Politechnika Wrocławska

Programowanie efektywnych algorytmów

Problem komiwojażera (TSP)

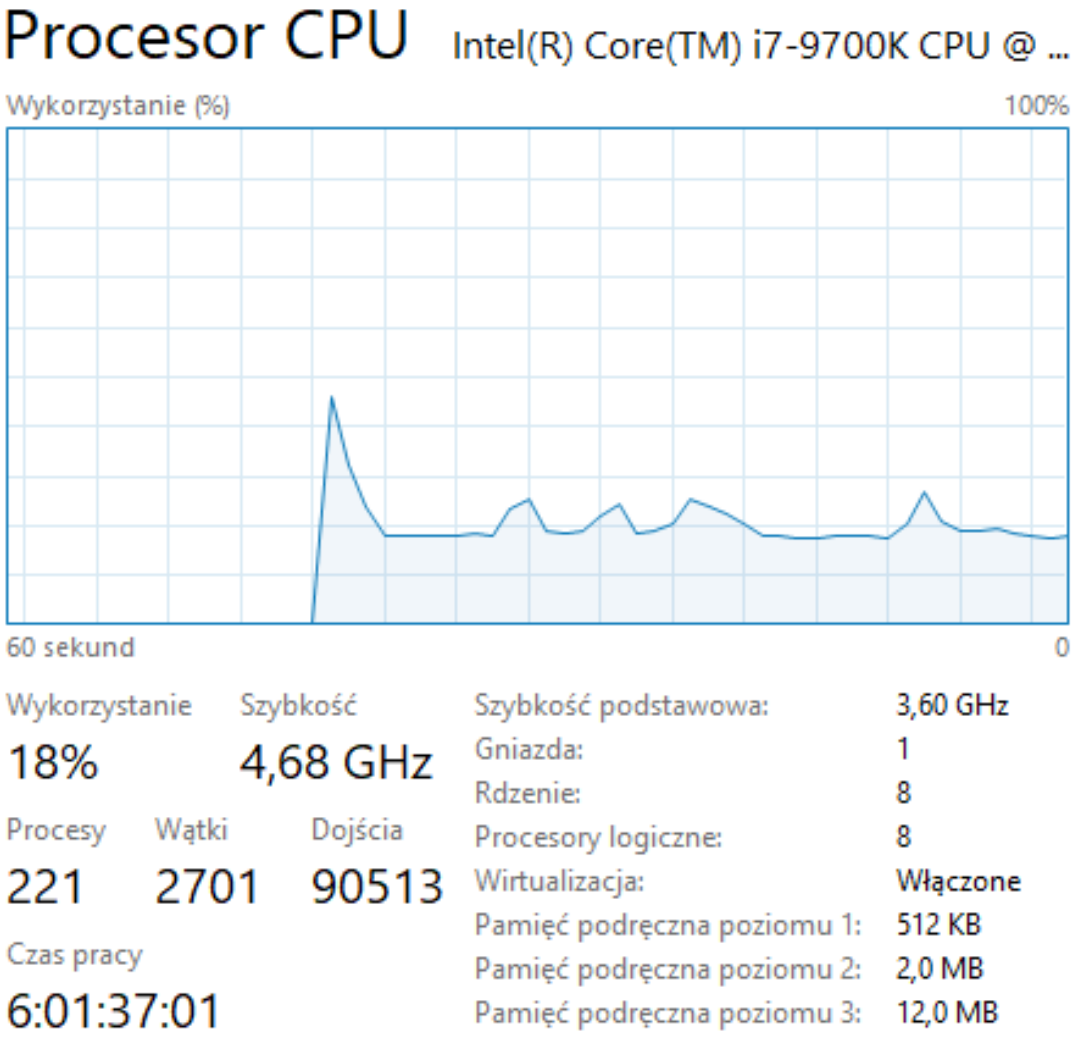
Krzysztof Zalewa

23.1.2025

Spis treści

1 Specyfikacja sprzętu użytego do badań

Badania zostały wykonanie na komputerze stacjonarnym o specyfikacji:
Procesor: Intel(R) Core(TM) i7-9700K CPU
Zegar: 3.60GHz
Wielkość pamięci RAM: 32,0 GB (dostępne: 31,8 GB)



Rysunek 1: Zurzycie procesora w trakcie wykonywania algorytmu Tabu search

2 Instancje użyte w badaniach

Do badań użyłem macierzy z TspLib[4.]
Macierze symetryczne - rat99.tsp,pr152.tsp,ts225.tsp oraz pr264.tsp
Macierze asymetryczne - ftv33.atsp,ftv64.atsp,kro124p.atsp* oraz ftv170.atsp
Gdzie najleprze znane rozwiązania to :

	Symetryczne				Asymetryczne			
Instancje	rat99	pr152	ts225	pr264	ftv33	ftv64	kro124p*	ftv170
	1211	73682	126643	49135	1286	1839	36230	2755

Table 1: Optymalne wyniki z TspLib

* Mimo tego że powinna być to instancja o 124 wierzchołkach po konwersji otrzymuję tylko 100 wierzchołków. Pozostałe instancje konwertują się poprawnie.

3 Tabu search

3.1 Opis Algorytmu

Algorytm tabu search jest algorytmem metahurystycznym służącym do rozwiązywania problemów optymalizacyjnych. Algorytm ten przechowuje część otrzymanych wcześniej rozwiązań w tablicy tabu. Takie podejście sprawia że szansa na ponowne wybranie tego samego rozwiązania znacznie maleje. Sprawia to też że jest mniejsza szansa na utknięcie w pętli. Algorytm ten można jeszcze usprawnić poprzez dodanie warunku krytycznego i strategii dywersyfikacji.
Warunek krytyczny: Jeżeli otrzymana wartość jest mniejsza od najlepszej do tej pory znalezionej wartości ale ścieżka znajduje się w tabu to i tak przypisujemy obecną wartość do najlepszej.
Strategia dywersyfikacji: Jeżeli przez określoną liczbę iteracji algorytmu wartość się nie poprawiła zmieniamy wybraną ścieżkę. W mojej implementacji zmiana ścieżki polega na wylosowaniu nowej przy użyciu funkcji std::shuffle.

3.2 Badanie wpływu metody przeszukiwania sąsiedztwa

W moim algorytmie zaimplementowałem dwie różne metody przeszukiwania sąsiedztwa.

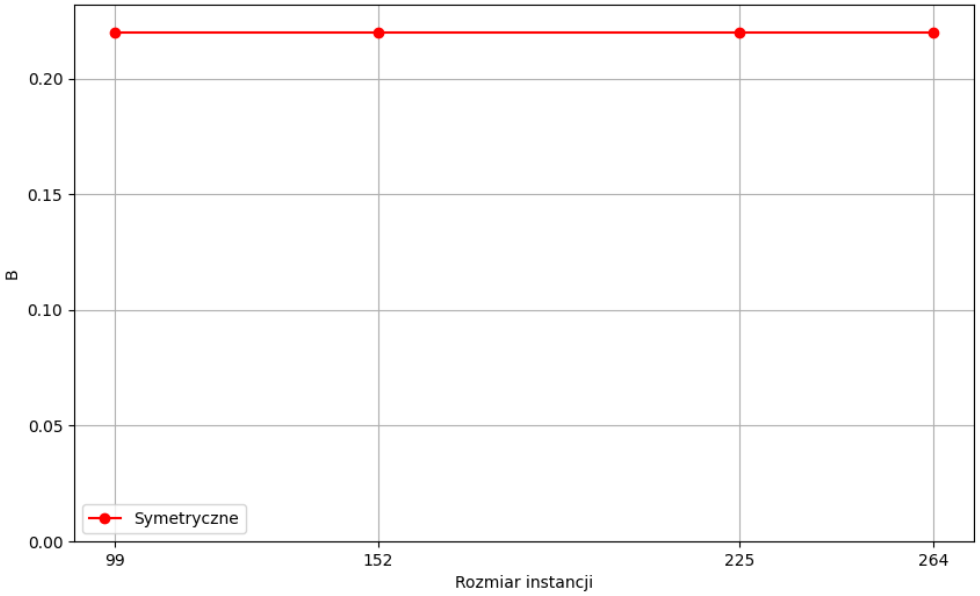
1. Swap - Wybieram dwa wierzchołki i zamienia je ze sobą. W mojej implementacji sprawdzam wszystkie unikalne możliwości więc generuję n*(n-1)/n sąsiadów.

2. Insert - wybieram wierzchołek i miejsce w tablicy. Wierzchołek kopiuje a następnie usuwam z tablicy. Na koniec kopię wierzchołka wstawiam w wybrane miejsce. W mojej implementacji obie wartości losuję. Wykonuję n*(n-1)/n losowań.

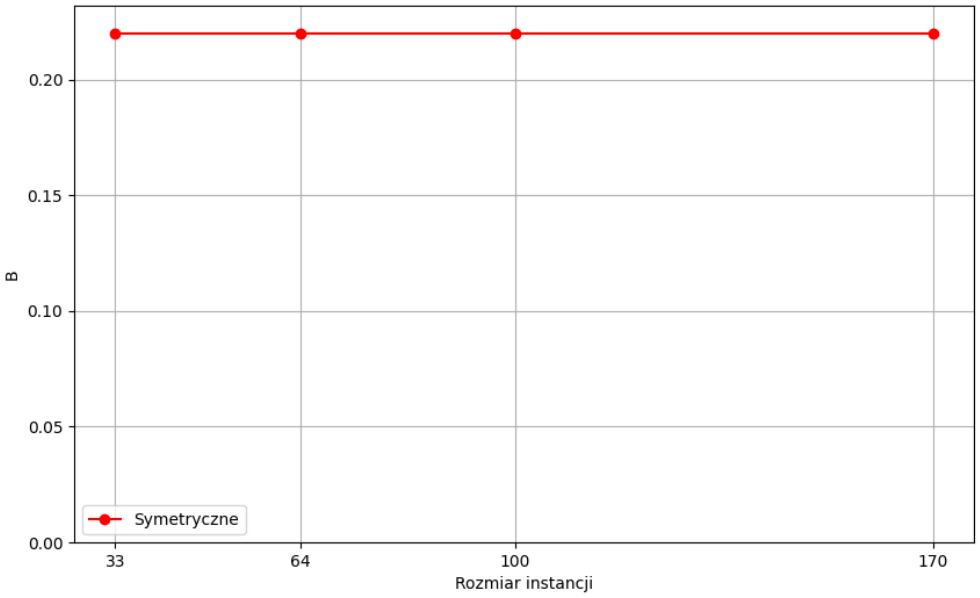
Hipoteza: W tabu search metoda przeszukiwania insert powinna być znacznie lepsza niż swap.
Badania: Wykonuję badania dla każdej z 8 instancji. Żeby otrzymać dobrą próbkę badanie powtarzam cztero krotnie dla każdej instancji. Więc wykonuje (4*ASYM + 4*ASM)*4 * 2 = 64 badań.

	Symetryczne				Asymetryczne			
Rozmiar[Liczba wierzchołków]	99	152	225	264	33	64	100	170
Błąd wzgledny[%]	15.32	16.95	17.42	13.84	18.27	18.76	18.65	18.76

Table 2: Błędy w wynikach algorytmu dla macierzy symetrycznych i niesymetrycznych



Rysunek 2: Wyniki badań dla macierzy symetrycznych



Rysunek 3: Wyniki badań dla macierzy asymetrycznych

Wnioski:

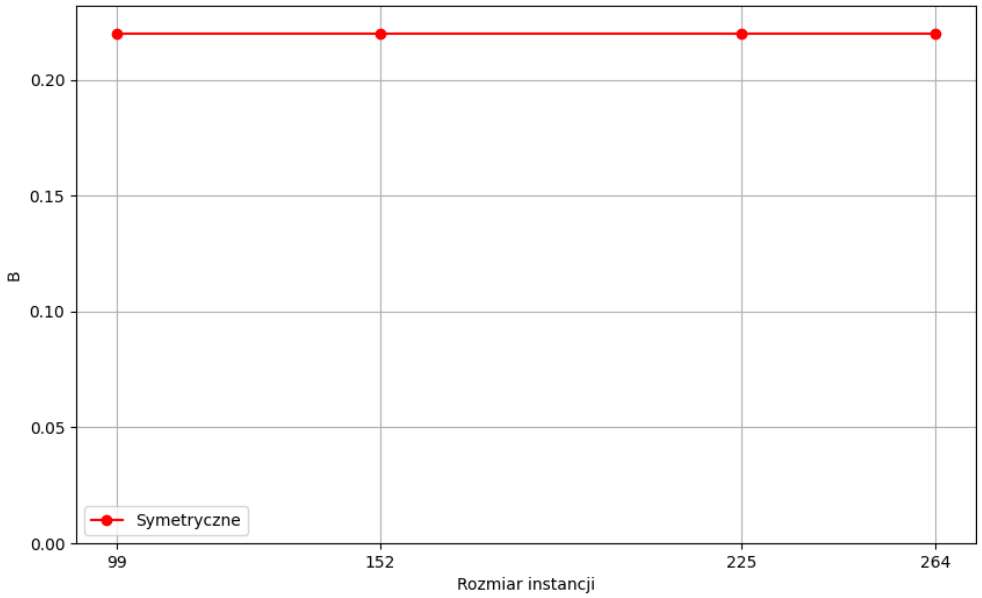
3.3 Badanie wpływu metody tworzenia pierwszego rozwiązania

W moim algorytmie zaimplementowałem dwie różne metody tworzenia pierwszego rozwiązania.

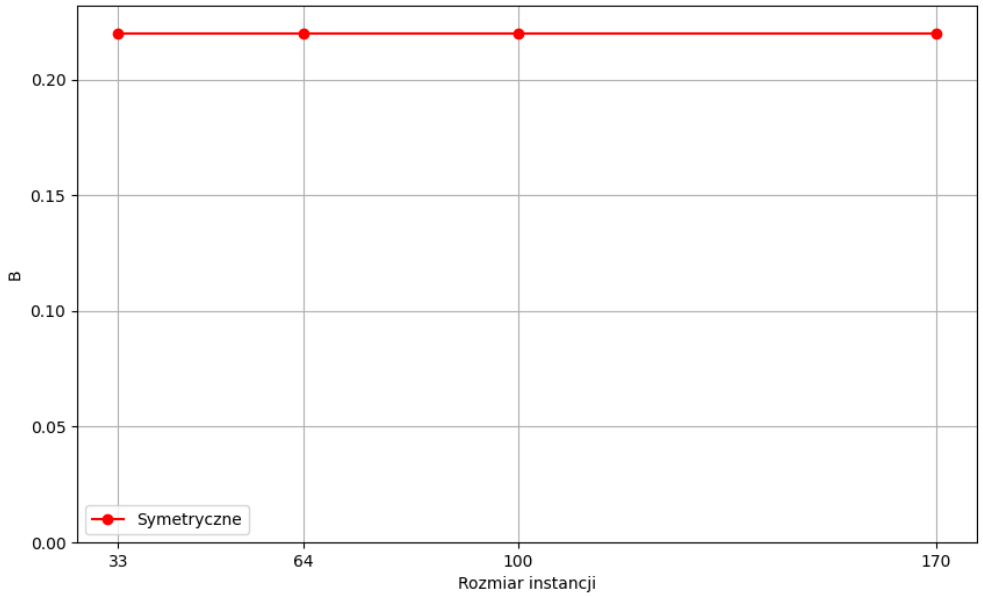
1. Losowa - Pierwsze rozwiązanie jest zupełnie losowe. Najpierw tworzę wektor z liczbami od 0 do n - 1. Następnie na tym wektorze używam funkcji shuffle.

2. Nearest neighbour - Korzystam z wcześniej zaimplementowanego algorytmu NN.

Hipoteza: Wyniki dla metody losowej będą znacznie gorsze (Większy błąd wzgledny) od NN. Ale jest też niewielka szansa na wyniki będące bliżej optymalnego rozwiązania. Jednakże wykonanie wielu pomiarów powinno wykluczyć znaczne odstępstwa.
Badania: Podobnie jak w poprzednim przypadku wykonuje badania dla każdej z 8 instancji po 4 powtórzenia. Więc znowu wykonuje (4*ASYM + 4*ASM)*4 * 2 = 64



Rysunek 4: Wyniki badań dla macierzy symetrycznych



Rysunek 5: Wyniki badań dla macierzy asymetrycznych

Wnioski:

3.4 Badanie wpływu ilości iteracji bez zmian

W mojej implementacji przekroczenie pewnej ilości bez zminay wyniku służy do dywersyfikacji. Do testów wybrałem wartości 25,50,75,100,125.
Hipoteza: Wraz ze wzrostem ilości iteracji bez zmian jakość rozwiązania też będzie rosła.
Badania: Dla każdej instancji testuje wszystkie 5 wartości. Żeby otrzymać dobrą próbkę badanie powtarzam cztero krotnie dla każdej instancji. Dla tego wykonuję (4*ASYM + 4*ASM)*5 * 4 = 160 badań.

Wnioski:

3.5 Badanie wpływu długości tablicy tabu

W mojej implementacji elementy pozostają w tablicy tak długo aż ilość elementów w tej tablicy nie przekroczy pewnej liczby. Długość tablicy zależna jest od ilości wierzchołków (n * mnożnik).Do testów wybrałem wartości 0.5,0.4,0.3,0.2,0.1.
Hipoteza: Jakość rozwiązania powinna rosnąć kiedy długość tabeli tabu maleje. Jednakże zbyt mała wartość spowodować cykliczne powracanie do już wygenerowanych rozwiązań.
Badania: Podobnie jak w ostatnim badaniu dla każdej instancji testuje wszystkie 5 wartości. Żeby otrzymać dobrą próbkę badanie powtarzam cztero krotnie dla każdej instancji. Dla tego wykonuję (4*ASYM + 4*ASM)*5 * 4 = 160 badań.

Wnioski:

3.6 Podsumowanie

3.7 Simulated anealing

3.8 Swap lub Insert

(4*ASYM + 4*ASM)*2 * 5

3.9 NN lub random

(4*ASYM + 4*ASM)*2 * 5

3.10 Długość epoki

(4*ASYM + 4*ASM)*5 * 4

3.11 Wielkość alfa

(4*ASYM + 4*ASM)*5 * 4

3.12 Temperatura startowa

(4*ASYM + 4*ASM)*5 * 4 1

3.13 Podsumowanie

ok 8,5h

3.14 Algorytm mrówkowy

3.15 Opis Algorytmu

Algorytm mrówkwy jest algorytmem metahurystycznym który modeluje zachowanie mrówek. Mrówki w wyruszają z mrowiska w poszukiwaniu jedzenia w losowych kierunkach. Po znalezieniu jedzenia mrówki wracają do mrowiska i pozostawiają za sobą feromony. Ilość zostawionych feromonów zależy od ilości znalezionego jedzenia. Następnie mrówki ponownie wyruszają z mrowiska ścieżki wybierają z pewnym prawdopodobieństwem które zależy od ilości feromonów. Jako że feromony parują to ścieżki nie uczęszczane zanikają a te prowadzące do dobrych rozwiązań są wzmacniane.
W mojej implementacji ścieżki wybierane są z pewnym prawdopodobieństwem wyliczonym ze wzoru:

$$p_{ij}^{k}(t) = \frac{a_{ij}(t)}{\sum l \in N_i a_{il}(t)}$$

Gdzie k to k-ta mrówka ;a i , j to ścieżka z i-tego wierzchołka do j-tego. Natomiast wartości początkowe feromonów szacowane są ze wzoru $\tau_0 = \frac{k}{C^{nn}}$ W tym wzorze C^{nn} to wynik algorytmu NN dla tej instancji.

3.16 Badanie wpływu typu rozkładu feromonów

W mojej implementacji mam dwa typy rozkładu feromonów

1. CAS - Algorytm cykliczny aktualizuje wartości feromonów po wybraniu całej ścieżki. Aktualizacja przebiega według wzoru $\delta\tau_{ij}^{k}(t, t+n) = n/L^{k}$ gdzie L^{k} to długość otrzymanej trasy.

2. QAS - Algorytm ilościowy aktualizuje wartości feromonów po wybraniu krawędzi Aktualizacja przebiega według wzoru $\delta\tau_{ij}^{k}(t, t+n) = n/d_{ij}$ gdzie d_{ij} to koszt przejścia z i do j.

Hipoteza: Algorytm QAS powinien dawać nieznacznie lepsze wyniki.
Badania: Dla obu metod wykonuje badania dla każdej z 8 instancji, powtarzam je cztero krotnie by otrzymać dobrą próbkę. Więc wykonuje (4*ASYM + 4*ASM)*2 * 4 = 64 badań.

Wnioski:

3.17 Badanie wpływu wartości rho

Wartość ρ to współczynnik parowania feromonów. Po tym jak wszystkie mrówki wybrały swoją ścieżkę wartości feromonów są przemnażane przez ρ. Dlatego też wartości ρ powinny być w zakresie 0 do 1. Dlatego wybrałem wartości 0.8,0.6,0.4,0.2
Hipoteza: Wysokie wartości ρ powinny zachęcać mrówki do wybierania różnych nie koniecznie optymalnych ścieżek. Natomiast niskie wartości powinny sprawiać że mrówki będą bardziej zachłannne
Badania: Dla każdej instancji testuje wszystkie 5 wartości. Żeby otrzymać dobrą próbkę badanie powtarzam cztero krotnie dla każdej instancji. Dla tego wykonuję (4*ASYM + 4*ASM)*4 * 4 = 128 badań.

Wnioski:

3.18 Badanie wpływu stosunku alfy do bety

W mojej implementacji używam tablic decyzyjnych gdzie

$$A_i = [a_{ij}(t)]_{Ni}$$

Wartości A_i wyliczam na początku nowego pokolenia. Natomiast elementy a_{ij} wyliczam ze wzoru:

$$a_{ij}t = \frac{[\tau_{ij}(t)]^\alpha * [\tau_{ij}(t)]^\beta}{\sum l \in N_i [\tau_{ij}(t)]^\alpha * [\tau_{ij}(t)]^\beta}$$

Do testów wybrałem 5 wartości $\frac{\alpha}{\beta}$ $\frac{1}{3}, \frac{2}{3}, 1, \frac{3}{2}, 3$
Hipoteza: Zwiększając element α zwiększa się szansa na wybranie ścieżki z dużą ilością feromonów. Natomiast Zwiększając β zwiększa się szansa na wybranie ścieżki z niewielką ilością feromonów. Kiedy $\alpha = \beta$ wynik będzie najgorszy. Ale dla małego α i dużego β powinien być najlepszy.
Badania: Dla każdej instancji testuje wszystkie 5 wartości. Żeby otrzymać dobrą próbkę badanie powtarzam cztero krotnie dla każdej instancji. Dla tego wykonuję (4*ASYM + 4*ASM)*4 * 4 = 128 badań.

Wnioski:

3.19 Podsumowanie

4 Wnioski z całego projektu

5 Źródła

1. https://www.javatpoint.com/what-is-a-tabu-search
2. https://www.geeksforgeeks.org/what-is-tabu-search/
3. https://www.baeldung.com/cs/tabu-search
4. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
5. https://eportal.pwr.edu.pl/pluginfile.php/209250/mod_resource/content/1/w8.pdf
6. https://www.geeksforgeeks.org/introduction-to-ant-colony-optimization/