|  |  |  |
| --- | --- | --- |
| Algorytmy ewolucyjne | | |
| Laboratorium nr 4 | | |
| Imię i nazwisko:  Jakub Dąbrowski | Prowadzący:  dr hab. inż. Iwona Karcz-Dulęba | Termin zajęć:  środa 15:15 – 16:45 TP |

1. Przebieg laboratorium

Laboratorium polegało na stworzeniu kolejnego algorytmu genetycznego wykorzystując sposób podzielenia długości łańcucha binarnego l na podłańcuchy oraz dwuwymiarową funkcję przystosowania.

Na początku działania algorytmu utworzono nową populację. Po podzieleniu wektora binarnego osobnika każdy z podłańcuchów należało zdekodować, gdzie użyto dekodowania z laboratorium 3 – wyliczenie wartości dziesiętnej z wartości binarnej. Do wyliczenia funkcji przystosowania wykorzystaną przykładową funkcję.

Wewnątrz procedury skorzystano z selekcji turniejowej, gdzie losowano t rodziców z każdego wiersza zainicjowanej populacji. Następnie na daną populację działało krzyżowanie jednopunktowe oraz mutacja z odpowiednimi współczynnikami prawdopodobieństwa. Po wykonanych działach następowało wyliczenie nowej funkcji przystosowania oraz obliczenie rezultatów uśrednionych oraz maksymalnych. Algorytm zależał od przebiegów pętli głównej, czyli od liczby generacji lg.

% PROCEDURA GA

P = population(m, l); % m wierszy, l kolumn ->INICJALIZACJA POPULACJI

f = evaluate(P, n); % P populacja, n ilość podłańcuchów ->FUNCKJA PRZYSTOSOWANIA

while i < lg

i = i + 1;

P1 = parent\_selection(P, f, m, t); % SELEKCJA RODZICÓW

P2 = recombine(P1, m, l, pc); % KRZYŻOWANIE

mutation(P2, m, l, pm); % MUTACJA

P = P2;

f = evaluate(P,n); % FUNCKJA PRZYSTOSOWANIA NOWEJ POPULACJI

f\_max(i, 1) = max(f); % WEKTOR F-MAX

f\_sr(i, 1) = mean(f); % WEKTOR F-ŚR

end

Fragment kodu przedstawiający implementację procedury algorytmu genetycznego.

% Podział podłańcuchów i wyliczenie funkcji przystosowania z wykorzystaniem dekodowania bin na dec

function f = evaluate(P, n)

m = size(P,1);

l = size(P,2);

for i=1:m

X{i,1} = P(i,1:10); % pierwszy podział

X{i,2} = P(i,11:20); % drugi podział

x1 = evaluate\_bd(X{i,1});

x2 = evaluate\_bd(X{i,2});

f(i,1) = (25-(x1-5)^2)\*cos(2\*x1-5)+(25-(x2-5)^2)\*cos(2\*(x2-5))+50; % dwuwymiarowa funkcja przystosowania

end

end

Fragment kodu przedstawiający implementację wyliczenie dwuwymiarowej funkcji przystosowania.

1. Badania symulacyjne algorytmu

Badania symulacyjne przeprowadzono przy użyciu selekcji turniejowej (w podpunktach 2.1 oraz 2.2 liczba losowanych rodziców t = 2). Warunkiem zatrzymania działania algorytmu jest liczba generacji lg = 15, a długość łańcucha binarnego równała się l = 20. Poniżej zostały podane parametry wejściowe, którymi manipulowano, aby uzyskać satysfakcjonujące wyniki.

% Parametry wejściowe – zmieniane podczas badań symulacyjnych

m=40; % rozmiar populacji m -> liczba parzysta

l=20; % długość łańcucha binarnego chromosomu l

lg=15; % liczba generacji lg, przebiegów pętli głównej

n=2; % ilość podłańcuchów

li=1:n; % wektor podziału podłańcuchów

pc=0.7; % prawdopodobieństwo krzyżowania pc

pm=0.12; % prawdopodobieństwo mutacji pm

t=2; % liczba rodziców

i=0; % początek pętli

Fragment kodu przedstawiający bazowe parametry wejściowe.

* 1. Wpływ parametrów na działanie algorytmu

1. Różna wielkość populacji

* m = 10



* m = 20



* m = 40



Spoglądając na otrzymane wyniki uśrednionej funkcji przystosowania widać, że dla populacji m = 40 wartości następnych generacji uzyskiwały wynik lepszy niż dla populacji mniejszej.

1. Różne prawdopodobieństwa krzyżowania

Wykresy w tym podpunkcie wygenerowano dla populacji m = 40 oraz prawdopodobieństwa mutacji pm = 0.12.

* pc = 0.25



* pc = 0.5



* pc = 0.75



Najkorzystniej prezentuje się wynik uzyskany przy prawdopodobieństwie krzyżowania równym pc = 0.25. Uśredniona funkcja przystosowania w kolejnych generacjach uzyskuje szybciej wyższą wartość.

1. Różne prawdopodobieństwa mutacji pojedynczego bitu

Wyniki w tym podpunkcie uzyskano dla populacji m = 40 oraz prawdopodobieństwa krzyżowania pc = 0.25

* pm = 0.01



* pm = 0.02



* pm = 0.05



Uśredniając wyniki i spoglądając na uśrednione wykresy zaobserwowano, że algorytm dobrze prezentował się dla pm = 0.02.

* 1. Przebieg ewolucji

W tym zadaniu należy zbadać przebieg ewolucji dla najlepszych uzyskane parametry uzyskane poprzednio:

* wielkość populacji m = 40
* prawdopodobieństwo krzyżowania pc = 0.25
* prawdopodobieństwo mutacji pojedynczego bitu pm = 0.02

Algorytmy ewolucyjne zależą od populacji początkowej (losowość). Z racji tego obliczenia zostały przeprowadzone dla takiego samej populacji początkowej o rozmiarze 40 x 20 (m x l).

1. wyłączona mutacja



1. wyłączone krzyżowanie



1. wyłączona mutacja i krzyżowanie



1. Jednorodnej populacji początkowej

* same zera (P = zeros(m,l))



* same jedynki (P = ones(m,l))



* 1. Wpływ na efektywność rozmiaru turnieju dla selekcji turniejowej

W tym zadaniu przyjęto miarę efektywności algorytmu jako najlepsze rozwiązanie znalezione przy zadanej liczbie generacji (lg = 15). Badanie przeprowadzono dla 2 rozmiaru turnieju (t = 2, 3) w selekcji turniejowej oraz dla 3 wartości rozmiaru populacji (m = 10, 20, 40).

* t = 2



* t = 3



* Tabelka porównująca maksymalne wartości populacji (f\_max)

|  |  |  |
| --- | --- | --- |
|  | t = 2 | t = 3 |
| m = 10 | 84.38 | 88.23 |
| m = 20 | 88.18 | 93.61 |
| m = 40 | 99.42 | 99.59 |

Analizując otrzymane wykresy oraz spoglądając na zestawienie maksymalnych wartości populacji można stwierdzić, że optymalnym współczynnikiem w selekcji turniejowej jest t = 3 (liczba losowanych rodziców). Dla największego rozmiaru populacji m = 40 otrzymany f\_max nie różnił się znacznie dla obu współczynników. Różnice natomiast widać przy rozmiarze populacji m = 10 i m = 20, gdzie wyraźnie uzyskują przewagę wyniki f\_max dla współczynnika t = 3.