Ewolucja Różnicowa

(Differential Evolution)

Wstęp do Ewolucji Różnicowej

Definicja: Ewolucja różnicowa (ang. Differential Evolution) to technika ewolucyjna opracowana przez K. Pricea i R. Storna w 1995 roku.

Pierwotne zastosowanie: Zadania o ciągłej dziedzinie funkcji przystosowania. Oznacza to, że ewolucja różnicowa została pierwotnie opracowana z myślą o rozwiązywaniu problemów w których funkcja przystosowania jest zdefiniowana dla ciągłych wartości zmiennych, a nie na przykład dyskretnych czy binarnych.

Zastosowania: Grupowanie, analiza obrazów, rozwiązywanie zadań programowania całkowitoliczbowego, uczenie sieci neuronowych.

Zasada działania Ewolucji Różnicowej

Opis:

- Ewolucja różnicowa opiera się na populacji osobników.

- Analogicznie do prostego algorytmu ewolucyjnego, populacja jest ulepszana w kolejnych iteracjach, dążąc do osiągnięcia optimum.

- Brak jednoznacznie określonego warunku stopu, najczęściej stosowanym warunkiem stopu jest osiągnięcie zadanej liczby iteracji.

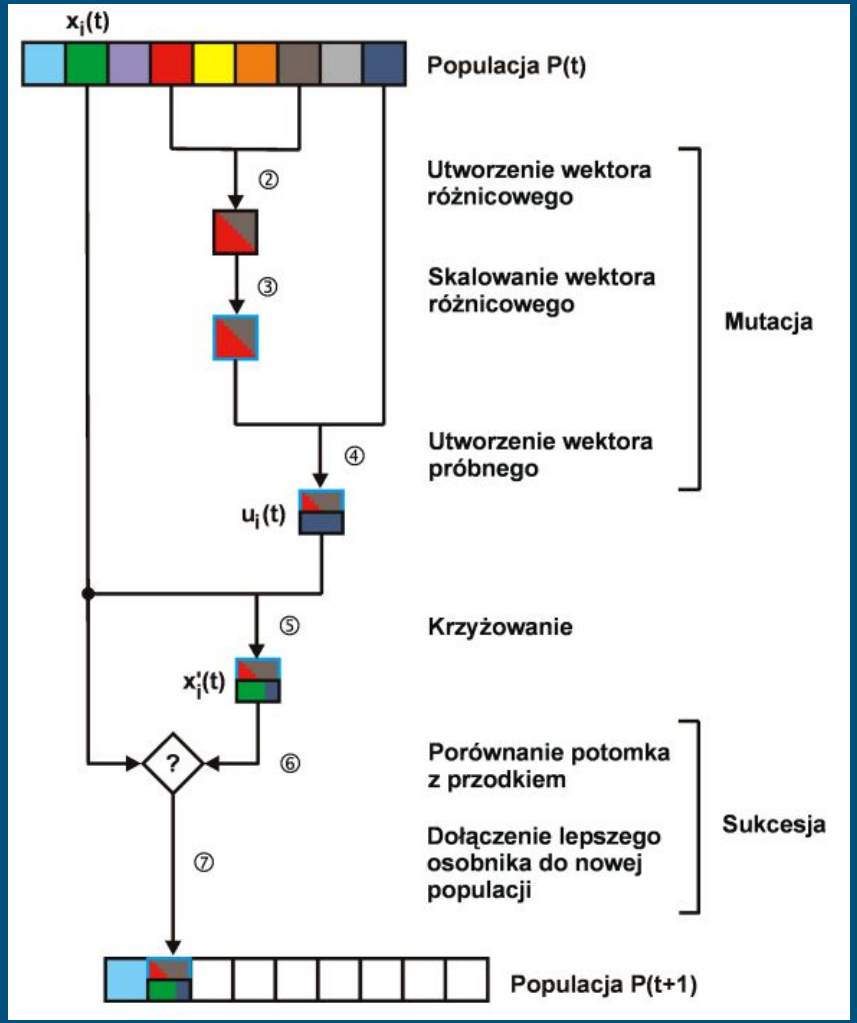
Różnice w stosunku do klasycznego algorytmu ewolucyjnego

- Brak selekcji.
  - W klasycznym algorytmie ewolucyjnym, po wykonaniu operacji mutacji i krzyżowania, dokonuje się selekcji, która polega na wyborze najlepszych osobników do utworzenia nowej populacji.
  - W Ewolucji Różnicowej nie stosuje się selekcji jako osobnego kroku algorytmu. Zamiast tego, decyzja o tym, czy dany osobnik powinien być częścią nowej populacji, podejmowana jest natychmiast po krzyżowaniu, na podstawie porównania przystosowania osobnika z przystosowaniem jego potomka.
- W Ewolucji Różnicowej mutacja jest stosowana przed, a nie po (jak w AE) procesem krzyżowania.
- W Ewolucji Różnicowej krzyżowanie generuje tylko jednego potomka.
  - W klasycznym algorytmie ewolucyjnym, zazwyczaj stosuje się różne metody krzyżowania, które mogą generować więcej niż jednego potomka na podstawie dwóch rodziców.
  - W Ewolucji Różnicowej zwykle stosuje się krzyżowanie, które generuje tylko jednego potomka na raz.
- Prosta metoda sukcesji: lepszy osobnik z pary przodek-potomek zostaje włączony do nowego pokolenia.

Ewolucja Różnicowa w formie pseudokodu

```
t = 0  // Inicjalizacja licznika iteracji

ustaw_parametry(F, CR)  // Ustawienie parametrów F i CR

utworz_poczatkowa_populacje(P(0), ns)  // Utworzenie początkowej populacji P(0) składającej się z ns osobników

while not warunek_stopu():  // Pętla wykonuje się, dopóki warunek stopu nie jest spełniony
    for kazdy osobnik xi(t) w populacji P(t):  // Dla każdego osobnika w populacji
        przystosowanie = oblicz_przystosowanie(xi(t))  // Obliczenie przystosowania osobnika xi(t)
        wektor_probny = mutacja(xi(t))  // Mutacja osobnika xi(t) w celu wygenerowania wektora próbnego
        potomek = krzyzowanie(xi(t), wektor_probny)  // Krzyżowanie osobnika xi(t) z wektorem próbnym w celu uzyskania potomka

        if przystosowanie_potomka(potomek) jest lepsze od przystosowanie:  // Jeśli przystosowanie potomka jest lepsze niż przystosowanie osobnika xi(t)
            dodaj_do_populacji(potomek, P(t + 1))  // Dodaj potomka do populacji P(t + 1)
        else:
            dodaj_do_populacji(xi(t), P(t + 1))  // W przeciwnym razie dodaj osobnika xi(t) do populacji P(t + 1)

    t = t + 1  // Zwiększenie licznika iteracji

wynik = znajdz_najlepszego_osobnika_w_populacji(P)  // Znalezienie najlepszego osobnika w populacji P
```

Parametr F: Określa, jak bardzo różnice między osobnikami w populacji będą powiększane lub zmniejszane podczas mutacji (zmienia wpływ wektora różnicowego na wektor próbny).
Parametr CR: Określa, w jakim stopniu komponenty wektora/osobnika rodzicielskiego będą zastępowane przez odpowiadające komponenty wektora próbnego podczas krzyżowania. Wartość z zakresu [0, 1].


$x_i(t)$
Populacja P(t)
②
Utworzenie wektora różnicowego
③
Skalowanie wektora różnicowego
Mutacja
④
Utworzenie wektora próbnego
$u_i(t)$
⑤
Krzyżowanie
$x'_i(t)$
?
⑥
Porównanie potomka z przodkiem
⑦
Dołączenie lepszego osobnika do nowej populacji
Sukcesja
Populacja P(t+1)

Przykład działania algorytmu ewolucji różnicowej

Załóżmy, że mamy do rozwiązania problem znalezienia minimum funkcji $f(x) = x^2$

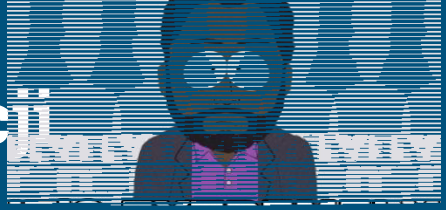
1. **Inicjalizacja populacji początkowej:** Losowo generujemy początkową populację osobników, gdzie każdy osobnik reprezentowany jest przez pewną wartość x (np. w zakresie [-10, 10]).
2. **Ewaluacja przystosowania:** Dla każdego osobnika w populacji obliczamy wartość funkcji przystosowania $f(x) = x^2$, gdzie x to wartość reprezentująca osobnika.
3. **Iteracyjne stosowanie operacji mutacji, krzyżowania i sukcesji:** <rozwinięte na następnym slajdzie>

Iteracyjne stosowanie operacji mutacji, krzyżowania i sukcesji

Dla każdego osobnika w populacji wykonujemy następujące kroki:

- **Mutacja:**
  - Losujemy trzech różnych osobników z populacji i na ich podstawie tworzymy wektor różnicowy (np. $x_2 - x_3$)
  - Wektor różnicowy jest modyfikowany przez zastosowanie skalaryzacji za pomocą parametru F
  - Następnie, na podstawie wektora różnicowego i losowo wybranego osobnika (np. x_1), generujemy wektor próbny ze wzoru $u = x_1 + F \cdot (x_2 - x_3)$
- **Krzyżowanie:**
  - Dokonujemy krzyżowania między osobnikiem a wektorem próbnym.
  - Krzyżowanie polega na łączeniu cech osobnika z cechami wektora próbnego w celu uzyskania potomka. Typowo stosuje się proste operacje jak np. mieszanie cech obu wektorów.
  - Krzyżowanie ma na celu zwiększenie różnorodności w populacji poprzez łączenie informacji genetycznej z dwóch rodziców.
- **Sukcesja:**
  - Porównujemy przystosowanie potomka z przystosowaniem rodzica.
  - Jeśli przystosowanie potomka jest lepsze niż przystosowanie rodzica, zastępujemy osobnika potomkiem w następnej populacji.
  - To podejście zapewnia ewolucję populacji w kierunku lepszych rozwiązań, podczas gdy jednocześnie zachowuje najlepsze osobniki znalezione dotychczas.

Ważne Informacje o algorytmie ewolucji różnicowej



- Na początku, odległości między sąsiednimi osobnikami są w przybliżeniu równe i relatywnie duże.
- Wraz z postępem poszukiwań, odległości między osobnikami zmniejszają się, co prowadzi do zbliżania się całej populacji do optimum. Ten proces nazywany jest zbieżnością algorytmu i reprezentuje przemieszczenie się populacji w kierunku optimum.
- Aby zapewnić równomierne rozmieszczenie osobników początkowych w przestrzeni poszukiwań, stosuje się losowanie wartości liczbowych dla każdego osobnika i dla każdego wymiaru przestrzeni poszukiwań.

Parametry algorytmu ewolucji różnicowej

1. **Rozmiar populacji (ns):**
   - Bezpośrednio wpływa na proces poszukiwań, eksplorując przestrzeń rozwiązań.
   - Im większa liczba osobników w populacji, tym więcej wektorów różnicowych można utworzyć, co zwiększa liczbę kierunków badań.
   - Przeprowadzone badania sugerują, że liczba osobników powinna być około dziesięciokrotnie większa od liczby wymiarów przestrzeni poszukiwań (ns ≈ 10·nx).
2. **Prawdopodobieństwo krzyżowania (CR):**
   - Decyduje o tym, ile elementów potomek odziedziczy od wektora różnicowego w procesie krzyżowania.
   - Wyższe wartości CR prowadzą do większego zróżnicowania nowej populacji, co sprzyja eksploracji.
   - Wartość parametru CR ma istotny wpływ na szybkość zbieżności algorytmu i jego odporność na utknięcie w lokalnych ekstremach.
3. **Czynnik skalujący (F):**
   - Reguluje wpływ wektora różnicowego na tworzenie wektora próbnego w procesie mutacji.
   - Małe wartości F sprzyjają szybszej zbieżności algorytmu, podczas gdy duże wartości ułatwiają eksplorację.
   - Dobór wartości F zależy również od rozmiaru populacji; im większa populacja, tym mniejsza wartość F, aby zachować odpowiednie eksploracyjne i intensywne działania.

Strategie Ewolucji Różnicowej

Algorytm ewolucji różnicowej (DE) może być modyfikowany poprzez zmianę trzech głównych elementów: metody wyboru wektora docelowego (x), liczby wykorzystanych wektorów różnicowych (y), oraz metody krzyżowania (z).

Strategie są oznaczane jako DE/x/y/z, gdzie symbole odnoszą się do tych trzech elementów algorytmu:

- x - metoda wyboru wektora docelowego
- y - liczba wykorzystanych wektorów różnicowych
- z - użyta metoda krzyżowania

Niektóre strategie DE wprowadzają samoadaptację, gdzie parametr γ (gamma) kontroluje wpływ najlepszego osobnika w populacji na generowanie nowych wektorów próbnych.

Badania wykazały, że różne strategie mają różne cechy. Na przykład, niektóre strategie cechują się lepszym rozproszeniem poszukiwań, podczas gdy inne zapewniają lepszą zbieżność algorytmu.

Strategie Ewolucji Różnicowej

DE/rand/1/bin:

- Wybór losowego wektora docelowego.
- Wykorzystanie jednego wektora różnicowego.
- Dobra równowaga między eksploracją a eksploatacją, co prowadzi do dobrego rozproszenia poszukiwań.

DE/best/1/z:

- Używanie najlepszego osobnika w populacji jako wektora docelowego.
- Wykorzystanie jednego wektora różnicowego.
- Bardzo dobra eksploatacja, co może przyspieszyć zbieżność algorytmu do optimum lokalnego.

DE/current-to-best/1+nv/z:

- Wykorzystanie co najmniej dwóch wektorów różnicowych, w tym osobnika najlepszego.
- Bardziej złożony schemat, który może zapewnić równowagę między eksploracją a eksploatacją, co może prowadzić do lepszej zbieżności algorytmu.

DE/rand-to-best/nv/z:

- Połączenie osobnika najlepszego i losowego jako wektora docelowego.
- Wykorzystanie nv wektorów różnicowych.
- Elastyczność między eksploracją a eksploatacją, kontrolowana przez parametr γ, co pozwala na dopasowanie strategii do konkretnego problemu optymalizacyjnego.
-

DE/x/nv/z:

- Wybór losowych wektorów docelowych.
- Użycie nv wektorów różnicowych, co pozwala na eksplorację w wielu kierunkach.
- Większa wartość nv umożliwia większą różnorodność i eksplorację przestrzeni poszukiwań.

Samoadaptacja

Samoadaptacja: Ewolucja różnicowa jest techniką samoadaptacyjną, która może samodzielnie dostosowywać swoje parametry w celu przejścia od szerokiej eksploracji przestrzeni poszukiwań do intensywnej eksploatacji.

Dynamiczne przełączanie strategii: Autorzy zaproponowali dynamiczne przełączanie się między strategią DE/rand/1/bin (rozproszenie poszukiwań) a strategią DE/current-to-best/2/bin (intensyfikacja poszukiwań) w zależności od aktualnego stanu populacji.

Dynamiczne zmiany parametrów F i CR: Istnieją różne podejścia do dynamicznej regulacji parametrów F i CR. Na przykład jedno z nich polega na liniowym zmniejszaniu CR do wartości od 1 do 0.7 i liniowym zwiększaniu F do wartości od 0.3 do 0.5 w trakcie iteracji algorytmu.

Samoadaptacyjne wyznaczanie F: Możliwe jest samoadaptacyjne określenie parametru F na podstawie przystosowania osobników w bieżącej populacji. Parametr ten jest obliczany na podstawie minimalnego i maksymalnego przystosowania w populacji.

Samoadaptacja prawdopodobieństwa CR: Prawdopodobieństwo rekombinacji CR może być również samoadaptacyjne. Niektórzy autorzy proponują próbkowanie CR z rozkładu normalnego, gdzie średnia jest obliczana na podstawie wartości CR, które poprawiają przystosowanie najlepszego osobnika.

Inne podejścia do samoadaptacji: Istnieją różne podejścia do samoadaptacji, takie jak losowanie parametrów F i CR za pomocą rozkładu normalnego lub wykorzystanie idei ewolucji różnicowej do samoadaptacji parametru F.

Plusy i minusy Ewolucji Różnicowej

Plusy dodatnie:

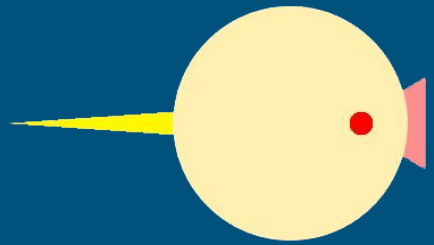
- **Skuteczność optymalizacji:** Ewolucja różnicowa jest skuteczną metodą optymalizacji, zwłaszcza w przypadku złożonych funkcji celu i problemów o dużej wymiarowości.
- **Samoadaptacja:** Możliwość samoadaptacji parametrów, takich jak współczynniki krzyżowania i skalowania, pozwala algorytmowi dostosować się do zmieniających się warunków i złożoności problemu.
- **Elastyczność:** Ewolucja różnicowa może być łatwo dostosowywana poprzez zmianę parametrów i strategii, co pozwala na lepsze dopasowanie do różnych rodzajów problemów.

Plusy ujemne:

- **Wymagany dobór parametrów:** Skuteczność ewolucji różnicowej jest często uzależniona od odpowiedniego dobrania parametrów, takich jak rozmiar populacji czy współczynniki krzyżowania i skalowania.
- **Wolniejsza zbieżność:** W porównaniu do niektórych zaawansowanych metod optymalizacji, ewolucja różnicowa może wymagać większej liczby iteracji, aby osiągnąć zbieżność do optymalnego rozwiązania.
- **Złożoność obliczeniowa:** Dla problemów o dużej wymiarowości lub złożonych funkcji celu, ewolucja różnicowa może być czasochłonna obliczeniowo, szczególnie jeśli liczba wymiarów przestrzeni poszukiwań jest duża.

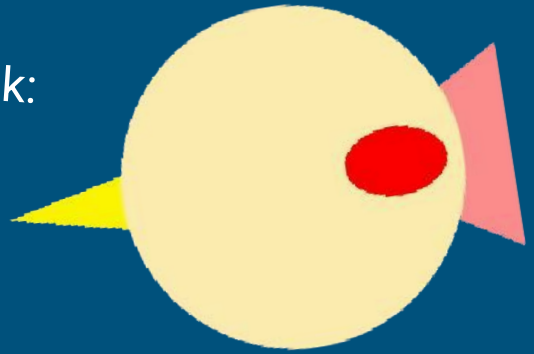
Przykład zastosowania algorytmów ewolucyjnych

To jest Adaś:

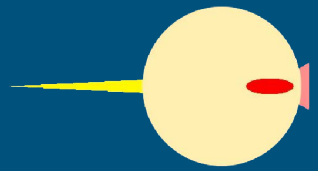


Adaś wraz z Bubkiem i Kokosem został stworzony niedawno jako populacja początkowa. Tak jak oni nie identyfikuje się on z żadną z płci.

Bubek:



Kokos:



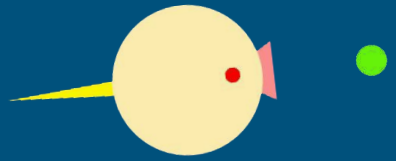
Przykład zastosowania algorytmów ewolucyjnych

To co definiuje Adasia, Bubka i Kokosa to ich genom. Genom przedstawiony w następujący sposób jako lista/zbiór/tablica poszczególnych genów:

int[] genome = {a, b, c, d, e, f, g};

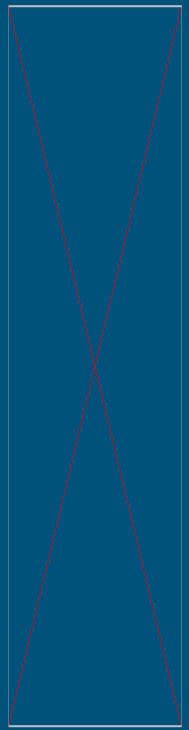
Każda liczba z listy odpowiada innej części stworka:
a - definiuje szybkość machania ogonem przez stworka
b - definiuje moc ogona stworka
c - definiuje wielkość stworka (a przez to także jego masę oraz ilość energii)
d - pole widzenia stworka
e - zasięg widzenia stworka
f - wielkość otworu gębowego stworka (im większy tym większa pożywienie może on pochłaniać - w tym inne mniejsze stworki)
g - minimalny % energii potrzebny do prokreacji

Przykład zastosowania algorytmów ewolucyjnych



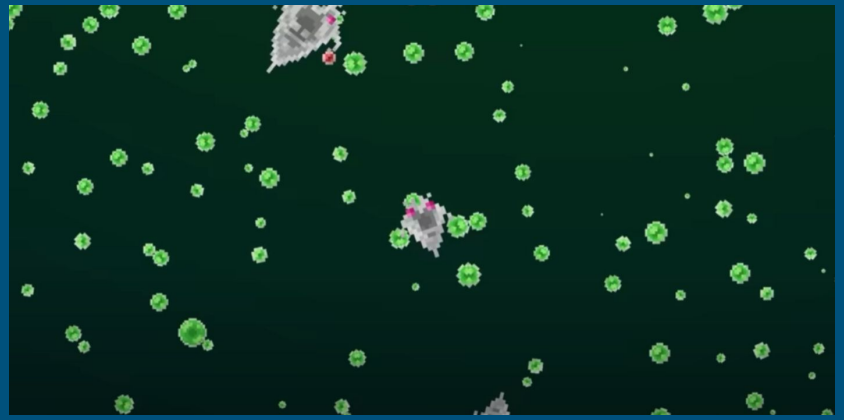
Zasady:

- Stworki poruszają się jedynie do przodu kierując się w kierunku najbliższego źródła pożywienia
- Stworki poruszają się w kierunku stworków które są w stanie zjeść i uciekają od tych które są w stanie zjeść je
- Jedzenie dodaje stworkom energię a ruch oraz rozmnażanie ją zabiera
- Gdy energia stworka spadnie do 0 stworek umiera
- Gdy stworek ma wymagany poziom energii oraz dotknie innego stworka dojdzie do stworzenia nowego stworka wybraną metodą ewolucyjną



Przykład zastosowania algorytmów ewolucyjnych

Niestety nie zdążyłem ukończyć programu na czas. Istnieje natomiast inny program, który oferuje podobne rozwiązania. Ewolucja odbywa się w nim jednak jedynie przez mutację. The Bibites: Digital Life.



https://www.youtube.com/watch?v=sEPh6bAQVP0

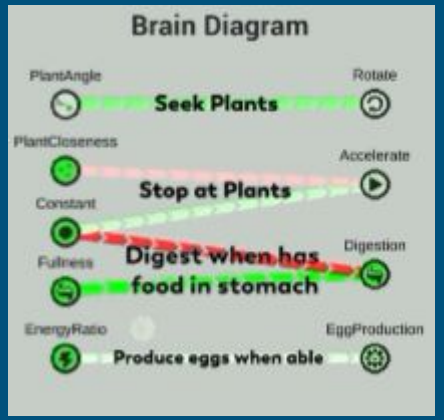


Diagram mózgu stworka który również chciałbym zaimplementować i oprzeć o genom.

Koniec