Rozszerzenie algorytmu MA-ES o heurystykę IPOP oraz zbadanie działania algorytmu poprzez porównanie z algorytmem CMA-ES.

Ireneusz Okniński, Damian Lubaszka[†]

Politechnika Warszawska, Instytut Informatyki, Nowowiejska 15/19, Warszawa, 00-665, Województwo Mazowieckie, Polska.

Współtwórcy: ireneusz.okninski.stud@pw.edu.pl;
damian.lubaszka.stud@pw.edu.pl;
[†]Autorzy wnieśli równy wkład w tę pracę.

Streszczenie

Celem projektu było rozszerzenie algorytmu Matrix Adaptation Evolution Strategy (MA-ES) o heurystykę Increasing Population Size (IPOP) oraz porównanie jego skuteczności z klasycznym algorytmem Covariance Matrix Adaptation Evolution Strategy (CMA-ES). W tym celu opracowano i zaimplementowano zmodyfikowaną wersję MA-ES umożliwiającą dynamiczne restartowanie procesu optymalizacji z powiększoną populacją w odpowiedzi na brak postępu. Porównanie algorytmów przeprowadzono przy użyciu benchmarku Black-Box Optimization Benchmarking (BBOB) w ramach frameworku COCO, uwzględniając różne funkcje testowe oraz wymiary przestrzeni poszukiwań. Analiza wyników opierała się na metrykach takich jak oczekiwany czas działania, odsetek udanych uruchomień oraz przebiegi wartości funkcji celu. Przeprowadzone eksperymenty potwierdziły, że rozszerzenie MA-ES o heurystykę IPOP może poprawić jego odporność na utknięcie w lokalnych minimach oraz zwiększyć skuteczność w trudniejszych przypadkach optymalizacyjnych.

Słowa kluczowe: MA-ES, IPOP, CMA-ES, BBOB, COCO, benchmark, metaheurystyka, optymalizacja

1 Wprowadzenie

Algorytmy ewolucyjne odgrywają istotną rolę w rozwiązywaniu problemów optymalizacji ciągłej, szczególnie tam, gdzie brak jest dostępu do informacji o pochodnych funkcji celu, a sama funkcja może być nieliniowa, szumowa lub mieć wiele minimów lokalnych. W takich przypadkach skuteczne eksplorowanie przestrzeni poszukiwań oraz unikanie pułapek lokalnych stanowi istotne wyzwanie.

Covariance Matrix Adaptation Evolution Strategy (CMA-ES)[1] to jeden z najbardziej znanych i skutecznych algorytmów optymalizacji stochastycznej, wykazujący wysoką jakość wyników w benchmarkach takich jak BBOB. Wadą CMA-ES jest jednak stosunkowo sześcienny koszt obliczeniowy związany z utrzymywaniem i aktualizacją pełnej macierzy kowariancji.

Alternatywą dla CMA-ES jest Matrix Adaptation Evolution Strategy (MA-ES)[2] – uproszczony wariant, który eliminuje konieczność zarządzania pełną macierzą kowariancji, zastępując ją prostszym mechanizmem adaptacji. Dzięki temu MA-ES charakteryzuje się mniejszą złożonością obliczeniową, starając się przy tym zachowywać konkurencyjną jakość rozwiązań.

W niniejszym projekcie zaproponowano rozszerzenie MA-ES o heurystykę Increasing Population Size (IPOP)[3] , której celem jest zwiększenie odporności algorytmu na utknięcie w minimach lokalnych poprzez restartowanie procesu optymalizacji z większą populacją. Mechanizm ten był wcześniej z powodzeniem stosowany w kontekście CMA-ES jednak w literaturze brak jest analiz dotyczących jego zastosowania w uproszczonym MA-ES.

W ramach pracy opracowano i zaimplementowano zmodyfikowany algorytm MA-ES z heurystyką IPOP, a następnie przeprowadzono jego porównanie ze standardowym algorytmem CMA-ES przy użyciu benchmarku BBOB udostępnianego w ramach frameworku COCO[4]. W eksperymentach uwzględniono różne funkcje testowe i wymiary przestrzeni poszukiwań, a wyniki oceniono przy użyciu standardowych metryk takich jak czas działania oraz skuteczność optymalizacji.

2 CMA-ES

2.1 Idea algorytmu

Algorytm *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) należy do klasy strategii ewolucyjnych, które iteracyjnie aktualizują rozkład prawdopodobieństwa używany do próbkowania punktów w przestrzeni rozwiązań. W szczególności, CMA-ES stosuje adaptację macierzy kowariancji, co pozwala mu efektywnie eksplorować przestrzeń wzdłuż najbardziej obiecujących kierunków.

Jedną z kluczowych cech CMA-ES jest jego zdolność do dostosowania rozmiaru i orientacji elipsoidy próbkowania w odpowiedzi na lokalną strukturę krajobrazu funkcji celu. Dzięki temu algorytm skutecznie radzi sobie z:

- funkcjami niestacjonarnymi, o silnie zakrzywionej topologii (np. wąskie doliny),
- optymalizacją funkcji z wieloma lokalnymi ekstremami (robustność względem lokalnych minimów),

- problemami skalowanymi i nieliniowymi, w których klasyczne metody gradientowe zawodzą.

CMA-ES nie wymaga znajomości gradientu ani innych pochodnych funkcji celu, co czyni go bardzo uniwersalnym w kontekście optymalizacji czarnej skrzynki. Z drugiej strony, algorytm jest kosztowny obliczeniowo ze względu na konieczność rozkładu macierzy kowariancji, co skutkuje złożonością obliczeniową rzędu $\mathcal{O}(n^3)$ na iterację.

2.2 Pseudokod

Poniżej przedstawiono pseudokod algorytmu CMA-ES, oparty na materiałach literaturowych oraz materiałach dydaktycznych wykorzystywanych podczas zajęć[5]. Algorytm został zaimplementowany samodzielnie, bez korzystania z gotowych bibliotek czy implementacji open-source. Poszczególne wzory zostały odwzorowane bezpośrednio w implementacji.

Algorithm 1 CMA-ES

Set parameters $\lambda, \mu, w_i, c_\sigma, d_\sigma, c_c, c_1, c_\mu$ according to Table 1
Initialize evolution paths $p_\sigma = 0$, $p_c = 0$, covariance matrix $C = I$, and generation counter $g = 0$
Choose mean $m \in \mathbb{R}^n$ and step-size $\sigma \in \mathbb{R}_{>0}$
while termination criterion not met **do**
    $g \leftarrow g + 1$
    for $k = 1$ to λ **do**
        $z_k \sim \mathcal{N}(0, I)$
        $y_k \leftarrow BDz_k \sim \mathcal{N}(0, C)$
        $x_k \leftarrow m + \sigma y_k \sim \mathcal{N}(m, \sigma^2 C)$
    end for
    Evaluate x_k and sort by fitness
    $\langle y \rangle_w \leftarrow \sum_{i=1}^{\mu} w_i y_{i:\lambda}$ ▷ weighted recombination
    $m \leftarrow m + c_m \sigma \langle y \rangle_w$
    $p_\sigma \leftarrow (1 - c_\sigma) p_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}} C^{-1/2} \langle y \rangle_w$
    $\sigma \leftarrow \sigma \cdot \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma\|}{E\|\mathcal{N}(0,I)\|} - 1\right)\right)$
    $h_\sigma \leftarrow \mathbf{1}\left(\frac{\|p_\sigma\|}{\sqrt{1-(1-c_\sigma)^{2g+1}}} < (1.4 + \frac{2}{n+1}) E\|\mathcal{N}(0,I)\|\right)$
    $p_c \leftarrow (1 - c_c) p_c + h_\sigma \sqrt{c_c(2 - c_c)\mu_{\text{eff}}} \langle y \rangle_w$
    $C \leftarrow (1 - c_1 - c_\mu) C + c_1 (p_c p_c^\top) + c_\mu \sum_{i=1}^{\mu} w_i y_{i:\lambda} y_{i:\lambda}^\top$
end while

2.3 Zastosowane współczynniki

Wartości parametrów zastosowanych w implementacji zostały przyjęte zgodnie z zaleceniami z literatury[6], jako że zostały one zaproponowane na podstawie szeroko zakrojonych eksperymentów optymalizacyjnych i analiz teoretycznych. W szczególności przyjęto:

- $\lambda = 4 + \lfloor 3 \ln n \rfloor$ – liczność populacji,
- $w_i = \ln\left(\frac{\lambda+1}{2}\right) - \ln i$ – wagi selekcji,
- $\mu_{\text{eff}} = \frac{1}{\sum w_i^2}$ – efektywna wielkość populacji,
- $c_\sigma = \frac{\mu_{\text{eff}}+2}{n+\mu_{\text{eff}}+5}$ – współczynnik ścieżki kroku,
- $d_\sigma = 1 + c_\sigma + 2\max\left(0, \sqrt{\frac{\mu_{\text{eff}}-1}{n+1}} - 1\right)$ – parametr odpowiedzialny za kontrolę kroku,
- $c_c = \frac{4+\mu_{\text{eff}}/n}{n+4+2\mu_{\text{eff}}/n}$ – współczynnik ścieżki kowariancji,
- $c_1 = \frac{2}{(n+1.3)^2+\mu_{\text{eff}}}$ – waga aktualizacji rank-one,
- $c_\mu = \min\left(1 - c_1, \frac{2(\mu_{\text{eff}}-2+1/\mu_{\text{eff}})}{(n+2)^2+\mu_{\text{eff}}}\right)$ – waga aktualizacji rank-μ,
- $c_m = 1$ – szybkość zmiany średniej rozkładu.

Zastosowanie tych wartości pozwala zapewnić stabilność i skuteczność algorytmu w typowych problemach benchmarkowych, takich jak funkcje z rodziny BBOB. Próby doboru współczynników alternatywnych najczęściej skutkowały pogorszeniem jakości działania algorytmu, dlatego zdecydowano się pozostać przy wartościach ustalonych przez ekspertów w dziedzinie.

2.4 MA-ES

2.4.1 Idea algorytmu

Matrix Adaptation Evolution Strategy (MA-ES) to uproszczona wersja klasycznego algorytmu CMA-ES, która eliminuje potrzebę przechowywania oraz aktualizacji pełnej macierzy kowariancji Głównym założeniem MA-ES jest unikanie kosztownych obliczeniowo operacji związanych z utrzymywaniem i aktualizowaniem pełnej macierzy kowariancji (jak ma to miejsce w klasycznym CMA-ES). Zamiast tego, MA-ES wykorzystuje pojedynczą macierz M, która pełni rolę liniowej transformacji wektora losowego próbkowanego z rozkładu normalnego.

Ważnym aspektem algorytmu jest mechanizm aktualizacji macierzy M, który odbywa się poprzez prostą, efektywną numerycznie regułę opartą na śledzeniu średnich kierunków ruchu w przestrzeni rozwiązań. Zamiast jawnej aktualizacji macierzy kowariancji, MA-ES modyfikuje M poprzez kombinację zmian ewolucyjnych ścieżek oraz rozkładów próbkowanych wektorów. Pozwala to zbliżyć się do zachowania klasycznego CMA-ES, ale przy znacznie mniejszych kosztach obliczeniowych.

Algorytm nie wymaga rozkładu macierzy, a jego operacje sprowadzają się do przemnożeń macierz-wektor oraz macierz-macierz. To sprawia, że MA-ES jest szczególnie atrakcyjny dla problemów o dużej liczbie wymiarów, gdzie klasyczne podejścia mogą być zbyt wolne, jednakże uproszczona budowa algorytmu ma też negatywny wpływ na jakość rozwiązań.

Macierz M osiąga stan stacjonarny wtedy, gdy wylosowane wektory po transformacji są zbliżone do standaryzowanego rozkładu normalnego, co oznacza, że strategia ustabilizowała się i efektywnie dopasowała do topologii funkcji celu.

2.4.2 Pseudokod

Poniżej przedstawiono pseudokod algorytmu MA-ES, oparty na materiałach literaturowych. Algorytm został zaimplementowany samodzielnie, bez korzystania z gotowych bibliotek czy implementacji open-source. Poszczególne wzory zostały odwzorowane bezpośrednio w implementacji.

Algorithm 2 MA-ES

Set parameters $\lambda, \mu, w_i, c_\sigma, d_\sigma, c_s, c_1, c_\mu$
Initialize evolution path $p_s = 0$, transformation matrix $M = I$, generation counter $g = 0$
Choose initial mean $m \in \mathbb{R}^n$ and step-size $\sigma \in \mathbb{R}_{>0}$
while termination criterion not met **do**
    $g \leftarrow g + 1$
    for $k = 1$ to λ **do**
        $z_k \sim \mathcal{N}(0, I)$
        $y_k \leftarrow M z_k$
        $x_k \leftarrow m + \sigma y_k$
    end for
    Evaluate x_k and sort by fitness
    Compute weighted recombination: $\langle y \rangle_w \leftarrow \sum_{i=1}^{\mu} w_i y_{i:\lambda}$
    $m \leftarrow m + \sigma \langle y \rangle_w$
    $\langle z \rangle_w \leftarrow \sum_{i=1}^{\mu} w_i z_{i:\lambda}$
    $p_s \leftarrow (1 - c_s) p_s + \sqrt{c_s(2 - c_s)\mu_{\text{eff}}} \langle z \rangle_w$
    $\sigma \leftarrow \sigma \cdot \exp\left(\frac{c_s}{d_\sigma} \left(\frac{\|p_s\|}{\sqrt{n}} - 1 \right) \right)$
    $M \leftarrow M(I + \frac{c_1}{2}(p_s p_s^\top - I) + \frac{c_\mu}{2} \sum_{i=1}^{\mu} w_i (z_{i:\lambda} z_{i:\lambda}^\top - I))$
end while

2.4.3 Zastosowane współczynniki

W eksperymentach wykorzystano analogiczne wartości parametrów do przytoczonych w rozdziale 2.3. Zrobiono to, aby uzyskać lepsze porównanie, poprzez eliminacje wpływu różnych hiperaparemtrów na działanie algorytmów.

2.5 IPOP

2.5.1 Idea algorytmu

IPOP (Increasing Population size with Offspring restarts) to technika restartu stosowana w algorytmach ewolucyjnych, która ma na celu poprawę eksploracji przestrzeni

poszukiwań poprzez stopniowe zwiększanie liczby osobników w populacji. Kluczowym założeniem jest to, że jeśli algorytm zatrzyma się w lokalnym minimum, restart z większą populacją może pomóc wyjść z tego minimum i znaleźć lepsze rozwiązania.

Podejście IPOP działa w cyklach: po spełnieniu określonego warunku restartu optymalizator jest resetowany. Przy każdym restarcie ponownie inicjalizuje się populację oraz (opcjonalnie) parametry algorytmu, takie jak średnia i krok. Jednocześnie rozmiar populacji potomków jest zwiększany, zazwyczaj poprzez mnożenie przez stały współczynnik (np. 2).

Dzięki temu IPOP łączy eksploatację (lokalne poszukiwania w ramach jednej instancji optymalizatora) z eksploracją (poszukiwania globalne przez restart z większą populacją). Technika ta jest szczególnie przydatna w złożonych, wielowymiarowych problemach optymalizacji z wieloma lokalnymi minimami. Warto zauważyć, że przyjęta początkowa liczebność nie ma istotnego znaczenia jeżeli dokonywane jest wiele restartów.

2.5.2 Pseudokod ogólny

Algorithm 3 General IPOP Scheme

Initialize initial parameters: $popsize$, $factor$
$best\_fitness \leftarrow \infty$
while termination criterion not met **do**
    $solutions, z \leftarrow$ `ask()` ▷ optimizer generates new candidate solutions
    $fitnesses \leftarrow f(solutions)$
    `tell(solutions, z, fitnesses)` ▷ updates internal state of optimizer based on evaluated solutions
    if a better solution was found **then**
        $best\_fitness \leftarrow$ updated value
    end if
    if restart condition is satisfied **then**
        $popsize \leftarrow popsize \cdot factor$
        Reinitialize optimizer with new $popsize$
    end if
end while

2.5.3 Zastosowane wariacje

W ramach niniejszego projektu zaimplementowano kilka odmian strategii IPOP, różniących się przyjętym kryterium restartu. Wszystkie warianty opierają się na tej samej idei dynamicznego zwiększania liczby osobników w populacji, ale stosują różne heurystyki do detekcji potrzeby restartu:

- **ThresholdIPOP** — restart wykonywany jest, jeśli średnia poprawa wartości funkcji celu w oknie ostatnich iteracji spada poniżej zadanego progu. Pozwala to reagować na spadek tempa konwergencji.

- **StagnationIPOP** — restart następuje po określonej liczbie iteracji bez jakiejkolwiek poprawy najlepszego znalezionego rozwiązania. To prosty, ale skuteczny sposób detekcji stagnacji.
- **PeriodicRestartIPOP** — restart uruchamiany cyklicznie co zadaną liczbę iteracji. Jest to najprostsza odmiana, pozbawiona mechanizmu detekcji stagnacji, ale gwarantująca regularne odświeżanie procesu optymalizacji.
- **BudgetIPOP** — restart inicjowany po wykorzystaniu zadanego budżetu ewaluacji funkcji celu. Ta wersja umożliwia łatwe kontrolowanie całkowitej liczby obliczeń, co bywa przydatne w środowiskach o ograniczonym czasie działania.

3 Doprecyzowanie dokumentacji wstępnej

3.1 Technologia i architektura

W realizacji projektu wykorzystano język Python wraz z bibliotekami: `numpy`, `argparse`, `csv`, `os`, `time`, `cocoex`, `cocopp` oraz `matplotlib`.

Zrezygnowano z wykorzystania oryginalnego pakietu `cma`, ponieważ zaimplementowano własną wersję algorytmu CMA-ES, co zapewniło większą kontrolę nad działaniem algorytmu. Z kolei z biblioteki `scipy` zrezygnowano, ponieważ wykresy generowane automatycznie przez narzędzie `cocopp` uznano za wystarczające do przeprowadzenia analizy wyników.

W celu lepszej organizacji kodu i ułatwienia eksperymentowania wprowadzono dodatkową warstwę abstrakcji, która pozwala oddzielić implementację algorytmów optymalizacyjnych od strategii restartu `IPOP`. Dzięki temu możliwe było elastyczne definiowanie scenariuszy testowych poprzez łączenie algorytmu bazowego z odpowiednim wrapperem.

3.2 Rodzaje metaheurystyki IPOP

W pierwotnym planie zakładano implementację restartów opartych na dwóch kryteriach: braku poprawy wartości funkcji celu przez określoną liczbę iteracji oraz zbyt małej zmienności w rozkładzie populacji. Po spełnieniu któregokolwiek z kryteriów algorytm miał zwiększyć liczebność populacji i zostać zrestartowany z nowymi punktami startowymi.

Ostatecznie zdecydowano się rozszerzyć planowany zakres o dodatkowe warianty restartu:

- **BudgetIPOP** – restart po przekroczeniu zadanego budżetu ewaluacji,
- **PeriodicRestartIPOP** – restart po upływie z góry ustalonej liczby iteracji.

3.3 Plan eksperymentów

Eksperymenty przeprowadzono z użyciem frameworka `COCO` oraz benchmarku funkcji testowych `BBOB`. Integrację z frameworkiem zrealizowano poprzez stworzenie funkcji `wrappera`, która łączy nasze implementacje optymalizatorów z interfejsem `COCO`.

Eksperymenty zostały zrealizowane w pliku `experiments.py`, który zawiera parametry wywołania algorytmu, w tym:

- ustalony `seed` (losowość eksperymentów jest kontrolowana - w eksperymentach używano `seed` = 10000),
- maksymalna liczba iteracji (a nie budżet ewaluacji),
- lista scenariuszy testowych.

Taki układ gwarantuje powtarzalność wyników pod warunkiem niezmieniania kolejności scenariuszy testowych.

Ponadto zrezygnowano z założenia o przeprowadzeniu testów algorytmów z identycznymi punktami startowymi. Stwierdzono, że losowe wartości m, σ przy wielu scenariuszach testowych oraz ich uśrednianiu może okazać się ciekawsze pod kątem wniosków. Wariant `exclude_ranges=True` oznacza, że po każdym restarcie korzystamy ze starych wartości wektora średniego μ i macierzy wariancji σi.

Eksperymenty przeprowadzono na pełnym zestawie 24 funkcji testowych `BBOB` dla sześciu różnych wymiarów przestrzeni poszukiwań, co stanowi rozszerzenie względem pierwotnych założeń, które obejmowały tylko podzbiór funkcji oraz 5D, 10D i 20D.

Podjęto również decyzję, aby nie ograniczać algorytmów poprzez budżet ewaluacji, lecz przez maksymalną liczbę iteracji optymalizatora. Dzięki temu strategia `IPOP` miała więcej przestrzeni do działania i mogła lepiej rozwinąć swoje możliwości.

Dla każdego badanego algorytmu (CMA-ES oraz MA-ES) przeprowadzono eksperymenty w różnych konfiguracjach heurystyk restartowych. Oto pełna lista użytych wariantów i ich skrótów:

- `cmaes` – podstawowy algorytm `CMA-ES` bez restartów,
- `s_m_30_2_True` – `CMA-ES` z heurystyką `StagnationIPOP`, limit stagnacji 30 iteracji, z losowaniem nowego punktu startowego (wariant `exclude_ranges`),
- `maes` – podstawowy algorytm `MA-ES` bez restartów,
- `s_m_15_2` – `StagnationIPOP` z limitem stagnacji 15 iteracji,
- `s_m_30_2` – `StagnationIPOP` z limitem stagnacji 30 iteracji,
- `s_m_30_2_True` – `StagnationIPOP` z limitem stagnacji 30 iteracji i losowaniem nowego punktu startowego (wariant `exclude_ranges`),
- `p_m_50_2` – `PeriodicRestartIPOP` z interwałem 50 iteracji,
- `p_m_100_2` – `PeriodicRestartIPOP` z interwałem 100 iteracji,
- `b_m_10000_2` – `BudgetIPOP` z limitem budżetu 10 000 ewaluacji na jedną próbę,
- `t_m_10_0.001_2` – `ThresholdIPOP` z progiem 10^{-3} i oknem 10 iteracji,
- `t_m_10_1e-06_2` – `ThresholdIPOP` z progiem 10^{-6} i oknem 10 iteracji,
- `t_m_20_0.001_2` – `ThresholdIPOP` z progiem 10^{-3} i oknem 20 iteracji,
- `t_m_20_1e-06_2` – `ThresholdIPOP` z progiem 10^{-6} i oknem 20 iteracji.

Sufiks `_2` oznacza, że każda konfiguracja została uruchomiona z dwoma niezależnymi losowaniami punktów startowych.

Choć nie było to wymagane w treści zadania, zdecydowano się rozszerzyć analizę o działanie heurystyki `IPOP` dla `CMA-ES`, co umożliwiło bardziej wszechstronną ocenę skuteczności strategii restartu.

Każdy scenariusz testowy uruchamiano 3 razy, a wyniki uśredniono. Ze względu na ograniczone zasoby obliczeniowe uznano, że trzy powtórzenia są wystarczające do oceny trendów działania algorytmów.

Dodatkowo każdy scenariusz uruchomiono w dwóch wariantach: z maksymalną liczbą iteracji ustawioną na 1000 oraz 5000, co pozwala ocenić wpływ długości optymalizacji na skuteczność heurystyk restartu.

3.4 Uruchomienie programu oraz lokalizacja danych wynikowych

Plik `experiments.py` pełni rolę głównego wejścia do systemu eksperymentalnego. Użytkownik może ustawić parametry eksperymentu takie jak liczba iteracji, ziarno losowości czy wybór funkcji benchmarkowych. Umożliwia on pełną integrację z frameworkiem `COCO` poprzez odpowiednie przygotowanie środowiska testowego i funkcji celu.

Każdy eksperyment jest powtarzalny przy założeniu niezmieniania kolejności uruchamianych scenariuszy. Wyniki zapisywane są w strukturze katalogów zgodnej z oczekiwaniami `cocopp`, co pozwala na automatyczne wygenerowanie wykresów porównawczych.

Eksperymenty, oprócz plików pochodzących z cocoex, generują dane wynikowe w formacie `csv`, zawierające:

- nazwę algorytmu i strategii restartu,
- numer funkcji testowej i wymiar przestrzeni,
- uzyskany wynik funkcji celu,
- informacje o czasie trwania algorytmu.

Kod dostępny jest w repozytorium[7], natomiast dane, które cechowały się dużą objętością udostępniono w chmurze[8].

4 Czas działania algorytmów

W niniejszym rozdziale przedstawiono średnie czasy wykonania jednej iteracji optymalizacji dla wybranych wariantów algorytmów. Wyniki uzyskano poprzez uśrednienie danych z trzech niezależnych uruchomień i zestawiono je osobno dla dwóch budżetów ewaluacji: 1000 oraz 5000 i zaprezentowano w tabelach 1 oraz 2.

Zgodnie z oczekiwaniami, algorytm `CMA-ES` cechuje się najdłuższym czasem wykonania spośród wszystkich porównywanych wariantów, zarówno przy budżecie 1000, jak i 5000 ewaluacji. Wynika to z jego struktury obliczeniowej oraz kosztownej adaptacji macierzy kowariancji.

Zastosowanie strategii restartów typu IPOP dla algorytmu `MA-ES` nieznacznie wpływa na czas wykonania — przy czym pozostaje on nadal krótszy niż dla `CMA-ES`, co potwierdza oczekiwaną efektywność adaptacyjnych metod bez kosztownej ewaluacji rozkładu populacji.

Interesującym przypadkiem jest scenariusz `s_c_30_2_True` dla `CMA-ES`, w którym zastosowano restartowanie przy stagnacji z nowymi punktami startowymi. Średni czas działania jest zbliżony do zwykłego `CMA-ES`, co wskazuje, że algorytm często osiągał zadowalające wyniki bez konieczności restartów, lub restart następował na tyle rzadko, że nie wpłynęło to istotnie na ogólny czas.

Wśród wariantów opartych na strategiach restartowych:

Tabela 1 Średni czas działania optymalizatora dla budżetu 1000 ewaluacji

Algorytm	**Średni czas [s]**
b_m_10000_2	402.66
cmaes	522.21
maes	426.23
p_m_100_2	408.86
p_m_50_2	403.86
s_c_30_2_True	536.33
s_m_15_2	418.08
s_m_30_2	405.17
s_m_30_2_True	405.45
t_m_10_0.001_2	441.80
t_m_10_1e-06_2	441.48
t_m_20_0.001_2	436.47
t_m_20_1e-06_2	436.11

Tabela 2 Średni czas działania optymalizatora dla budżetu 5000 ewaluacji

Algorytm	**Średni czas [s]**
b_m_10000_2	2235.70
cmaes	2917.96
maes	2450.13
p_m_100_2	2255.70
p_m_50_2	2396.07
s_c_30_2_True	2939.68
s_m_15_2	2592.31
s_m_30_2	2438.01
s_m_30_2_True	2325.41
t_m_10_0.001_2	2638.71
t_m_10_1e-06_2	2409.36
t_m_20_0.001_2	2410.06
t_m_20_1e-06_2	2465.74

- Najszybsze działanie przy budżecie 1000 oraz 5000 wykazuje konfiguracja `b_m_10000_2` oraz `s_m_30_2`.
- Najwolniejsze działanie przy budżecie 1000 oraz 5000 odnotowano dla `s_c_30_2_True` oraz `t_m_10_0.001_2`, co może wynikać z częstszych restartów lub kosztowniejszych operacji związanych z monitorowaniem stagnacji (Pominięto zwykły CMA-ES w tym stwierdzeniu)

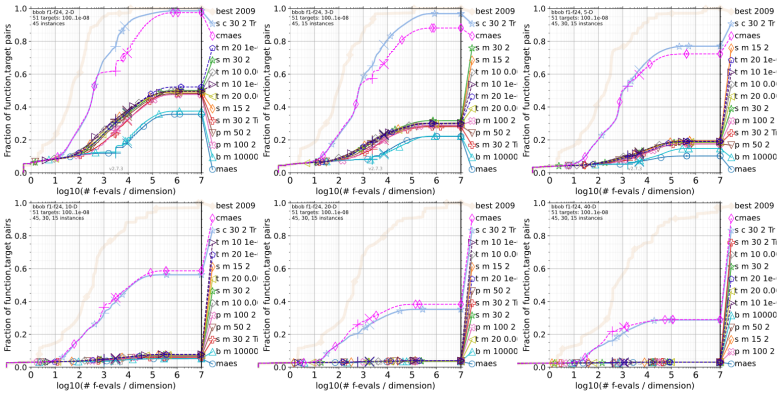
5 Uzyskane wyniki

W niniejszym rozdziale przedstawiono analizę skuteczności i jakości działania badanych algorytmów ewolucyjnych w zadaniach optymalizacj z wykorzystaniem benchmarku COCO BBOB.

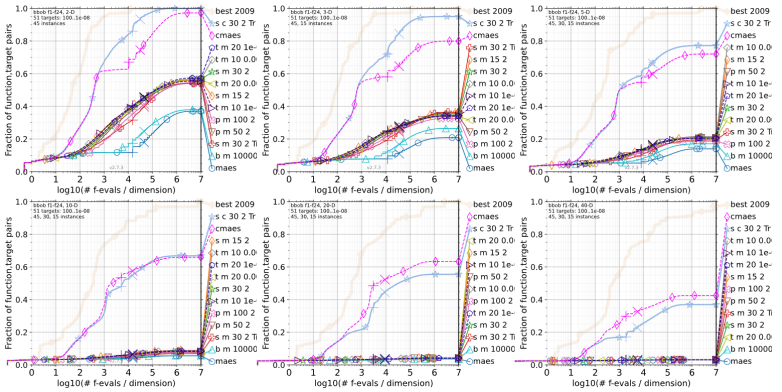
Z uwagi na obszerność danych wyjściowych (dziesiątki funkcji, różne wymiary, wiele wariantów metryk), zdecydowano się na prezentację jedynie najważniejszych fragmentów, które prowadzą do kluczowych wniosków. Pełna dokumentacja wyników – w tym wykresy ECDF oraz szczegółowe tabele ERT – została dołączona jako osobne pliki na dysku i może posłużyć jako materiał do pogłębionej analizy.

5.1 Analiza ogólna skuteczności

Pierwszym krokiem analizy była ocena ogólnej skuteczności algorytmów pod względem liczby iteracji algorytmów potrzebnych na osiągnięcie określonych poziomów wartości funkcji celu. W tym celu wykorzystano zbiorcze wykresy *Runtime profiles over all targets*, które są widoczne na rysunkach 1 oraz 2.



Rysunek 1 Zbiorcze profile czasów osiągnięcia celu dla budżetu 1000 ewaluacji.



Rysunek 2 Zbiorcze profile czasów osiągnięcia celu dla budżetu 5000 ewaluacji.

Jak widać na obu wykresach, różnice między algorytmami są znaczące, zarówno przy dla 1000 iteracji jak i przy 5000 iteracji.

Podstawowa wersja **MA-ES** okazuje się najmniej skuteczna – zwłaszcza w środowisku ograniczonym czasowo oraz dla trudniejszych funkcji.

Wprowadzenie mechanizmu restartów w postaci heurystyki **IPOP** poprawia działanie MA-ES, zwłaszcza w przypadku funkcji o niskim wymiarze. W niższych wymiarach, gdzie istnieje większa szansa na trafienie restartem w korzystny obszar przestrzeni poszukiwań, IPOP pozwala MA-ES osiągać akceptowalne wyniki, często wyraźnie lepsze niż wersja bazowa.

Spośród testowanych wariantów restartów najlepsze rezultaty osiągnięto dla strategii **ThresholdIPOP**, a następnie dla **StagnationIPOP**. Obie te heurystyki były skuteczne w kontekście przełamywania stagnacji optymalizacji i podejmowania prób eksploracji nowych obszarów. Co istotne, obserwacje te są zgodne z oczekiwaniami wynikającymi z konstrukcji tych heurystyk.

Warto jednak podkreślić, że dla funkcji o wyższych wymiarach efektywność IPOP znacząco spada. Restartowanie w takiej przestrzeni wiąże się z większym ryzykiem "rozmycia"budżetu ewaluacji na mało produktywne iteracje, przez co algorytm może nie mieć wystarczająco czasu, aby osiągnąć zadowalający rezultat.

Podczas dalszej analizy będziemy się skupiać tylko na wynikach uzyskanych dla 5000 iteracji.

5.2 Studium przypadków

Aby dokładniej zrozumieć działanie badanych algorytmów, przeprowadzono analizę szczegółową dla wybranych funkcji testowych, które reprezentują różne typy wyzwań optymalizacyjnych.

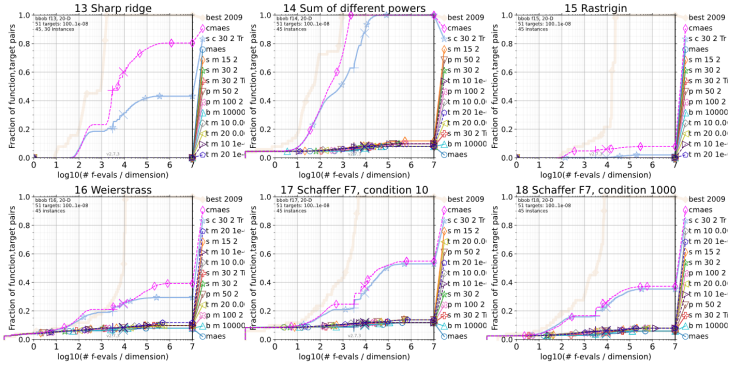
5.2.1 Funkcje, w których dominuje CMA-ES

Na rysunku 3 przedstawiono przypadki, w których algorytm **CMA-ES** oraz jego wariant z mechanizmem restartów (*IPOP*) zdecydowanie przewyższają podejścia oparte na **MA-ES**, zarówno w wersji bazowej, jak i z zastosowaniem heurystyk restartowych. Wyraźnie widać, że uproszczona architektura MA-ES, nawet z dodatkowymi mechanizmami restartów, nie jest w stanie konkurować ze znacznie bardziej dopracowaną strategią adaptacji macierzy kowariancji w CMA-ES.

Warto również zauważyć, że w analizowanych przypadkach to klasyczny CMA-ES osiągał lepsze wyniki niż jego wariant z IPOP, co wskazuje, że restartowanie nie zawsze przynosi korzyści. W pewnych sytuacjach, takich jak szybka i stabilna konwergencja, restarty mogą wręcz zakłócić proces optymalizacji i niepotrzebnie rozproszyć budżet ewaluacji. Jest to zjawisko potwierdzone również w literaturze przedmiotu. Przedstawione funkcje (f13–f18) pełnią tu rolę reprezentatywnych przykładów takich zachowań.

5.2.2 Funkcje, w których CMA-ES i MA-ES IPOP mają problemy

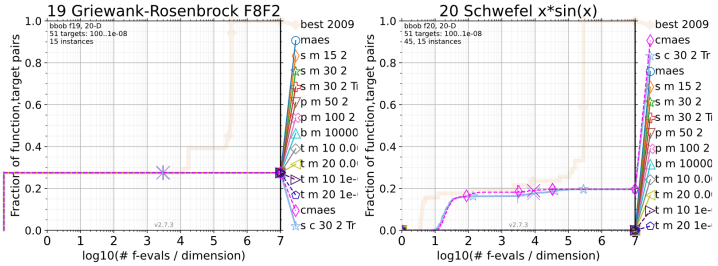
Na rysunku 4 zaprezentowano przypadki, w których zarówno CMA-ES, jak i MA-ES (w wariancie z IPOP) wykazują ograniczoną skuteczność. Dla funkcji **f19** żaden z analizowanych algorytmów nie był w stanie znaleźć rozwiązania znacząco lepszego



Rysunek 3 Profile uruchomienia dla funkcji f13–f18 w wymiarze 20.

niż wartość startowa, co wskazuje na szczególną trudność tej funkcji dla omawianych metod optymalizacyjnych.

Dla funkcji **f20** można zauważyć, że CMA-ES był w stanie osiągnąć przynajmniej częściowy sukces, generując niezerowe wartości funkcji celu, podczas gdy MA-ES całkowicie zawiódł. Mimo to wyniki CMA-ES pozostają dalekie od wyników referencyjnych z benchmarku BBOB 2009, co sugeruje, że również ten algorytm ma swoje ograniczenia w kontekście bardziej złożonych i trudnych do eksploracji przestrzeni poszukiwań.



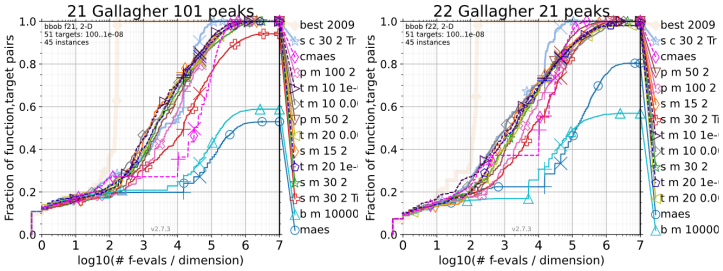
Rysunek 4 Profile uruchomienia dla funkcji f19–f20 w wymiarze 20.

5.2.3 Funkcje, w których MA-ES z IPOP wypada dobrze

Na rysunku 5 przedstawiono przypadki funkcji, dla których algorytm **MA-ES z heurystyką IPOP** osiąga porównywalne wyniki do klasycznego **CMA-ES**. Taka sytuacja występuje jednak wyłącznie w niskowymiarowych przestrzeniach (2D i 3D) i należy do rzadkości. Niemniej jednak pokazuje to, że w sprzyjających warunkach MA-ES, wspomagany przez odpowiednią strategię restartów, jest w stanie skutecznie znaleźć optimum funkcji celu.

Warto podkreślić, że bez zastosowania heurystyki IPOP wyniki MA-ES byłyby znacząco gorsze. Obserwacja ta potwierdza zasadność stosowania mechanizmu restartów w tego typu podejściach. Przedstawiony przypadek dobrze ilustruje, że IPOP nie

tylko poprawia jakość wyników, ale wręcz umożliwia sensowne wykorzystanie MA-ES w praktyce.



Rysunek 5 Profile uruchomienia dla funkcji f21–f22 w wymiarze 2.

5.3 Jakość końcowych rozwiązań

Oprócz czasu osiągnięcia celu, przeanalizowano również jakość końcowych rozwiązań na podstawie wartości ERT (Expected Running Time) oraz skuteczności osiągnięcia zadanych poziomów dokładności względem optimum (Δf_{opt}). Wybrane wyniki przedstawiono w tabeli 3, gdzie analizowana była funkcja f5 w przestrzeni pięciowymiarowej.

W tabeli zestawiono reprezentatywne algorytmy: bazowy MA-ES, jego ulepszone wersje z heurystyką restartu (np. Threshold IPOP oraz Stagnation IPOP), a także CMA-ES jako punkt odniesienia. Widoczna jest wyraźna dominacja CMA-ES, który osiąga wszystkie targety przy minimalnej liczbie wywołań funkcji celu i stuprocentowej skuteczności (45/45 udanych prób). Z drugiej strony, zwykły MA-ES praktycznie nie osiąga żadnego z trudniejszych celów, a jego ERT jest kilkadziesiąt razy wyższe.

Warto jednak zauważyć, że niektóre konfiguracje IPOP w przypadku MA-ES znacząco poprawiają wyniki – np. konfiguracja Threshold osiąga 21 sukcesów na 45 i znacznie niższy ERT względem wersji bazowej. Pokazuje to, że odpowiednie mechanizmy restartu mogą skutecznie niwelować słabości prostszego algorytmu.

W ogólności niestety wyniki ERT nie były zadowalające dla IPOP MA-ES dla większych wymiarowości optymalizowanych funkcji, co ma swoje odzwierciedlenie w wynikach widocznych na przytoczonych wcześniej rysunkach.

6 Podsumowanie

Celem niniejszego projektu było przeanalizowanie działania wybranych algorytmów ewolucyjnych – w szczególności CMA-ES, MA-ES oraz jego wariantu z heurystyką IPOP – w kontekście optymalizacji funkcji testowych z benchmarku COCO BBOB. Eksperymenty zostały przeprowadzone w sposób systematyczny i z zachowaniem wysokiej powtarzalności, co pozwoliło na wyciągnięcie wiarygodnych wniosków na temat mocnych i słabych stron poszczególnych podejść.

Wyniki analizy w dużej mierze potwierdziły oczekiwania wynikające z literatury i wcześniejszych obserwacji. CMA-ES okazał się najbardziej uniwersalnym i skutecznym

Tabela 3 Porównanie wartości ERT (średnia liczba wywołań funkcji celu, w nawiasach odchylenie standardowe dla udanych prób) dla funkcji f5 w wymiarze 5, dla wybranych poziomów Δf_{opt}.

Algorytm	ERT (10^{-2})	ERT (10^{-5})	ERT (10^{-7})	Sukcesy [45]
MA-ES (bazowy)	8.6e4 (8e4)	8.6e4 (1e5)	8.6e4 (8e4)	2
Threshold IPOP (20, 0.001)	6324 (7827)	6324 (4720)	6324 (5833)	21
Stagnation IPOP (30)	1.2e4 (2e4)	1.2e4 (1e4)	1.2e4 (2e4)	12
CMA-ES	19 (12)	19 (12)	19 (12)	45
Stagnation IPOP (30) CMA-ES	20 (10)	20 (10)	20 (10)	45

algorytmem, zdolnym do radzenia sobie z różnorodnymi typami funkcji, również w wysokich wymiarach. MA-ES z IPOP, mimo swojej prostoty, pokazał się jako metoda konkurencyjna w prostszych zadaniach.

Z praktycznego punktu widzenia, projekt ten był cennym doświadczeniem pozwalającym lepiej zrozumieć nie tylko teoretyczne podstawy działania badanych algorytmów, ale również wyzwania związane z ich rzeczywistym zastosowaniem: dobór parametrów, skalowalność z wymiarem, znaczenie budżetu ewaluacji czy wpływ charakterystyki funkcji celu. Szczególnie wartościowe okazało się obserwowanie, jak algorytmy zachowują się w skrajnych sytuacjach – przy ograniczonym czasie, w wysokowymiarowej przestrzeni lub na funkcjach trudnych do eksploracji.

Podsumowując, była to praca, która – mimo że dotyczyła algorytmów – w równym stopniu dotykała aspektów poznawczych, inżynierskich i badawczych. Pokazała, jak wiele można dowiedzieć się o algorytmie nie tylko czytając jego opis, ale obserwując go w działaniu. Wierzymy, że zdobyte doświadczenie stanowi solidną bazę do dalszych eksperymentów w dziedzinie optymalizacji ewolucyjnej i metaheurystyk.

Literatura

[1] Auger, A., Hansen, N.: A Restart CMA Evolution Strategy With Increasing Population Size (2005). https://doi.org/10.1109/CEC.2005.1554902

[2] Beyer, H.-G., Sendhoff, B.: Simplify your covariance matrix adaptation evolution strategy. IEEE Transactions on Evolutionary Computation (2017) https://doi.org/10.1109/TEVC.2017.2680320

[3] Zaborski, M.: Algorytm CMA-ES i jego wybrane rozszerzenia. Referat seminaryjny, Politechnika Warszawska, grudzień 2020. Dostęp online: https://pages.mini.pw.edu.pl/~mandziukj/2020-12-16.pdf

[4] Auger, A., Hansen, N.: COCO: A platform for comparing continuous optimizers in a black-box setting. ArXiv preprint (2016) 1603.08785

[5] Arabas, J.: ALHE - Wykład 10: Algorytmy Heurystyczne. https://elektron.elka.pw.edu.pl/~jarabas/ALHE/wyklad10.pdf

[6] Hansen, N.: The CMA Evolution Strategy: A Tutorial (2023). https://arxiv.org/abs/1604.00772

[7] AndFirst: AMHE - Algorytmy Metaheurystyczne. https://github.com/AndFirst/AMHE

[8] Unknown: Experimental Results – Google Drive Folder. https://drive.google.com/drive/folders/1L3V6HrFXuZEL4lXxPfFL_N5R_termJPu?usp=sharing