POP - Projekt - Sprawozdanie końcowe

Adam Czupryński Szymon Makuch

3 lutego 2025

1 Temat projektu

Ewolucja różnicowa z modyfikacją wdrażającą nieszablonowy model zastępczy (surrogate model) w celu optymalizacji procesu selekcji osobników z populacji do ewaluacji funkcji celu.

2 Opis problemu

Ewolucja różnicowa (DE) jest skutecznym algorytmem optymalizacji globalnej, jednak jej główną wadą jest duża liczba wymaganych ewaluacji funkcji celu. W przypadku gdy obliczenie wartości funkcji celu jest czasochłonne lub kosztowne, może to znacząco ograniczać praktyczne zastosowanie algorytmu. Rozwiązaniem tego problemu może być zastosowanie modelu zastępczego (surrogate model), który aproksymuje wartość funkcji celu na podstawie wcześniej obliczonych punktów.

3 Implementacja

Projekt został zaimplementowany w języku Python. Główny algorytm ewolucji różnicowej znajduje się w pliku differential_evolution.py, natomiast jego zmodyfikowana wersja wykorzystująca model zastępczy w surrogate_de.py. Funkcje testowe z benchmarku CEC oraz pomocnicze funkcje do przeprowadzania eksperymentów znajdują się odpowiednio w plikach functions.py oraz experiments.py. Skrypty parameter_tuning.py oraz tree_parameters_tuning.py służą do wyznaczenia optymalnych parametrów algorytmów ewolucji różnicowej oraz modelu zastępczego.

Wykresy zostały wygenerowane przy użyciu biblioteki pyplot. Informacje dotyczące sposobu odczytywania informacji z wykresu pudełkowego można znaleźć na stronie: `https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.boxplot.html`.

4 Ewolucja różnicowa

Zaimplementowaliśmy algorytm ewolucji różnicowej z dwoma kryteriami stopu: maksymalną liczbą generacji oraz brakiem poprawy najlepszego osobnika przez określoną liczbę generacji. Maksymalna liczba ewaluacji jest więc równa iloczynowi maksymalnej liczby generacji i wielkości populacji. Zawsze jednak istnieje szansa na zatrzymanie algorytmu przez kryterium stopu stagnacji.

Do testowania algorytmu zostały użyte następujące zakresy parametrów:

1. CR - prawdopodobieństwo krzyżowania. Wartość parametru CR jest definicyjnie narzucona i musi znajdować się w przedziale od 0 do 1.
2. F - współczynnik mutacji.
3. Population size - wielkość populacji - dodatnia liczba całkowita.
4. Max generations - maksymalna liczba generacji - dodatnia liczba całkowita.

4.1 Wyznaczanie parametrów ewolucji

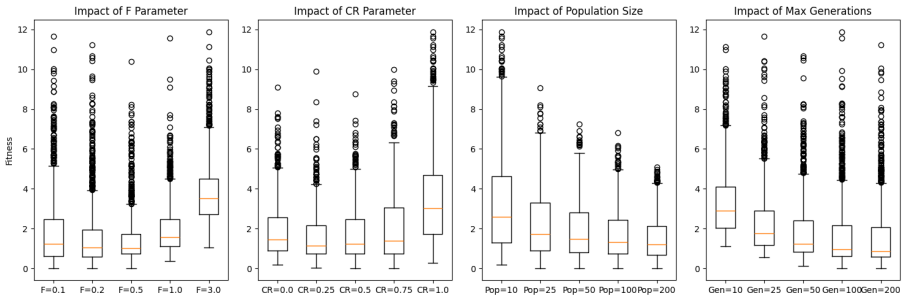
Badanie zaczęliśmy od wyznaczania optymalnych parametrów ewolucji różnicowej. W tym celu przeprowadziliśmy eksperymenty dla funkcji Shifted Rotated Griewank z benchmarku CEC o 10 wymiarach i następujących wartości parametrów:

Miara F	Miara CR	Wielkość populacji	Maksymalna liczba generacji
0.1	0.0	10	10
0.2	0.25	25	25
0.5	0.5	50	50
1.0	0.75	100	100
3.0	1.0	200	200

Tablica 1: Parametry użyte w eksperymentach.

Przeprowadziliśmy eksperymenty na wszystkich kombinacjach wartości parametrów, dla każdego badania wykonaliśmy 30 iteracji.

Dla każdej wartości uśredniliśmy wyniki uzyskane poprzez zmienianie pozostałych parametrów:



Rysunek 1: Wpływ parametrów na jakość algorytmu

Spośród testowanych wartości parametrów najlepsze wyniki jakości uzyskano dla wartości parametrów: F = 0.5, CR = 0.25, wielkość populacji = 200, maksymalna liczba generacji = 200. Parametry F i CR mają swoje optymalne wartości pomiędzy 0 a 1, zaś zwiększanie wielkości populacji i maksymalnej liczba generacji zawsze prowadziło do poprawy jakości rozwiązania. Zwiększało to jednak znacznie czas obliczeń.

5 Model zastępczy

W ramach projektu jako model zastępczy stworzyliśmy drzewo regresyjne. Model wykorzystuje algorytm minimalizacji błędu średniokwadratowego (MSE), pozwalając na kontrolę głębokości drzewa i minimalnej liczby próbek w liściach. Implementacja znajduje się w pliku custom_tree.py.

Proces trenowania modelu DecisionTreeRegressor jest realizowany poprzez rekurencyjne budowanie struktury drzewa, gdzie każdy węzeł reprezentuje punkt decyzyjny oparty na wartości konkretnej cechy. Proces zaczyna się od inicjalizacji drzewa z określonymi parametrami kontrolującymi jego wzrost. Głównym elementem procesu trenowania jest procedura budowy drzewa, która rozpoczyna się od korzenia i rekurencyjnie tworzy kolejne węzły. W każdym węźle algorytm musi podjąć decyzję: czy utworzyć liść, czy kontynuować podział. Decyzja ta jest podejmowana na podstawie warunków stopu: osiągnięcie maksymalnej głębokości lub zbyt mała liczba próbek.

Jeśli podział jest możliwy, algorytm poszukuje najlepszego sposobu podziału danych poprzez iterowanie po wszystkich dostępnych cechach i potencjalnych progach podziału. Dla każdej kombinacji cechy i progu obliczany jest błąd średniokwadratowy powstałych podzbiorów. Wybierany jest podział minimalizujący sumę błędów obu powstałych grup.

Aby zwiększyć wydajność, progi podziału nie są wybierane ze wszystkich możliwych wartości cechy, lecz z wybranych percentyli, co znacząco redukuje przestrzeń poszukiwań. Implementacja wykorzystuje również operacje zwektoryzowane numpy, co przyspiesza obliczenia w porównaniu do tradycyjnych pętli.

Po znalezieniu najlepszego podziału, dane dzielone są na dwie grupy, a proces jest rekurencyjnie powtarzany dla każdej z nich. Tworzy to strukturę drzewiastą, gdzie każdy węzeł wewnętrzny zawiera regułę podziału (cechę i próg), a każdy liść wartość przewidywaną - średnią wartość zmiennej celu dla próbek, które trafiły do tego liścia.

Dzięki takiej strukturze można przewidywać wartości dla nowych próbek. Proces predykcji polega na przejściu od korzenia do odpowiedniego liścia, kierując się w każdym węźle regułami podziału i zwróceniu wartości zapisanej w liściu.

Model inicjowany jest 3 parametrami:

1. Maksymalna głębokość drzewa.
2. Minimalna liczba próbek do podziału.
3. Minimalna liczba próbek w liściu.

Podczas inicjacji do historii dodawane są punkty z początkowej populacji. Model trenowany jest całą historią próbek przy każdej generacji populacji (wymagane jest zadane minimum próbek treningowych). Do historii dodawane są punkty z każdej faktycznej ewaluacji funkcji celu. Model wykorzystywany jest do predykcji wartości dla wszystkich nowych kandydatów. Parametr ewolucji różnicowej top_percentage odpowiada za określenie ile procent najlepszych przewidywanych rozwiązań będzie rzeczywiście ewaluowanych.

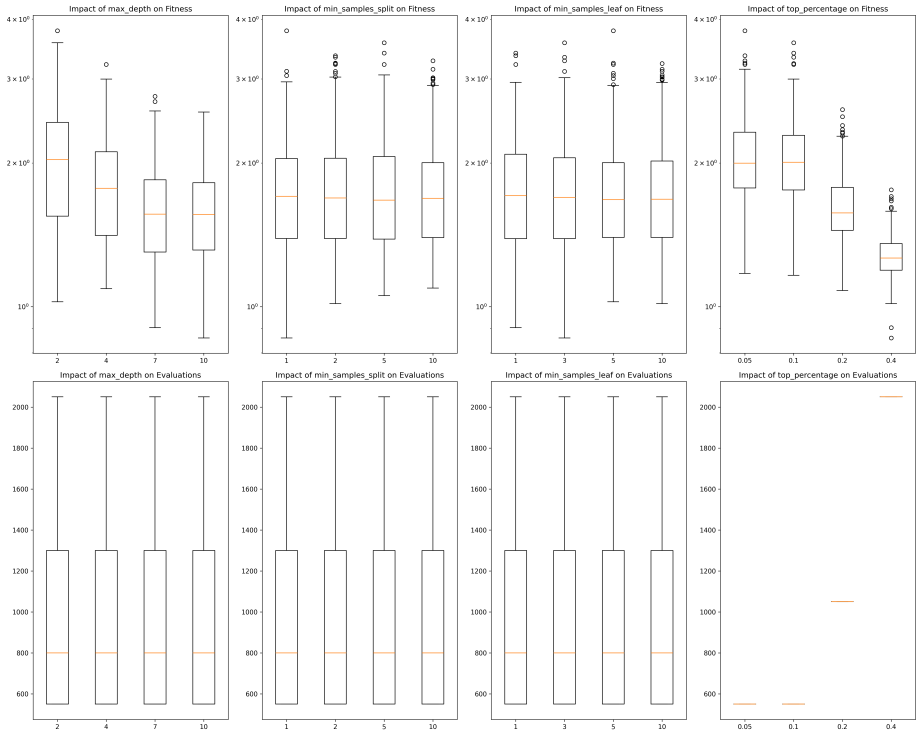
5.1 Wyznaczanie parametrów drzewa regresyjnego

Aby wyznaczyć optymalne parametry drzewa regresyjnego, algorytm uruchomiliśmy dla dziesięciowymiarowej wersji funkcji Shifted Rotated Griewank z benchmarku CEC. Wyznaczyliśmy parametry drzewa regresyjnego dla wszystkich kombinacji wartości parametrów uruchomionych 30 razy:

Głębokość	Próbki przy podziale	Próbki w liściu	Procent ewaluowanych
2	1	1	0.05
4	2	3	0.1
7	5	5	0.2
10	10	10	0.4

Tablica 2: Parametry użyte w eksperymentach.

Dla każdej wartości uśredniliśmy wyniki uzyskane poprzez zmienianie pozostałych parametrów.



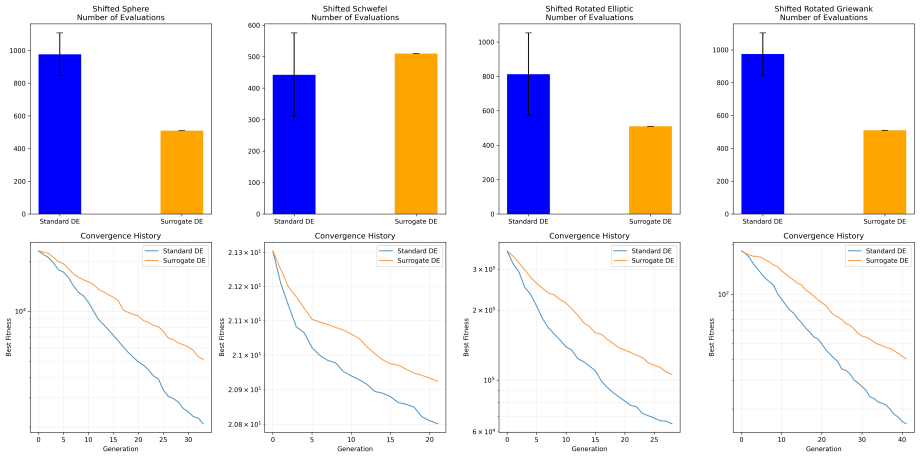
Rysunek 2: Wpływ parametrów drzewa regresyjnego na jakość znajdowanych rozwiązań

Zmiana parametrów drzewa regresyjnego miała niewielki wpływ na jakość modelu zastępczego. Nieznacznie lepsze wyniki w porównaniu do reszty wartości parametrów uzyskano dla drzewa o maksymalnej głębokości 7, minimalnej liczbie próbek do podziału 5 oraz minimalnej liczbie próbek w liściu 5.

Ciekawym parametrem ewolucji różnicowej z modelem zastępczym drzewa regresyjnego jest procent ewaluowanych rozwiązań. Na wykresie dobrze widać kompromis między liczbą wykonywanych ewaluacji a jakością znajdowanych rozwiązań.

6 Porównanie wydajności

Do testowania algorytmów wykorzystaliśmy funkcje testowe z benchmarku CEC: Shifted Sphere, Shifted Schwefel, Shifted Rotated Elliptic oraz Shifted Rotated Griewank.



Rysunek 3: Porównanie standardowego DE z DE wykorzystującym model zastępczy

Wykresy przedstawiają wartości uśrednione po 50 powtórzeniach, dla każdej funkcji osobno. W górnym rzędzie znajduje się liczba ewaluacji wykonywanych przez algorytm ewolucji, a w dolnym najlepsze wartości w zależności od numeru generacji. W większości przypadków algorytm z modelem zastępczym wykonuje mniej ewaluacji funkcji celu. Jedyną funkcją, dla której otrzymaliśmy inne wyniki, była Shifted Schwefel, w przypadku której standardowa ewolucja o wiele szybciej znajdowała optymalne rozwiązanie, przez co kryterium stopu stagnacji kończyło wywoływanie algorytmu, ukracając liczbę ewaluacji. Ewolucja z modelem zastępczym z powodu wolniejszego osiągania optymalnych rozwiązań nie spełniała kryterium stagnacji. Na wykresach w dolnym rzędzie wyraźnie rysuje się różnica w jakości obydwu algorytmów. Algorytm z modelem zastępczym zwracał gorsze wyniki, co jest zgodne z oczekiwaniami, ponieważ model zastępczy jest tylko aproksymacją funkcji celu.

	Standardowa	**Zastępcza**	**Różnica (%)**
Shifted Sphere	976.40	510.00	-47.77%
Shifted Schwefel	442.20	510.00	+15.33%
Shifted Rotated Elliptic	812.40	510.00	-37.22%
Shifted Rotated Griewank	974.80	510.00	-47.68%
Średnia	801.45	510.00	-36.37%

Tablica 3: Średnia liczba ewaluacji funkcji celu

	Standardowa	**Zastępcza**	**Różnica (%)**
Shifted Sphere	172.55	1246.18	+622.17%
Shifted Schwefel	20.66	20.62	-0.18%
Shifted Rotated Elliptic	22176.33	34553.15	+55.81%
Shifted Rotated Griewank	2.59	12.41	+379.10%
Średnia	5593.03	8958.09	+60.17%

Tablica 4: Średnia jakość najlepszego rozwiązania

Przeprowadzone eksperymenty pokazały, że zastosowanie modelu zastępczego w ewolucji różnicowej pozwala na zmniejszenie liczby ewaluacji funkcji celu (dla testowanych funkcji średnio 36.37 %). Jakość rozwiązań uzyskiwanych przez algorytm z modelem zastępczym jest średnio o 60.17% gorsza niż w przypadku standardowej ewolucji różnicowej. W związku z tym, zastosowanie modelu zastępczego w ewolucji różnicowej może być uzasadnione jedynie w przypadku, gdy koszt obliczeń funkcji celu jest bardzo wysoki.