**PSZT Dokumentacja do projektu**

**Paweł Martyniuk, Iwo Sokal**

**Treść zadania**

MM.AG3 Podział kart na 2 stosy Masz N kart z wartościami od 1 do N. Przy pomocy algorytmu genetycznego należy podzielić je na dwa stosy, gdzie suma wartości kart na pierwszym stosie ma wartość jak najbliższą do A, a iloczyn wartości kart na drugim stosie jak najbliższa wartości B. WE: liczba kart N, suma kart A, iloczyn kart B, satysfakcjonujący poziom dopasowania w %. WY: podział kart na stosy z wynikami działań.

**Założenia:**

Program został napisany w języku C++. Mamy N kart z wartościami od 1 do N. Każda karta musi być przypisana do jednego ze stosów kart. Zastosowane zostało kodowanie binarne: jeżeli karta ma wartość 0 to należy do zbiory A, natomiast w przeciwnym przypadku do zbioru B. Celem algorytmu jest przypisanie tak kart do obu grup, aby suma w zbiorze A oraz iloczyn kart w zbiorze B były najbliższe wartościom nadanym z satysfakcjonującym poziomem dopasowania. Przyjęliśmy, że dla najlepszych osobników wartość funkcji przystosowania zbiega do 0. Im wartość jest dalej od zera tym osobnik jest gorzej przystosowany. Warunkiem zatrzymania algorytmu uznaliśmy za osiągnięcie satysfakcjonującego poziomu dopasowania albo wykonanie 1000 iteracji algorytmu.

**Podział zadań:**

Iwo Sokal:

* Inicjalizacja pierwszego pokolenia
* Ocena Osobników
* Selekcja osobników

Paweł Martyniuk:

* Krzyżowanie osobników
* Mutacja
* Testowanie i tworzenie statystyk

**Sposób realizacji**

W celu znalezienia najefektywniejszego rozwiązania problemu zdecydowaliśmy się zaimplementować kilka różnych metod dla niektórych funkcji.

Zaimplementowane metody:

1. Pierwsze pokolenie - wypełnienie losowymi osobnikami
2. Krzyżowanie
3. Jednopunktowe z tym samym punktem krzyżowania dla wszystkich osobników
4. Jednopunktowe z losowym punktem krzyżowania dla każdej krzyżowanej pary
5. Dwupunktowe z tymi samymi punktami krzyżowania dla wszystkich osobników
6. Dwupunktowe z losowymi punktami krzyżowania dla każdej krzyżowanej pary
7. Mutacja - dla danego osobnika zmieniany jest jeden bit na przeciwny. Przy starcie jako jeden z argumentów przekazywane jest prawdopodobieństwo bazowe mutacji. Jest to mutacja dla karty o wartości 1. Każda następna karta ma prawdopodobieństwo mutacji o 20% mniejsze od poprzedniej. Ma to za zadanie, aby karty o wyższych wartościach rzadziej mutowały od tych o wartościach mniejszych. Gdybyśmy często zmieniali karty o dużych wartościach wtedy wartość funkcji dopasowania by się zmieniała skokowo, a dzięki temu odbywa się to łagodnie.
8. Ocena przystosowania: Max (dopasowanie A, dopasowanie B)

dopasowanie A: |aspiracjaA - sumaA|/ aspiracjaA

dopasowanie B: |aspiracjaB - iloczynB|/ aspiracjaB

1. Selekcja:
2. Losowa - spośród populacji rodziców wybieramy N losowych osobników.
3. Turniejowa - spośród populacji losujemy dwóch osobników i wybieramy spośród nich tego który ma lepszą ocenę przystosowania.

**Testowanie:**

Dane dobrane do tej części testów: A = 19400, B = 6.89557e+197, poziom dopasowania = 0.3, bazowe prawdopodobieństwo mutacji = 50, prawdopodobieństwo krzyżowania = 80, liczba osobników = 100, liczba kart = 200.

Wykresy zależności najlepszego osobnika danej populacji od generacji

Selekcja losowa, krzyżowanie jednopunkowe Selekcja turniejowa, krzyżowanie jednopunkowe

Selekcja turniejowa, krzyżowanie dwupunktowe (80, 150) Selekcja turniejowa, krzyżowanie dwupunktowe losowe

Czas pracy programu w zależności od N kart i K osobników:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N K** | 10 | 100 | 200 | 400 |
| 10 | 0.076 | 0.548 | 1.117 | 2.222 |
| 50 | 0.150 | 1.128 | 2.185 | 4.432 |
| 100 | 0.180 | 1.390 | 2.709 | 5.349 |
| 200 | 0.326 | 2.517 | 4.900 | 9.560 |

**Wnioski, przemyślenia:**

Bez porównania selekcja turniejowa daje lepsze wyniki od losowej, natomiast krzyżowanie dwupunkowe pozwala lepiej znajdywać bardziej zróżnicowane wyniki.

W celu usprawnienia działania algorytmu można by było dodać sukcesje generyczną, czyli losować n elementów z n-elementowej populacji, posortować względem jakości dopasowania i wybrać tylko połowę najlepszych z nich. Dzięki temu mielibyśmy pewność że każda kolejna generacja będzie nie gorsza od poprzedniej.

Obserwując zjawisko że krzyżowanie dwupunktowe jest lepsze od jednopunktowego można by zaimplementować krzyżowanie trójpunktowe, być może by się okazało jeszcze lepsze.