Obliczenia wysokiej wydajności

Zagadnienie komiwojażera

porównanie metody sekwencyjnej oraz równoległej

Maciej Siwicki Paweł Pagacz

18.10.2023

Spis treści

1 Wstęp

Obliczenia sekwencyjne są standardową metodą programowania: komputer wykonuje każdą operację obliczeniową osobno w podanej kolejności. Sekwencyjny model obliczeniowy zakłada, że w danym momencie tylko jedna operacja może być wykonywana i jest to prawda dla systemu składającego się z jednego komputera z jednym procesorem. Większość nowoczesnych komputerów jednakże zawiera procesory wielordzeniowe i każdy rdzeń takiego procesora może wykonywać niezależne operacje obliczeniowe. z tego powodu obliczenia równoległe znajdują zastosowanie w rozwiązywaniu wielu problemów w celu przyspieszenia czasu obliczeń.
W projekcie wykorzystany został problem komiwojażera. Ma on wiele praktycznych zastosowań, od planowania tras logistycznych po projektowanie układów mikroelektronicznych.

2 Cel projektu

Celem projektu jest zapoznanie się z problemem komiwojażera wraz z implementacją algorytmu sekwencyjnego oraz równoległego.

2.1 Zakres projektu

W skład projektu wchodzi:

1. Przedstawienie problemu komiwojażera wraz z opisem rozwiązania metodą szeregową oraz równoległą.
2. Wygenerowanie danych do analizy.
3. Implementacja metody szeregowej.
4. Implementacja metody równoległej.
5. Wizualizacja wyników algorytmów na wykresach.
6. Porównanie czasów działania zaimplementowanych metod.

3 Część teoretyczna

3.1 Problem komiwojażera

Problem komiwojażera jest problemem obliczeniowym polegającym na znalezieniu minimalnego cyklu Hamiltona w grafie G, którego łukom przypisane są dodatnie współczynniki wagi. w najbardziej popularnej interpretacji wierzchołki grafu reprezentują miasta, zaś łuki i przyporządkowane im współczynniki odpowiadają połączeniom między miastami i odległościom między nimi.

Istnieje wiele sposobów rozwiązywania problemu komiwojażera. Jednym z nich jest metoda sprawdzenia wszystkich rozwiązań (*ang. Brute Force*). Przyjmując n jako liczbę wszystkich miast, dla drugiego miasta mamy (n-1) możliwości, a dla trzeciego (n-2) itd. Można zatem zauważyć, że rozwiązanie problemu tą metodą wymagałoby rozpatrzenia $\frac{(n-1)!}{2}$ kombinacji. w tabeli poniżej przedstawiono kilka przykładowych wraz ze złożonością obliczeniową.

Tabela 1: Złożoność obliczeniowa algorytmów

Algorytm	Złożoność obliczeniowa
Sprawdzenie wszystkich wariantów	$O(n!)$
Algorytm Helda-Karpa	$O(n^2 2^n)$

Do dalszych rozważań wybrany został algorytm sprawdzania wszystkich wariantów.

3.2 Metoda sekwencyjna

Do obliczenia metody sekwencyjnej wykorzystano algorytm back-track:

1. Rozważ miasto 1 jako punkt początkowy i końcowy. Ponieważ trasa ma charakter cykliczny, za punkt wyjścia możemy uznać dowolny punkt,
2. Wygeneruj dane do analizy, zbuduj drzewo (graf), zainicjalizuj potrzebne zmienne,
3. Wybierz krok do wykonania (wybierz wierzchołek grafu),
4. Jeśli krok można wykonać, wykonaj algorytm do następnego poziomu drzewa (grafu),
5. Jeśli osiągniesz terminalny wierzchołek drzewa, zapisz ten wynik jako jedno z rozwiązań problemu,
6. Po zakończeniu rekurencji lub znalezieniu rozwiązania, wróć do poprzedniego stanu, aby rozważyć inne możliwości,
7. Kontynuuj proces cofania i wybierania innych opcji aż do momentu, gdy zostaną wypróbowane wszystkie możliwe ścieżki lub zostanie znalezione optymalne rozwiązanie.

3.3 Metoda równoległa

W metodzie równoległej można natomiast:

1. Równolegle wygenerować potomków,
2. Zrównoleglić lokalne przeszukiwanie w ramach pojedynczej gałęzi.

Liczba wątków może być oparta na liczbie dostępnych rdzeni procesora.

3.4 Wyliczenie teoretycznego przyspieszenia

Dla wybranego algorytmu metoda sekwencyjna posiada złożoność obliczeniową $O(n!)$. Złożoność obliczeniowa metody równoległej wynosi natomiast $\frac{(n)!}{p} + p$. Wynika to z konieczności przeszukania wszystkich możliwych rozwiązań przez p procesorów i następnie porównaniu wyników. Teoretyczne przyspieszenie metody równoległej może wynieść zatem $\frac{pn!}{n!+p^2}$, gdzie p oznacza liczbę wątków.

3.5 Dane

Dane wygenerowane zostały za pomocą skryptu *generate_ input.py.* w tabeli 2 przedstawiono przykładowe współrzędne punktów.

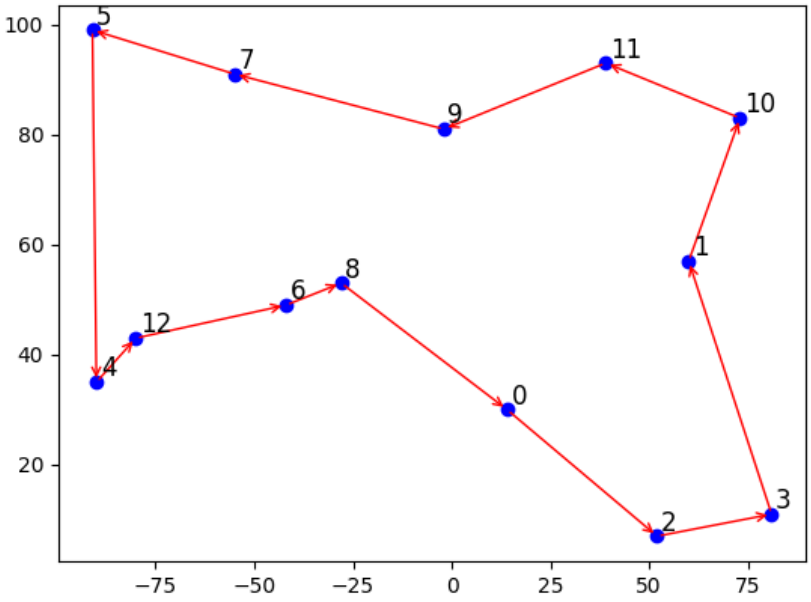
x	y
14	30
60	57
52	7
81	11
-90	35
-91	99
-42	49

Tabela 2: Współrzędne euklidesowe miast

4 Wyniki

4.1 Wizualizacja cyklu

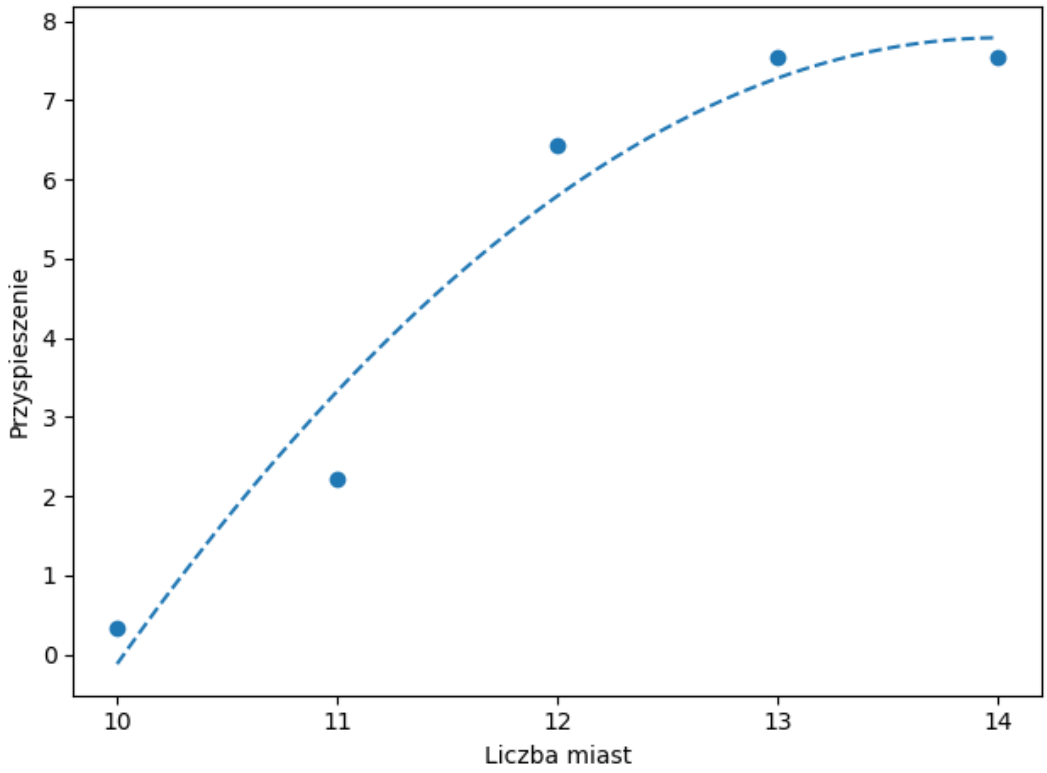
W celu sprawdzenia poprawności algorytmów zwizualizowano najkrótszy cykl. Na rysunku 1 przedstawiono przykładowe rozwiązanie dla 13 miast.



Rysunek 1: Wizualizacja rozwiązania

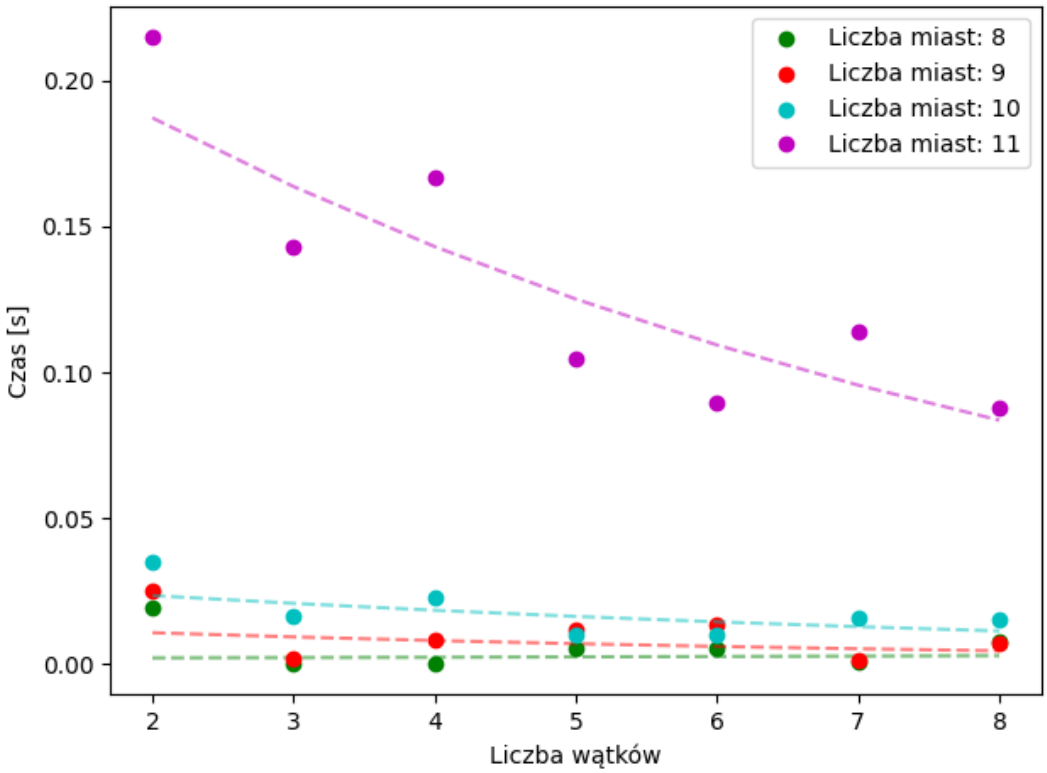
4.2 Wyznaczenie przyspieszenia

W celu wyznaczenia przyspieszenia porównano czasy działania algorytmu sekwencyjnego oraz równoległego dla 6 i 8 procesorów. Wykres 2 przedstawia stosunek czasu obliczeń metody równoległej oraz sekwencyjnej obliczeń dla różnej ilości miast.

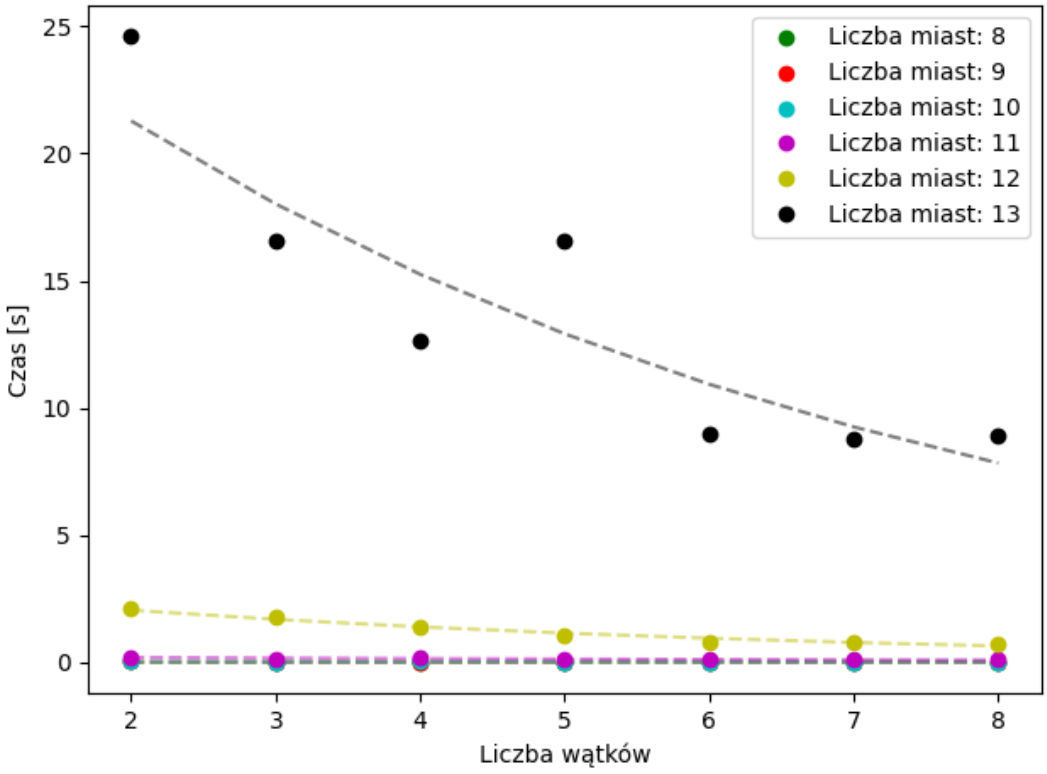


Rysunek 2: Zależność przyspieszenia od ilości miast dla komputera wyposażonego w 8 rdzeni

Dla mniejszej liczby miast przyspieszenie nie było zauważalne ze względu na stosunkowo niski czas obliczeń sekwecyjnych. Dodatkowo na wykresach 3, 4 przedstawiono czasy obliczeń dla różnej ilości miast oraz ilości wątków użytych do obliczeń. Wykres 3 przedstawia czasy dla zakresu miast 8-11. Wykres 4 został uzupełniony o kolejną liczbę miast w celu lepszej wizualizacji spadku czasu.



Rysunek 3: Zależność czasu obliczeń od ilości miast dla różnej ilości wątków



Rysunek 4: Zależność czasu obliczeń od ilości miast dla różnej ilości wątków

Średnie przyspieszenie przedstawiono w tabeli 3.

ilość wątków	średnie przyspieszenie	teoretyczne przyspieszenie
6	5,91	5,99
8	7,60	7,99

Tabela 3: Porównanie teoretycznego przyspieszenia z wyznaczonym

5 Wnioski

1. Dla stosunkowo niedużej liczby miast czas obliczeń równoległych jest większy w porównaniu do czasu obliczeń sekwencyjnych. Związane jest to z czasem włączenia dodatkowych wątków, który jest dłuższy niż czas obliczeń dla niewielkiej liczby miast.

2. Dla 5 wątków obserwowany jest wyższy czas obliczeń niż dla 4 wątków. Może być to spowodowane działaniem pętli *Parrarel.For*, która zmniejsza liczbę wątków przypisaną do zadania,

3. W celu zwiększenia przyspieszania należałoby przypisywać nieobciążonym wątkom nieobliczonych wartości gałęzi drzewa rozwiązań. Dodając do tego funkcję odcięcia przyspieszenie mogłoby wzrosnąć,

4. Metoda zdaję się być skalowalna, ponieważ czasy obliczeń wzrastają w miarę zwiększania rozmiaru problemu N, co jest oczekiwanym zachowaniem,

5. Wyznaczone przyspieszenia z symulacji są porównywalne co do przyspieszeń teoretycznych. Oznacza to, że wzór na teoretyczną wartość przyspieszenia wyznaczony został prawidłowo.

6 Literatura

[1] *Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford* Wprowadzenie do algorytmów