Rozwiązanie zadania komiwojażera przy pomocy algorytmu genetycznego. 1) Wersja sekwencyjna i zrównoleglona (pamięć wspólna).

Onaszkiewicz Przemysław, Gadawski Łukasz

30 listopada 2015

1 Cel zadania

Celem pierwszego etapu projektu jest implementacja wersji sekwencyjnej algorytmu genetycznego działającego na pojedynczym procesorze do rozwiązywania problemu komiwojażera. A także zastosowanie dyrektyw *OpenMP* w celu dokonania zrównoleglenia wykonania algorytmu na wielu procesorach z pamięcią wspólną.

2 Problem komiwojażera

Problem zdefiniowany jest następująco: mając listę miast oraz odległości między nimi, znajdź najkrótszą, dozwoloną drogę (początkiem oraz końcem drogi jest ten sam wierzchołek), zawierającą każde miasto dokładnie raz. Przy czym można zacząć od dowolnego miasta oraz kolejność miast jest dowolna. Inaczej problem polega na znalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważonym. Problem należy do zbioru problemów NP-trudnych.

3 Algorytm genetyczny

W 1975 roku przez Johna Hollanda został wynaleziony algorytm genetyczny, którego zadaniem było działanie analogiczne do procesu ewolucji na ziemi. Pseudokod prostego algorytmu genetycznego (ang. *Simple Genetic Algorithm*) został przedstawiony poniżej:

Sterowanie algorytmem genetycznym odbywa się poprzez warunek stopu, którym może być satysfakcjonująca wartość funkcji celu lub ilość generacji. Typowy algorytm genetyczny rozpoczyna się inicjacją populacji wstępnej. W przypadku problemu komiwojażera **populacją** będzie lista osobników, natomiast każdy **osobnik** będzie konkretną listą miast, czyli jednym ze stanów z całej przestrzeni stanów. W przypadku problemu komiwojażera przestrzenią stanów jest zbiór wszystkich możliwych kombinacji dróg pomiędzy miastami. Następnie dokonywana jest ocena wartości każdego osobnika w populacji. W naszym

Algorithm 1 Simple Genetic Algorithm

```
1: procedure SIMPLE GENETIC ALGORITHM
2:     t ← 0
3:     initilize P^t
4:     evaluate P^t
5: while (not stop_condition):
6:     O^t ← reproduce P^t
7:     crossover O^t
8:     mutate O^t
9:     evaluate O^t
10:    P^{t+1} ← O^t
11:    t ← t + 1
```

przypadku będzie to obliczenie drogi każdego z zainicjowanych stanów. Kolejno następuje *reprodukcja* populacji, która zostanie opisana dokładniej w kolejnym punkcie. Następnie wykonywane są operacje krzyżowania oraz mutacji, czyli "urozmaicanie" aktualnej populacji, a w dalszej części wytypowanie populacji, która będzie stanowiła najlepszych osobników do reprodukcji w kolejnej generacji.

3.1 Reprodukcja

Reprodukacja polega na wytypowaniu osobników do populacji potomnych, ale tak aby preferować osobniki lepiej przystosowane, czyli posiadające lepszą wartość funkcji przystosowania. Ocena jakości osobników dokonywana jest na podstawie funkcji przystosowania. W przypadku problemu komiwojażera mniejsza droga jest lepszym rezultatem, a zatem dąży się do minimalizacji drogi, którą reprezentuje każdy osobnik. Podczas reprodukcji następuje losowy wybór osobników do populacji potomnej, ale prawdopodobieństwo wylosowania każdego z nich nie jest jednakowe. W przypadku osobników lepiej przystosowanych występuje większe prawdopodobieństwo. Wartość prawdopodobieństwa obliczana jest zgodnie z następującym wzorem:

$$p_i = \frac{len\_sum - len_i}{\sum_i^I (len\_sum - x_i)}, \text{ gdzie}$$

len_sum - suma długości wszystkich osobników w populacji,
len_i - długość i-tego osobnika, dla którego jest liczone prawdopodobieństwo,

Następnie mając prawdopodobieństwa osobników w populacji następuje obliczenie dystrybuanty prawdopodobieństwa dla każdego osobnika. Podczas losowania osobników następuje losowanie liczby z przedziału $(0.00, 1.00)$, a następnie wybranie osobnika, który odpowiada danej wartości dystrybuanty. Taki dobór nazywany jest selekcją ruletkową (ang. *round-wheel selection*).

3.2 Mutacja

Operacja mutacji wykonywana jest z określoną wartością prawdopodobieństwa, która jest parametrem algorytmu. Zgodnie z prawdopodobieństwem podejmowana jest decyzja o zmutowania konkretnego osobnika. Proces polega na losowej

zamianie pozycji dwóch miast w początkowej liście miast. Dzięki temu zostaje wprowadzone całkowicie losowe zróżnicowanie dwóch osobników. Przeważnie wartość tego parametru ustawiona jest na niską wartość, tak aby wprowadzać różnorodność w każdym pokoleniu, ale aby również nie zakłócać polepszania wartości funkcji dostosowania w pokoleniu.

3.3 Krzyżowanie

Operacja krzyżowania również ma na celu wprowadzenie różnorodności w osobnikach w kolejnych pokoleniach. Jest wykonywana z określoną wartością prawdopodobieństwa, która jest parametrem algorytmu. Wszystkie osobniki w populacji są dobierane w pary. Następnie dla każdej z pary zgodnie z parametrem jest podejmowana decyzja o dokonaniu operacji. W dalszej kolejności losowany zostaje indeks krzyżowania. Zgodnie z tym indeksem pierwsza część chromosomu, czyli listy miast, jest zachowywana, natomiast druga część listy miast jest zastępowana wzajemnie z drugim osobnikiem z pary.

4 Implementacja

4.1 Tworzenie mapy miast

W naszej implementacji mapa została odwzorowana jako obiekt posiadający wielkość, zbiór miast oraz strukturę danych *mapa* gdzie kluczem jest para miasto-miasto oraz odległość między taką na mapie.

4.1.1 Wczytanie z pliku

4.1.2 Losowa generacja mapy

4.2 Konfiguracja programu

Konfiguracja programu jest możliwa przy pomocy pliku *app.properties*. Umożliwia on zdefiniowanie następujących parametrów:

- **population_size** - wielkość populacji w każdej iteracji algorytmu,
- **propability_of_crossover** - prawdopodobieństwo krzyżowania,
- **propability_of_mutation** - prawdopodobieństwo mutacji,
- **generation_number** - liczba pokoleń, czyli iteracji algorytmu, w przypadku naszej implementacji jest to warunek stopu.

Przykładowa zawartość pliku konfiguracyjnego została zamieszczona poniżej.

```
population_size=5
propability_of_crossover=0.20
propability_of_mutation=0.01
generation_number=3
```

5 Wnioski

6 Podsumowanie