Badanie operacyjne i teoria złożoności obliczeniowej
Projekt – problem komiwojażera

Zespół:

1. Agnieszka Szczurek
2. Piotr Konsek
3. Krzysztof Płachno

A. Teoria problemu komiwojażera

Cykl Hamiltona (ang. Hamilton circuit) jest ścieżką w grafie, która przechodzi przez wszystkie jego wierzchołki dokładnie jeden raz i wraca do wierzchołka startowego. Jeśli ścieżka nie wraca do wierzchołka startowego, ale przechodzi przez każdy wierzchołek grafu dokładnie jeden raz, to nazywamy ją ścieżką Hamiltona (ang. Hamilton path). Graf posiadający cykl Hamiltona nazywamy grafem Hamiltona (ang. Hamilton graph). W przypadku cykli Hamiltona nie istnieje jak dotąd proste kryterium, które dla dowolnego grafu pozwala stwierdzić istnienie cyklu Hamiltona. Rozwiązano jedynie przypadki szczególne. Problem znajdowania takich cykli określa się jako trudny obliczeniowo. Problem komiwojażera jest zagadnieniem optymalizacyjnym polegającym na znalezienie minimalnego cyklu Hamiltona w grafie pełnym. Nazwa pochodzi od typowej ilustracji problemu, przedstawiającej go z punktu widzenie wędrownego sprzedawcy. Dane jest n miast, które komiwojażer ma odwiedzić, oraz odległości między nimi. Celem jest znalezienie najkrótszej, najbardziej optymalnej ścieżki łączącej wszystkie miasta oraz zaczynające i kończące się w tym samym miejscu.

B. Opis naszego programu

1. Ogólny opis interfejsu programu

   Program dostarcza możliwość generowania losowych grafów o zadanej liczbie wierzchołków, tworzenia własnych grafów planarnych na drodze wyklikania ich wierzchołków na białej planszy, tworzenia własnych grafów kołowych, zapisania utworzonego grafu lub wczytania wcześniej zapisanego z pliku oraz skorzystania z pokazowych grafów.
   Dla grafów ilości wierzchołków mniejszej niż 150 na białej planszy pokazywana jest od razu „poprawna ścieżka” kolorem czerwonym wygenerowana za pomocą klasy Hamilton.java oraz sukcesywnie uaktualniana na zielono aktualnie najlepsza ścieżka wygenerowana przez nasz algorytm.

2. Ogólny opis algorytmu

   Program wykorzystuje algorytm genetyczny. Dla zadanego grafu generuje losową populację domyślnie zawierającą tysiąc osobników, gdzie każdy osobnik reprezentuje jedną z możliwych cykli Hamiltona (tras komiwojażera). W każdym cyklu pętli głównej programu

następuje najpierw reprodukcja populacji wg algorytmu ruletkowego (najlepsze osobniki – te reprezentujące najkrótsze ścieżki, mają największe szanse na wejście do nowego pokolenia). Następnie osobniki krzyżują się między sobą oraz mutują (zgodnie z wyspecyfikowanymi metodami). Pod koniec pętli następuje ocena populacji – odszukanie najlepszego i najgorszego osobnika oraz wyliczenie średniej wartości tras osobników. W zależności od wybranej metody zakończenia program decyduje czy kontynuować algorytm czy też zakończyć, traktując trasę najlepszego osobnika bieżącej populacji jako rozwiązanie problemu.

3. Metoda kodowania

   Program stosuje „porządkową" metodę kodowania trasy w osobnikach.

   „Bazą" dla tego kodowania jest ciąg wierzchołków posortowany rosnąco wg ich indeksów:

   1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9

   zakodowana ścieżka poniżej:

   1 1 2 1 4 1 3 1 1

   oznacza trasę 1-2-4-3-8-5-9-6-7. Kodowanie to wygląda w ten sposób, że dany element zakodowanej ścieżki oznacza n-ty indeks miasta z jeszcze pozostałym w „bazie". Po wybraniu danego miasta jego indeks usuwamy z „bazy" a indeksy wierzchołków zsuwamy, zapełniając lukę.

4. Metody krzyżowania
   Zastosowane kodowanie ścieżek pozwala na wykorzystanie standardowych operacji krzyżowania

   a) Jednopunktowe
      polega na „rozcięciu" dwóch zakodowanych ścieżek w jednym losowym miejscu (tym samym dla obu ścieżek), zamianie odciętych części i złączenia ich powtórnie w zakodowane ścieżki.
      Algorytm dopuszcza rozcięcie za ostatnim elementem zakodowanej ścieżki co oznacza krzyżowanie o osobnikach potomnych takich samych jak macierzyste.

   b) Dwupunktowe
      polega na „rozcięciu dwóch zakodowanych ścieżek w dwóch losowych miejscach (tych samych dla obu ścieżek) wymianie między osobnikami środkowych kawałków i złączenia w nowe osobniki.
      Algorytm dopuszcza rozcięcia przed pierwszym i za ostatnim elementem zakodowanej ścieżki co oznacza krzyżowanie o osobnikach potomnych takich samych jak macierzyste.

5. Metody selekcji
   Określają które osobniki z danego pokolenia przetrwają i wezmą udział w kodowaniu.
   a) Z połowicznym zastępowaniem
      Połowa lepszych osobników populacji bierze udział w krzyżowaniu. Po tym procesie populacja składa się ze niezmodyfikowanych rodziców i ich dzieci (każda para rodziców ma dwójkę dzieci)
   b) Z całkowitym zastępowaniem
      Cała populacja bierze udział w krzyżowaniu. Po tym procesie populacja składa się tylko z dzieci (każda para rodziców produkuje dwójkę dzieci)

6. Metody mutacji
   Prawdopodobieństwo zajścia mutacji jest domyślnie ustawione na to, że w zajdzie ona dla jednego procenta populacji.
   a) Wierzchołkowa
      Polega na odkodowaniu ścieżki do prostej postaci np.:
      1 – 2 – 3 oznacza trasę wierzchołek 1 – wierzchołek 2 – wierzchołek 3
      i w tej zakodowanej postaci losowego wybrania wierzchołków, które zamienimy miejscami w ścieżce odkodowanej. (Najpierw losujemy jeden wierzchołek, potem drugi i zamieniamy) Po mutacji na powrót kodujemy ścieżkę.
      Algorytm dopuszcza zamianę wierzchołka z samym sobą, co oznacza brak mutacji.

   b) Porządkowa
      Polega na zmianie którejś z wartości zakodowanej ścieżki na inną losową (spełniającą ograniczenie kodowania – na n-tej pozycji nie może być wartość większa niż n).

7. Reprodukcja
   Zastosowano metodę ruletkową. Najpierw wyliczane są aktualne długości ścieżek. Wybierana jest najdłuższa z nich. Następnie wyliczane są oceny osobników – najdłuższa ścieżka w populacji pomniejszona o długość ścieżki danego osobnika. (Najlepszą ocenę uzyska osobnik z najkrótszą ścieżką) Oceny osobników są sumowane i ocena każdego z nich jest dzielona przez tą sumę, aby uzyskać procentową udział danej oceny w sumie. Następuje odcinkowe uszeregowanie osobników wg ich ocen od 0 do wartości 1 i losowanie wartości double z przedziału [0,1] i wg tego na odcinku którego osobnika wypadnie ta wartość ten osobnik przechodzi do nowego pokolenia.

8. Metody zakończenia

   a) Limit iteracji
      Algorytm kończy się po wykonaniu się wyspecyfikowanej ilości iteracji. Wynikiem jest długość ścieżki najlepszego osobnika z ostatniej populacji.

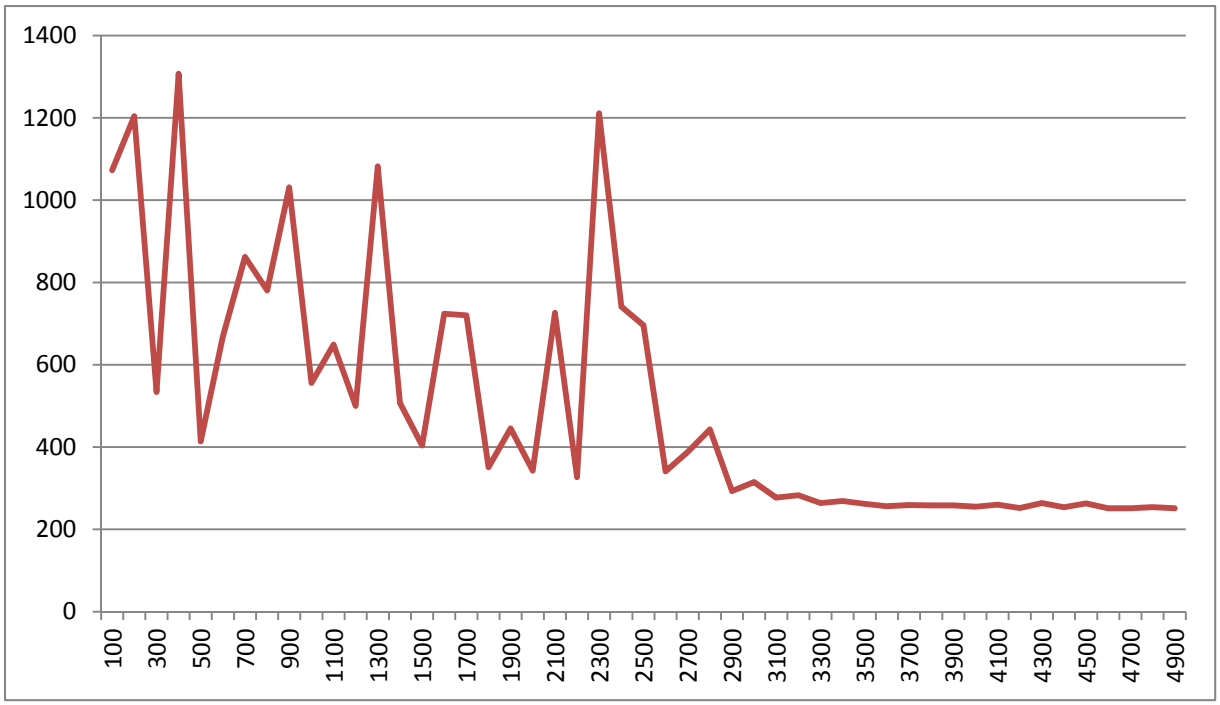
   b) Osiągnięcie wypłaszczenia
      Po każdej iteracji algorytm wyszukuje z ostatnich 200 pokoleń najlepszego i najgorszego osobnika i porównuje je z najlepszym osobnikiem z bieżącego pokolenia. Jeśli

podobieństwo bieżącego najlepszego do tych ekstremalnych z 200 ostatnich pokoleń wynosi 98% to algorytm się zakańcza. Obie wartości – ilość rozważanych pokoleń wstecz oraz wartość podobieństwa jest do wyspecyfikowania.

c) Podobieństwo do „poprawnego" rozwiązania
Za pomocą zewnętrznej biblioteki Hamilton.java wyznaczamy „poprawne" rozwiązanie dla danego problemu i jeśli nasz osiągnie rozwiązanie o wyspecyfikowanym prawdopodobieństwie do „poprawnego" rozwiązania to algorytm się zakańcza.

C. Dodatkowe opracowania

Wykres ilości iteracji do osiągnięcia wypłaszczenia w zależności od rozmiaru grafu.
Od rozmiaru populacji od 3300 zmniejszyliśmy rozmiar populacji z 1000 do 600.



Przeprowadziliśmy badanie jakie połączenie mutacji i sukcesji daje najlepsze osiągnięcie wypłaszczenia. Wykresy krzywych zbieżności w załączniku. Nie zaobserwowaliśmy żadnych zależności co do efektywności poszczególnych metod.
Najprawdopodobniej jest to wynik tego, że algorytm czasami szybciej dojdzie do jakiegoś minimum lokalnego i zakończy swoje działanie z wynikiem nieoptymalnym.