Przeszukiwanie i optymalizacja - dokumentacja końcowa

Maksym Bieńkowski, Jędrzej Grabski

29.01.2025

Temat projektu: Algorytm roju cząstek z modyfikacjami dotyczącymi współczynnika bezwładności

1 Analiza problemu

Ideą Algorytmu Roju Cząstek (PSO - Particle Swarm Optimalization), jest symulowanie populacji ("roju"), która rozwija się na podstawie wiedzy pojedynczych osobników ("cząstek") oraz pewnej wiedzy dzielonej. Każda z cząstek posiada swoją prędkość oraz pozycję w przestrzeni rozwiązań. Ponadto zapamiętywane jest najlepsze rozwiązanie znalezione do tej pory przez każdą z cząstek (optimum lokalne, niewspółdzielone z resztą populacji), a także najlepsze rozwiązanie z całego roju (optimum globalne, współdzielone przez wszystkie cząstki).

Prędkość i-tej cząstki w epoce $k+1$ dana jest następującym wzorem:

$$V_i(k+1) = wV_i(k) + \phi_p r_1(P_i(k) - X_i(k)) + \phi_g r_2(P_g(k) - X_i(k))$$

gdzie w oznacza współczynnik bezwładności, P_i położenie optimum lokalnego, P_g położenie optimum globalnego, współczynniki r_1 i r_2 losowane są z rozkładem $U[0,1]$, a ϕ_p oraz ϕ_g oznaczają kolejno parametryzowane współczynniki wagi - poznawczy i społeczny. Na podstawie powyższego wzoru obserwujemy, że wektor prędkości tworzony jest na podstawie trzech składowych, a współczynnik bezwładności określa wagę składowej będącej prędkością w poprzedniej iteracji. Im mniejsza wartość tego współczynnika, tym bardziej zwrotne i skłonne do eksploatacji są cząstki. Ze zwiększeniem wartości współczynnika bezwładności wiąże się natomiast większa skłonność do eksploracji przestrzeni, co może jednak skutkować "przestrzeliwaniem" optimów lokalnych.

2 Badane rozwiązanie

Problem sformułowany w poprzedniej sekcji spróbujemy zniwelować poprzez wprowadzenie dynamicznej zmiany współczynnika bezwładności, uzależniając go od liczby wykonanych iteracji. Współczynnik będzie stopniowo zmniejszany się w miarę pracy algorytmu. Umożliwi to skupienie się na eksploracji w początkowej fazie algorytmu, a

następnie bardziej precyzyjne zbieganie wokół optimów pod koniec pracy. W iteracji k wartość współczynnika bezwładności opisana jest wzorem

$$w_k = w_{k-1}u^{-k}$$

gdzie $u \in [1.001, 1.005]$ na podstawie literatury.

3 Sposób przeprowadzania badań, przyjęte założenia

Badania przeprowadzone zostały w ciągłej, ograniczonej wielowymiarowej przestrzeni z dobrze zdefiniowanymi wartościami funkcji celu w każdym punkcie. Zbadane zostało działanie algorytmu zarówno dla funkcji jedno-, jak i wielomodalnych.Ograniczenia przestrzeni zostały zrealizowane w wariancie lamarkowskim, przy pomocy rzutowania na ograniczenia w każdym wymiarze ograniczeń kostkowych. Ustaliliśmy klasyczne ograniczenia $[-2.048, 2.048]$ dla każdego wymiaru.

Algorytmy badaliśmy na następujących funkcjach:

- **Funkcja sferyczna**

  $$y(x) = \sum_{i=0}^{d} x_i^2$$

  unimodalna funkcja kwadratowa, trywialne zadanie optymalizacji, minimum globalne $f(x^*) = 0$ dla $x^* = 0$.

- **Funkcja Rastrigina**

  $$f(x) = Ad + \sum_{i=1}^{d} \left(x_i^2 - A\cos(2\pi x_i)\right)$$

  klasyczna wielomodalna funkcja stosowana do ewaluacji algorytmów optymalizacyjnych posiadająca wiele minimów lokalnych. Minimum globalne $f(x^*) = 0$ dla $x^* = 0$. Zastosowaliśmy klasyczny wariant z $A = 10$.

- **Funkcja Ackleya**

  $$f(x) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

  Kolejna znana wielomodalna funkcja benchmarkowa, przyjęliśmy $a = 20, b = 0.2, c = 2\pi$. Minimum globalne $f(x^*) = 0$ dla $x^* = 0$.

- **Funkcja Rosenbrocka**

  $$f(x) = \sum_{i=1}^{d-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$$

  Liczba optimów funkcji zależy od wymiarowości - dla $d <= 3$ posiada ona jedno minimum globalne $f(x^*) = 0$ dla $x^* = 1$. Charakteryzuje się płaską, wykrzywioną doliną, dotarcie do optimum wymaga precyzyjnej nawigacji i bywa kosztowne czasowo.

W powyższych równaniach d oznacza rozmiar przestrzeni. Wyniki były uśredniane z 50 uruchomień dla wymiarowości $d \in \{5, 10, 20\}$. Rozwiązanie było oceniane względem dwóch kryteriów - wartości f. celu w znalezionym punkcie oraz szybkości zbieżności algorytmów - iteracji wymaganych do odnalezienia tego punktu.

3.1 Parametry algorytmu

Ze względu na dużą liczbę parametrów, część z nich musiała zostać ustalona, aby posiadany przez nas sprzęt sprostał wymaganiom czasowym:

- We wszystkich przypadkach populacja liczyła 50 osobników.
- Przy każdym uruchomieniu $\phi_p = \phi_g = 2.0$.
- Początkowe pozycje cząstek generowane były zgodnie z rozkładem jednostajnym w granicach przestrzeni rozwiązań.
- Początkowa prędkość cząstek była generowana z rozkładem jednostajnym w zakresie $[-\frac{x_{max,j}-x_{min,j}}{2}, \frac{x_{max,j}-x_{min,j}}{2}]$ dla każdego wymiaru j, gdzie x_{max} i x_{min} oznaczają odpowiednio górne i dolne ograniczenie w tym wymiarze.
- Za każdym razem algorytmy były uruchamiane z maksymalną liczbą 300 iteracji oraz $eps = 1e - 5$. Algorytm zapamiętuje, w której iteracji dotarł do epsilona, co pozwala zmierzyć zbieżność do punktu o określonej jakości. W praktyce tak niska wartość okazała się oczywiście zbyt niska dla wielowymiarowych funkcji i wymagałaby indywidualnego arbitralnego strojenia w zależności od funkcji i wymiaru, brana była pod uwagę przy badaniu 6.
- Wartość współczynnika bezwładności lub, w przypadku dynamicznego współczynnika bezwładności $w = 0.1$
- Wartość współczynnika u używanego do obniżania wartości bezwładności $u = 1.0001$

Początkowa wartość współczynnika bezwładności została wybrana eksperymentalnie przez zbadanie najlepszej wartości f. Rosenbrocka w 10 wymiarach w zależności od jego wartości. Wyniki znajdują się w tabeli 1.

DI / ω_0	0.05	0.1	0.2	0.4	0.7
Nie	36	34	47	62	1798
Tak	30	29	40	60	1539

Tabela 1: Zaokrąglona wartość f. Rosenbrocka dla najlepszego znalezionego punktu w zależności od startowego współczynnika bezwładności

Badanie było przeprowadzone dla różnych funkcji, wyniki były zbliżone. Analogiczna próba została przeprowadzona w celu ustalenia optymalnej wartości współczynnika u.

4 Wyniki badań

Porównanie ze względu na jakość rozwiązań

DI / Dim	5	10	20
Nie	$8.90e^{-12}$	0.0184	2.2144
Tak	$6.93e^{-12}$	0.0243	2.2864

Tabela 2: Wartość f. sferycznej dla najlepszego znalezionego punktu w zależności od wymiaru

DI / Dim	5	10	20
Nie	3.546	29.157	413.303
Tak	2.516	33.790	414.403

Tabela 3: Wartość f. Rosenbrocka dla najlepszego znalezionego punktu w zależności od wymiaru

DI / Dim	5	10	20
Nie	7.959	18.453	44.838
Tak	7.697	19.518	46.823

Tabela 4: Wartość f. Rastrigina dla najlepszego znalezionego punktu w zależności od wymiaru

DI / Dim	5	10	20
Nie	0.346	1.486	3.102
Tak	0.402	1.578	3.038

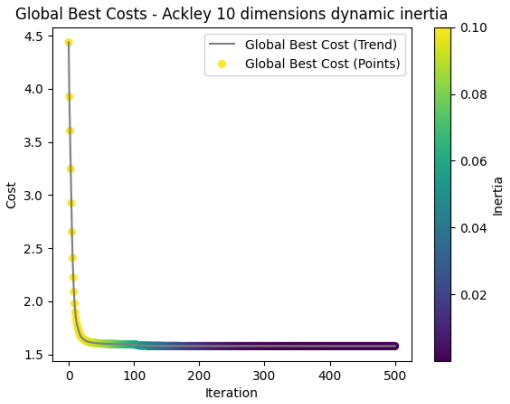
Tabela 5: Wartość f. Ackleya dla najlepszego znalezionego punktu w zależności od wymiaru

Porównanie ze względu na zbieżność

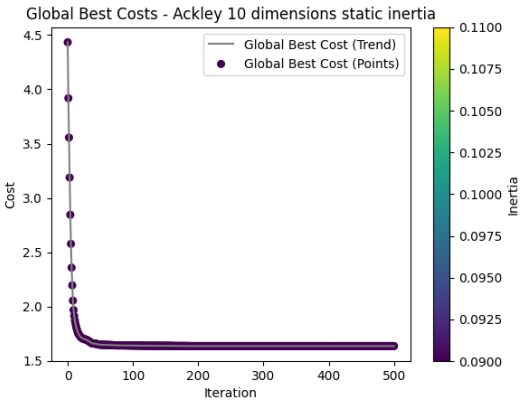
DI / Funkcja	Sferyczna	Ackley	Rosenbrock	Rastrigin
Nie	37	57	NA	NA
Tak	27	243	NA	NA

Tabela 6: Uśrednione iteracje do znalezienia *eps*, 2 wymiary

Zmiana najlepszego punktu na przestrzeni iteracji dla wybranych funkcji

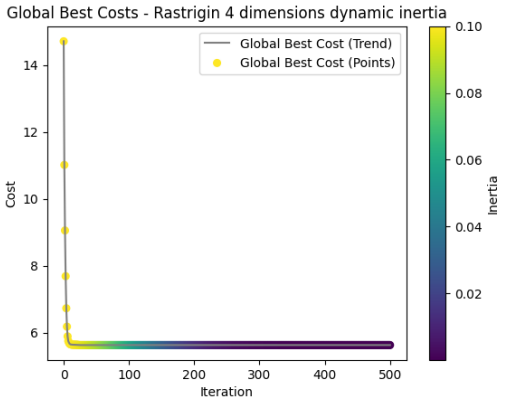


(a) Tempo zbieżności algorytmu z dynamiczną inercją - f. Ackleya, 10 wymiarów

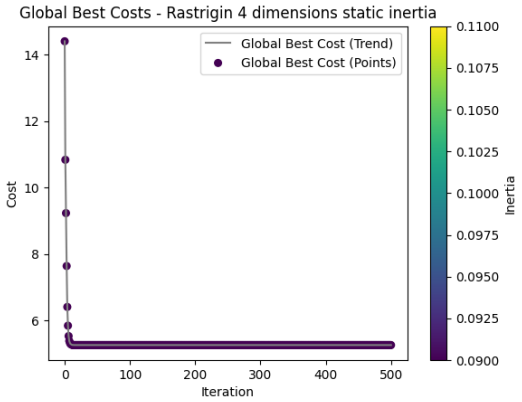


(b) Tempo zbieżności algorytmu ze statyczną inercją - f. Ackleya, 10 wymiarów

Rysunek 1: Porównanie tempa zbieżności algorytmu dla dynamicznej i statycznej inercji w 20 wymiarach dla 300 iteracji

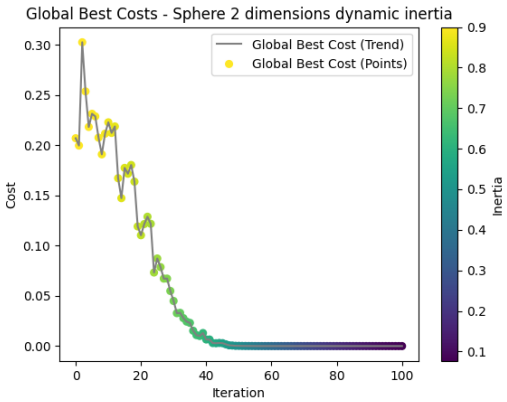


(a) Tempo zbieżności algorytmu z dynamiczną inercją - f. Rastrigina, 4 wymiary

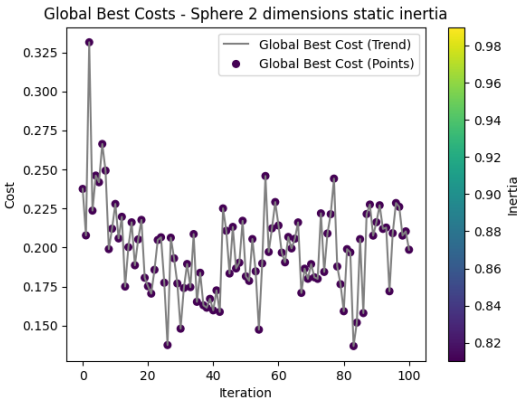


(b) Tempo zbieżności algorytmu ze statyczną inercją - f. Rastrigina, 4 wymiary

Rysunek 2: Porównanie tempa zbieżności algorytmu dla dynamicznej i statycznej inercji w 4 wymiarach dla 300 iteracji

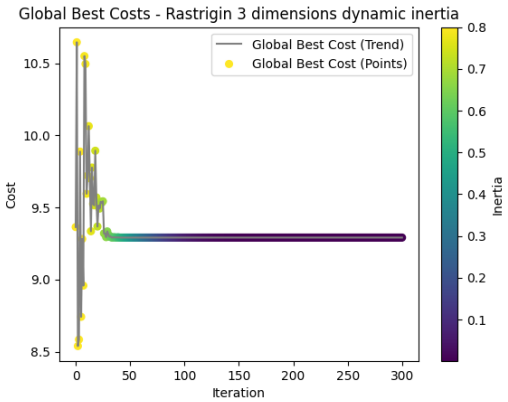


(a) Tempo zbieżności algorytmu z dynamiczną inercją - f. sferyczna, 2 wymiary

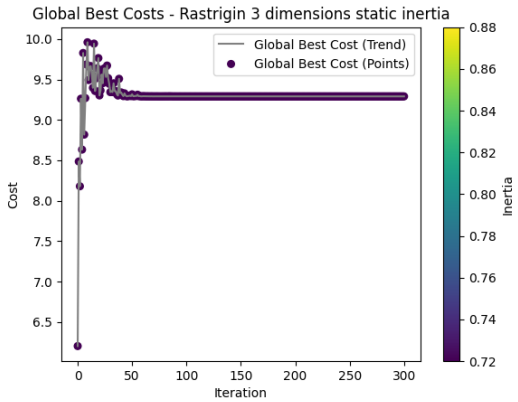


(b) Tempo zbieżności algorytmu ze statyczną inercją - f. sferyczna, 2 wymiary

Rysunek 3: Porównanie tempa zbieżności algorytmu dla dynamicznej i statycznej inercji w 2 wymiarach dla 100 iteracji



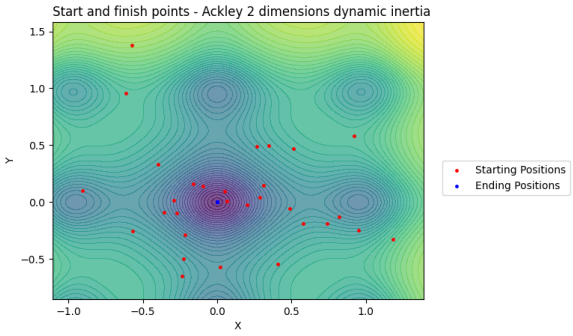
(a) Tempo zbieżności algorytmu z dynamiczną inercją - f. Rastrigina, 3 wymiary



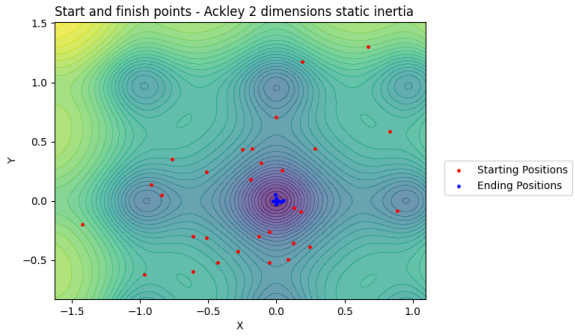
(b) Tempo zbieżności algorytmu ze statyczną inercją - f. Rastrigina, 3 wymiary

Rysunek 4: Porównanie tempa zbieżności algorytmu dla dynamicznej i statycznej inercji w 3 wymiarach dla 300 iteracji

Wizualizacja punktów przed i po działaniu algorytmu



(a) Uśrednione punkty startowe i końcowe - f. Ackleya, 2 wymiary, dynamiczna inercja



(b) Tempo zbieżności algorytmu ze statyczną inercją - f. Rastrigina, 2 wymiary

Rysunek 5: Uśrednione punkty startowe i końcowe - f. Ackleya, 2 wymiary, statyczna inercja

Wnioski

Na podstawie tabel 2, 3, 4, 5 możemy stwierdzić, że zastosowanie dynamicznego współczynnika bezwładności nie wydaje się negatywnie wpływać na jakość znajdowanych rozwiązań.

Wykresy 1 oraz 2 wydają się pokazywać, że dla nastaw, które wydawały się optymalne, w większych wymiarowościach nie jest widoczne tak duże przyspieszenie zbieżności, jak możnaby się spodziewać. Nie wykluczamy, że jest to spowodowane konkretnymi przyjętymi parametrami algorytmu. Przeprowadziliśmy także badania dla mniejszej liczby iteracji, jednak wydaje się, że w większej wymiarowości poprawa nie jest odczuwalna. W analizowanej literaturze przyspieszenie wystąpiło wyłącznie dla niektórych funkcji benchmarkowych - być może obierając inne przypadki wyniki byłyby bardziej obiecujące.

Przy doborze innych parametrów udało się nam zauważyć korzyści wynikające z zastosowania dynamicznego współczynnika bezwładności, w szczególności dla:

- wysokiej bezwładności początkowej
- niższego wymiaru problemu
- niższej liczby możliwych iteracji
- mniejszej populacji

Wobec tego, jeśli obliczenie funkcji celu jest kosztowne i możemy sobie pozwolić wyłącznie na niższą liczbę iteracji, istnieje korzyść z zastosowania tego rozwiązania. Ponadto, zgodnie z wykresem 3, dzięki gradualnemu obniżaniu bezwładności, algorytm ma mniejszą tendencję do przypadkowego błądzenia po przestrzeni rozwiązań i dokładniej eksploatuje znalezione minima. Porównanie 4 pokazuje, że wersja z dynamiczną

bezwładnością wydaje się mieć również tendencje do znajdywania podobnych rozwiązań dla gorszych punktów startowych. W wizualizacji 5 obserwujemy, że przy zastosowaniu dynamicznej inercji punkty końcowe są gęściej skoncentrowane w optimum niż w przypadku bez inercji. Jest to spowodowane zwiększoną eksploatacją w końcowych fazach algorytmu. Dla niższej wymiarowości i odpowiednich parametrów, algorytm znajdował odpowiednio dobre punkty szybciej, co zobrazowane zostało w tabeli 6.

W kwestii porównania zachowań algorytmu w zależności od funkcji, najtrudniejszym zadaniem wydaje się funkcja Rastrigina. Dla wyższych wymiarowości niż 4 nie znajdował on konsekwentnie optimum globalnego i cechował się tendencją do utykania w napotkanych optimach lokalnych, co także obrazuje porównanie 4.

5 Bibliografia

- Particle swarm optimization. (1995). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/488968
- Jiao, B., Lian, Z., & Gu, X. (2006). A dynamic inertia weight particle swarm optimization algorithm. Chaos Solitons & Fractals, 37(3), 698–705. https://doi.org/10.1016/j.chaos.2006.09.063