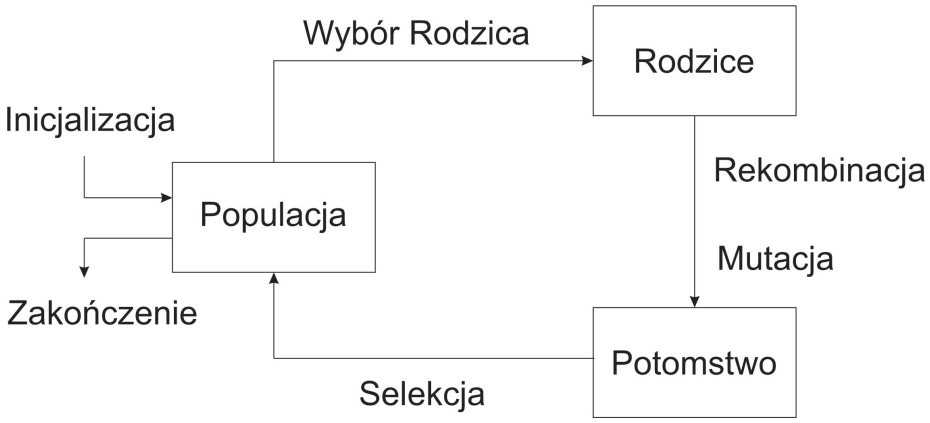
PSZT Zadanie 1

Zuzanna Santorowska & Artur Mazur

Listopad 2020

1 Temat

Modyfikujemy klasyczny algorytm ewolucyjny losowo łącząc osobniki w pary (aż do śmierci). Niech funkcja oceny (podczas szukania minimum) dla każdego z osobników w parze zwraca minimum z funkcji oceny pary: $q(i) = q(j) = min(q(i), q(j))$. Osobnik w klasycznym algorytmie ewolucyjnym będzie punktem (x, y) a w wersji zmodyfikowanej, będzie parą punktów (x_1, y_1) i (x_2, y_2).



Rysunek 1: Schemat Algorytmu Ewolucyjnego

2 Pytania

Jak porównać klasyczny algorytm i algorytm zmodyfikowany skoro osobnik ma inną definicję?
W przypadku mutacji nierównomiernej, jak można zdefiniować/obliczyć współczynnik b w funkcji Δ ?

3 Wybór rodzica

3.1 Proporcjonalna (ruletkowa)

Prawdopodobieństwo wyboru osobnika jest wprost proporcjonalne do jego wartości funkcji celu.

3.2 Turniejowy

Losowo łączymy osobniki w pary turniejowe. Następnie lepszy z osobników wygrywa i zostaje dopuszczony do rozmnażania. Tam metoda ma wiele zalet między innymi nie trzeba porządkować całej populacji ze względu na funkcję celu, co często jest kosztowane a w przypadku bardzo dużych zbiorów danych może nawet być niewykonywalne w sensownym czasie. Należy jedynie nie umieć porównać osobniki ze sobą.

4 Rekombinacja

4.1 Krzyżowanie jednopunktowe

Losowo wybieramy punkt przecięcia genotypu. Następnie powstaje dwójka potomków poprzez zwykłą wymianę części genotypów rodziców.

4.2 Krzyżowanie dwupunktowe

Wybieramy losowo 2 punkty przecięcia genotypu. Następnie dzieci powstają poprzez wymianę „środkowych" części genotypów rodziców.

4.3 Krzyżowanie uśredniające

W tym krzyżowaniu, każdy składnik genotypu potomka powstaje za pomocą wzoru:

$$y_j = x_j^1 \cdot w + x_j^2 \cdot (1 - w) \tag{1}$$

gdzie w jest losowo wybraną wartością z $U(0, 1)$. (1)

5 Mutacja

Mutacja nie jest koniecznym elementem algorytmu ewolucyjnego.
Dla tego przed rozpoczęciem działania algorytmu, ustalane będzie prawdopodobieństwo mutacji. W przypadku gdy uznamy że nie chcemy uwzględniać mutacji, jej prawdopodobieństwo wyniesie 0.

5.1 Mutacja równomierna

Ten rodzaj mutacji ma na celu zmianę konkretnego genu. Zmiana ta następuje na zasadzie:

$$x'_k = \langle L(k); P(k) \rangle \tag{2}$$

gdzie L(k) to lewy, a P(k) to prawy koniec przedziału możliwych wartości dla k-tego genu. Możemy te przedziały regulować parametrem n w pokazany sposób:

$$L(k) = (1 - n) \cdot x_k \tag{3}$$

$$P(k) = (1 + n) \cdot x_k \tag{4}$$

Zakładamy tutaj także że n będzie liczbą należącą do przedziału U(0,1).
Taki sposób wyznaczania przedziału gwarantowałby jego symetryczność- uzyskanie wartości mniejszych lub większych od początkowej byłoby równie prawdopodobne.

5.2 Mutacja nierównomierna (?)

W przeciwieństwie do zaproponowanej wyżej mutacji, ta będzie zależała od numeru populacji. Mutacja poszczególnego genu będzie zachodziła zgodnie ze wzorem:

$$X_k^{'} = \begin{cases} X_k + \Delta(t, P(k) - X_k) & \text{gdy wylosowano 0} \\ X_k - \Delta(t, X_k - L(k)) & \text{gdy wylosowano 1} \end{cases} \tag{5}$$

Zakładamy oczywiście że przed użyciem powyższego wzoru, będziemy losować liczbę ze zbioru 0,1. Natomiast funkcja Δ wyraża się wzorem:

$$\Delta(t, y) = y \cdot (1 - r)^{(1 - t/T) \cdot b} \tag{6}$$

gdzie r jest liczbą losową z przedziału [0,1], T oznacza maksymalny numer populacji a b to parametr systemu określający stopień niejednorodności.

5.3 Mutacja gaussowska

W przypadku tej mutacji, modyfikujemy gen tak aby przyjmował on wartość losową z rozkładem Gaussa o wartości oczekiwanej równej wartości przed mutacją czyli przebiegałaby zgodnie ze wzorem:

$$X_k^{'} = X_k + N(0, \sigma) \tag{7}$$

Parametr σ może zostać ustalony przed rozpoczęciem działania algorytmu albo uzależniony od numeru pokolenia tak aby stopniowo się zmniejszał. Wtedy w k-tym pokoleniu parametr ten mógłby przyjmować wartość zgodną ze wzorem:

$$\sigma_k = \sigma \cdot \frac{T - k}{T} \tag{8}$$

gdzie T określa maksymalną wartość pokolenia. Zgodnie z tym wzorem dla ostatniego pokolenia wartość σ wynosiłaby 0, co nie ma wpływu dla naszego algorytmu - ostatnia populacja nie będzie podlegała mutacji ani rekombinacji.

6 Selekcja

Tak jak w przypadku doboru rodzica również tu możemy stosować strategię ruletkową czy turniejową. Tutaj warto jednak wspomnieć o możliwości worzenia elity. Gdy szkoda nam odrzucać najlepsze jednostki ze względu na możliwość zgubienia bardzo dobrego rozwiązania możemy Wprowadzić elitę. Pewna liczba najlepszych osobników jest przekazywana do następnego pokolenia. Tym sposobem zmniejszamy nieco właściwości eksploracyjne algorytmu a zwiększamy te eksploatacyjne.

7 Testy

Poniżej zamieszczam w tabelach rodzaje poszczególnych operatorów dla danych testów. Notację zapożyczyłam z książki [1]. Zaimplementowaliśmy 3 funkcje oceny: Prostą wielowymiarową funkcję kwadratową(chcieliśmy ocenić czy algorytm zachowuje się zgodnie z oczekiwaniami), funkcję Weierstrass'a oraz rozszerzoną funkcję Schaffers'a. Przeprowadziliśmy testy dla wersji zmodyfikowanej oraz niezmodyfikowanej algorytmu ewolucyjnego.

7.1 Prawdopodobieństwo Mutacji

Test miał służyć zbadaniu jak parametr prawdopodobieństwa mutacji wpływa na szybkość zbiegania algorytmu. Przetestowałam szybkość zbiegania dla prawdopodobieństw 0.1, 0.5 i 1.0 dla wszystkich trzech funkcji oceny.

Operator	Rodzaj	Parametry
Reprezentacja	Zmiennoprzecinkowa	-
Rekombinacja	Jednopunktowa	$p = 0.85$
Mutacja	Gaussowska	$\sigma = 1.0$ $p =?$
Wybór rodzica	Ruletkowy	-
Selekcja	Elitarna	-

Z powyższych rysunków możemy wnioskować że w przypadku funkcji Expanded przy większym procencie mutacji średnia populacji bardziej odbiega od minimum globalnego, co jest logiczne ponieważ osobniki są bardziej rozrzucane po przestrzeni w porównaniu do małego stopnia mutacji gdy osobniki są bardziej skupione w minimach ale powoduje to możliwość utknięcia w minimacj lokalnych. Wykres pokazujący najlepszych osobników pokazuje że im większe było prawdopodobieństwo mutacji tym algorytm szybciej zbiegał do minimum globalnego. Widać że przy małym prawdopodobieństwie mutacji algorytm zbiega ale bardzo powoli.

W przypadku funkcji Weierstrassa wyniki pokazują że gdy prawdopodobieństwo mutacji jest za wysokie (1) algorytm gorzej zbiega do minimum jednak w przypadku prawdopodobieństw 0.5 i 0.1 nie widać wielkich różnic.

8 Conclusion

"I always thought something was fundamentally wrong with the universe"

Literatura

[1] J.E.Smith A.E.Eiben. *Introduction to Evolutionary Computing.* Springer, 1995.