Koncepcja Algorytmu Chwastowego

Mateusz Kasznia

2018-03-15

Abstract

W dokumencie znajduje się opracowanie na temat algorytmu chwastowego w kontekście jego zastosowania dla polioptymalizacji.

Contents

1 Wstęp

Algorytm IWO (*Invasive Weed Optimalization*) stanowi stosunkowo nową metodę rozwiązywania zadania optymalizacji, stworzoną na Uniwersytecie w Teheranie i wywodzącą się z algorytmów ewolucyjnych. Twórcy metody stosowali ją zarówno dla zadań optymalizacji ciągłej, jak i dyskretnej, które dotyczyły m.in. konfiguracji anten, obciążenia generatorów prądotwórczych, systemu rekomendacji na stronach www oraz równowagi Nasha w grach ekonomicznych.

2 Idea i opis działania algorytmu

Algorytm IWO jest metodą optymalizacji, w której technika penetracji przestrzeni poszukiwań, oparta na transformacji pojedynczego pełnego rozwiązania danego problemu w inne, została zainspirowana obserwacją cech charakterystycznych dla chwastów – ich gwałtownego rozprzestrzeniania się oraz szybkiego przystosowywania się do warunków otoczenia. Zarys algorytmu ma następującą postać:

Algorithm 1 Schemat algorytmu IWO

```
utworz pierwsza populacje osobnikow;
for all osobniki z populacji do
    oblicz wartosc funkcji przystosowania;
end for
while kryterium stopu nie jest spelnione do
    for all osobniki z populacji do
        wyznacz liczbe ziaren;
        for all ziarna do
            oblicz odleglosc miedzy upadkiem ziarna a jego osobnikiem macierzystym;
            na podstawie wartosci odleglosci skonstruuj nowego osobnika;
            oblicz wartosc funkcji przystosowania osobnika
        end for
    end for
    wybierz osobniki stanowiace nowa populacje;
end while
```

Należy podkreślić, że wartość funkcji przystosowania decyduje o liczbie ziaren rozsiewanych przez daną roślinę. Z kolei fragment pseudokodu mówiący o skonstruowaniu nowego osobnika wyznacza jedno z rozwiązań zadania optymalizacji. Upadek ziarna następuje w określonej odległości od rośliny macierzystej. Odległość ta definiuje nam również jak bardzo chwast "dziecko" różni się od chwasta "rodzica". Realizujemy to, uzależniając liczbę przeprowadzanych transformacji/mutacji jakie wykonujemy, żeby przejść od "rodzica" do "dziecka" właśnie od tej odległości. Sama odległość opisana jest przez rozkład normalny, gdzie odchylenie standardowe , obliczone dla iteracji algorytmu, obcięte jest do wartości nieujemnych. Łączna liczba iteracji, równoważna łącznej liczbie populacji chwastów, decyduje o kryterium stopu algorytmu. Odchylenie standardowe zmniejszane jest wraz z każdą kolejną iteracją zgodnie z następującą formułą:

$$\sigma_{iter} = \left(\frac{iter_{max} - iter}{iter_{max}}\right)^m (\sigma_{init} - \sigma_{fin}) + \sigma_{fin}$$

Symbole σ_{init} oraz σ_{fin} reprezentują, odpowiednio, początkową i końcową wartość odchylenia standardowego, natomiast m jest współczynnikiem modulacji nieliniowej. Ponieważ chwasty macierzyste mogą rozsiewać znaczne liczby nasion, zachodzi konieczność usunięcia chwastów najsłabiej przystosowanych do środowiska w liczbie zapewniającej stałą liczebność kolejnych populacji.

3 IWO krok po kroku

3.1 Incjalizacja początkowej populacji

Losowo, spośród możliwych rozwiązań, wybieramy te, które będą wstanowić populację początkową

3.2 Rozmnażanie

3.2.1 Liczenie przystosowania osobnika macierzystego

Obliczamy wartość każdego rozwiązania i w oparciu o nią, oraz najmniejszą i największą wartość rozwiązań w populacji, określamy ile "ziaren" może wytworzyć każde z rozwiązań. Im wyższa wartość danego rozwiązania tym więcej "dzieci" może ono mieć.

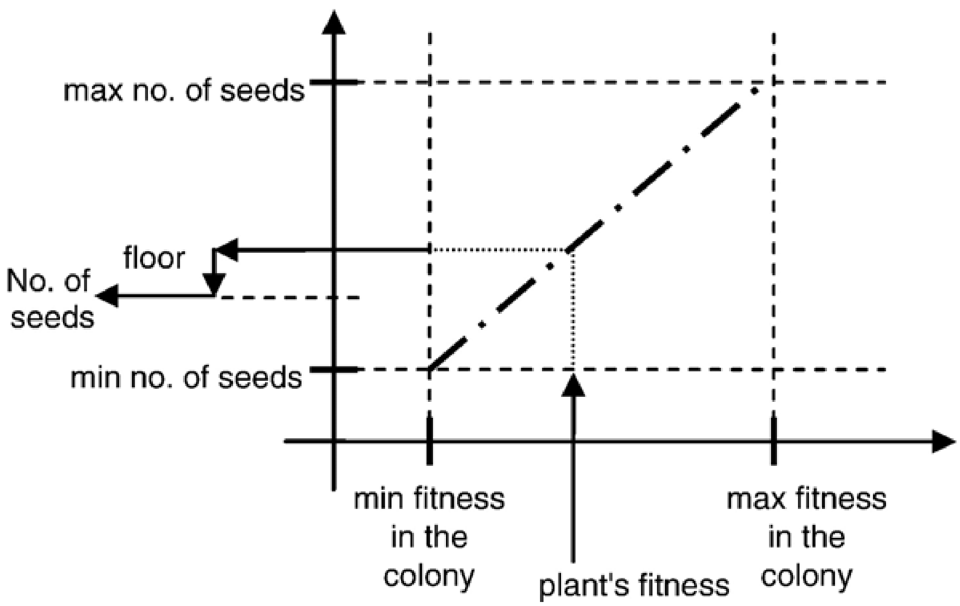


Figure 1: Seeds number

Formuła: $Seed(i) = \left(\frac{(Seed_{max} - Seed_{min})(Fit(i) - Fit_{min})}{Fit_{max} - Fit_{min}}\right)$

Gdzie $Seed_i$ to liczba ziaren jakie może wytworzyć i-ty osobnik, a Fit_i to wartość jego funkcji fitness.

3.2.2 Rozpraszanie potomków

Wygenerowane nasiona zostają losowo rozrzucone wokół swoich rodziców, jednak w każdej z iteracji, maleje odchylenie standardowe odległości od rośliny macierzystej, zgodnie ze wzorem z podpunktu *Idea i opis działania algorytmu* co prowadzi do powstawania skupisk roślin reprezentujących rozwiązania wysokiej o wartości.

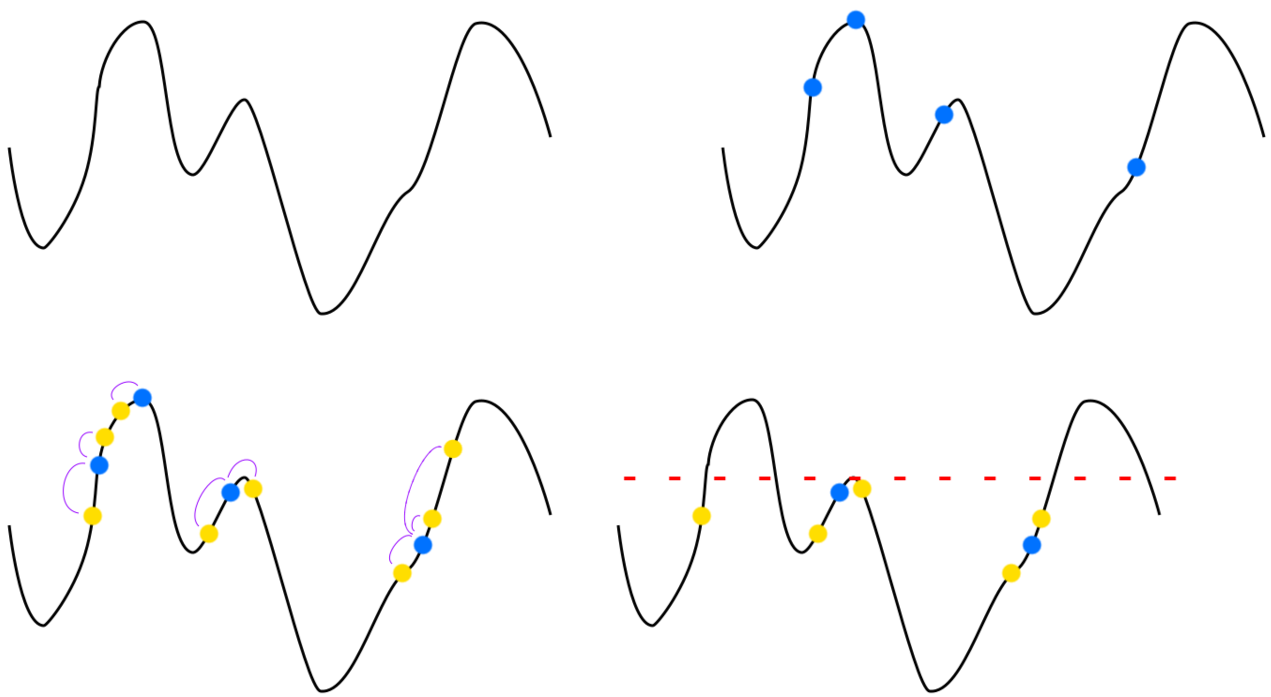
3.2.3 Tworzenie nowych rozwiązań

W tym etapie, w zależności od odległości jaką wyznaczyliśmy etap wcześniej, tworzymy osobniki potomne, przekształcając je z postaci "rodzica" do nowej postaci - "dziecka". Ilość wykonanych transformacji/mutacji zależy od tego, jak daleko ziarno danego osobnika upadło od rośliny macierzystej. Przykładowe sposoby mutacji opisane są w podrozdziale *Warianty*.

3.2.4 Eliminacja najsłabszych osobników

W momencie gdy uzyskamy maksymalną ilość roślin w kolonii, po wysianiu kolejnych następuje etap eliminacji, mający na celu usuwanie rozwiązań o niskiej wartości. Polega on na przeliczeniu funkcji wartościującej (fit) dla każdej rośliny i jej nasion z ostatniej iteracji (traktujemy je wszystkie jako jedną roślinę, dzięki czemu, jeśli słaby rodzic dał w swoim potomstwie dobre rozwiązanie, to ma szansę przetrwać) i odrzucaniu najsłabszych z nich tak długo, aż zrównamy liczbę chwastów z ich limitem.

3.2.5 Wizualizacja



- Najpierw wybierzmy sobię jakąś losową funkcję której minimum chcemy znaleźć.
- Następnie określmy kilka, losowych, początkowych rozwiązań (pierwszą populację).
- Teraz, w zależności od tego jak dobre jest dane rozwiązanie, pozwólmy mu wytworzyć określoną liczbę potomków i rozrzućmy ich w jego okolicy.
- Następnie (w przypadku gdy przekroczyliśmy limit populacji) usuńmy x najgorszych, tak aby zrównać ilość rozwiązań z limitem.

4 Warianty

W różnych wariantach tego algorytmu możemy używać różnych funkcji mutowania czy decydowania o sposobie “rozsiewu”. Klika przykładowych to:

- Inver (mutowanie): Mutacja ta dokonuje się w obrębie jednego osobnika. Przykładowo: mając osobnika z numerami miast przez które prowadzi trasa (np. (1,5,3,7,6,5)) i wykonując na nim Inver, wybieramy dwa losowe punkty, rozcinamy go w tych miejscach, odwracamy kolejność środkowego ciągu miast a następnie wstawiamy go z powrotem.

- Inver-over (mutowanie): Oparte na klasycznej inwersji, jednak o ile pierwszy punkt rozcięcia znajduje się w osobniku mutowanym, o tyle drugi może mieścić się w dowolnym osobniku należącym do populacji. Dzięki temu, mutacja ta ma charakter częściowo krzyżowy.

- Spreading (rozsiewanie wykorzystywane w *exIWO*): Miejsce wybieramy całkowicie losowo.
- Dispersing (rozsiewanie wykorzystywane w wersji podstawowej oraz w *exIWO*): Odległość od miejsca rozsiania jest skoligowana z tym jak bardzo potomek różni się od rodzica. Jest to standardowy sposób rozsiewania używany w IWO. Odległości te są losowane dookoła rodzica, tak aby ich zbiór "układał się" zgodnie z rozkładem normalnym.
- Rolling down (rozsiewanie wykorzystywane w *exIWO*): Miejsce wybierane jest, biorąc pod uwagę "wartość" jego sąsiadów. (Im bardziej wartościowi sąsiedzi, tym potencjalnie lepsze miejsce). Polega to na tym, że określamy m ruchów które ziarno może wykonać i k sąsiadów z którymi będzie sprawdzane. Następnie, dla danego rodzica (R) wyznaczamy losowo k rozwiązań (będących pojedynczą transformacją rozwiązania macierzystego) i wybieramy najlepsze z nich. Cały proces powtarzamy m razy (w kolejnych iteracjach rolę rodzica odgrywa najlepsze rozwiązanie wybrane w poprzedniej iteracji) i dopiero końcowy chwast zostaje potomkiem R.

5 Zastosowanie w polioptymalizacji (MOIWO)

Algorytm ten wychodzi z tych samych obserwacji natury co IWO i ma podobny przebieg. Różnica jest widoczna jedynie w sposobie wyznaczania funkcji fit (najlepszych osobników). W przypadku zagadnień polioptymalizacji, trudnością staje się zbalansowanie rozwiązania tak, aby dawało jak najlepsze wyniki dla każdego z optymalizowanych parametrów. Jednym z podejść, do tego zagadnienia, jest wyznaczenie funkcji wspólnej dla każdego z parametrów i dodanie do nich odpowiednich wag. Niestety, dobór wag jest bardzo trudny i niekiedy, nawet mała zmiana, ma olbrzymi wpływ na otrzymane rozwiązanie. Drugim podejściem, jest wyznaczenie zbioru rozwiązań *Pareto*. Zawiera on wszystkie zadowalające rozwiązania, które nie dominują siebie nawzajem. Jego wielkość jest zależna od ilości optymalizowanych parametrów. W MOIWO używamy do tego celu *fuzzy dominance based sorting*.

5.1 Fuzzy dominance based sorting

Pierwszym krokiem, tego podejścia, jest obliczenie *fuzzy dominance* dla każdego z rozwiązań, a następnie uporządkowanie ich rosnąco. *Fuzzy dominance* działa obustronnie i dlatego dla rozwiązań A i B, możemy policzyć stopień dominacji, zarówno A nad B jak i B nad A. Liczymy go następująco:

$$\mu_a(a,b) = \frac{\prod_i min(a_i,b_i))}{\prod_i a_i}$$

gdzie $\mu_a(a,b)$ oznacza stopień dominacji a nad b

$$\mu_p(a,b) = \frac{\prod_i min(a_i,b_i))}{\prod_i b_i}$$

gdzie $\mu_p(a,b)$ oznacza stopień dominacji b nad a

Fuzzy dominance sorting polega natomiast na policzeniu, dla każdego rozwiązania, maksymalnej wartości bycia zdominowynym przez inne, a następnie posortowania rozwiązań rosnąco po tym stopniu zdominowania. Dzięki czemu jako pierwsze uzyskamy rozwiązanie najsłabiej (lub w ogóle nie) zdominowane przez inne.

W przypadku rozwiązań znajdujących się na tym samym, niezdominowanym froncie, liczymy dla nich *crowding distance* i wybieramy to, z większą jego wartością. *Crowding distance* liczymy według następującego wzoru:

$$cd_k(x_{[i,k]}) = \frac{z_k(x_{[i+1,k]}) - z_k(x^k_{[i-1,k]})}{z_k^{max} - z_k^{min}}$$

A tak po ludzku: chodzi o to, aby wybrać rozwiązanie optymalne (niezdominowane przez inne) mające najwięcej przestrzeni dookoła siebie. Jest to ważne, ponieważ wyszukując rozwiązania w miejscach o ich małym zagęszczeniu, mamy dużo większą szansę na odkrycie kolejnych "pokrywających front Pareto", czyli będących rozwiązaniami optymalnymi, których jeszcze nie odnaleźliśmy.

5.2 Optymalizacja ze stałymi

Mamy z nią do czynienia, gdy chcemy aby pewne wartości składowych problemu które optymalizujemy pozostały niezmienne (lub w pewnych zakresach). Przykładem może być planowanie trasy naszego pojazdu pomiędzy miastami, z założeniem, że chcemy zobaczyć jak najwięcej, im więcej dróg będzie autostradami tym lepiej, a na paliwo chcemy wydać dokładnie 500 zł.

Najpowszechniejszym rozwiązaniem problemu polioptymalizacji ze stałymi, jest po prostu uwzględnienie różnicy wartości danej stałej w konkretnym rozwiązaniu i docelowej wartości tej stałej, podczas liczenia funkcji fit dla danej rośliny (rozwiązania).

6 Źródła

- *A novel numerical optimization algorithm inspired from weed colonization* A.R. Mehrabian, C.Lucas
- *Multi-objective optimization with artificial weed colonies* Debarati Kundu, Kaushik Suresh, Sayan Ghosh, Swagatam Das a, B.K. Panigrahi, Sanjoy Das
- *The Expanded Invasive Weed Optimization Metaheuristic for Solving Continuous and Discrete Optimization Problems* Henryk Josiński, Daniel Kostrzewa, Agnieszka Michalczuk, Adam Świtoński
- *Multi-Objective Optimization Using Genetic Algorithms: A Tutorial* Abdullah Konak, David W. Coit, Alice E. Smith
- *Fuzzy-Pareto-Dominance and its Application in Evolutionary Multi-Objective Optimization* Mario Koppen, Raul Vicente-Garcia, and Bertram Nickolay