Projektowanie i Symulacja Algorytmów - Projekt

Rozwiązanie problemu wyznaczania tras pojazdów (VRP) za pomocą algorytmu genetycznego

Jacek Myjkowski 259062
Mateusz Muzioł 259223

31 stycznia 2025

1 Wprowadzenie

Problemem, jakim zajmuje się ten projekt będzie problem wyznaczania tras pojazdów - VRP (Vehicle Routing Problem). Jest to jeden z przypadków rozszerzenia problemu komiwojażera.

VRP znajduje zastosowanie w wielu dziedzinach, między innymi:

- Planowanie dostaw w firmach kurierskich i spożywczych,
- Gospodarka odpadami, czyli planowanie tras wywozu śmieci,
- Dostarczanie przesyłek pocztowych,
- Dystrybucja paliwa na stacje benzynowe,
- Odśnieżanie,
- Logistyka miejska, w tym wykorzystanie zróżnicowanej floty pojazdów,
- Dostawy w czasie rzeczywistym, gdzie ważna jest szybka obsługa zamówień.

1.1 Opis problemu wyznaczania tras pojazdów

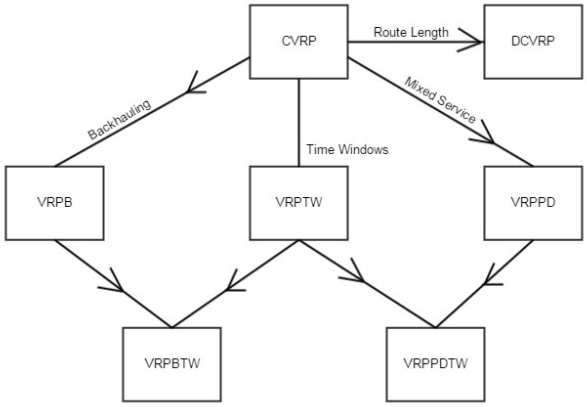
Problem wyznaczania tras pojazdów (VRP) polega na wyznaczeniu optymalnych tras dla floty pojazdów, które muszą dostarczyć towary do wielu miejsc, minimalizując całkowity koszt transportu. Celem jest zminimalizowanie odległości pokonanej przez wszystkie pojazdy.

1.2 Różne rodzaje dodatkowych ograniczeń VRP

VRP może być rozbudowany o dodatkowe ograniczenia, takie jak:

- **Pojemność pojazdów:**
  Każdy pojazd ma ograniczoną pojemność, co oznacza, że może zabrać tylko określoną liczbę ładunków.
- **Okna czasowe:**
  Niektóre miejsca docelowe mogą wymagać dostawy w określonym przedziale czasowym.

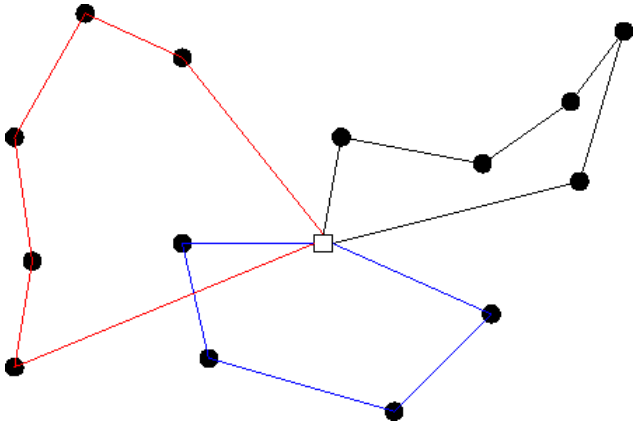
- **Maksymalny czas pracy kierowców:**
  Każdy pojazd (kierowca) może mieć ograniczony czas pracy lub dzienny limit przejechanych kilometrów.
- **Liczba pojazdów:**
  Flota pojazdów może być ograniczona do określonej liczby, co wpływa na planowanie tras.
- **Koszty operacyjne:**
  Różne pojazdy mogą mieć różne koszty operacyjne, takie jak paliwo, opłaty drogowe, itp.
- **Różne typy pojazdów:**
  Flota może składać się z różnych typów pojazdów, które mają różne pojemności i koszty.
- **Ograniczenia drogowe:**
  Niektóre drogi mogą mieć ograniczenia dotyczące wagi lub rozmiaru pojazdów.
- **Priorytety dostaw:**
  Niektóre dostawy mogą mieć wyższy priorytet i muszą być dostarczone wcześniej.



Rysunek 1: Związki pomiędzy poszczególnymi wariantami VRP

1.3 Teoria i złożoność obliczeniowa

VRP jest problemem NP-trudnym [1]. Do jego rozwiązania najlepiej sprawdzają się algorytmy metaheurystyczne, takie jak algorytm genetyczny, przeszukiwanie Tabu, Symulowane wyżarzanie lub ALNS (Adaptive Large Neighborhood Search). Instancję problemu można zwizualizować za pomocą grafu



Rysunek 2: Graf ilistrujący VRP

2 Przegląd literatury

Problem routingu pojazdów (VRP) jest szeroko badanym zagadnieniem z zakresu badań operacyjnych, które koncentruje się na optymalizacji tras dostaw lub odbioru towarów, minimalizując koszty i spełniając określone ograniczenia[5]. W literaturze można znaleźć, że klasyczny VRP zakłada z góry znaną listę klientów oraz ustalone czasy przejazdu i obsługi, a celem jest wyznaczenie najtańszych tras, które odwiedzą każdego klienta tylko raz[4]. Ze względu na NP-trudność problemu VRP, dla dużych instancji stosuje się algorytmy heurystyczne i metaheurystyczne.

Wraz z rozwojem logistyki i wzrostem jej wymagań, klasyczny VRP doczekał się wielu rozszerzeń i wariantów, które lepiej odzwierciedlają realne scenariusze:

- **Pojemnościowy VRP (CVRP)** uwzględnia ograniczoną ładowność pojazdów, co wymusza ich powrót do depotu w celu załadunku.[5]
- **VRP z oknami czasowymi (VRPTW)** dodaje ograniczenia czasowe, w których dostawy muszą być zrealizowane.[5]
- **VRP z odbiorem i dostawą (VRPPD)** obejmuje zarówno dostawy do klientów, jak i odbiór towarów od nich.[5]
- **Dynamiczny VRP (DVRP)** to problem, w którym zlecenia pojawiają się dynamicznie, co wymaga ciągłej reoptymalizacji tras.[5]
- **Feeder Vehicle Routing Problem (FVRP)** wykorzystuje heterogeniczną flotę pojazdów, taką jak ciężarówki i motocykle, co ułatwia dostawy w zatłoczonych obszarach miejskich. Motocykle spotykają się z ciężarówkami w punktach transferowych, aby załadować towar.[5]
- **Real-Time Feeder Vehicle Routing Problem (RTFVRP)** to dynamiczna wersja FVRP, w której zlecenia pojawiają się w czasie rzeczywistym.[5]
- **Vehicle Routing Problem with Workload Balance (WBVRP)** uwzględnia zrównoważenie obciążenia pracą kierowców, minimalizując różnice w trasach pod względem odległości, liczby klientów i ilości dostarczonych towarów.[6]

Cały czas powstają też nowe podejścia do rozwiązywania problemu VRP:

- **Algorytm DIWPSO (Dynamic Inertia Weight Particle Swarm Optimization)**, który dynamicznie dostosowuje wagę inercji cząstek w algorytmie optymalizacji rojem cząstek. DIWPSO jest szczególnie przydatny w dynamicznych problemach routingu, takich jak RTFVRP.[5]
- **Mechanizm "joint" w FVRP**, gdzie motocykle spotykają się z ciężarówkami w punktach połączeń zamiast wracać do magazynu.[5]
- **Reoptymalizacja w czasie rzeczywistym**, która pozwala na dynamiczne dostosowywanie tras w DVRP i RTFVRP.[5]
- **Zastosowanie kar za zmianę trasy**, która wymusza wykorzystanie doświadczenia kierowców z poprzednich tras w dynamicznych modelach RTFVRP.[5]
- **Uwzględnienie obciążenia kierowców (WBVRP)**, które minimalizuje nierówności w obciążeniu pracą.[6]
- **Kompaktowość i atrakcyjność wizualna tras**, co zwiększa akceptację planów przez kierowców i zmniejsza ich dezorientację.[6]

Wśród algorytmów stosowanych w VRP, źródła wymieniają: • Algorytmy dokładne, takie jak branch-and-bound i branch-and-cut, jednak są one ograniczone do rozwiązywania mniejszych instancji problemu. • Algorytmy heurystyczne, w tym tabu search (TS), simulated annealing (SA), algorytmy genetyczne (GA). • Algorytmy metaheurystyczne, takie jak optymalizacja kolonii mrówek (ACO), sieci neuronowe (NN), particle swarm optimization (PSO). • Algorytmy hybrydowe, które łączą różne podejścia, aby poprawić jakość rozwiązania.

3 Model matematyczny problemu

W klasycznym ujęciu problem sformułowany jest w postaci grafu nieskierowanego $\Gamma = (\Psi, \epsilon)$, gdzie Ψ oznacza zbiór wierzchołków, do których przypisane jest zapotrzebowanie, natomiast ϵ zbiór krawędzi, do których przypisane są koszty przewozu ewentualnie czas lub długość trasy.

Minimalizowana jest funkcja

$$minC = \sum_{r \in R} \sum_{f \in \Psi} \sum_{g \in \Psi} c_{fg} x_{fgr}$$

gdzie:
r - wierzchołki pomiędzy, którymi odbywa się przewóz,- pojazd należący do zbioru jednorodnych (identycznych) pojazdów R,
f, g - wierzchołki pomiędzy, którymi odbywa się przewóz,
c_{fg} - koszt przewozu pomiędzy wierzchołkami f i g,
x_{fgr} - zmienna binarna określająca, czy pomiędzy wierzchołkami f i g pojazd r wykonuje przewóz.

4 Organizacja pracy

Wykres Gantta

Zadanie	Tydzień rozpoczęcia	Czas trwania	Tydzień zakończenia	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Opracowanie tematu projektu	1	1	1															
Przegląd literatury	2	2	3															
Implementacja	4	4	7															
Walidacja Implementacji	8	1	8															
Poprawa implementacji	9	3	11															
Testy implementacji	12	1	12															
Opracowanie wniosków	13	1	13															
Sporządzenie sprawozdania	13	2	14															

Rysunek 3: Wykres Gantta przedstawiający harmonogram prac nad projektem

Środowisko pracy

Wymagania

- python

Środowisko wirtualne

```
# Utworzenie środowiska, aktywacja i instalacja potrzebnych paczek
python3 -m venv venv
source venv/bin/activate
python -m pip install -r requirements.txt

# Dezaktywacja środowiska wirtualnego
deactivate
```

5 Wykorzystane algorytmy i środowisko badawcze

5.1 Algorytmy

W projekcie zestawiono ze sobą trzy algorytmy, przy czym skupiono się na tym najbardziej skomplikowanym - genetycznym. Dokonany został jego fine-tuning aby działał efektywnie i skutecznie.

5.1.1 Przegląd zupełny

Algorytm możliwy do zastosowania dla małej liczby celów. Polega na przejrzeniu wszystkich dostępnych opcji. Daje gwarancję znalezienia najlepszego rozwiązania przy bardzo dużym koszcie obliczeniowym.

5.1.2 Losowego przeszukiwania

5.1.3 Algorytm genetyczny

Reprezentacja chromosomu:
Każdy chromosom reprezentuje możliwe przypisanie klientów do tras pojazdów (kolejność odwiedzanych miejsc).

Ocena fitness:
Funkcja oceny może obejmować całkowitą pokonaną odległość, czas dostawy, koszty paliwa, liczbę pojazdów użytych itp.

Krzyżowanie i mutacja:
Tworzenie nowych rozwiązań poprzez wymianę części tras między pojazdami lub wprowadzanie drobnych zmian w kolejności obsługi klientów.

Selekcja:
Najlepsze trasy przechodzą do kolejnych iteracji, co zbliża algorytm do optymalnych rozwiązań.

5.2 Implementacja algorytmu genetycznego

Implementacja algorytmu genetycznego składa się z następujących etapów:

1. **Inicjalizacja populacji:**
   - Funkcja: create_initial_population
   - Opis: Generuje początkową populację tras poprzez losowe przetasowanie wierzchołków i podzielenie ich między dostępne pojazdy.
   - Szczegóły: Wierzchołki grafu (bez wierzchołka bazowego 'A') są losowo mieszane i dzielone na trasy dla każdego pojazdu. Powstaje lista tras dla całej populacji.
2. **Ocena populacji:**
   - Funkcja: evaluate_population
   - Opis: Oblicza całkowity koszt tras dla każdego osobnika w populacji i sortuje populację na podstawie wyników fitness (całkowity koszt).
   - Szczegóły: Dla każdej trasy w populacji obliczany jest koszt całkowity przy użyciu funkcji calculate_route_cost. Populacja jest sortowana według kosztów, aby wybrać najlepsze trasy.
3. **Selekcja turniejowa:**
   - Funkcja: tournament_selection
   - Opis: Wybiera podzbiór populacji losowo i zwraca osobnika z najlepszym wynikiem fitness.
   - Szczegóły: Losowo wybierany jest podzbiór populacji, a następnie wybierany jest osobnik z najlepszym wynikiem fitness z tego podzbioru.
4. **Krzyżowanie:**
   - Funkcja: crossover
   - Opis: Łączy trasy dwóch rodziców, aby stworzyć dwie nowe trasy potomne. Wykorzystuje funkcję order_crossover do przeprowadzenia krzyżowania.
   - Szczegóły: Trasy dwóch rodziców są łączone, aby stworzyć nowe trasy potomne. Funkcja order_crossover zapewnia, że nowe trasy zawierają unikalne wierzchołki.

5. **Mutacja:**
    - Funkcja: mutate
    - Opis: Wprowadza losowe zmiany w chromosomie poprzez zamianę dwóch wierzchołków, aby wprowadzić różnorodność.
    - Szczegóły: Z losowym prawdopodobieństwem dwa wierzchołki w trasie są zamieniane miejscami, aby wprowadzić różnorodność do populacji.
6. **Algorytm genetyczny:**
    - Funkcja: genetic_algorithm
    - Opis: Inicjalizuje populację, ocenia ją, a następnie iteracyjnie wykonuje selekcję, krzyżowanie i mutację, aby ewoluować populację w kierunku lepszych rozwiązań.
    - Szczegóły: Algorytm rozpoczyna się od inicjalizacji populacji, następnie ocenia populację, a w każdej iteracji wykonuje selekcję, krzyżowanie i mutację, aby poprawić populację. Proces ten jest powtarzany przez określoną liczbę generacji.

5.3 Środowisko badawcze

Środowisko badawcze zostało przygotowane w języku Python

5.3.1 Generowanie grafów

Do implementacji algorytmów będziemy wykorzystywać syntetycznie generowane grafy nie-skierowane.

5.3.2 Przechowywanie wyników

Wyniki działania algorytmu zapisywane są w folderze *results* w formacie JSON:

```
[
  {
    "name":"<FILENAME_WITH_GRAPH_STRUCTURE>",
    "nodes_count": "<NODES_IN_GRAPH>",
    "edges_count": "<EDGES_IN_GRAPH>",
    "vehicles_amounts": [
      {
        "vehicles_amount": "<AMOUNT_OF_VEHICLES>",
        "execution_time": "<EXECUTION_TIME>",
        "best_routes": [
          [
            "<LIST_OF_NODES_FOR VEHICLE_1>"
          ],
          [
            "<LIST_OF_NODES_FOR_VEHICLE_2>"
          ], [...]
      }, {...}
    ]
  },
  {...}, ...
]
```

Każda ścieżka zaczyna się i kończy w wierzchołku A, który uznawany jest za bazę.

6 Plan badań

Celem projektu jest zbadanie różnicy w czasie i jakości otrzymywanego rozwiązania przy użyciu trzech algorytmów do rozwiązywania VRP.

Z racji iż algorytm przeglądu zupełnego oraz algorytm przeszukiwania losowego nie pozwalają się sparametryzować w takim stopniu jak algorytm genetyczny, to właśnie na nim skupimy całą swoją uwagę, a dwa pierwsze algorytmy posłużą jako odnośniki.

W algorytmie genetycznym będziemy badać wpływ takich jego parametrów na osiągi jak:

- Wielkość populacji
- Liczba generacji
- Wskaźnik(prawdopodobieństwo) mutacji
- Wielkość selekcji

Dla każdego z tych parametrów przyjmiemy kilka wartości, aby sprawdzić jaka jest ich tendencja i jaki jest wpływ na wynik. W jednym teście zmieniać się będzie tylko jeden parametr, a reszta będzie miała wartości domyślne.

Dodatkowo w każdym teście mierzymy czas, aby móc sprawdzić wpływ tego parametru na złożoność obliczeniową algorytmu, a przez to na bezpośredni czas wykonania.

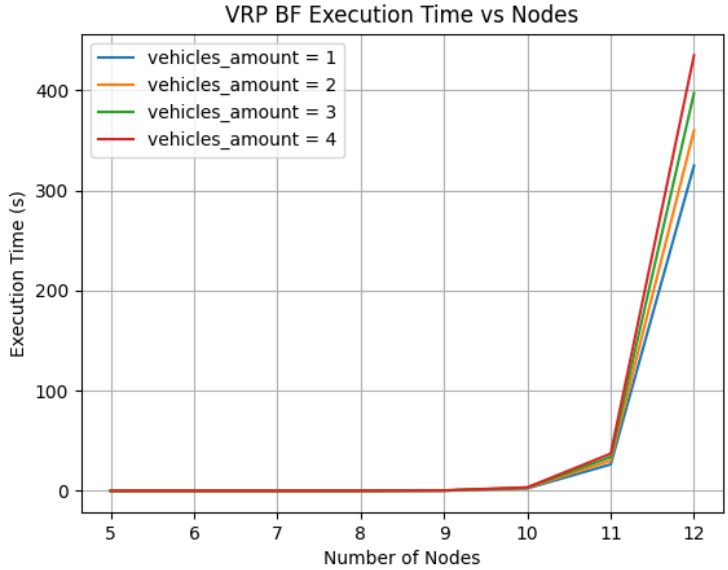
Każdy algorytm będzie wykonany 10 krotnie dla każdej z wartości testowanego aktualnie parametru, a następnie wyciągnięte zostaną uśrednione wyniki.

7 Badania

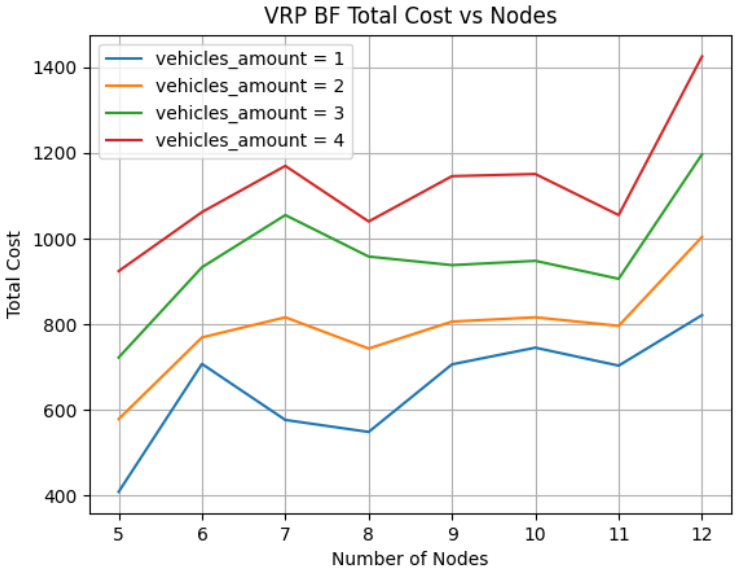
W tym rozdziale przedstawiono wyniki pomiarów czasu wykonania algorytmów przeglądu zupełnego, przeszukiwania losowego oraz algorytmu genetycznego w zależności od różnych parametrów.

Przegląd zupełny

Dla przeglądu zupełnego jedynym parametrem jest liczba pojazdów, dla której sprawdzana jest optymalna ścieżka. Poniższe wykresy przedstawiają zależności czasu wykonania algorytmu oraz najkrótszą znalezioną ścieżkę od ilości wierzchołków w grafie.



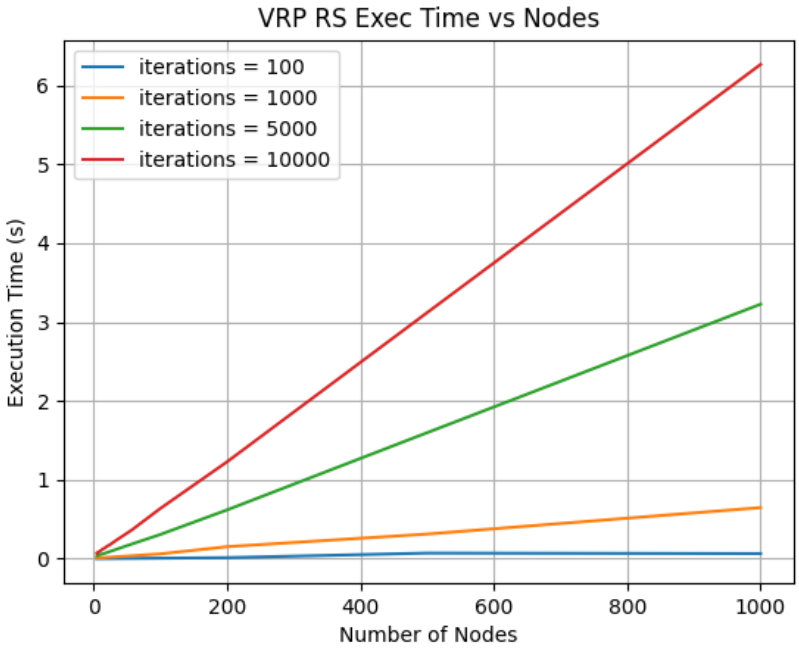
Rysunek 4: Wykres czasu wykonania od liczby węzłów w zależności od liczby pojazdów



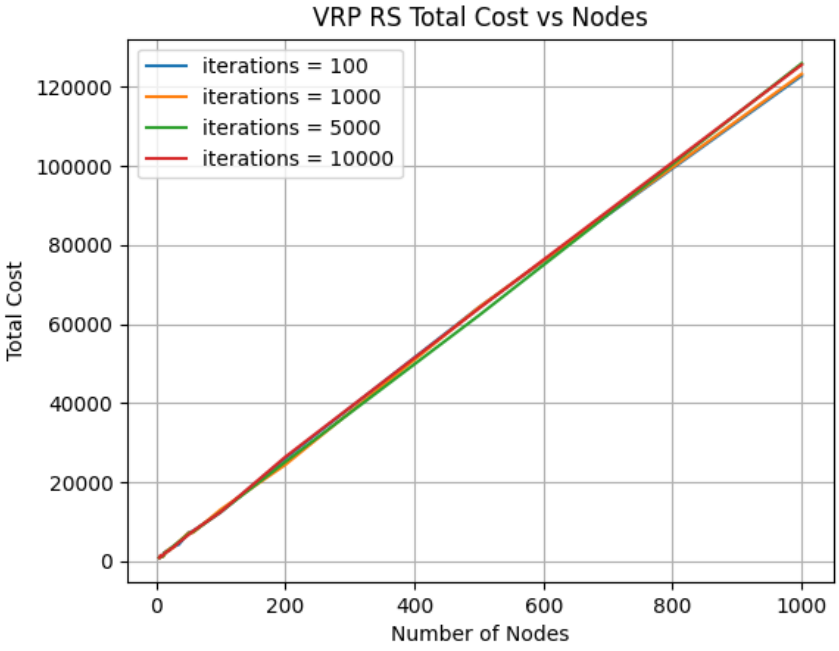
Rysunek 5: Wykres najkrótszej ścieżki od liczby węzłów w zależności od liczby pojazdów

Przeszukiwanie losowe

Przeszukiwanie losowe zostało wykonane dla 4 pojazdów. Poniższe wyniki przedstawiają czasy wykonania algorytmu i najkrótsze znalezione ścieżki od liczby węzłów i zależą od liczby iteracji algorytmu.



Rysunek 6: Wykres czasu wykonania od liczby węzłów w zależności od liczby iteracji

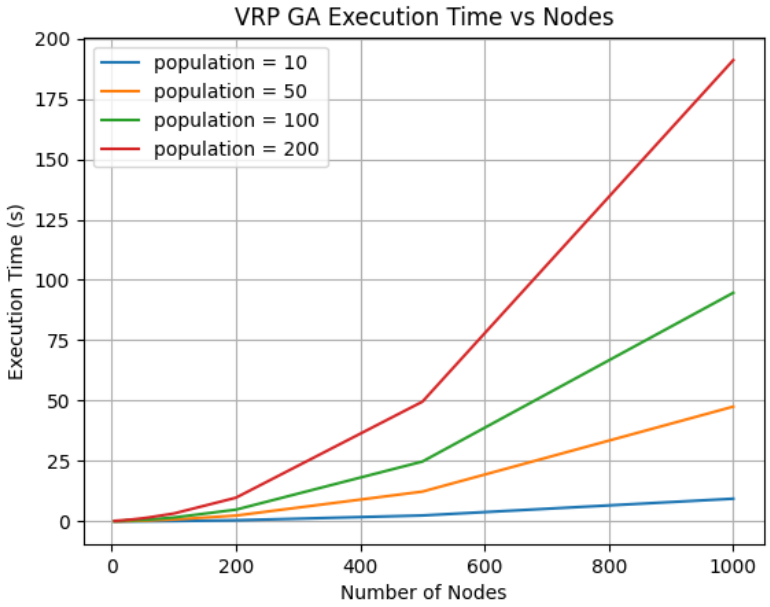


Rysunek 7: Wykres najkrótszej ścieżki od liczby węzłów w zależności od liczby iteracji

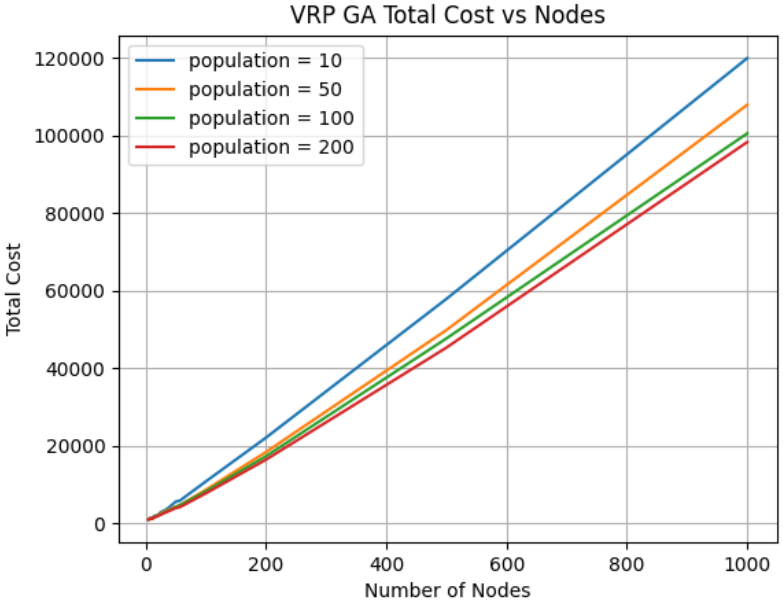
Algorytm genetyczny - fine tuning

Dla algorytmu genetycznego zostało sprawdzonych więcej parametrów takich jak wielkość populacji, liczba generacji, wskaźnik mutacji oraz wielkość selekcji. Wszystkie pomiary zostały wykonane dla 4 pojazdów.

Wielkość populacji

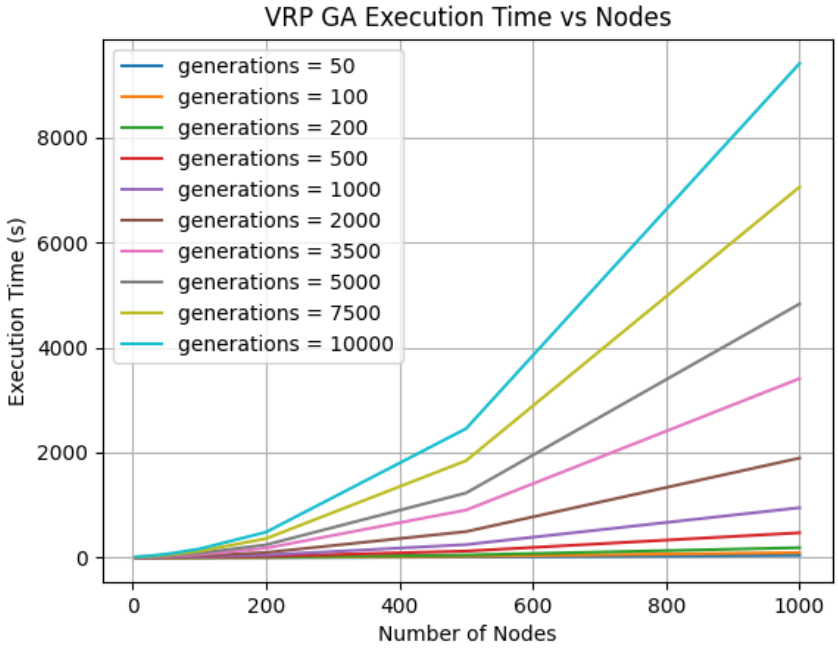


Rysunek 8: Wykres czasu wykonania od liczby węzłów w zależności od wielkości populacji

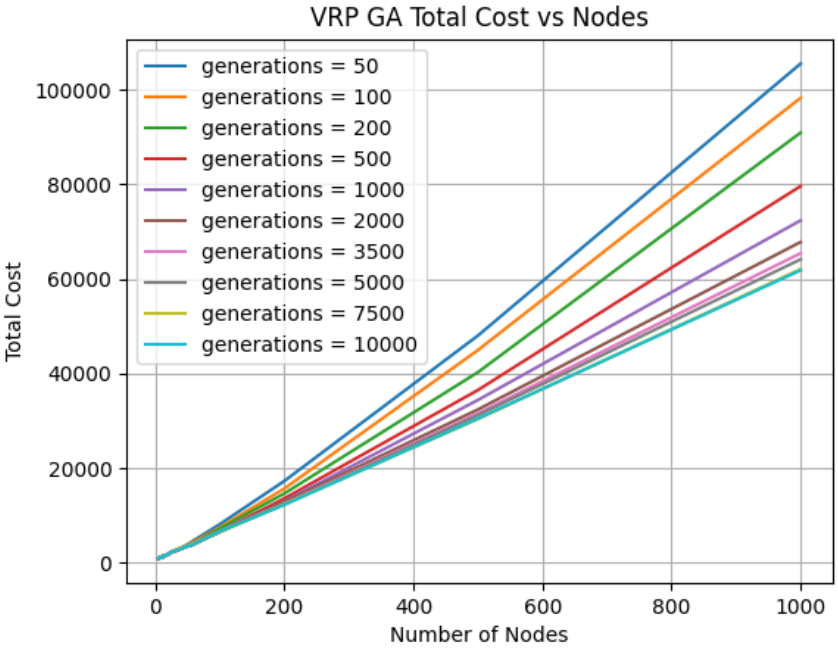


Rysunek 9: Wykres najkrótszej ścieżki od liczby węzłów w zależności od wielkości populacji

Liczba generacji

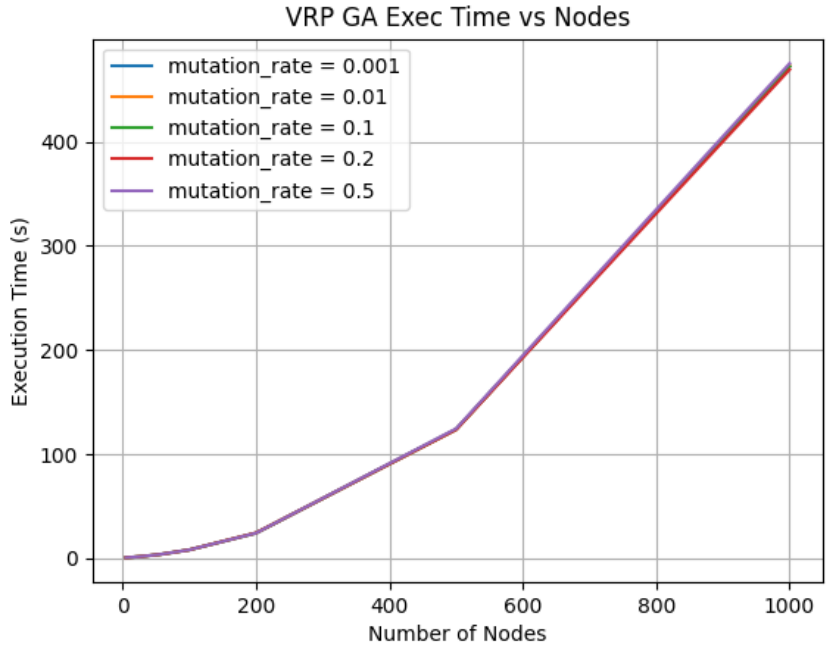


Rysunek 10: Wykres czasu wykonania od liczby węzłów w zależności od liczby generacji

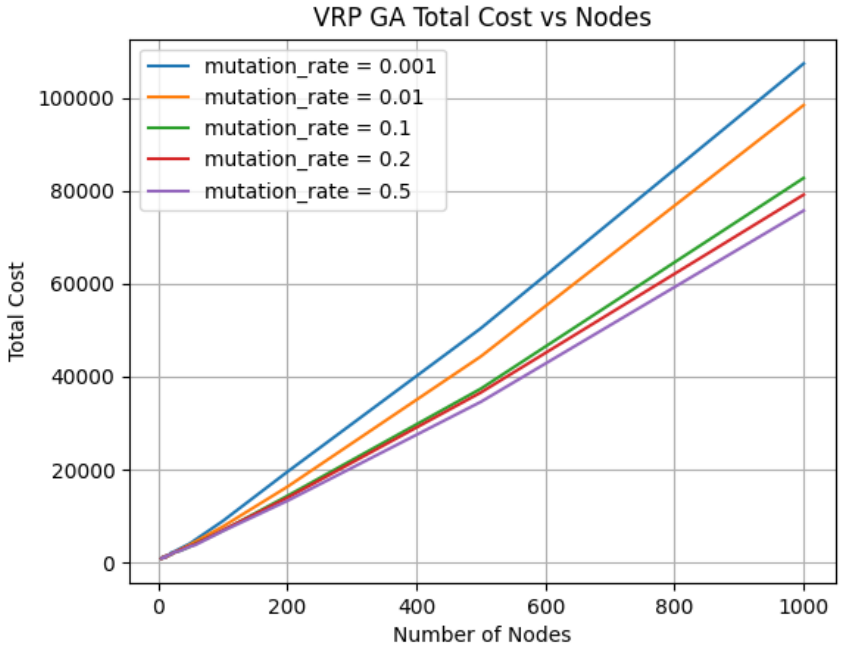


Rysunek 11: Wykres najkrótszej ścieżki od liczby węzłów w zależności od liczby generacji

Wskaźnik mutacji

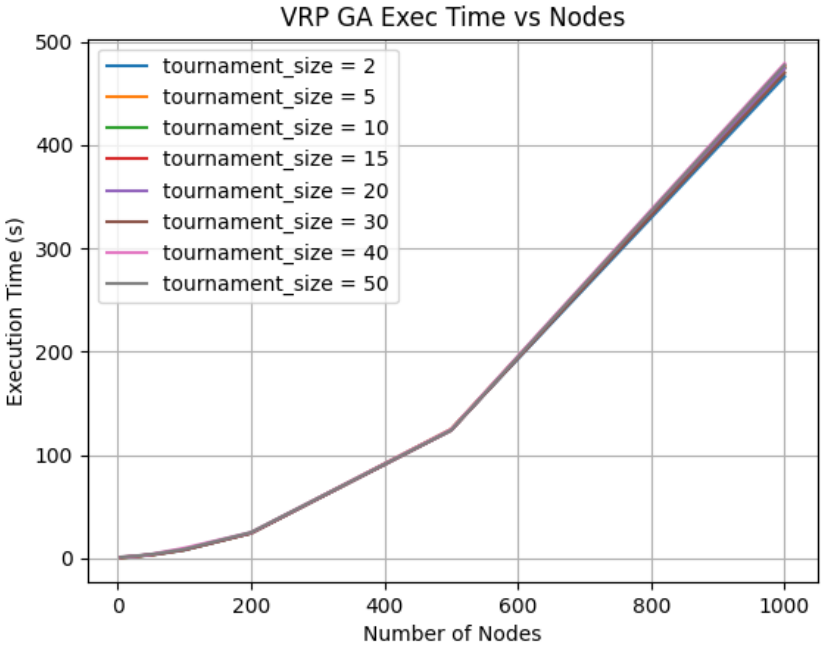


Rysunek 12: Wykres czasu wykonania od liczby węzłów w zależności od wskaźnika mutacji

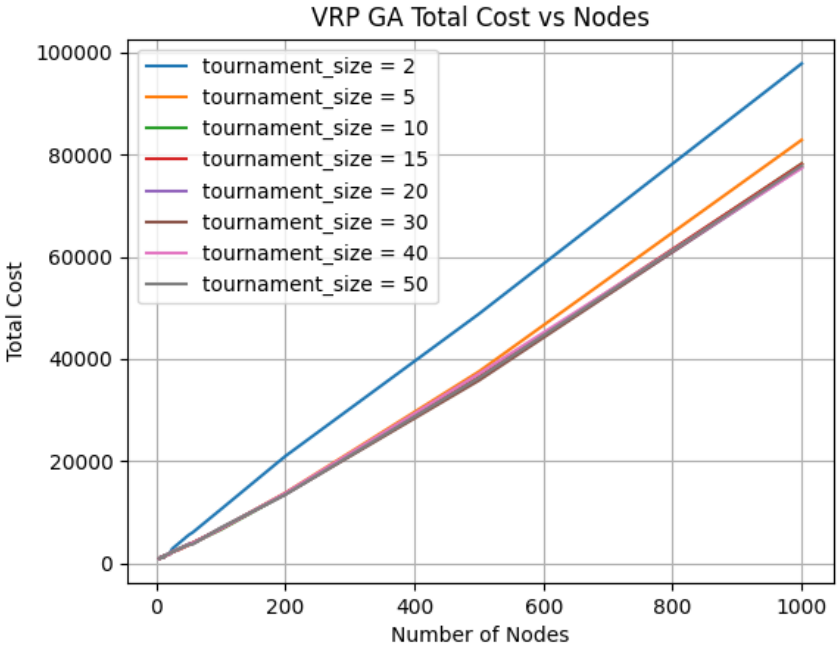


Rysunek 13: Wykres najkrótszej ścieżki od liczby węzłów w zależności od wskaźnika mutacji

Wielkość selekcji



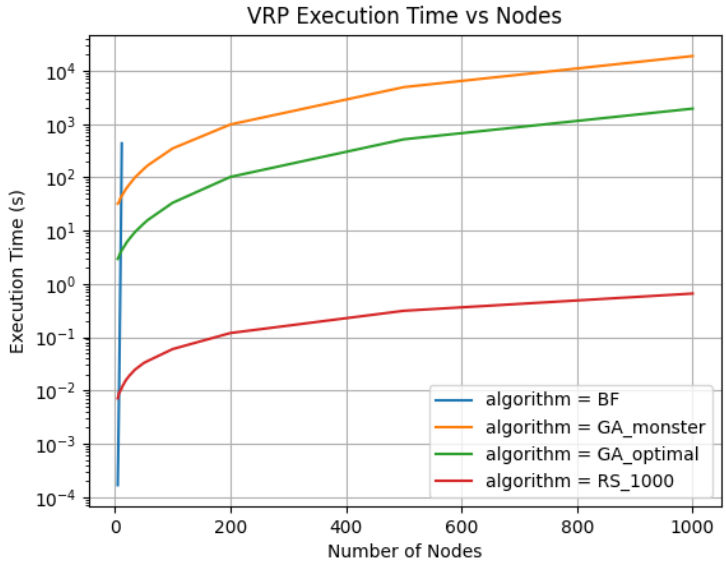
Rysunek 14: Wykres czasu wykonania od liczby węzłów w zależności od wielkości selekcji



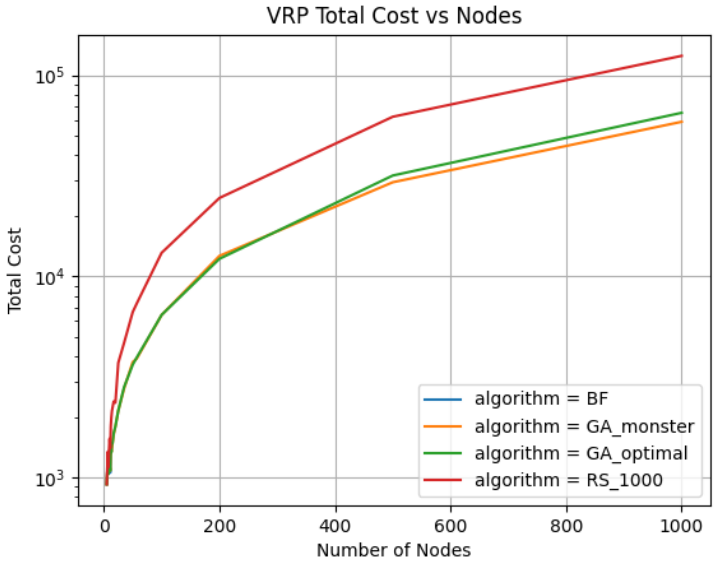
Rysunek 15: Wykres najkrótszej ścieżki od liczby węzłów w zależności od wielkości selekcji

Algorytm genetyczny - porównanie

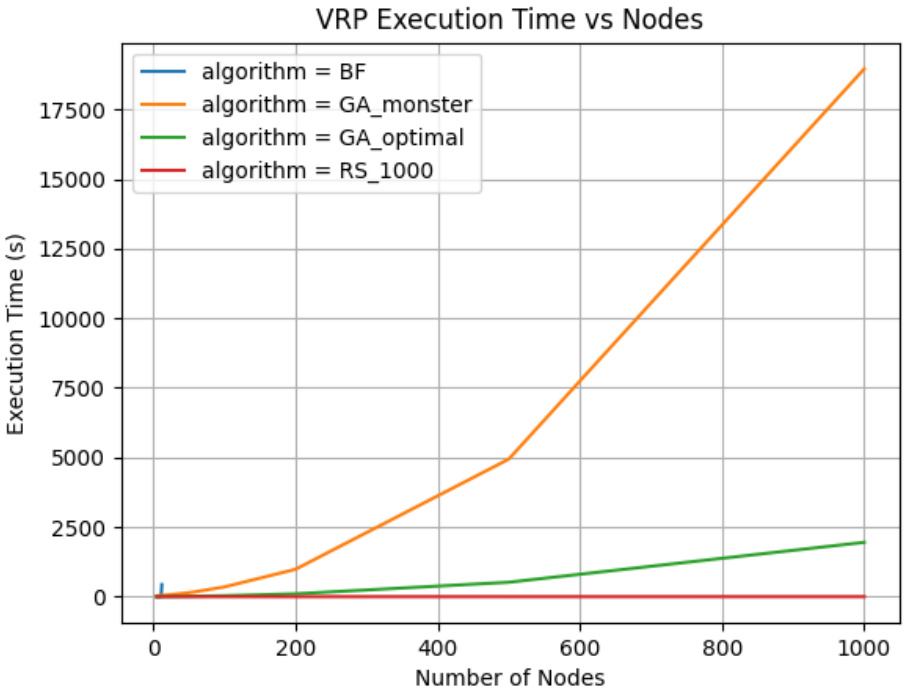
Na podstawie powyższych wykresów znalezione zostały takie wartości parametrów algorytmu, aby uzyskać jak najlepszą jakość rozwiązania przy minimalizacji czasu wykonania. Podczas zwiększania wartości parametrów osiągany jest moment, w którym czas wykonania algorytmu dalej rośnie, a jakość rozwiązania zostaje na tym samym poziomie. Na poniższych wykresach porównane zostały dwa algorytmy genetyczne - jeden dostrojony na osiągnięcie jak najlepszego rozwiązania bez zwracania uwagi na czas wykonania, drugi zaś zachowujący jak najlepszy współczynnik czas wykonania/jakość. Dzięki takiemu porównaniu możemy zobrazować sobie, na które parametry algorytmu powinniśmy zwracać największą uwagę. Na wykresach zamieszczono również wyniki działania algorytmów przeglądu zupełnego i losowego jako referencję.



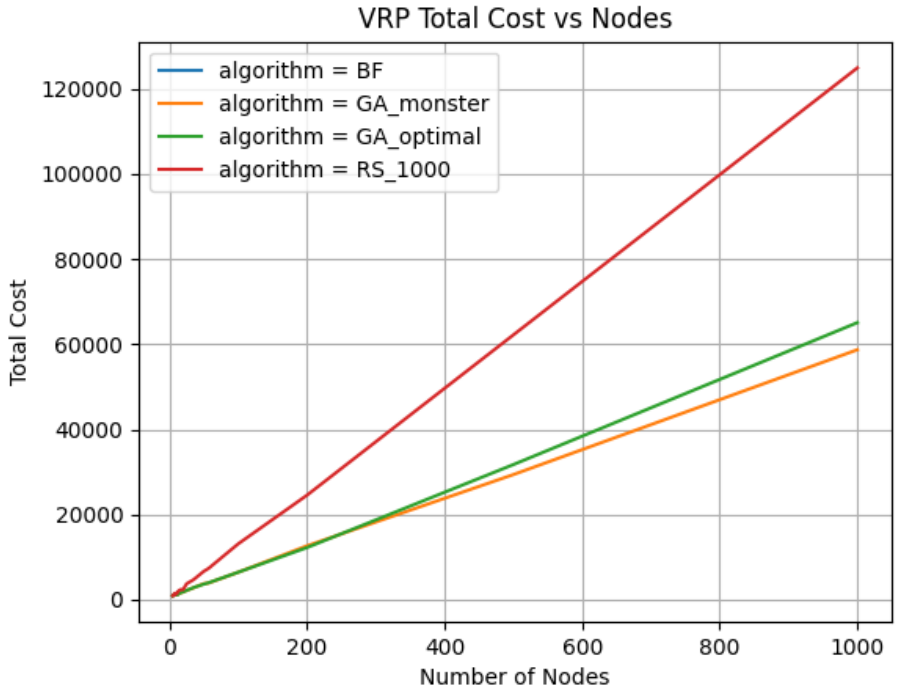
Rysunek 16: Wykres czasu wykonania od liczby węzłów dla badanych algorytmów - skala logarytmiczna



Rysunek 17: Wykres najkrótszej ścieżki od liczby węzłów dla badanych algorytmów - skala logarytmiczna



Rysunek 18: Wykres czasu wykonania od liczby węzłów dla badanych algorytmów - skala liniowa



Rysunek 19: Wykres najkrótszej ścieżki od liczby węzłów dla badanych algorytmów - skala liniowa

8 Wnioski

Analizując wyniki pomiarów pierwszą obserwacją jest fakt, że przegląd zupełny nie nadaje się do rozwiązywania instancji problemu większych niż 10 wierzchołków.

Przeszukiwanie losowe wypadło zaskakująco dobrze. Patrząc na wykresy możemy dojść do wniosku, że narzut czasowy spowodowany zwiększeniem liczby iteracji jest zbyt duży względem jakości wyniku, który wcale się nie poprawia. Dla 100 iteracji czas wykonania jest bardzo krótki, a jakość wyniku praktycznie taka sama jak dla większych ilości iteracji. Nie jest ona aż tak dobra jak dla algorytmu genetycznego z odpowiednimi parametrami, lecz czas wykonania algorytmu jest nieporównywalnie krótszy.

Algorytm genetyczny został sprawdzony dla różnych wartości parametrów takich jak wielkość populacji, liczba generacji, współczynnik mutacji oraz wielkość selekcji. Aby zwiększyć czytelność wyników został on sprawdzony tylko dla 4 pojazdów. We wszystkich sprawdzonych przypadkach im większa wartość danych parametrów, tym lepsza jakość rozwiązania. Wielkość populacji i liczba generacji ma wprost proporcjonalny wpływ na złożoność obliczeniową i czas wykonania algorytmu. Z drugiej strony, zwiększanie wielkości populacji niekoniecznie poprawia jakość wyniku, a cały czas zwiększa złożoność obliczeniową. Przeanalizowaliśmy również wyniki dla tych wartości parametrów, które dadzą nam najlepszy wynik, bez zważania na czas wykonania. Na podstawie wyników zbadaliśmy również taki wariant algorytmu, który zapewni nam najlepszy stosunek jakość/czas. W tym celu zostały wybrane takie wartości parametrów, dla których wykres najkrótszej ścieżki zaczynał się wypłaszczać.

Na końcu porównane zostały wszystkie cztery algorytmy - przegląd zupełny, przeszukiwanie losowe oraz maksymalnie "dokręcony" algorytm i zoptymalizowany algorytm genetyczny. Wykresy z wynikami zostały przedstawione na dwa sposoby - z osią Y w skali logarytmicznej oraz liniowej. Z uwagi na możliwość sprawdzenia algorytmu przeglądu zupełnego tylko dla 12 wierzchołków, a co za tym idzie, stosunkowo niskich wartości najkrótszej ścieżki w porównaniu do pozostałych algorytmów, daje nam to możliwość zobrazowania jak przegląd zupełny wypada w porównaniu z algorytmami przeszukiwania losowego oraz genetycznym. Na wykresach widać, że oba warianty "genetyka" dają nam porównywalnie dobrą jakość rozwiązania, lecz wariant "dokręcony" znajduje je w nieporównywalnie dłuższym czasie. Algorytm przeszukiwania losowego daje nam dwukrotnie gorsze rozwiązania, lecz robi to w czasie krótszym o rzędy wielkości. Algorytm przeglądu zupełnego kompletnie nie nadaje się dla instancji problemu większych niż kilkanaście wierzchołków.

Podsumowując, jeżeli zależy nam na jakości rozwiązania, a czas i wykorzystanie zasobów obliczeniowych gra drugorzędną rolę, algorytm genetyczny będzie bardzo dobrym wyborem. Zakładając, że w prawdziwym świecie instancje tego konkretnego problemu nie będą aż tak duże, dobrze zoptymalizowany algorytm genetyczny pozwoli nam na znalezienie dobrego jakościowo rozwiązania w rozsądnym czasie. Przeszukiwanie losowe działa bardzo szybko, lecz biorąc pod uwagę niską jakość znajdowanego przez nie rozwiązania, zysk na czasie nie jest tego warty. Przegląd zupełny natomiast nada się tylko i wyłącznie dla bardzo niewielkich instancji problemu, gwarantując nam znalezienie najlepszego możliwego rozwiązania.

9 Literatura

[1] Claudia Archetti et al. “Complexity of the VRP and SDVRP”. In: *Transportation Research Part C: Emerging Technologies* 19.5 (2011). Freight Transportation and Logistics (selected papers from ODYSSEUS 2009 - the 4th International Workshop on Freight Transportation and Logistics), pp. 741–750. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2009.12.006. URL: https://www.sciencedirect.com/science/article/pii/S0968090X0900151X.

[2] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. “The vehicle routing problem: State of the art classification and review”. In: *Computers Industrial Engineering* 99 (2016), pp. 300–313. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2015.12.007. URL: https://www.sciencedirect.com/science/article/pii/S0360835215004775.

[3] Bruno Scalia C. F. Leite. *The Vehicle Routing Problem: Exact and Heuristic Solutions. Understand how to solve complex routing problems with Python.* https://towardsdatascience.com/the-vehicle-routing-problem-exact-and-heuristic-solutions-c411c0f4d734. dostęp z dnia: 12.2024. 2023.

[4] KHAIRUDDIN OMAR MOURAD ZIROUR LIONG CHOONG YEUN WAN ROSMANIRA ISMAIL. “EHICLE ROUTING PROBLEM: MODELS AND SOLUTIONS”. In: *ournal of Quality Measurement and Analysis* 4 (2008), pp. 205–218.

[5] J. Behnamian M. Salehi Sarbijan. “A mathematical model and metaheuristic approach to solve the real-time feeder vehicle routing problem”. In: *Computers Industrial Engineering* 185 (2023).

[6] Fernando Yáñez-Concha Rodrigo Linfati and John Willmer Escobar. “Mathematical Models for the Vehicle Routing Problem by Considering Balancing Load and Customer Compactness”. In: *Sustainability* 14 (2022).

[7] *Vehicle routing problem - Wikipedia — en.wikipedia.org.* https://en.wikipedia.org/wiki/Vehicle_routing_problem. dostęp z dnia: 12.2024.