Projektowanie Efektywnych Algorytmów

Projekt

07/11/2023

263896 Patryk Jurkiewicz

(2) Held-Karp

1. Sformułowanie zadania

Zadanie, które przed nami stoi, polega na stworzeniu rozwiązania opartego na algorytmie Held-Karpa, mającego na celu rozwiązanie problemu komiwojażera. Problem komiwojażera jest jednym z kluczowych problemów w dziedzinie optymalizacji kombinatorycznej, polegającym na znalezieniu najkrótszej trasy, która prowadzi przez wszystkie miasta (wierzchołki) dokładnie jeden raz, a na końcu powraca do miasta startowego. Algorytm Held-Karpa, będący techniką dynamicznego programowania, stanowi nasze narzędzie do efektywnego rozwiązania tego trudnego problemu. Naszym celem jest stworzenie aplikacji, która będzie w stanie samodzielnie rozwiązywać problem komiwojażera przy użyciu algorytmu opartego na Held-Karpie. W ramach tej aplikacji zostanie zaimplementowany cały proces, począwszy od wczytania danych wejściowych (informacje o miastach i odległościach między nimi), poprzez wykorzystanie algorytmu Held-Karpa do znalezienia optymalnej trasy, aż po przedstawienie wyników w czytelnej formie.

Raport końcowy, który przygotujemy, będzie zawierał opis algorytmu Held-Karpa, ilustrując jego działanie i zasady. Nasza aplikacja będzie miała zastosowanie w analizie i rozwiązywaniu problemów komiwojażera, a także znajdzie zastosowanie w dziedzinach takich jak logistyka, planowanie tras, czy rozwijanie algorytmów optymalizacyjnych. Dzięki niej będziemy w stanie skutecznie optymalizować trasy, co ma ogromne znaczenie zarówno w dziedzinie biznesowej, jak i naukowej, poprawiając efektywność działań oraz generując oszczędności czasu i zasobów. Danie, które stawiamy przed sobą, obejmuje opracowanie i implementację algorytmu bazującego na znanej technice dynamicznego programowania, jaką jest algorytm Held-Karpa, w celu rozwiązania problemu komiwojażera. Ten problem ma ogromne znaczenie w wielu dziedzinach, takich jak logistyka, planowanie tras czy rozwijanie algorytmów optymalizacyjnych, dlatego też nasze działania mają na celu stworzenie aplikacji, która skutecznie znajdzie najkrótszą trasę łączącą wszystkie miasta, odwiedzając każde z nich dokładnie raz, z powrotem wracając do miasta startowego. W praktyce, aplikacja ta będzie służyć do analizy i rozwiązania problemu komiwojażera, a także znajdzie zastosowanie w konkretnej branży, gdzie wyznaczanie optymalnych tras jest kluczowym elementem efektywnego funkcjonowania.

2. Metoda

Algorytm Held-Karp to zaawansowane narzędzie służące do rozwiązywania problemu komiwojażera (TSP), który jest jednym z najbardziej znanych problemów optymalizacyjnych. Celem TSP jest znalezienie najkrótszej trasy, która odwiedza każde z zadanego zbioru miast dokładnie raz, a następnie wraca do miasta początkowego. Rozwiązanie tego problemu ma liczne praktyczne zastosowania, takie jak planowanie tras dostaw, projektowanie obwodów drukowanych, czy optymalizacja tras w systemach nawigacyjnych. Algorytm Held-Karp wykorzystuje podejście programowania dynamicznego, co oznacza, że skupia się na rozwiązywaniu podproblemów i łączeniu ich wyników, aby znaleźć ostateczne rozwiązanie problemu. Jednym z kluczowych elementów algorytmu jest tablica dynamiczna, w której przechowywane są wyniki pośrednie. Dla każdej maski miast, algorytm oblicza najkrótsze trasy odwiedzania miast i zapisuje wyniki w tej tablicy. Algorytm bada wszystkie możliwe kombinacje tras, aby znaleźć tę o minimalnym koszcie. W tym celu porównuje długości tras, wybierając najlepszą opcję. Po zakończeniu przeglądu, algorytm wybiera trasę o najmniejszej długości i zapisuje ją w wynikowym rozwiązaniu. Proces ten jest wielokrotnie powtarzany, aby uzyskać dokładniejsze wyniki, a następnie obliczany jest średni czas trwania rozwiązywania problemu. Ostatecznie, algorytm Held-Karp daje możliwość znalezienia optymalnych rozwiązań problemów TSP, ale ma swoje ograniczenia w postaci kosztownego obliczeniowo przeglądu wszystkich możliwych kombinacji. Dlatego jest stosowany głównie w przypadkach, gdzie dokładność jest kluczowa, a inne, bardziej efektywne algorytmy nie są dostępne lub nie dają satysfakcjonujących wyników.

3. Algorytm

Część 1: Inicjalizacja i Wczytywanie Danych

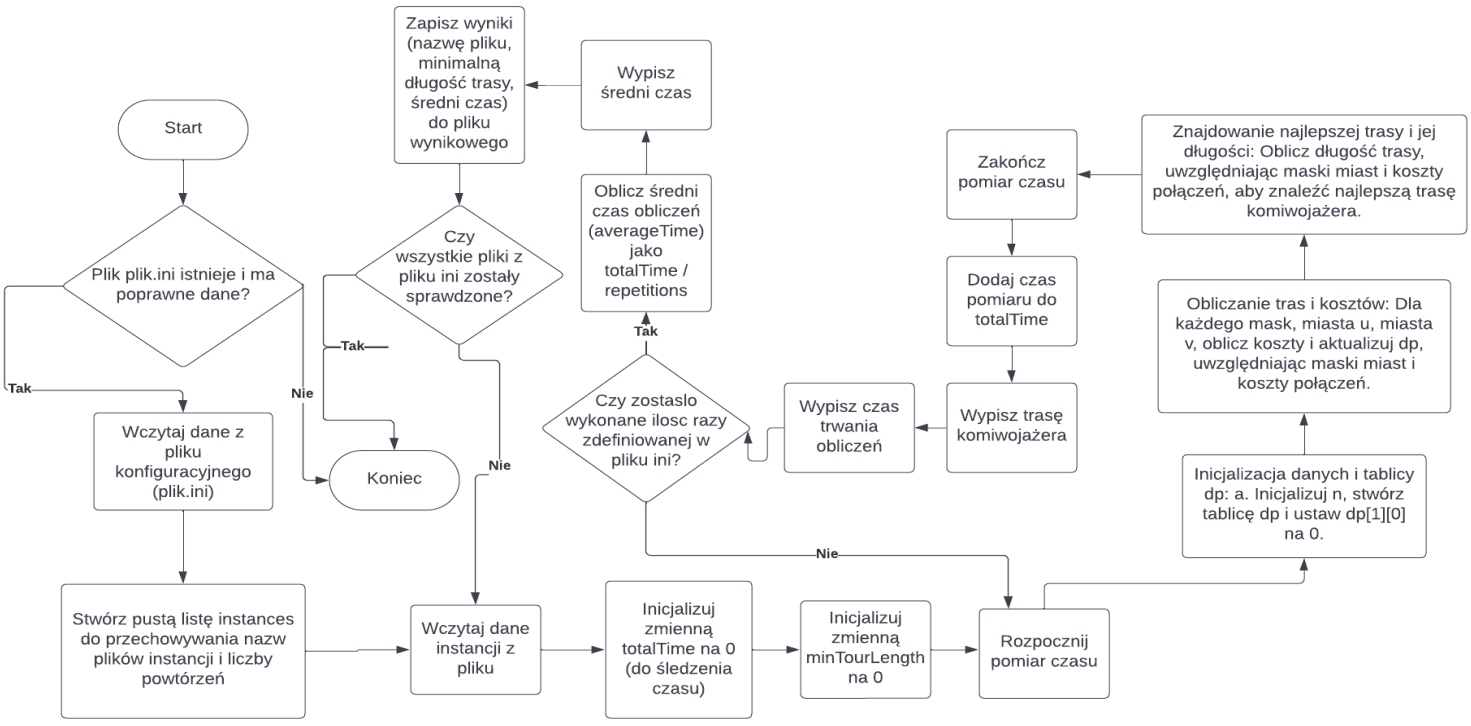
Program rozpoczyna swoje działanie od otwarcia pliku "plik.ini", w którym zawarte są nazwy plików z danymi oraz liczba powtórzeń dla każdej instancji. Następnie inicjalizowane są zmienne, takie jak totalTime i minTourLength, potrzebne do śledzenia czasu wykonania i minimalnej długości trasy.

Część 2: Rozwiązywanie Problemu Komiwojażera

Algorytm rozwiązywania problemu komiwojażera jest realizowany w pętli, który iteruje przez wszystkie wczytane instancje problemu TSP. Dla każdej instancji, program odczytuje nazwę pliku z danymi oraz liczbę powtórzeń, a następnie przystępuje do rozwiązywania problemu. Wewnątrz tej pętli, algorytm rozpoczyna przetwarzanie każdej instancji poprzez odczyt danych z pliku. Następnie algorytm przystępuje do wykonywania określonej liczby powtórzeń w poszukiwaniu optymalnej trasy. Dla każdej iteracji, algorytm korzysta z algorytmu Helda-Karpa do obliczenia trasy, śledzi czas wykonania, a także aktualizuje minimalną długość trasy (minTourLength), jeśli znaleziona trasa jest krótsza niż poprzednia najlepsza trasa. Cały ten proces pozwala na rozwiązanie problemu komiwojażera dla każdej instancji i zbieranie wyników

Część 3: Wyświetlanie Wyników i Zakończenie Programu

Po zakończeniu przetwarzania danej instancji, program oblicza średni czas wykonania i wypisuje wyniki, w tym nazwę pliku, minimalną długość trasy oraz średni czas wykonania. Wyniki te są również zapisywane do pliku "wyniki.txt". Na koniec program zamyka wszystkie otwarte pliki i kończy swoje działanie.



Rys 1. Schemat blokowy algorytmu

4. Dane testowe

W celu przetestowania i dostrojenia naszego algorytmu wybraliśmy zestaw konkretnych instancji problemu komiwojażera. Te instancje posłużą nam do oceny skuteczności i poprawności działania algorytmu oraz do określenia odpowiednich parametrów.

Do wykonania badań wybrano następujący zestaw instancji ze stron:
http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/
http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/XML-TSPLIB/instances/

tsp_6_1.txt

tsp_6_2.txt

tsp10.txt

tsp12.txt

tsp13.txt

tsp14.txt

tsp15.txt

tsp17.txt

burma14.txt

gr21.txt

gr24.txt

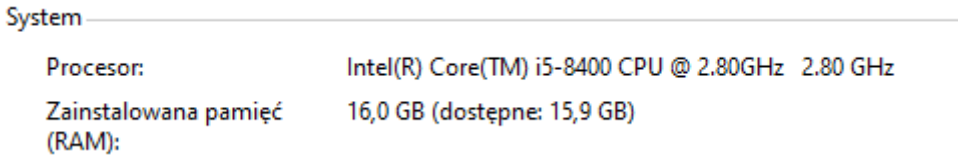
fri26.txt

5. Procedura badawcza

Aby wykonać ten proces, należy wypełnić plik ini liniami z nazwami plików z danymi oraz ilością, ile razy mają zostać powtórzone. Później, należy uruchomić plik wykonywalny (exe). Wyniki zostaną zapisane w pliku "wyjscie.txt" w formacie, który przypomina strukturę przedstawioną w poniżej:

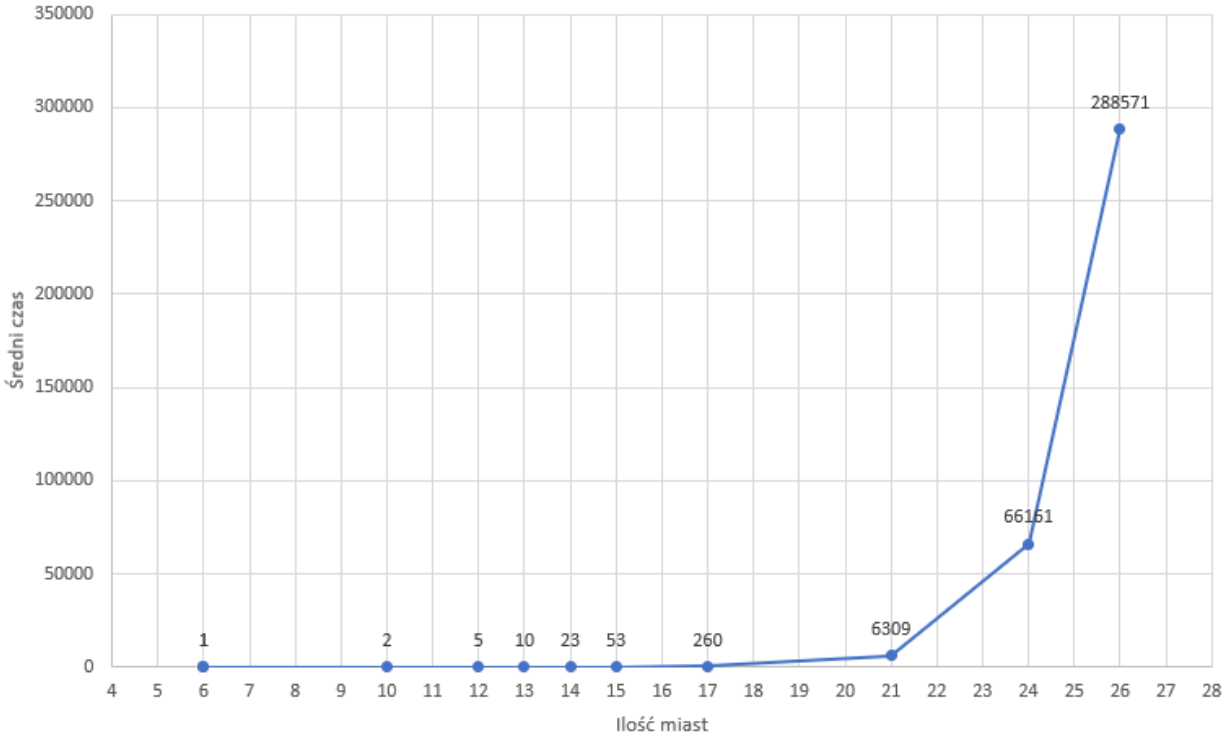
Plik: nazwa_pliku
Minimalna długość trasy: trasa
Średni czas: czas

Program był testowany na komputerze o tej specyfikacji:

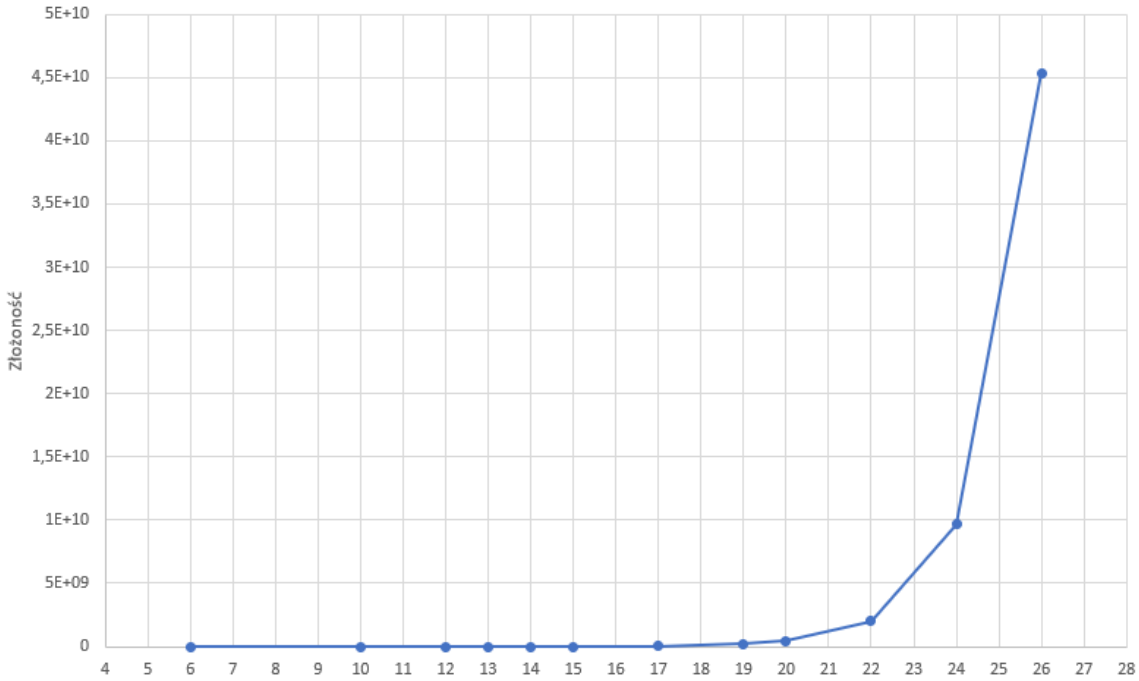


6. Wyniki

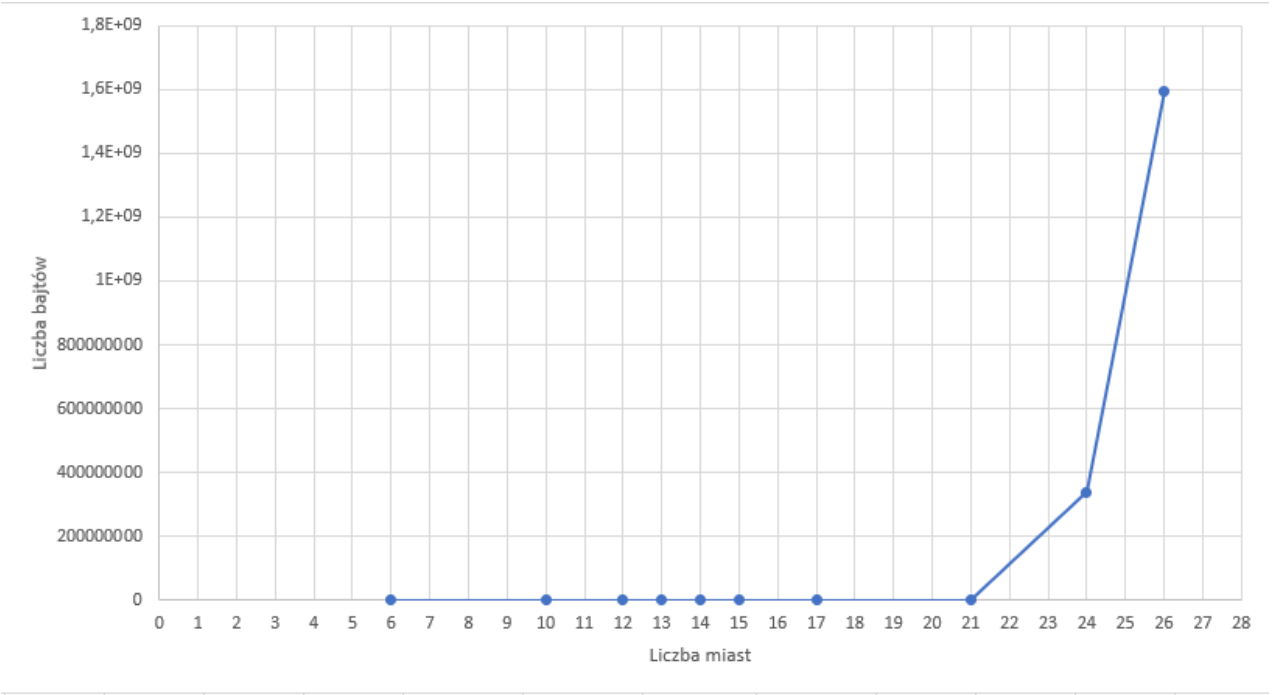
Wyniki zostały zgromadzone w pliku o nazwie wyniki.txt. Wyniki przedstawione zostały w postaci wykresu zależności czasu uzyskania rozwiązania problemu od wielkości instancji (rysunek 2).



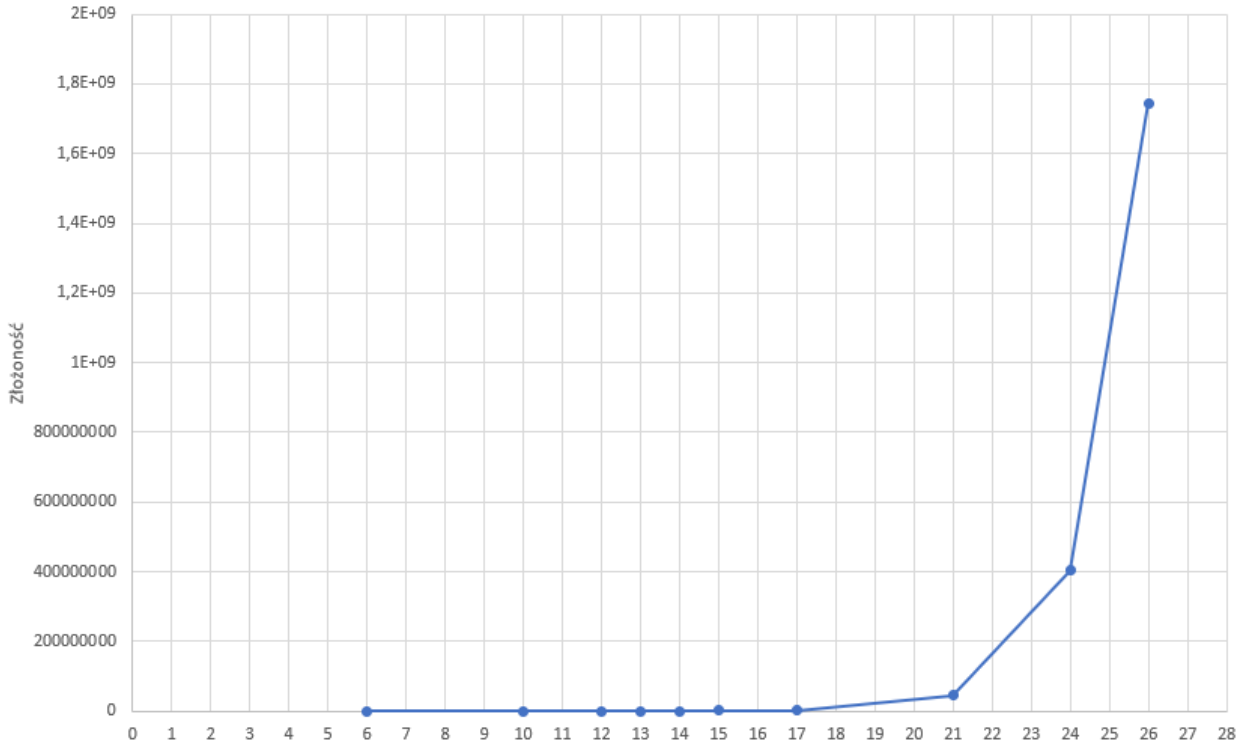
Rys 2. Wykres przedstawiający ilość milisekund potrzebnych na poradzenie sobie z problemem komiwojażera w zależności od liczby miast.



Rys 3. Wykres przedstawiający złożoność O(n^2*2^n) w zależności od n odpowiadającej liczbie miast.



Rys 4. Wykres przedstawiający liczbę bajtów potrzebnych na poradzenie sobie z problemem komiwojażera w zależności od liczby miast.



Rys 5. Wykres przedstawiający złożoność O(n*2^n) w zależności od n odpowiadającej liczbie miast.

7. Analiza wyników i wnioski

Podsumowując analizę przedstawionych danych (rysunek 2.), można zauważyć, że czas potrzebny do rozwiązania problemu komiwojażera znacząco rośnie wraz z liczbą miast, co sugeruje, że trudność tego zadania rośnie wykładniczo w zależności od liczby miast, a krzywa wzrostu czasu wykazuje wyraźny charakter wykładniczy. Warto jednak zaznaczyć, że pomimo tego wykładniczego wzrostu, jest on niższy niż w przypadku użycia brute force. Porównując te wyniki z teoretyczną złożonością obliczeniową wynoszącą O(n^2*2^n) *(rysunek 3.), można dojść do wniosku, że oba wykresy prezentują podobny trend, co sugeruje poprawność przeprowadzonych badań. Podobnie sytuacja ma się w przypadku analizy ilości potrzebnej pamięci (rysunek 4.) i teoretycznej złożoności obliczeniowej wynoszącej O(n*2^n)* (rysunek 5.). W praktyce oznacza to, że dla każdej nowej instancji problemu, czas potrzebny na jego rozwiązanie rośnie nieliniowo w stosunku do liczby miast, co sprawia, że rozwiązywanie problemu komiwojażera dla dużej liczby miast staje się niezwykle trudne i czasochłonne. Warto zaznaczyć, że w przypadku przeprowadzonych badań komputer nie był w stanie obsłużyć większej liczby miast niż 26 ze względu na znaczną potrzebę pamięci do przeprowadzenia obliczeń.