RAPORT ZALICZENIOWY

Nieklasyczne Metody Optymalizacji

Cuckoo search

(algorytm kukułki)

Autorki:

Julia Nykiel (jn108767@student.sgh.waw.pl)

Justyna Zbiegień (jz107961@student.sgh.waw.pl)

Warszawa 2022

Wprowadzenie

Cuckoo search (CS), czyli algorytm kukułki to metoda globalna należąca do grupy algorytmów inspirowanych naturą. Została przedstawiona w 2009 roku przez Xin-She Yang oraz Suash Deb i bazuje na strategii lęgowej kukułek. Okazuje się bowiem, że wspomniany gatunek ptaków wyróżnia się nie tylko charakterystycznym głosem, któremu zawdzięcza swoją nazwę, ale również stylem życia. Jest ptakiem skrytym i tajemniczym, częściej słyszanym niż widzianym, a jego biologia lęgowa odbiega znacznie od typowego zachowania ptaków.

Strategia pasożytów lęgowych

Strategia lęgowa kukułek fascynowała człowieka od najdawniejszych czasów. Arystoteles jako pierwszy stwierdził, że ten gatunek to pasożyt lęgowy już w IV wieku p.n.e., a kolejne obserwacje potwierdziły jego przypuszczenia (Rosin i Tryjanowski, 2010).

Kukułki, w celu zwiększenia przeżywalności swojego gatunku, podrzucają swoje jaja do gniazd innych ptaków. Jaja składane są pojedynczo, a w celu zatuszowania oszustw jedno z jaj gospodarzy zostaje wyrzucone z gniazda lub połknięte przez samicę kukułki. Dzięki temu liczba jaj znajdujących się w gnieździe pozostaje taka sama i trudniej jest zauważyć jakąś zmianę. Cała pasożytnicza procedura wymaga sporego sprytu i przygotowania. Według ornitologów samce pomagają samicą skupiając na sobie uwagę okolicznych ptaków. Ptaki ruszają na zwiady lub do walki i pozostawiają gniazda puste, dzięki czemu samica może w spokoju podmienić jajo. Ponadto wybranie gatunku przybranych rodziców nie może być przypadkowe, każda kukułka wybiera ten gatunek, którego jaja najbardziej przypominają ubarwienie jej własnych jaj. Tym sposobem zwiększają skuteczność podstępu. Co więcej, podrzucenie musi odbyć się w odpowiednim momencie procesu lęgu potencjalnych opiekunów, ponieważ jajo podrzucone w złym okresie jest wysoce narażone na odrzucenie. Pisklę kukułki z reguły wykluwa się najwcześniej. Mimo tego, że jest jeszcze ślepe posiada instynkt dzięki któremu wypycha jaja lub świeżo wyklute pisklęcia z gniazda. Pozbywa się konkurentów o pokarm i opiekę, zwiększając tym samym swoje szanse na przeżycie.

Istnieje jednak prawdopodobieństwo, że jajo bądź pisklę kukułki zostanie wykryte przez przybranego rodzica. W takim przypadku zostaje ono wyrzucone z gniazda lub ptaki porzucają gniazdo i zakładają nowe, w innym miejscu.

Loty Levy'ego

Ptaki, owady, jak i inne zwierzęta przemieszczają się w dość specyficzny sposób. Jest to seria prostych kroków, po których następuje nagła, gwałtowna zmiana kierunku poruszania się. Ten ruch w przybliżony sposób da się zapisać matematycznie. Jedna z propozycji tego rodzaju zapisu to tzn. lot Levy'ego (Yang i Deb, Cuckoo Search via Levy Flights, 2009).

Lot Levy'ego to błądzenie losowe o losowych kierunkach i długościach kroku pochodzących z rozkładu Levy'ego. Pojedynczy ruch można zapisać następująco:

$$x_i^{t+1} = x_i^t + \alpha \oplus Levy(\lambda),$$

gdzie x_i^t to aktualna pozycja i-tego punktu, x_i^{t+1} to jego przyszła pozycja, α to długość kroku, a $Levy(\lambda)$ to wartość uzyskana z rozkładu. W praktyce często korzysta się z trochę inaczej

wyglądającego, ale oznaczającego to samo wzoru (Yang, Nature-Inspired Optimization Algorithms, 2014):

$$x_i^{t+1} = x_i^t + rand(size(x_i)) \oplus stepsize,$$

gdzie $rand(size(x_i))$ reprezentuje losowy wybór kierunku z rozkładu jednostajnego, a $stepsize$ to wielkość kroku. Policzenie wielkości kroku wymaga trochę większego wkładu niż wylosowanie kierunku ruchu. Istnieje kilka sposobów uzyskania szukanej wielkości. Jednym z nich jest algorytm Mantegna (*Mantegna's algorithm*), który generuje pseudolosowe liczby pochodzące z rozkładu Levy'ego przy użyciu dwóch pomocniczych zmiennych pochodzących z rozkładów normalnych, v$\sim N(0,1)$ i $u \sim N(0, \sigma_u^2)$ połączonych w następującą relację:

$$s = \frac{u}{|v|^{1/\beta}},$$

gdzie β to pewien parametr skalujący z przedziału [1,2] (w praktyce często używa się $\beta = 3/2$), a nieznaną wariancję u można policzyć jak poniżej:

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\beta\,\Gamma[(1+\beta)/2]\;2^{(\beta-1)/2}} \right\}^{1/\beta}$$

Jednak w implementacji często pomija się użycia tego wzoru ze względu na funkcję gamma i używa się odgórnie ustalonej, stałej wartości wariancji. Po policzeniu s można przejść do końcowej formuły:

$$stepsize = \alpha s \cdot (x_i - x_{best}),$$

gdzie α to współczynnik skalujący, często ustalany na poziomie 0.01, a x_{best} to najlepszy punkt (tzn. *global best*), czyli punkt dla którego funkcja celu osiąga najlepszą wartość (najmniejszą lub największą w zależności od problemu optymalizacyjnego). Tak otrzymaną wielkość kroku $stepsize$ można podstawić do wspomnianej wcześniej formuły otrzymując nową pozycję punktu.

Przebieg algorytmu

Wiedząc już jak będą się poruszać kukułki, możemy przejść dalej – do trzech najważniejszych reguł algorytmu. Czytając je należy myśleć o jajach kukułki jak o potencjalnych rozwiązaniach.

1. Każda kukułka składa jedno jajo w danym momencie czasu w losowo wybranym gnieździe.
2. Najlepsze gniazda z wysoką jakością jaj będą przenoszone na następne pokolenia (elityzm).
3. Liczba dostępnych gniazd jest stała, a jajo podrzucone przez kukułkę jest wykryte z prawdopodobieństwem $p_a \in [0,1]$.

Ogólnym celem jest zastąpienie słabego rozwiązania w gnieździe, nowym i potencjalnie lepszym rozwiązaniem (jajem kukułki). W podstawowym podejściu każde gniazdo może mieć tylko jedno jajo, zatem jajo, gniazda i kukułki są nierozróżnialne.

Algorytm rozpoczyna się od wygenerowania początkowej populacji. Zamiana rozwiązania może dokonać się dzięki losowemu wybraniu kukułki (jednego z aktualnych rozwiązań) i przesunięciu jej przy użyciu lotu Levy'ego uzyskując tym samym nowe jajo (potencjalne rozwiązanie). Dzięki podstawieniu nowego rozwiązania do funkcji celu uzyskujemy jego jakość, nazywaną również przystosowaniem. Następnie losowo zostaje wybrane gniazdo zawierające inne jajo (losowe wybranie rozwiązanie), jego jakość zostaje obliczona podobnie jak poprzednio. Jajo znajdujące się w tym gnieździe jest kandydatem do wymiany, jednak to czy kukułka może podmienić jajo, czy nie zależy od obliczonych wartości jakości

jaj (ich przystosowań). Kukułka nie podmieni jaja jeśli jest ono źle przystosowane – ma inny kolor, rozmiar etc., bo wtedy jajo ma mniejsze szanse na przeżycie od tego znajdującego się aktualnie w gnieździe. Dla problemów minimalizacji w gnieździe pozostanie jajo z niższą wartością funkcji celu, a dla maksymalizacji z wyższą. Jednak to nie koniec procesu, ponieważ istnieje pewne prawdopodobieństwo, że podłożone jajo zostanie wykryte przez nowego rodzica.

To prawdopodobieństwo $\boldsymbol{p_a}$ odpowiada za balans pomiędzy lokalnym, a globalnym błądzeniem w algorytmie, dając dzięki temu lepszą możliwość przeszukiwania przestrzeni rozwiązań. Standardowa wartość to 0.25, jednak w różnych badaniach rozważa się również zmieniające się wartości tego parametru. Krok algorytmu odpowiadający za zmianę położenia gniazda z prawdopodobieństwem $\boldsymbol{p_a}$ w momencie wykrycia przekrętu kukułki w praktyce rozpoczyna się od generowania liczb losowych z przedziału [0,1]. Na ich podstawie decyduje się, czy dane gniazdo zostanie przeniesione, czy nie w zależności od tego, czy przekraczają zadany próg. W podstawowym algorytmie wykryte gniazda zostają przeniesione – losuje się im nowe miejsce (generuje się nowe, randomowe rozwiązanie).

W późniejszych publikacjach nowa pozycja bazuje na podobieństwie jaj, jako że to właśnie od jakości jaja zależy to, czy zostanie ono wykryte jako obce. Podobieństwo wyrażone jest poprzez odległość między rozwiązaniami w gniazdach i nowa pozycja liczona jest za pomocą formuły (Yang i Karamanoglu, Nature-inspired computation and swarm intelligence, 2020):

$$x_i^{t+1} = x_i^t + \alpha s \oplus (x_j^t - x_k^t),$$

gdzie x_j^t i x_k^t to aktualne rozwiązania uzyskane przez losową permutację.

Podsumowując wszystkie informacje, podstawowy algorytm kukułki może być zapisany w postaci poniższego pseudokodu:

Wygeneruj początkową populację $\boldsymbol{n}$ gniazd $\boldsymbol{x_1}, \dots, \boldsymbol{x_n}$.

Dopóki ($\boldsymbol{t} < \boldsymbol{iter}$) lub (kryterium stopu)

Wybierz losowo i -tą kukułkę i wygeneruj nowe jajo (rozwiązanie) posługując się lotem Levy'ego, oceń jakość tego rozwiązania $\boldsymbol{F(x_i)}$.

Wybierz losowo j -te gniazdo $\boldsymbol{x_j}$ i oceń jakość tego rozwiązania $\boldsymbol{F(x_j)}$.

Jeżeli $\boldsymbol{F(x_i)} > \boldsymbol{F(x_j)}$ dla maksymalizacji lub $\boldsymbol{F(x_i)} < \boldsymbol{F(x_j)}$ dla minimalizacji

Zamień $\boldsymbol{x_j}$ na $\boldsymbol{x_i}$

Porzuć rozwiązania z prawdopodobieństwem $\boldsymbol{p_a}$ i zbuduj nowe gniazdo używając losowego przesunięcia.

Zapamiętaj najlepsze gniazda.

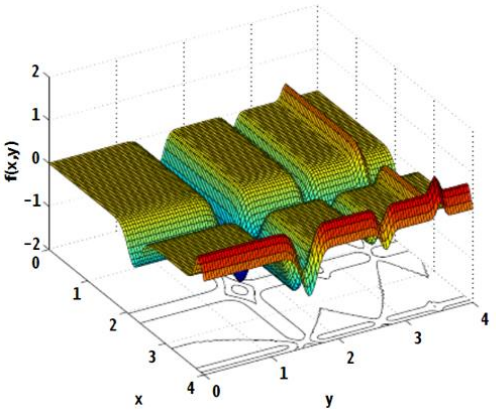
Utwórz ranking dla osobników z populacji i znajdź aktualnego $\boldsymbol{x_{best}}$.

Oznaczenia:

funkcja celu $\boldsymbol{F(x)}$, szukane rozwiązanie $\boldsymbol{x} = [\boldsymbol{x1}, \boldsymbol{x2}, \dots, \boldsymbol{xm}]$, liczba gniazd $\boldsymbol{n}$, i-te gniazdo $\boldsymbol{x_i}$ ($\boldsymbol{i} = \boldsymbol{1}, \boldsymbol{2}, \dots, \boldsymbol{n}$), przystosowanie i -tego jaja $\boldsymbol{F(x_i)}$, prawdopodobieństwo wykrycia obcego jaja w gnieździe $\boldsymbol{p_a}$, maksymalna liczba iteracji $\boldsymbol{iter}$, aktualna iteracja $\boldsymbol{t}$.

Implementacja i przykład użycia

W celu przedstawienia przykładowego rozwiązania zagadnienia optymalizacyjnego opartego na tej metodzie, algorytm kukułki został zaimplementowany w programie R. Inspiracją do napisania dołączonego do tego raportu kodu źródłowego był kod napisany w MATLABie zamieszczony w książce *Nature-inspired computation and swarm intelligence* (Yang i Karamanoglu, Nature-inspired computation and swarm intelligence, 2020). Poniższe wyniki bazują na minimalizacji często używanej funkcji Michalewicza danej wzorem:

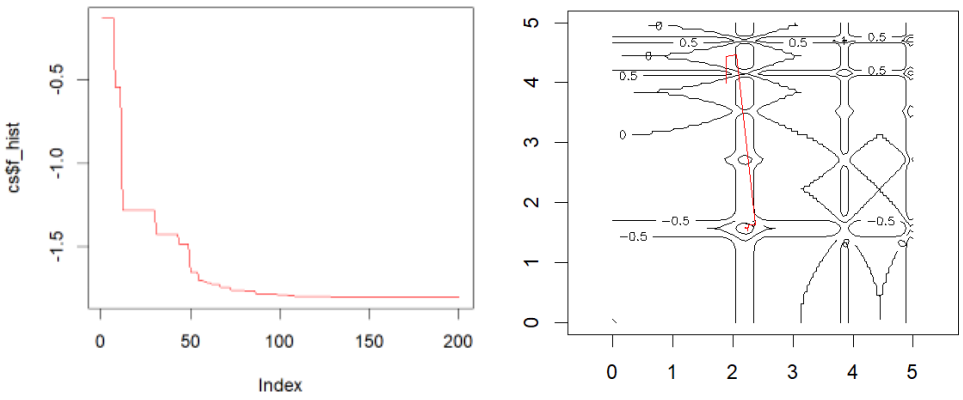


Rysunek 1 Funkcja Michalewicza

$$F(x) = \sum_{i=1}^{d} \sin(x_i) sin^{2m}\left(\frac{ix_i^2}{\pi}\right),$$

gdzie przyjęto że $m = 10$, argumenty funkcji celu to wektory dwuelementowe, a przestrzeń przeszukiwania jest dana jako $(x, y) \in [0,5] \times [0,5]$. Wiadomo, że dla takiego problemu minimum globalne $F_\star \approx -1.8013$ znajduje się w punkcie $(2.20319, 1.57049)$.

Wyniki otrzymane przy użyciu zaimplementowanego algorytmu w R dla $n = 5$, $p_a = \mathbf{0.25}$ i $iter = 200$ ilustrują poniższe wykresy. Pierwszy z nich pokazuje najniższe otrzymane wartości funkcji celu w kolejnych iteracjach $F(x_{best})$, a kolejny obrazuje przemieszczanie się pomiędzy odpowiadającymi tym wartością punktami po funkcji celu. Przyjęta liczba gniazd i iteracji jest niska, jako że obszar przeszukiwania nie jest duży.



Z wykresów wynika, że w pierwszych iteracjach uzyskana wartość funkcji celu znacznie maleje, podczas gdy w kolejnych krokach zmiany są mniejsze. Co więcej z drugiego wykresu można wywnioskować, że algorytm początkowo przemieścił optymalny punkt w stronę minimum lokalnego, jednak w kolejnych iteracjach skierował się w odpowiednią stronę trafiając w okolicę minimum globalnego. Finalne wyniki prezentują się poniżej i na ich podstawie można stwierdzić, że uzyskane wartości są bardzo bliskie znanemu minimum lokalnemu dla tego problemu. Algorytm spełnił swoje zadanie i znalazł punkt bardzo bliski minimum globalnemu już po niespełna 150 iteracjach, zatem może być on z powodzeniem stosowany do tego typu problemów.

```
> cs$x_opt
[1] 2.202901 1.570793
> cs$f_opt
[1] -1.801303
```

Podsumowanie

Algorytm kukułki to bardzo ciekawy koncept oparty na obserwacji natury umożliwiający globalną optymalizację. Parametr $\boldsymbol{p_a}$ sterując proporcją między lokalnością i globalnością przeszukiwań, usprawnia dojście do szukanego minimum lub maksimum. Dzięki użyciu lotów Levy'ego jako sposobu poruszania się kukułek algorytm zyskuje wiele na efektywności w stosunku do innych metod optymalizacyjnych. Przestrzeń rozwiązań jest przeszukiwana szybciej dzięki wariancji rozkładu Levy'ego pozwalającej na wykonanie większych kroków w krótszym czasie, co znacznie redukuje ilość potrzebnych iteracji. Dodatkowym atutem CS jest mała liczba parametrów wejściowych. Każdy dodatkowy parametr znacznie zwiększa czas trwania optymalizacji, dlatego też algorytmy SA i PSO okazały się być mniej efektywne i mniej skuteczne od CS podczas rozwiązywania wielu problemów optymalizacji.

Odwołania

Rosin, Z. i Tryjanowski, P. (2010). Tajemnice kukułczego gniazda. *Salamandra*.

Yang, X.-S. (2014). *Nature-Inspired Optimization Algorithms.* Elsevier.

Yang, X.-S. i Deb, S. (2009). Cuckoo Search via Levy Flights. *Conference: Nature & Biologically Inspired Computing*.

Yang, X.-S. i Karamanoglu, M. (2020). *Nature-inspired computation and swarm intelligence.* Academic Press.