI-POP-CMA-ES polega na początkowym przydzieleniu połowy dostępnego budżetu każdej ze strategii, a następnie wybieraniu takiej, która posiada większy budżet w danej chwili. Decyzja podejmowana jest po każdym restarcie. W przypadku jednakowego budżetu dla obu strategii wybierane jest podejście zwiększania populacji $\lambda_{IP}^{(i)}$. Kryteria stopu procesu ewolucji są takie same jak w metodzie IPOP-CMA-ES, z dodatkowym sposobem terminacji z uwagi na liczbę iteracji. Maksymalna liczba generacji każdego uruchomienia strategii zależy od liczności populacji aktualnej strategii (zmiennej λ) i wynosi $100+50\frac{n+3}{\sqrt{\lambda}}$. Ograniczenie to zostało wprowadzone by zapobiec długotrwałej eksploracji o dużej liczbie punktów. Metoda BI-POP-CMA-ES na testowanych przez jej autorów funkcjach unimodalnych zachowuje się podobnie jak metoda IPOP-CMA-ES. Uruchamiana jest jedna iteracja strategii zwiększania populacji, która już z wyjściową wielkością populacji $\lambda_{IP}^{(0)} = \lambda^{(0)}$ jest w stanie znaleźć rozwiązanie optymalne. Działanie BI-POP-CMA-ES dla takich funkcji jest więc bliskie bazowej metodzie CMA-ES. W przypadku optymalizacji na funkcjach wielomodalnych, algorytm BI-POP-CMA-ES naprzemiennie uruchamia obie strategie, robiąc jeden krok ewolucyjny z użyciem rosnących populacji, a

następnie kilka kroków z użyciem małych. Sekwencja ewolucji dla strategii zwiększania liczby punktów realizuje proces eksploracji przestrzeni problemu, podczas gdy podejście małych populacji dokonuje eksploatacji aktualnych obszarów przyciągania ekstremów lokalnych. Ta idea pozwala wydłużyć czas trwania całego procesu ewolucji, w stosunku do metody IPOP-CMA-ES o takim samym budżecie. Sprawia to, że BI-POP-CMA-ES dokonuje poprawy szybkości zbieżności w stosunku do algorytmu IPOP-CMA-ES i w efekcie do bazowego CMA-ES.

Algorytmy IPOP-CMA-ES oraz BI-POP-CMA-ES posiadają rozszerzenia strategii, które mogą być wdrażane podczas osiągnięcia kryterium stopu. Modyfikacje NIPOP oraz NBIPOP[41] dotyczą sposobu zarządzania adaptacją długości kroku $\sigma^{(t)}$ oraz alternatywnym sposobem zarządzania wielkością populacji. Pierwsza z nich jest zmianą algorytmu IPOP-CMA-ES polegającą na zmniejszeniu wyjściowej długości kroku $\sigma^{(0)}$, w momencie zwiększenia liczności populacji zgodnie z wzorem (2.20). Metoda ta oznaczana jest jako NIPOP-CMA-ES i opiera się na zmniejszeniu długości kroku zgodnie z wzorem:

$$\sigma^{(0)} \leftarrow \frac{\sigma^{(0)}}{\rho_{\sigma dec}},$$

gdzie $\rho_{\sigma dec}$ jest czynnikiem zmniejszającym krok w kolejnej sekwencji ewolucji, o sugerowanej wartości $\rho_{\sigma dec} = 1.6$. W przypadku gdy wyjściowa długość kroku $\sigma^{(0)}$ jest mała, metoda CMA-ES jest w stanie dokonać eksploatacji mniejszych obszarów przyciągania lokalnych ekstremów, podczas gdy ustawienie większej wartości $\sigma^{(0)}$ może przyspieszyć optymalizację w kierunku lokalnego minimum. Początkowo relatywnie niska wartość kroku zostanie przez algorytm adaptacji szybko zwiększona, jeśli zajdzie taka konieczność (rozdział 2.1.2), przywracając domyślną zdolność CMA-ES do bardziej globalnej optymalizacji. Drugim rozszerzeniem strategii restartu jest metoda NBIPOP-CMA-ES. Modyfikacja ta dotyczy algorytmu BI-POP-CMA-ES, w którym strategię zwiększania wielkości populacji po restarcie zastąpiono przez NIPOP-CMA-ES. Strategia małych populacji λ_S została zamieniona strategią λ_U losowego przydziału stałej kroku o domyślnej liczności populacji, zdefiniowaną następująco:

$$\lambda_U^{(i)} \leftarrow \lambda^{(0)}$$
$$\sigma^{(0)} \leftarrow \sigma^{(0)} 10^{-2U[0,1]}$$

Działanie metody NBIPOP-POP-CMA-ES polega na adaptacyjnym przydzielaniu kwantów pozostałego budżetu strategiom, a następnie uruchomieniu obu strategii. Jednym z

kryteriów restartu jest zużycie przydzielonej liczby ewakuacji funkcji celu. Rozdział budżetu następuje po czasie aż oba podejścia osiągną kryterium stopu. Strategia, której udało się znaleźć najlepszy jak dotąd punkt dostaje dwa razy więcej budżetu niż druga. W przypadku gdy obie strategie znalazły takie samo rozwiązanie, lub gdy nie znalazły jeszcze żadnego, każdej z nich przydzielana jest jednakowa liczba dopuszczalnej ewaluacji funkcji celu. Parametrem metody jest domyślna wartość wywołań funkcji celu, po której ma nastąpić restart obu strategii. Liczba ta jest kwantem całkowitego budżetu przydzielonemu algorytmowi. Metoda NBIPOP-POP-CMA-ES realizuje, podobnie jak BI-POP-CMA-ES, sekwencję ewolucji mającą na celu przełączanie pomiędzy eksploracją a eksploatacją funkcji celu. NBIPOP-POP-CMA-ES implementuje jednakże proces rywalizacji pomiędzy strategią zwiększającą populację przy jednoczesnym zmniejszeniu wyjściowej stałej kroku, a bazowym algorytmem CMA-ES z losową początkową wartością kroku. Takie podejście pozwala adaptacyjne przydzielać większy budżet podejściu skuteczniejszemu w danej chwili. Sprawia to, że dokonywana jest poprawa szybkości zbieżności w stosunku do BI-POP-CMA-ES i w efekcie także do CMA-ES.

2.2 Ewolucja różnicowa

Ewolucja różnicowa[53] jest metodą która pierwotnie powstała jako rozwiązanie problemu aproksymacji wielomianami Czebyszewa[59, Rozdział 1.2], w kontekście poszukiwania współczynników układu wielomianów ortogonalnych najlepiej przybliżającego zadaną grupę punktów lub krzywą. Wielomiany te są istotną częścią teorii aproksymacji, znajdując zastosowanie w interpolacji Lagrange'a. Pierwowzorem ewolucji różnicowej był algorytm genetycznego symulowanego wyżarzania[35], opracowany przez jednego z współautorów opisywanej w tym rozdziale metody. Zaproponowana technika rozwiązywała problem aproksymacji wielomianowej, jednak z uwagi na niskie tempo zbieżności, niewysoką zdolność eksploracji przestrzeni problemu oraz skomplikowany proces strojenia parametrów, nie mogła być postrzegana jako praktyczne narzędzie optymalizacyjne. W ramach prac nad wyeliminowaniem wymienionych problemów, zastąpiono binarne kodowanie przez zmiennoprzecinkowe oraz zastosowano arytmetykę wektorową. Tak uzyskany ciągły optymalizator genetycznego wyżarzania był podstawą ewolucji różnicowej, w której mutacja opiera się na dodaniu ważonego wektora różnicy dwóch losowo wybranych punktów z po-

pulacji. Testy tak zmodyfikowanej metody wykazały, że mutacja różnicowa w połączeniu z jednorodnym krzyżowaniem oraz jednorodną selekcją par punktów, nie wymaga współczynnika wyżarzania do zachowania równowagi pomiędzy eksploracją a eksploatacją.

Algorytm ewolucji różnicowej (nazywany również DE) jest stochastyczną metaheurystyką populacyjną, która optymalizuje zadaną funkcję celu z użyciem wprowadzonego mechanizmu mutacji różnicowej. Metoda ta jest szeroko stosowana w szczególności dla problemów o dużej liczbie wymiarów, gdyż jest nieskomplikowana implementacyjnie, a do działania nie potrzebuje obliczania oraz estymacji gradientu optymalizowanej funkcji. Ewolucja różnicowa nie wymaga, aby funkcja celu q była różniczkowalna. Z uwagi na brak konieczności użycia gradientu, algorytm DE może by także używany do poszukiwania rozwiązań problemów, których funkcja celu jest nieciągła, zaszumiona lub zmieniająca się w czasie. Ewolucja różnicowa została zaprezentowana jako algorytm optymalizacji globalnej[53] i obecnie jest jedną z najszerzej stosowanych metaheurystyk, używanych do poszukiwań optimum globalnego. Skuteczności metody DE dowodzi szereg jej zastosowań w wielu dziedzinach nauki[46], takich jak inżynieria, medycyna, czy genetyka.

2.2.1 Opis algorytmu ewolucji różnicowej

Algorytmy z rodziny ewolucji różnicowej bazują na schemacie zaprezentowanym w przez autorów w pierwotnym artykule[53]. Rysunek 2.4 przedstawia zarys koncepcji, która będzie dalej nazywana wyjściową. Algorytm ten przetwarza populacje $X^{(t)}$, które posiadają λ punktów. W kolejnych generacjach, dla każdego punktu $\mathbf{x}_i^{(t)}$, generowany jest nowy kandydat $\mathbf{z}_i^{(t)}$. Potomek $\mathbf{x}_i^{(t+1)}$ wybierany jest na podstawie wyniku rywalizacji wartością funkcji celu, pomiędzy rozpatrywanym punktem a kandydatem. W taki sposób metoda tworzona jest populacja $X^{(t+1)}$.

Istnieje kilka wariantów schematu algorytmu ewolucji różnicowej, które zostały zaproponowane przez autorów bazowej metody. Oznaczane są one zgodnie z przyjętą notacją DE/R/k/C, gdzie R wyznacza sposób selekcji, k liczbę wektorów różnicowych, a C sposób krzyżowania.

Selekcja

Selekcja w algorytmie ewolucji różnicowej dokonywana jest na kilku poziomach jego działania. W początkowej fazie determinuje punkt $\mathbf{x}_{base,i}$, który stanie się bazowym w procesie

1: $t \leftarrow 1$

2: initialize($X^{(1)} = \{\mathbf{x}_1^{(1)}, ..., \mathbf{x}_\lambda^{(1)}\}$)

3: evaluate $(X^{(1)})$

4: **while** !stop **do**

5: **for** $i = 1$ **to** λ **do**

6: define the base vector $\mathbf{x}_{base,i}$

7: $\{r_1, ..., r_{2k}\} \sim U(1, ..., \lambda)$

8: $\mathbf{d}_i^{(t)} \leftarrow \sum_{j=1}^{k} \left(\mathbf{x}_{r_{2j-1}}^{(t)} - \mathbf{x}_{r_{2j}}^{(t)}\right)$

9: $\mathbf{y}_i^{(t)} \leftarrow \mathbf{x}_{base,i} + F \cdot \mathbf{d}_i^{(t)}$

10: $\mathbf{z}_i^{(t)} \leftarrow$ crossover $\left(\mathbf{y}_i^{(t)}, \mathbf{x}_i^{(t)}\right)$

11: evaluate $\left(\mathbf{z}_i^{(t)}\right)$

12: **if** $q\left(\mathbf{z}_i^{(t)}\right) \leqslant q\left(\mathbf{x}_i^{(t)}\right)$ **then**

13: $\mathbf{x}_i^{(t+1)} \leftarrow \mathbf{z}_i^{(t)}$

14: **else**

15: $\mathbf{x}_i^{(t+1)} \leftarrow \mathbf{x}_i^{(t)}$

16: **end if**

17: **end for**

18: $t \leftarrow t + 1$

19: **end while**

Rysunek 2.4: Zarys algorytmu ewolucji różnicowej

mutacji. Punkt ten wyznaczony jest niezależnie dla każdego i-tego elementu populacji $X^{(t)}$, stając się genetycznym rodzicem i-tego potomka kandydującego kolejnej generacji. Sposób wyznaczania punktu bazowego jest pierwszym parametrem schematu ewolucji różnicowej charakteryzującym reprodukcję. Istnieje kilka podstawowych wariantów selekcji:

DE/rand/k/C : W tej strategii każdy punkt bazowy jest realizacją zmiennej losowej o rozkładzie jednostajnym określoną na populacji: $\mathbf{x}_{base,i} = \mathbf{x}_{(\sim U(1,...,\lambda))}$. Losowy sposób doboru punktów bazowych zwiększa oczekiwany poziom różnorodności przyszłej populacji.

DE/best/k/C : Jest to podejście, w którym punkt bazowy jest najlepszym punktem bieżącej populacji: $\mathbf{x}_{base,i} = \widehat{\mathbf{x}}$. Nie ma konieczności niezależnego wyznaczania punktu bazowego każdego elementu populacji, gdyż jest on identyczny dla każdej genera-

cji. Stosowanie takiej techniki zwiększa nacisk na eksploatacje wokół najlepszego rozwiązania.

DE/rand-to-best/k/C : Jest to strategia łącząca podejście **DE/rand** wraz ze **DE/best**, w której punkt bazowy jest ważoną sumą punktu losowego oraz najlepszego w populacji: $\mathbf{x}_{base,i} = (1-\gamma)\mathbf{x}_{(\sim U(1,...,\lambda))} + \gamma\widehat{\mathbf{x}}$. Wartość $\gamma \in [0,1]$ pozwala kontrolować zachłanność strategi.

DE/current-to-best/k/C : W tej strategi wykorzystywany jest punkt bazowy wyznaczony z użyciem podejścia **DE/best** oraz aktualny, i-ty punkt populacji $X^{(t)}$ oznaczony jako $\mathbf{x}_i$. Strategia ta wymaga co najmniej dwóch wektorów różnicowych ($k \geqslant 2$). Pierwszy z wektorów jest różnicą punktu najlepszego oraz aktualnego, natomiast drugi oraz kolejne wyznaczane są zgodnie ze wzorem na $\mathbf{d}_i^{(t)}$ (Rysunek 2.4, linia 8): $\mathbf{x}_{base,i} = \mathbf{x}_i + F \cdot (\widehat{\mathbf{x}} - \mathbf{x}_i)$

DE/current-to-rand/k/C : Jest to strategia podobna do **DE/current-to-best**, w której do budowy pierwszego wektora różnicowego używa się punktu losowego, zamiast punktu najlepszego. $\mathbf{x}_{base,i} = \mathbf{x}_i + F \cdot (\mathbf{x}_{\mathbf{r_b}} - \mathbf{x}_i)$ gdzie $r_b \sim U(1,...,\lambda)$

Algorytm ewolucji różnicowej, tworząc punkt kandydujący, używa mechanizmu mutacji. W tym celu stosowany jest wybór punktów wchodzących w skład wektorów różnicowych. W wyjściowym algorytmie DE, każdy z tych punktów jest realizacją zmiennej losowej określonej na zbiorze punktów, w której: $\{r_1,...,r_{2k}\} \sim U(1,...,\lambda)$. Wartość k jest drugim parametrem schematu DE i określa liczbę wektorów różnicowych.

Ostateczna konstrukcja populacji dla kolejnej generacji odbywa z użyciem deterministycznej sukcesji. Punkt kandydujący wyznaczony poprzez proces reprodukcji, mutacji i krzyżowania, staje się potomnym tylko wtedy, gdy poziom jego jakości jest wyższy od rodzica. W przeciwnym wypadku punkt rodzicielski przeżywa do kolejnej generacji. Tak zdefiniowana sukcesja zapewnia nie pogarszanie się średniego poziomu jakości w następujących po sobie populacjach.

Mutacja różnicowa

Operator mutacji w algorytmie DE tworzy mutanta $\mathbf{y}_i^{(t)}$ poprzez dodanie do punktu bazowego $\mathbf{x}_{base,i}$ przeskalowanej sumy wektorów różnicowych. Zarówno liczba wektorów różni-

cowych, jak i sposób tworzenia punktu $\mathbf{x}_{base,i}$, są parametrami charakteryzującymi zadany schemat ewolucji różnicowej, które zostały opisane w rozdziale 2.2.1[Selekcja].

Wektory różnicowe są najbardziej charakterystycznym elementem metody DE, na którym bazuje operator mutacji. Sposób budowy pojedynczego wektora polega na odjęciu od siebie dwóch punktów, wylosowanych z rozkładem jednostajnym z populacji X. Tak powstały wektor, mogący w zależności od parametru k być sumą zadanej liczby wektorów, bierze udział w tworzeniu mutanta $y_i^{(t)}$, zgodnie z poniższymi wzorami:

$$\mathbf{d}_i^{(t)} \leftarrow \sum_{j=1}^{k} \left(\mathbf{x}_{r_{2j-1}}^{(t)} - \mathbf{x}_{r_{2j}}^{(t)} \right)$$

$$\mathbf{y}_i^{(t)} \leftarrow \mathbf{x}_{base,i} + F \cdot \mathbf{d}_i^{(t)}$$

Skuteczność koncepcji wektorów różnicowych opiera się na fakcie zależności pomiędzy rozkładem punktów populacji wokół jej środka, a rozkładem mutantów wokół punktu bazowego. Odległość pomiędzy punktami może być dobrym estymatorem różnorodności, rozumianej w kontekście liczby obszarów przyciągania minimów lokalnych, wokół których znajduje się populacja. Duże wektory różnicowe mogą sugerować fazę eksploracyjną algorytmu, w której punkty znajdują się w obrębie różnych ekstremów, zwiększając poziom różnorodności populacji. Małe odległości pomiędzy punktami mogą być natomiast wskaźnikiem eksploatacji jednego minimum lokalnego. Fakt ten może być wykorzystany do skuteczniejszego zarządzania procesem optymalizacji, np. poprzez zwiększanie kroku w fazie eksploatacji, a zmniejszanie go podczas eksploracji. Ewolucja różnicowa implementuje ten mechanizm poprzez operator mutacji, używający przeskalowanej (współczynnikiem F) sumy k wektorów różnicowych, utworzonych z losowo wyselekcjonowanych punktów. W taki sposób algorytm DE jest w stanie zastosować różny krok dla różnych kierunków, biorąc pod uwagę lokalną dynamikę funkcji celu.

Stosowanie wektorów różnicowych w procesie generacji punktów potomnych pozwala osiągnąć duży stopień przeszukiwania przestrzeni problemu. Liczba wszystkich możliwych wektorów różnicowych określona jest przez wzór[32]:

$$\binom{\lambda}{2k} 2k! \approx \mathcal{O}(\lambda^{2k}) \tag{2.21}$$

Powyższa zależność wyznacza liczność zbioru wszelkich możliwych kierunków, które mogą być wygenerowane przez populację. Zwiększanie wielkości populacji lub liczby wektorów różnicowych powoduje zwiększenie siły eksploracyjności metody.

Mutacja, oparta na dodaniu do punktu bazowego przeskalowanej sumy wektorów różnicowych, powoduje uzależnienie potomków od rozproszenia w populacji. Macierz kowariancji rozkładu generującego wektory różnicowe jest proporcjonalna do empirycznej macierzy kowariancji populacji rodzicielskiej. Twierdzenie 2.2.1 dowodzi tego faktu.

Twierdzenie 2.2.1. *Jeśli* $X = \{\mathbf{x}_j : j = 1, \ldots, \lambda\}$ *będzie populacją rodzicielską w algorytmie DE (zgodnym z rysunkiem 2.4), wówczas rozkład wektorów różnicowych* $\mathbf{d}_i$*, gdy indeksy* $\{r_1, ..., r_{2k}\}$ *są niezależnymi realizacjami tej samej zmiennej losowej w zbiorze* $\{1, ..., \lambda\}$*, a* $\mathbf{S}(X)$ *oznacza empiryczną macierz kowariancji populacji* X*, jest scharakteryzowany przez wartość oczekiwaną oraz macierz kowariancji równą odpowiednio:*

$$E\left[\mathbf{d}_i\right] = \mathbf{0}, \quad \mathbf{\Sigma}\left[\mathbf{d}_i\right] = 2k\frac{\lambda - 1}{\lambda}\mathbf{S}(X) \tag{2.22}$$

Dowód. Jeśli pojedynczy indeks j został wygenerowany z użyciem zmiennej losowej o rozkładzie jednostajnym na zbiorze $\{1, ..., \lambda\}$, wówczas współrzędne punktu $\mathbf{x}_j$ wylosowanego z X są opisane dyskretną zmienną losową, której rozkład wartości oczekiwanej oraz macierz kowariancji wynoszą:

$$E\left[\mathbf{x}_j\right] = \frac{1}{\mu}\sum_{i=1}^{\mu} x_i, \quad \mathbf{\Sigma}\left[\mathbf{x}_j\right] = \frac{\lambda - 1}{\lambda}\mathbf{S}(X)$$

W przypadku odejmowaniu dwóch niezależnych realizacji tej samej zmiennej losowej, jej wartości oczekiwane są odejmowane, a macierze kowariancji dodawane.

$$E\left[\mathbf{d}_i\right] = E\left[\sum_{j=1}^{k}\left(\mathbf{x}_{r_{2j-1}} - \mathbf{x}_{r_{2j}}\right)\right] = \sum_{j=1}^{k}\left(E\left[\mathbf{x}_{r_{2j-1}}\right] - E\left[\mathbf{x}_{r_{2j}}\right]\right) = \mathbf{0}$$

$$\begin{aligned}\mathbf{\Sigma}\left[\mathbf{d}_i\right] &= \mathbf{\Sigma}\left[\sum_{j=1}^{k}\left(\mathbf{x}_{r_{2j-1}} - \mathbf{x}_{r_{2j}}\right)\right] \\ &= \sum_{j=1}^{k}\left(\mathbf{\Sigma}\left[\mathbf{x}_{r_{2j-1}}\right] + \mathbf{\Sigma}\left[\mathbf{x}_{r_{2j}}\right]\right) \\ &= 2k\mathbf{\Sigma}\left[\mathbf{x}_j\right] = 2k\frac{\lambda - 1}{\lambda}\mathbf{S}(X)\end{aligned}$$

Co dowodzi 2.2.1. □

Zerowa wartość oczekiwana wektorów różnicowych sprawia, że w kolejnych generacjach nie będzie występowało zjawisko dryfu genetycznego (zjawisko Wrighta), będącego procesem polegającym na fluktuacji częstości występowania danej cechy w populacji, która nie wynika z mutacji, ani sposobu selekcji. Populacje nie zostaną więc zdominowane

przez pewne cechy (współrzędne), ani żadna z nich nie ulegnie losowej eliminacji. Macierz kowariancji rozkładu generowanego przez wektory różnicowe zależy w największym stopniu od ich liczby. Zwiększanie parametru k wpływa więc na poziom różnorodności genetycznej, poprzez zwiększanie siły mutacji. Warto również zauważyć, że skoro indeksy $\{r_1, ..., r_{2k}\}$ są niezależnymi realizacjami tej samej zmiennej losowej, to w szczególności mogą one wszystkie być parami równe, co skutkować będzie zerową wartością zmiany w wyniku mutacji.

Współczynnik skalujący

Parametr F w algorytmie DE pełni rolę wzmocnienia w operatorze mutacji różnicowej. W typowym podejściu[50] wartość współczynnika F jest liczbą rzeczywistą zawartą w przedziale $F \in [0, 2]$. Niskie wartości współczynnika F powodują zmniejszenie kroku ewolucji, wykonywanego w stronę wektorów różnicowych, co może powodować wydłużenie procesu zbieżności do optimum. Wyższe wartości parametru skalującego mogą wspierać fazę eksploracji, zwiększają jednakże prawdopodobieństwo ominięcia optimum. Wartość F musi być więc odpowiednio niska, aby zapewniać eksplorowanie minimów o małych obszarach przyciągania, jednocześnie będąc odpowiednio wysoka, aby zapewnić poziom różnorodności populacji umożliwiający eksplorację. Z uwagi na to, że operator mutacji różnicowej posiada wbudowaną zdolność samoadaptacji długości kroku, wynikającą ze sposobu tworzenia wektorów różnicowych (zjawisko opisane szerzej w rozdziale 2.2.1[Mutacja różnicowa]), dobór współczynnika F pełni tylko rolę wspomagającą adaptację kroku, głównie w początkowych fazach procesu optymalizacji. Szereg badań empirycznych wykazuje, że dobór zbyt dużej wartości parametru F może powodować przedwczesną zbieżność[33][29], a rekomendowana wartość współczynnika skalującego jest zawarta w przedziale $F \in [\frac{1}{2}, 1]$ [38][53][50].

Krzyżowanie

Operator krzyżowania w algorytmie ewolucji różnicowej implementuje ideę dyskretnej rekombinacji pomiędzy genetycznym rodzicem $\mathbf{x}_i^{(t)}$ a mutantem $\mathbf{y}_i^{(t)}$. Tworzony jest punkt kandydujący $\mathbf{z}_i^{(t)}$ posiadający kod genetyczny będący kombinacją materiałów obu punktów, biorących udział w krzyżowaniu. Definicja tego operatora genetycznego jest trzecim parametrem schematu ewolucji różnicowej, charakteryzującym krzyżowanie. Istnieją dwa

najpopularniejsze warianty krzyżowania, zaproponowane przez autorów metody DE w pierwotnym artykule:

DE/R/k/bin : Podejście to nazywane jest dwumianowym (*ang. binomial*) i polega na losowych próbach wyboru materiału genetycznego od krzyżowanych punktów, niezależnie dla każdej j-tej współrzędnej (j-tego genu).

$$z_{i,j}^{(t)} = \begin{cases} y_{i,j}^{(t)} & \text{if} \quad rand(0,1) \leqslant C_r \\ x_{i,j}^{(t)} & \text{w przeciwnym wypadku} \end{cases} \qquad 1 \leqslant j \leqslant n$$

Parametr C_r, określający poziom prawdopodobieństwa wyboru współrzędnej od mutanta, jest stały dla wszystkich prób. Rozkład opisujący zadaną liczbę takich sukcesów jest więc opisany rozkładem Bernoulliego. Gdy wartość C_r będzie niska, może się zdarzyć, że punktem kandydującym, a w efekcie sukcesji potomkiem, stałby się rodzic $\mathbf{x}_i^{(t)}$. Aby uniemożliwić spadek różnorodności genetycznej populacji wymaga się niekiedy, aby co najmniej jedna współrzędna w wynikowym punkcie kandydującym $\mathbf{z}_i^{(t)}$ musiała pochodzić od mutanta. Coraz częściej jednakże, na gruncie matematycznej analizy, wykazywany jest brak konieczności używania tego wymagania[13].

DE/R/k/exp : W tej strategii, począwszy od losowej współrzędnej, wyznaczany jest przedział współrzędnych mutanta, który jest kopiowany do punktu kandydującego. Punkt kandydujący $\mathbf{z}_i^{(t)}$ posiada więc materiał rodzica genetycznego, za wyjątkiem ciągu genów skopiowanych od mutanta.

$$z_{i,j}^{(t)} = \begin{cases} y_{i,j}^{(t)} & \text{if} \quad j-1 \in \{m(\text{mod n}), \ldots, (m+l)(\text{mod n})\} \\ x_{i,j}^{(t)} & \text{w przeciwnym wypadku} \end{cases} \qquad 1 \leqslant j \leqslant n$$

Indeks m stanowi początkową współrzędną przedziału i jest opisany z użyciem rozkładu jednostajnego na zbiorze $[1, \ldots, n]$. Wielkość kopiowanego przedziału determinowana jest poprzez zmienną losową l opisaną rozkładem wykładniczym z prawdopodobieństwem sukcesu C_r, w którym sekwencja współrzędnych mutanta traktowana jest jako bufor cykliczny.

Wielkość populacji

Zgodnie ze wzorem 2.21, wielkość populacji λ ma bezpośredni wpływ na liczbę możliwych kierunków wektorów różnicowych, a więc na potencjalną zdolność eksploracji algorytmu

DE. Im większa populacja, tym więcej możliwych kierunków wygenerowanych przez mechanizm mutacji różnicowej, które mogą zostać eksplorowane. Należy jednak pamiętać, że wraz ze wzrostem liczby punktów w populacji, rośnie nakład obliczeniowy generacji, a więc także przewidywany czas zbieżności całego algorytmu. Zgodnie z zaleceniami autorów metody[53] wielkość populacji powinna zawierać się w przedziale $\lambda \in [3n, 10n]$.

Empiryczne badania[42] wpływu wielkości populacji na jakość optymalizacji w algorytmie DE (na typowej grupie problemów CEC opisywanej szerzej w rozdziale 4.1.1), ukazują zależność pomiędzy badaną wielkością a naciskiem selektywnym. Populacje rzędu $\lambda = 2n$, w schematach z największym naciskiem na eksploatację np. **DE/best/1**, wykazują najszybszą zbieżność, posiadając jednocześnie największą tendencje do stagnacji lub przedwczesnej zbieżności do lokalnych minimów. Z drugiej strony badania dowodzą, że algorytm **DE/rand-to-best/2**, dla dużych populacji $\lambda = 8n$ oraz $\lambda = 10n$, charakteryzuje się niewielkim ryzykiem przedwczesnego zbiegania do lokalnych ekstremów i stagnacji, kosztem dużo większego czasu zbieżności. Optymalny dobór wielkości populacji zależy więc od założonego poziomu nacisku selektywnego metody.

Nakład obliczeniowy

Jedna iteracja algorytmu ewolucji różnicowej posiada nakład obliczeniowy rzędu $\mathcal{O}(n \cdot k \cdot \lambda)$. Metoda DE posiada kilka czynników wpływających na taki poziom nakładu.

Generowanie wektorów różnicowych wymaga przeprowadzenia operacji odejmowania dwóch punktów, do czego niezbędne jest n operacji na wartościach współrzędnych. Liczba wektorów różnicowych wchodzących w skład sumy $\mathbf{d}_i^{(t)}$, ustalona przez parametr k, multiplikuje ilość operacji. Do obliczenia sumy k wektorów różnicowych potrzebne jest $k \cdot n$ operacji dodawania. Wyznaczenie mutanta $\mathbf{y}_i^{(t)}$ wymaga n operacji mnożenia na $\mathbf{d}_i^{(t)}$ oraz n operacji dodawania współrzędnych wektora wynikowego i punktu bazowego $\mathbf{x}_{base,i}$.

Niezależny sposób generowania punktów potomnych dla każdego elementu populacji sprawia, że wielkość populacji wpływa na nakład obliczeniowy. Wartość parametru λ multiplikuje wszystkie wyznaczone powyżej liczby niezbędnych operacji. Wynikowy nakład obliczeniowy jednej iteracji algorytmu ewolucji różnicowej jest więc rzędu $\mathcal{O}(n \cdot k \cdot \lambda)$ i jest zależny liniowo od wymiarowości problemu.

2.2.2 Linie rozwojowe algorytmu DE

Bazowa wersja metody DE posiada szereg problemów, które utrudniają jej uniwersalne stosowanie. Ewolucja różnicowa jest globalnym optymalizatorem o dużej tendencji eksploracji przestrzeni, ale stosunkowo niskiej zdolności eksploatacji znalezionego obszaru przyciągania [44]. Przekłada się to wprost na niższe tempo zbieżności w stosunku do metod o większym nacisku na przeszukiwanie lokalne (np. CMA-ES), szczególnie na funkcjach unimodalnych.

Kolejnym problemem metody DE jest jej duża wrażliwość na sposób parametryzacji. Jakość wyników uzyskiwanych przez algorytm ewolucji różnicowej zależy od doboru odpowiedniego wariantu metody DE. Zgodnie z analizą schematów notacji DE/R/k/C w rozdziale 2.2.1, wewnętrzne parametry metody pełnia kluczową rolę w zarządzaniu różnorodnością populacji oraz szybkości zbieżności procesu optymalizacji. Optymalny dobór wariantu metody DE wymaga znajomości typu funkcji celu lub podejścia eksperymentalnego. Utrudnia to użycie ewolucji różnicowej jako uniwersalnego optymalizatora globalnego.

Algorytm DE cechuje się brakiem inwariantności rotacyjnej[12] - metoda nie posiada właściwości niezmienności poziomu skuteczności, przy obrotach optymalizowanej funkcji celu. Inwariantność tyczy się zachowania porównywalnej wydajności optymalizacji, przy przekształceniach jednorodnych funkcji celu. Klasyczny algorytm ewolucji różnicowej posiada zdolność inwariantności przy skalowaniu i translacji[45]. Brak inwariantności przy obrotach funkcji celu wynika z zastosowania operatora krzyżowania, który wymienia materiał genetyczny punktów wzdłuż osi układu współrzędnych przestrzeni problemu. Rozbieżność w skuteczności optymalizacji metody DE przy obrotach funkcji celu uniemożliwia skuteczną parametryzację algorytmu, nawet w kontekście jednej grupy problemów. Ten fakt utrudnia użycie ewolucji różnicowej jako uniwersalnego optymalizatora globalnego

Poprawa szybkości zbieżności

Jednym ze sposobów zwiększenia tempa zbieżności metody DE jest poprawa sposobu doboru jej parametrów. Zgodnie z opisem algorytmu w rozdziale 2.2.1, metoda posiada kilka współczynników, których wybór wpływa bezpośrednio na równowagę pomiędzy eksploracją a eksploatacją oraz zdolność wydłużania kroku. Kombinacja przyjętego schematu DE/R/k/C, prawdopodobieństwa sukcesu w krzyżowaniu C_r, współczynnika skalujące-

go F oraz wielkości populacji λ, charakteryzuje sposób działania metody DE. Istnieją prace[56] dokonujące klasyfikacji sposobów strojenia algorytmów ewolucji różnicowej, w oparciu o zaproponowaną taksonomię, w kontekście parametryzacji pojedynczej generacji. Takie podejście jest pewnym gotowym przepisem, umożliwiającym łatwiejsze osiąganie założonej charakterystyki działania metody DE. Z uwagi na to, że uzyskiwana takim sposobem strojenia jakość nie wykracza poza możliwości klasycznego algorytmu ewolucji różnicowej, poprawa szybkości zbieżności jest więc tylko względna. Znane są metody implementujące automatyczny proces strojenia, traktując parametry, które mają wartości stałe w bazowej metodzie DE, jako dynamicznie zmienne. Algorytm SADE[48] samodzielnie dobiera współczynniki C_r oraz F, losując je z rozkładem normalnym o niskiej wariancji, dla każdego punktu z osobna.

Kolejnym sposobem poprawy szybkości zbieżności jest rozciąganie populacji w kierunku największej poprawy. Algorytm JADE[61] wprowadza nowy wariant reprodukcji do notacji DE/R/k/C. Zaproponowana przez autorów strategia **DE/current-to-ρbest/k/C** proponuje wykorzystanie archiwalnych punktów gorszej jakości z poprzednich generacji. Punkty te są punktami kandydującymi, które przegrały z rodzicem podczas sukcesji, tworząc zwiększające się co generacje archiwum, oznaczane przez $\mathbf{A}$. Wzór tworzący mutanta $y_i^{(t)}$ w algorytmie JADE wygląda następująco:

$\mathbf{y}_i^{(t)} \leftarrow \mathbf{x}_{rand_1} + F \cdot (\widehat{\mathbf{x}}^{\rho} - \mathbf{x}_{rand_1}) + F \cdot (\mathbf{x}_{rand_2} - \mathbf{x}_{rand}^{P \cup A})$, gdzie $\mathbf{x}_{rand_k} = \mathbf{x}_{(\sim U(1,\ldots,\lambda))}$. Punkt $\widehat{\mathbf{x}}^{\rho}$ wyznaczany jest na podstawie ρ procent punktów bieżącej populacji $\mathbf{P}$. Punkty $\mathbf{x}_{rand_k}$ oraz $\mathbf{x}_{rand}^{P \cup A}$ losowane są z rozkładem jednostajnym odpowiednio, tylko z bieżącej populacji $\mathbf{P}$, oraz ze zbioru zsumowanej populacji $\mathbf{P}$ z archiwum $\mathbf{A}$. Zaproponowana przez autorów strategia **DE/current-to-ρbest/k/C** jest generalizacją strategii **DE/current-to-best/k/C**(Rozdział 2.2.1[Selekcja]), z wykorzystaniem archiwum punktów dostarczającym dodatkowej informacji o kierunku postępu optymalizacji. Dodanie do wzoru na punkt bazowy $y_i^{(t)}$ czynnika przeciągającego go w kierunku poprawy jakości, z perspektywy archiwalnych punktów $\mathbf{x}_{rand}^{P \cup A}$ gorszej jakości z poprzednich generacji, zwiększa zachłanność działania metody. Zwiększenie szybkości zbieżności, w stosunku do klasycznego DE, będzie przy takim podejściu zależeć od modalności optymalizowanej funkcji. Im więcej ekstremów posiadać będzie funkcja celu, tym gorzej będzie się sprawdzać podejście zachłanne. Z tego powodu algorytm JADE powinien być postrzegany jako rozszerzenie puli wariantów ewolucji różnicowej, które tylko dla pewnych typów problemów może poprawić

szybkość zbieżności.

Zmniejszenie wrażliwości na parametryzację

Sposobem na zmniejszenie wrażliwości metody DE na sposób jej parametryzacji jest implementacja mechanizmu automatycznego strojenia. Metoda jDE[10] wprowadza mechanizm zmieniający w czasie działania algorytmu wartości podstawowych parametrów metody DE. Mechanizm zaimplementowany w jDE umożliwia przydzielanie różnych wartości współczynnika skalującego F oraz prawdopodobieństwa krzyżowania C_r, dla każdego punktu populacji, który jest używany do tworzenia wektorów różnicowych. Początkowo parametry te są inicjowane wartościami: $F = 0.5$, $C_r = 0.9$. Wartości parametrów danego punktu są dziedziczone od rodzica z poprzedniej generacji, z zaimplementowanym mechanizmem mutacji. Każdy z dziedziczonych parametrów może być zmieniony losowo z pewnym rozkładem prawdopodobieństwa. Tak zmodyfikowane parametry mogą być przekazane do punktu w kolejnej generacji tylko wtedy, gdy punkt kandydujący z sukcesem przejdzie proces rywalizacji. Algorytm jDE dokonuje automatycznego doboru tylko dwóch podstawowych parametrów parametrów - F oraz C_r, gdy w niektórych strategiach parametrów do strojenia może być więcej. Proces doboru nowych współczynników F oraz C_r implementujący dziedziczenie w efekcie wykorzystuje tylko wiedzę lokalną, z poprzedniej populacji, nie zapisując i nie uwzględniając historycznych udanych ustawień parametrów.

Algorytm SHADE[57] proponuje technikę adaptacji parametrów dla DE, która wykorzystuje pamięć udanych ustawień parametrów do wyboru przyszłych wartości parametrów. Metoda SHADE dokonuje adaptacji parametrów wraz z każdą kolejną generacją. SHADE jest rozszerzeniem algorytmu JADE (opisanego w podrozdziale 2.2.2[Poprawa szybkości zbieżności]), który wykorzystuje archiwalne punkty z poprzednich generacji do aktualizacji parametrów rozkładu generującego współczynniki F oraz C_r, zgodnie z rozkładem normalnym. W algorytmie JADE historyczne punkty służą tylko do aktualizacji parametrów dwóch rozkładów normalnych - średniej μ_{CR} w rozkładzie generującym prawdopodobieństwa krzyżowania oraz średniej μ_F w rozkładzie generującym współczynnik skalujący. Autorzy metody SHADE zaproponowali rozwiązanie adaptacji parametrów opartej na przechowywaniu historii współczynników punktów zwycięskich. Zamiast generować nowe parametry losowo, w oparciu o pewne rozkłady wokół średnich μ_{CR} i μ_F, SHADE używa pamięci historycznej M_{CR}, M_F, która przechowuje zestawy parametrów

C_r oraz F, które dobrze działały w przeszłości. Nowe parametry C_r oraz F, dla bieżącej populacji, generowane są poprzez bezpośrednie próbkowanie przestrzeni parametrów w pobliżu jednej z tych przechowywanych par M_{CR} i M_F. Metoda SHADE implementuje rzeczywistą adaptację parametrów F oraz C_r sprawiając, że ich dobór nie wymaga znajomości typu funkcji celu. Niestety ten mechanizm adaptacyjny dotyczy tylko dwóch podstawowych parametrów DE, wybór typu strategii oraz jego parametrów dalej wymaga podejścia eksperymentalnego, co utrudnia użycie ewolucji różnicowej jako uniwersalnego optymalizatora globalnego.

Adaptacja parametrów w metodzie SHADE wymaga zbudowania historii współczynników punktów zwycięskich. Oczekuje się, że metoda będzie działać tym skuteczniej im więcej punktów w historii będzie się znajdować. Wskazane jest, aby początkowe populacje były duże, a późniejsze populacje malały wraz z każdą kolejną generacją. Algorytm L-SHADE[58] jest rozszerzeniem metody SHADE o technikę liniowej redukcji wielkości populacji, która w każdej kolejnej generacji zmniejsza liczność populacji, zgodnie z liniową zależnością. Idea ta pozwala na zwiększenie skuteczności działania mechanizmu adaptacji oraz lepsze rozłożenie nacisku selektywnego, w początkowych fazach optymalizacji skierowanego na eksplorację i z każdą kolejną generacją dążącego do eksploatacji.

Rozdział 3

Definicja algorytmu różnicowej strategii ewolucyjnej i analiza jego właściwości

Charakterystyki algorytmów CMA-ES oraz ewolucji różnicowej przedstawione w rozdziale 2 ukazują istotne ograniczenia metod z tych rodzin, w kontekście ich użycia do optymalizacji globalnej. Iteracyjne przybliżanie dodatnio określonej macierzy kowariancji w strategii CMA-ES obarczone jest dużym poziomem nakładu obliczeniowego. Tym samym mechanizm implementujący pożądaną własność dla wszystkich algorytmów optymalizacji stochastycznej (rozdział 1.1), czyli dopasowywanie się do poziomic funkcji celu, ogranicza działanie metody CMA-ES w zakresie jej stosowania w problemach o dużej liczbie wymiarów. Klasyczna ewolucja różnicowa natomiast, implementując mechanizm stochastycznego różnicowania punktów, posiada nakład obliczeniowy liniowo zależny od wymiarowości, a więc może być łatwiej niż CMA-ES wykorzystywana dla wielowymiarowych problemów. Metoda ta nie posiada jednak wykazanej wprost cechy dopasowywania się do poziomic funkcji celu i może także manifestować dużą zależność skuteczności działania od obrotów, bądź kształtu funkcji celu. Wydaje się, że idealnym łącznikiem pomiędzy tymi dwoma rodzinami metaheurystyk byłby algorytm, który implementowałby sposób działania metody CMA-ES i cechę dopasowania do kształtu funkcji celu, używając do tego nieskomplikowanej numerycznie idei mutacji różnicowej. Poniższy rozdział jest definicją algorytmu wpisującego się w tę ideę, będącego połączeniem metody CMA-ES z ewolucją różnicową i wprowadzającej metodę różnicowej strategii ewolucyjnej — DES (*ang. Differential*

Evolution Strategy).

Jak dotąd w literaturze z dziedziny algorytmów ewolucji różnicowej oraz CMA-ES spotkać można zaledwie kilka koncepcji przedstawiających możliwość współdzielenia mechanizmów pomiędzy tymi rodzinami metod. Autorzy podejść HybridCMAESandDE[34] oraz MMOED[51] proponują cykliczne przełączanie pomiędzy populacjami, w których punkty generowane są przez CMA-ES oraz DE. W metodzie SPACMA[43] strategia mutacji zaproponowanego algorytmu jest kombinacją mutacji różnicowej oraz reguły generacyjnej CMA-ES. Idea ścieżki ewolucji wykorzystywanej przez CMA-ES (rozdział 2.1.2) została natomiast wykorzystana przez algorytm DEEP[37] do wzbogacenia formuły generowania nowych punktów, poprzez dodanie skumulowanego wektora przesunięć punktów środkowych. Wszystkie z przytoczonych metod są pewnym sposobem przeplatania ewolucji różnicowej i CMA-ES, a nie rzeczywistym połączeniem obu metod w jedną.

3.1 Definicja algorytmu DES

Algorytm ewolucyjny będący przedmiotem niniejszej rozprawy doktorskiej, jest metodą generującą populacje zgodne co do parametrów rozkładu z CMA-ES, lecz bez konieczności używania kosztownych przekształceń algebry macierzowej. Idea opiera się na tworzeniu punktów potomnych będących kombinacją wektorów różnicowych (pomiędzy losowo wybranymi punktami rodzicielskimi) oraz losowego wektora o rozkładzie normalnym (wygenerowanego zgodnie z kierunkiem przesunięć środka populacji przed i po selekcji). Algorytm dodatkowo używa archiwum populacji oraz punktów środkowych przeszłych generacji. Twierdzenia 2.1.1 oraz 2.2.1 ukazują podobieństwo pomiędzy empirycznym rozkładem generowanym przez CMA-ES, a rozkładem punktów generowanych przez wektory różnicowe w algorytmie DE. Poniższy rozdział definiuje algorytm różnicowej strategi ewolucyjnej (DES) oraz dowodzi możliwości zachowania właściwości adaptacji empirycznej macierzy kowariancji i generowania kolejnych populacji z rozkładem zgodnym z CMA-ES, bez konieczności obliczania, estymacji oraz nawet przechowywania zdefiniowanej explicite macierzy kowariancji.

Zgodnie z twierdzeniem 2.1.1, równanie 2.12 na macierz kowariancji w algorytmie CMA-ES bazuje na ważonej sumie empirycznych macierzy kowariancji punktów z poprzednich generacji oraz na macierzy rank-1 (opisanej w rozdziale 2.1.2), opierającej się

na ważonej sumie przesunięć punktów środkowych. Twierdzenie 2.2.1 pokazuje sposób modelowania macierzy kowariancji, tym samym pierwszej części formuły (2.12). Sposób modelowania drugiej części równania, a więc macierzy rank-1, ukazuje poniższa obserwacja.

Obserwacja 3.1.1. *Jeśli* $\mathbf{v} = [\xi, 0, ..., 0]^\mathsf{T} \in \mathbb{R}^n$ *oznaczać będzie losowy wektor, gdzie* $\xi \sim N(0,1)$ *jest zmienną losową o rozkładzie normalnym standaryzowanym, oraz jeśli macierz* $\mathbf{B} = \begin{bmatrix} \mathbf{b} & \mathbf{0} & \dots \end{bmatrix}$ *będzie macierzą przekształcenia liniowego, gdzie* $\mathbf{b} \in \mathbb{R}^n$ *jest wektorem kolumnowym, wówczas wektor losowy* $\mathbf{w} = \mathbf{Bv}$ *jest scharakteryzowany przez wartość oczekiwaną równą zero oraz macierz kowariancji* $\mathbf{\Sigma}[\mathbf{w}] = \mathbf{b}\mathbf{b}^\mathsf{T}$.

Dowód. Wektor losowy $\mathbf{v}$ jest scharakteryzowany przez wartość oczekiwaną równą zero oraz macierz kowariancji w postaci:

$$\mathbf{\Sigma}[\mathbf{v}] = \begin{bmatrix} 1 & 0 & \dots \\ 0 & 0 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

Macierz kowariancji wektora $\mathbf{w}$ może być więc przedstawiona w postaci:

$$\mathbf{\Sigma}[\mathbf{w}] = \mathbf{\Sigma}[\mathbf{bv}] = \mathbf{b}\mathbf{\Sigma}[\mathbf{v}]\mathbf{b}^\mathsf{T} = \mathbf{b}\mathbf{b}^\mathsf{T}$$

co dowodzi 3.1.1. □

Wynika z tego, że jest możliwe generowanie losowych wektorów o macierzy kowariancji $\mathbf{b}\mathbf{b}^\mathsf{T}$, poprzez wymnożenie standaryzowanej zmiennej losowej o rozkładzie normalnym przez wektor $\mathbf{b}$. Naturalną konsekwencją obserwacji 3.1.1 jest wyjaśnienie, w jaki sposób algorytm CMA-ES akumuluje kolejne wektory przesunięcia następujących po sobie wektorów punktów środkowych[27][47]. Ważona suma $\mathbf{w}$ z k niezależnych jednowymiarowych zmiennych losowych o rozkładzie normalnym ξ_i, wagach p_i, wymnożonych przez odpowiadające im wektory $\mathbf{b}_i$ przyjmuje postać:

$$\mathbf{w} = \sum_{i=1}^{k} p_i \xi_i \mathbf{b}_i$$

oraz jest scharakteryzowana rozkładem z zerową wartością oczekiwaną i macierzą kowariancji wynoszącą:

$$\mathbf{\Sigma}[\mathbf{w}] = \sum_{i=1}^{k} p_i^2 \mathbf{b}_i \mathbf{b}_i^\mathsf{T}$$

Innym sposobem agregacji jednowymiarowych wektorów losowych do postaci macierzy kowariancji jest użycie losowej kombinacji tych wektorów. Niech $\mathbf{b}_i \in \mathbb{R}^n$ dla $i = 1, \ldots, k$ będzie zbiorem wektorów, a ζ będzie zmienną losową opisana rozkładem prawdopodobieństwa $\text{Prob}[\zeta = i] = p_i, i = 1, \ldots, k$. Wówczas wektor $\mathbf{v}$ zdefiniowany jako:

$$\mathbf{v} = \mathbf{b}_\zeta \cdot \xi$$

gdzie $\xi \sim N(0, 1)$ jest zmienną losową o rozkładzie normalnym standaryzowanym, opisany jest rozkładem z zerową wartością oczekiwaną i macierzą kowariancji wynoszącą:

$$\mathbf{\Sigma}[\mathbf{v}] = \sum_{i=1}^{k} p_i \mathbf{b}_i \mathbf{b}_i^\mathsf{T}$$

Obserwacja 3.1.1 oraz jej opisane konsekwencje, stwarzają możliwość modelowania macierzy rank-1. Ten fakt, wraz z twierdzeniami 2.1.1 oraz 2.2.1, daje możliwość zachowania właściwości adaptacji empirycznej macierzy kowariancji i generowania kolejnych populacji z rozkładem zgodnym z CMA-ES, bez konieczności używania skomplikowanego obliczeniowo procesu obliczania lub estymacji macierzy kowariancji. Metody implementujące te obserwacje są przedmiotem poniższych podrozdziałów.

3.1.1 Metoda CMA-DE

Algorytm CMA-DE (*ang. Covariance Matrix Adaptation - Differential Evolution*) modeluje proces adaptacji macierzy kowariancji, tak aby rozkłady generowanych punktów w kolejnych populacjach były zgodne, co do parametrów rozkładu, z metodą CMA-ES. Rysunek 3.1 ukazuje zarys CMA-DE. W każdej generacji metoda ta jest scharakteryzowana poprzez populację, która zawiera λ losowo wygenerowanych punktów. Po ewaluacji, tylko grupa μ najlepszych punktów przechodzi selekcję i bierze dalszy udział w procesie reprodukcji. Następnie punkt środkowy $\mathbf{m}^{(t+1)}$, będący ważoną średnią tej grupy punktów, pozwala wyznaczyć wektor $\mathbf{\Delta}^{(t)}$ wyrażający przesunięcie pomiędzy punktami środkowymi obecnej i poprzedniej generacji oraz jego ostateczne skumulowanie w wektorze $\mathbf{p}^{(t)}$. Punkt środkowy $\mathbf{m}^{(t+1)}$ staje się punktem wyjściowym w procesie generowania całej nowej populacji λ mutantów, która będzie stanowić wejście w kolejnej iteracji procesu ewolucji. Proces generowania mutantów opiera się na wyznaczaniu wektorów różnicowych $\mathbf{d}_i^{(t)}$, z których każdy jest ważoną sumą czterech elementów:

1. wektora różnicowego pomiędzy pomiędzy punktami z grupy μ najlepszych z iteracji $t - \tau_3$

2. iloczynu wektora przesunięcia punków środkowych z generacji $t - \tau_2$ oraz zmiennej losowej o rozkładzie normalnym standaryzowanym,

3. iloczynu wektora kumulującego przesunięcia punków środkowych z generacji $t - \tau_1$ oraz zmiennej losowej o rozkładzie normalnym standaryzowanym,

4. wielowymiarowego wektora losowego o rozkładzie normalnym standaryzowanym.

Indeksy $\tau_1, \tau_2, \tau_3 \sim G_{c_{cov}}(1, ..., t)$ są losowane zgodnie z rozkładem geometrycznym, przeskalowanym do zakresu $1, ..., t$, w którym prawdopodobieństwo wylosowania generacji τ wynosi:

$$P\{\tau\} = \alpha c_{cov} \cdot (1 - c_{cov})^{t-\tau}, \qquad \tau \in \{1, ..., t\}$$

gdzie $\alpha = \frac{1}{1-(1-c_{cov})^t}$ jest współczynnikiem normalizującym, który gwarantuje sumowanie się prawdopodobieństw do jedności.

Algorytm CMA-DE w tak zaproponowanym formacie, modeluje proces adaptacji punktu środkowego $\mathbf{m}^{(t)}$ oraz macierzy kowariancji zgodnie z metodą CMA-ES. Własność ta jest sformułowana w postaci poniższego twierdzenia, które ukazuje jak rozkład wektorów różnicowych generowanych przez CMA-DE aproksymuje wielowymiarowy rozkład normalny używany do generowania punktów przez CMA-ES .

Twierdzenie 3.1.2. *Rozkład prawdopodobieństwa używany do generowania wektorów różnicowych $\mathbf{d}_i^{(t)}$ w algorytmie CMA-DE jest scharakteryzowany przez wartość oczekiwaną oraz macierz kowariancji równe odpowiednio:*

$$E\left[\mathbf{d}_i^{(t)}\right] = \mathbf{0} \tag{3.1}$$

$$\begin{aligned}\boldsymbol{\Sigma}\left[\mathbf{d}_\mathbf{i}^{(\mathbf{t})}\right] =& c_\mu \frac{\mu - 1}{\mu} \sum_{\tau=1}^{t} (1 - c_{cov})^{t-\tau} \mathbf{S}\left(X_\mu^{(\tau)}\right) \\ &+ c_1 \sum_{\tau=1}^{t} (1 - c_{cov})^{t-\tau} \mathbf{p}^{(\tau)} \left(\mathbf{p}^{(\tau)}\right)^\mathsf{T} \\ &+ c_\mu \sum_{\tau=1}^{t} (1 - c_{cov})^{t-\tau} \boldsymbol{\Delta}^{(\tau)} \left(\boldsymbol{\Delta}^{(\tau)}\right)^\mathsf{T} + (1 - c_{cov})^t \mathbf{I}\end{aligned} \tag{3.2}$$

Dowód. Zgodnie z twierdzeniem 2.2.1:

$$E\left[\mathbf{x}_j^{(\tau)} - \mathbf{x}_k^{(\tau)}\right] = \mathbf{0} \tag{3.3}$$

$$\boldsymbol{\Sigma}\left[\mathbf{x}_j^{(\tau)} - \mathbf{x}_k^{(\tau)}\right] = 2\frac{\mu - 1}{\mu} \mathbf{S}\left(X_\mu^{(\tau)}\right) \tag{3.4}$$

1: $t \leftarrow 1$, $\mathbf{p}^{(0)} \leftarrow 0$

2: inicjacja$\left(X^{(1)} = \left\{\mathbf{x}_1^{(1)}, ..., \mathbf{x}_\lambda^{(1)}\right\}\right)$

3: $\mathbf{m}^{(1)} = \dfrac{1}{\lambda}\sum_{i=1}^{\lambda}\mathbf{x}_i^{(1)}$

4: evaluacja$\left(X^{(1)}\right)$

5: **while** !stop **do**

6: $\quad\mathbf{m}^{(t+1)} = \dfrac{1}{\mu}\sum_{i=1}^{\mu}\mathbf{x}_i^{(1)}$

7: $\quad\mathbf{\Delta}^{(t)} \leftarrow \mathbf{m}^{(t+1)} - \mathbf{m}^{(t)}$

8: $\quad\mathbf{p}^{(t)} \leftarrow (1 - c_c)\mathbf{p}^{(t-1)} + \sqrt{\mu c_c(2 - c_c)}\mathbf{\Delta}^{(t)}$

9: $\quad$**for** $i = 1$ **to** λ **do**

10: $\quad\quad\tau_1, \tau_2, \tau_3 \sim G_{c_{cov}}\{1, ..., t\}$

11: $\quad\quad j, k \sim U(1, ..., \mu)$

12: $\quad\quad\mathbf{d}_i^{(t)} \leftarrow \sqrt{\frac{c_1}{\alpha c_{cov}}}\mathbf{p}^{(\tau_1)} \cdot N(0, 1)$

13: $\quad\quad\quad\quad+\sqrt{\frac{c_\mu}{\alpha c_{cov}}}\mathbf{\Delta}^{(\tau_2)} \cdot N(0, 1)$

14: $\quad\quad\quad\quad+\sqrt{\frac{c_\mu}{2\alpha c_{cov}}}\left(\mathbf{x}_j^{(\tau_3)} - \mathbf{x}_k^{(\tau_3)}\right)$

15: $\quad\quad\quad\quad+(1 - c_{cov})^{t/2} \cdot N(\mathbf{0}, \mathbf{I})$

16: $\quad\quad\mathbf{x}_i^{(t+1)} \leftarrow \mathbf{m}^{(t+1)} + \mathbf{d}_i^{(t)}$

17: $\quad$**end for**

18: $\quad$evaluacja$\left(X^{(t+1)}\right)$

19: $\quad t \leftarrow t + 1$

20: **end while**

Rysunek 3.1: Zarys algorytmu CMA-DE, który modeluje proces adaptacji macierzy kowariancji zgodny z metodą CMA-ES.

Z obserwacji 3.1.1 wynika, że:

$$E[\mathbf{p}^{(\tau)} \cdot N(0, 1)] = \mathbf{0} \tag{3.5}$$

$$\mathbf{\Sigma}[\mathbf{p}^{(\tau)} \cdot N(0, 1)] = \mathbf{p}^{(\tau)}\left(\mathbf{p}^{(\tau)}\right)^{\mathsf{T}} \tag{3.6}$$

$$E[\mathbf{\Delta}^{(\tau)} \cdot N(0, 1)] = \mathbf{0} \tag{3.7}$$

$$\mathbf{\Sigma}[\mathbf{\Delta}^{(\tau)} \cdot N(0, 1)] = \mathbf{\Delta}^{(\tau)}\left(\mathbf{\Delta}^{(\tau)}\right)^{\mathsf{T}} \tag{3.8}$$

Wektor różnicowy $\mathbf{d}_i^{(t)}$ jest zdefiniowany jako ważona średnia czterech niezależnych losowych czynników, których wartości oczekiwane są zgodnie z (3.3), (3.5) oraz (2.15)

równe zero, stąd:

$$E\left[\mathbf{d}_i^{(t)}\right] = \mathbf{0} \tag{3.9}$$

co dowodzi (3.1).

Niezależność czynników wchodzących w skład $\mathbf{d}_i^{(t)}$ pozwala na ważone sumowanie macierzy kowariancji:

$$\begin{aligned}\mathbf{\Sigma}\left[\mathbf{d}_i^{(t)}\right] = &\sum_{\tau_1=1}^{t} \alpha c_{cov}(1-c_{cov})^{t-\tau_1}\mathbf{\Sigma}\left[\sqrt{\frac{c_1}{\alpha c_{cov}}}\mathbf{p}^{(\tau_1)}\cdot N(0,1)\right]\\ &+\sum_{\tau_2=1}^{t} \alpha c_{cov}(1-c_{cov})^{t-\tau_2}\mathbf{\Sigma}\left[\sqrt{\frac{c_\mu}{\alpha c_{cov}}}\mathbf{\Delta}^{(\tau_2)}\cdot N(0,1)\right]\\ &+\sum_{\tau_3=1}^{t} \alpha c_{cov}(1-c_{cov})^{t-\tau_3}\mathbf{\Sigma}\left[\sqrt{\frac{c_\mu}{2\alpha c_{cov}}}\left(\mathbf{x}_j^{(\tau_3)}-\mathbf{x}_k^{(\tau_3)}\right)\right]\\ &+(1-c_{cov})^t\cdot\mathbf{I}\end{aligned} \tag{3.10}$$

Po zmianie indeksów i ekstrakcji mnożników otrzymujemy wzór:

$$\begin{aligned}\mathbf{\Sigma}\left[\mathbf{d}_i^{t}\right] = &\sum_{\tau=1}^{t}(1-c_{cov})^{t-\tau}c_1\mathbf{\Sigma}\left[\mathbf{p}^{(\tau)}\cdot N(0,1)\right]\\ &+\sum_{\tau=1}^{t}(1-c_{cov})^{t-\tau}c_\mu\mathbf{\Sigma}\left[\mathbf{\Delta}^{(\tau)}\cdot N(0,1)\right]\\ &+\sum_{\tau=1}^{t}(1-c_{cov})^{t-\tau}\frac{c_\mu}{2}\mathbf{\Sigma}\left[\left(\mathbf{x}_j^{(\tau)}-\mathbf{x}_k^{(\tau)}\right)\right]\\ &+(1-c_{cov})^t\cdot\mathbf{I}\end{aligned} \tag{3.11}$$

Zastosowanie twierdzeń 2.2.1 oraz 3.1.1 prowadzi do:

$$\begin{aligned}\mathbf{\Sigma}\left[\mathbf{d}_i^{(t)}\right] = &\sum_{\tau=1}^{t}(1-c_{cov})^{t-\tau}c_1\mathbf{p}^{(\tau)}\left(\mathbf{p}^{(\tau)}\right)^{\mathsf{T}}\\ &+\sum_{\tau=1}^{t}(1-c_{cov})^{t-\tau}c_\mu\mathbf{\Delta}^{(\tau)}\left(\mathbf{\Delta}^{(\tau)}\right)^{\mathsf{T}}\\ &+\sum_{\tau=1}^{t}(1-c_{cov})^{t-\tau}c_\mu\frac{\mu-1}{\mu}\mathbf{S}\left(D_\mu^{(\tau)}\right)\\ &+(1-c_{cov})^t\cdot\mathbf{I}\end{aligned} \tag{3.12}$$

Co dowodzi (3.2).

□

Powyższa prawidłowość oraz twierdzenie 2.1.1 dowodzi identyczności macierzy kowariancji w kolejnych populacjach w algorytmach CMA-DE oraz CMA-ES (przy założeniu

jednakowych wag we wzorze (2.7)):

$$\mathbf{\Sigma}\left[\mathbf{d}_i^{(t)}\right] \stackrel{d}{=} \mathbf{C}^{(t)} \tag{3.13}$$

Jeśli w populacji (t) macierze te były identyczne w obu algorytmach, to w populacji $(t+1)$ również będą identyczne.

Zgodnie z twierdzeniem 3.1.2, pierwsze trzy czynniki sumy wchodzące w skład $\mathbf{\Sigma}\left[\mathbf{d}^{(t)}\right]$ są wynikiem procedury wygładzania wykładniczego przeszłych wartości macierzy kowariancji empirycznej $\mathbf{S}\left(X_\mu^{(\tau)}\right)$, macierzy $\mathbf{p}_c^{(\tau)}\left(\mathbf{p}_c^{(\tau)}\right)^\mathsf{T}$ oraz $\mathbf{\Delta}^{(\tau)}\left(\mathbf{\Delta}^{(\tau)}\right)^\mathsf{T}$. Takie działanie wymaga utrzymywania historii wszystkich przeszłych populacji oraz przesunięć punktów środkowych. W generacji t sumaryczna liczba wartości pamiętanych przez metodę jest rzędu $(\mu+1)\cdot t\cdot n$, co może powodować duże zapotrzebowanie pamięciowe. Taki problem nie występuje w algorytmie CMA-ES, gdyż stan macierzy kowariancji w danej iteracji akumuluje już wszystkie poprzednie stany.

3.1.2 Metoda DES

Algorytm Różnicowej Strategii Ewolucyjnej DES (*ang. Differential Evolution Strategy*) jest modyfikacją metody CMA-DE, zwiększającą efektywność optymalizacji opartej na modelowaniu macierzy kowariancji oraz ograniczającą nakład pamięciowy. Metoda implementuje modelowanie macierzy kowariancji inspirowane algorytmem CMA-ES, bazując na rozwiązaniach wykorzystywanych w CMA-DE.

W celu ograniczenia zapotrzebowania pamięciowego, DES używa prostej idei średniej kroczącej, zamiast mechanizmu wygładzania wykładniczego, do wyznaczania macierzy wektorów różnicowych $\mathbf{d}_i^{(t)}$. Metoda wykorzystuje okno czasowe H przeszłych generacji, a więc liczba wartości liczbowych pamiętanych przez przez metodę w generacji t, gdy $t \geqslant H$, jest rzędu $\mu\cdot H\cdot n$. Takie podejście, przy odpowiednio dobranym parametrze H, pozwala znacząco ograniczyć zapotrzebowanie pamięciowe, zachowując jednocześnie własność efektywnej akumulacji przeszłych populacji. Zgodnie z (3.12) wpływ odległych historycznie populacji τ w algorytmie CMA-DE zmniejsza się wykładniczo wraz ze wzorem $(1-c_{cov})^{t-\tau}$, gdzie t jest aktualną liczbą wszystkich generacji. Wpływ na aktualny kształt macierzy kowariancji już dziesiątej historycznej populacji, wynosi mniej niż 0.1%: $(1-0.5)^{10}\approx 0.0009$ dla $c_{cov}=0.5$. Oznacza to, że praktyczne oddziaływanie historycznych populacji może zostać ograniczone do niewielkiego podzbioru H przeszłych generacji, po-

1: $t \leftarrow 1$

2: initialize$\Big(X^{(1)} = \Big\{\mathbf{x}_1^{(1)}, ..., \mathbf{x}_\lambda^{(1)}\Big\}\Big)$

3: $\mathbf{m}^{(1)} = \dfrac{1}{\lambda}\sum_{i=1}^{\lambda}\mathbf{x}_i^{(1)}$

4: evaluate$\Big(X^{(1)}, \mathbf{m}^{(1)}\Big)$

5: **while** !stop **do**

6: $\quad\mathbf{m}^{(t+1)} = \dfrac{1}{\mu}\sum_{i=1}^{\mu}\mathbf{x}_i^{(1)}$

7: $\quad\boldsymbol{\Delta}^{(t)} \leftarrow \mathbf{m}^{(t+1)} - \mathbf{m}^{(t)}$

8: $\quad$**if** $t = 1$ **then**

9: $\quad\quad\mathbf{p}^{(t)} \leftarrow \boldsymbol{\Delta}^{(t)}$

10: $\quad$**else**

11: $\quad\quad\mathbf{p}^{(t)} \leftarrow (1 - c_c)\mathbf{p}^{(t-1)} + \sqrt{\mu c_c(2 - c_c)}\boldsymbol{\Delta}^{(t)}$

12: $\quad$**end if**

13: $\quad$**for** $i = 1$ **to** λ **do**

14: $\quad\quad$pick at random $\tau_1, \tau_2, \tau_3 \in \{1, ..., H\}$

15: $\quad\quad j, k \sim U(1, ..., \mu)$

16: $\quad\quad\mathbf{d}_i^{(t)} \leftarrow \sqrt{\frac{c_d}{2}}\left(\mathbf{x}_j^{(t-\tau_1)} - \mathbf{x}_k^{(t-\tau_1)}\right)$

17: $\quad\quad\quad\quad+\sqrt{c_d}\boldsymbol{\Delta}^{(t-\tau_2)} \cdot N(0, 1)$

18: $\quad\quad\quad\quad+\sqrt{1 - c_d}\mathbf{p}^{(t-\tau_3)} \cdot N(0, 1)$

19: $\quad\quad\quad\quad+(1 - c_{cov})^{t/2} \cdot N(\mathbf{0}, \mathbf{I})$

20: $\quad\quad\mathbf{x}_i^{(t+1)} \leftarrow \mathbf{m}^{(t+1)} + \mathbf{d}_i^{(t)}$

21: $\quad$**end for**

22: $\quad$evaluate$\Big(X^{(t+1)}, \mathbf{m}^{(t+1)}\Big)$

23: $\quad t \leftarrow t + 1$

24: **end while**

Rysunek 3.2: Zarys algorytmu DES

mijając odległe czasowo populacje. Wielkość H powinna zostać dobrana eksperymentalnie, w zależności od innych współczynników, typowo nie przekraczając wartości $H = 10$.

Dodatkową własnością metody DES jest ewaluacja funkcji celu w punkcie $\mathbf{m}^{(t)}$, będącym środkiem aktualnej populacji. Cecha ta, obarczona bardzo niskim kosztem tylko jednej dodatkowej ewaluacji na generacje, pozwala zazwyczaj zwiększyć jakość ostatecznych wyników procesu optymalizacji[1].

Rysunek 3.2 prezentuje zarys algorytmu DES, w którym widoczne są opisane wyżej mechanizmy. Linia 22 ukazuje dodatkową ewaluację punktu środkowego, a w linii 14 widoczne jest losowanie numerów historycznych populacji, ze zbioru ograniczonego tylko do H ostatnich generacji. Zaproponowane zmiany w stosunku do algorytmu CMA-DE nie zmieniają właściwości modelowania procesu adaptacji macierzy kowariancji zgodnego z metodą CMA-ES. Poniższe twierdzenie dowodzi tej tezy ukazując, jak rozkład wektorów różnicowych generowanych przez DES, aproksymuje rozkład wektorów różnicowych w algorytmie CMA-DE.

Twierdzenie 3.1.3. *Rozkład prawdopodobieństwa używany do generowania wektorów różnicowych $\mathbf{d}_i^{(t)}$ w algorytmie DES jest scharakteryzowany przez zerową wartość oczekiwaną oraz macierz kowariancji równą:*

$$\begin{aligned}\mathbf{\Sigma}[\mathbf{d}_i^{(t)}] =& \frac{1}{H} c_d \frac{\mu - 1}{\mu} \sum_{\tau=t-1}^{t-H} \mathbf{S}\left(D_\mu^{(\tau)}\right) \\ &+ \frac{1}{H}(1 - c_d) \sum_{\tau=t-1}^{t-H} \mathbf{p}^{(\tau)} \left(\mathbf{p}^{(\tau)}\right)^\mathsf{T} \\ &+ \frac{1}{H} c_d \sum_{\tau=t-1}^{t-H} \mathbf{\Delta}^{(\tau)} \left(\mathbf{\Delta}^{(\tau)}\right)^\mathsf{T} + (1 - c_{cov})^t \mathbf{I} \end{aligned} \tag{3.14}$$

Dowód. Jedynymi różnicami w sposobie generowania wektorów różnicowych $\mathbf{d}_i^{(t)}$ w CMA-DE oraz DES jest rozkład prawdopodobieństwa wylosowania pary punktów z przeszłych populacji, przeszłych historycznie punktów środkowych $\mathbf{m}^{(t)}$ oraz wektorów przesunięć punktów środkowych $\mathbf{\Delta}^{(t)}$. Szereg transformacji (3.2) – (3.12) może być więc analogicznie zastosowany, gdy zmienione zostaną składniki sumy z $\alpha c_{cov}(1 - c_{cov})^{t-\tau}$ do $1/H$ oraz oznaczenie stałej będącej współczynnikiem wagowym tych składników z c_μ do c_d. W efekcie, wektory różnicowe posiadają oczekiwaną wartość równą zero oraz oczekiwaną macierz kowariancji równą:

$$\begin{aligned}\mathbf{\Sigma}\left[\mathbf{d}_i^{(t)}\right] =& \sum_{\tau_1=1}^{H} \frac{1}{H}(1 - c_d)\mathbf{p}^{(t-\tau_1)} \left(\mathbf{p}^{(t-\tau_1)}\right)^\mathsf{T} \\ &+ \sum_{\tau_2=1}^{H} \frac{1}{H} c_d \mathbf{\Delta}^{(t-\tau_2)} \left(\mathbf{\Delta}^{(t-\tau_2)}\right)^\mathsf{T} \\ &+ \sum_{\tau_3=1}^{H} \frac{1}{H} \frac{\mu - 1}{\mu} c_d \mathbf{S}\left(D_\mu^{(t-\tau_3)}\right) \\ &+ (1 - c_{cov})^t \cdot \mathbf{I} \end{aligned} \tag{3.15}$$

Po reorganizacji indeksów otrzymuje się (3.14). □

Powyższa prawidłowość oraz twierdzenie 3.1.2 dowodzi podobieństwa formuł generowania wektorów różnicowych w algorytmach CMA-DE oraz DES. Wynika z tego, że zaproponowana metoda DES modeluje proces adaptacji macierzy kowariancji analogicznie do metody CMA-ES. Podobnie więc jak CMA-ES[28], metody CMA-DE oraz DES są niewrażliwe na translacje, rotację, odbicia i skalowania funkcji celu.

3.2 Ilustracja działania algorytmów DES i CMA-DE

Twierdzenia 3.1.3 oraz 3.1.2, odpowiednio dla metod DES oraz CMA-DE, potwierdzają podobieństwo procesu modelowania macierzy kowariancji z metodą CMA-ES. W przypadku metody DES są jednak trzy różnice w stosunku do CMA-ES. Pierwszą z nich jest użycie mutacji różnicowej, która bazuje na mieszaninie jednowymiarowych rozkładów normalnych, wzdłuż wektorów przesunięć punktu środkowego populacji, a nie na wielowymiarowym rozkładzie normalnym (jak dzieje się to w CMA-ES). Takie połączenie rozkładów oraz dodanie addytywnego standaryzowanego normalnego szumu wielowymiarowego sprawia, że ostateczny rozkład generowanych punktów metody DES jest wprawdzie ciągły, jednak realizowany inaczej niż w bazowym algorytmie CMA-ES. Kolejnymi różnicami jest ewaluacja punktu środkowego w każdej generacji oraz brak adaptacji długości kroku.

Algorytm CMA-ES został scharakteryzowany przez swoich autorów jako efektywna metoda optymalizacji lokalnej, która posiada również dobre właściwości podczas optymalizacji globalnej. Poniższe podrozdziały mają zilustrować działanie zaproponowanych metod CMA-DE oraz DES, w kontekście prawidłowości odwzorowania cech ewolucyjnej strategii adaptacji macierzy kowariancji, oraz w kontekście skuteczności działania na problemach optymalizacji lokalnej.

3.2.1 Dynamika zmian wartości własnych macierzy kowariancji generowanych punktów

Jednym ze sposobów badania jakości dopasowywania się metody do poziomic funkcji celu w optymalizacji lokalnej, jest obserwacja dynamiki zmian wartości własnych empirycznych macierzy kowariancji w kolejnych generacjach. Autorzy metody CMA-ES użyli tej techniki ([17][Rozdział 5]) do wykazania poprawności adaptacji macierzy kowariancji. W poniższym rozdziale badanie to zostanie powtórzone dla metod CMA-ES, CMA-DE oraz

DES, w celu wykazania podobieństwa, co do parametrów, rozkładów punktów generowanych przez te algorytmy.

Eksperyment ten polega na empirycznym wykazaniu zmiany początkowej macierzy kowariancji $\mathbf{C}^{(0)}$ do postaci zgodnej z poziomicami funkcji celu zadanej, wypukłej funkcji kwadratowej (o dodatnio określonej macierzy Hessego $\mathbf{H}$) postaci:

$$q(x) = x^T \mathbf{H} x$$
$$q(x) = q_4(x) = \sum_{i=1}^{n} 10^{6 \cdot \frac{i-1}{n-1}} \cdot (x_i)^2$$

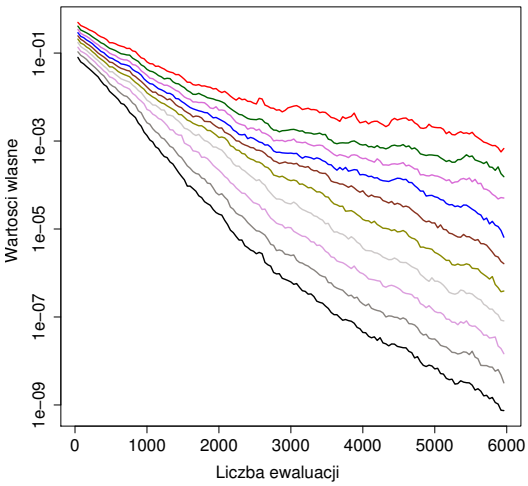
Aby warunek ten mógł być spełniony macierz kowariancji w jednej z kolejnych generacji musi stać się wprost proporcjonalna do odwrotności hesjanu funkcji $q_4(x)$ (tabela 3.1):

$$\mathbf{C} \propto \mathbf{H}^{-1}$$

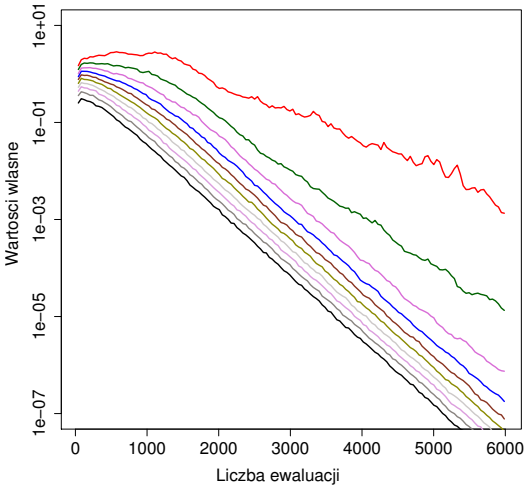
Populacją startową dla każdego testowanego algorytmu jest populacja punktów wygenerowanych zgodnie z rozkładem jednostajnym w przedziale $[0.1, 0.3]^n$. Macierz kowariancji rozkładu generującego populację startową jest macierzą diagonalną $\mathbf{C}^{(0)} = diag(0.00(3))$. Poprzez śledzenie wartości własnych empirycznych macierzy kowariancji kolejnych populacji generowanych przez badane algorytmy można stwierdzić, czy i kiedy osiągną one wymagane wartości, a więc czy metoda skutecznie realizuje proces adaptacji do poziomic funkcji celu. Z uwagi na to, że wartościami własnymi każdej macierzy diagonalnej są wszystkie współczynniki leżące na jej głównej przekątnej wynika, że dla $\mathbf{C}^{(0)}$ wszystkie n wartości własnych przyjmuje wartość w przybliżeniu 0. Poziomy te stanową punkt wyjściowy obserwacji, w którym punktem, do którego wykresy powinny dążyć są wartości własne proporcjonalne do tych wynikających z funkcji $q_4(x)$, a więc równych $\left\{10^{-6 \cdot \frac{i-1}{n-1}} \mid i = 1, \ldots, n\right\}$.

Rysunek 3.3 przedstawia dynamikę zmian wartości własnych macierzy kowariancji empirycznej, w funkcji liczby ewaluacji funkcji celu określonej jako $q_4(x)$ (tabela 3.1). Wykres został stworzony dla 10 wymiarowej przestrzeni, $n = 10$. W każdej kolejnej populacji, począwszy od startowej, estymowana jest empiryczna macierz kowariancji na podstawie wygenerowanych punktów. Różne kolory linii ukazują dynamikę zmian dziesięciu wartości własnych. Wykres przedstawia uśrednione wartości na podstawie pięćdziesięciu niezależnych uruchomień algorytmu.

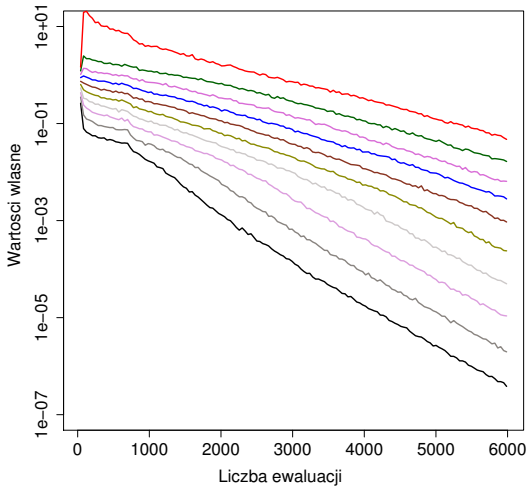
Gdy cecha dopasowywania się linii jednakowego poziomu gęstości rozkładu do poziomic funkcji celu zachodzi, wartości własne dla funkcji $q_4(x)$ stają się proporcjonalne do $\left\{10^{-6 \cdot \frac{i-1}{n-1}} \mid i = 1, \ldots, n\right\}$, a więc proporcje kolejnych wartości własnych są zachowane. Na



(a) CMA-ES



(b) CMA-DE



(c) DES

Rysunek 3.3: Wykres dynamiki zmian wartości własnych empirycznej macierzy kowariancji dla funkcji q_4 i wymiarowości $n = 10$. Każda linia przedstawia przebieg zmian pierwiastka odpowiednio jednej z n wartości własnej.

rysunku 3.3 widać, że przypadku algorytmu DES, zbiór wartości własnych szybko staje się w przybliżeniu jednorodny (odległości pomiędzy kolejnymi wartościami własnymi są podobne), dokładnie tak jak ma to miejsce w przypadku CMA-ES. Metoda CMA-DE generuje nowe punkty, tak że jej macierz kowariancji jest niejednorodna, np. proporcja między pierwszą a drugą wartością własną jest wiele razy większa niż między dziewiątą a dziesiątą. CMA-DE nie ujawnia w pełni właściwości dopasowywania do poziomic funkcji

celu i ma tendencje do podejmowania zbyt dużych kroków w niektórych kierunkach przestrzeni przeszukiwań. Ten efekt może być niepożądanym skutkiem geometrycznego rozkładu prawdopodobieństwa użytego do wyboru historycznych wektorów przesunięć punktu środkowego (Rysunek 3.1, wiersze 10,12-13).

Podobieństwa wykresu dynamiki zmian wartości własnych empirycznych macierzy kowariancji kolejnych generacji, dla algorytmów DES i CMA-ES, dowodzi posiadania przez te metody cechy dopasowywania się macierzy kowariancji generowanych punktów do odwrotności hesjanu funkcji celu. W przedstawionej w niniejszym rozdziale metodzie DES, linie gęstości prawdopodobieństwa rozkładu generowanych punktów są więc elipsoidami o podobnym kształcie co poziomice funkcji celu, co potwierdza twierdzenia i własności przytoczone w podrozdziale 3.1.2.

3.2.2 Dynamika zmian wartości funkcji celu generowanych punktów

Algorytm CMA-ES został scharakteryzowany przez swoich autorów m.in jako efektywna metoda optymalizacji lokalnej. W celu wykazania tej tezy dokonali oni testów zbieżności tej metody na grupie przygotowanych funkcji ([26][Rozdział 3], [6][Rozdział 4]). W poniższym rozdziale badanie to zostanie powtórzone dla algorytmów CMA-ES(w dwóch wariantach), DE, CMA-DE oraz DES, w celu ukazania podobieństw lub różnic w dynamice zmian wartości funkcji celu punktów generowanych przez te metody. Tabela 3.1 przedstawia jedno-modalne funkcje celu, które zostaną użyte w eksperymencie. Wartość f_{stop} jest poziomem bazowym, którego osiągnięcie dla konkretnej funkcji będzie tożsame z osiągnięciem poziomu optimum dla zadanego wymiaru.

Każda z zaproponowanych funkcji ma na celu zilustrować działanie konkretnej właściwości metod optymalizacji lokalnej. Dynamika zmian wartości najlepszego punktu na dobrze uwarunkowanej funkcji kwadratowej $q_1(\mathbf{x})$ pokazuje, jak szybko algorytm adaptuje długość kroku. Obserwacja tempa zbieżności na funkcjach źle uwarunkowanych (np. $q_2(\mathbf{x})$, $q_3(\mathbf{x})$, $q_4(\mathbf{x})$) pozwala zweryfikować zdolność algorytmu do adaptacji kroku dla każdej wymiarowości z osobna. Dla funkcji *sharp ridge* oraz *parabolic ridge* ($q_6(\mathbf{x})$ oraz $q_7(\mathbf{x})$) krok wzdłuż pierwszej współrzędnej powinien rosnąć szybko, aby maksymalizować tempo optymalizacji. W tym samym czasie krok algorytmu na pozostałych współrzędnych powinien redukować zasięg, aby zwiększać precyzję. Funkcja $q_5(\mathbf{x})$ pozwala testować różne poziomy

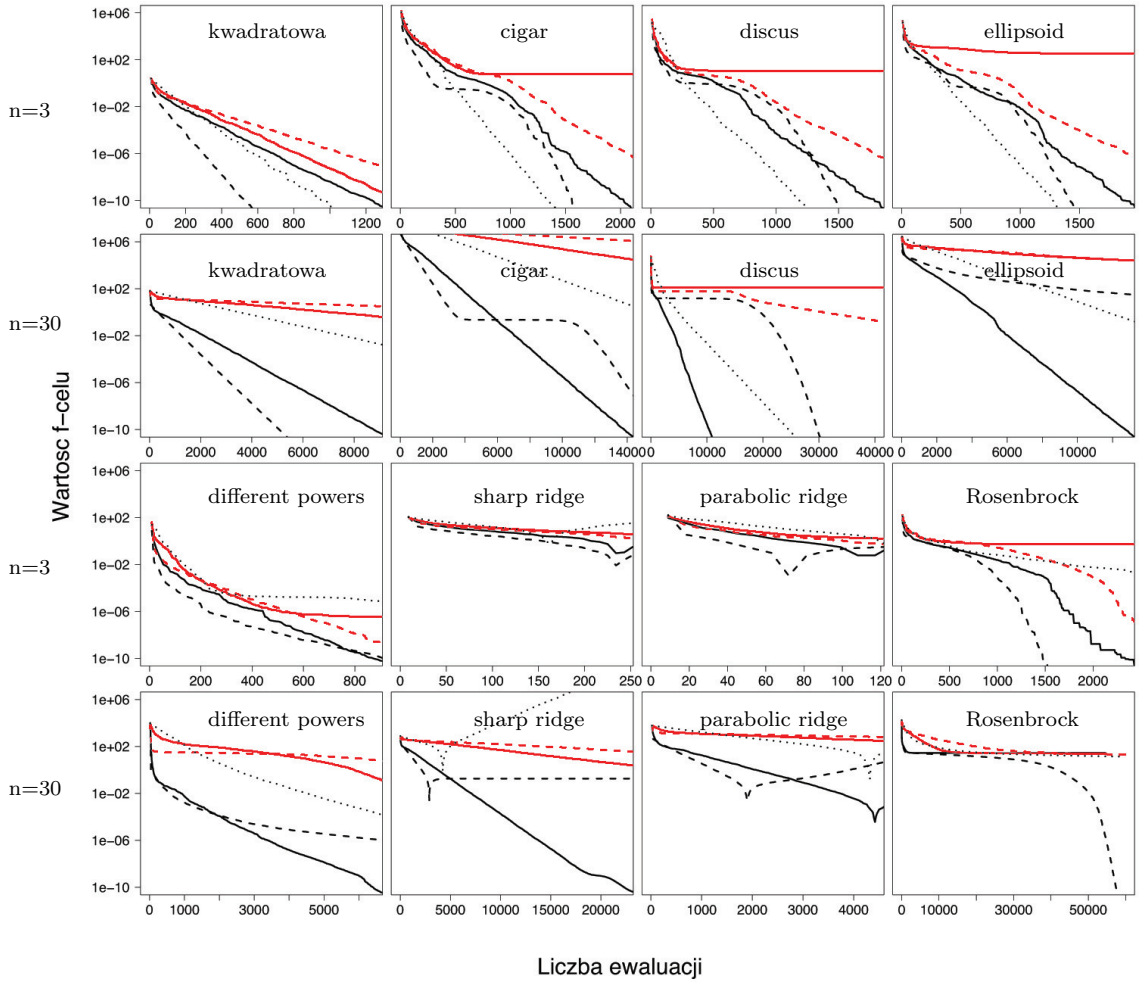
Funkcja	f_{stop}	Nazwa
$q_1(\mathbf{x}) = \sum_{i=1}^{n}(x_i)^2$	10^{-10}	kwadratowa
$q_2(\mathbf{x}) = (x_1)^2 + 10^6 \sum_{i=2}^{n}(x_i)^2$	10^{-10}	cigar
$q_3(\mathbf{x}) = 10^6(x_1)^2 + \sum_{i=2}^{n}(x_i)^2$	10^{-10}	discus
$q_4(\mathbf{x}) = \sum_{i=1}^{n} 10^{6 \cdot \frac{i-1}{n-1}} \cdot (x_i)^2$	10^{-10}	ellipsoid
$q_5(\mathbf{x}) = \sum_{i=1}^{n}(x_i)^{2(1+5\frac{i-1}{n-1})}$	10^{-10}	different powers
$q_6(\mathbf{x}) = x_1 + 100 \sum_{i=2}^{n}(x_i)^2$	-10^{10}	sharp ridge
$q_7(\mathbf{x}) = x_1 + 100\sqrt{\sum_{i=2}^{n}(x_i)^2}$	-10^{10}	parabolic ridge
$q_8(\mathbf{x}) = \sum_{i=1}^{n-1}\left(100(x_i^2 - x_{i+1}^2) + (x_i - 1)^2\right)$	10^{-10}	Rosenbrock

Tabela 3.1: Proste jednomodalne funkcje celu użyte w eksperymencie.

dynamiki w zależności od kierunku optymalizacji. Funkcja Rosenbrock'a $q_8(\mathbf{x})$ pozwala zbadać dynamikę zbieżności do optimum, które jest zlokalizowane w wąskiej i dodatkowo zagiętej dolinie.

Poniżej przestawione zostały wyniki badań na funkcjach opisanych w tabeli 3.1, które zostały użyte do analizy dynamiki zbieżności dla algorytmów CMA-ES, CMA-DE oraz DES. Z uwagi na to, że zarówno CMA-DE jak i DES nie są wyposażone w mechanizm mnożnika długości kroku, przeprowadzone zostały badania dla algorytmu CMA-ES zarówno z włączonym, jaki i wyłączonym mnożnikiem kroku (stała wartość $\sigma = 1$). Dla porównania, na wykresy naniesione zostały także wyniki eksperymentu przeprowadzonego dla ewolucji różnicowej w wersji DE/rand/1/bin. Dokładny opis parametrów użytych dla każdego z algorytmów opisany został w artykule [2][Sekcja *Appendix*] oraz w dodatku A do rozprawy.

Rysunek 3.4 ukazuje wykresy zbieżności algorytmów dla wymiarowości $n = 3$ oraz $n = 30$, przeprowadzone dla każdej funkcji celu z tabeli 3.1. Dla każdego uruchomienia śledzone były wartości najlepszego jak dotąd znalezionego punktu. Wykresy prezentują uśrednione wartości z 50 niezależnych uruchomień dla każdego algorytmu i każdego problemu. Z uwagi na to, że w przypadku funkcji *sharp ridge* oraz *parabolic ridge* ($q_6(\mathbf{x})$ oraz $q_7(\mathbf{x})$) wartości



Rysunek 3.4: Zbieżność wartości bezwzględnych najlepszego dotychczas znalezionego rozwiązania dla funkcji: DES (linia czarna ciągła), CMA-ES (linia czarna przerywana), CMA-DE (linia czerwona ciągła), CMA-ES bez adaptacji σ (linia czerwona przerywana) oraz DE/rand/1/bin (linia kropkowana)

funkcji mogą być ujemne, na wykresie raportowane są wartości bezwzględne funkcji celu.

Dla funkcji kwadratowej $q_1(\mathbf{x})$ zarówno CMA-DE, DES, jak i pozostałe algorytmy posiadają liniową zbieżność. Najszybsza dynamika zmian może być zauważona w przypadku CMA-ES, a najwolniejsza dla CMA-ES pozbawionego adaptacji mnożnika kroku. Fakt ten pokazuje istotność procesu adaptacji wartości σ w metodzie CMA-ES, wpływającej na jej ostateczną efektywność.

Metoda DES charakteryzuje się zbieżnością liniową, bądź w przybliżeniu liniową, również dla funkcji źle uwarunkowanych $q_2(\mathbf{x})$, $q_3(\mathbf{x})$, $q_4(\mathbf{x})$ (*cigar*, *discus* oraz *elipsoid*), podobnie jak algorytm DE/rand/1/bin. Wyniki metody CMA-ES, zarówno w wersji klasycznej, jak i bez adaptacji σ, ukazują chwilowe momenty stagnacji zbieżności dla problemów

źle uwarunkowanych. Zjawisko to jest szczególnie widoczne dla wyższej wymiarowości $n = 30$. W przypadku metody CMA-DE widoczne jest znacząco niższe tempo zbieżności niż w DES. Dla problemów w wymiarowości $n = 30$ widoczne są dla tej metody momenty spadku dynamiki, a nawet całkowitej stagnacji, które nie występują w przypadku DES.

Algorytm DES cechuje się praktycznie stałym tempem zbieżności również dla problemu $q_5(\mathbf{x})$, dla którego poziomice funkcji celu nie są elipsoidami. W przypadku metody CMA-ES dynamika zmian zmniejsza się, wraz ze zbliżaniem się do optimum. Ewolucja różnicowa zachowuje się podobnie jak DES, zachowując w przybliżeniu stały poziom zbieżności. Zarówno CMA-ES bez adaptacji σ, jak i CMA-DE, posiadają niską dynamikę zmian dla funkcji $q_5(\mathbf{x})$ w stosunku do DES.

Dla problemów *sharp* oraz *parabolic ridge* ($q_6(\mathbf{x})$ i $q_7(\mathbf{x})$) proces optymalizacji przebiega wolniej dla metody DES niż dla CMA-ES. Zmiany monotoniczności widoczne na wykresach tych funkcji są związane z tym, że wartości funkcji celu stają się w pewnym momencie ujemne, a na wykresie raportowane są wartości bezwzględne. W przypadku algorytmu z rodziny DE proces optymalizacji początkowo jest wolniejszy niż w CMA-ES oraz DES ale wraz ze wzrostem liczy ewaluacji znacząco przyspiesza, finalnie przekraczając dynamiką DES oraz CMA-ES. Zjawisko to jest w szczególności widoczne dla funkcji *sharp ridge* w wymiarowości $n = 30$. W przypadku CMA-DE, jak i CMA-ES bez adaptacji σ, proces optymalizacji przebiega bardzo wolno dla obu funkcji.

Zarówno DE, CMA-DE, jak i CMA-ES bez adaptacji σ, nie zdołały rozwiązać problemu Rosenbrocka ($q_8(\mathbf{x})$), dla obu wymiarowości. Poziom zbieżności metody DES dla $n = 3$ jest nieco niższy niż w CMA-ES, a w przypadku wyższej wymiarowości DES nie znalazł optimum w zadanym, maksymalnym budżecie liczby ewaluacji.

Rozdział 4

Różnicowa strategia ewolucyjna na tle innych metod optymalizacji

Poprzedni rozdział wykazał wysoką skuteczność metody DES w zadaniach optymalizacji lokalnej, zarówno na funkcji kwadratowej, jak i na grupie problemów unimodalnych. Poniższe podrozdziały mają na celu weryfikację skuteczności różnicowej strategii ewolucyjnej w procesie optymalizacji globalnej. W tym celu zaproponowana przez autora metoda zostanie skonfrontowana z wiodącymi algorytmami, biorącymi udział w popularnych konkursach optymalizacyjnych. Dla każdego z wybranych benchmarków optymalizacyjnych przedstawione i omówione zostaną wyniki rywalizacji metody DES z najlepszymi algorytmami, wygrywającymi poszczególne konkursy optymalizacyjne.

4.1 Rodziny benchmarków i sposoby oceny

Najpopularniejszymi konkursami optymalizacji ciągłej są CEC oraz BBOB, które zbudowane są m.in z szeregu jednokryterialnych problemów optymalizacyjnych z ograniczeniami kostkowymi. Dla każdej funkcji i zadanej wcześniej wymiarowości, testowany algorytm powinien być uruchomiony wielokrotnie w niezależny sposób. Określona jest maksymalna liczba ewaluacji funkcji celu (MaxFEs), którą metoda może wykorzystać w każdym uruchomieniu do znalezienia optimum.

Benchmarki CEC oraz BBOB udostępniane są w formie zewnętrznego oprogramowania lub biblioteki, a więc uruchamianie wszystkich zaimplementowanych funkcji wielokrotnie, w celu jak najlepszego dopasowania parametrów metody do optymalizowanych

funkcji, jest możliwe. Oznacza to, że benchmarki te niedoskonale implementują prawdziwy problem czarnej skrzynki (*ang.Black box*), w którym algorytm przed przystąpieniem do optymalizacji nie posiada żadnych informacji o rozwiązywanej funkcji oraz w którym istnieją ograniczenia na ilość ewaluacji funkcji celu. Przykładem konkursu optymalizacyjnego, który lepiej oddaję specyfikę benchmarków czarnej skrzynki, może być konkurs BBComp.

Skuteczność rożnicowej strategii ewolucyjnej jako optymalizatora globalnego, zdolnego do konkurowania z wiodącymi metodami rozwiązującymi problemy wielomodalne, została potwierdzona w tym rozdziale w oparciu o benchmarki CEC, BBOB oraz BBComp. Poniższe podrozdziały charakteryzują odpowiednio każdy z nich oraz opisują metodę oceny wyników.

4.1.1 Benchmark CEC

Benchmark CEC jest rozwijany od 2005 roku, w ramach corocznej konferencji IEEE CEC (*ang.Congress on Evolutionary Computation*). Użyta w ramach tego rozdziału wersja[5], powstała na potrzeby konkursu optymalizacyjnego organizowanego w ramach konferencji CEC 2017. Zbiór, dla jednokryterialnych problemów o ograniczeniach kostkowych, składa się z 30 zaimplementowanych funkcji. Problemy zostały podzielone na grupy problemów będącymi w zależności od numeru funkcji odpowiednio:

(1-3) Problemy unimodalne, które charakteryzują się zróżnicowaną dynamiką zmian wartości funkcji celu, w zależności od kierunku

(4-10) Proste problemy wielomodalne zawierające funkcje celu z regularnymi wzorami geometrycznymi lokalnych optimów.

(11-20) Problemy hybrydowe, które są liniową kombinacją funkcji wielomodalnych użytych na podzbiorze zmiennych, gdzie wzorce zmian są różne dla różnych grup zmiennych.

(21-30) Problemy złożone będące liniowymi kombinacjami funkcji wielomodalnych zastosowanych dla całego zestawu zmiennych. Wzorce zmian są charakterystyczne dla każdej funkcji i wykluczają się wzajemnie.

Wymiarowość przestrzeni przeszukiwań wynosi $n = 10, 30, 50, 100$, dla każdej z 30 optymalizowanych funkcji. Maksymalna liczba ewaluacji funkcji celu w każdym uruchomieniu wynosi: MaxFEs $= 10000 \cdot n$. Problemy konkursowe posiadają ograniczone dziedziny, określone dla każdego z n wymiarów w przedziale $x_i \in [-100, 100]$. Liczba wymaganych niezależnych uruchomień, dla każdej pary funkcja i wymiarowość, wynosi 51. Dla każdego problemu optymalizacyjnego znana jest dokładna wartość optimum funkcji celu. Podczas analizy wyników brany jest pod uwagę błąd, będący różnicą między wartością funkcji celu rozwiązania oraz wartością optymalną. Gdy raportowana wartość błędu jest mniejsza niż 10^{-8} przyjmuje się, że osiągnięto punkt optymalny.

Benchmark CEC dostarczany jest w formie biblioteki implementującej konkursowe funkcje. Proces zbierania danych oraz wykreślania wykresów odbywa się poza benchmarkiem i jest przeprowadzany we własnym zakresie. W celu porównywania się z innymi algorytmami należy więc dokonać samodzielnie ich testów, bądź bazować na danych udostępnianych przez autorów tych metod lub twórców konkursu.

4.1.2 Benchmark BBOB

Benchmark BBOB (*ang.Black-Box Optimization Benchmarking*) jest rozwijany razem z oprogramowaniem COCO[21] (*ang.Comparing Continuous Optimizers*) i od 2009 roku jest używany w ramach konferencji GECCO (*ang.Genetic and Evolutionary Computation Conference*). Zbiór składa się z 24 funkcji. Problemy zostały podzielone na grupy problemów będącymi w zależności od numeru funkcji odpowiednio:

(1-5) Problemy separowalne, z funkcjami celu, w których optymalizację można przeprowadzić oddzielnie dla każdej zmiennej.

(6-9) Funkcje, które charakteryzują się przeciętnie lub dobrze uwarunkowanymi macierzami kowariancji.

(10-14) Problemy unimodalne, źle uwarunkowane, m.in takie jak *ellipsoid*, *discus*, *cigar*, *sharp ridge* and *different powers*. Funkcje z tej grupy w dużej mierze odpowiadają problemom unimodalnym (1-3) z benchmarku CEC2017.

(15-19) Problemy wielomodalne o strukturze globalnej, podobnie jak problemy złożone z benchmarku CEC2017.

(20-24) Funkcje wielomodalne, których struktura globalna jest nieregularna i zaszumiona.

Wymiarowość przestrzeni przeszukiwań wynosi $n = 2, 3, 5, 10, 20, 40$, dla każdej z 24 optymalizowanych funkcji. Maksymalna liczba ewaluacji funkcji celu w każdym uruchomieniu wynosi: MaxFEs $= 10000 \cdot n$. Problemy konkursowe posiadają ograniczenia i są określone dla każdego z n wymiarów w przedziale $x_i \in [-5, 5]$. Dodatkową informacją jest fakt, że optimum dla większości funkcji znajduje się w przedziale $x_i \in [-4, 4]$. Liczba wymaganych niezależnych uruchomień dla każdej funkcji, w zadanej wymiarowości, wynosi 15. Dla każdego problemu optymalizacyjnego znana jest dokładna wartość optimum funkcji celu. Podczas analizy wyników brany jest pod uwagę błąd, będący różnicą między wartością funkcji celu rozwiązania oraz wartością optymalną. Gdy raportowana wartość błędu jest mniejsza niż 10^{-8}, przyjmuje się, że osiągnięto punkt optymalny.

Benchmark BBOB dostarczany jest wraz z oprogramowaniem COCO, które kontroluje cały proces testowania metody oraz samodzielnie generuje wykresy i podsumowuje wyniki. Proces ten jest automatyczny, samoczynnie porównujący wyniki badanej metody z innymi, wiodącymi dla tego benchmarku, algorytmami. Dane z uruchomień innych algorytmów, potrzebne do generowania wykresów i oceny, dostarczane są wraz z oprogramowaniem przez autorów benchmarku.

4.1.3 Benchmark BBComp

Benchmark BBComp[39] (*ang.Black Box Optimization Competition*) był rozwijany pomiędzy rokiem 2015 a 2019 przez Uniwersytet Ruhry w Bochum. Jest konkursem optymalizacyjnym, który zdecydowanie lepiej niż konkursy BBOB oraz CEC oddaje ideę konkursu czarnej skrzynki (*ang. Black box optimization*). Użyta w ramach tego rozdziału ścieżka benchmarku nosi nazwę *BBComp2016-1OBJ* i jest grupą jednokryterialnych problemów o ograniczeniach kostkowych. Zbiór składa się z 1000 problemów o zupełnie nieznanej charakterystyce. Wymiarowość przestrzeni przeszukiwań wynosi $n = 2, 4, 5, 8, 10, 16, 20, 32, 40, 64$, gdzie dla każdego z wymiarów zdefiniowane jest 100 problemów. Maksymalna liczba ewaluacji funkcji celu w każdym uruchomieniu wynosi: MaxFEs $= 100 \cdot n^2$. Dziedziny funkcji są ograniczone i określone dla każdego z n wymiarów w przedziale $x_i \in [0, 1]$. Poza tym przedziałem wartość funkcji jest nieokreślona. Wartość funkcji celu punktu optymalnego jest nieznana, niemożliwe jest więc samodzielne oblicze-

nie błędu.

Benchmark BBComp zbudowany jest w architekturze klient-serwer. Dostarczone przez autorów oprogramowanie pozwala na łączenie się z serwerem benchmarku z poziomu różnych języków programowania i wysyłanie zapytań o wartość funkcji celu w danym punkcie. Funkcje konkursowe są więc dla uczestników rzeczywiście nieznane. Nie są dostarczane ani jawnie, ani pośrednio w postaci skompilowanej w bibliotece. Każdy zarejestrowany i zweryfikowany przez organizatorów użytkownik może do każdej ścieżki benchmarku, a więc do każdego problemu, podejść tylko raz. Nie ma więc możliwości, aby poprzez wielokrotne uruchamianie problemów konkursowych zmienić lub dostosować parametry testowanego algorytmu. Archiwalne wyniki porównujące skuteczność metod są udostępniane przez autorów benchmarku na ich stronie.

4.1.4 Wizualizacja wyników poprzez krzywe ECDF

W celu wizualnego przedstawienia wyników rankingu skuteczności metod w rozwiązywania problemów optymalizacyjnych można stosować krzywe zwane ECDF[21] (*ang. Empirical Cumulative Distribution Function*). Krzywe ECDF starają się opisywać przeciętny postęp metody optymalizacji w funkcji zużytego budżetu ewaluacji funkcji celu. Postęp ten jest wyrażany za pomocą arbitralnie wprowadzonej skali, pozwalającej na porównywanie i uśrednianie wyników dla różnych zadań optymalizacji.

Kluczowym elementem podczas tworzenia wykresu ECDF jest określenie progów jakości. W celu określenia zbioru progów jakości należy dla każdego problemu i wymiarowości zdefiniować najniższą q_l oraz najwyższą q_h wartość funkcji celu kiedykolwiek występującą w rezultatach którejkolwiek z metod mających wchodzić w skład wykresu. Próg uznaje się za osiągnięty, jeśli jakość najlepszego punktu osiągnie lub przekroczy wartość określoną przez próg. Zwyczajowo[22], progi jakości takiego wykresu ECDF zawierają się w przedziale $[q_l, q_h]$ i iloraz dwóch kolejnych progów ma wartość $10^{0.2}$.

Aby było możliwe wykreślenie krzywej należy dla każdego uruchomienia monitorować wartość funkcji celu najlepszego jak dotąd znalezionego punktu i zapisywać poziom jego jakości, po osiągnięciu zdefiniowanego procenta wartości MaxFEs. Do powstania krzywej ECDF wymagany jest zbiór par, przyporządkowujący określone procenty MaxFEs i wartość funkcji celu najlepszego jak dotąd znalezionego punktu. Posiadając takie dane dla każdej trójki: wymiar, problem i numer uruchomienia, można w efektywny sposób po-

znać jakość badanej metody na tle innych, bez konieczności długotrwałego uruchamiania algorytmów, z którymi następuje porównanie.

Krzywe ECDF ukazują proporcje ułamka osiągniętych progów w zadanym budżecie, gdzie budżet jest przedstawiony na osi odciętych (oś x). Oś rzędnych (oś y) ukazuje proporcję osiągniętych odpowiednich progów jakości określoną w przedziale $y \in [0, 1]$. Wykres ECDF ukazuje w przybliżony sposób przeciętną jakość wyników, jakie osiąga metoda po wykorzystaniu konkretnej liczby ewaluacji funkcji celu na zadanej funkcji, grupie funkcji, bądź całym benchmarku.

W przypadku wielu niezależnych uruchomień tej samej metody dla tego samego problemu i wymiarowości, wartości proporcji osiągniętych progów jakości (oś y) są uśredniane osobno w każdym punkcie budżetu, będącego kolejnym procentem MaxFEs (oś x). Dla każdego punktu x, wartość y stanowi procent osiągniętych progów, a więc krzywa ECDF jest niemalejąca. Istnieje możliwość agregacji wyników wielu różnych problemów, grup problemów, lub przedstawienia wyników rankingu skuteczności metod w ujęciu całego benchmarku. W tym celu wyniki dla wielu problemów są uśredniane dla każdego procenta MaxFEs z osobna.

Każdy punkt na krzywych ECDF dla zadanego budżetu (oś x) ukazuje proporcję osiągniętych progów jakości (oś y). Pole powierzchni pod każdą krzywą ECDF może służyć do porównywania jakości pomiędzy algorytmami. Jeśli przyjąć, że preferujemy metody jak najszybciej dające jak najlepsze rozwiązania, wówczas metoda jest tym lepsza w optymalizacji, im pole powierzchni pod jej wykresem ECDF jest większe.

4.1.5 Sposób uwzględniania ograniczeń kostkowych

Wszystkie opisane w tym rozdziale rodziny benchmarków zawierają problemy o ograniczeniach kostkowych. Zbiór punktów dopuszczalnych $\mathbb{F}$ opisany jest w postaci:

$$\mathbb{F} = \{\mathbf{x} : l_i \leqslant x_i \leqslant u_i, \quad i = 1, \ldots, n\},$$

gdzie l_i oznacza dolne, a u_i górne ograniczenie na i-tej współrzędnej. Rozwiązanie musi znajdować się w zbiorze dopuszczalnym $\mathbb{F}$, a poza nim każda z funkcji benchmarku może być niezdefiniowana. Opisywana w tej pracy metoda DES posiada szereg mechanizmów stochastycznych, takich jak np. losowy addytywny szum wielowymiarowy. Pojawienie się punktu spoza zbioru $\mathbb{F}$ jest więc możliwe, nawet pomimo wygenerowania populacji starto-

wej wewnątrz ograniczenia. Istnieje wiele metod radzenia sobie z tą sytuacją. Przykładowo można posłużyć się metodą naprawy, która każdemu punktowi niedopuszczalnemu przyporządkowuje, zgodnie z pewną regułą, punkt ze zbioru $\mathbb{F}$. Wybór metody uwzględniania ograniczeń może zakłócać działanie metody optymalizacyjnej, dlatego wybór ten jest kluczową decyzją podczas rozwiązywania problemów optymalizacyjnych.

Istnieją badania pokazujące wpływ sposobu uwzględniania ograniczeń na skuteczność optymalizacji problemów benchmarkowych. W artykule [8] przeprowadzona została eksperymentalna analiza na 7 różnych algorytmach, w których przebadano 17 sposobów uwzględniania ograniczeń i scharakteryzowano ich wydajność w procesach eksploracji i eksploatacji. Wnioski płynące z tego badania dowodzą znaczącego wpływu metody uwzględniającej ograniczenia na jakość wyników optymalizacji problemów z ograniczeniami kostkowymi. Analiza porównawcza zawarta w przytoczonym artykule pozwoliła podjąć decyzję o wyborze optymalnego sposobu uwzględniania ograniczeń w algorytmie DES opisanego poniżej.

W algorytmie DES jakość punktów wewnątrz zbioru dopuszczalnego nie jest modyfikowana. Każdemu punktowi, u którego którakolwiek współrzędna znajduje się poza ograniczeniami, przypisywana jest wartość sztucznej funkcji celu $q_A(x)$, której wartość jest kombinacją wartości najgorszego jak dotąd znalezionego rozwiązania $q_{\max}$ oraz kwadratowej funkcji kary:

$$\begin{aligned} q_A(\mathbf{x}) =& q_{\max} + \sum_{x_j > u_j} (x_j - u_j)^2 + \sum_{x_j < l_j} (l_j - x_j)^2, \text{ gdzie} \\ q_{\max} =& \max_{t,i,\mathbf{x}_i^{(t)} \in \mathbb{F}} q\left(\mathbf{x}_i^{(t)}\right). \end{aligned}$$

Wartość sztucznej funkcji celu $q_A(x)$ rośnie kwadratowo wraz z odległością punktu niedopuszczalnego od najbliższego ograniczenia.

Sposobem na zmniejszenie potencjalnej liczby pojawiających się punktów poza zbiorem dopuszczalnym, w początkowych generacjach, jest odpowiednie zdefiniowanie populacji startowej. Jeśli populacja ta jest tworzona poprzez równomierne losowanie w całym zbiorze $\mathbb{F}$, wówczas oczekiwana długość wektorów różnicowych sprawi, że duża cześć punktów pierwszych generacji będzie generowana poza zbiorem dopuszczalnym. Dlatego wskazane jest zainicjowanie populacji na obszarze zawartym w $\mathbb{F}$. Obszar inicjacji użyty

w eksperymentach jest ograniczony przez:

$$l_i + \frac{1}{3}(u_i - l_i) \leqslant x_i \leqslant u_i - \frac{1}{3}(u_i - l_i)$$

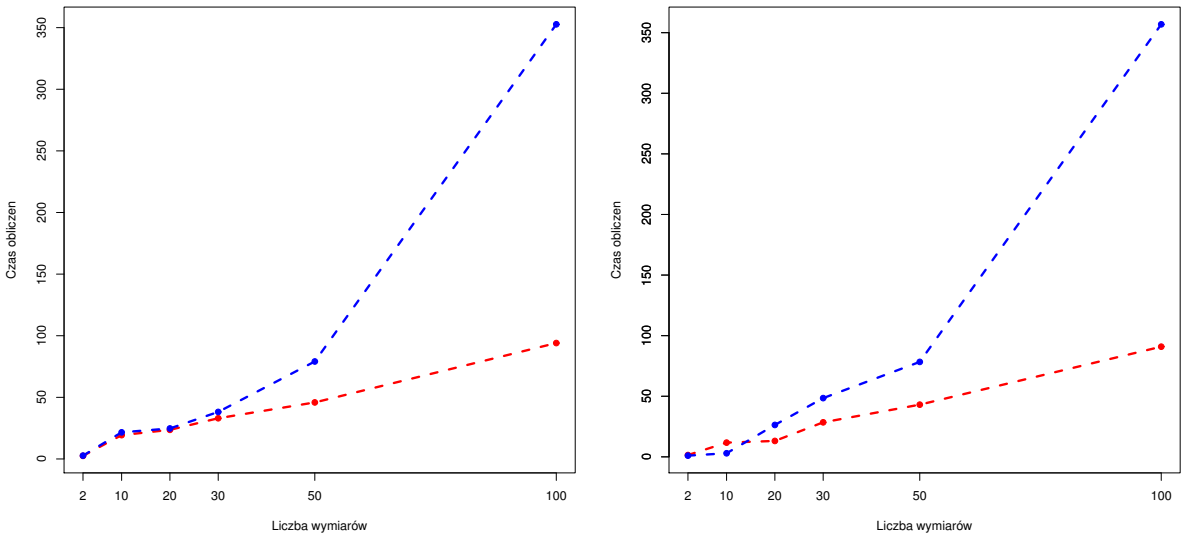
Ograniczenie obszaru $\mathbb{F}$ w celu wylosowania populacji początkowej sprawia, że zmniejsza się oczekiwana długość wektora różnicowego. Po przyłożeniu takich wektorów różnicowych do środka populacji, znajdującego się statystycznie w środku obszaru z którego następuje losowanie, ryzyko wygenerowania nowego punktu naruszającego ograniczenie już w drugiej generacji jest niskie.

4.2 Wyniki zastosowania algorytmu DES na benchmarku CEC

Poniższy rozdział prezentuje wyniki porównawcze algorytmu DES na benchmarku CEC, w wersji opisanej w podrozdziale 4.1.1. Dane algorytmów do porównań, biorących udział w konkursie, pochodzą z zasobów udostępnionych przez autorów benchmarku[5] na ich stronie[54]. W przypadku algorytmu DES oraz CMA-ES, uruchomione zostały wszystkie zadania benchmarkowe, a wyniki niezbędne do wykreślenia krzywych ECDF zostały samodzielnie zebrane.

4.2.1 Testy zależności czasowych

Z uwagi na to, że benchmark CEC jest dostarczany w formie biblioteki implementującej jasno zdefiniowane funkcje, istnieje możliwość implementacji dodatkowych testów na problemach konkursowych. Biblioteka daje możliwość uruchomienia dowolnej funkcji z benchmarku, zapewniając wygodny dostęp do ich implementacji. Możliwe jest więc przeprowadzenie porównawczych testów czasowych, porównujących wybrane algorytmy. Rysunek 4.1 obrazuje wykres zależności potrzebnego czasu obliczeń od liczby wymiarów, dla algorytmów DES oraz CMA-ES. Do wykonania pomiarów wybrane zostały dwie funkcje z benchmarku CEC: funkcja numer 1, będąca prostym problemem unimodalnym o dobrze uwarunkowanej macierzy kowariancji, oraz funkcja numer 21, będąca problemem złożonym z wielu funkcji wielomodalnych. Do pomiarów czasu wykorzystana została funkcja wykorzystująca możliwości powłoki systemowej do rzeczywistego pomiaru czasu, jaki zadany proces spędził na procesorze. Oba testowane algorytmy używały implementacji w



(a) Problem nr 1

(b) Problem nr 21

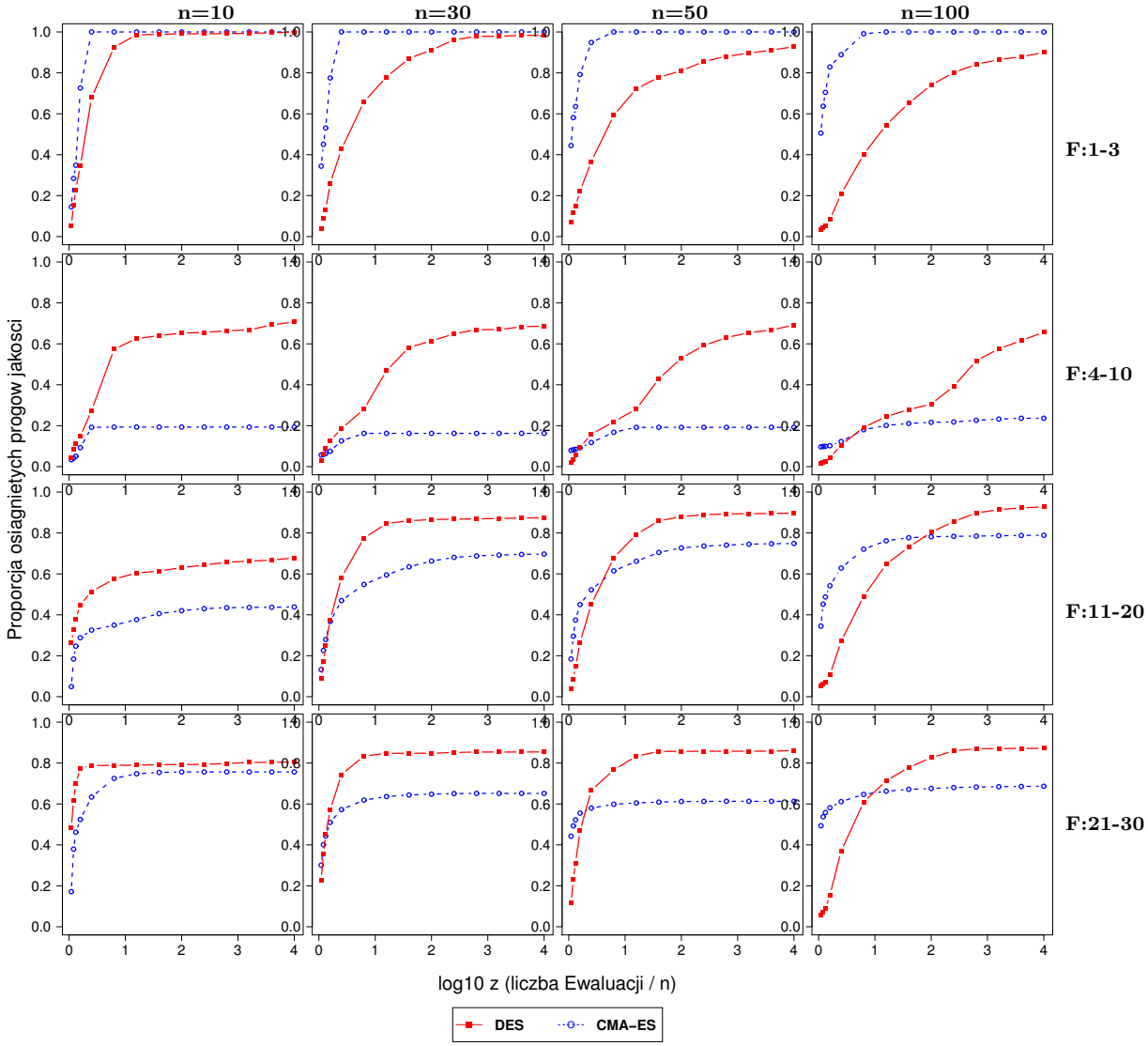
Rysunek 4.1: Wykres zależności czasu obliczeń (sekund) od liczby wymiarów (zmiennej n) dla problemów z CEC. Liczba ewaluacji funkcji celu dla każdego z algorytmów była równa $10000 \cdot n$. Linia niebieska obrazuje CMA-ES, linia czerwona DES.

tym samym języku programowania, którym był język programowania R. Każda metoda posiadała ten sam maksymalny budżet wynoszący $10000 \cdot n$ ewaluacji funkcji celu. Wszystkie metody, dla każdego wymiaru i problemu, zostały uruchomione 51 razy. Czas obliczeń raportowany na wykresie jest średnią liczbą sekund, jakie proces powłoki systemowej dla danego algorytmu zajął na procesorze, po odjęciu czasu, jaki dla tych parametrów osiągnął algorytm błądzenia przypadkowego. W tym celu, dla każdego problemu i wymiarowości, wyznaczony został czas zużyty na stworzenie oraz ewaluację wszystkich punktów z zadanego budżetu, dla metody błądzenia przypadkowego, która losowo wybierała punkty dla których obliczana była wartość funkcji celu. Zabieg ten pozwala uniezależnić wykresy zależności czasu obliczeń od różnic w implementacji i trudności obliczeń wszystkich funkcji celu.

Wyniki ukazane na wykresie 4.1 obrazują różnicę w nakładzie czasowym dla algorytmów DES oraz CMA-ES. Zarówno dla prostej funkcji unimodalnej, jak i dla funkcji złożonej, DES wykazuje liniową zależność czasu obliczeń do liczby wymiarów problemu. W przypadku zależności czasowej algorytmu CMA-ES, zauważalna jest rosnąca, wraz z liczbą wymiarów, rozbieżność w stosunku do zależności czasowej metody DES. Dla pro-

blemów powyżej 50 wymiarów widoczne jest załamanie od początkowej, w przybliżeniu liniowej zależności. Można podejrzewać, że wraz z dalszym zwiększaniem liczy wymiarów będą następowały kolejne załamania, a wykres w większej skali dla CMA-ES będzie przypominał aproksymowaną liniowo funkcję kwadratową.

4.2.2 Wyniki DES na tle CMA-ES



Rysunek 4.2: Zbiorczy wykres krzywych ECDF prezentujących wyniki algorytmów DES oraz CMA-ES na benchmarku CEC. Każdy z wykresów prezentuje wyniki porównań proporcji osiągniętych progów jakości w funkcji budżetu, dla zadanej wymiarowości (zmienna n) oraz konkretnej grupy funkcji (F).

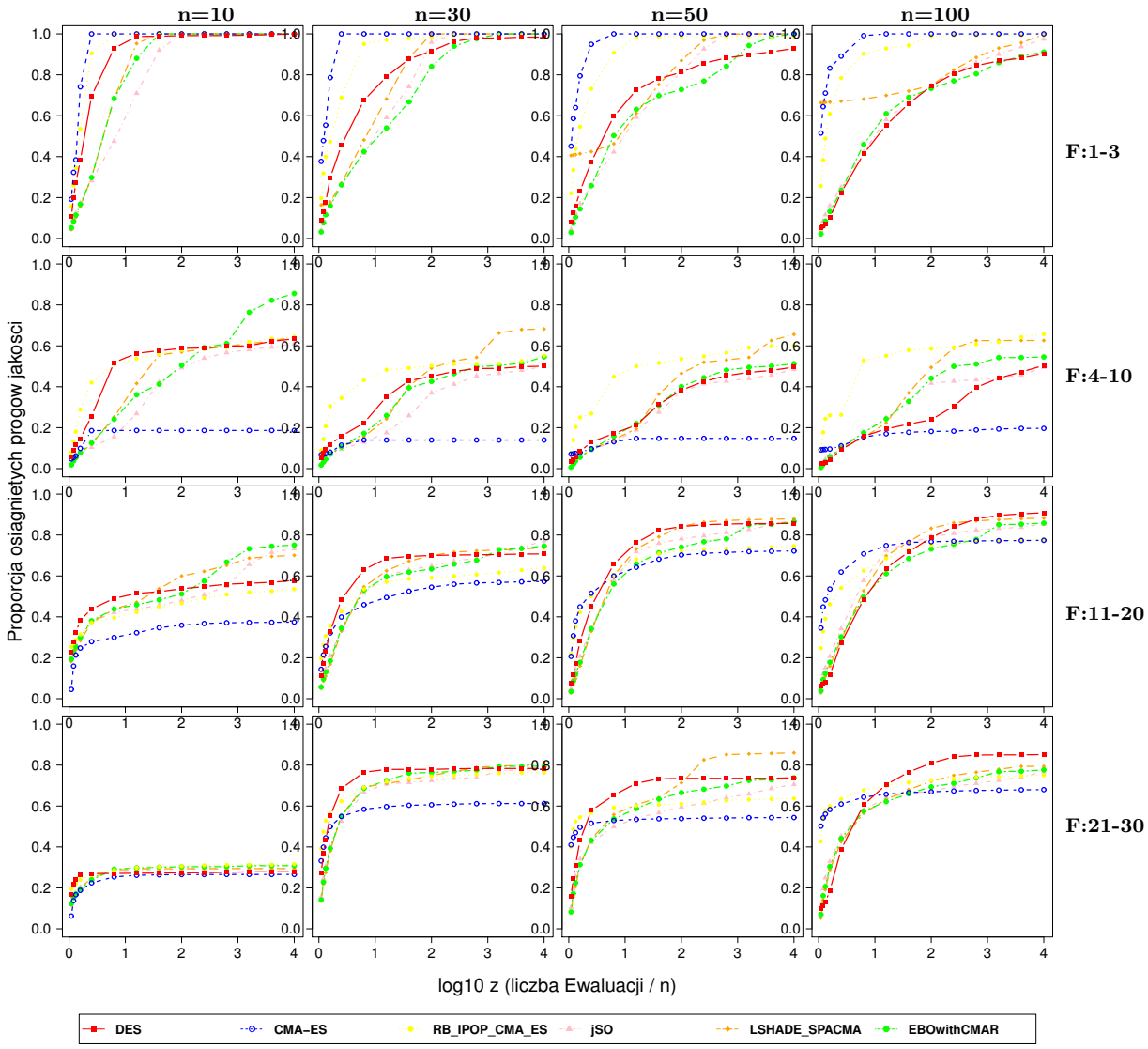
Poniższy podrozdział przedstawia wyniki benchmarku CEC, tylko w kontekście rywa-

lizacji pomiędzy metodami DES oraz CMA-ES. W tym celu, oba algorytmy, rozwiązywały wszystkie problemy konkursowe, a wymagane dane zostały zebrane. Wykreślone na rysunku 4.2 krzywe ECDF prezentują przeciętny postęp optymalizacji w funkcji zużytego budżetu ewaluacji funkcji celu, w podziale na grupy funkcji benchmarkowych oraz liczbę wymiarów. Budżet na wszystkich wykresach ECDF wyrażony jest jako liczba ewaluacji funkcji celu podzielona przez liczbę wymiarów i przedstawiony w skali logarytmicznej.

Wyniki pokazują, że dla problemów unimodalnych (1-3) metoda CMA-ES osiąga lepsze wyniki od DES, biorąc pod uwagę zarówno szybkość zbieżności, jak i precyzję w optymalnej lokalizacji. DES nie jest w stanie osiągnąć precyzji na poziomie porównywalnym z CMA-ES. Wskazuje to na konieczność poprawy efektywności eksploatacyjnej, być może związanej z dalszym rozwojem metody DES w kierunku badań nad mnożnikiem długości kroku. W przypadku problemów wielomodalnych (4-10, 11-20, 21-30) algorytm DES, we wszystkich przypadkach, osiąga znacznie lepsze wyniki niż CMA-ES, zwłaszcza dla wyższych wymiarów. Może to świadczyć o przewadze DES nad metodą CMA-ES pod kątem zdolności eksploracyjnej. Można podejrzewać, że wolniejsza zbieżność i mniejsza precyzja DES mogą być pomocne w osiąganiu dobrej jakości rozwiązań. Globalna struktura funkcji wielomodalnej może być lepiej widoczna, gdy pomija się wysoce dynamiczne artefakty poprzez użycie mniej precyzyjnej metody.

4.2.3 Wyniki DES na tle innych uczestników konkursu

W tym podrozdziale przedstawione zostały wyniki konkursu CEC organizowanego w ramach konferencji CEC'2017. W konkursie wzięło udział wzięło udział 16 algorytmów, w którym organizatorzy wymagali od uczestników uruchomienia benchmarku i dostarczenia wymaganych danych. Zgodnie z wynikami zaprezentowanymi na konferencji i zamieszczonymi na stronie organizatorów, zwycięzcą konkursu została metoda EBO[36], będąca kombinacją algorytmu optymalizacji rojowej oraz CMA-ES. Miejsca od drugiego do czwartego zajęły różne wersje algorytmu LSHADE [11, 4, 43]. Metoda DES została, w jednej z wersji raportu, sklasyfikowana na miejscu piątym. Kilka wysoko ocenionych metod obejmowało CMA-ES jako swoje części. Takim algorytmem był np. RB-IPOP-CMA-ES[7], który był ulepszeniem bazującej na CMA-ES metody IPOP-CMA-ES. Wszystkie wymienione powyżej metody były najwyżej ocenionymi metodami w konkursie w ramach konferencji CEC'2017. Na podstawie danych udostępnionych przez autorów konkursu zostały wy-



Rysunek 4.3: Zbiorczy wykres krzywych ECDF prezentujących wyniki algorytmu DES na tle innych uczestników konkursu CEC. Każdy z wykresów prezentuje wyniki porównań proporcji osiągniętych progów jakości w funkcji budżetu, dla zadanej wymiarowości (zmienna n) oraz konkretnej grupy funkcji (F).

kreślone krzywe ECDF, porównujące wyniki tych algorytmów. Rysunek 4.3 przedstawia wykresy prezentujące przeciętny postęp optymalizacji w funkcji zużytego budżetu ewaluacji funkcji celu, w podziale na grupy funkcji benchmarkowych oraz liczbę wymiarów. Postęp budżetu na wszystkich wykresach ECDF wyrażony jest w postaci funkcji liczby ewaluacji funkcji celu podzielonej przez liczbę wymiarów i przedstawiony w skali logarytmicznej. Szczegółowe wyniki DES na tle innych uczestników konkursu CEC, w ujęciu dla każdej funkcji konkursowej z osobna, zamieszczone są w dodatku B.

Wyniki pokazują, że wraz ze wzrastaniem wymiarowości oraz trudności problemów,

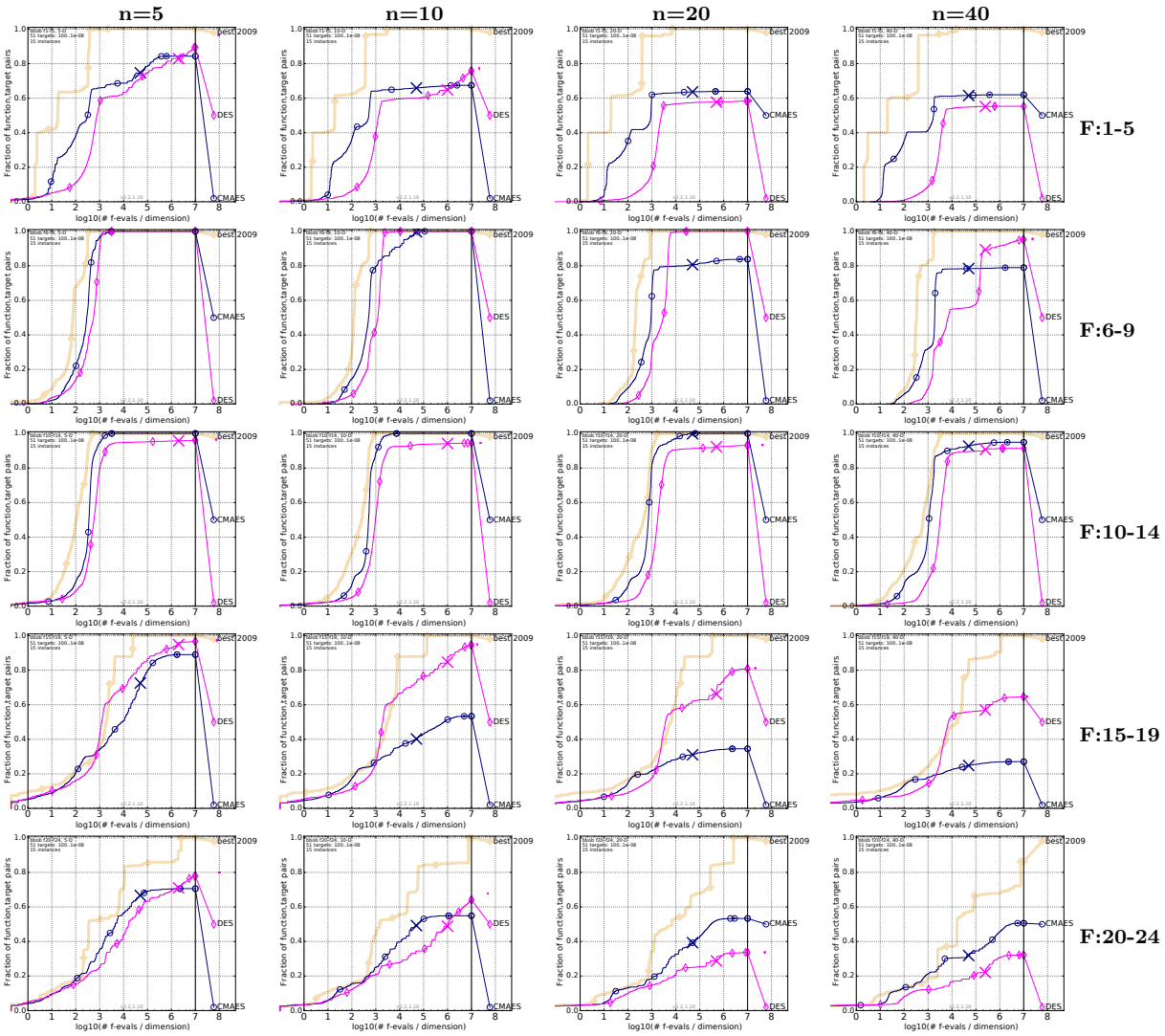
algorytm DES zbliża się coraz bardziej do zwycięzców konkursu. W przypadku problemów unimodalnych (1-3) metoda DES wyraźnie przegrywa, w szczególności w szybkości zbieżności, z metodami bazującymi na estymacji macierzy kowariancji (algorytmy CMA-ES oraz RB-IPOP-CMA-ES). Można także zauważyć, że zwycięzca całego konkursu, metoda EBOwithCMAR, posiada tempo zbieżności oraz osiągane progi jakości scharakteryzowane podobnie, jak w przypadku metody DES. Dla prostych problemów unimodalnych EBOwithCMAR również ustępuje szybkością zbieżności i skutecznością optymalizacji algorytmom z rodziny CMA-ES. Wykresy krzywych ECDF dla EBOwithCMAR oraz DES pokrywają się, w szczególności dla najwyższych wymiarowości problemów unimodalnych (1-3). Dla prostych problemów wielomodalnych (4-10) metoda DES osiąga wyraźnie wyższą proporcję progów jakości od CMA-ES. Różnice w szybkości zbieżności pomiędzy DES a metodami LSHADE-SPACMA oraz jSO są znacząco niższe niż dla problemów unimodalnych (1-3). Algorytm DES, wraz z narastaniem trudności problemów, coraz mniej odstaje pod względem tempa zbieżności od najlepszych metod z konkursu. Dla funkcji hybrydowych (11-20) oraz trudnych problemów złożonych (21-31) metoda DES na wykresach ECDF najszybciej osiąga punkty o wysokiej jakości, bądź znajduje się w czołówce w całym konkursie. Dla trudnych problemów złożonych (21-31), dla najwyższej wymiarowości, metoda DES osiąga statystycznie najlepsze rozwiązania, wyprzedzając nawet całościowego zwycięzce konkursu - metodę EBOwithCMAR. Wyniki algorytmu DES, na tle innych uczestników konkursu CEC, potwierdzają obserwację wynikającą z wyników rywalizacji tylko metod DES oraz CMA-ES na tym benchmarku. Metoda DES jest tym skuteczniejsza, im bardziej złożona jest optymalizowana funkcja celu, a problem posiada większą wymiarowość.

4.3 Wyniki zastosowania algorytmu DES na benchmarku BBOB

Rozdział ten prezentuje wyniki porównawcze algorytmu DES na benchmarku BBOB. Dane z innych algorytmów, potrzebne do generowania wykresów i oceny, dostarczane są wraz z oprogramowaniem przez autorów benchmarku. Implementacja algorytmu DES została podłączona do środowiska benchmarku BBOB oraz wybrane zostały algorytmy do porównań. Wszystkie prezentowane w tym rozdziale wykresy zostały automatycznie

wygenerowane przez środowisko benchmarku, które kontroluje cały proces testowania oraz samodzielnie generuje krzywe ECDF.

4.3.1 Wyniki DES na tle CMA-ES



Rysunek 4.4: Zbiorczy wykres krzywych ECDF prezentujących wyniki algorytmów DES oraz CMA-ES na benchmarku BBOB. Każdy z wykresów prezentuje wyniki porównań proporcji osiągniętych progów jakości w funkcji budżetu, dla zadanej wymiarowości (zmienna n) oraz konkretnej grupy funkcji (F).

Podrozdział ten przedstawia wyniki benchmarku BBOB w kontekście rywalizacji pomiędzy metodami DES oraz CMA-ES. W tym celu oba algorytmy zostały uruchomione w dostarczonym przez autorów benchmarku środowisku, który zebrał wymagane dane. Rysunek 4.4 prezentuje krzywe ECDF ukazujące przeciętny postęp optymalizacji w funkcji

zużytego budżetu ewaluacji funkcji celu, w podziale na grupy funkcji benchmarkowych oraz liczbę wymiarów. Budżet na wszystkich wykresach ECDF wyrażony jest jako liczba ewaluacji funkcji celu podzielona przez liczbę wymiarów i przedstawiony w skali logarytmicznej. Na każdym z wykresów została naniesiona przez środowisko benchmarku krzywa obrazująca zwycięzcę konkursu BBOB2009 - metodę BIPOP-CMAES[19].

Krzywe ECDF dla problemów separowalnych (1-5) oraz problemów unimodalnych źle uwarunkowanych (10-14) pokazują, że metoda CMA-ES osiąga lepsze wyniki niż DES. Tempo zbieżności oraz ostateczna proporcja osiągnietych progów jakości dla metody DES jest niższa niż w przypadku CMA-ES. Dla problemów o niskim lub przeciętnym poziomie uwarunkowania macierzy kowariancji (6-9), tempo zbieżności DES jest niższe niż CMA-ES, aczkolwiek ostateczna jakość rozwiązania uzyskiwanego przez DES jest wyższa niż w CMA-ES. Przewaga metody DES rośnie wraz ze zwiększaniem się wymiarowości przestrzeni przeszukiwań. Metoda DES osiąga znacząco lepsze rezultaty w porównaniu do CMA-ES dla grupy problemów wielomodalnych (15-19), zwłaszcza dla wyższych wymiarowości. Dla problemów z wielomodalnych o nieregularnej strukturze (20–24) DES osiąga lepszy ostateczny poziom jakości rozwiązania jedynie dla niższych wymiarowości.

4.3.2 Wyniki DES na tle innych uczestników benchmarku BBOB

W tym podrozdziale przedstawione zostały wyniki konkursu BBOB wygenerowane poprzez dostarczoną przez autorów platformę COCO. Wybrane do porównań z DES algorytmy to metody z czołówki rankingu benchmarku BBOB. Zbiór ten obejmuje trzy algorytmy oparte na CMA-ES: BIPOP-CMAES[19] (oznaczany na krzywych ECDF również jako best 2009 - zwycięzca konkursu BBOB2009), IPOP-CMAES[3] oraz NiPOP-aCMA-ES[41]. Do porównań z DES wybrany też został algorytm RL-SHADE, będący nowoczesnym wariantem ewolucji różnicowej. Rysunek 4.5 przedstawia wykresy prezentujące przeciętny postęp optymalizacji w funkcji zużytego budżetu ewaluacji funkcji celu, w podziale na grupy funkcji benchmarkowych oraz liczbę wymiarów. Budżet na wszystkich wykresach ECDF wyrażony jest jako liczba ewaluacji funkcji celu podzielona przez liczbę wymiarów i przedstawiony w skali logarytmicznej. W przypadku najwyższej wymiarowości (40 wymiarów) liczba metod naniesionych na krzywe ECDF jest ograniczona do tych, do których oprogramowanie COCO posiadało niezbędne dane.

Wyniki pokazują, że metoda DES jest najskuteczniejsza w trudnych problemach wie-

lomodalnych oraz źle uwarunkowanych funkcjach unimodalnych, w których jest w stanie zbliżać się do wiodących nowoczesnych technik z rodziny CMA-ES. Różnica pomiędzy zaawansowanymi strategiami z rodziny CMA-ES a DES, w końcowej jakości rozwiązań, nie zmienia się znacząco wraz z wymiarowością problemów. W przypadku klasycznego algorytmu CMA-ES oraz nowoczesnego wariantu ewolucji różnicowej RL-SHADE, metoda DES dla części grup problemów wygrywa, bądź jej ostateczne rezultaty są zbliżone. Tempo zbieżności metody DES w odniesieniu do wiodących technik CMA-ES (BIPOP-CMAES, NiPOP-aCMA-ES, IPOP-CMAES) jest zazwyczaj niższe. Podobna szybkość zbieżności w tych technikach, na początku procesu optymalizacji, wynikać może ze zmiennej wielkości populacji, adaptacji parametrów i restartów, które są stosowane we wszystkich rozważanych strategiach. Dlatego można przypuszczać, że strategia oparta na restartach i adaptacji parametrów, która wykorzystuje DES jako algorytm bazowy, mogłaby poprawić efektywność optymalizacji DES, tak jak IPOP-, BIPOP- lub NiPOP-aCMA-ES poprawiają wydajność zwykłego algorytmu CMA-ES.

4.4 Wyniki zastosowania algorytmu DES na benchmarku BBComp

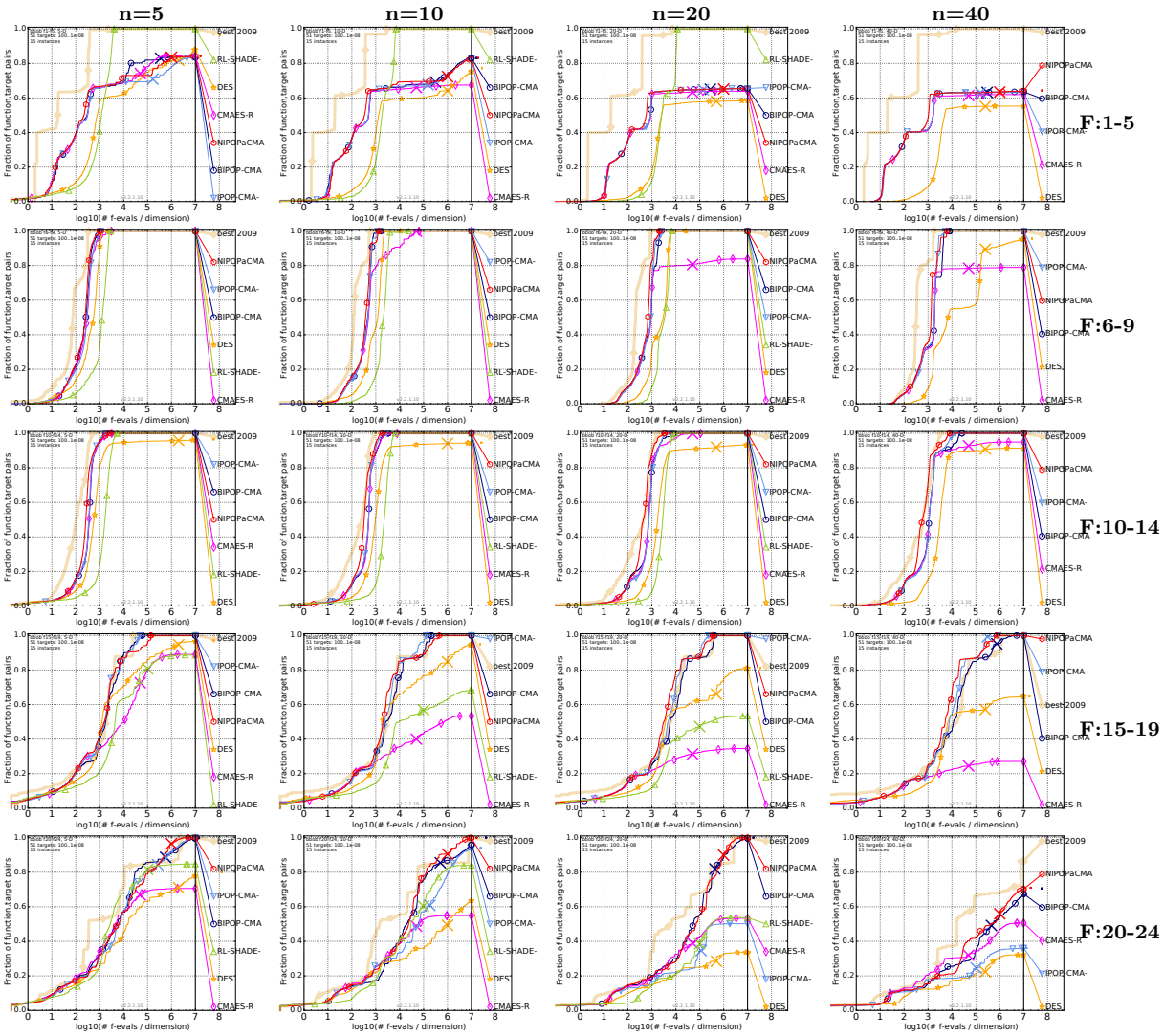
W tym rozdziale zaprezentowane są wyniki porównawcze algorytmu DES na rzeczywistym konkursie czarnej skrzynki BBComp, w wersji opisanej w podrozdziale 4.1.3. Wszystkie prezentowane w tym rozdziale wykresy zostały wygenerowane przez organizatorów i zamieszczone na stronie konkursu. Z uwagi na to, że BBComp jest konkursem zamkniętym, bez udostępniania implementacji problemów w postaci jakiegokolwiek oprogramowania, nie ma możliwości wyboru algorytmów do porównań na tym benchmarku. Dodatkowo, w konkursie BBComp brała także udział metoda CMA-DE, zdefiniowana w rozdziale 3.1.1, której wyniki są uwzględnione na wykresach konkursowych.

Rysunek 4.6 przedstawia wykresy prezentujące przeciętny postęp optymalizacji w funkcji zużytego budżetu ewaluacji funkcji celu, w podziale na liczbę wymiarów oznaczoną jako zmienna D. Postęp budżetu na wszystkich wykresach ECDF wyrażony jest w postaci funkcji liczby ewaluacji funkcji celu, przedstawionej w skali logarytmicznej. Warunki konkursowe nie wymagały od uczestników podawania opisu, ani nawet nazwy algorytmu, z którym przystępują do konkursu. Zaprojektowanie konkursu w architektu-

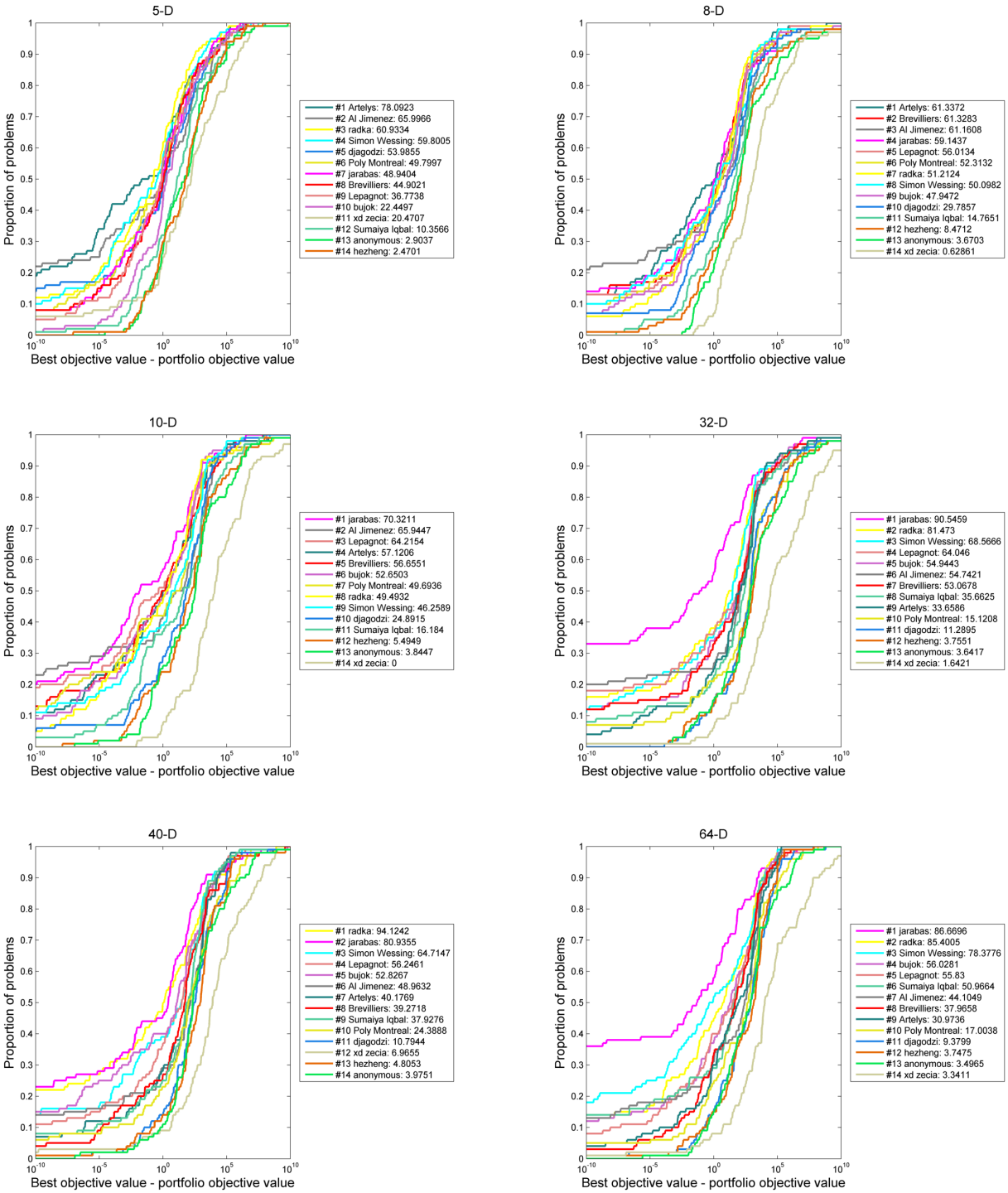
rze klient-serwer sprawiło, że organizatorzy nie mieli dostępu do implementacji metod biorących udział w konkursie. Dlatego właśnie opisy metod na wykresach zawierają tylko nazwę użytkownika uczestnika, poza kilkoma wyjątkami uczestników, którzy jawnie zadeklarowali nazwy algorytmów.

Zwycięzcą konkursu została metoda oznaczana na wykresach jako *Simon Wessing*, będąca kombinacją algorytmów Neldera-Meada, CMA-ES oraz L-BFGS-B. Trzecie miejsce w klasyfikacji ogólnej zajęła metoda DES, oznaczana na wykresach nazwą *jarabas*. Kolejne miejsca zajęli użytkownicy *Al Jimenez* - algorytm CTA[31], *Artelys Knitro* - metoda używająca oprogramowania Artelys Knitro, *Lepagnot* wykorzystujący algorytm o nazwie MSS oraz użytkownik *Brevilliers* z metodą BSA-DE-SA. Metoda CMA-DE została na wykresach opisana nazwą *djagodzi*. O technikach używanych przez resztę uczestników nie wiadomo nic więcej.

Wyniki pokazują, że metoda DES (oznaczona na wykresach różową linią), która zajęła trzecie miejsce w klasyfikacji generalnej konkursu, osiąga najlepsze wyniki dla wyższych wymiarowości. Wraz ze wzrastaniem wymiarowości problemów, poprawia się jakość optymalizacji metody DES i miejsce w rankingu. Już od D=10 wymiarów algorytm DES zajmuje pierwsze, bądź jednokrotnie drugie (dla D=40), miejsce w rankingu wszystkich metod biorących udział w konkursie. Dla największych wymiarowości (D=64) metoda DES zwyciężyła w ujęciu wszystkich problemów, znacząco wyprzedzając wszystkie inne algorytmy w początkowym tempie zbieżności. Wyniki DES w porównaniu z metodą CMA-DE (oznaczoną na wykresach niebieską linią), pokazują znaczącą przewagę dla algorytmu DES. CMA-DE, która zajęła dziesiąte miejsce w klasyfikacji generalnej konkursu, osiąga lepszy wynik rankingu niż DES tylko dla najniższych wymiarowości (D=5).



Rysunek 4.5: Zbiorczy wykres krzywych ECDF prezentujących wyniki algorytmu DES na tle innych uczestników konkursu BBOB. Każdy z wykresów prezentuje wyniki porównań proporcji osiągniętych progów jakości w funkcji budżetu, dla zadanej wymiarowości (zmienna n) oraz konkretnej grupy funkcji (F).



Rysunek 4.6: Zbiorczy wykres krzywych ECDF prezentujących wyniki algorytmu DES na tle innych uczestników konkursu BBComp. Każdy z wykresów prezentuje wyniki porównań proporcji osiągniętych progów jakości w funkcji budżetu, dla zadanej wymiarowości (podpis nad każdym wykresem) i wszystkich problemów benchmarku BBComp. Metoda DES oznaczona jest linią różową i nazwą *jarabas*. Metoda CMA-DE oznaczona jest linią niebieską i nazwą *djagodzi*. Liczby przy każdej nazwie algorytmu oznaczają ranking metody dla każdego wykresu zbiorczego.

Rozdział 5

Podsumowanie

Cel rozprawy sformułowany w rozdziale 1.2 został osiągnięty. Zaproponowany został algorytm, który modeluje dynamikę zmian rozkładów generowanych populacji w algorytmie CMA-ES, bez konieczności estymacji bądź przybliżania rzeczywistej macierzy kowariancji. Zaprezentowana w rozdziale 3 metoda różnicowej strategii ewolucyjnej (DES) implementuje kluczowe rozwiązania ewolucyjnej strategii adaptacji macierzy kowariancji, skupiając się na zależnościach pomiędzy punktami, a nie rozkładami prawdopodobieństwa. Zaproponowana metoda zastępuje kosztowny proces estymacji macierzy kowariancji i generowania realizacji wielowymiarowego rozkładu normalnego, koncepcją mutacji różnicowej. Zgodnie z przeprowadzonymi eksperymentami opisanymi w rozdziale 4.2.1, DES zachowuje liniową względem liczby wymiarów zależność czasową dla szeregu różnych funkcji celu. Algorytm różnicowej strategii ewolucyjnej posiada także potwierdzoną doświadczalnie (rozdział 3.2) tendencję do dopasowywania się do konturów funkcji celu w stopniu analogicznym do CMA-ES.

W rozdziale 2 prezentującym krytyczną analizę problemów metaheurystyk z rodzin CMA-ES i DE, zaprezentowany został przegląd wiodących technik optymalizacji dla tych rodzin algorytmów. Poznanie problemów i sposobów radzenia sobie z ograniczeniami tych rozwiązań, umożliwiło późniejszą analizę wyników rywalizacji metaheurystyk z rodzin CMA-ES i DE z wprowadzoną metodą DES. Wizualizacja wyników poprzez krzywe ECDF opisane w rozdziale 4.1.4, pozwoliła przedstawiać w łatwy sposób postęp optymalizacji na tle innych metod, również ujęciu zbiorczym dla wielu różnych zadań optymalizacji jednocześnie. Szereg testów przeprowadzonych w rozdziale 4 potwierdził wysoką skuteczność algorytmu DES na tle wiodących metod optymalizacyjnych, w procesie optymalizacji glo-

balnej. Zgodnie z wynikami eksperymentalnymi dla benchmarku BBOB, średnia wydajność DES jest porównywalna do CMA-ES. Dla zestawu funkcji konkursowych CEC, wyniki metody różnicowej strategii ewolucyjnej są znacznie lepsze niż dla CMA-ES, zwłaszcza w przypadku skomplikowanych problemów wielomodalnych i dużej liczby wymiarów. Z szeregu przeprowadzonych eksperymentów można wywnioskować, że DES jest metodą optymalizacji konkurencyjną w stosunku do CMA-ES jak również wobec innych metod optymalizacyjnych biorących udział w konkursach. Metoda DES jest tym skuteczniejsza, im bardziej złożona jest optymalizowana funkcja celu, a problem posiada większą wymiarowość. Potwierdzeniem tej tezy jest wysoka skuteczność metody DES w procesie uczenia konwolucyjnych i rekurencyjnych sieci neuronowych, opisana w pracy [30].

Dalszy rozwój DES będzie dwukierunkowy. Jednym z kierunków jest poszukiwanie poprawy efektywności w optymalizacji lokalnej. Obecna wersja DES nie zawiera mechanizmu analogicznego do adaptacji mnożnika kroku, który umożliwia CMA-ES osiąganie doskonałej szybkości i precyzji zbieżności. Zgodnie z rozważaniami w rozdziale 2.2.1[Współczynnik skalujący], dobór współczynnika F pełni tylko rolę wspomagającą adaptację kroku, głównie w początkowych fazach procesu optymalizacji. Operator mutacji różnicowej posiada wbudowaną zdolność samoadaptacji długości kroku bazującą na sposobie doboru punktów tworzących wektory różnicowe. Należy więc przeprowadzić badania, czy w przypadku używania mutacji różnicowej przez metodę DES, dodatkowa adaptacja mnożnika długości kroku jest niezbędna. Innym sposobem poprawy szybkości i precyzji zbieżności w optymalizacji lokalnej może być zmiana w sposobie liczenia punktu środkowego. Każdorazowe wyznaczanie i ewaluacja punktu środkowego, jak to obecnie ma miejsce w DES, może być nieskuteczne, gdy optymalizacja nie jest w fazie eksploatacji. Badania ustalające sposób wykrywania takiej sytuacji pozwoliłyby uzależnić sposób obliczania punkt środkowego od fazy optymalizacji tak, by zwiększyć tempo zbieżności. Ponadto, obliczanie punktu środkowego gdy optymalizowana funkcja nie jest w przybliżeniu kwadratowa w otoczeniu optimum lokalnego, może być bezcelowe. W takim wypadku, warunkowe uruchamianie obliczeń punktu środkowego pozwoliłoby oszczędzać budżet ewaluacji funkcji celu. Drugim kierunkiem rozwoju metody DES jest poprawa efektywności w realizacji zadań optymalizacji globalnej. Zastosowanie w DES popularnych technik implementowanych w nowoczesnych algorytmach z rodzin CMA-ES i DE, opisanych w rozdziałach odpowiednio 2.1.3 oraz 2.2.2, może skutkować poprawą zdolności eksploracyjnych. Przeprowadzenie badań

w kierunku opracowania strategii stosowania inteligentnych restartów oraz dostosowania parametrów metody, może poprawić globalną efektywność optymalizacji. Uzmiennienie liczebności populacji, stosując np. redukcję liczby punktów w populacji w każdej kolejnej generacji, pozwoliłoby np. stosować liczniejsze populacje w fazie eksploracji, wpływając na poprawienie się efektywności optymalizacji globalnej.

Bibliografia

[1] J. Arabas and R. Biedrzycki. Improving evolutionary algorithms in a continuous domain by monitoring the population midpoint. *IEEE Transactions on Evolutionary Computation*, 21(5):807–812, 2017.

[2] J. Arabas and D. Jagodziński. Toward a matrix-free covariance matrix adaptation evolution strategy. *IEEE Transactions on Evolutionary Computation*, 24(1):84–98, 2020.

[3] A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1769–1776 Vol. 2, Sep. 2005.

[4] N. H. Awad, M. Z. Ali, and P. N. Suganthan. Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems. In *IEEE Congress on Evolutionary Computation*, pages 372–379, 2017.

[5] N. H. Awad, M.Z. Ali, J.J. Liang, B.Y. Qu, and Ponnuthurai N Suganthan. Problem definitions and evaluation criteria for the CEC 2017 special session and competition on real-parameter optimization. *Technical Report, Nanyang Technological University, Singapore and Jordan University of Science and Technology and Zhengzhou University, China*, 2016.

[6] H. G. Beyer and B. Sendhoff. Simplify your Covariance Matrix Adaptation Evolution Strategy. *IEEE Transactions on Evolutionary Computation*, 21(5):746–759, 2017.

[7] R. Biedrzycki. A version of IPOP-CMA-ES algorithm with midpoint for CEC 2017 single objective bound constrained problems. In *IEEE Congress on Evolutionary Computation*, pages 1489–1494, 2017.

[8] R. Biedrzycki, J. Arabas, and D. Jagodziński. Bound constraints handling in differential evolution: An experimental study. *Swarm and Evolutionary Computation*, 50:100453, 2019.

[9] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35:268–308, 01 2001.

[10] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657, 2006.

[11] J. Brest, M. S. Maučec, and B. Bošković. Single objective real-parameter optimization: Algorithm jSO. In *IEEE Congress on Evolutionary Computation*, pages 1311–1318, 2017.

[12] F. Caraffini and F. Neri. Rotation invariance and rotated problems: An experimental study on differential evolution. In K. Sim and P. Kaufmann, editors, *Applications of Evolutionary Computation*, pages 597–614, Cham, 2018. Springer International Publishing.

[13] S. Dasgupta, A. Biswas, S. Das, and A. Abraham. The population dynamics of differential evolution: A mathematical model. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1439–1446, 2008.

[14] N. Hansen. The CMA evolution strategy. *http://www.cmap.polytechnique.fr/ nikolaus.hansen/cmaesintro.html*.

[15] N. Hansen. *Verallgemeinerte individuelle Schrittweitenregelung in der Evolutionsstrategie*. Technical University Berlin, 1998.

[16] N. Hansen. The CMA Evolution Strategy: A Tutorial. ArXiv e-prints, arXiv:1604.00772, 2016, pp.1-39, 2005.

[17] N. Hansen. The CMA evolution strategy: a comparing review. In Jose A Lozano, editor, *Towards a new evolutionary computation: advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.

[18] N. Hansen. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In *Workshop Proceedings of the GECCO Genetic and Evolutionary Computation Conference*, pages 2389–2395. ACM, July 2009.

[19] N. Hansen. Benchmarking a bi-population CMA-ES on the BBOB-2009 function testbed. *GECCO (Companion)*, 07 2009.

[20] N. Hansen. The CMA Evolution Strategy: A Tutorial. 2016.

[21] N. Hansen, A. Auger, D. Brockhoff, D. Tusar, and T. Tusar. Coco: Performance assessment, 2016.

[22] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík. Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '10, page 1689–1696, New York, NY, USA, 2010. Association for Computing Machinery.

[23] N. Hansen and S. Kern. Evaluating the CMA evolution strategy on multimodal test functions. In X. Yao et al., editors, *Parallel Problem Solving from Nature PPSN VIII*, volume 3242 of *LNCS*, pages 282–291. Springer, 2004.

[24] N. Hansen, S. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11:1–18, 02 2003.

[25] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 312–317, May 1996.

[26] N. Hansen and A. Ostermeier. Convergence properties of evolution strategies with the derandomized covariance matrix adaptation. *In Proceedings of EUFIT97, 5th Europ. Congr. on Intelligent Techniques and Soft Computing*, pages 650–654, 1997.

[27] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[28] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, and A. Auger. Impacts of invariance in search: When CMA-ES and PSO face ill-conditioned and non-separable problems. *Applied Soft Computing*, 11(8):5755–5769, 2011.

[29] J. Chiou and F. Wang. A hybrid method of differential evolution with application to optimal control problems of a bioprocess system. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, pages 627–632, 1998.

[30] D. Jagodziński, Ł. Neumann, and P. Zawistowski. Deep neuroevolution: Training neural networks using a matrix-free evolution strategy. In *Neural Information Processing*, pages 524–536, Cham, 2021. Springer International Publishing.

[31] A. J. Jiménez. Sparse Hessian factorization in curved trajectories for unconstrained minimization. *Optimization Methods and Software*, 29(1):1–9, 2014.

[32] R. Joshi and A. C. Sanderson. Minimal representation multisensor fusion using differential evolution. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pages 266–273, 1997.

[33] R. Joshi and A. C. Sanderson. Minimal representation multisensor fusion using differential evolution. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 29(1):63–76, 1999.

[34] H. Kämpf and D. Robinson. A hybrid CMA-ES and HDE optimisation algorithm with application to solar energy potential. *Appl. Soft Comput.*, 9(2):738–745, 2009.

[35] K.Price. Genetic annealing. *Dr. Bobb's J*, 19(10):127–132, 1994.

[36] A. Kumar, R. K. Misra, and D. Singh. Improving the local search capability of Effective Butterfly Optimizer using covariance matrix adapted retreat phase. In *IEEE Congress on Evolutionary Computation*, pages 1835–1842, 2017.

[37] Y. L. Li, Z. H. Zhan, Y. J. Gong, W. N. Chen, J. Zhang, and Y. Li. Differential Evolution with an evolution path: A DEEP evolutionary algorithm. *IEEE Transactions on Cybernetics*, 45(9):1798–1810, 2015.

[38] I. Lopez-Cruz, G. van Willigenburg, and G. van Straten. Efficient differential evolution algorithms for multimodal optimal control problems. *Appl. Soft Comput.*, 3:97–122, 09 2003.

[39] I. Loshchilov and T. Glasmachers. Black box optimization competition.

[40] I. Loshchilov, T. Glasmachers, and H. Beyer. Large scale black-box optimization by limited-memory matrix adaptation. *IEEE Transactions on Evolutionary Computation*, accepted for publication, 2018.

[41] I. Loshchilov, Marc Schoenauer, and Michèle Sebag. Alternative restart strategies for CMA-ES. In Carlos A. Coello Coello, Vincenzo Cutello, Kalyanmoy Deb, Stephanie Forrest, Giuseppe Nicosia, and Mario Pavone, editors, *Parallel Problem Solving from Nature - PPSN XII*, pages 296–305, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[42] R. Mallipeddi and P. N. Suganthan. Empirical study on the effect of population size on differential evolution algorithm. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 3663–3670, 2008.

[43] A. W. Mohamed, A. A. Hadi, A. M. Fattouh, and K. M. Jambi. LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems. In *IEEE Congress on Evolutionary Computation*, pages 145–152, 2017.

[44] N. Noman and H. Iba. Accelerating differential evolution using an adaptive local search. *IEEE Transactions on Evolutionary Computation*, 12(1):107–125, 2008.

[45] K. Opara and J. Arabas. Differential evolution: A survey of theoretical analyses. *Swarm and Evolutionary Computation*, 44:546–558, 2019.

[46] V. P. Plagianakos, D. K. Tasoulis, and M. N. Vrahatis. *A Review of Major Application Areas of Differential Evolution*, pages 197–238. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[47] J. Poland and A. Zell. Main vector adaptation: A CMA variant with linear time and space complexity. In *Proc. Genet. Evol. Comput. Conf.*, pages 1050–1055. Morgan Kaufmann, 2001.

[48] A. K. Qin and P. N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1785–1791 Vol. 2, 2005.

[49] R. Raymond and N. Hansen. A simple modification in CMA-ES achieving linear time and space complexity. In *Parallel Problem Solving from Nature – PPSN X*, pages 296–305, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[50] R.Storn. On the usage of differential evolution for function optimization. *Proceedings of the Biennial Conference of the North American Fuzzy Information Processing Society*, pages 519 – 523, 07 1996.

[51] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam. Multi-method based orthogonal experimental design algorithm for solving CEC2017 competition problems. In *IEEE Congress on Evolutionary Computation*, pages 1350–1357, 2017.

[52] H. Stegherr, M. Heider, and J. Hähner. Classifying metaheuristics: Towards a unified multi-level classification system. *Natural Computing: An International Journal*, 21(2):155–171, jun 2022.

[53] R. Storn and K. Price. Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[54] P.N. Suganthan. http://www.ntu.edu.sg/home/epnsugan/.

[55] Y. Sun, T.Schaul, F.Gomez, and J.Schmidhuber. A linear time natural evolution strategy for non-separable functions. In *GECCO (Companion)*, pages 61–62. ACM, 2013.

[56] T. Chiang, Ch. Chen, and Y. Lin. Parameter control mechanisms in differential evolution: A tutorial review and taxonomy. In *2013 IEEE Symposium on Differential Evolution (SDE)*, pages 1–8, 2013.

[57] R. Tanabe and A. Fukunaga. Success-history based parameter adaptation for differential evolution. In *2013 IEEE Congress on Evolutionary Computation*, pages 71–78, 2013.

[58] R. Tanabe and A. S. Fukunaga. Improving the search performance of shade using linear population size reduction. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 1658–1665, 2014.

[59] V.Feoktistov. *Differential Evolution: In Search of Solutions (Springer Optimization and Its Applications)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[60] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. In *Technical Report SFI-TR-95-02-010*, 1995.

[61] J. Zhang and A. C. Sanderson. JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958, 2009.

Dodatki

Dodatek A

Parametry algorytmów użytych w eksperymentach

Dodatek przedstawia ustawienia parametrów poszczególnych metod użytych w eksperymentach w rozdziale 3 oraz rozdziale 4:

- CMA-ES

  $\lambda = 4 + \lfloor 3\log(n) \rfloor$, $\mu = \lfloor \frac{\lambda}{2} \rfloor$, $w_i = \beta \cdot (\log(\mu + 0.5) - \log(i))$, gdzie β jest współczynnikiem normalizującym,
  $c_\sigma = \frac{\mu_{eff} + 2}{n + \mu_{eff} + 5}$, gdzie $\mu_{eff} = 1/\sum(w_i)^2$,
  $d_\sigma = 1 + 2\max\left(0, \sqrt{\frac{\mu_{eff} - 1}{n+1}} - 1\right) + c_\sigma$,
  $c_c = \frac{4 + \mu_{eff}/n}{n + 4 + 2\mu_{eff}/n}$, $c_1 = \frac{2}{(n+1.3)^2 + \mu_{eff}}$,
  $c_\mu = \min\left(1 - c_1, 2\frac{\mu_{eff} - 2 + 1/\mu_{eff}}{(n+2)^2 + \mu_{eff}}\right)$

- CMA-DE

  $\lambda = 4 + \lfloor 3\log(n) \rfloor$, $\mu = \lfloor \frac{\lambda}{2} \rfloor$, $c_d = \frac{\mu}{\mu + 2}$, $c_c = \frac{1}{\sqrt{n}}$, $c_{cov} = \frac{2}{n^2}$.

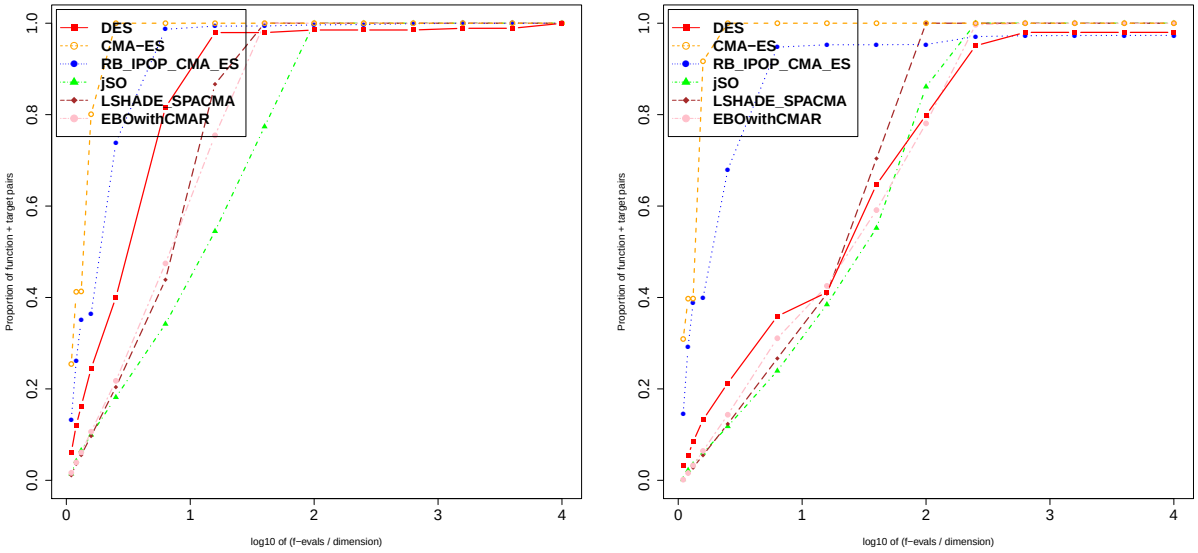
- DES

  $\lambda = 4 + \lfloor 3\log(n) \rfloor$ (Rozdział 3.2), $\lambda = 4n$ (Rozdział 4.2 oraz Rozdział 4.3),
  $\mu = \lfloor \frac{\lambda}{2} \rfloor$, $c_d = \frac{\mu}{\mu + 2}$, $c_c = \frac{1}{\sqrt{n}}$,
  $c_{cov} = \frac{2}{n^2}$, $H = 6 + 3\sqrt{n}$, $\varepsilon = 10^{-6}$.

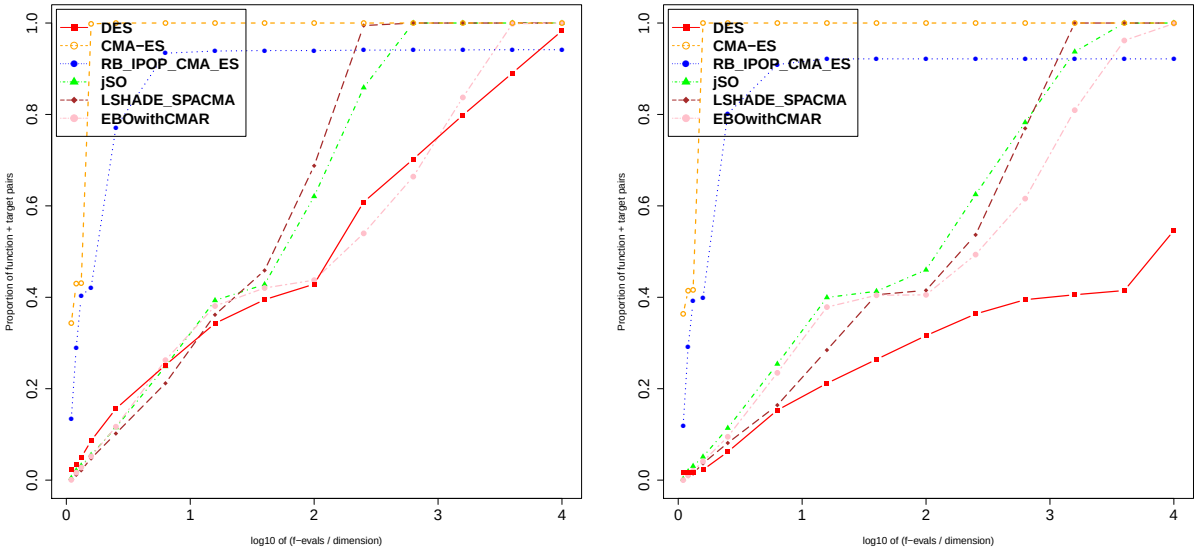
Dodatek B

Szczegółowe wyniki zastosowania algorytmu DES na benchmarku CEC

Poniższy dodatek ukazuje wykresy krzywych ECDF prezentujące wyniki algorytmu DES na tle innych uczestników konkursu CEC, w ujęciu dla każdej funkcji konkursowej z osobna. Każdy z wykresów prezentuje wyniki porównań proporcji osiągniętych progów jakości w funkcji budżetu, dla zadanej wymiarowości (zmienna n) oraz konkretnej funkcji konkursu. Szczegółowy opis konkursu optymalizacyjnego CEC, wraz z opisem problemów wchodzących w jego skład, znajduje się w podrozdziale 4.1.1.

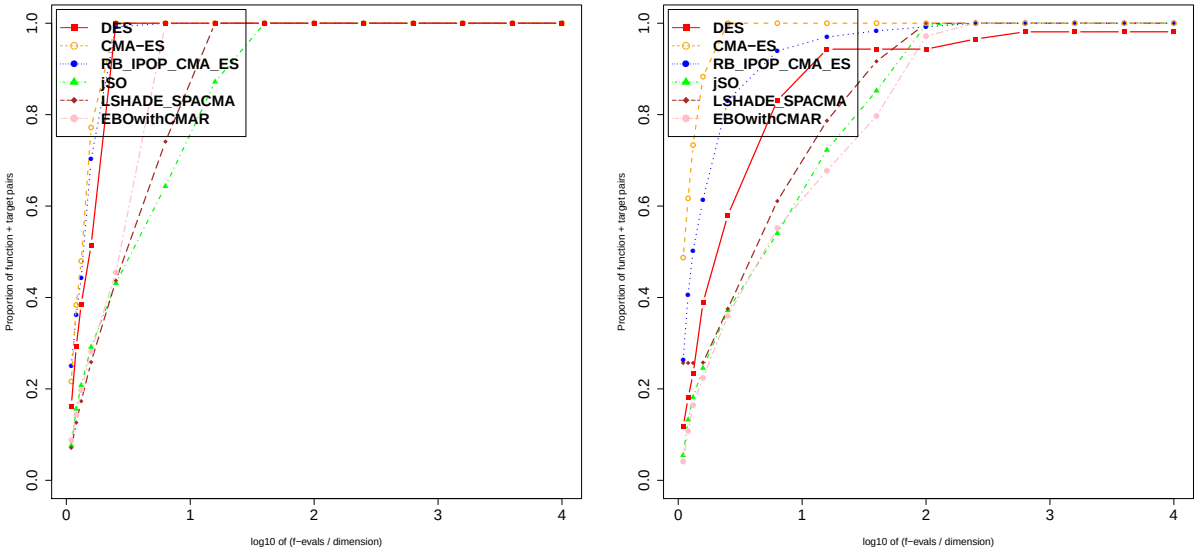


(a) n=10 (b) n=30



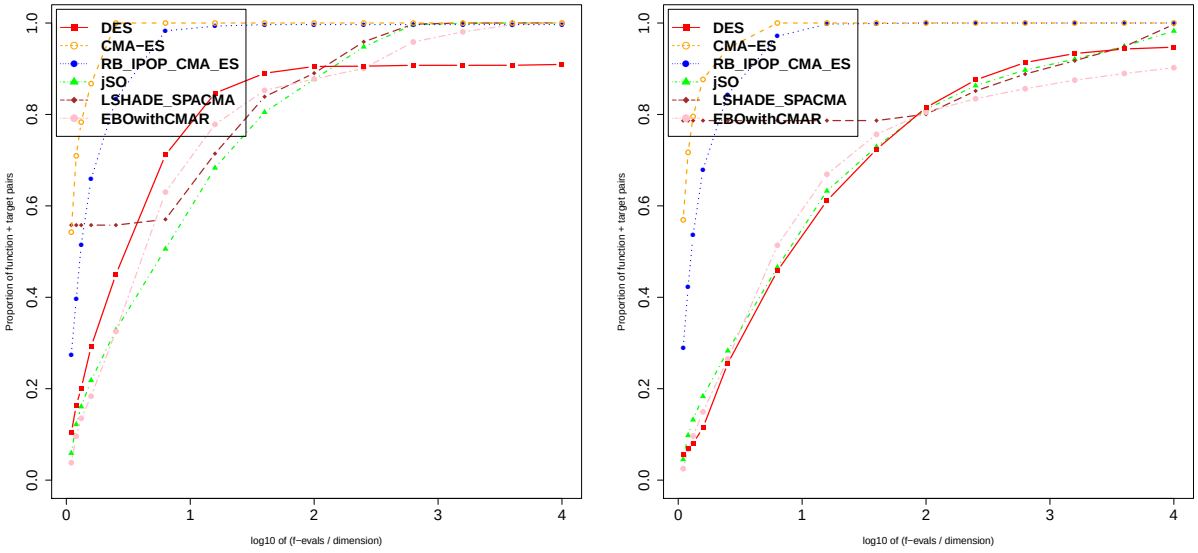
(c) n=50 (d) n=100

Rysunek B.1: Funkcja 1 benchmarku CEC.



(a) n=10

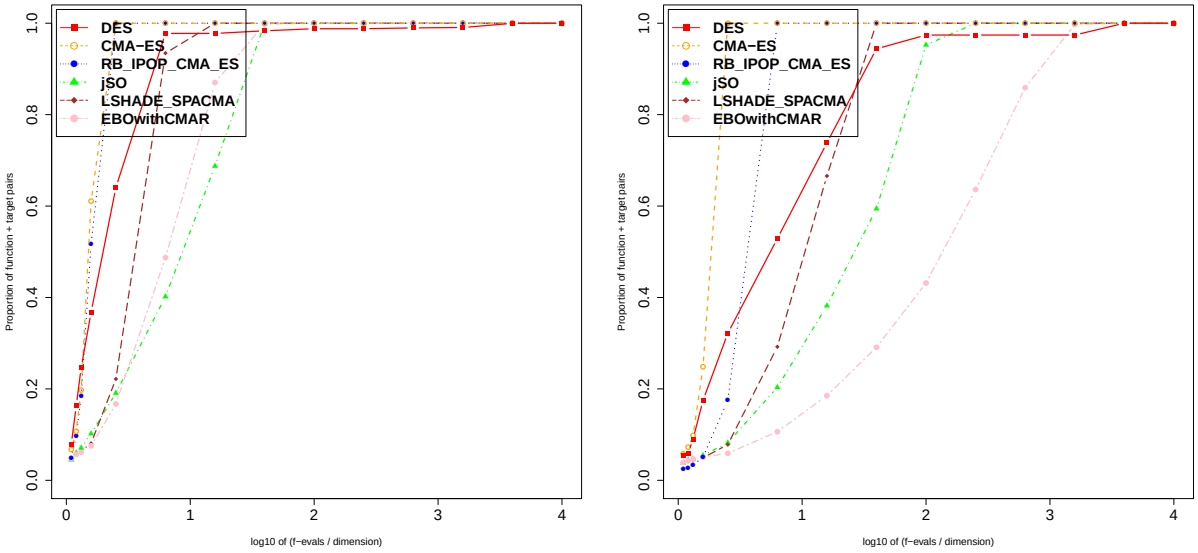
(b) n=30



(c) n=50

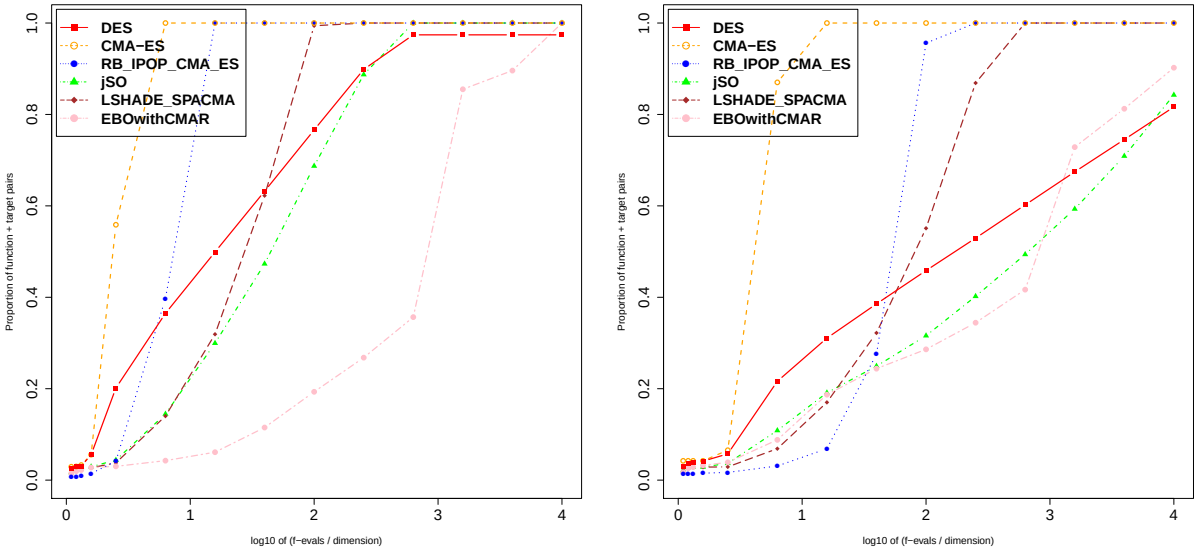
(d) n=100

Rysunek B.2: Funkcja 2 benchmarku CEC.



(a) n=10

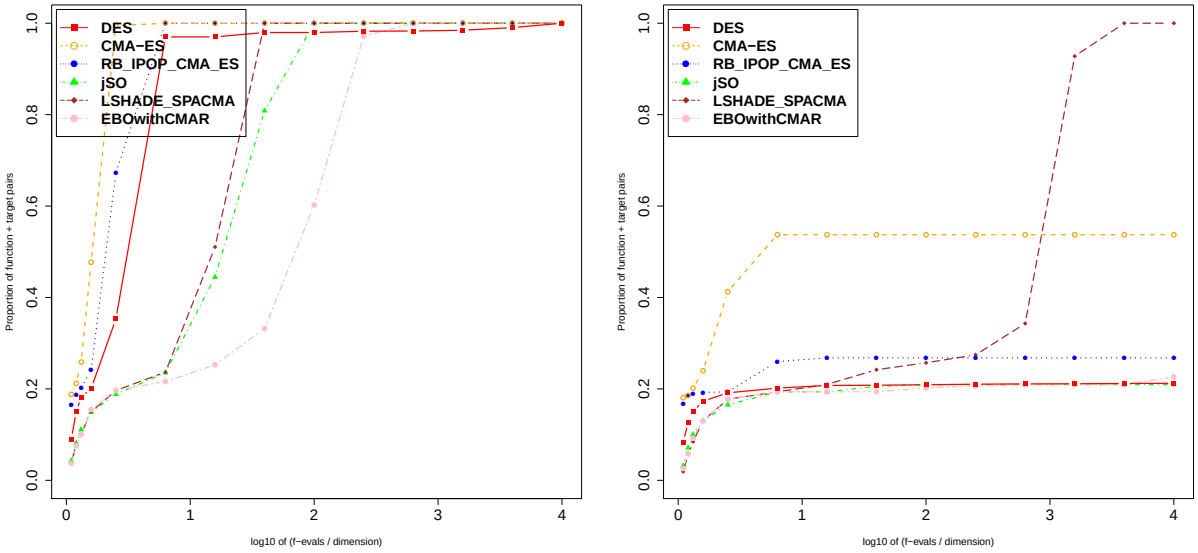
(b) n=30



(c) n=50

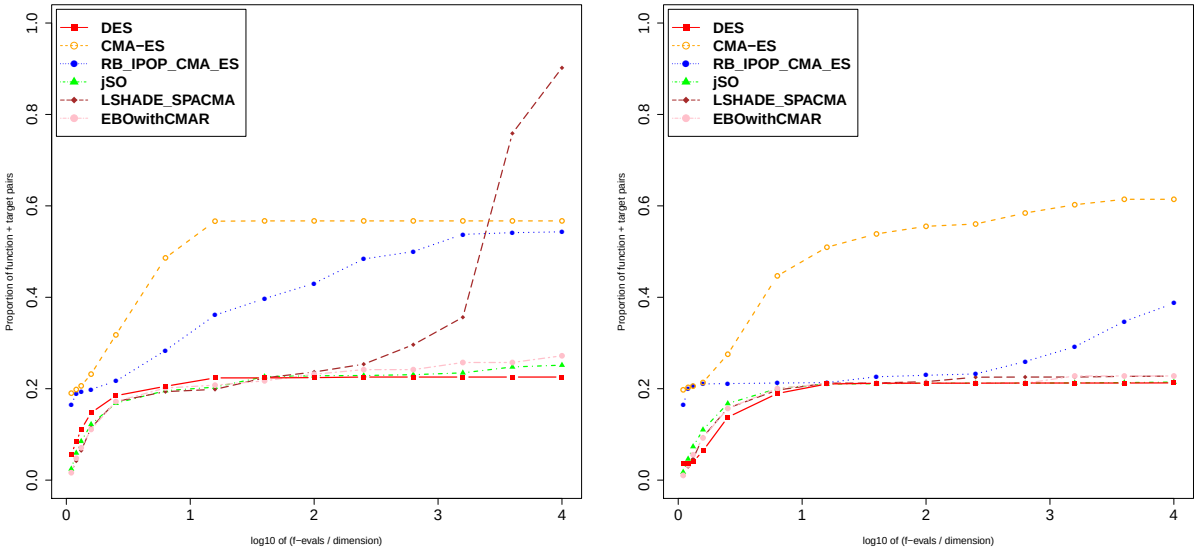
(d) n=100

Rysunek B.3: Funkcja 3 benchmarku CEC.



(a) n=10

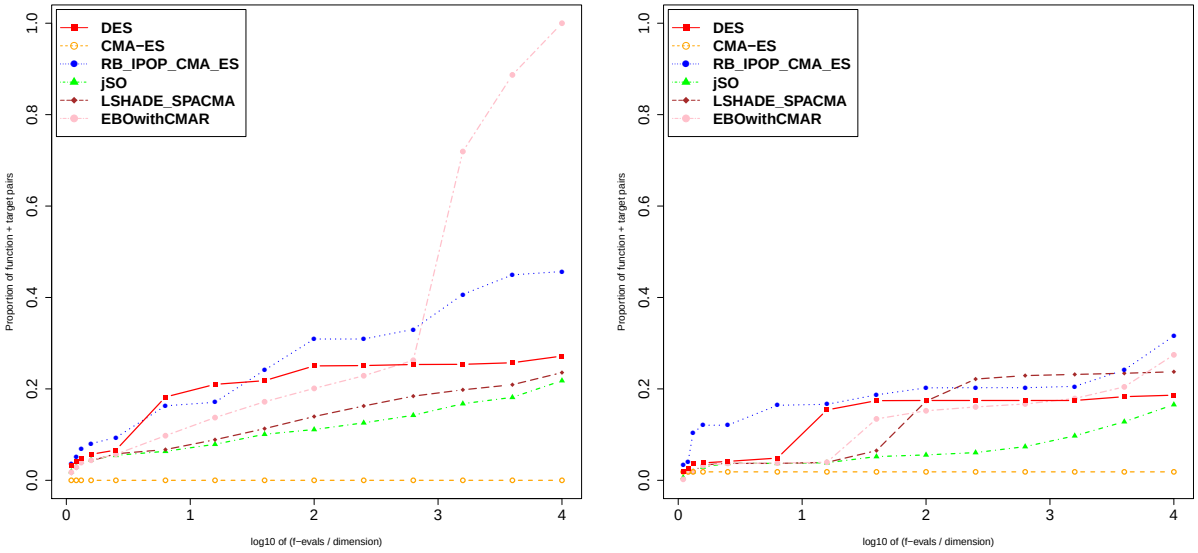
(b) n=30



(c) n=50

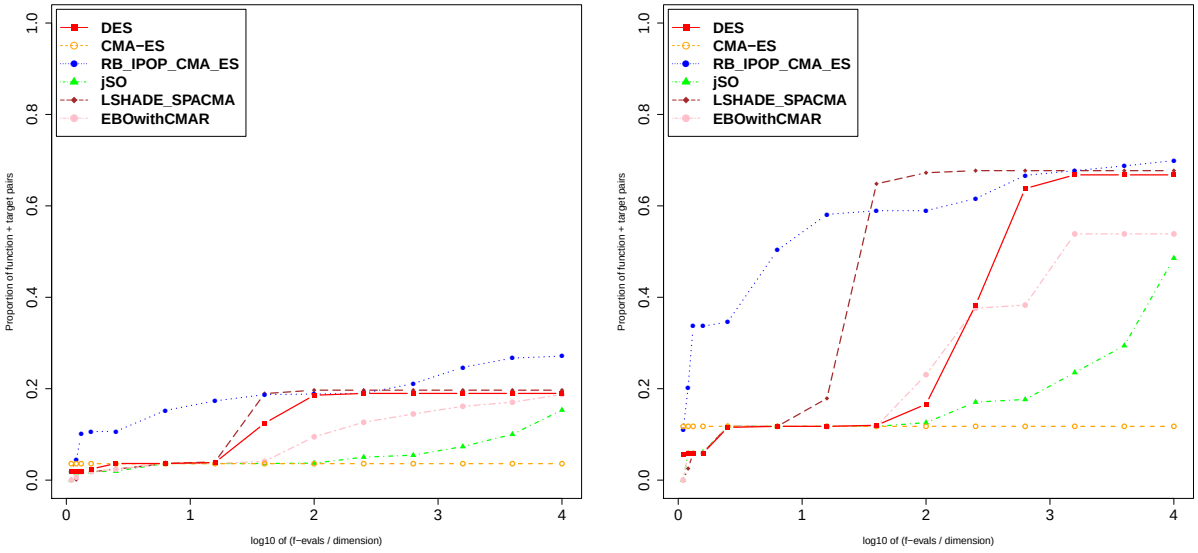
(d) n=100

Rysunek B.4: Funkcja 4 benchmarku CEC.



(a) n=10

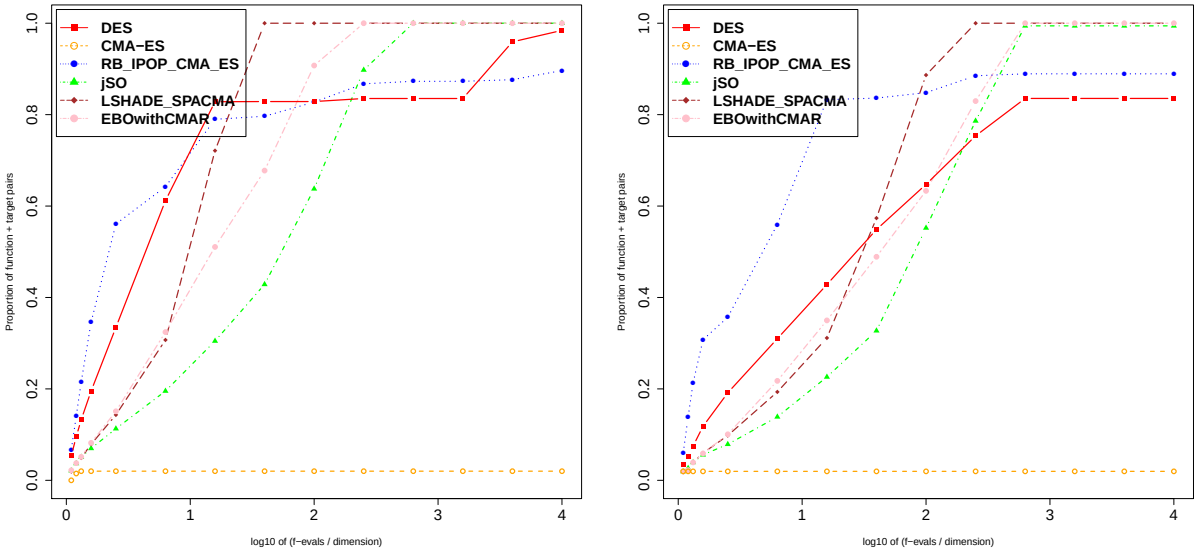
(b) n=30



(c) n=50

(d) n=100

Rysunek B.5: Funkcja 5 benchmarku CEC.



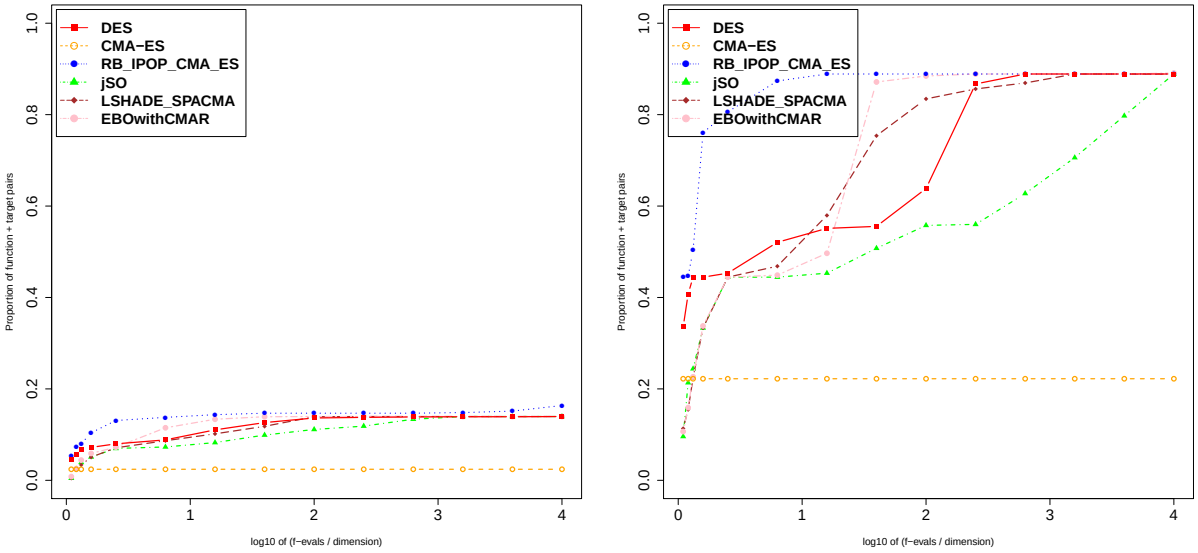
(a) n=10

(b) n=30

(c) n=50

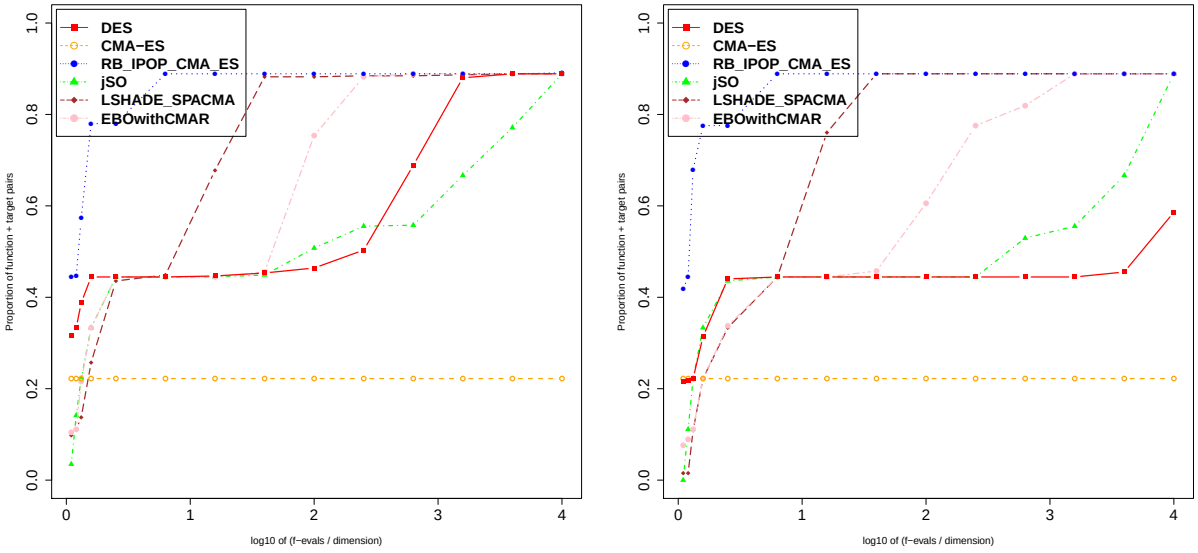
(d) n=100

Rysunek B.6: Funkcja 6 benchmarku CEC.



(a) n=10

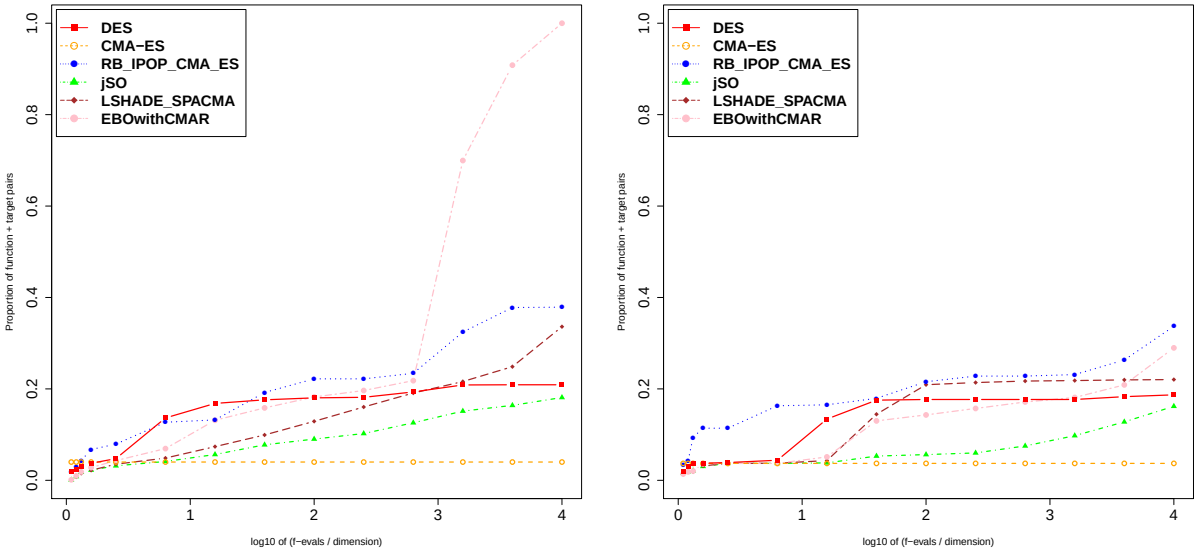
(b) n=30



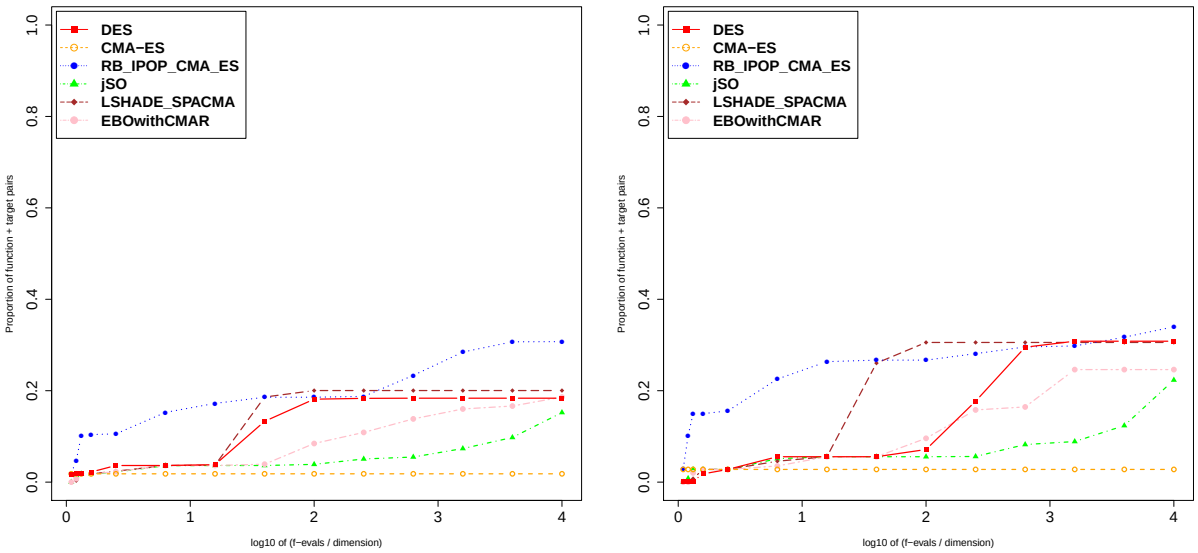
(c) n=50

(d) n=100

Rysunek B.7: Funkcja 7 benchmarku CEC.

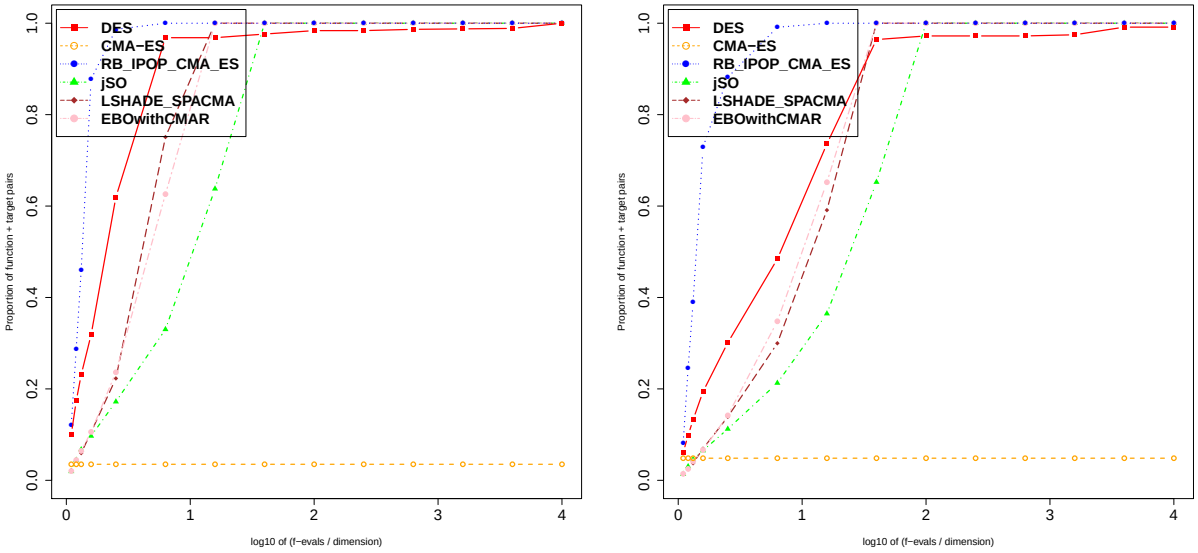


(a) n=10 (b) n=30



(c) n=50 (d) n=100

Rysunek B.8: Funkcja 8 benchmarku CEC.



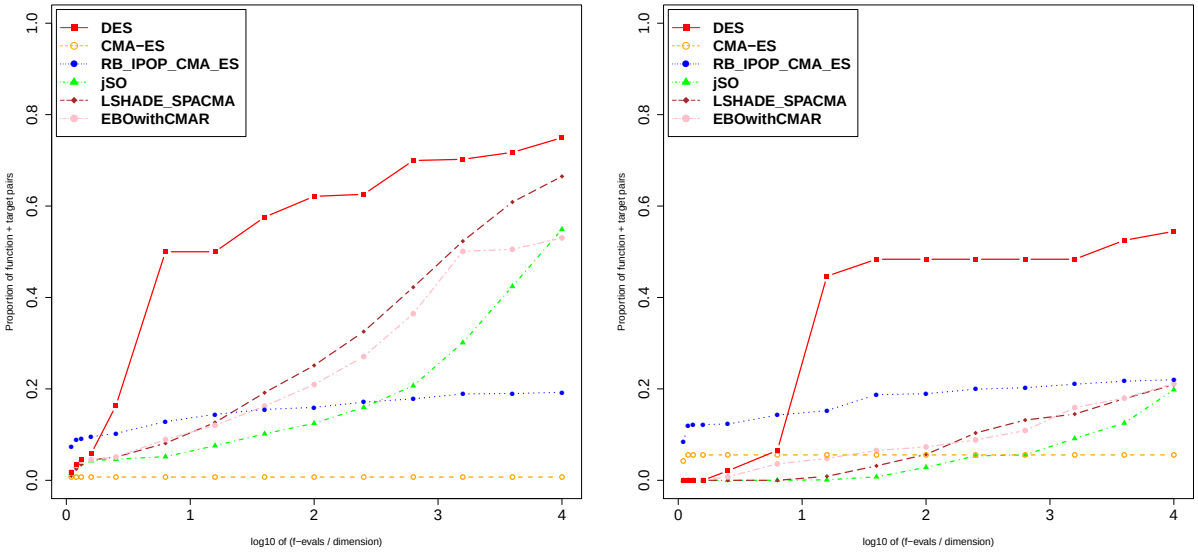
(a) n=10

(b) n=30

(c) n=50

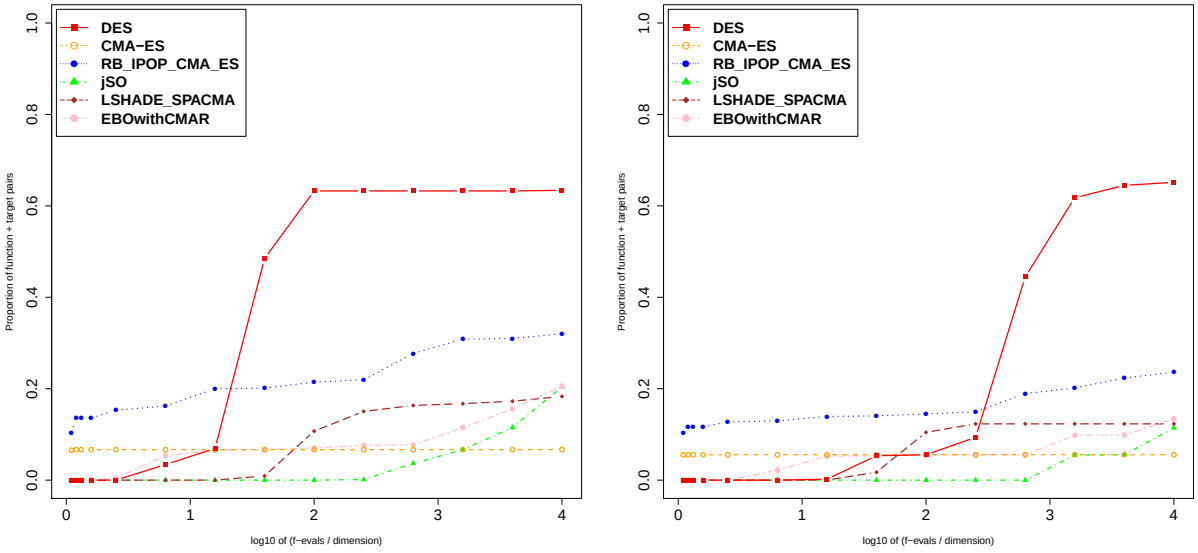
(d) n=100

Rysunek B.9: Funkcja 9 benchmarku CEC.



(a) n=10

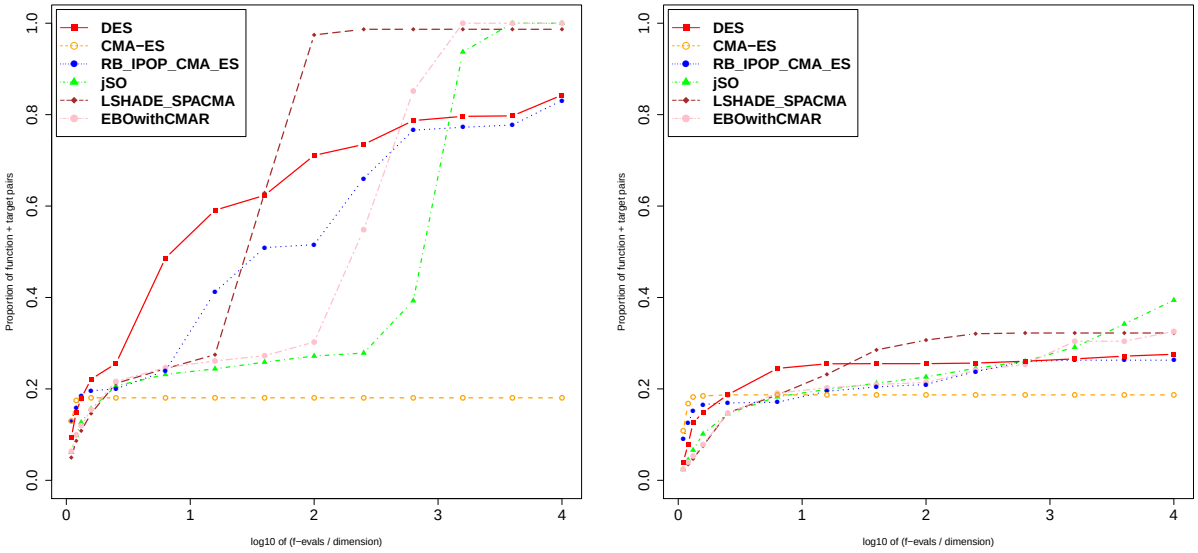
(b) n=30



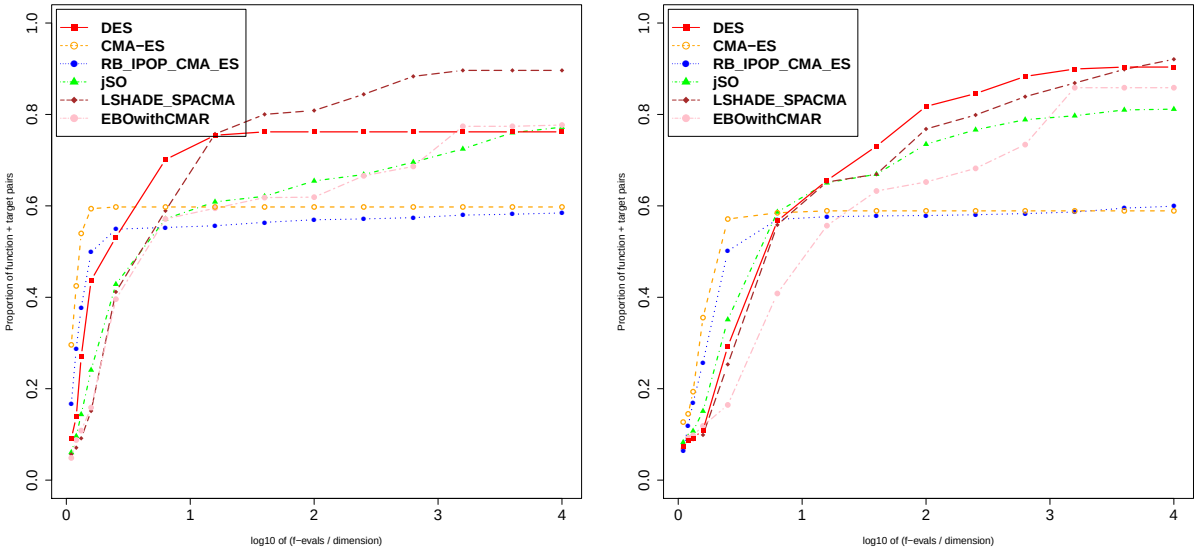
(c) n=50

(d) n=100

Rysunek B.10: Funkcja 10 benchmarku CEC.

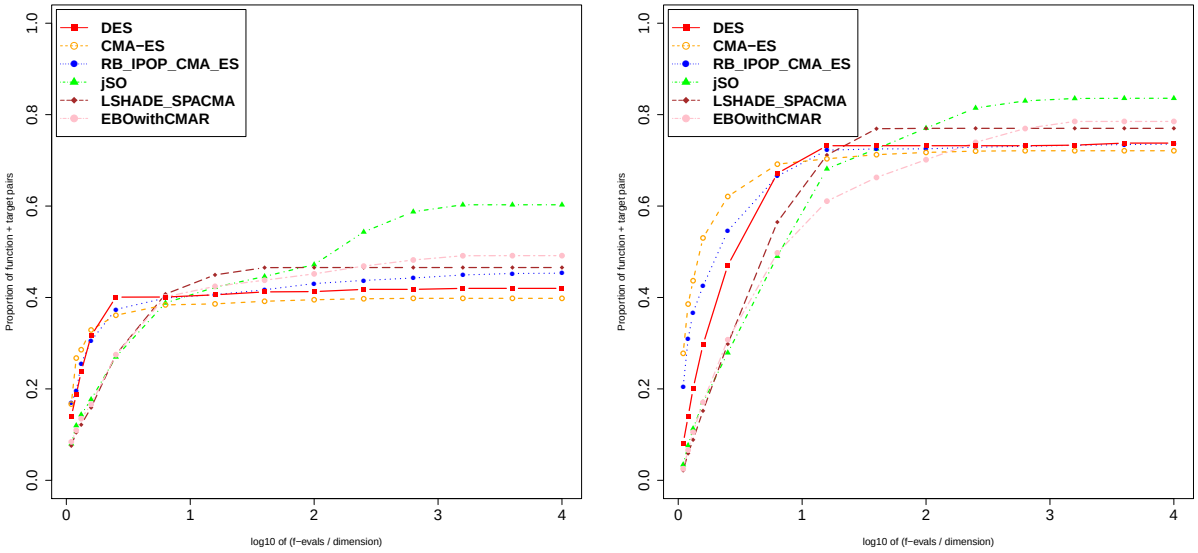


(a) n=10 (b) n=30



(c) n=50 (d) n=100

Rysunek B.11: Funkcja 11 benchmarku CEC.



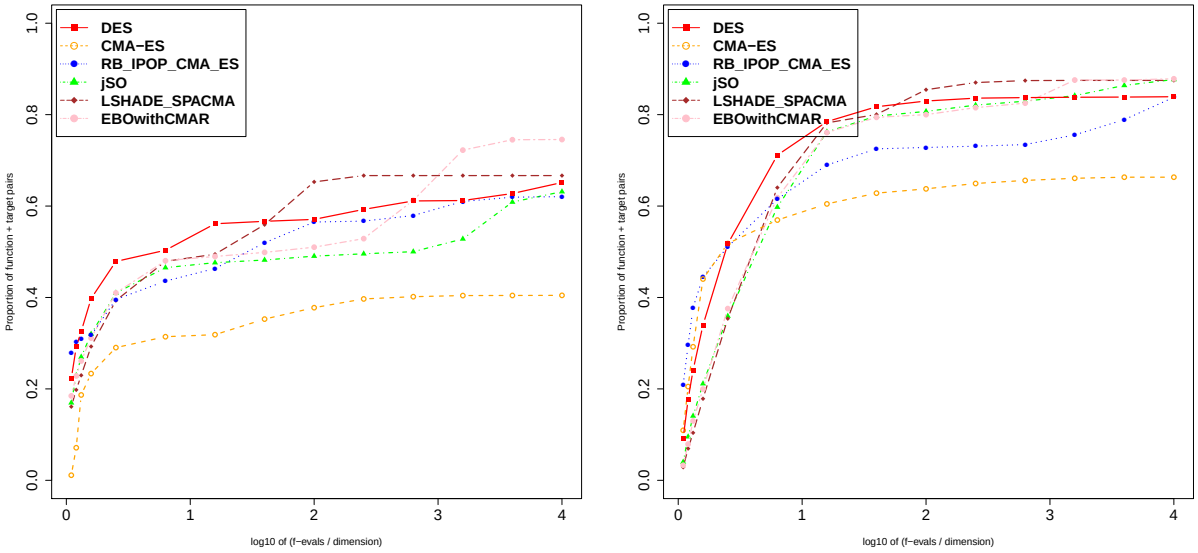
(a) n=10

(b) n=30

(c) n=50

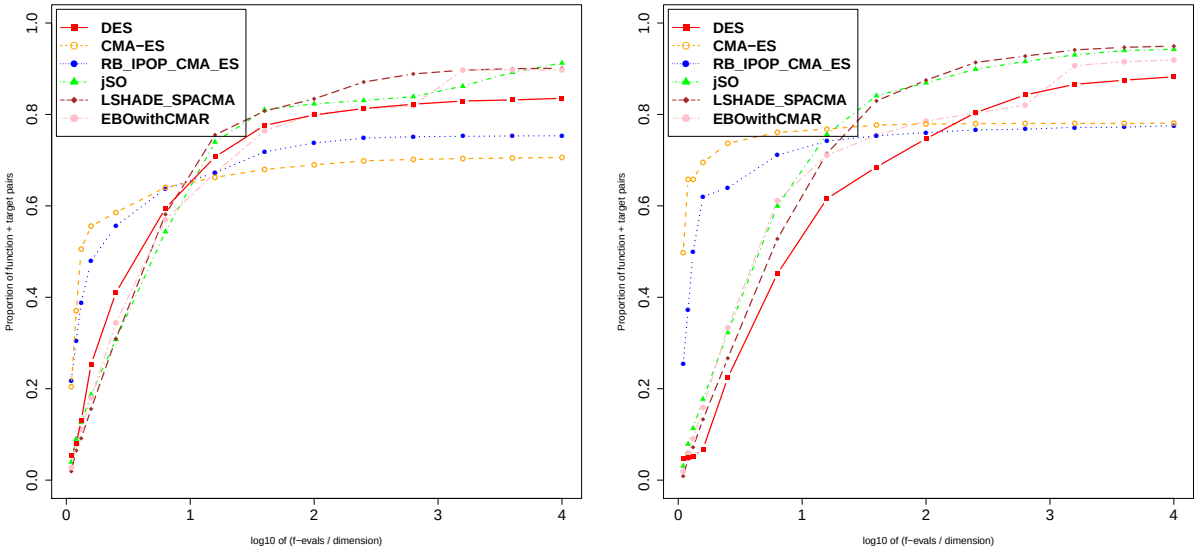
(d) n=100

Rysunek B.12: Funkcja 12 benchmarku CEC.



(a) n=10

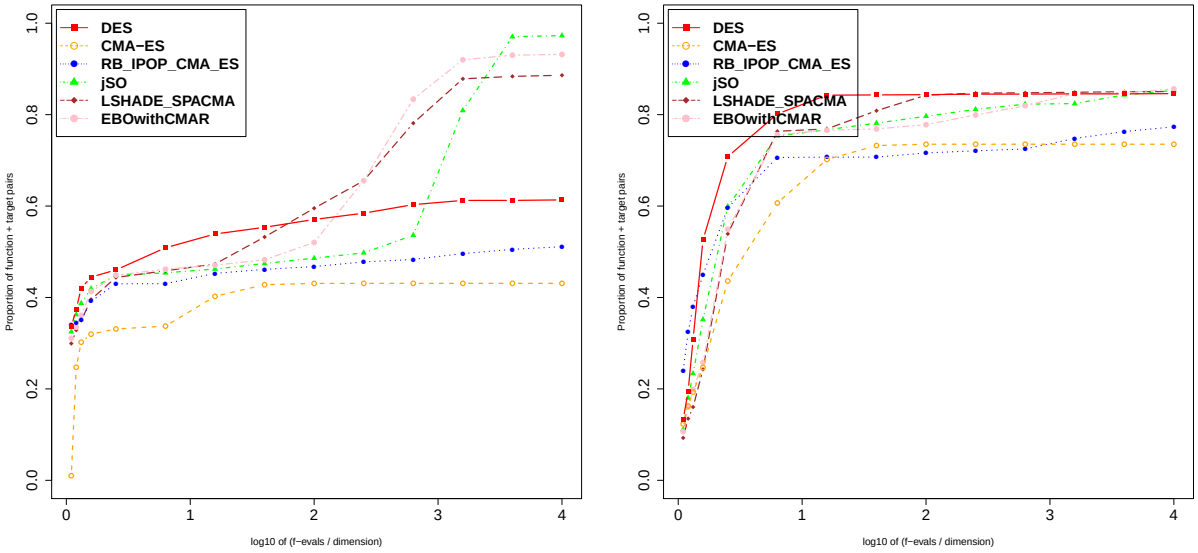
(b) n=30



(c) n=50

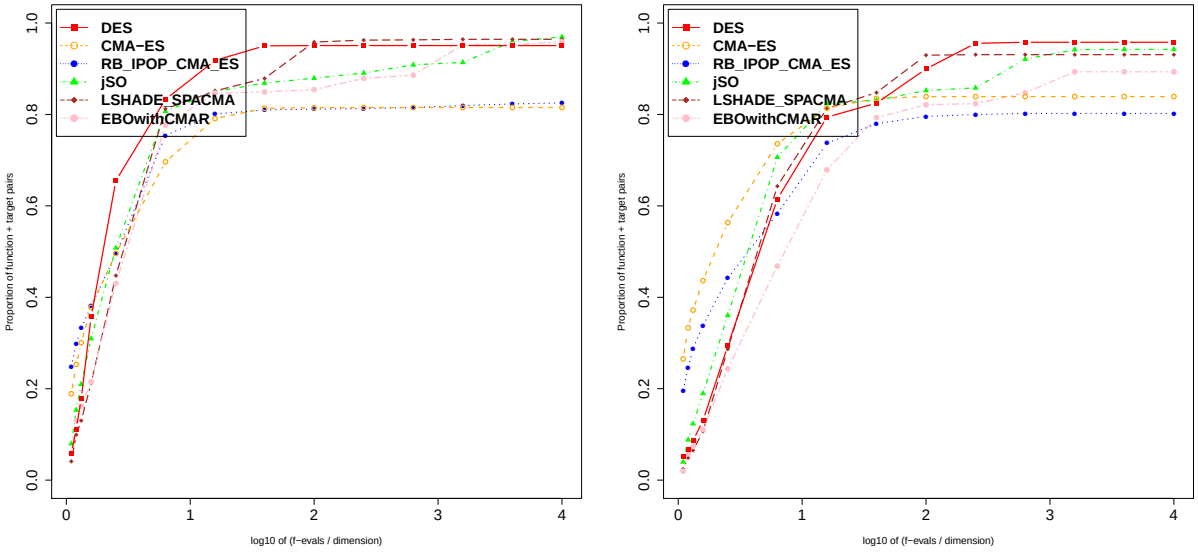
(d) n=100

Rysunek B.13: Funkcja 13 benchmarku CEC.



(a) n=10

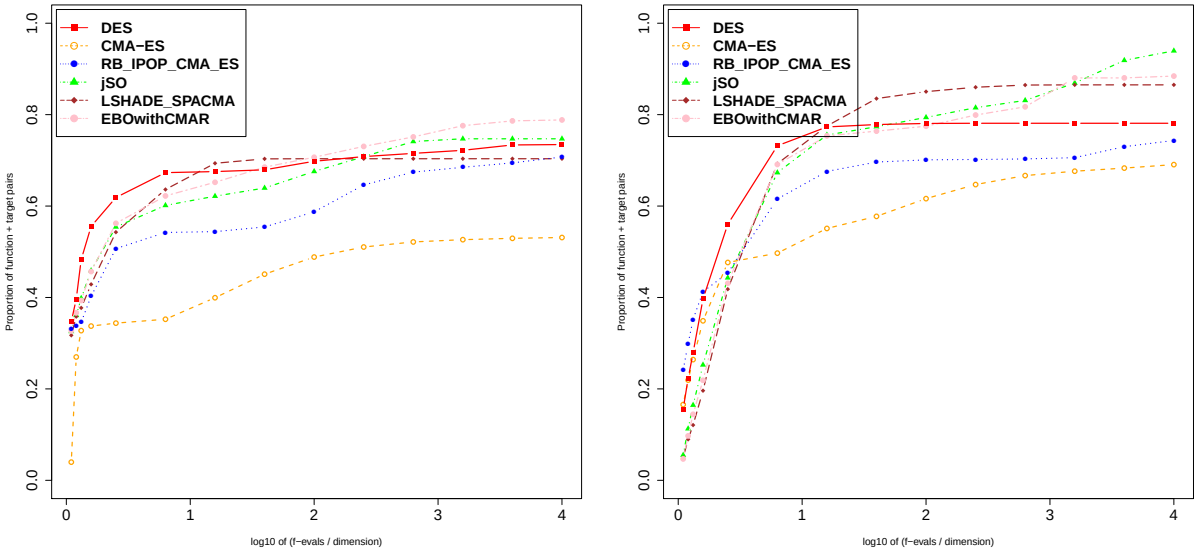
(b) n=30



(c) n=50

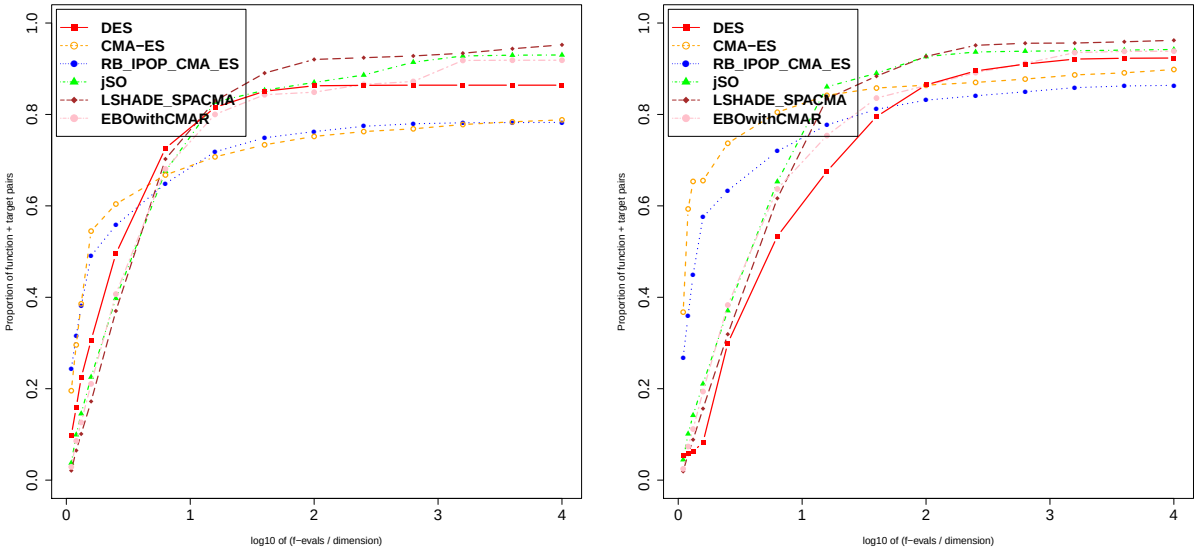
(d) n=100

Rysunek B.14: Funkcja 14 benchmarku CEC.



(a) n=10

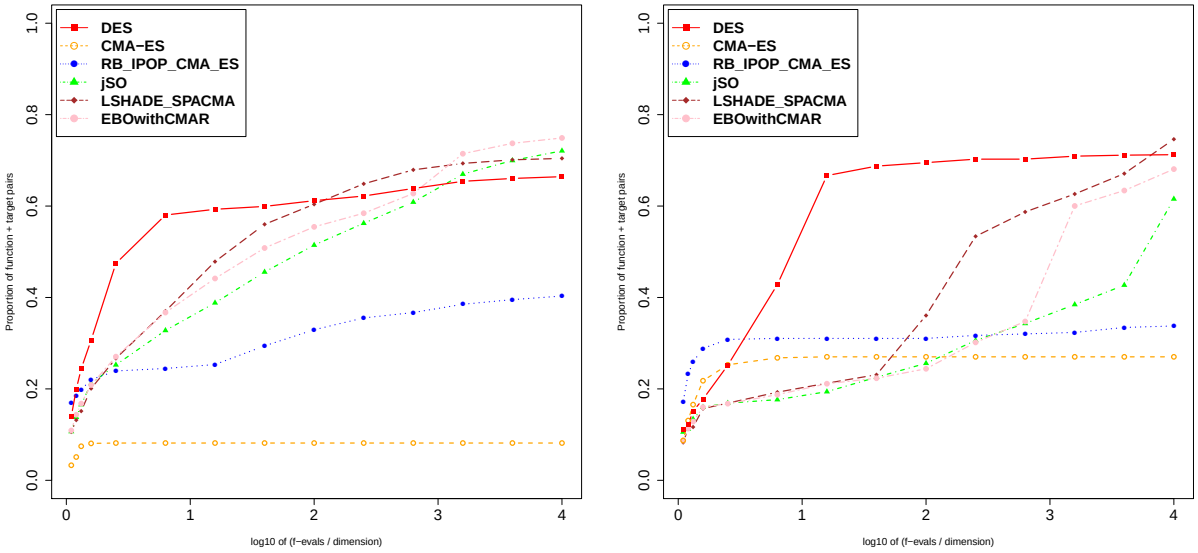
(b) n=30



(c) n=50

(d) n=100

Rysunek B.15: Funkcja 15 benchmarku CEC.



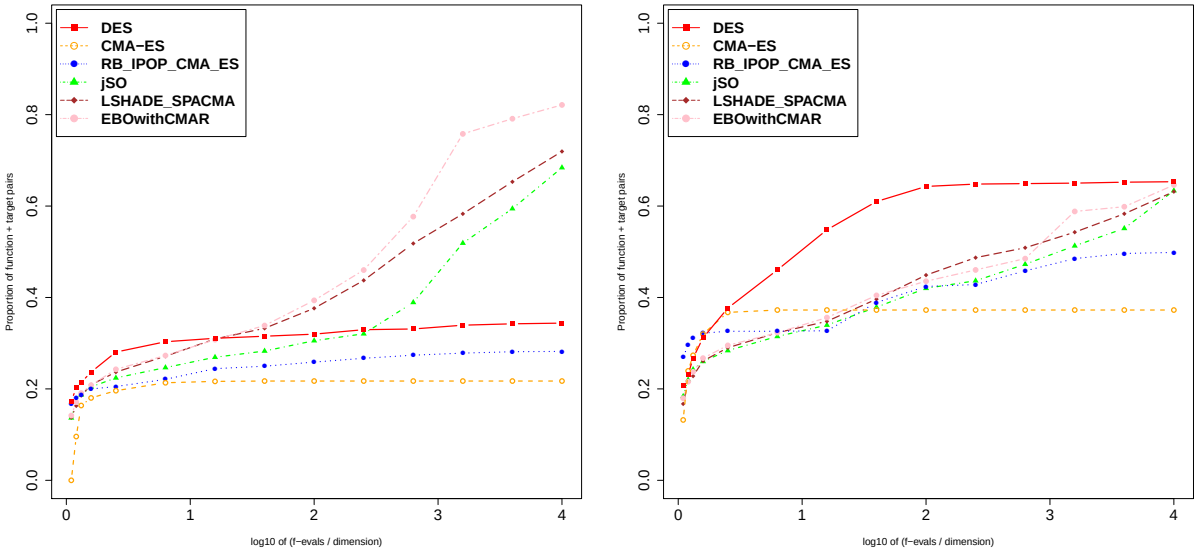
(a) n=10

(b) n=30

(c) n=50

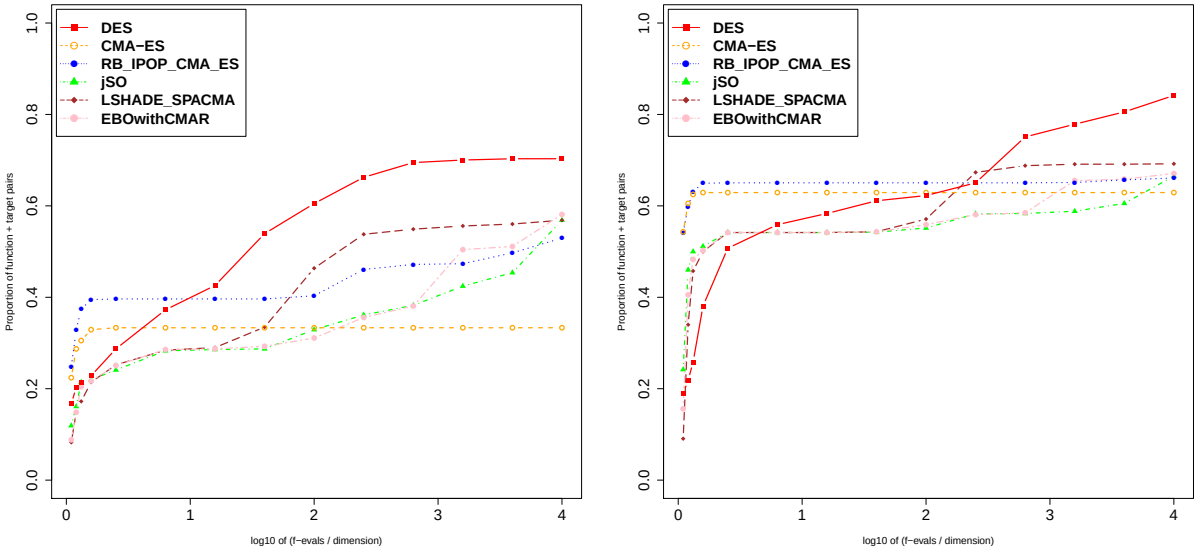
(d) n=100

Rysunek B.16: Funkcja 16 benchmarku CEC.



(a) n=10

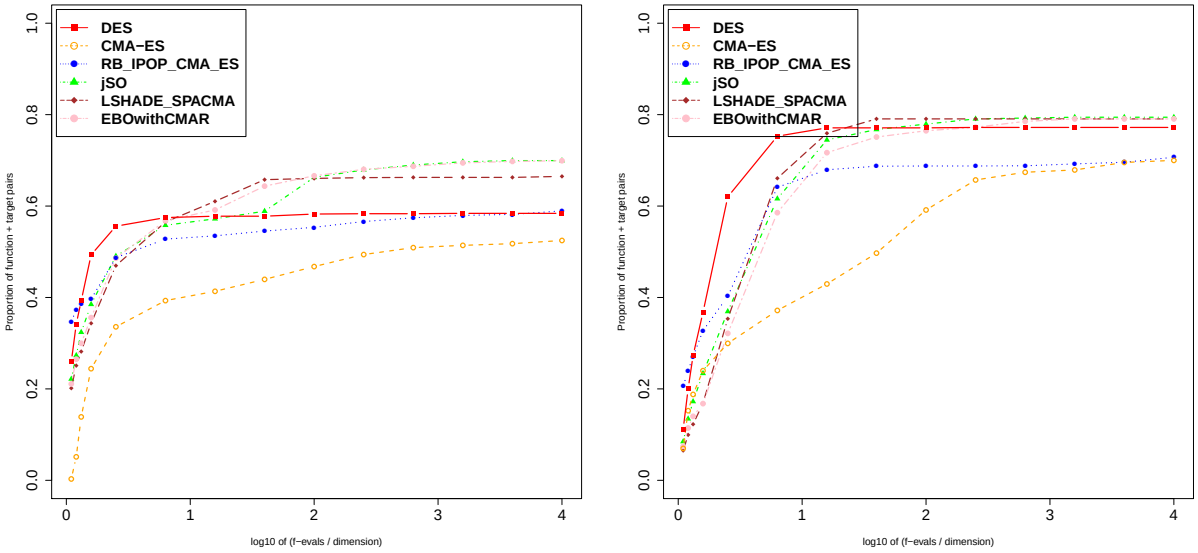
(b) n=30



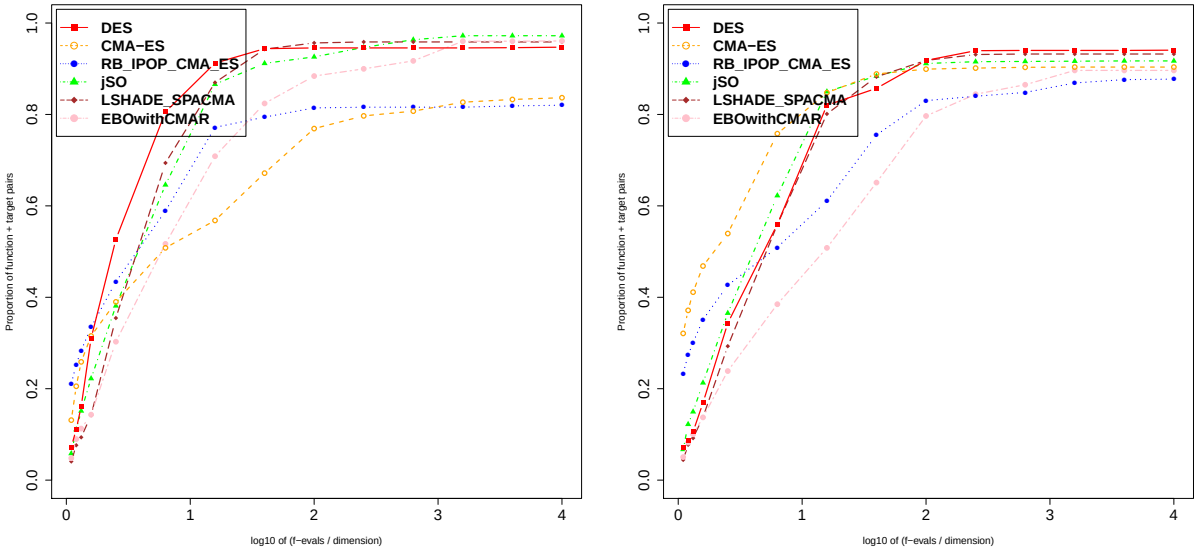
(c) n=50

(d) n=100

Rysunek B.17: Funkcja 17 benchmarku CEC.

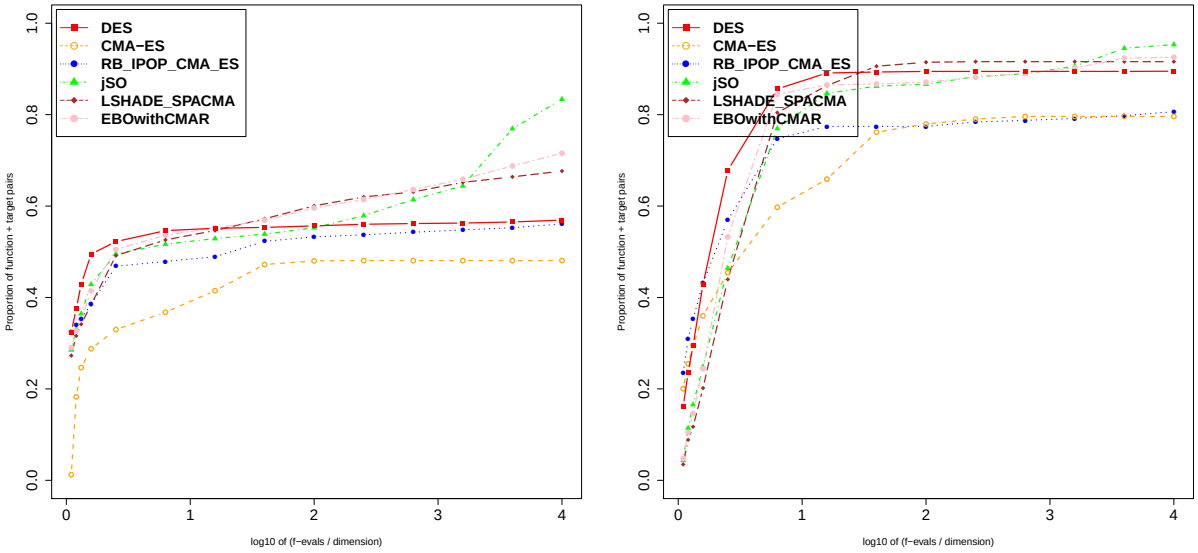


(a) n=10 (b) n=30



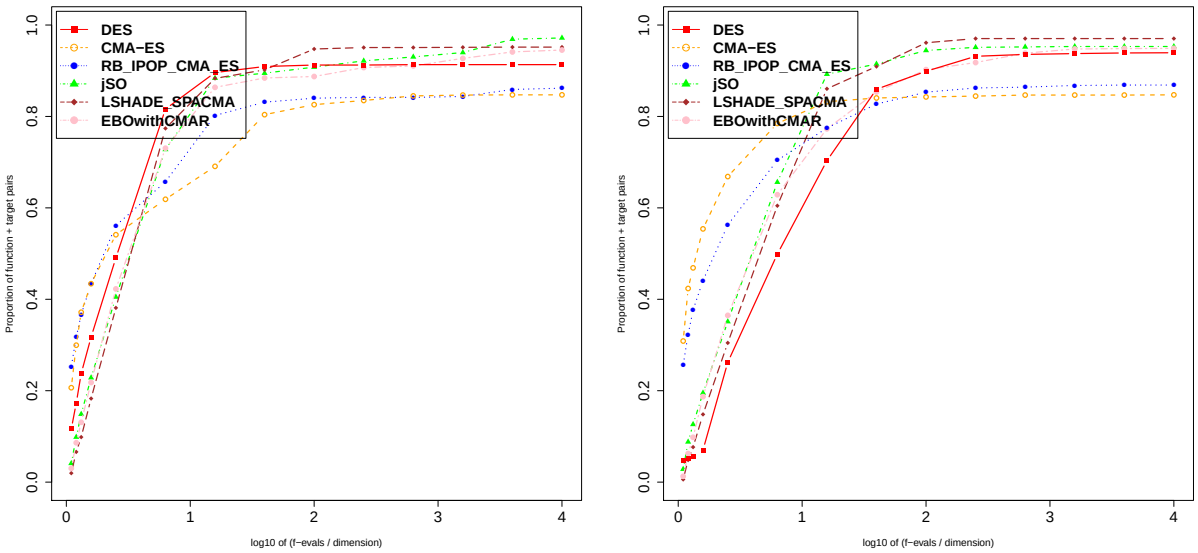
(c) n=50 (d) n=100

Rysunek B.18: Funkcja 18 benchmarku CEC.



(a) n=10

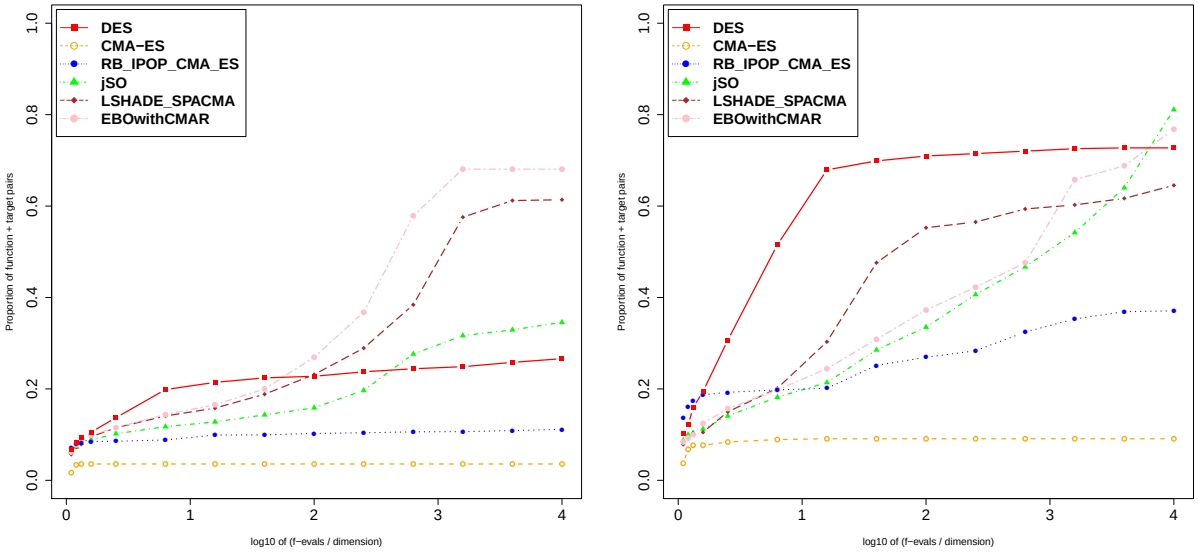
(b) n=30



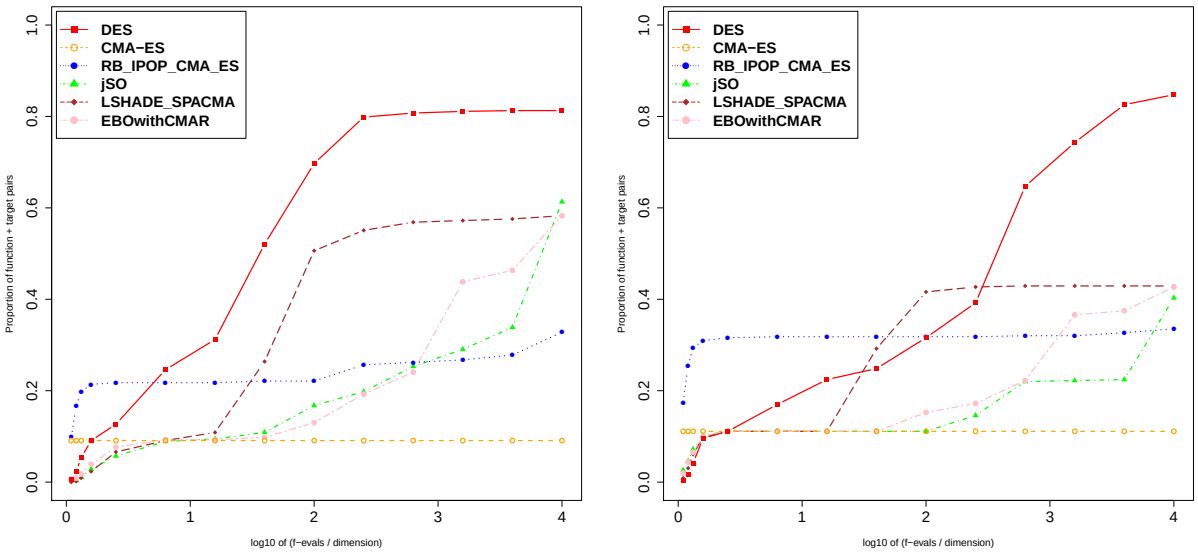
(c) n=50

(d) n=100

Rysunek B.19: Funkcja 19 benchmarku CEC.

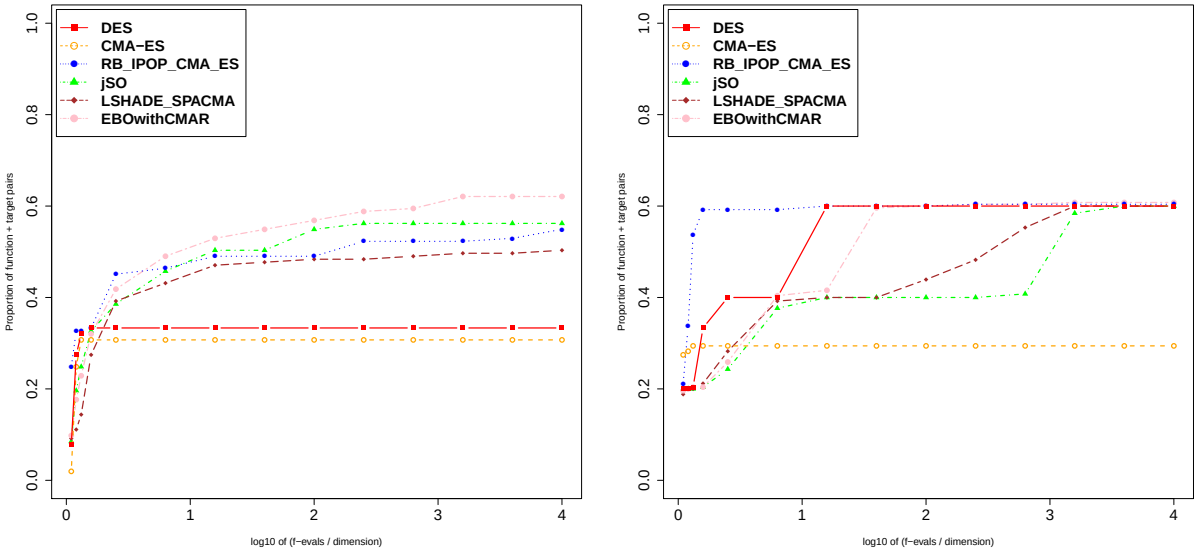


(a) n=10 (b) n=30



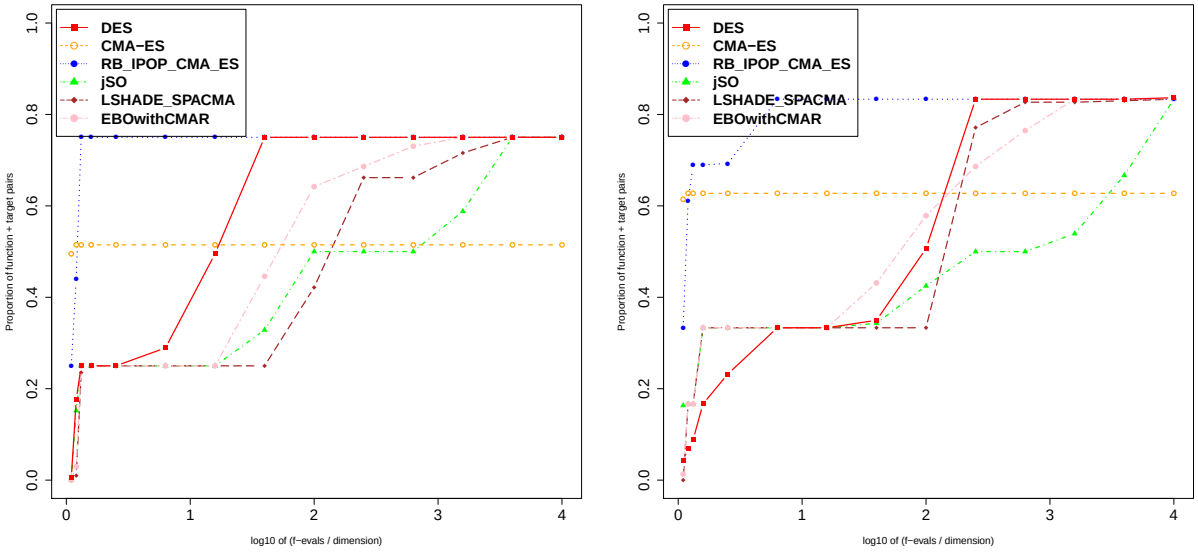
(c) n=50 (d) n=100

Rysunek B.20: Funkcja 20 benchmarku CEC.



(a) n=10

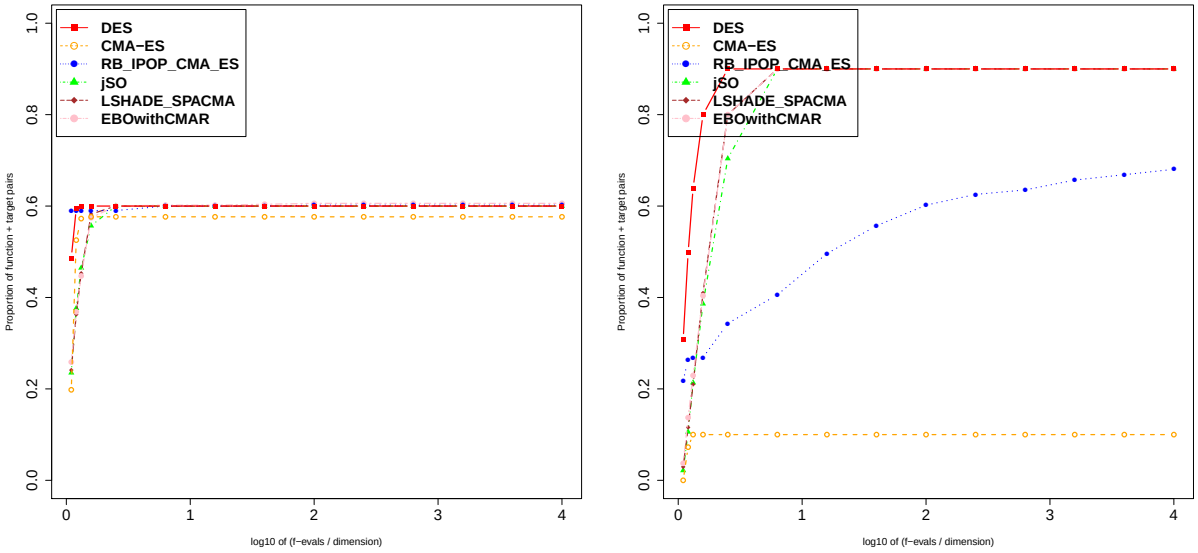
(b) n=30



(c) n=50

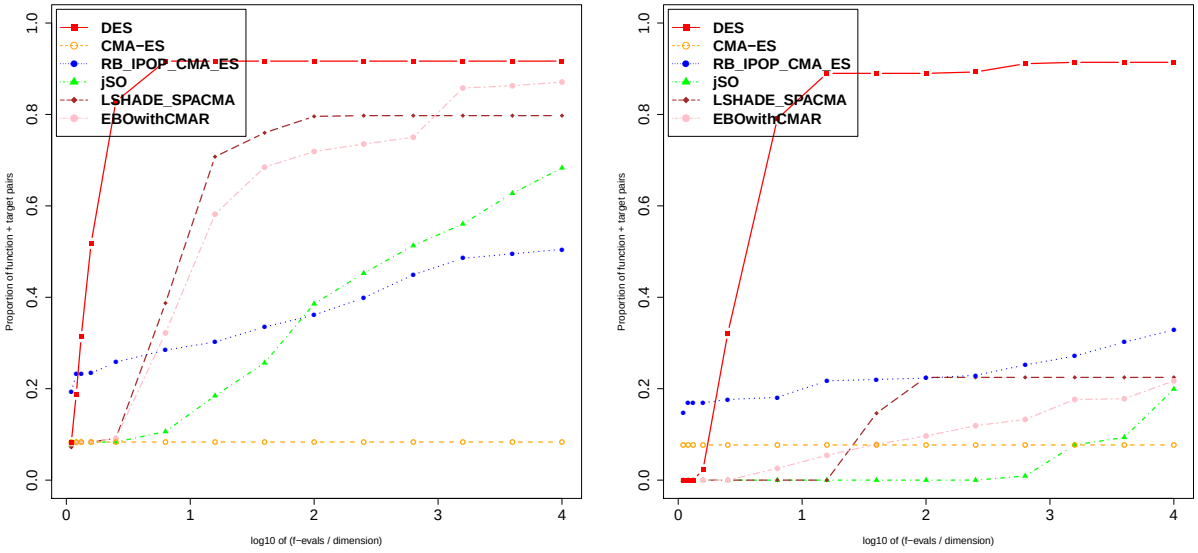
(d) n=100

Rysunek B.21: Funkcja 21 benchmarku CEC.



(a) n=10

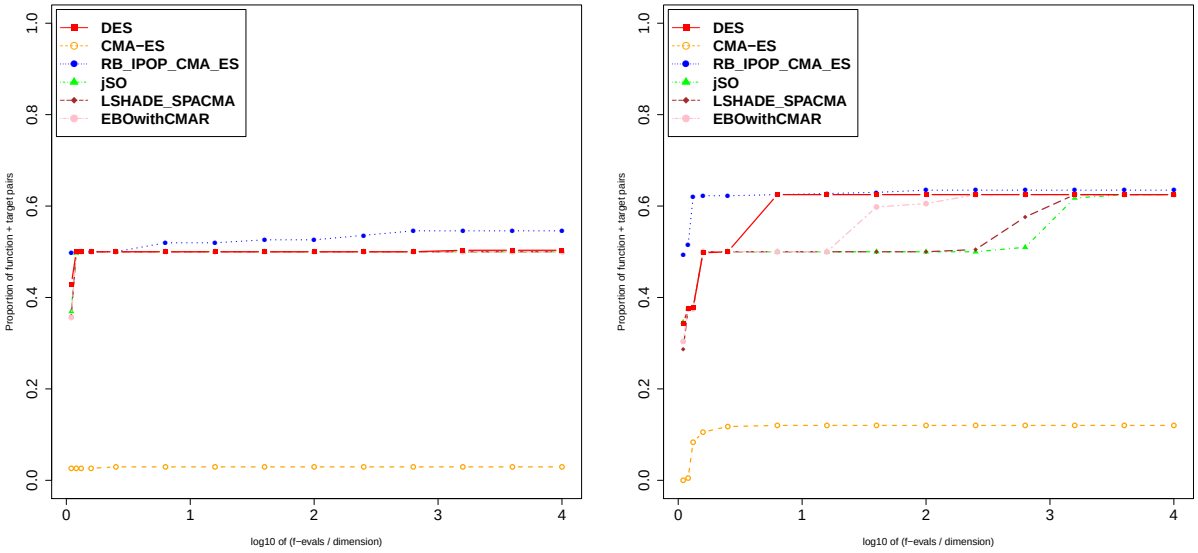
(b) n=30



(c) n=50

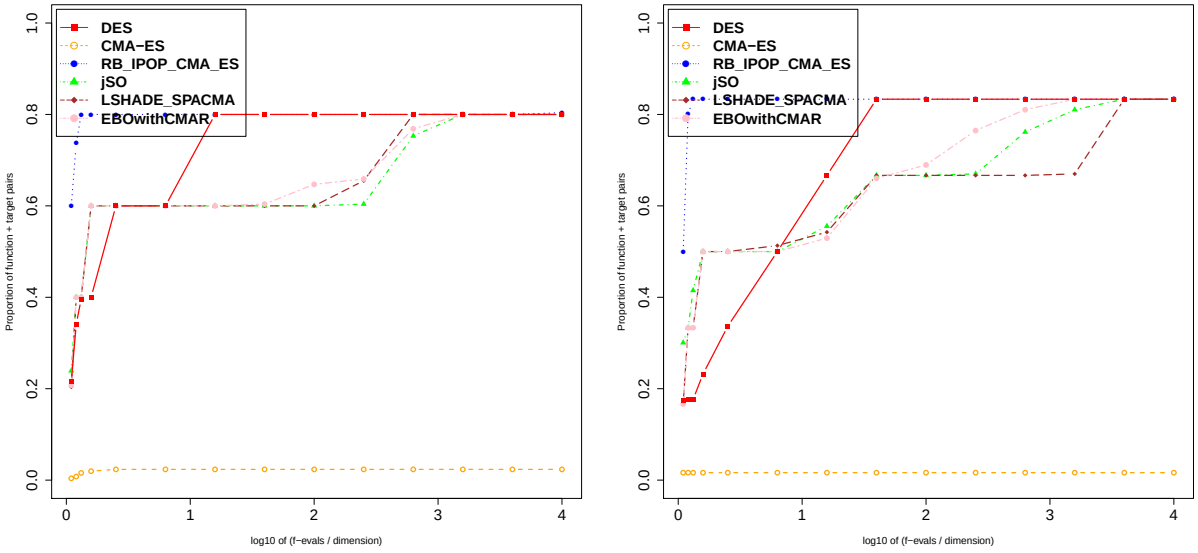
(d) n=100

Rysunek B.22: Funkcja 22 benchmarku CEC.



(a) n=10

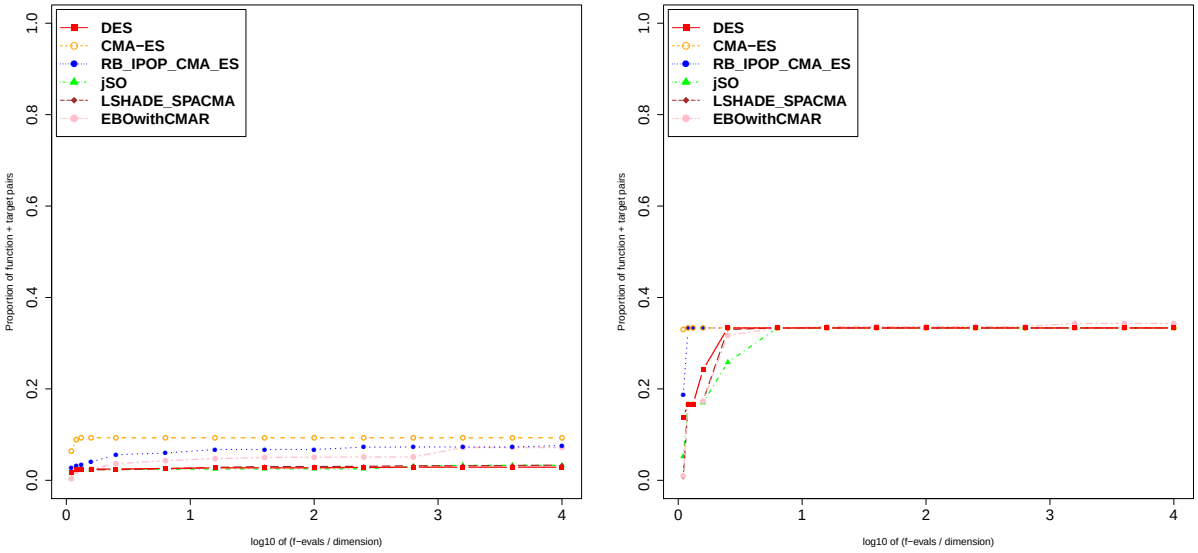
(b) n=30



(c) n=50

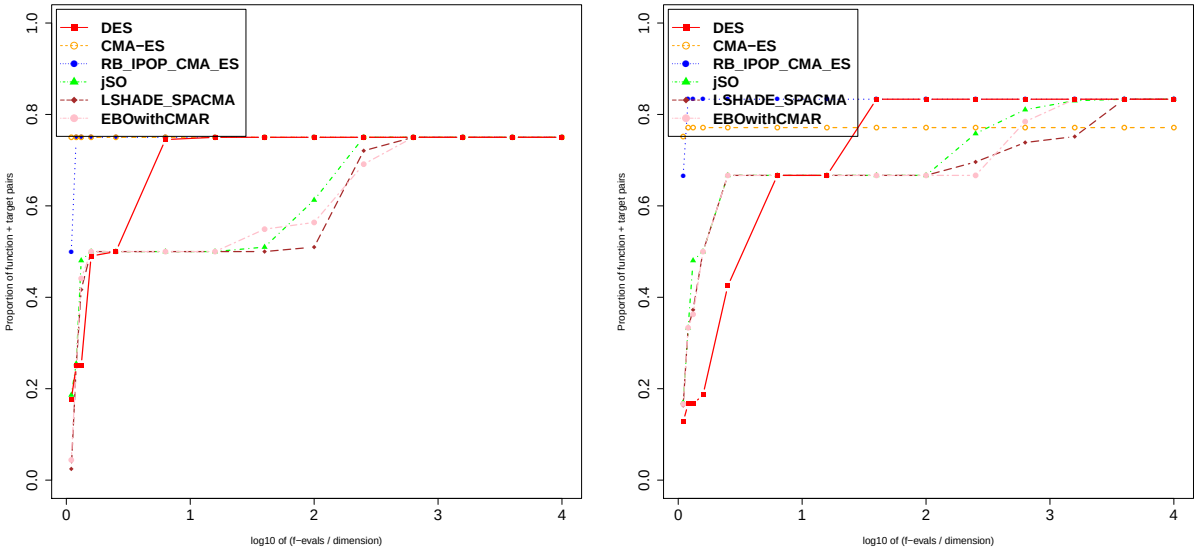
(d) n=100

Rysunek B.23: Funkcja 23 benchmarku CEC.



(a) n=10

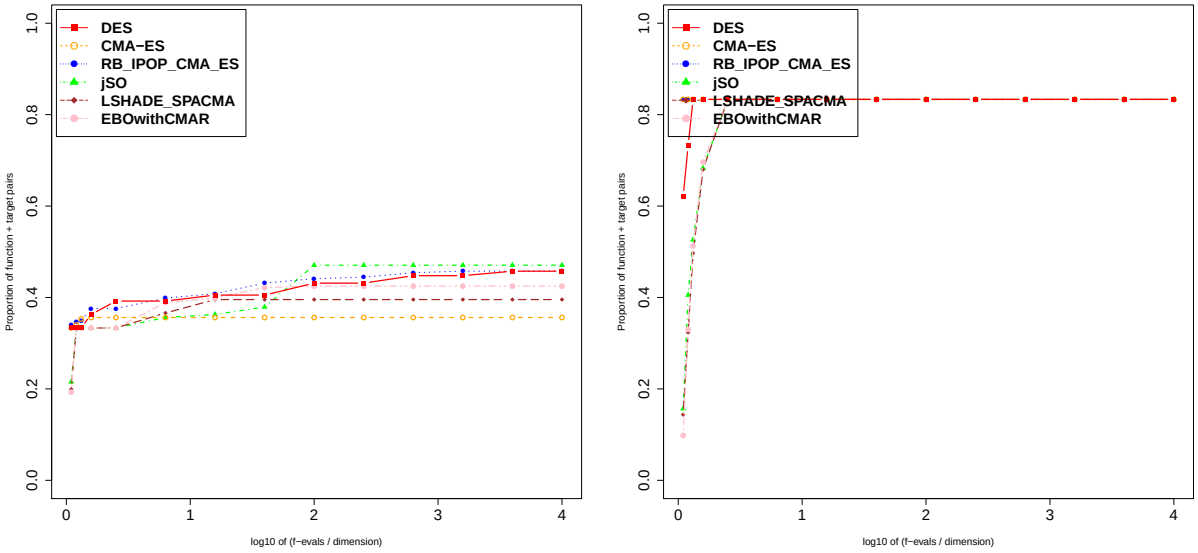
(b) n=30



(c) n=50

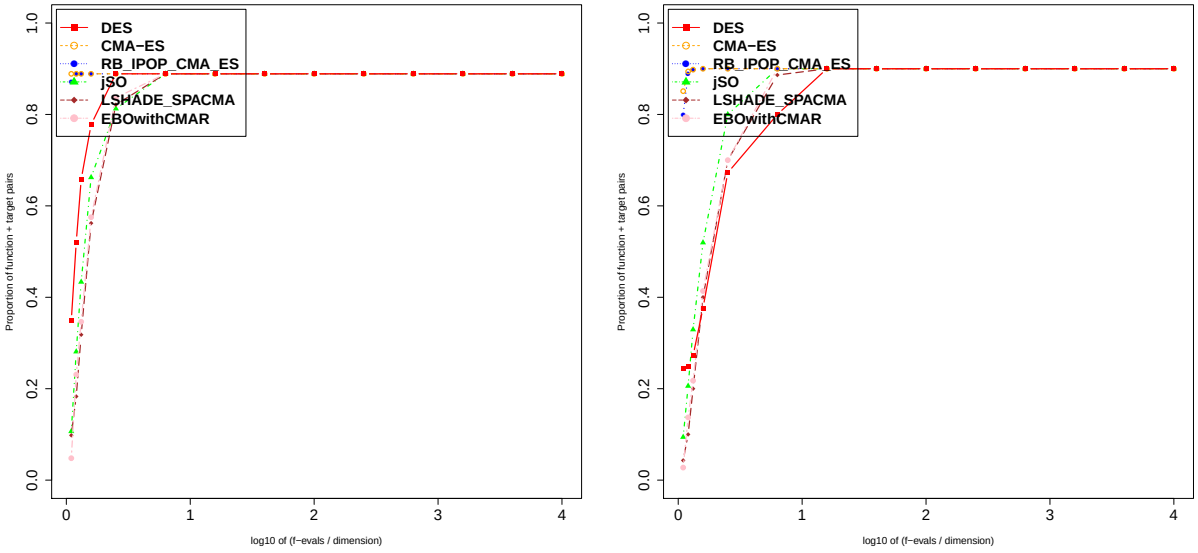
(d) n=100

Rysunek B.24: Funkcja 24 benchmarku CEC.



(a) n=10

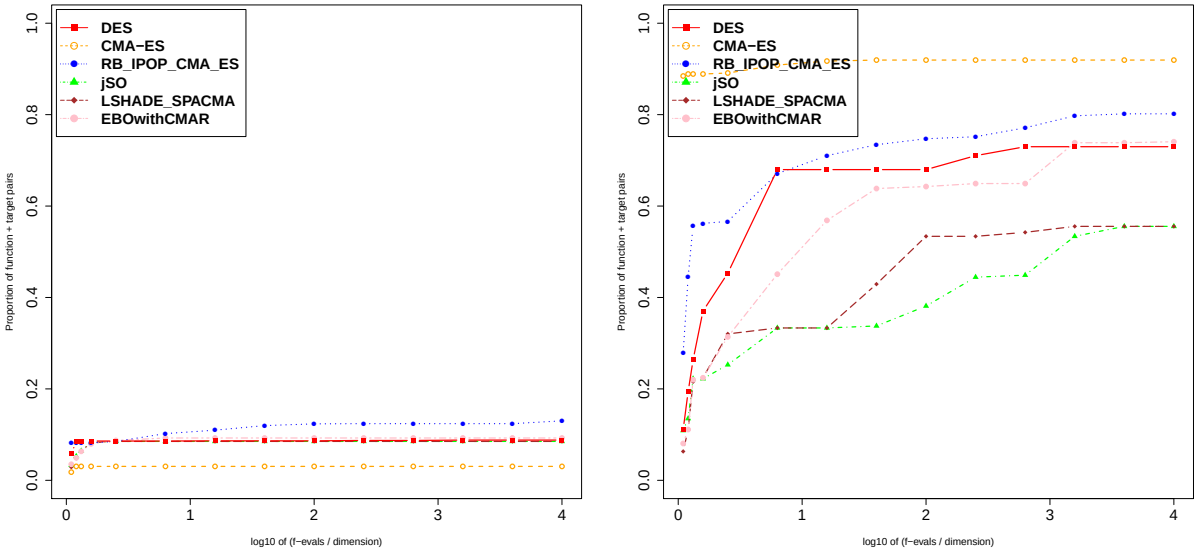
(b) n=30



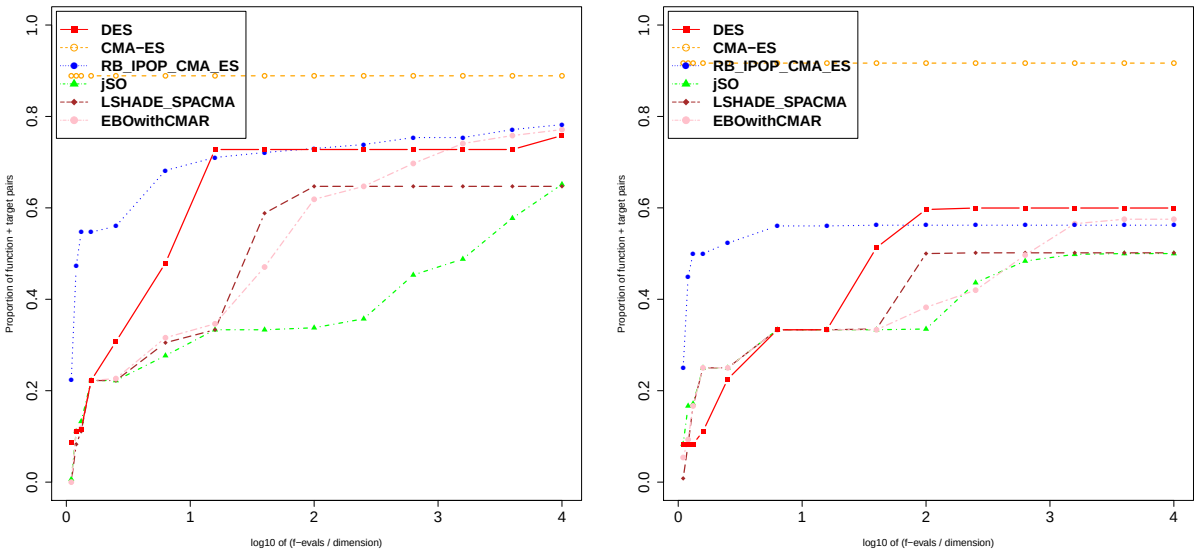
(c) n=50

(d) n=100

Rysunek B.25: Funkcja 25 benchmarku CEC.

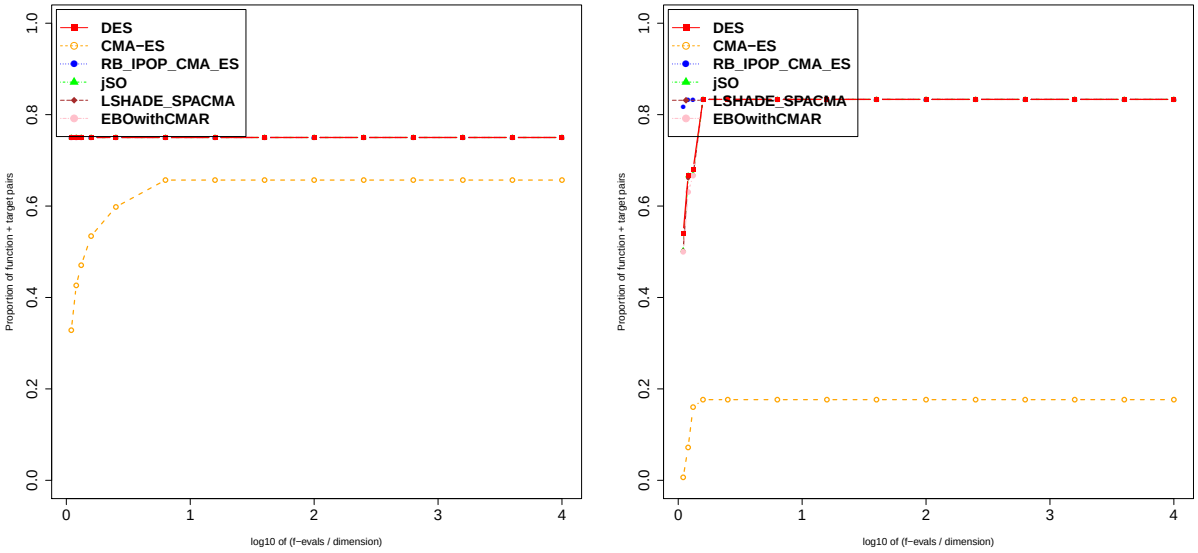


(a) n=10 (b) n=30



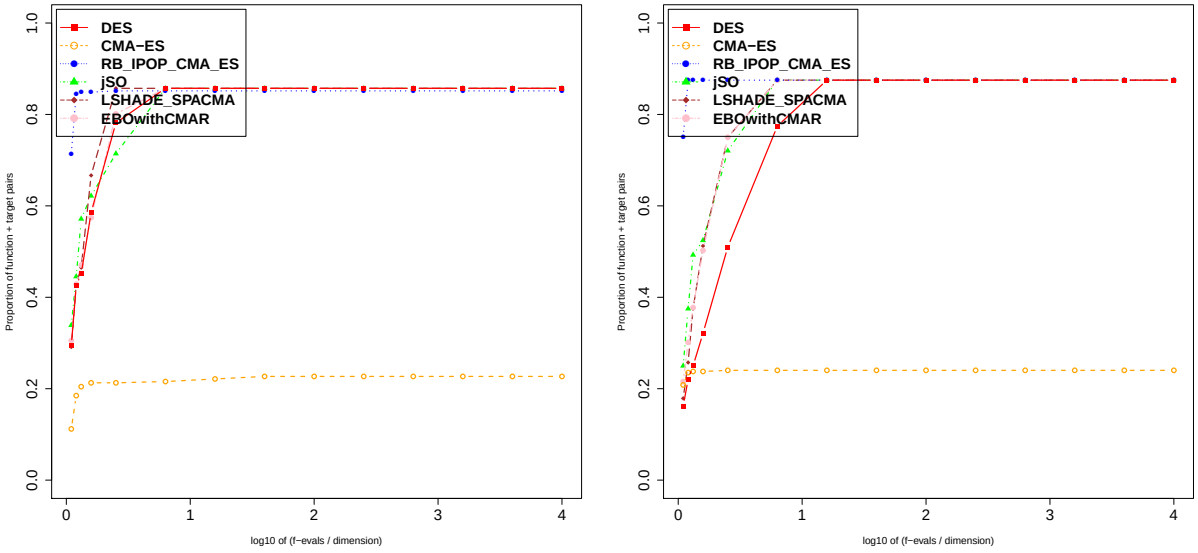
(c) n=50 (d) n=100

Rysunek B.26: Funkcja 26 benchmarku CEC.



(a) n=10

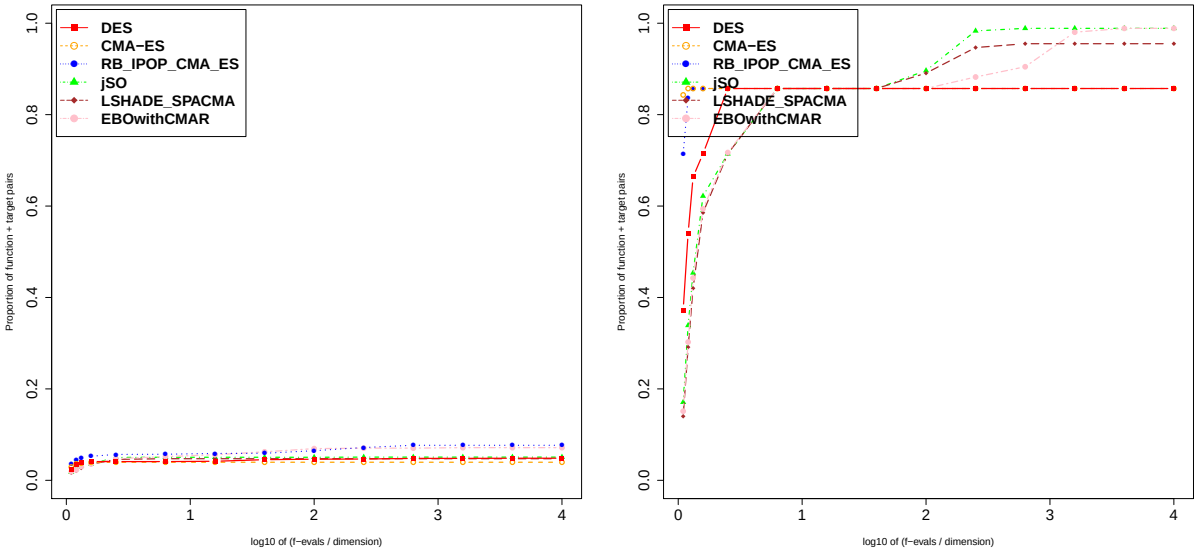
(b) n=30



(c) n=50

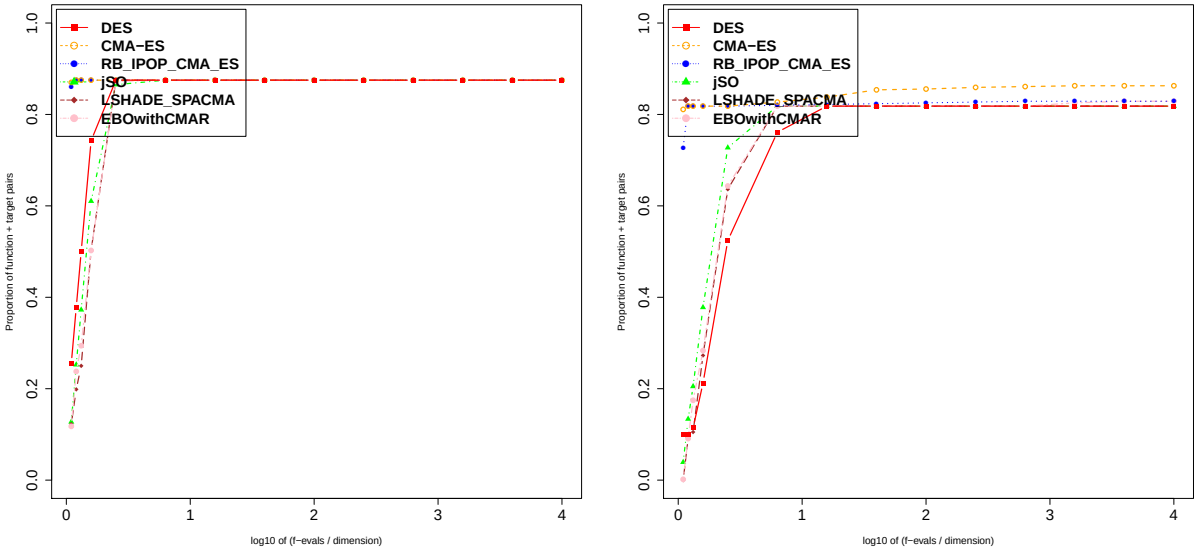
(d) n=100

Rysunek B.27: Funkcja 27 benchmarku CEC.



(a) n=10

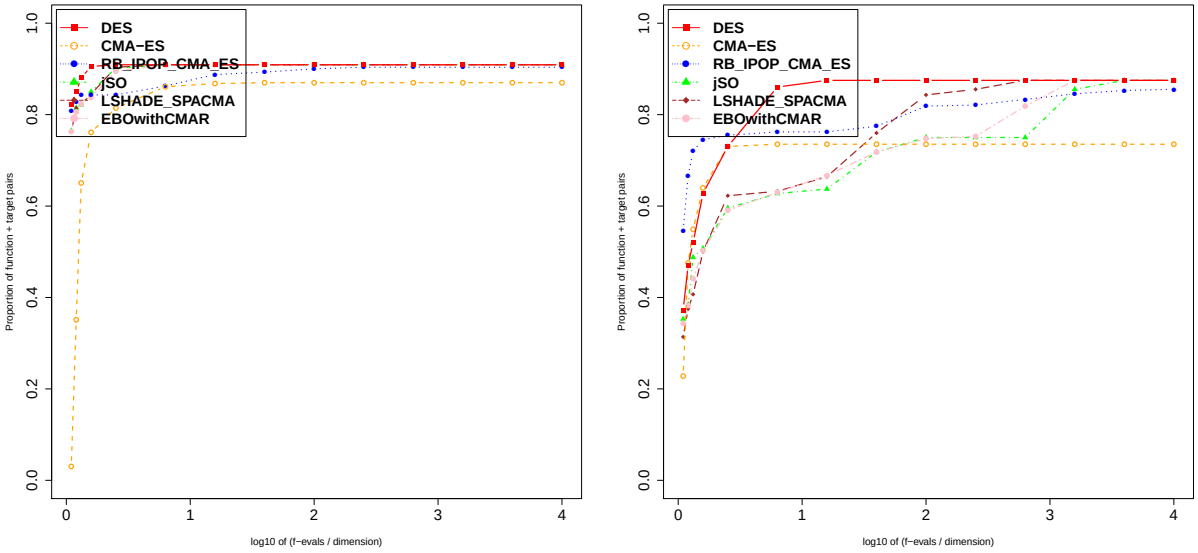
(b) n=30



(c) n=50

(d) n=100

Rysunek B.28: Funkcja 28 benchmarku CEC.



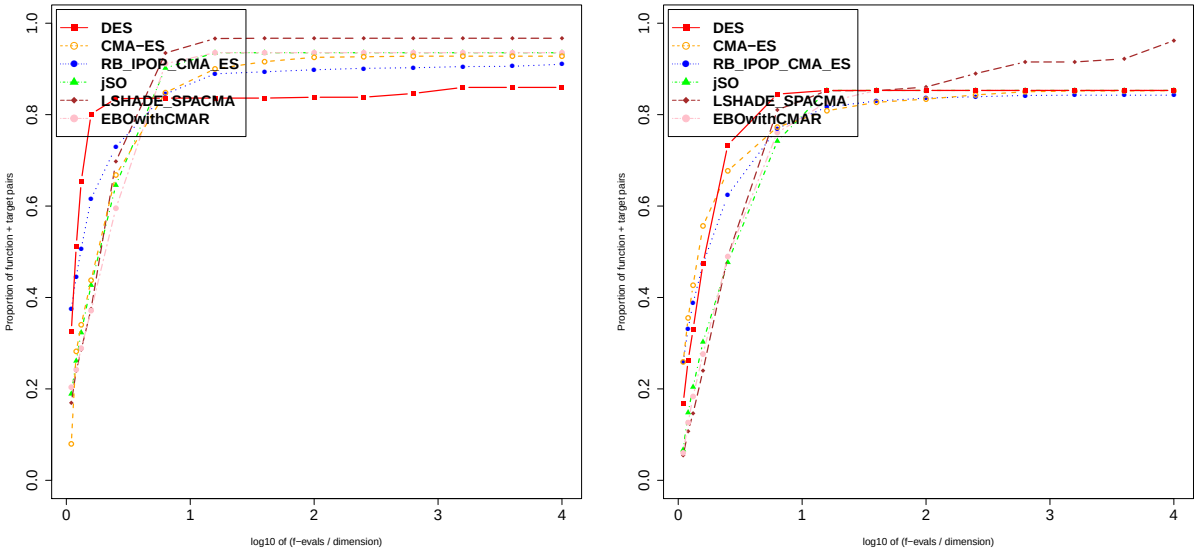
(a) n=10

(b) n=30

(c) n=50

(d) n=100

Rysunek B.29: Funkcja 29 benchmarku CEC.



(a) n=10

(b) n=30

(c) n=50

(d) n=100

Rysunek B.30: Funkcja 30 benchmarku CEC.