Gliwice, 17.06.2015 r.

Laboratorium

Programowania Komputerów

Temat:

Rozwiązanie problemu komiwojażera przy użyciu algorytmu genetycznego

Autor: Alicja Lachman

Informatyka, semestr 2, grupa 1

Prowadzący: dr inż. Adam Gudyś

1. Temat

Rozwiązanie problemu komiwojażera polega na znalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważonym, czyli innymi słowy znalezieniu najkrótszej trasy przebiegającej dokładnie jeden raz przez każde z zadanych miast. Problem ten jest NP-trudny. Przyjęto, że rozważany graf jest symetryczny, tj. odległość z miasta A do miasta B jest taka sama jak z miasta B do miasta A.

Jedną ze znanych metod rozwiązania problemu komiwojażera jest użycie algorytmu genetycznego. Uzyskane w ten sposób rozwiązanie jest rozwiązaniem heurystycznym, tj. niekoniecznie najbardziej optymalnym, ale znajdującym się dość blisko optimum.

2. Analiza, projektowanie

2.1 Algorytmy, struktury danych, ograniczenia specyfikacji

Do przechowywania danych w programie wykorzystano tablice dynamiczne. Ich rozmiar określany jest na podstawie danych wczytanych z pliku wejściowego.

Do znalezienia rozwiązania problemu wykorzystano, zgodnie z wymogami zadania, algorytm genetyczny, który zostanie szerzej opisany w podpunkcie 2.2

2.2. Analiza problemu, podstawy teoretyczne

Algorytm genetyczny wzorowany jest na darwinowskiej teorii ewolucji, a stosowana w nim terminologia została zapożyczona z biologii. Zatem, rozwiązania danego problemu, w tym przypadku kolejność odwiedzanych miast, reprezentowane są przez populację osobników. Każdy osobnik posiada chromosom – określający kolejność odwiedzanych miast – oraz tworzące go geny – poszczególne miasta. W procesie reprodukcji powstają nowe osobniki (rozwiązania), które dziedziczą cechy swoich rodziców. Jak w przyrodzie, największe szanse na reprodukcję mają najlepsze osobniki.

Etapy przebiegu algorytmu genetycznego:

- **generowanie populacji początkowej**
  Generacja populacji początkowej polega na wylosowaniu określonej ilości rozwiązań. Ilość rodziców zdefiniowana jest w pliku wejściowym i równa się wielkości populacji.

- **dobór sposobu kodowania**
  W programie wykorzystano reprezentację ścieżkową, która w najbardziej naturalny sposób oddaje problem komiwojażera. W reprezentacji tej trasa 2-4-3-1 (gdzie liczby oznaczają numer odwiedzanego miasta) zapisywana jest w postaci chromosomu [2 4 3 1].

- **ocena jakości uzyskanych osobników**
  Ocenę uzyskanych osobników wylicza się na podstawie symetrycznej macierzy odległości pomiędzy miastami, która jest dostarczana w pliku wejściowym.

- **krzyżowanie**
  W programie wykorzystano operator krzyżowania z porządkowaniem, zwany OX.
  W operatorze tym potomek powstaje przez wybranie podtrasy od jednego rodzica i pozostawienie wzajemnego uporządkowania miast z drugiego rodzica. W krzyżowaniu OX korzysta się z faktu, że w reprezentacji ścieżkowej istotna jest uporządkowanie miast, a nie ich pozycja. Do krzyżowania zostają wybrane osobniki z lepszej połowy, oprócz osobnika najlepszego w danym pokoleniu, który jest przepisywany bez zmian do pokolenia następnego.

- **mutacja**
  Aby zapobiec utknięciu w lokalnym minimum rozwiązań, zastosowano proces mutacji. Każdy chromosom, oprócz rozwiązania dotąd najlepszego, które ma pozostać niezmienione, ma 50% szans na wystąpienie mutacji. Mutacja polega na zamienieniu miejscami dwóch losowo wybranych genów.

- **warunek zatrzymania**
  Warunkiem zatrzymania działania algorytmu genetycznego jest wygenerowanie określonej liczby pokoleń, która określana jest w pliku wejściowym.

3. Specyfikacja zewnętrzna

3.1. Obsługa programu

Przed uruchomieniem programu należy podać jako argumenty polecenia ścieżki do dwóch plików: pliku wejściowego, w którym znajdują się dane opisane szczegołówo w podpunkcie 3.2 oraz pliku wyjściowego, do którego zapisane zostaną wyniki pracy algorytmu.
W przypadku niewprowadzenia odpowiedniej ilości argumentów lub błędnych argumentów, użytkownik zostanie o tym poinformowany odpowiednim komunikatem oraz nastąpi zakończenie pracy programu. Po poprawnym wczytaniu danych z plików, następuje rozpoczęcie pracy algorytmu. W co tysięcznym pokoleniu wyświetlany jest najlepszy rezultat z danego pokolenia. Po zakończeniu pracy algorytmu na ekranie wyświetlone zostaje najlepsze rozwiązanie wraz z odpowiadającym mu „chromosomem”, czyli kolejnością odwiedzenia miast.

3.2 Format danych wejściowych

Jako pierwszy argument polecenia należy podać ścieżkę do pliku tekstowego o następującym formacie:

liczba miast w rozważanym problemie
ilość osobników w populacji
liczba pokoleń
symetryczna macierz odległości pomiędzy miastami

Każda informacja powinna znajdować się w osobnym wierszu pliku. W przypadku macierzy odległości dopuszczalne są dodatnie liczby zmiennoprzecinkowe, w pozostałych przypadkach program oczekuje dodatnich liczb całkowitych. Na końcu wiersza nie powinny występować żadne znaki białe poza przejściem do nowej linii. W przypadku macierzy odległości, kolejne wartości mogą być od siebie oddzielone spacją lub tabulacją.

3.3 Komunikaty

W przypadku podania błędnej ilości argumentów polecenia, wyświetlony zostanie komunikat:

```
Bledna ilosc parametrow!
Nacisnij dowolny klawisz, aby zakonczyc
```

Jeśli podany jako argument polecenia plik wejściowy nie istnieje, wyświetlony zostanie napis:

```
Blad otwarcia pliku!
Nacisnij dowolny klawisz, aby zakonczyc
```

Jeśli plik wejściowy nie jest odpowiednio sformatowany, użytkownik zostanie poinformowany przez:

```
Nieprawidlowy format pliku!
Nacisnij dowolny klawisz aby zakonczyc
```

4. Specyfikacja wewnętrzna

4.1. Zmienne

We wszystkich funkcjach zmienne sterujące pętli oraz zmienne przechowujące pomocnicze wartości losowe nazwane są pojedynczymi literami alfabetu.

Najważniejsze zmienne funkcji main:

`*plikIN, *plikOUT`	Wskaźniki na strukturę FILE, wskazują odpowiednio na plik wejściowy oraz wyjściowy
`liczbaMiast`	Zmienna typu int, odczytywana z pliku wejściowego, określa ilość miast w rozważanym problemie komiwojażera
`populacja`	Zmienna typu int, odczytywana z pliku wejściowego, określa ilość osobników w populacji
`iloscPokolen`	Zmienna typu int, odczytywana z pliku wejściowego, określa ilość pokoleń, dla których należy zastosować algorytm genetyczny
`**rodzice`	Wskaźnik na tablicę wskaźników na int, zawiera zbiór pierwotnych rozwiązań problemu, tzw. rodziców
`**pokolenie`	Wskaźnik na tablicę wskaźników na int, zawiera kolejne rozwiązania problemu
`**macierzOdleglosci`	Wskaźnik na tablicę wskaźników na float, zawiera odległości pomiędzy poszczególnymi miastami
`dystans`	Zmienna typu float, przechowuje wyliczoną dla danego rozwiązania odległość pomiędzy miastami

4.2 Funkcje

`int losowanie(int zakres)`	Funkcja zwracająca losową liczbę całkowitą z zakresu <0; zakres-1>
`int sprawdzPowtorzenia(int *chromosom,int iloscLiczb, int liczba)`	Funkcja sprawdzająca, czy do chromosu nie wylosowano powtarzającego się genu (numeru miasta)
`int *losuj(int ilosc)`	Funkcja zwracająca wskaźnik na tablicę wylosowanych rozwiazań, przy czym liczby (kolejność odwiedzanych miast) nie mogą się powtarzać. Funkcja ta wykorzystywana jest do generowania rodziców.
`void generujRodzicow(int **rodzice, int iloscMiast, int iloscRodzicow)`	Funkcja generująca pierwsze pokolenie rozwiązań - rodziców
`float obliczKoszt(int iloscMiast, int *chromosom, float **macierzOdleglosci)`	Funkcja obliczająca koszt podróży pomiędzy poszczególnymi miastami, zgodnie z kolejnością zawartą w chromosomie
`void sortowanie(int **rozwiazania, int liczbaMiast, int populacja, float **macierzOdleglosci)`	Funkcja sortująca rosnąco rozwiązania (chromosomy) według ich kosztów Zastosowano algorytm sortowania przez wybór.
`void rozmnazanie(int **rodzice, int **pokolenie, int iloscRodzicow, int rozmiarGenu)`	Funkcja rozmnażania, działająca na podstawie algorytmu krzyżowania z porządkowaniem
`void mutacja(int **pokolenie, int dlugoscGenu, int populacja)`	Funkcja dokonująca mutacji, tj. zamiany losowych genów w chromosomie. Mutacja następuje z prawdopodobieństwem 50%.
`void przepisz(int **A, int **B, int iloscWierszy, int iloscKolumn)`	Funkcja przepisująca dane z dwuwymiarowej tablicy A do dwuwymiarowej tablicy B

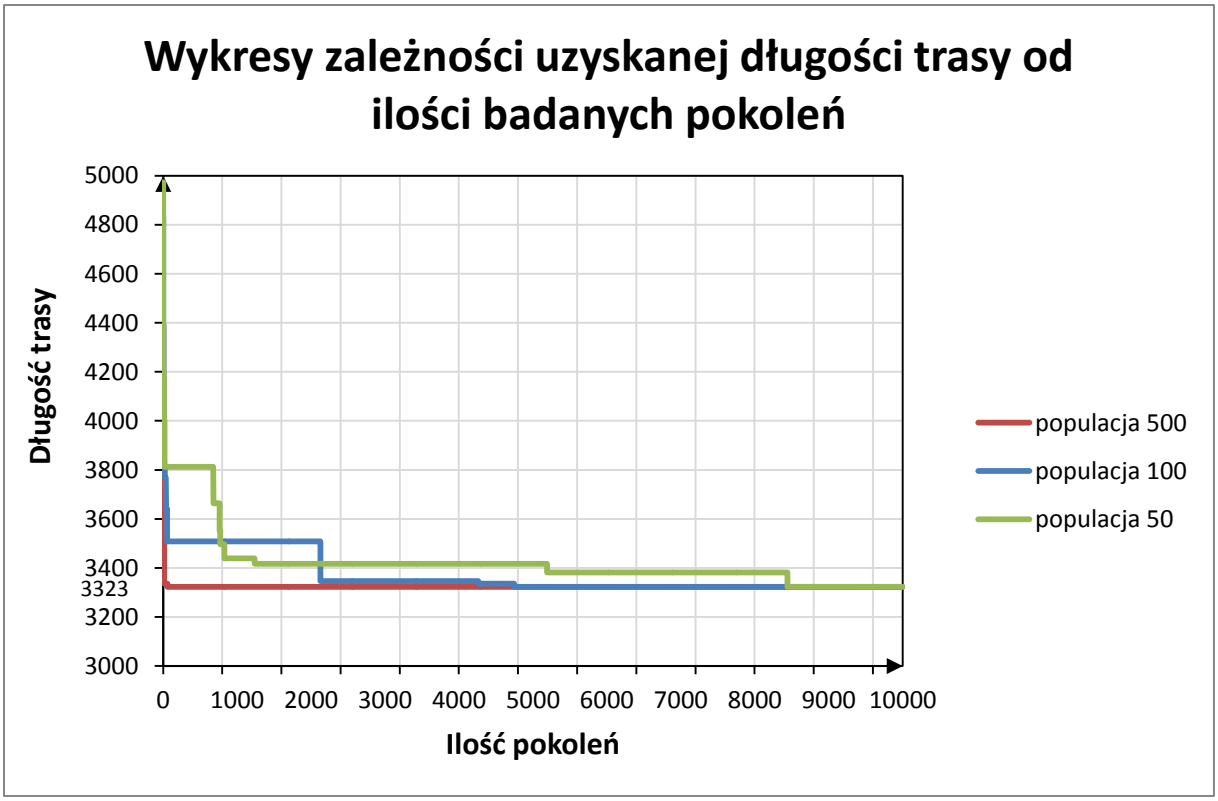
5. Testowanie

Testowanie zostało przeprowadzone dla kilku problemów o znanych rozwiązaniach, zaczerpniętych ze strony http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/. W każdym przypadku, tj. dla 14, 17 i 29 miast, program znajdywał rozwiązanie optymalne lub bliskie optymalnemu. Zależało to od zadanych przez użytkownika parametrów, tj. ilości osobników w populacji a także ilości pokoleń. Poniżej przedstawiono wykres obrazujący wyniki pracy programu szukającego rozwiązania dla 14 miast, 10 000 pokoleń oraz zmiennej liczby osobników w populacji. Rozwiązanie optymalne jest znane i wynosi 3323.

Dla 50 osobników w populacji, rozwiązanie optymalne zostało znalezione w 8444 pokoleniu.

Dla 100 osobników w populacji, rozwiązanie optymalne znalezione zostało w 4748 pokoleniu.

Dla 500 osobników w populacji, rozwiązanie optymalne zostało znalezione w 60 pokoleniu.



6. Wnioski

Algorytm genetyczny okazał się doskonałym algorytmem do rozwiązywania problemu komiwojażera. Znalezienia rozwiązania optymalnego zależy od liczby miast w problemie, zadanej ilości osobników w populacji oraz ilości badanych pokoleń. Ze względu na duże znaczenie losowości w przeprowadzanych działaniach (np. rozmnażanie, mutacje), nie zawsze udaje się uzyskać dokładnie rozwiązanie optymalne, a jedynie rozwiązanie zbliżone do optimum. Zwiększenie liczby osobników w populacji oraz ilości badanych pokoleń pozwala na zminimalizowanie występowania takich sytuacji.

7. Bibliografia

1. Michalewicz Zbigniew, *Algorytmy genetyczne + struktury genetyczne = programy ewolucyjne*, Warszawa, Wydawnictwo Naukowo-Techniczne, 2003, ISBN 83-204-2881-5
2. Debudaj-Grabysz A., Deorowicz S., Widuch J., *Algorytmy i struktury danych. Wybór zaawansowanych metod*, Gliwice, Wydawnictwo Politechniki Śląskiej, 2012, ISBN 978-83-7335-938-3
3. Sivanandam S.N., Deepa S.N., *Introduction to Genetic Algorithms*, Springer, 2008, ISBN 978-3-540-73189-4, [dostęp przez Google Books 25.05.2015 r.]
4. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ [dostęp: 23.05.2015 r.]