XXX Seminarium

ZASTOSOWANIE KOMPUTERÓW W NAUCE I TECHNICE 2020

Oddział Gdański PTETiS

doi: 10.32016/1.71.04

SKUTECZNOŚĆ NOWOCZESNYCH ALGORYTMÓW OPTYMALIZACJI CZERPIĄCYCH INSPIRACJĘ Z PROCESÓW NATURALNYCH

Zuzanna KLAWIKOWSKA[1], Bartosz PUCHALSKI[2]

Politechnika Gdańska, Wydział Elektrotechniki i Automatyki
1. e-mail: zklawikowska97@gmail.com
2. tel.: 58 347 23 57 e-mail: bartosz.puchalski@pg.edu.pl

Streszczenie: Ze względu na nieistnienie uniwersalnego algorytmu optymalizacji rozwiązującego wszystkie problemy naukowo-techniczne opracowywanie nowych i wydajniejszych obliczeniowo algorytmów optymalizacyjnych wciąż jest popularnym zadaniem. Przeglądając literaturę z dziedziny optymalizacji można zauważyć trend tworzenia „wymyślnych" algorytmów opartych na procesach naturalnych. W artykule sprawdzono skuteczność nowopowstałych algorytmów meta-heurystycznych zainspirowanych życiem owadów i zwierząt – czarnych wdów (algorytm BWO) oraz szarego wilka (algorytm GWO). Skuteczność działania wybranych algorytmów porównano z klasycznym algorytmem quasi-Newtonowskim BFGS oraz strategią ewolucyjną CMA-ES, które charakteryzują się solidnym uwarunkowaniem matematycznym. W celach porównawczych wykorzystano 3 wybrane funkcje testowe. W ramach badań sprawdzono również wpływ liczby zmiennych decyzyjnych na czas uzyskiwania rozwiązania.

Słowa kluczowe: optymalizacja, algorytmy optymalizacji, benchmark, meta-heurystyki.

1. WSTĘP

W dzisiejszych czasach nieodzownym elementem wykorzystywanym w nauce i technice są obliczenia oraz symulacje wspomagane komputerowo skupione w dziedzinie CSE (Computational Science and Engineering). Ze względu na gwałtowny postęp technologiczny, który powoduje współmierny przyrost zasobów i mocy obliczeniowej współczesnych komputerów oraz urządzeń cyfrowych, prężnie rozwijające się badania w ramach dziedziny CSE nad nowymi algorytmami komputerowymi pozwalają na rozwiązanie złożonych problemów technicznych i naukowych, w tym także kompleksowych zadań optymalizacji. Nowoczesne algorytmy stają się na tyle skomplikowane, że przerastają możliwości obliczeniowe człowieka. W przypadku złożonych zadań optymalizacji właściwie jedynym wymaganiem jest znalezienie przez algorytm rozwiązania suboptymalnego, które pod względem jakościowo-czasowym będzie lepsze/szybsze od rozwiązania zaproponowanego przez człowieka.

W istocie osoba pracująca w obrębie ram i metod skupionych wokół dziedziny CSE (w domyśle naukowiec czy inżynier) w swojej codziennej działalności spotyka się z zadaniami optymalizacji praktycznie na każdym kroku. Kontakt ten sprowadza się do 1) wykorzystywania zewnętrznych bibliotek, które zawierają gotowe do użycia algorytmy lub 2) opracowywaniu własnych programów i skryptów rozwiązujących dany problem. W obydwu przypadkach autorzy oprogramowania komputerowego najczęściej starają się implementować algorytmy zgodne z obecnym stanem wiedzy.

Po prześledzeniu światowych czasopism z dziedziny optymalizacji można zauważyć pewien niepokojący trend wśród algorytmów czerpiących inspiracje z naturalnych procesów zachodzących w przyrodzie, a zwłaszcza biologicznych. Trend ten związany jest z powstawaniem nowych „wymyślnych" algorytmów optymalizacji, które zdaniem autorów są skuteczniejsze od innych konkurencyjnych. O ile generalnie taki stan stanowi postęp, to w niektórych przypadkach pomysłowość autorów jest wręcz zadziwiająca i może budzić wątpliwości [1]. W trendzie, o którym mowa można również zauważyć chęć autorów nowych publikacji do stworzenia uniwersalnego algorytmu pozwalającego na skuteczne rozwiązywanie dowolnego problemu optymalizacji.

W artykule wzięto pod uwagę dwa nowe algorytmy optymalizacji, które czerpią inspirację z procesów występujących w środowisku naturalnym. tj. algorytm Czarnej wdowy (BWO) [2] oraz algorytm Szarego wilka (GWO) [3]. Wymienione algorytmy należące do rodziny algorytmów meta-heurystycznych zostały porównane pod względem jakości rozwiązania oraz czasu obliczeń z klasycznym algorytmem quasi-Newtonowskim BFGS [4] oraz popularną strategią ewolucyjną CMA-ES [5].

2. OPIS AGORYTMÓW

2.1. BFGS

Algorytm BFGS należy do rodziny algorytmów quasi-Newtonowskich i jest ich najbardziej popularnym przedstawicielem. Algorytmy z tej rodziny charakteryzują się tym, że wykorzystywana w nich macierz Hessego nie jest obliczana w sposób analityczny na podstawie drugich pochodnych cząstkowych funkcji celu, a jest aproksymowana na podstawie różnic skończonych przybliżeń gradientu funkcji celu. Quasi-Newtonowskie metody optymalizacji stanowią uogólnienie metody numerycznej Siecznych, która w tym wypadku jest wykorzystana do znalezienia pierwiastków pierwszej pochodnej funkcji celu, ale dla problemu wielowymiarowego. Algorytm metody BFGS dany jest w następującej postaci [4]:

- INICJALIZACJA: Określ punkt startowy metody x_0, startową aproksymację odwrotności macierzy Hessego H_0 oraz tolerancję normy gradientu ϵ, ustaw krok metody $k \leftarrow 0$;
- Dopóki $\|\nabla f_k\| > \epsilon$ wykonuj:
    1. Określ kierunek poszukiwania minimum
    $$p_k = -H_k \nabla f_k$$
    2. Ustaw $x_{k+1} = x_k + \alpha_k p_k$, gdzie długość kroku a_k jest wyznaczona spełniając warunki Wolfe'a
    3. Określ $s_k = x_{k+1} - x_k$ oraz $y_k = \nabla f_{k+1} - \nabla f_k$
    4. Wyznacz H_{k+1} na podstawie:
    $H_{k+1} = (\boldsymbol{I} - \rho_k s_k y_k^T) H_k (\boldsymbol{I} - \rho_k y_k s_k^T) + \rho_k s_k s_k^T$, gdzie $\rho_k = \frac{1}{y_k^T s\_k}$
    5. Zwiększ krok $k \leftarrow k + 1$

W literaturze można spotkać dwa główne warianty metody BFGS, tj. L-BFGS oraz BFGS-B. Pierwszy wariant charakteryzuje się ograniczoną pamięcią, w której są przechowywane wektory gradientu, na podstawie których wyznaczana jest aproksymacja macierzy Hessego, natomiast drugi wariant obsługuje proste ograniczenia typu „box" na zmienne decyzyjne. W artykule została wykorzystana wersja połączona algorytmu, tj. L-BFGS-B.

2.2. CMA-ES

Algorytm CMA-ES (Covariance Matrix Adaptation-Evolution Strategy) należy do rodziny algorytmów znanych pod nazwą Strategii Ewolucyjnych (ES). Algorytmy te najczęściej wykorzystywane są do optymalizacji funkcji celu, które charakteryzują się następującymi właściwościami: 1) informacja o gradiencie nie jest dostępna, 2) ilorazy różnicowe funkcji są bezużyteczne, 3) w funkcji celu występują nieciągłości, 4) w funkcji celu występują wartości odstające (outlier) oraz jest ona zaszumiona, 5) funkcja celu jest wielomodalna. W takich warunkach, do znajdowania coraz to lepszych rozwiązań w czasie, algorytmy ES wykorzystują procesy stochastyczne. Algorytm CMA-ES opiera swoje działanie na losowaniu z wielowymiarowego rozkładu normalnego $\mathcal{N}(\boldsymbol{m}, \mathbf{C})$ potencjalnych rozwiązań oraz modyfikowaniu tego rozkładu w kolejnych iteracjach algorytmu w celu znalezienia rozwiązania optymalnego. Modyfikacje rozkładu normalnego w kolejnych iteracjach algorytmu polegają odpowiednio na: 1) aktualizacji wartości oczekiwanych rozkładu $\boldsymbol{m}$, 2) rekursywnych obliczeniach ścieżki ewolucji dla macierzy kowariancji rozkładu normalnego $\boldsymbol{p_C}$ oraz długości kroku algorytmu $\boldsymbol{p_\sigma}$, 3) aktualizacji macierzy kowariancji $\mathbf{C}$ oraz 4) aktualizacji długości kroku σ.

Ze względu na dużą złożoność algorytmu CMA-ES zostanie przedstawiona jego ogólna forma odpowiadająca poszukiwaniu minimum funkcji celu $f: \mathbb{R}^n \rightarrow \mathbb{R}$ wedle idei strategii ewolucyjnych:

- INICJALIZACJA: Zainicjalizuj parametry $\boldsymbol{\theta}$ rozkładu prawdopodobieństwa, z którego będzie losowany wektor potencjalnego rozwiązania, ustaw wielkość populacji λ
- Dopóki nie jest spełniony warunek stopu wykonuj:
    1. Losuj wektory potencjalnego rozwiązania z rozkładu prawdopodobieństwa $P(\boldsymbol{x}|\boldsymbol{\theta}) \rightarrow \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda \in \mathbb{R}^n$
    2. Wyznacz wartości funkcji celu f na podstawie wektorów $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$
    3. Wykonaj aktualizację parametrów rozkładu prawdopodobieństwa na podstawie prawa F_θ
    $$\boldsymbol{\theta} \leftarrow F_\theta\big(\boldsymbol{\theta}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda, f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_\lambda)\big)$$

2.3. BWO

Black Widow Optimization (BWO) to algorytm meta-heurystyczny oparty na ewolucji naturalnej, inspirowany cyklem życia czarnych wdów.

Samice czarnych wdów spędzają większość swojego życia w jednym miejscu, tkając sieć. W trakcie godów, za pomocą feromonów, przyciągają samca do swojej sieci. Samiec zjadany jest w trakcie lub tuż po zapłodnieniu. Następnie samica składa jaja. Świeżo wyklute pająki zaczynają zjadać siebie nawzajem, dzięki czemu przeżywają tylko najsilniejsze osobniki. BWO przypomina klasyczny algorytm genetyczny. Występują w nim fazy inicjalizacji, selekcji, krzyżowania i mutacji. To co odróżnia te algorytmy to faza kanibalizmu.

Inicjalizacja polega na stworzeniu początkowej populacji pająków w postaci:

$$p_{początkowa} = \begin{bmatrix} pająk_1 \\ pająk_2 \\ \vdots \\ pająk_n \end{bmatrix} = \begin{bmatrix} x_{11}, x_{12}, \ \ldots, x_{1N_{var}} \\ x_{21}, x_{22}, \ \ldots, x_{2N_{var}} \\ \vdots \\ x_{n1}, x_{n2}, \ \ldots, x_{nN_{var}} \end{bmatrix} \quad (1)$$

gdzie: x_{ij} – zmienne decyzyjne, których wartości reprezentowane są przez liczby zmienno-przecinkowe, n – liczebność początkowej populacji, N_{var} – liczba zmiennych decyzyjnych.

Tworzona w ten sposób populacja stanowi populację rodzicielską dla pierwszego pokolenia. W fazie selekcji w sposób losowy wybierane jest k par rodziców. Krzyżowanie polega na stworzeniu potomstwa zgodnie ze wzorem 2.

$$\begin{cases} potomek_1 = \alpha \times rodzic_1 + (1 - \alpha) \times rodzic_2 \\ potomek_2 = \alpha \times rodzic_2 + (1 - \alpha) \times rodzic_1 \end{cases} \quad (2)$$

gdzie: α – losowy parametr.

Proces krzyżowania powtarzany jest $N_{var}/2$ razy, w związku z czym otrzymywane jest N_{var} nowych osobników z jednej pary.

Można rozróżnić dwa rodzaje kanibalizmu – partnerski oraz pomiędzy rodzeństwem. Partnerski polega na usunięciu słabszego z rodziców – samca. Kanibalizm pomiędzy rodzeństwem polega na pozostawieniu najlepiej przystosowanych j potomków i usunięciu pozostałych. Przystosowanie poszczególnych osobników określone jest przez odpowiadającą im wartość funkcji celu – im wartość wyższa, tym osobnik lepiej przystosowany. W ten sposób w dalszym etapie brane są pod uwagę tylko najlepsze osobniki. Faza krzyżowania i kanibalizmu powtarzana jest dla wszystkich par.

Mutacja polega na losowym wybraniu m liczby nowopowstałych osobników, a następnie zamianie dwóch losowych elementów danego osobnika.

Nową populację stanowią osobniki, które nie zostały usunięte w fazie kanibalizmu, a proces rozpoczyna się od nowa, aż do spełnienia warunku stopu.

Algorytm BWO dzięki tworzeniu dużej liczby nowych potomków i usuwaniu najsłabszych pozwala zachować

balans pomiędzy przeszukiwaniem rozległego obszaru rozwiązań, a zbieżnością rozwiązania do optimum w znalezionych obszarach.

2.4. GWO

Grey Wolf Optimizer (GWO) to kolejny algorytm meta-heurystyczny inspirowany życiem zwierząt, a dokładniej skłonnościami stadnymi i terytorialnymi gatunku szarego wilka. Wilk szary żyje w ściśle hierarchicznych watahach, gdzie osobnik alfa dominuje nad osobnikiem beta, a ten nad delta. Na samym końcu hierarchii znajdują się osobniki omega. Wilki przejawiają zachowania społeczne. Przykładem takiego zachowania jest wspólne polowanie, które dzieli się na tropienie zwierzyny, ściganie i jej okrążanie, aż ta przestanie się ruszać oraz atak na zdobycz. To właśnie technika polowania oraz hierarchia watahy wilków stanowiła inspirację dla algorytmu GWO.

Modelowanie hierarchii w algorytmie GWO polega na oznaczeniu kolejnych trzech najlepszych rozwiązań jako odpowiednio alfa, beta i delta. Pozostałe rozwiązania określane są jako omega.

Polowanie, czyli proces optymalizacji przewodzony jest przez alfę, betę i deltę. Osobniki omega podążają za tymi trzema wilkami. Proces ten można podzielić na trzy etapy: okrążanie zwierzyny, atakowanie zwierzyny (eksploatacja) oraz tropienie zwierzyny (eksploracja).

Model opisany wzorami 3 i 4 przedstawia matematyczną interpretację okrążania zwierzyny

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_p(k) - \vec{X}(k)\right| \tag{3}$$

$$\vec{X}(k+1) = \vec{X}_p(k) - \vec{A} \cdot \vec{D} \tag{4}$$

gdzie: k – aktualna iteracja, $\vec{A}$ i $\vec{C}$ – wektory współczynników, $\vec{X}_p$ – wektor pozycji ofiary, $\vec{X}$ – wektor pozycji wilka.

Wektory $\vec{A}$ i $\vec{C}$ obliczane są zgodnie z wzorem 5 i 6.

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \tag{5}$$

$$\vec{C} = 2 \cdot \vec{r}_2 \tag{6}$$

gdzie: elementy $\vec{a}$ są liniowo zmniejszane od 2 do 0 podczas przebiegu iteracji, $\vec{r}_1$ i $\vec{r}_2$ – losowe wektory, których elementy zawierają się w zbiorze $\langle 0; 1\rangle$.

W abstrakcyjnej przestrzeni poszukiwań nie wiadomo, gdzie znajduje się ofiara (optimum). W celu zasymulowania polowania przez wilki zakłada się, że alfa (najlepsze potencjalne rozwiązanie), beta i delta posiadają największą wiedzę na temat lokalizacji ofiary. Na podstawie lokalizacji trzech najlepszych osobników reszta wilków (omegi) aktualizują swoje pozycje, zgodnie z równaniami 7-13

$$\vec{D}_\alpha = \left|\vec{C}_1 \cdot \vec{X}_\alpha(k) - \vec{X}(k)\right| \tag{7}$$

$$\vec{D}_\beta = \left|\vec{C}_2 \cdot \vec{X}_\beta(k) - \vec{X}(k)\right| \tag{8}$$

$$\vec{D}_\delta = \left|\vec{C}_3 \cdot \vec{X}_\delta(k) - \vec{X}(k)\right| \tag{9}$$

$$\vec{X}_1(k) = \vec{X}_\alpha(k) - \vec{A}_1 \cdot \vec{D}_\alpha \tag{10}$$

$$\vec{X}_2(k) = \vec{X}_\beta(k) - \vec{A}_2 \cdot \vec{D}_\beta \tag{11}$$

$$\vec{X}_3(k) = \vec{X}_\delta(k) - \vec{A}_3 \cdot \vec{D}_\delta \tag{12}$$

$$\vec{X}(k+1) = \frac{\vec{X}_1(k)+\vec{X}_2(k)+\vec{X}_3(k)}{3} \tag{13}$$

Można zauważyć, że alfa, beta i delta szacują pozycję ofiary, a inne wilki losowo aktualizują swoje pozycje wokół niej.

W naturze wilki szare atakują ofiarę, dopóki ta nie przestanie się ruszać. W celu matematycznego zamodelowania zbliżania się do ofiary zmniejszana jest wartość $\vec{a}$. Wektor $\vec{A}$ zawiera losowe wartości z zakresu $\langle -a; a\rangle$. Jeżeli $|A| < 1$ to wilki atakują w kierunku zwierzyny, w przeciwnym wypadku oddalają się od niej i mogą znaleźć rozwiązanie lepsze od aktualnego.

Wektor $\vec{C}$ zawierający losowe wartości z zakresu $\langle 0; 2\rangle$ wprowadza wagi dla ofiary w celu losowego zwiększenia $(C > 1)$ lub zmniejszenia $(C < 1)$ wpływu zdobyczy na zdefiniowanie odległości. Dzięki wprowadzeniu losowego współczynnika wzmocniona zostaje eksploracja oraz wychodzenie z lokalnych optimów.

Podsumowując, proces poszukiwań rozpoczyna się od inicjalizacji losowej populacji wilków szarych (potencjalnych rozwiązań). W trakcie działania algorytmu wilki alfa, beta i delta szacują pozycję ofiary (optimum). Pozostałe wilki aktualizują swoją odległość od ofiary. Parametr a jest zmniejszany od 2 do 0 wpływając na wektor $\vec{A}$. Gdy $|A| > 1$ wilki rozbiegają się. Zbiegają się w kierunku ofiary, gdy $|A| < 1$. Algorytm GWO kończy się, gdy zostanie spełnione kryterium stopu.

3. PORÓWNANIE SKUTECZNOŚCI ALGORYTMÓW

3.1. Metoda porównawcza

Zbadano przedstawione algorytmy optymalizacyjne z użyciem trzech funkcji testowych dla różnej liczby zmiennych decyzyjnych. Pierwszą funkcją testową jest funkcja Rastrigina (Rast), opisana wzorem 14. Posiada ona minimum globalne $f(\boldsymbol{x}) = 0$ dla $\boldsymbol{x} = 0$. Typowo do poszukiwania minimum nakłada się ograniczenia na zmienne decyzyjne wynoszące $x_i \in \langle -5{,}12; 5{,}12\rangle$.

$$f(x) = An + \sum_{i=1}^{n}[x_i^2 - A\cos(2\pi x_i)] \tag{14}$$

gdzie: $A = 10$, n – liczba zmiennych decyzyjnych, x_i zmienne decyzyjne.

Kolejną wykorzystaną funkcją jest funkcja sferyczna (Sphere) opisana wzorem 15, posiadająca minimum globalne $f(\boldsymbol{x}) = 0$ dla $\boldsymbol{x} = 0$.

$$f(x) = \sum_{i=1}^{n} x_i^2 \tag{15}$$

gdzie: n – liczba zmiennych decyzyjnych, x_i – zmienne decyzyjne.

Jako trzecią funkcję wykorzystano funkcję Rosenbrocka (Rosen) opisaną za pomocą wzoru 16.

$$f(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2] \tag{16}$$

gdzie: n – liczba zmiennych decyzyjnych, x_i – zmienne decyzyjne

Tak zdefiniowana funkcja posiada minimum globalne $f(\boldsymbol{x}) = 0$ dla $\boldsymbol{x} = \boldsymbol{J}_{1,n}$, gdzie $\boldsymbol{J}_{1,n}$ oznacza wektor wierszowy wypełniony jedynkami o rozmiarach $1 \times n$.

Tablica 1. Zestawienie uzyskanych optimów

3.2. Przedstawienie wyników

W tablicy 1 zestawiono uzyskane minima badanych funkcji przez zbadane algorytmy optymalizacji.

	BFGS	CMA-ES	BWO	GWO	BFGS	CMA-ES	BWO	GWO	BFGS	CMA-ES	BWO	GWO
n	2				10				20			
Sphere	0	0	0	0	1.21E-11	2.59E-15	3.14E-09	0	0	1.61E-14	3.88E-06	0
Rast	4.26E-14	15.10	0	0	5.68E-14	4.97	2.69E-03	0	2.84E-14	378.00	2.84E-14	3.55
Rosen	1.27E-11	3.85E-17	7.50E-03	9.76E-07	2.01E-10	1.11E-14	8.43	6.2354	2.15E-09	3.98	1265.38	16.18

Można zauważyć, że algorytmy prawidłowo przybliżają znane minima poza 9 przypadkami (CMA-ES – 2 @ Rast, CMA-ES – 10 @ Rast, BWO – 10 @ Rosen, GWO – 10 @ Rosen, CMA-ES – 20 @ Rast, CMA-ES – 20 @ Rosen, BWO – 20 @ Rosen, GWO – 20 @ Rast, GWO – 20 @ Rosen).

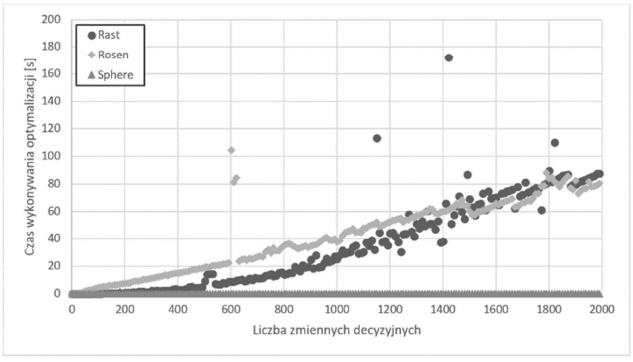
W kontekście wyników zamieszczonych w Tablicy 1 można zauważyć, że algorytm BFGS odnalazł poprawnie minimum dla wszystkich badanych funkcji. Algorytm CMA-ES okazał się nieskuteczny dla wielomodalnej funkcji Rastrigina. Algorytmy BWO oraz GWO okazały się nieskuteczne dla funkcji Rosenbrocka dla n=10 oraz n=20.

W tablicy 2 przedstawiono znaczące parametry dla poszczególnych algorytmów optymalizacji. Parametry nieujęte w tablicy 2 przyjmują wartości domyślne dostępne w dokumentacji dla: L-BFGS-B [6], CMA-ES [7], BWO [8], GWO [9].

Tablica 2. Parametry algorytmów optymalizacyjnych

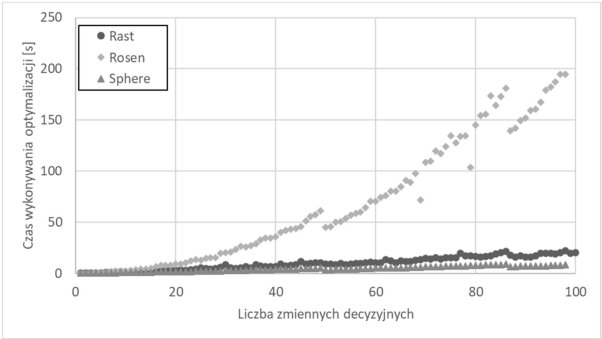
CMA-ES		BWO		GWO	
Sigma	1	Liczebność populacji	60	Liczebność populacji	30
		Maksymalna liczba iteracji	500	Maksymalna liczba iteracji	500

Sprawdzono w jaki sposób zmienia się czas obliczeń algorytmów wraz ze wzrostem liczby zmiennych decyzyjnych. Na rysunku 1 przedstawiono tę zależność dla algorytmu BFGS. Czas wykonywania tego algorytmu wydłuża się najbardziej dla funkcji Rosenbrocka, a najmniej dla najprostszej z wykorzystywanych funkcji, czyli sferycznej. Widać również, że w miarę zwiększania liczby zmiennych decyzyjnych czas wykonywania obliczeń wzrasta w przybliżeniu liniowo. Algorytm L-BFGS charakteryzuje się złożonością czasową typu $\mathcal{O}(n \cdot m)$ [10].



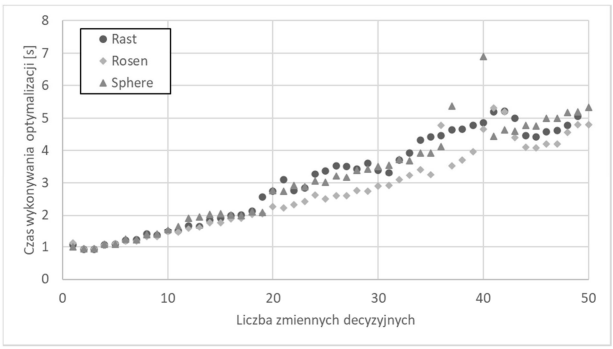
Rys. 1. Zależność czasu obliczeń od liczby zmiennych decyzyjnych dla algorytmu BFGS

Rysunek 2 przedstawia omówione zależności dla algorytmu CMA-ES. Ponownie tempo wzrostu czasu obliczeń jest największe dla funkcji Rosenbrocka, a najmniejsze dla funkcji sferycznej. W tym wypadku widać kwadratowy wzrost czasu obliczeń wraz ze zwiększaniem się liczby zmiennych decyzyjnych. Algorytm CMS-ES charakteryzuje się złożonością czasową typu $\mathcal{O}(n^2)$ [11].

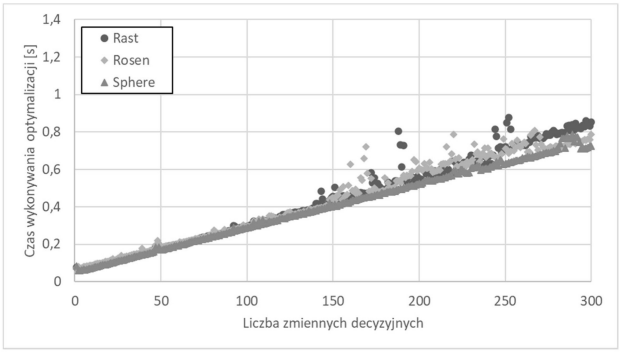


Rys. 2. Zależność czasu obliczeń od liczby zmiennych decyzyjnych dla algorytmu CMA-ES

Wykresy uzyskane z wykorzystaniem algorytmu BWO i GWO przedstawiono odpowiednio na rysunku 3 i 4. Warto zauważyć, że na obydwóch wykresach czasy obliczeń dla różnych funkcji są względnie porównywalne z algorytmem BFGS. Z rysunków 3 i 4 również można wywnioskować, że dla danego zakresu zmian liczby zmiennych decyzyjnych wzrost czasu obliczeń jest w przybliżeniu liniowy, co świadczy o zbliżonej złożoności czasowej tych algorytmów.



Rys. 3. Zależność czasu obliczeń od liczby zmiennych decyzyjnych dla algorytmu BWO



Rys. 4. Zależność czasu obliczeń od liczby zmiennych decyzyjnych dla algorytmu GWO

Algorytm CMA-ES w porównaniu z resztą algorytmów najczęściej nie znajdował oczekiwanego optimum zwłaszcza dla funkcji Rastrigina. Co więcej charakteryzował się najdłuższymi czasami wykonywania algorytmu. Algorytm BFGS dla każdego przypadku znalazł oczekiwane optimum, jednak czas wykonywania obliczeń dla funkcji Rosenbrocka znacząco wzrasta wraz z zwiększającą się liczbą zmiennych decyzyjnych. Algorytmy BWO i GWO nie zawsze znajdowały oczekiwane optimum. Algorytmy te okazały się nieskuteczne dla funkcji Rosenbrocka dla przypadków, w których liczba zmiennych decyzyjnych przekraczała 10.

4. WNIOSKI KOŃCOWE

W niniejszym artykule przedstawiono cztery algorytmy optymalizacyjne i porównano je pod względem skuteczności jakościowej wyznaczania optimum dla trzech wybranych funkcji testowych. Sprawdzono również w jaki sposób zmieniał się czas wykonywania obliczeń w zależności od liczby zmiennych decyzyjnych. Pokazano, że algorytmy oparte na procesach naturalnych, tj. BWO i GWO w większości przypadków sprawdziły się lepiej od strategii ewolucyjnej CMA-ES i gorzej od klasycznego algorytmu quasi-Newtonowskiego BFGS. Pomimo w przybliżeniu kwadratowej złożoności czasowej algorytm BFGS odszukał minimum dla każdego badanego przypadku (tablica 1). Badane algorytmy meta-heurystyczne pomimo, korzystniejszej złożoności czasowej, nie były w stanie znaleźć optimum badanych funkcji dla każdego przypadku wyszczególnionego w tablicy 1.

Celem artykułu było zwrócenie uwagi na wschodzący problem nagminnego tworzenia algorytmów optymalizacji, które czerpią inspiracje z różnych zjawisk występujących w przyrodzie, przez co pozornie mogą wydawać się innowacyjne i skuteczne, lecz w gruncie rzeczy nadają się do ograniczonej klasy problemów optymalizacji oraz w istocie są kolejnym klonem dobrze znanych algorytmów genetycznych ze zmienioną nomenklaturą oraz operatorami. Można stwierdzić, że dziedzina nauki związana z algorytmami optymalizacji jest w pewnym sensie niepotrzebnie zalewana algorytmami, które kopiują dobrze znane metody zmyślnie ubrane w nowe ramy przez co faktycznie tłumią prawdziwy naukowy rozwój metod opartych o meta-heurystyki.

Należy pamiętać, że algorytm optymalizacyjny należy dobierać odpowiednio do rozwiązywanego problemu. Proces doboru należy poprzedzić analizą właściwości danego zadania w celu ich wykorzystania w kontekście doboru czy projektu algorytmu optymalizacji. Z jednej strony ślepe podążanie za obecnie obserwowanym trendem związanym z wykorzystaniem algorytmów opartych na procesach naturalnych może prowadzić do nieadekwatnego ich użycia w stosunku do postawionego problemu. Klasyczne algorytmy optymalizacji, np. te bazujące na gradiencie czy macierzy Hessego w pewnych warunkach generują bardziej zadowalające rezultaty. Z drugiej strony nie należy podchodzić zbyt sceptycznie do powstawania nowych algorytmów meta-heurystycznych, gdyż w znacznym stopniu przyczyniają się one do rozwoju metod optymalizacji, a uzyskiwane przez nie wyniki potrafią być ostatnią deską ratunku tam, gdzie klasyczne metody są nieskuteczne lub nie można ich zastosować.

5. BIBLIOGRAFIA

1. Sörensen K.: Metaheuristics—the metaphor exposed. International Transactions, Operational Research, Nr 22 (1), 2015, s. 3-18.
2. Hayyolalam V., Kazem, A. A. P.: Black widow optimization algorithm: A novel meta-heuristic approach for solving engineering optimization problems, Engineering Applications of Artificial Intelligence, Nr 87, 2020, s.103249.
3. Mirjalili S., Mirjalili S. M., Lewis A.: Grey wolf optimizer, Advances in engineering software, Nr 69, 2014, s. 46-61.
4. Nocedal, J., Wright, S. J.: Numerical Optimization (2nd ed.), Springer, Berlin, New York 2006.
5. Hansen N.: The CMA evolution strategy: a comparing review, Towards a new evolutionary computation, Springer, Berlin, Heidelberg, 2006, s. 75-102.
6. https://docs.scipy.org/doc/scipy/reference/optimize.minimize-lbfgsb.html, [Dostęp: 16.09.2020].
7. http://cma.gforge.inria.fr/apidocs-pycma/cma.evolution_strategy.html#fmin, [Dostęp: 16.09.2020].
8. https://github.com/hayyolalam/Hlm, [Dostęp: 16.09.2020].
9. https://www.mathworks.com/matlabcentral/fileexchange/44974-grey-wolf-optimizer-gwo, [Dostęp: 16.09.2020].
10. Saputro, D. R. S., Widyaningsih, P.: imited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method for the parameter estimation on geographically weighted ordinal logistic regression model (GWOLR), AIP Conference Proceedings, Nr 1868 (1), 2017, s. 40009.
11. Hansen, N., Müller, S. D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), Evolutionary computation, Nr 11 (1), 2003, s.1-18.

THE EFFECTIVENESS OF MODERN OPTIMIZATION ALGORITHMS INSPIRED BY NATURAL PROCESSES

Due to the lack of a universal optimization algorithm which solves all scientific and technical problems, developing new and more computationally efficient optimization algorithms is still a popular challenge. Reviewing the literature on optimization there is a trend to create "fancy" algorithms based on natural processes. The article examines the effectiveness of

newly developed meta-heuristic algorithms inspired by insects and animals - black widows (BWO algorithm) and grey wolf (GWO algorithm). The effectiveness of the selected algorithms was compared with the classical quasi-Newtonian BFGS algorithm and the evolutionary strategy CMA-ES, which are characterized by a solid mathematical background. Three selected benchmark functions were used for comparison purposes. The study also included a test of the influence of the number of design variables on the time complexity.

Keywords: optimization, optimization algorithms, benchmark, meta-heuristics.