Algorytmy Genetyczne - projekt

Marcin Fabrykowski

14 lipca 2012

Spis treści

1 Opis problemu

Mamy dany zbiór N klocków o zadanych wymiarach W i H. Naszym zadaniem jest ułożenie wieży z posiadanych klocków. Zależy nam, aby wieża była jak najwyższa oraz jak najszersza.

Klocki możemy układać tylko z w płaszczyźnie XY, oraz możemy je obracać wokół osi OZ.

2 Opis rozwiązania problemu dwukryterialności

W moim projekcie postanowiłem położyć większy nacisk na wysokość wieży, aniżeli na jej szerokość. Za każdy postawiony klocek, osobnik dostaje 1000pkt, natomiast punkty za szerokość są przyznawane w wartości równej tej szerokości.

Powoduje to, że punktacja za wysokość jest decydująca, a następnie z najwyższych rozwiązań wybieramy najszersze.
Punkty za szerokość przyznawane są tylko dla pierwszych 10 klocków.

3 Szczegółowy opis kodowania

Osobnik kodowany jest jako tablica numerów klolejno kładzionych klocków. Klocek definiowany jest za pomocą następujących parametrów:

1. Szerokosc: float itsW
2. Wysokosc: float itsH
3. Obrot: int itsObrot
4. Przesunięcie: float itsX

Przesunięcie klocka kodowane jest w zakresie $< 0, 1 >$, co jest rozumiane jako procent długości klocka spodniego nad którym znajduje się środek ciężkości klocka nadrzędnego.
Dla przykładu: dla wartości 0, środek ciężkości klocka kładzionego, znajduje się nad lewą krawędzią klocka dolnego. Dla wartości 0.5, klocek górny ma środek ciężkości nad środkiem ciężkości klocka dolnego

Przeliczanie wartości kodowanej przesunięcia na wymiar przesunięcia względnego odbywa się przy użyciu wzoru:

$$x = w_1 * dx - \frac{w_2}{2}$$

gdzie: x - pozycja klocka górnego zadanym przez temat formacie
w_1 - szerokość klocka dolnego
w_2 - szerokość klocka górnego
dx - wartość procentowego przesunięcia odczytywana z osobnika

4 Szczegółowy opis inicjacji populacji bazowej

Tworzenie populacji bazowej przebiega w następujący sposób:

1. Tworzymy osobnika układającego wszystkie klocki w kolejności zadanej z danych wejściowych

2. Zamieniamy miejscami w kolejności dwa losowe klocki. Zamian takich wykonujemy w ilości odpowiadającej rozmiarowi osobnika.

3. Dla każdego klocka losujemy czy będzie on obrócony oraz procent przesunięcia.

5 Opis użytej funkcji dostosowania

W funkcji dostosowania sprawdzamy od klocka $i = 1$ do $i = popsize$ czy wieża jest stabilna.
Zakładamy, że wartość 0 na osi OX znajduje się na wysokości lewego boku kładzionego klocka. Wtedy zauważamy, że środek jego ciężkości znajduje się w $\frac{w_2}{2}$.
Przyjmujemy oznaczenia:
w_1 - szerokość klocka dolnego
w_2 - szerokość klocka górnego
dx - przesunięcie klocka górnego odczytane z osobnika
k - pozycja klocka dolnego względem punktu 0
l - pozycja klocka górnego względem punktu 0
następnie dokładamy klocek od dołu i sprawdzamy czy taki układ jest stabilny, tj. środek ciężkości wyższego znajduje się nad podstawą dolnego. Jeżeli tak, to obliczamy środek ciężkości układu dwóch klocków i dokładamy kolejny klocek od dołu. Jeśli środek ciężkości znajduje się nad podstawą dołożonego klocka, uznajemy to za stabilne. Postępujemy tak, aż dojdziemy do podstawy. Uznajemy wtedy, że wieża do badanego klocka jest stabilna.
Powtarzamy wtedy postępowanie biorąc jako pierwszy i+1-wszy klocek.

W przypadku nie powodzenia, przerywamy procedurę i przechodzimy do obliczania szerokości.

Szerokość obliczana jest według algorytmu:

1. przyjmujemy wartość min jako lewy bok podstawy
2. przyjmujemy wartość max jako prawy bok podstawy
3. przechodzimy do następnego klocka
4. sprawdzamy czy pozycja lewej krawędzi jest mniejsza niż min. Jeśli tak to min=pozycja lewego boku
5. jeśli pozycja prawego boku jest większa niż max, to max=pozycja prawego boku
6. powtarzamy dla 10 klocków licząc od podstawy

do wartości dostosowania dodajemy szerokość wierzy wyrażoną jako $max - min$.

6 Opis użytych metod krzyżowania, mutacji oraz selekcji

6.1 Metody krzyżowania

Zakładamy przyjęcie następujących stałych:

1. wielkość populacji: 500
2. liczba generacji: 500
3. metoda selekcji: RankSelector
4. prawdopodobieństwo mutacji: 0.01
5. prawdopodobieństwo krzyżowania: 0.8

Badanie metod krzyżowania przedstawia poniższa tabela

	OrderCrossover	CycleCrossover	PartialMatchCrossover
czas	20s	–	21s
klockow	230	–	238
szerokosc	29	–	35

Powyższe zestawienie skłania mnie do wybrania metody PartialMatchCrossover, ponieważ metoda daje lepsze wyniki przy nieznacznie dłuższym czasie wykonywania.

6.2 Metody mutacji

Wykorzystałem własną funkcję mutacji.
Testowane były 4 różne metody mutacji:

1. zamieniająca losowe dwa klocki
2. zamieniająca losowe dwa klocki oraz zmiana obrotu klocka
3. zamieniająca losowe dwa klocki oraz zmiana obrotu klocka oraz zmiana przesunięcia klocka
4. zamieniająca losowe dwa klocki oraz zmiana przesunięcia klocka

Testy wykazały, że wykorzystanie funkcji zmieniających obrót i/lub przesunięcie powodowało drastyczny spadek ilości ułożonych klocków do wartości rzędu 5-30 klocków, dlatego wybrana została funkcja zmieniająca jedynie kolejność klocków.

6.3 Metody selekcji

Zakładamy przyjęcie następujących stałych:

1. wielkość populacji: 500
2. liczba generacji: 500
3. metoda krzyżowania: PartialMatchCrossover
4. prawdopodobieństwo mutacji: 0.01
5. prawdopodobieństwo krzyżowania: 0.8

Testy wykazały następujące wyniki

	RankSelector	TournamentSelector	RouletteWheelSelector
czas	23s	6s	6s
klockow	252	7	36
szerokosc	34	31	24

Zauważamy, że metoda rankingowa daje znacząco lepsze efekty, dlatego zdecydowałem się na tę właśnie metodę.

7 Podsumowanie wybranych parametrół

Ostatecznie zostały wybrane następujące parametry algrorytmu:

- Metoda krzyżowania: PartialMatchCrossover
  Uzasadnienie w odpowiedniej sekcji
- Metoda selekcji: RankSelector
  Uzasadnienie w odpowiedniej sekcji
- Metoda mutacji: własna metoda
  Uzasadnienie w odpowiedniej sekcji
- Wielkość populacji: 1000
  Testy wykazały poprawę wyników przy zwiększeniu populacji do 1000 osobników
- Ilość pokoleń: 1000
  Testy wykazały poprawę wyników przy zwiększeniu liczby pokoleń do 1000
- Prawdopodobieństwo mutacji: 0.01
  Większe wartości powodują znaczne pogorszenie wyników
- Prawdopodobieństwo krzyżowania 0.8
  Wartości oscylujące około 0,8 - 0.9 nie powodują znacznych zmian w wynikach