Żuki

Joanna Gawrońska

Oktawia Szczypiór

Spis treści:

Wprowadzenie:

- Przedmiotem analizy jest algorytm optymalizacji roju żuka (BSO). Analizy tej dokonano na podstawie artykułu Beetle Swarm Optimization Algorithm:Theory and Application, opublikowanew 2018 roku autorstwa Tiantian Wang, Long Yang oraz Qiang Liu. W prezentacji algorytm BSO został porównany z algorytmami PSO i BAS. Wyniki pokazały, że w porównaniu z obecnie popularnymi algorytmami optymalizacji BSO może pozwalać uzyskiwać lepsze wyniki niż dotychczasowe algorytmy.

ALGORYTMY METAHEURYSTYCZNE

Heurystyka - to metoda znajdowania rozwiązań („niepełny algorytm”) która pozwala na znalezienie w akceptowalnym czasie przynajmniej „dostatecznie dobrego” przybliżonego rozwiązania problemu. Metod tych używa się np. wtedy, gdy pełny algorytm jest nieznany lub zbyt kosztowny obliczeniowo.

Z kolei **metaheurystyka** to ogólny algorytm do rozwiązywania problemów obliczeniowych, zazwyczaj optymalizacyjnych. Określenie oznacza „heurystykę wyższego poziomu” co wynika z faktu, że algorytmy tego typu nie rozwiązują bezpośrednio żadnego problemu, a jedynie podają sposób na utworzenie odpowiedniego algorytmu. Metody te często są inspirowane mechanizmami biologicznymi czy fizycznymi. Zaliczyć do nich można np.: algorytmy ewolucyjne, sztuczne sieci neuronowe, systemy rozmyte, algorytmy immunologiczne czy systemy mrówkowe

Owady-cechy

Owady mają zmieniający się chemiczny system sensoryczny, który wykrywa różne bodźce środowiskowe. Czółki owadów są ważnymi receptorami chemicznymi. Odgrywają one głównie działanie węchowe i dotykowe, a niektóre mają nawet funkcję słuchową. Pomagają owadom komunikować się, znajdować płeć przeciwną, znajdować pożywienie i wybierać miejsca tarła. Ludzie często wykorzystują tę właściwość owadów do uwalniania substancji o specyficznych lotnych zapachach, aby przyciągać lub unikać owadów szkodliwych dla roślin.

Rogaty chrząszcz charakteryzuje się wyjątkowo długimi czółkami, niekiedy nawet czterokrotnie dłuższymi niż jego ciało.
Ten rodzaj długich anten ma dwie podstawowe funkcje:
-eksploracja otaczającego środowiska.
Na przykład, napotykając przeszkodę, czujnik może dostrzec jego rozmiar, kształt i twardość.

-chwytanie zapachu jedzenia lub znalezieniu potencjalnych partnerów poprzez wychylenie anteny ciała. Po wykryciu wyższego stężenia zapachu po jednej stronie czółka chrząszcz obróci się w tym samym kierunku, w przeciwnym razie zwróci się na drugą stronę. Zgodnie z tą prostą zasadą chrząszcze mogą skutecznie znajdować pożywienie.

Particle Swarm Optimization

- Ideą algorytmu PSO jest iteracyjne przeszukiwanie przestrzeni rozwiązań problemu przy pomocy roju cząstek.
- Każda z cząstek posiada swoją pozycję w przestrzeni rozwiązań, prędkość oraz kierunek w jakim się porusza.
- Zapamiętywane jest najlepsze rozwiązanie znalezione do tej pory przez każdą z cząstek (rozwiązanie lokalne), a także najlepsze rozwiązanie z całego roju (rozwiązanie globalne).
- Prędkość ruchu poszczególnych cząstek zależy od położenia najlepszego globalnego i lokalnego rozwiązania oraz od prędkości w poprzednich krokach. Na czacie zostanie przedstawiony wzór pozwalający na obliczenie prędkości danej cząstki.

WZÓR NA OBLICZENIE PRĘDKOŚCI DANEJ CZĄSTKI

$$v \leftarrow \omega v + \varphi_I r_I (I - x) + \varphi_g r_g (g - x),$$

gdzie:

- v - predkość czastki
- ω - współczynnik bezwładności, określa wpływ predkości w poprzednim kroku
- φ_I - współczynnik dażenia do najlepszego lokalnego rozwiazania
- φ_g - współczynnik dażenia do najlepszego globalnego rozwiazania
- I - położenie najlepszego lokalnego rozwiazania
- g - położenie najlepszego globalnego rozwiazania
- x - położenie czastki
- r_l, r_g - losowe wartości z przedziału $(0, 1)$

ALGORYTM PSO

- Dla każdej czastki ze zbioru:
  - Wylosuj pozycje poczatkowa z przestrzeni rozwiazań
  - Zapisz aktualna pozycje czastki jako najlepsze lokalne rozwiazanie
  - Jeśli rozwiazanie to jest lepsze od najlepszego rozwiazanie globalnego, to zapisz je jako najlepsze
  - Wylosuj predkość poczatkowa
- Dopóki nie zostanie spełniony warunek stopu (np. minie określona liczba iteracji):
  - Dla każdej czastki ze zbioru:
    - Wybierz losowe wartości parametrów rl i rg
    - Zaktualizuj predkość czastki wg powyższego wzoru
    - Zaktualizuj położenie czastki w przestrzeni
    - Jeśli aktualne rozwiazanie jest lepsze od najlepszego rozwiazania lokalnego:
      - Zapisz aktualne rozwiazanie jako najlepsze lokalnie
    - Jeśli aktualne rozwiazanie jest lepsze od najlepszego rozwiazania globalnego:
      - Zapisz aktualne rozwiazanie jako najlepsze globalnie

Beetle Antennae Search Algorithm

- Algorytm wyszukiwania anten chrząszcza (BAS), zainspirowany zachowaniem poszukiwawczym chrząszczy długorogich. Algorytm BAS imituje funkcję anten i mechanizm chodzenia losowego chrząszczy w naturze, a następnie realizuje są dwa główne etapy wykrywania i wyszukiwania. Na koniec algorytm jest porównywany z 2 dobrze znanymi funkcjami testowymi, w których wyniki liczbowe potwierdzają skuteczność proponowanego algorytmu BAS.

Zalety algorytmu BAS

- Algorytm wyszukiwania anteny żukowej (BAS) może automatycznie realizować proces optymalizacji bez znajomości konkretnej formy funkcji i informacji o gradiencie.
- Mniejsza złożoność związana z jego projektowaniem i zdolnością do rozwiązania problemu optymalizacji w krótszym czasie, ponieważ jego indywidualna liczba jest tylko jedna.

Model matematyczny-BAS

1. Niech $f(x^t)$ oznacza funkcję, którą optymalizujemy, gdzie zmienna $x^t = [x^1, , , x^n]$, t oznacza liczbę iteracji.
Randomizujemy orientację chrząszcza:

$$\overrightarrow{b} = \frac{rand(k, 1)}{||rand(k, 1)||}$$

gdzie $rand(,)$ oznacza losową orientację chrząszcza, jest to matryca wymiarów $k \times 1$, a parametr k jest zgodny z wymiarami zmiennej niezależnej. $\overrightarrow{b}$ jest wektorem jednostkowym.

2. Współrzędne dwóch anten uzyskane w każdej iteracji:

$$\boldsymbol{x}_r = \boldsymbol{x}^{t-1} + d^{t-1}\overrightarrow{\boldsymbol{b}}$$
$$\boldsymbol{x}_l = \boldsymbol{x}^{t-1} - d^{t-1}\overrightarrow{\boldsymbol{b}}$$

gdzie x_r i x_l odpowiednio współrzędne prawej i lewej anteny
d- to długość wykrywania anten, którą można ustalić w kroku δ.

$$\delta^t = \lambda_1 \delta^{t-1}$$

gdzie λ_1 jest stałą, określa prędkość zaniku kroku wyszukiwania.

$$d_0^t = \delta^t / c$$

gdzie c-stała

3. Rozwiązując powyższe współrzędne, otrzymamy iteracyjną formułę zmiennej niezależnej

$$\boldsymbol{x}^t = \boldsymbol{x}^{t-1} - \delta\overrightarrow{\boldsymbol{b}}\, sign(f(\boldsymbol{x}_l) - f(\boldsymbol{x}_r))$$

gdzie aktualizacja współrzędnych x_t jest związana ze współrzędnymi poprzedniego kroku x_{t-1}.

Przykład

Czterostopniowy proces optymalizacji chrząszcza.

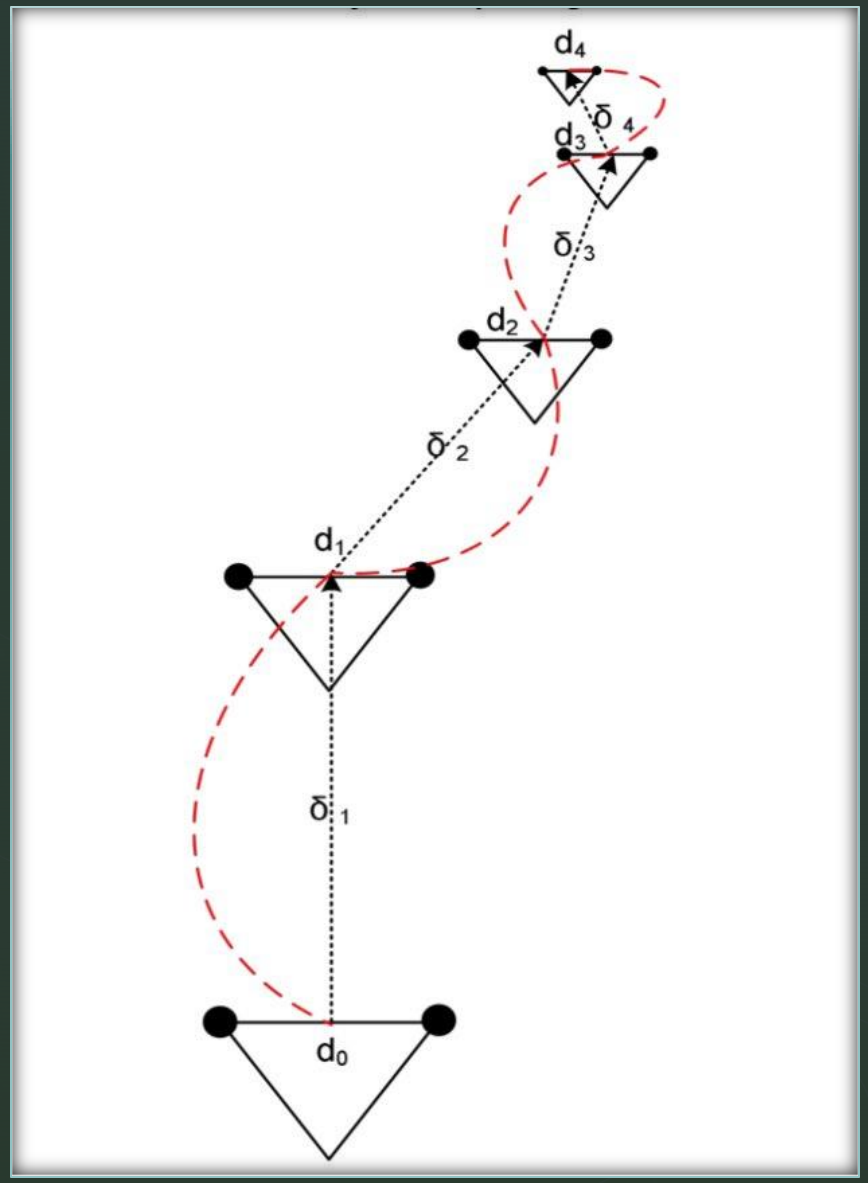
-czarny trójkąt reprezentuje chrząszcza

-czarne pełne koła po obu stronach reprezentują brodę chrząszcza

-litera d z liczbami {1,2,3,4} reprezentuje odległość między dwiema antenami

-delta z liczbami {1,2,3,4} reprezentuje długość kroku

-czerwona linia przerywana przedstawia trajektorię funkcji fitness



Algorytm BAS i jego wady

Przy ciągłym pogłębianiu eksperymentu z zastosowaniem BAS wskazano jego wady:

- wydajność algorytmu w radzeniu sobie z funkcjami wysokowymiarowymi nie jest zadowalająca
- wynik interakcji jest bardzo zależny od początkowej pozycji chrząszcza.

Dlatego, wybór pozycji początkowej ma duży wpływ na wydajność i skuteczność optymalizacji. Tiantian Wang, Long Yang oraz Qiang Liu zainspirowani algorytmem optymalizacji roju wprowadzili dalsze usprawnienia w algorytmie BAS, rozszerzając jednostkę na grupę.

- Jak opisano w poprzednich dwóch sekcjach, algorytm BAS jest skierowany tylko do osobników i nie uwzględnia połączeń między grupami.
- PSO koncentruje się na wpływie populacji na pojedynczą cząsteczkę, ignorując jej własną ocenę w procesie wyszukiwania.

Dlatego w niniejszym dokumencie zaproponowano zintegrowanie modeli BAS i PSO, wprowadzając koncepcję BSO. Każda cząstka w PSO jest scharakteryzowana jako żuk. BSO skonstruowany tą metodą może dobrze przezwyciężyć problemy słabej stabilności, tendencji do wpadania do lokalnego optimum i innych powodowanych przez algorytm PSO. Proces początkowej pozycji i prędkości roju żuka jest taki sam jak w przypadku standardowego PSO. Jednak w procesie iteracyjnym sposób aktualizowania pozycji roju chrząszcza nie polega już tylko na najlepszym w historii rozwiązaniu i aktualnym globalnym optymalnym rozwiązaniu osobnika chrząszcza, ale dodaje pomysł wyszukiwania anteny chrząszcza. Osoby w BSO będą porównywać wartości funkcji sprawności ich lewej i prawej strony podczas każdej iteracji i będą porównywać lepsze wartości tych dwóch, które można również wykorzystać do aktualizacji pozycji roju chrząszcza. Zaktualizowaną formułę położenia roju chrząszcza można wyrazić w następujący sposób

Beetle Swarm Optimization-BSO

- Algorytm optymalizacji roju żuka, z ang.beetle swarm optimization (BSO)-algorytm metaheurystyczny zwany optymalizacja grupy, czy też roju chrząszczy. Algorytm łączy mechanizm żerowania chrząszcza z algorytmem optymalizacji grupy i ustanawia model matematyczny, stosując go do funkcji unimodalnych, funkcji multimodalnych porównawczych o stałym wymiarze.

BSO-model matematyczny

Zmiany prędkości oraz położenia w algorytmie BSO zapisujemy jako:

$$X_{is}^{k+1} = x_{is}^{k} + \lambda v_{is}^{k} + (1-\lambda)\xi_{is}^{k},$$

$$v_{is}^{k+1} = \omega v_{is}^{k} + c_1 r_1 (p_{is}^{k} - x_{is}^{k}) + c_2 r_2 (p_{gs}^{k} - x_{gs}^{k}),$$

$$\xi_{is}^{k+1} = \delta^{k} \cdot v_{is}^{k} \cdot sign(f(x_{rs}^{k}) - f(x_{ls}^{k})),$$

$$x_{rs}^{k+1} = x_{rs}^{k} + v_{is}^{k} \cdot \frac{d}{2}; x_{ls}^{k+1} = x_{ls}^{k} - v_{is}^{k} \cdot \frac{d}{2}.$$

więcej na czacie...

BSO-cechy

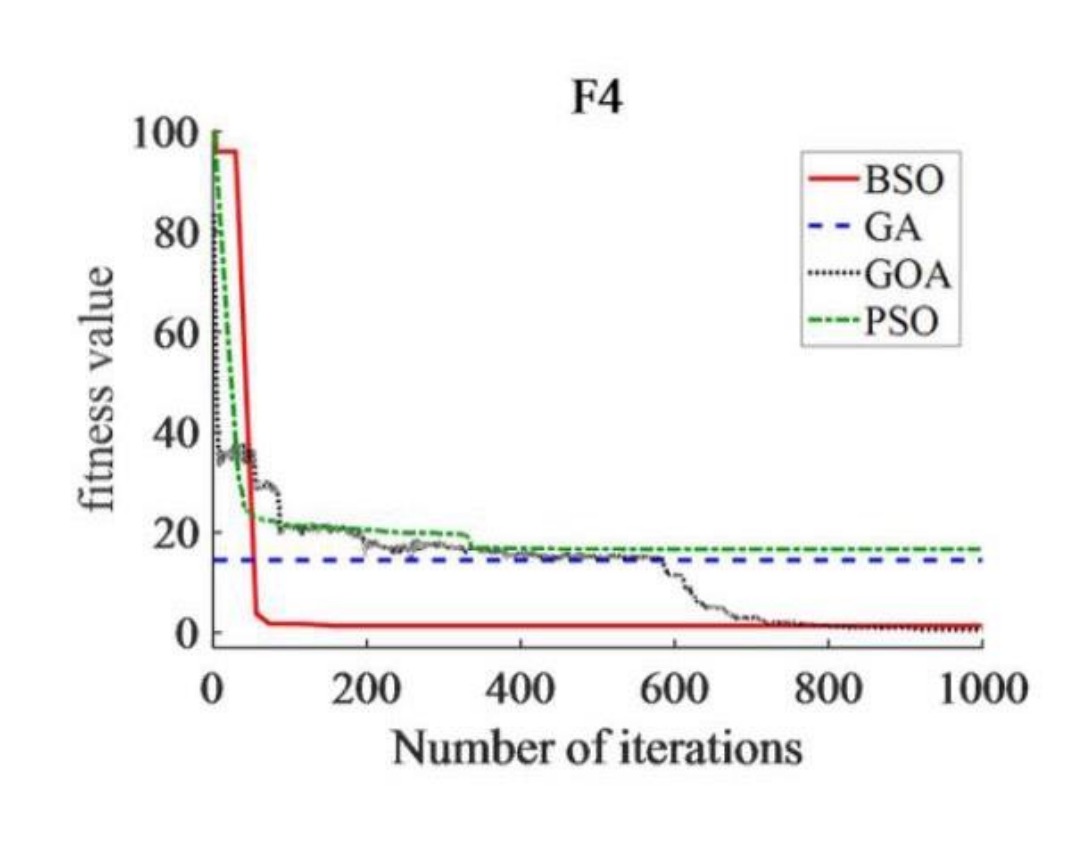
- Inspiracja algorytmami BSA i PSO pozwoliła na wprowadzenie ulepszonego algorytmu optymalizacji roju żuków (BSO).
- W tym algorytmie każdy żuk reprezentuje potencjalne rozwiązanie problemu optymalizacji, oraz każdy żuk odpowiada wartości sprawności określonej przez funkcję sprawności.
- Podobnie jak algorytm PSO, żuki również dzielą się informacjami, ale odległość i kierunek żuków zależy od ich prędkości i intensywności informacji wykrywanych przez ich długie czułki.

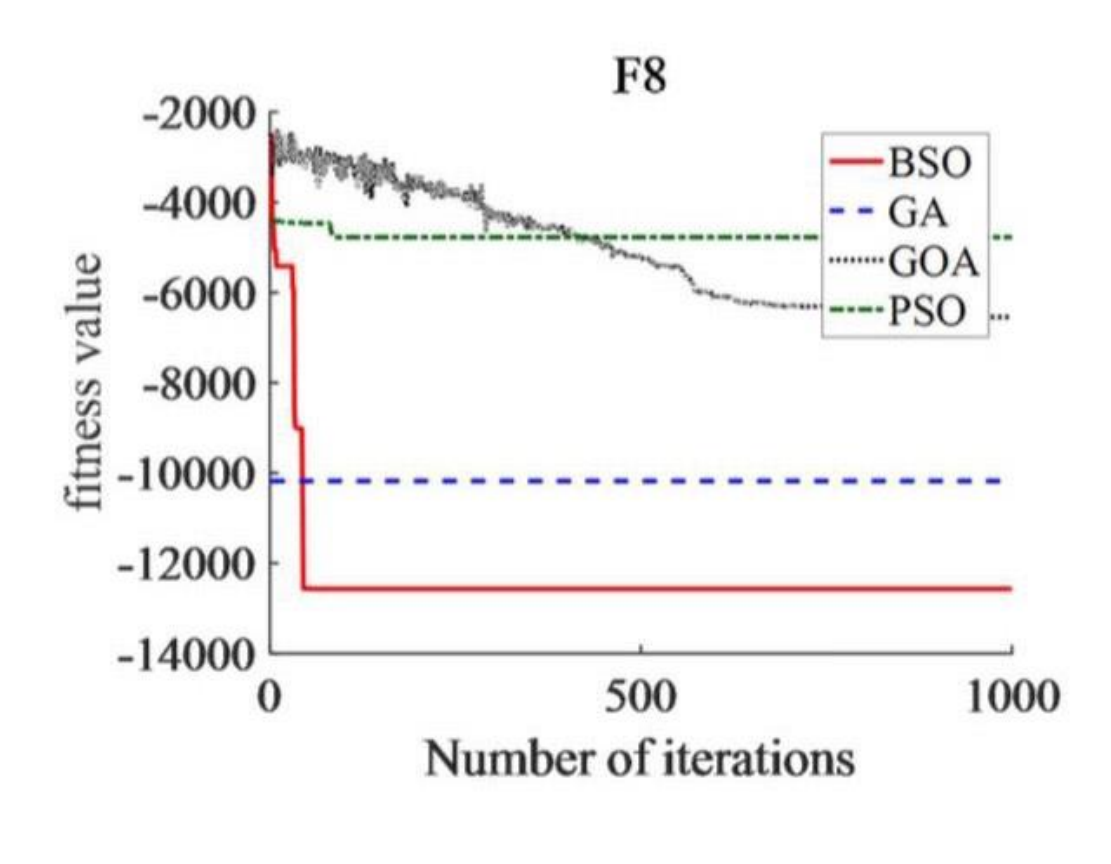
Porównanie wyników optymalizacji uzyskanych dla funkcji multimodalnych unimodalnych, multimodalnych i o stałym wymiarze

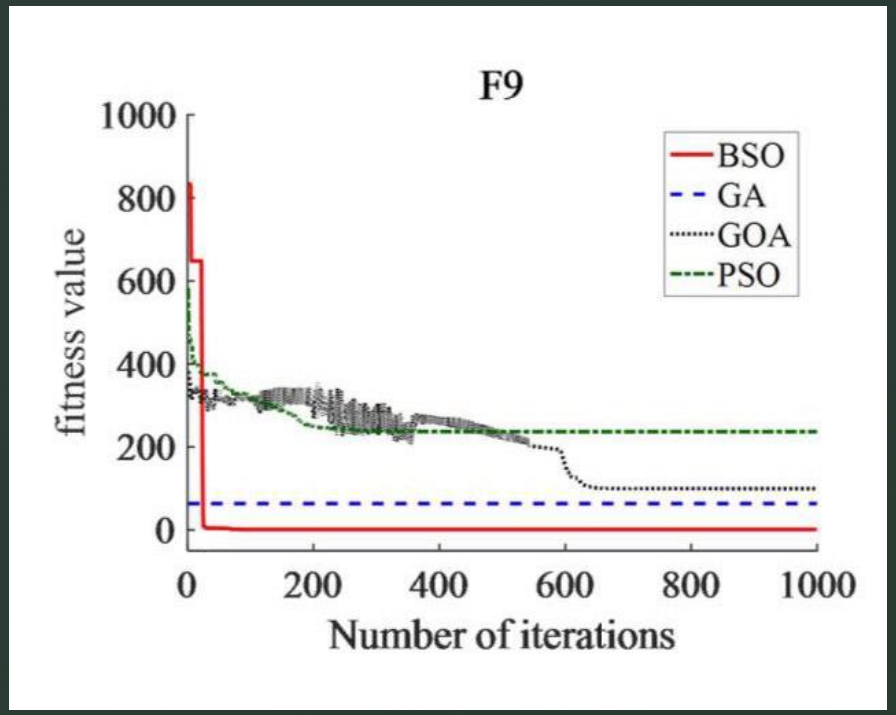
Table 5 Comparision of optimization results obtained for the unimodal, multimodal, and fixed-dimension multimodal functions

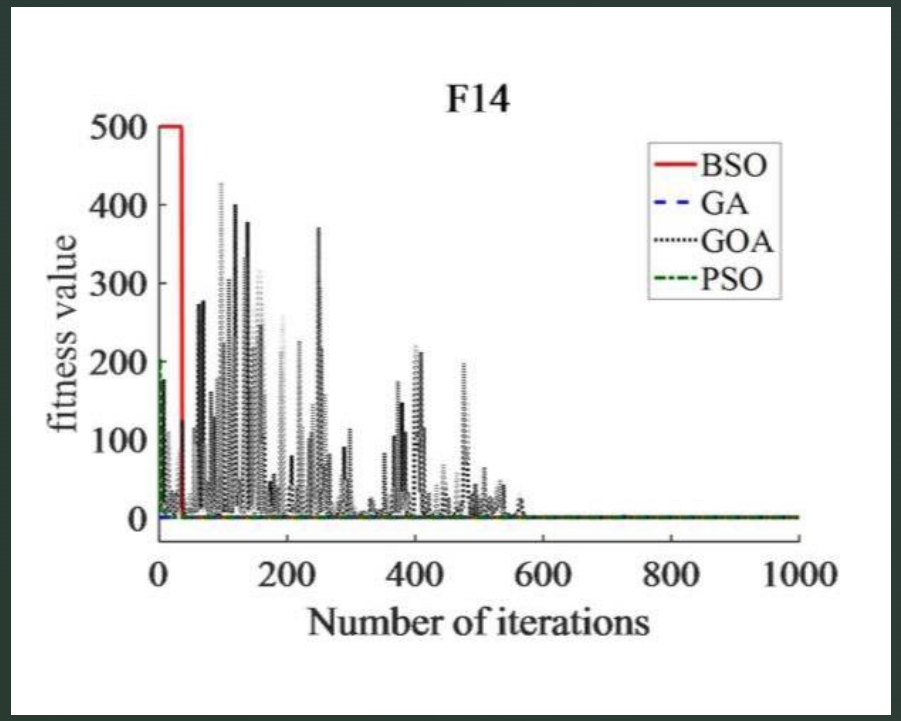
F	BSO			PSO			GA			GOA		
	ave	*std*	*ave_time*(s)	*ave*	*std*	*ave_time*(s)	*ave*	*std*	*ave_time*(s)	*ave*	*std*	*ave_time*(s)
F1	0	9.36E-76	0.5153	0	0	0.4597	0.0025	0.0017	3.7335	0.4004	0.3342	144.5615
F2	1.02E-04	3.92E-04	0.6127	1.3333	3.4575	0.5099	0.008	0.0068	3.7362	1.3612	2.0519	29.6388
F3	0	3.31E-72	0.8765	1.67E+02	912.8709	0.636	7.66E+03	2.34E+03	6.0088	0	0	29.8757
F4	3.55E-09	1.07E-08	0.4999	0	0	0.459	15.7727	4.8173	3.6927	2.50E-05	1.20E-05	29.5846
F5	0.6578	1.4017	0.6432	1.51E+04	3.41E+04	0.5247	43.927954	32.6768	3.7723	3.01E+03	1.64E+04	29.5752
F6	0	0	0.5081	0	0	0.4591	0.0007	0.0011	3.7253	0	0	29.5096
F7	5.17E-04	4.47E-04	0.6382	2.98E-04	0.0003	0.5219	0.0019	0.0009	3.9028	0.0737	0.1023	29.5672
F8	-1.79E+03	173.3453	0.6503	-1.40E+03	85.7482	0.532	-9.78E+03	373.5056	3.8002	-1.74E+03	183.2	29.7437
F9	0.4311	0.9305	0.5215	5.1785	9.0057	0.4683	59.7404	8.75764	3.7708	5.3052	2.9227	29.5213
F10	0.1097	0.4177	0.6282	4.6379	8.4257	0.5253	0.007	0.0051	3.7441	0.6931	0.9474	29.5833
F11	0.1267	0.0849	0.7203	0.1348	0.0926	0.5779	0.0725	0.1001	3.7637	0.1227	0.0638	29.7993
F12	7.00E-06	3.76E-05	1.424	0	0	0.9052	36.1241	9.0446	4.0032	0.0011	0.0059	29.9328
F13	0.0011	0.0034	1.4382	0	0	0.9123	57.65	12.9744	4.0068	0.0022	0.0044	29.9379
F14	0.998	1.54E-16	1.9211	0.998	0	3.1104	0.998	0	3.8205	0.998	0	12.3002
F15	0.0015	0.0036	0.586	0.0042	0.0117	0.4993	0.0039	0.00718	1.5158	0.0035	0.0067	19.9701
F16	-1.0316	6.71E-16	0.4534	-1.0316	0	0.4272	-1.0316	0	1.2441	-1.0316	0	10.306
F17	0.3979	0	0.5045	0.3979	0	0.5767	0.3979	0	1.2171	0.3979	0	10.2556
F18	3	1.03E-15	0.3853	3	0	0.4031	3.9	4.9295	1.2144	5.7	14.7885	10.3014
F19	-3.8609	0.0034	0.8683	-3.6913	0.1247	0.64	-3.8627	0	1.5927	-3.8369	0.1411	20.205
F20	-3.1256	0.3735	0.8685	-2.1198	0.5567	0.6541	-3.2625	0.0605	1.894	-3.2698	0.0607	29.4666
F21	-9.8164	1.2818	0.5519	-1.0902	0.8326	0.9072	-5.9724	3.37309	1.9346	-7.0499	3.2728	20.2475
F22	-10.0513	1.3381	0.6845	-1.0196	0.4063	1.0713	-7.3119	3.4237	2.1298	-7.3062	3.4705	20.4859
F23	-9.1069	2.4111	0.9185	-1.2161	0.6276	1.3545	-5.7112	3.5424	2.4214	-8.6298	3.0277	20.5744

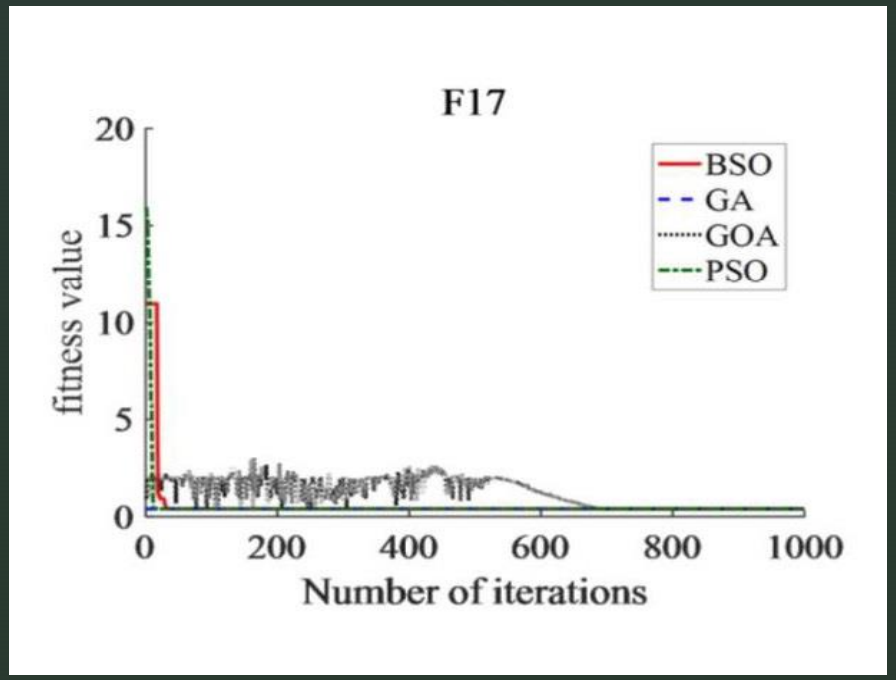
Porównanie wykresów algorytmów wybranych funkcji z tabeli

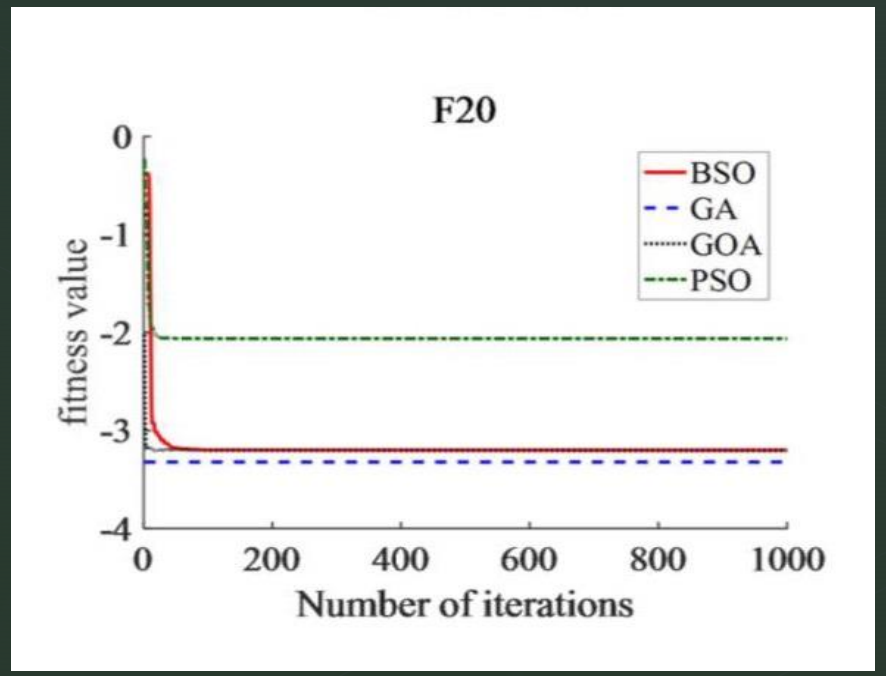





F9
1000
800
600
400
200
0
fitness value
0
200
400
600
800
1000
Number of iterations
BSO
GA
GOA
PSO


F14
500
400
300
200
100
0
fitness value
0
200
400
600
800
1000
Number of iterations
BSO
GA
GOA
PSO


F17
20
15
10
5
0
fitness value
0
200
400
600
800
1000
Number of iterations
BSO
GA
GOA
PSO

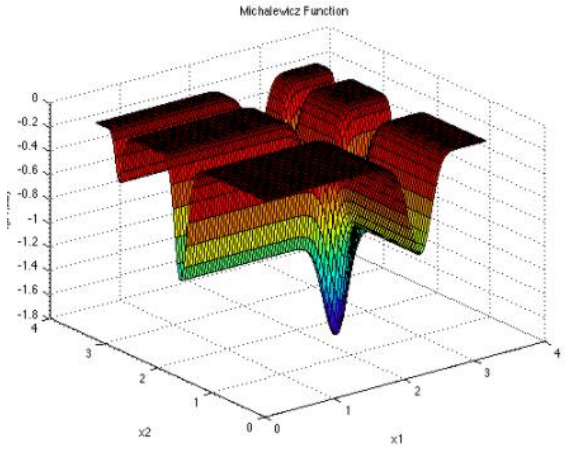

F20
0
-1
-2
-3
-4
fitness value
0
200
400
600
800
1000
Number of iterations
BSO
GA
GOA
PSO

Analiza zachowania krzywych

Krzywe konwergencji BS0, GA, GOA i PSO porównano na wykresach dla kilku funkcji testowych. Rysunek pokazuje, że BSO ma dobrą zdolność przetwarzania funkcji unimodalnych, funkcji multimodalnych i funkcji o stałym wymiarze, a proces przetwarzania jest bardzo stabilny. Zwłaszcza przy rozwiązywaniu bardziej złożonych funkcji o stałym wymiarze BSO wykazuje bardziej oczywistą przewagę niż inne algorytmy. Można zauważyć, że BSO jest wystarczająco konkurencyjny w stosunku do innych najnowocześniejszych algorytmów meta-heurystycznych.

Przykład

Funkcja Michalewicza



$$f(x) = -sin(x_i) \cdot (sin(\frac{i \cdot x_i^2}{\pi}))^{2m},$$

dla: $i = 1, \ldots, n;\ m = 10;\ 0 \leqslant x_i \leqslant \pi$

Algorytm RStudio - Funkcja Michalewicza obliczona za pomocą algorytmu BSO

```
1  library(devtools)
2  library(rBAS)
3  mich <- function(x){
4    y1 <- -sin(x[1])*(sin((x[1]^2)/pi))^20
5    y2 <- -sin(x[2])*(sin((2*x[2]^2)/pi))^20
6    return(y1+y2)
7  }
8  result <-
9    BSOoptim(fn = mich,
10             init = NULL,
11             lower = c(-6,0),
12             upper = c(-1,2),
13             n = 100,
14             step = 5,
15             s = 10,seed = 1, trace = F)
16 result$par; result$value
```

ROZWIĄZANIE

Minimum globalne

Przy m = 10

Dla (x1, x2) = (-4.965998, 1.570796)

Wtedy

F(x1, x2) = -1.967851

Wnioski

- Wyniki pokazują, że w porównaniu z obecnie popularnymi algorytmami optymalizacji algorytm BSO może nadal dawać bardzo konkurencyjne wyniki, a także ma dobrą solidność i szybkość działania. Ponadto algorytm BSO wykazuje również wyższą wydajność w przypadku ograniczeń nieliniowych. W porównaniu z innymi algorytmami optymalizacji, BSO może skutecznie i stabilnie obsługiwać problemy optymalizacji wielu celów.

Bibliografia:

https://arxiv.org/ftp/arxiv/papers/1808/1808.00206.pdf

https://ieeexplore.ieee.org/abstract/document/8955813

http://www.ieee-jas.org/fileZDHXBEN/journal/article/zdhxbywb/2020/2/PDF/JAS-2019-0399.pdf

https://www.groundai.com/project/convergence-analysis-of-beetle-antennae-search-algorithm-and-its-applications/1

https://arxiv.org/ftp/arxiv/papers/1808/1808.00206.pdf

https://www.mathworks.com/matlabcentral/fileexchange/64881-bas-beetle-antennae-search-algorithm-for-optimization

Dziękujemy za uwagę