**PROJEKTOWANIE EFEKTYWNYCH ALGORYTMÓW**

PROJEKT

11/12/2021

252736 Hutnik Szymon

Genetyczny (6)

|  |  |
| --- | --- |
| Strona | Spis treści |
| 2 | Treść zadania |
| 3 | Opis metody |
| 4 | Opis algorytmu |
| 6 | Dane testowe |
| 7 | Procedura badawcza |
| 9 | Wyniki |
| 11 | Analiza |

1. Treść zadania

Opracować, napisać, zbadać rozwiązanie problemu komiwojażera w wersji optymalizacyjnej algorytmem genetycznym.

Problem komiwojażera (eng. *Travelling Salesman Problem*) polega na znalezieniu minimalnego cyklu Hamiltona (przejście przez wszystkie wierzchołki tylko raz, startując i kończąc w tym samym punkcie) w pełnym grafie ważonym.

1. Opis metody

Algorytm genetyczny polega na stworzeniu populacji, gdzie każda jednostka posiada pewne cechy. Następnie w kolejnych iteracjach algorytmu symulowane jest krzyżowanie genetyczne między najlpiej przystosowanymi osobnikami (w przypadku TSP są to najniższe koszty przejścia ścieżki). Rodziców wybiera się jedną z metod selekcji, w przypadku tego programu jest to turniej. Następnie nowa jednostka może zostać zmutowana, w losowy sposób (w przypadku TSP losowa modyfikacja ścieżki, np. swap dwóch losowych wierzchołków). Kiedy wygenerowana zostanie nowa populacja, następuje połączenie jej z już istniejącą w wybrany przez nas sposób, np. usunięcie wszystkich starych jednostek.

1. Opis algorytmu

Rozwiązanie zaimplementowano w postaci programu opisanego przez poniższe diagramy:

Diagram

Description automatically generated

Rysunek 1: Ogólny diagram czynności programu

Najpierw inicjalizowane są zmienne. Po wczytaniu danych z konsoli następuje uruchomienie właściwej części algorytmu, następnie wypisywany jest wynik oraz czas wykonania właściwego algorytmu.

Diagram

Description automatically generated

Rysunek 2: Szczegółowy diagram czynności algorytmu

1. Dane testowe

Dane, na których była badana efektywność algorytmu pochodzą ze zbioru udostępnionego przez dr Rudego. Do badania użyto wartości z następujących plików:

* m15.txt, 259
* gr21.txt, 2707
* gr48.txt, 5046
* gr96.txt, 55209
* lin105.txt, 14379
* gr137.txt, 69853
* gr202.txt, 69853
* lin318.txt, 42029
* gr431.txt, 171414

http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/files/pea/instances.zip

1. Procedura badawcza

Należało zbadać poprawność znajdowanych rozwiązań w ciągu 10 minut.

Na początku metodą prób i błędów zostały dostosowane parametry algorytmu, następnie każdą instancję badaną wykonano 1 raz.

Pomiar czasu został wykonany przy użyciu bilbioteki chrono. Po otrzymaniu wyniku należy go podzielić przez liczbę powtórzeń wywołań algorytmu.

Obraz zawierający tekst

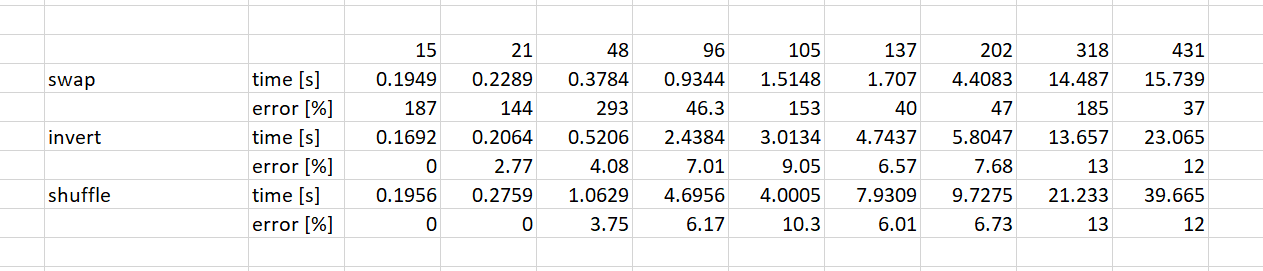
Opis wygenerowany automatycznie



1. Wyniki

Wyniki opracowano w programie Excel:

Rysunek 3: Czasy działania algorytmu dla *n* wierzchołków w zależności od metody- graf



Rysunek 4: Czasy działania i błąd procentowy algorytmu dla *n* wierzchołków - tabela

Rysunek 5: Błąd procentowy dla *n* wierzchołków w zależności od metody – graf

Text

Description automatically generated

Rysunek 6: Screenshot consoli z programem

\*dane testowe nie są kolejnymi wartościami n stąd połączenie znanych wyników linią

1. Analiza

Na podstawie przeprowadzonego badania można zauważyć, że metody invert oraz shuffle dają zdecydowanie lepsze rezultaty niż swap, w dalszej analizie będę się odwoływał do tych 2 metod. Krzywa wzrostu czasu względem wielkości instancji rośnie w przybliżeniu liniowo (Rysunek 3). Warto zwrócić uwagę, że złożoność pamięciowa w symulowanym wyżarzaniu jest pomijalna, zaś złożoność czasowa jest stosunkowo niewielka i można ją łatwo zmieniać ustawiając parametry algorytmu oraz wybierając inny algorytm chłodzenia oraz warunek stopu. SA należy do grupy algorytmów podających najlepszy osiągnięty wynik w zadanym czasie, zazwyczaj otrzymuje się błąd, który w przeprowadzonych przeze mnie badaniach kształtował się na poziomie 10% i powoli rósł dla większych instancji.

Porównanie dotychczas zrealizowanych algorytmów pokazuje jak dużą przewagę czasową ma szukanie przybliżonego rozwiązania zamiast dokładnego. Czas wykonania symulowanego wyżarzania jest bezkonkurencyjny w porównaniu z algorytmem Helda-Karpa. Takie niedokładne podejście pozwala na szukanie rozwiązań dla algorytmów o wysokiej złożoności, mimo błędu dostajemy przynajmniej przybliżony wynik.