



**Politechnika
Śląska**

PRACA MAGISTERSKA

„Rozwiązanie problemu Cargo Theft VRP z użyciem algorytmów
heurystycznych”

inż. Jakub Kaniowski

Nr albumu 281314

Kierunek: Informatyka

Specjalność: Oprogramowanie systemowe

PROWADZĄCY PRACĘ

Dr inż. Jacek Widuch

KATEDRA ALGORYTMIKI I OPROGRAMOWANIA

Wydział Automatyki Elektroniki i Informatyki

GLIWICE 2023

Tytuł pracy:

„Rozwiązanie problemu Cargo Theft VRP z użyciem algorytmów heurystycznych”

Streszczenie:

Niniejsza praca dotyczy tematyki algorytmów heurystycznych i wykorzystania ich na przykładzie problemu optymalizacyjnego trasowania pojazdów w wariacie z występującym ryzykiem kradzieży tzw. Cargo Theft Vehicle Routing Problem (CTVRP).

Celem pracy jest zapoznanie się z istniejącymi wybranymi algorytmami heurystycznymi poprzez dostosowanie ich implementacji dla wskazanego problemu. Następnie należy przeprowadzić badania eksperymentalne.

Słowa kluczowe:

Algorytmy, heurystyki, problem marszrutyzacji pojazdów, problem komiwożera

Thesis title:

„Solving the Cargo Theft VRP using heuristic algorithms”

Abstract:

This master's thesis concerns the topic of heuristic algorithms and their utilization in the context of the Cargo Theft Vehicle Routing Problem (CTVRP), which involves the risk of theft.

The aim of this work is to become familiar with selected existing heuristic algorithms by adapting their implementations to the specified problem. Subsequently, experimental research is to be conducted.

Keywords:

Cargo theft vehicle routing problem, VRP, optimization problem, algorithms, heuristic

Spis treści

Rozdział 1	Wstęp.....	9
Rozdział 2	Analiza tematu.....	11
2.1.	Problematyka wyznaczania tras pojazdów	11
2.2.	Podstawowe definicje i pojęcia	13
2.2.1.	Problem marszrutyżacji pojazdów (ang. Vehicle Routing Problem)	13
2.2.2.	Pojęcia uzupełniające	14
2.2.3.	Problem komiwojażera (ang. Travelling Salesman Problem).....	15
2.2.4.	Problem chińskiego listonosza	16
2.2.5.	Definicja problemu marszrutyżacji pojazdów i warianty rozszerzające	17
2.3.	Rozwiązywanie problem trasowania pojazdów	21
2.3.1.	Rozwiązanie problemu	21
2.3.2.	Algorytmy heurystyczne – zasada działania	22
2.3.3.	Algorytm Przeszukiwania lokalnego	25
2.3.4.	Algorytm Symulowanego wyżarzania	26
2.3.5.	Algorytm Przeszukiwania tabu	28
2.3.6.	Pozostałe algorytmy	30
2.4.	Problematyka VRP w ujęciu światowym	31
Rozdział 3	Przedmiot pracy.....	33
3.1.1.	Główny program – Solver	33
3.1.2.	Ogólna zasada działania programu Solver	39
3.2.	Szczegóły implementacyjne	40
3.2.1.	Sposoby generacji nowych rozwiązań	41
3.2.2.	Implementacja algorytmu przeszukiwania lokalnego	43
3.2.3.	Implementacja algorytmu symulowanego wyżarzania	43
3.2.4.	Implementacja algorytmu przeszukiwania tabu	44
Rozdział 4	Badania eksperymentalne.....	46
4.1.	Przedmiot badań	46
4.2.	Opis środowiska badawczego.....	47
4.3.	Zbiór danych i założenia przyjęte do badań	47
4.3.1.	Założenia	48
4.3.2.	Przygotowanie zbioru danych dla wariantu Cargo Theft.....	49
4.4.	Walidacja algorytmów.....	49
4.4.1.	Program referencyjny HeuristicLab Optimizer	50

4.4.2.	Walidacja algorytmu lokalnego przeszukiwania	50
4.4.3.	Walidacja algorytmu symulowanego wyżarzania	52
4.4.4.	Walidacja algorytmu przeszukiwania tabu	54
4.5.	Przeprowadzenie badań	56
4.5.1.	Wyznaczanie współczynnika schładzania dla algorytmu symulowanego wyżarzania	57
4.5.2.	Rodzaj algorytmu, a zestaw parametrów	57
4.5.3.	Zmiana współczynnika alfa na końcowe wyniki działania algorytmu symulowanego wyżarzania	58
4.6.	Uzyskane wyniki	58
4.6.1.	Wyniki badania zależności zmian współczynnika alfa na końcowe wyniki działania algorytmu symulowanego wyżarzania	58
4.6.2.	Wyniki przeprowadzonych głównych badań	60
4.6.3.	Analiza wyników Serii I	61
4.6.4.	Analiza wyników Serii II	62
4.6.5.	Analiza wyników Serii III	63
4.6.6.	Graficzna reprezentacja wyników – interpretacja	64
4.6.7.	Jakość optymalizacji	72
Rozdział 5	Podsumowanie	74
Bibliografia	76
Spis skrótów i symboli	78
Lista dodatkowych plików, uzupełniających tekst pracy	79
Spis rysunków	80
Spis tabel	82

Rozdział 1

Wstęp

Rozwiązywanie problemu Cargo Theft Vehicle Routing Problem (w skrócie CTVRP), należy do problemów dotyczących marszrutyzacji pojazdów, ze ściśle określonymi ograniczeniami, takimi jak ilość pojazdów we flocie i ich maksymalna ładowność, której nie można przekroczyć. Ideą optymalizacji tego problemu, jest minimalizacja kosztów całkowitych podróży, składających się z kosztu pokonania całej trasy (głównie odległości) oraz wyjątkowym dla tego problemu – ryzyka kradzieży towarów, które pojawia się dla każdego z odwiedzanych przez pojazd lokalizacji, które zostały wyznaczone przez algorytm dokonujący optymalizacji. Dziedzina poruszanego problemu wywodzi się z popularnych problemów optymalizacyjnych - problemu komiwojażera oraz chińskiego listonosza.

Algorytmami umożliwiającymi rozwiązywanie tego typu problemów, są z całą pewnością algorytmy heurystyczne, które pozwalają na uzyskanie rozwiązania względnie dobrego, lecz niekoniecznie optymalnego spośród dziedziny wszystkich rozwiązań. Dlatego też, w niniejszej pracy, zostanie poruszona problematyka dotycząca trasowania pojazdów i sposoby ich rozwiązywania wybranymi algorytmami za pomocą heurystyk. Zostaną przeprowadzone badania mające na celu sprawdzić otrzymywaną jakość rozwiązań dla problemu CTVRP, w zależności od wykorzystanych w tym celu trzech algorytmów – lokalnego przeszukiwania, symulowanego wyżarzania oraz przeszukiwania tabu.

Celem niniejszej pracy jest implementacja wspomnianych trzech algorytmów heurystycznych, zdolnych do rozwiązywania problemu CTVRP zgodnych z narzucanymi przez ten wariant ograniczeniami. Wykorzystując zaimplementowane algorytmy w postaci aplikacji komputerowej, zostaną przeprowadzone eksperymentalne badania, mające ocenić efektywność wykonanych implementacji. Przed przystąpieniem do właściwej części badań, powinna nastąpić weryfikacja poprawności działania tychże algorytmów. Do tego celu posłużą zbiory danych przygotowane do problemu Capacited Vehicle Routing Problem (w skrócie CVRP), który jest zbliżonym wariantem do opisywanego CTVRP. Taka decyzja

jest podyktowana faktem, iż zbiory danych dla wariantu CVRP są ściśle określone – tzn. posiadają znalezione optimum, co znacząco uprości proces ewentualnego poszukiwania błędów w implementowanych algorytmach.

Niniejsza praca składa się z pięciu rozdziałów. Rozdział 2 porusza kwestie związane z wprowadzeniem do tematyki trasowania pojazdów jak i również zagadnień związanych z wykorzystanymi algorytmami heurystycznymi i ich wariantów. Ponadto w omawianym rozdziale, znajduje się przegląd literatury w postaci zastosowań problematyki marszrutyzacji pojazdów w sytuacjach rzeczywistych i płynących z tego tytułu korzyści. Rozdział 3 zawiera szczegóły implementacyjne programu wykonanego na potrzeby przeprowadzenia badań. Ponadto opisano sposób korzystania z aplikacji wraz z opisem plików wejściowych (akceptowanych przez program) jak i plików wyjściowych zawierających wygenerowane rozwiązania. Rozdział 4 dotyczy wszelkich kwestii związanych z przygotowaniem oraz przeprowadzeniem badań, łącznie z interpretacją pozyskanych wyników. Z kolei Rozdział 5, dotyczy ogólnego podsumowania pracy i skonkretyzowanych wniosków.

Rozdział 2

Analiza tematu

Tematyka problemu trasowania pojazdów jest dziedziną rozwiązującą problemy logistyczne, niemniej jednak jest to o tyle interesujący przypadek, że łączy on wiele dziedzin nauki. Można więc powiedzieć, że jest to problem wymagający interdyscyplinarnego podejścia. Logistyka definiuje problem, natomiast matematyka i współczesna algorytmika dostarczają rozwiązanie.

Rozwiązywanie problemów marszrutyzacji pojazdów, przynosi ze sobą wiele korzyści z uwagi na charakter optymalizacyjny. Stosowane jest głównie w firmach transportowych, które szukają redukcji kosztów prowadzenia floty pojazdów, minimalizacji zużycia zasobów firmy (jak np. paliwo) oraz minimalizacji ryzyka napotykanego w ramach działalności takiego przedsiębiorstwa. Poszukiwane jest zazwyczaj rozwiązanie optymalne, o ile możliwe jest jego szybkie rozwiązanie. Wszelkie kwestie związane z podstawowymi pojęciami jak i również z rozwiązywaniem problemów marszrutyzacji zostaną omówione w niniejszym rozdziale.

2.1. Problematyka wyznaczania tras pojazdów

Poruszanie się eksploracja jest czynnikiem nieustająco napędzającym rozwój naszej cywilizacji. Pokonywanie odległości, zmiana lokalizacji była naturalnym elementem życia pradawnych ludów koczowniczych, którzy aby przetrwać musieli wędrować. Obecnemu człowiekowi w drodze rozwoju i ewolucji, dość intuicyjnie przychodzi wyznaczenie trasy na bieżąco – np. wycieczki, czy też codziennej trasy do sklepu. W gruncie rzeczy można podjąć taki ciąg decyzji, które sprawiają, że podróż potrwa jak najmniej czasu oraz wysiłek fizyczny będzie niewielki (np. chodzenie po płaskim terenie). Oczywiście nie musi to być regułą. Można podjąć szereg chybionych decyzji przez co

np. podróż wydłuży się i nie będzie optymalna. Dlatego też, z biegiem czasu i towarzyszącym temu rozwojem, zaczęto interesować się planowaniem podróży w taki sposób, aby było to efektywne. Już w starożytności człowiek planował trasy pokonywane przez karawany, tak aby były one względnie bezpieczne dla towaru jak i zwierząt, które były wykorzystywane w celach transportowych. Nikt nie wysłałby transportu dóbr i towarów bez zaznajomienia się z terenem, dogodnymi miejscami na odpoczynek i zagrożeniami panującymi na szlaku, gdyż wiązałoby się to z licznymi stratami – krótko mówiąc podróż nie byłaby optymalna.

Żyjąc obecnie w XXI wieku mimo tylu minionych lat i epok, wojen, kataklizmów, nadal można spotkać się z pojęciem karawany. Współcześnie za taką karawanę można uznać transport samochodowy, a ponieważ żyjemy w czasach ciągłej optymalizacji w myśl szeroko pojętego kapitalizmu (*minimalizacja kosztów, maksymalizacją zysków*) – postanowiono na tyle ile to możliwe minimalizować koszty transportu na świecie (obecnie poza minimalizacją kosztów, dochodzi również minimalizacja zanieczyszczeń środowiska). Właśnie to, jest jednym z głównych czynników jakimi zajmuje się współczesna logistyka. Z biegiem lat udało się zdefiniować problematykę w rozumieniu matematycznym, co pozwoliło na wyszukiwanie rozwiązania w sposób metodyczny i ściśle określony (zdefiniowany). W wyniku postępującej cyfryzacji, stało się możliwe optymalizowanie rozwiązań za pomocą komputera. Powstał więc nowy kierunek badań z dziedziny algorytmiki, dotyczący optymalizacji ruchu pojazdów. Zajmuje się on szeroko pojętą definicją problemów optymalizacyjnych (tj. proces przekształcenia problemu rzeczywistego do modelu teoretycznego) i badaniem sposobów uzyskiwania rozwiązań.

Pierwsze wzmianki dot. rozważań matematycznych na tematy optymalizacji tras pojazdów, pojawiły się już w latach czterdziestych ubiegłego wieku (np. Problem Komiwojażera), tak rozwój półprzewodników i ogólny wzrost mocy obliczeniowej komputerów, dopiero w latach dziewięćdziesiątych sprawił, że problem optymalizacji (marszrutyzacji pojazdów) zyskał szersze zainteresowanie pośród badaczy, a ich odkrycia, wpłynęły na sposób zarządzania transportem. Obecna problematyka wyznaczania tras pojazdów, potrafi być wyjątkowo dostosowana do rzeczywistego problemu/warunków stojących przed planistą, stąd zadania dot. marszrutyzacji pojazdów (z ang. Vehicle Routing Problem – VRP) posiadają wiele wariantów, poruszających zarówno kwestie techniczne, (ograniczeń fizycznych transportu – jak np. rozmiary, masa ładunków, ograniczenia ładowności pojazdów) jak i kwestie organizacyjne (czas pracy kierowcy, czas otwarcia punktów docelowych, czy np. statystyki dot. wypadków, kradzieży na danych odcinkach dróg). Dziedzina problemów VRP jest z powodzeniem wykorzystywana w systemie dostaw towarów między węzłami (tj. magazynem a klientami). Warto wiedzieć, że istnieje również problem tzw. chińskiego listonosza (ang. Chinese postman problem), który jest pokrewnym zagadnieniem związanym w tym przypadku z problemem ARP (ang. Arc Routing

Problem) [1]. Jest on szczególnie przydatny w problematyce dowozu poczty, usług oczyszczania miasta, czy w odsnieżaniu ulic. Różnice merytoryczne pomiędzy tymi zadaniami zostaną poruszone w punktach 2.2.3, 2.2.4 oraz 2.2.1.

Poruszany w tytule pracy magisterskiej problem trasowania pojazdów w wariancie Cargo-Theft stanowi uzupełnienie jednego z podstawowych wariacji problemów marszrutyzacji pojazdów – Capacited – VRP.

Wszystkie rodzaje problemów łączy jeden wspólny mianownik, a jest nim – wyznaczenie rozwiązania akceptowalnego – tzn. takiego, w którym w skończonym czasie wielomianowym, można otrzymać rozwiązanie satysfakcjonujące, bliskie optymalnemu. Za takie rozwiązania odpowiedzialna jest algorytmika. Zadania z dziedziny optymalizacji dobrze jest rozwiązywać w sposób metodyczny i zorganizowany, z punktu widzenia informatyki – do tego właśnie służą algorytmy, które umożliwiają wygenerowanie rozwiązania na podstawie zdefiniowanego zbioru danych.

Warto dodać że w ostatniej dekadzie poza wzrostem mocy obliczeniowej, dochodzi do jednoczesnej miniaturyzacji komputerów, co pozwoliło na otwarcie nowej ścieżki badań – dot. wyznaczania tras w czasie rzeczywistym. Jest to szczególnie przydatne w dziedzinie autonomii ruchu, stąd też dziedzina ta nie tylko jest wykorzystywana stricte jako narzędzie wykorzystywane przez logistykę, lecz również z dużym powodzeniem służy w robotyce mobilnej. Dziedzina ta wspomagana sieciami neuronowymi (tj. sztuczną inteligencją) pozwala na tworzenie pojazdów, potrafiących na podstawie danych z czujników otoczenia zinterpretować środowisko i poprowadzić pojazd tak, aby ominąć wszelkie przeszkody i zagrożenia. Powyższe osiągnięcia techniki nie byłyby możliwe, bez podstawowej wiedzy z algorytmiki i teorii danych oraz pierwotnych definicji problemów dot. wyznaczania tras.

2.2. Podstawowe definicje i pojęcia

2.2.1. Problem marszrutyzacji pojazdów (ang. Vehicle Routing Problem)

Problem marszrutyzacji pojazdów można zdefiniować jako problem decyzyjny, dotyczący optymalizacji tras przejazdu dla pewnej określonej liczby (floty) pojazdów, którą zazwyczaj określa się mianem homogenicznej (wariant głównie rozpatrywany) – mając do czynienia z identycznymi pojazdami wchodzącymi w skład floty, lub niehomogenicznej – w tym przypadku pojazdy (ich masa, ładowność, spalanie paliwa) mogą się różnić od siebie.

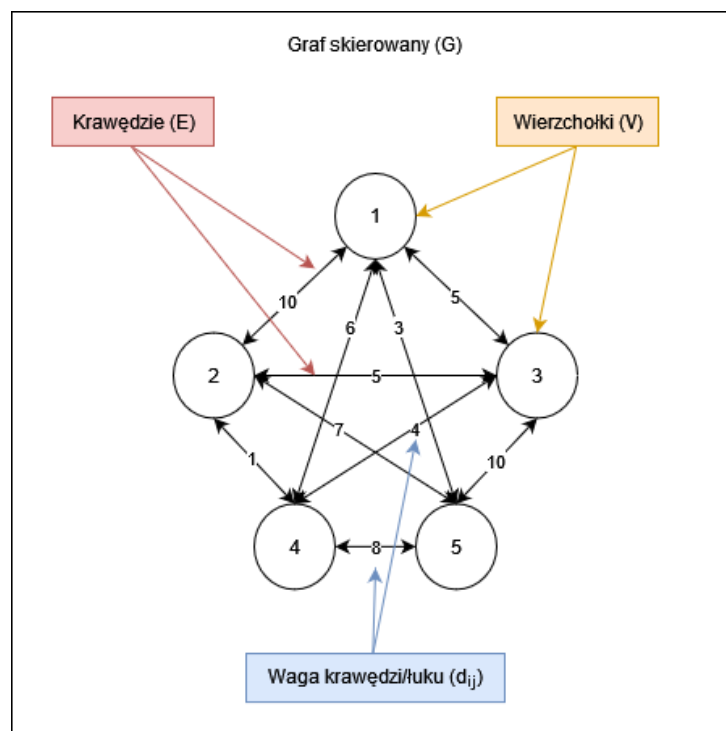
Problem VRP charakteryzuje się również koniecznością jednorazowego odwiedzenia wszystkich punktów (lokacji), przy czym zazwyczaj punktem dopuszczającym wyruszenie

w podróż jak i powrót stanowi magazyn (ang. Depot). Optymalizacja polegająca na minimalizacji kosztów całkowitych podróży odbywa się pod pewnymi warunkami i ograniczeniami, które są specyficzne dla każdego z wielu wariantów VRP (opisanych w punkcie 2.2.5). Otrzymane rozwiązanie jest zbiorem tras dla każdego z pojazdów w flocie – zatem można zauważyć tutaj podobieństwo do problemu komiwojażera (również jest to problem NP-trudny), jednak tematyka VRP jest lepiej skalowalna i posiada przewagę w postaci planowania tras dla floty pojazdów co jest pożądane w wymaganiach stawianych przez współczesną logistykę i różnorodne dziedziny życia codziennego.

Uznaje się, że problem marszrutyzacji pojazdów został po raz pierwszy zdefiniowany przez G. B. Dantzig i J. H. Ramser w 1959 roku [2]. Swoje rozważania opierali na przykładzie dystrybucji paliw w której mieli dostęp do jednakowych cystern, a celem było zaopatrzenie wszystkich stacji paliw, przy jak najkrótszej przebytej drodze [2].

2.2.2. Pojęcia uzupełniające

Graf - w kontekście wyznaczania tras pojazdów jest główną strukturą danych dzięki którym możliwa jest implementacja jakiegokolwiek algorytmu rozwiązującego poruszany problem. Wynika to przede wszystkim z analogii która istnieje pomiędzy problemem rzeczywistym a omawianą strukturą danych. Odnosząc się do tematyki niniejszej pracy, zostanie zastosowane pewne powiązanie w którym przyjęto za wierzchołki – miejscowości – punkty do odwiedzenia, a jako krawędzie grafu – drogi [3] [4], tak jak pokazano na rysunku Rys. 2.1.



Rys. 2.1 Przykład grafu pełnego, skierowanego, z wagami

Problem marszrutyzacji pojazdów, opiera się o wyznaczaniu trasy przejazdu na mapie (a konkretniej drogach – które można sparametryzować), pomiędzy istniejącymi punktami kluczowymi do odwiedzenia (tj. klienci), o danych współrzędnych geograficznych.

W kontekście algorytmiki, to zadanie stanowi nieco bardziej rozbudowany problem poszukiwania najkrótszych ścieżek w grafie. Graf, to uporządkowana struktura danych, która składa się ze skończonego zbioru wierzchołków grafu V i krawędzi grafu E oraz wag d_{ij} [3] – które w reprezentacji graficznej odpowiadają:

- Wierzchołki grafu - punkty płaszczyzny (dla VRP: klienci/lokalizacje);
- Krawędzie grafu – odpowiadają za połączenia pomiędzy wierzchołkami grafu [5]. W kontekście VRP, jest to fizyczne połączenie między wierzchołkami, determinujące możliwość przejazdu z punktu do punktu przez wybrany pojazd. W dużym uproszczeniu, krawędź grafu stanowi drogę. Ponadto waga krawędzi jest pewną funkcją wagową, która może określać koszty podróży (odległość, zagrożenia, opłaty za podróż). Głównym celem optymalizacji problemu VRP jest wyznaczanie ścieżek w grafie w taki sposób, aby trasa przejazdu generowała możliwie jak najmniejszy koszt (tzn. wagowo była bliska możliwego minimum).

Opisany graf dla tego typu problemu można scharakteryzować jako:

- skierowany - gdyż można ustalać kierunek, w którym może udać się pojazd
- pełny – ponieważ każdy wierzchołek jest połączony w relacji „każdy z każdym”,
- ważony – ponieważ definiowana jest waga każdej krawędzi jako koszt przejazdu.

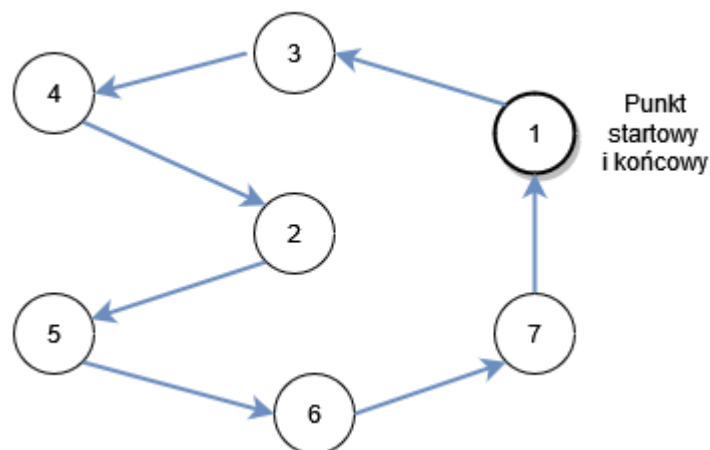
Optymalizacja – to poszukiwanie najlepszego względem obranego kryterium rozwiązania, za pomocą metod matematycznych. Może przyjmować formę minimalizacji kryterium, lub też maksymalizacji. W kontekście VRP, optymalizacja najczęściej dotyczy wielu kryteriów – zależy od wariantu problemu, stąd też stosuje się optymalizację wielokryterialną. Polega ona na użyciu kilku funkcji jakości, co umożliwia jednoczesne uwzględnianie tych parametrów w doborze rozwiązania optymalnego.

2.2.3. Problem komiwojażera (ang. Travelling Salesman Problem)

Problem komiwojażera, jest uznawany za jeden z najstarszych problemów optymalizacyjnych z dziedziny rozwiązywania problemów grafowych. Badacze zajmują się

tym problemem już od wczesnych lat trzydziestych ubiegłego wieku, kiedy to Karl Menger (austriacki matematyk) sformułował problem komiwojażera [6], który dziś można zdefiniować następująco:

Komiwojażer (inaczej handlarz obwoźny) ma za zadanie odwiedzić dokładnie jeden raz każdą z wybranych miejscowości i powrócić do punktu z którego rozpoczął swą wędrówkę. Znane są koszty podróży pomiędzy każdą parą miejscowości. Celem nadrzędnym tego zadania, jest wyznaczyć trasę w taki sposób, aby sumaryczny koszt podróży był jak najmniejszy.



Rys. 2.2 Przykład graficzny dla problemu Komiwojażera

Okazuje się bowiem, że mimo bardzo prostego sformułowania problem jest nadal inspirujący dla wielu naukowców, z uwagi na bardzo dużą złożoność obliczeniową. Tego typu problem zaklasyfikowano jako NP-trudny [6], co oznaczać będzie, że (aktualnie) nie istnieje rozwiązanie, które w czasie wielomianowym jest w stanie osiągnąć minimum (czyli w tym przypadku, rozwiązania optymalnego w całej dziedzinie problemu).

Definiując problem z punktu widzenia algorytmiki i struktur danych, problem komiwojażera polega na znalezieniu najkrótszego cyklu Hamiltona¹ w ważonym grafie.

2.2.4. Problem chińskiego listonosza

Problem chińskiego listonosza, został zdefiniowany nieco później, bo w 1962 roku przez chińskiego matematyka Meigu Guan (lub Mei-ku Kuan) [7]. Definicja tego problemu brzmi następująco:

¹ Cykl Hamiltona – to zamknięta droga, w której każdy z wierzchołków grafu jest odwiedzony dokładnie raz (poza wierzchołkiem początkowym).

Listonosz mający dostarczyć listy wyrusza z placówki pocztowej i musi przejsć przez wszystkie ulice co najmniej jeden raz, przy czym końcowym punktem jego podróży będzie placówka pocztowa z której wyruszył. Celem tego zadania jest wyznaczenie takiej trasy, aby listonosz poniósł jak najmniejszy koszt podróży [7], [8].

Z pozoru to zadanie brzmi dość podobnie do opisywanego wcześniej problemu komiwojażera, jednak istnieją pewne istotne różnice na które należy zwrócić szczególną uwagę:

- Listonosz ma za zadanie przejść przez wszystkie ulice co najmniej jeden raz – poszukiwany będzie cykl Eulera¹ (a nie jak w przypadku TSP – Hamiltona). Jest to cecha charakterystyczna zadań z dziedziny ARP (Arc Routing Problem) [9]
- Złożoność obliczeniowa problemu niejednoznaczna – W przypadku grafów skierowanych lub nieskierowanych – czas rozwiązania jest wielomianowy. W przypadku grafów mieszanych – jest już problem NP-trudny.

2.2.5. Definicja problemu marszrutyzacji pojazdów i warianty rozszerzające

2.2.5.1. Klasyczna definicja problemu VRP [10]

Dane:

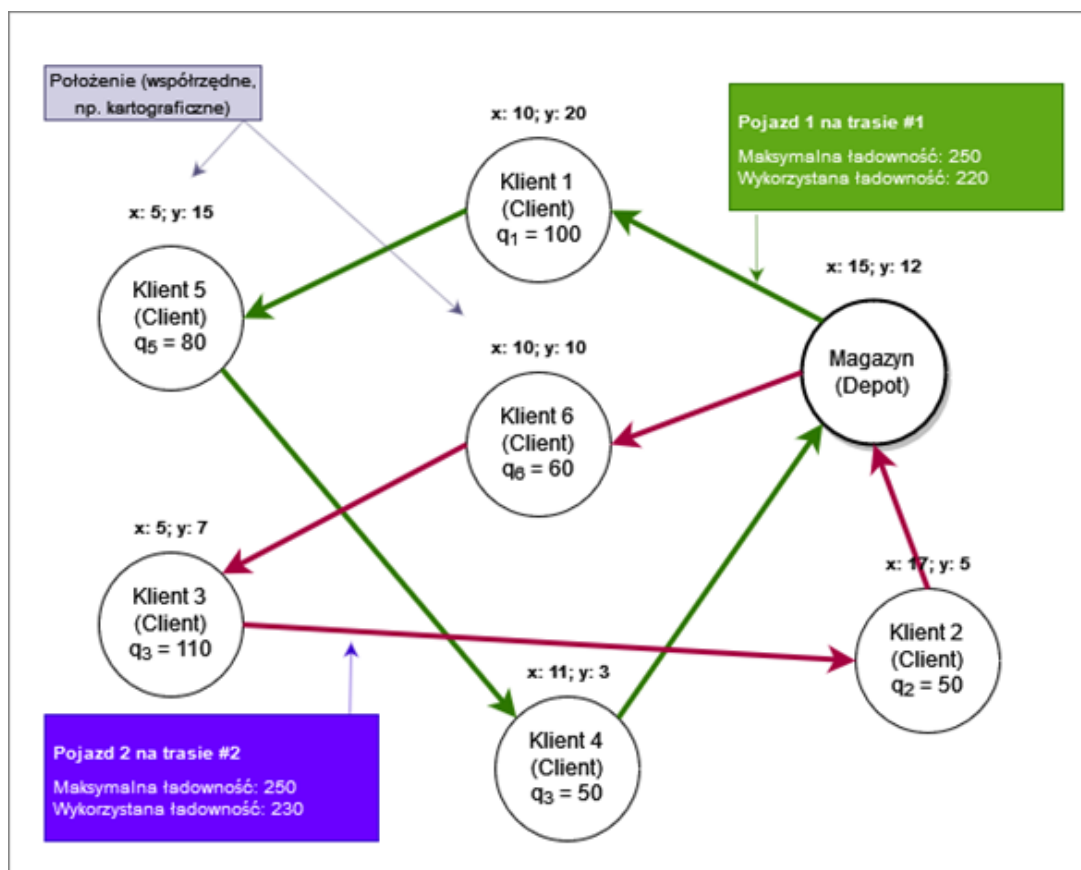
- Skierowany graf: $G = (V, E)$ z funkcją wagową $d : E \rightarrow R_{\geq 0}$ (Rys. 2.1)
- Graf G zawiera n wierzchołków (numerowane od 1..n) $V = \{1, \dots, n\}$; gdzie pierwszy wierzchołek grafu jest zawsze magazynem (ang. Depot), a pozostałe wierzchołki reprezentują lokacje (miasta/klientów)
- Macierz $D = (d_{ij})$ przechowuje wartości funkcji wagowej d . d_{ij} to waga łuku (ij) – oznacza to koszt przejazdu z wierzchołka i do j (gdzie $i \in V$ oraz $j \in V$)
- Flota pojazdów jest homogeniczna (posiadają taką samą ładowność i inne potrzebne właściwości)
- Pojazdy wyruszają z magazynu, dostarczając towar do klientów lub miast
- Każdy węzeł (wierzchołek), ma zdefiniowaną ilość towaru $q_i \geq 0$, która ma zostać do niego dostarczona przez pojazd z floty. (Uwaga, dla magazynu tj. $i = 1$ nie definiuje się ilości dostarczanego towaru)

¹ Cykl Eulera – to zamknięta droga w grafie, która przechodzi przez każdą z jego krawędzi dokładnie jeden raz.

Założenia:

- Każdy wierzchołek $v \in V$ musi zostać odwiedzony dokładnie jeden raz, przez dokładnie jeden pojazd z floty.
- Początek i koniec każdej trasy zawsze w magazynie (tj. $v = 1$)
- Całkowity przewożony ładunek na danej trasie, nie może przekroczyć maksymalnej ładowności pojazdu.

Specyfika problemu VRP oraz możliwa duża objętościowo ilość danych sprawia, że problemy marszrutyzacji pojazdów, również jest klasyfikowany jako problem NP-trudnym [2], stąd wymagane do jego rozwiązania jest użycie algorytmów heurystycznych.



Rys. 2.3 Przykład reprezentacji graficznej (rozwiązanie) problemu VRP

Obecnie problem marszrutyzacji pojazdów jest szeroko wykorzystywany w logistyce i organizacji (np. przystanków) zbiorowej komunikacji. Pozwala on znacząco redukować ilość przebytych kilometrów przez pojazdy, czas przejazdu lub oczekiwania na transport przez klienta. Ponadto optymalizacja pozwala również na zmniejszenie emisji zanieczyszczeń do atmosfery, zwiększenie zadowolenia klientów czy podniesienie bezpieczeństwa transportu jak i ludności. Jest to jednak możliwe dzięki rozszerzeniu

problematyki o dodatkowe parametry do zoptymalizowania. Obecnie jednym z popularniejszych wariantów problemu są [4] [9] [11]:

- VRP – Vehicle Routing Problem

Jest to najbardziej podstawowy typ problemu. Polega na minimalizacji kosztu przejazdu i wyznaczenia tras dla wszystkich pojazdów tak, aby zostały odwiedzone wszystkie lokacje, a pojazdy były równomiernie wykorzystane.

- CVRP – Capacited Vehicle Routing Problem

Jest to bardzo popularny problem rozszerzający wariant podstawowy. Uwzględnia on ilość ładunku do przetransportowania. Może to być zdefiniowana masa, lub ilość sztuk towaru (np. paleta) do przewiezienia. Wariant pozwala na takie rozplanowanie tras, aby żaden z pojazdów nie został przeładowany.

- CTWVRP - Capacited with Time Window Vehicle Routing Problem

Wariant problemu rozszerzający dwa powyższe o zależność czasową dostaw. Planowanie tras musi odbyć się w taki sposób, aby transport towaru pod wskazany adres odbywał się w godzinach otwarcia danego punktu. Jeśli pojazd przyjedzie za wcześnie/za późno –prawdopodobnie będzie wybrana inna, lepsza (o ile istnieje) możliwość, czyli optymalizowany jest tutaj również czas podróży.

- MDVRP – Multi Depot Vehicle Routing Problem

Ten wariant problemu zakłada, że ilość magazynów w siatce połączeń jest większa niż jeden. Co istotne, podróż pojazdem nie musi kończyć się w magazynie z którego transport wyruszył, co znacząco zmienia postrzeganie problemu.

- GVRP - Green Vehicle Routing Problem

Jest to wariant marszrutyzacji pojazdów, który kładzie nacisk na minimalizację emisji zanieczyszczeń przez pojazdy wyruszające w trasę. Może również ingerować w dobór floty pojazdów na podstawie świadectw emisyjności (np. większe wykorzystanie pojazdów niskoemisyjnych np. z normą EURO-6, niż pojazdów z normą EURO-3 – przy założeniu że flota pojazdów nie jest homogeniczna).

- EVRP – Electric Vehicle Routing Problem

Wariant problemu VRP, który został dostosowany do dzisiejszego postępu elektryfikacji pojazdów. Zakłada optymalizację trasy pod kątem dostępności potencjalnych punktów ładowania pojazdów elektrycznych w taki sposób, aby pojazd nie był narażony na rozładowanie baterii uniemożliwiające dalszą podróż.

- CTVRP – Cargo-Theft Vehicle Routing Problem

Problem Cargo Theft będący przedmiotem badań w niniejszej pracy magisterskiej, pozwala uwzględnić ryzyko kradzieży towaru i koszty związane z takim procederem. Dotyczy on zarówno kwestii ładowności pojazdów jak i również odległości punktów dostaw. Występuje w wariancie z jednym magazynem, a ryzyko jest definiowane z osobna dla każdej z lokalizacji.

Najczęściej poprzez ryzyko rozumie się koszt związany ze stratą towaru w przypadku jego kradzieży (np. straty finansowe).

Dane dot. ryzyka kradzieży mogą być opierane na statystykach policyjnych [4], wyrażonych np. w wartościach procentowych (0% - brak ryzyka kradzieży, 100% - pewne ryzyko, że towar zostanie skradziony), przypisywane na danym odcinku trasy pomiędzy punktami do odwiedzenia.

- SBRP – School Bus Routing Problem [12]

Jest to problem marszrutyzacji autobusów szkolnych, nieco odmienny od dotychczasowych problemów z dziedziny VRP. Jest to również problem wielokryterialny, gdyż składa się z wielu problemów do optymalizacji takich jak:

- wybór odpowiednich przystanków autobusowych – problem optymalizacyjny polegający na takim doborze istniejących przystanków (np. komunikacji miejskiej) aby z przystanku mogło skorzystać jak najwięcej uczniów,

- problem generacji optymalnej trasy – można to uznać jako klasyczny problem VRP,

- potrzeba dostosowania podróży do czasu godzin lekcyjnych – może wynikać potrzeba połączenia wariantów Multi depot VRP oraz Capacited with Time Window VRP (z uwagi na np. obsługę wielu szkół przez autobusy szkolne i dostosowanie ich przejazdów do pory rozpoczęcia zajęć).

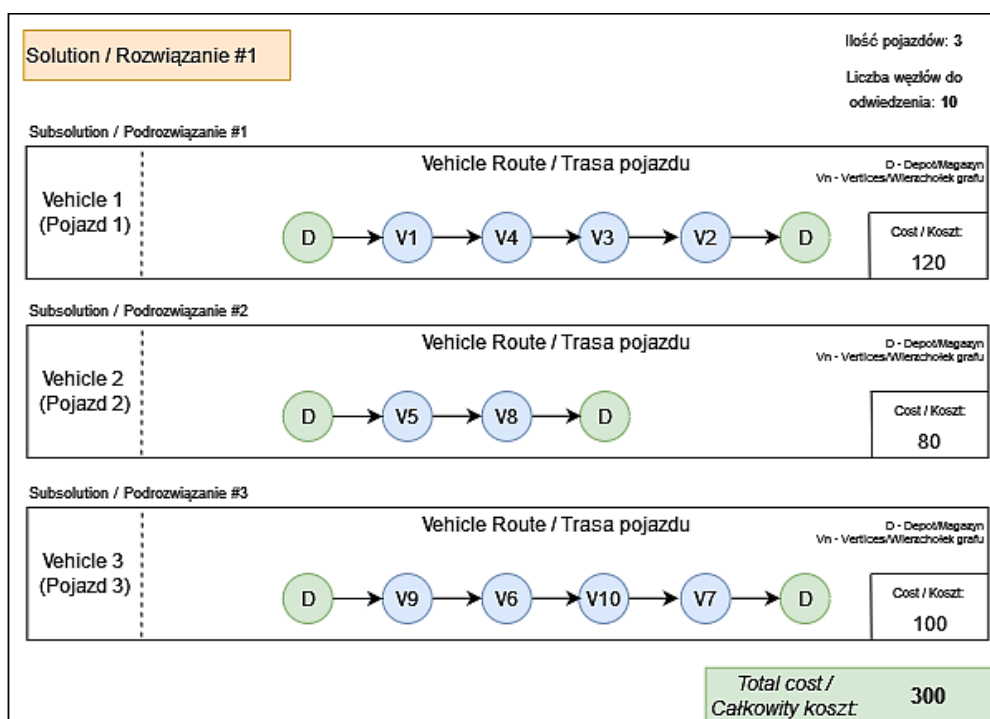
Należy pamiętać, iż w celu dokładniejszego odzwierciedlenia dowolnego problemu występującego w rzeczywistości, możliwe jest łączenie wariantów VRP ze sobą. Można stwierdzić, że tematyka problemów marszrutyzacji pojazdów jest bardzo szeroka, niemal nieograniczona jeśli chodzi o definiowanie nowych wariantów.

2.3. Rozwiązywanie problem trasowania pojazdów

Rozbudowane problemy decyzyjno-optymalizacyjne takie jak opisany problem komiwojażera, czy VRP wymagają zastosowania algorytmów heurystycznych z uwagi na niemal nieskończenie długie czasy generacji rozwiązań.

2.3.1. Rozwiązanie problemu

W kontekście całej problematyki trasowania pojazdów, mianem rozwiązania problemu określa się uporządkowaną listę lokalizacji (wierzchołków grafu) tworzących tym samym trasę do pokonania przez komiwojażera, lub flotę pojazdów (w przypadku VRP), przy czym dla problemu marszrutyzacji - rozwiązanie jest złożone tzn. każdy pojazd we flocie, posiada swoje własne rozwiązanie częściowe, czyli trasę do pokonania. Miara określającą jakość całego rozwiązania stanowi całkowity koszt podróży. Dla każdego pojazdu, możliwe jest wyznaczenie kosztu jednostkowego dla danej trasy. Przykładowa struktura rozwiązania problemu VRP, dla floty trzech pojazdów i dziesięciu lokalizacji do odwiedzenia została przedstawiona na Rys. 2.4:



Rys. 2.4 Struktura rozwiązania problemu VRP

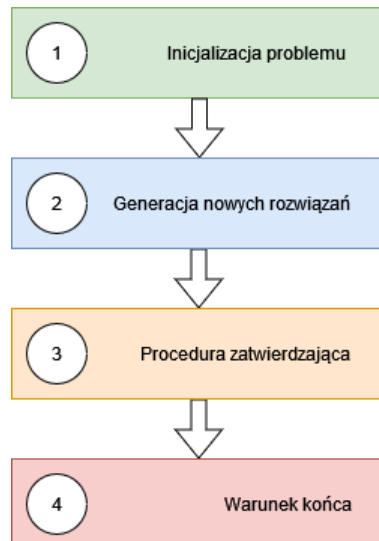
2.3.2. Algorytmy heurystyczne – zasada działania

Stosując heurystyki można otrzymać rozwiązanie problemu na akceptowalnie dobrym poziomie jakościowym, w skończonym czasie działania algorytmu. Jest to nieoceniona wartość płynąca z użycia tego typu algorytmów, często w zupełności wystarczająca do rozwiązywania rzeczywistych problemów. Jak dotąd nie otrzymano rozwiązania optymalnego w całej dziedzinie problemu algorytmami dokładnymi, tak aby przy dużej złożoności problemu czas działania algorytmu nie rósł wykładniczo [13].

W celu rozwiązania problemu VRP, można skorzystać z różnorodnych algorytmów heurystycznych takich jak:

- i) Algorytm przeszukiwania lokalnego (ang. *Local Search*)
- ii) Algorytm przeszukiwania tabu (ang. *Tabu Search*)
- iii) Algorytm symulowanego wyżarzania (ang. *Simulated Annealing*)
- iv) Algorytm mrówkowy (ang. *Ant Algorithm* lub *Ant Colony Optimization*)
- v) Algorytm genetyczny (ang. *Genetic Algorithm*)

Wszystkie algorytmy jednak mają podobną zasadę działania [13] [14]. Została ona przedstawiona na Rys. 2.5.



Rys. 2.5 Ogólna zasada działania algorytmów heurystycznych

2.3.2.1. Faza 1 – Inicjalizacja problemu

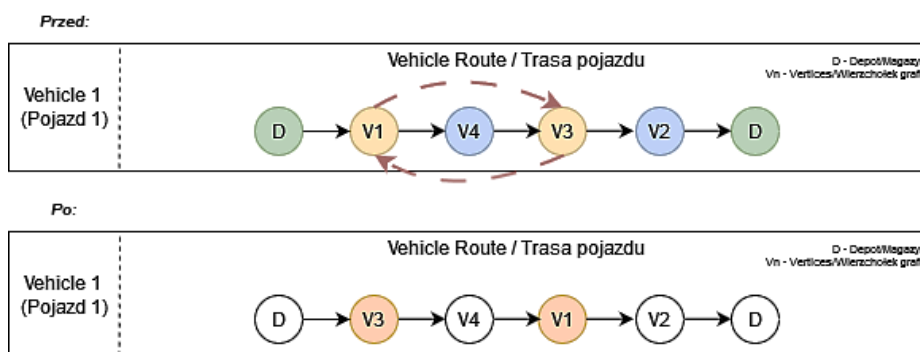
W fazie początkowej (Rys. 2.5 – Inicjalizacja problemu), następuje wygenerowanie losowego rozwiązania, zgodnego z kryteriami postawionymi w wariancie danego problemu VRP. Takie rozwiązanie, może być w teorii wystarczające, jednak w kolejnych iteracjach algorytmu, proponowane jest nowe rozwiązanie.

2.3.2.2. Faza 2 – Generacja nowych rozwiązań

W fazie drugiej algorytmu (Rys. 2.5), przedstawiane jest nowe rozwiązanie, jednakże wywodzące się z rozwiązania początkowego. W literaturze [4], zazwyczaj korzysta się z kilku wariacji tworzenia nowego rozwiązania dla problemu VRP:

a) Metoda generacji poprzez podmianę (z ang. Swap)

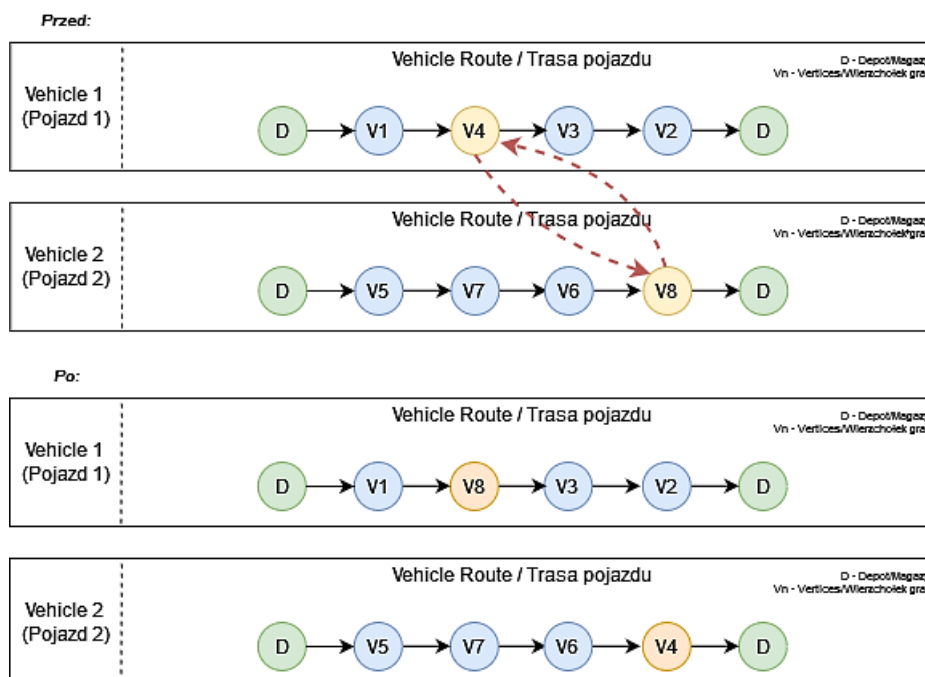
Polega na wylosowaniu dwóch pojazdów (inaczej- dwóch pod rozwiązań problemu) i zamiany pomiędzy nimi poszczególnych wylosowanych węzłów.



Rys. 2.6 Metoda „Swap”

b) Metoda generacji poprzez zamianę kolejności (z ang. Exchange)

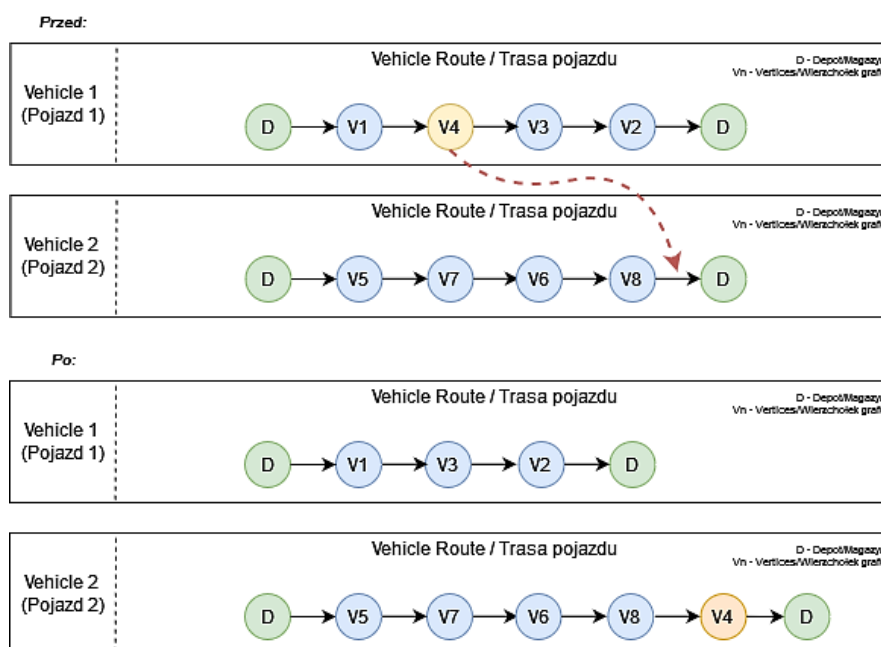
Polega na wylosowaniu pojazdu i modyfikacji jego rozwiązania w taki sposób, w którym dwa losowo wybrane węzły zostaną wzajemnie zamienione.



Rys. 2.7 Metoda „Exchange”

c) Metoda generacji dodawania i ujmowania węzłów (z ang. Shift-End)

Polega na wybraniu dwóch losowych pojazdów, gdzie w pierwszym następuje usunięcie losowo wybranego węzła, a następnie przypisaniu go na końcu trasy pojazdu drugiego.



Rys. 2.8 Metoda „Shift-End”

W ogólnym rozrachunku, nowe rozwiązanie nie oznacza generacji wszystkiego od nowa. Następuje tylko modyfikacja istniejącego, losowo wybranego rozwiązania częściowego.

2.3.2.3. Faza 3 – Procedura zatwierdzająca

W fazie trzeciej algorytmu, następuje porównanie dwóch rozwiązań - obecnie przyjętego jako akceptowalne i nowo wygenerowanego w fazie drugiej. Tutaj kryteria akceptacji są specyficzne dla algorytmu jaki został wybrany. To jest kluczowy punkt, który wyszczególnia różnice w rozwiązaniach. Można więc podsumować, że kryteria akceptacji rozwiązań definiują każdy z algorytmów heurystycznych. Algorytm Symulowanego Wyżarzania, czy Przeszukiwania Tabu będą miały inne warunki akceptacji nowych rozwiązań, gdyż należy pamiętać, że istnieje ryzyko utknięcia takiego algorytmu w którymś z minimum lokalnych dla danego rozwiązania. Aby uniknąć takiej sytuacji, umożliwia się dopuszczenie nowo wygenerowanego rozwiązania gorszego, pod pewnymi warunkami (specyficznymi dla każdego z algorytmów), tak aby umożliwić w kolejnych iteracjach przeszukiwania dalszej dziedziny rozwiązań.

2.3.2.4. Faza 4 – Warunek końca algorytmu

W ostatniej fazie algorytmu zazwyczaj pod koniec każdej iteracji, sprawdzany jest warunek zakończenia algorytmu. Jest on inny dla każdego z rodzajów algorytmów, jednak często stosuje się również praktykę ograniczania liczby iteracji algorytmu do wskazanej liczby, lub działania dążącego do ograniczenia czasowego pracy algorytmu. Poszukiwanie rozwiązania akceptowalnego może trwać bardzo długo, a z każdą kolejną iteracją wynik niekoniecznie musi być lepszy, dlatego zakłada się określony czas na działanie algorytmu, po przekroczeniu którego algorytm zakończy swoje działanie. Ograniczenie liczby iteracji może być równoważne z ograniczeniem czasowym.

2.3.3. Algorytm Przeszukiwania lokalnego

Jest najprostszym ze wszystkich algorytmów heurystycznych, prostota obejmuje zarówno część implementacyjną (z punktu widzenia programisty) jak i część logiczną algorytmu. Bazuje na zasadzie „lokalnych ulepszeń”, co może być interpretowane jako zaleta, ale również i wada z uwagi na potencjalną charakterystykę do błędzenia w minimach lokalnych danego rozwiązania.

Algorytm Przeszukiwania lokalnego cechuje szybkość działania, stosowanie prostych heurystyk oraz dwie metody akceptacji nowego rozwiązania, które definiują dwie wersje tego algorytmu [13]:

- Wersję zachłanną (z ang. *Greedy Algorithm*) [13] (Rys. 2.9) – w której można zauważyć, iż nowe rozwiązanie jest wybierane w momencie znalezienia dowolnego lecz lepszego od dotychczasowego rozwiązania dla danej iteracji.

1. Wygeneruj początkowe rozwiązanie (**punkt startowy**).
2. Oblicz całkowity koszt dla początkowego rozwiązania (aktualne rozwiązanie).
3. **Powtarzaj**, dopóki nie nowe rozwiązanie będzie lepsze od obecnie najlepszego:
 - a. **Powtarzaj**, dopóki nie przeszukasz całego sąsiedztwa obecnego rozwiązania:
 - i. Wybierz aktualnie analizowane rozwiązanie, oblicz jego całkowity koszt.
 - ii. Czy jest ono lepsze od obecnie najlepszego?
(Jeśli tak – zaakceptuj je, przerwij przeszukiwanie sąsiedztwa, jeśli nie, kontynuuj dalej)
 - b. Jeżeli nie znaleziono lepszego sąsiada, zakończ algorytm.
4. Zwróć najlepsze znalezione rozwiązanie

Rys. 2.9 Zachłanny algorytm przeszukiwania lokalnego

- Wersję największego spadku (z ang. *Steepest descent*) (Rys. 2.10) [13] – inaczej, wersja stroma algorytmu – gdzie wybrane zostaje najlepsze rozwiązanie spośród całego sąsiedztwa.

1. Wygeneruj początkowe rozwiązanie (**punkt startowy**).
2. Oblicz całkowity koszt dla początkowego rozwiązania (aktualne rozwiązanie).
3. **Powtarzaj**, dopóki nowe rozwiązanie będzie lepsze od obecnie najlepszego:
 - a. **Powtarzaj**, dopóki nie przeszukasz całego sąsiedztwa obecnego rozwiązania:
 - i. Znajdź sąsiadów aktualnego rozwiązania poprzez wprowadzenie pewnych zmian.
 - ii. Dla każdego z sąsiadów:
 - Oblicz całkowity koszt tego rozwiązania.
 - Jeśli wartość całkowitego kosztu dla sąsiada jest lepsza niż dla poprzednio znalezionego, to zaakceptuj sąsiada jako nowe rozwiązanie.
 - b. Jeśli nie znaleziono lepszego sąsiada, zakończ algorytm.
4. Zwróć najlepsze znalezione rozwiązanie.

Rys. 2.10 Stromy algorytm przeszukiwania lokalnego

2.3.4. Algorytm Symulowanego wyżarzania

Algorytm wywodzący się z stosowanej w hutnictwie metody wyżarzania stali. Jest to jeden z rodzajów obróbki cieplnej, mający na celu poprawę stabilności fizykochemicznej metalu, tzn. obrabiany metal posiada lepsze właściwości mechaniczne i posiada jednorodną strukturę drobnoziarnistą [15]. Bez tego procesu stal wytwarzana w hutach byłaby podatna na działanie sił wewnętrznych, we względnie losowych miejscach materiału (dochodziłoby do pęknięć materiału, deformacji). Z fizycznego punktu widzenia, dążymy do osiągnięcia równowagi termodynamicznej materiału [13] [15].

Może się wydawać, że taki proces nie będzie miał nic wspólnego z problematyką optymalizacji tras pojazdów, jednak należy zwrócić uwagę na to, jak przeprowadzana jest taka procedura. Proces ten przebiega wieloetapowo (z uwagi na żądaną charakterystykę stali), gdzie każdy z etapów zawsze rozpoczyna się i kończy tak samo, tzn. metal musi zostać rozgrzany do pewnej względnie wysokiej temperatury, a następnie temperatura ta, ma być powoli i stopniowo obniżana. Rezultatem takiego działania jest stabilny materiał, który w początkowej fazie procesu, był daleko od punktu równowagi. Można więc uznać, że im wyższa temperatura nieobrobionego materiału, tym znajduje się on dalej od punktu równowagi termodynamicznej. Przetwarzając to stwierdzenie na potrzeby problematyki

optymalizacji, można stwierdzić, że rozwiązania w „dużej temperaturze” będą dalekie od optimum, z kolei rozwiązania powstałe na wskutek „schładzania”, będą znacznie bliżej rozwiązania optymalnego. Niepodważalną wadą takiego procesu jest czas trwania, który zarówno w rzeczywistych procesach metalurgicznych jak i algorytmicznie jest z reguły długi, natomiast zdecydowaną zaletą i cechą charakterystyczną algorytmu symulowanego wyżarzania jest możliwość przechodzenia do różnych stanów, niekoniecznie lepszych od dotychczasowo poznanych. Rozwiązanie gorsze jest dopuszczane tylko pod pewnym znanym prawdopodobieństwem przedstawionym w równaniu (3) [13]:

W przypadku kiedy nowe rozwiązanie jest **lepsze** niż poprzednie:

$$P\{\text{akceptacji } X'\} = 1 \quad (1)$$

W przypadku kiedy nowe rozwiązanie jest **gorsze** niż poprzednie:

Kryterium akceptacji:

$$P\{\text{akceptacji } X'\} = e^{\left(\frac{f(x)-f(x')}{t}\right)} \quad (2)$$

$$a < P\{\text{akceptacji } X'\} \quad (3)$$

Gdzie: $f(x)$ - rozwiązanie poprzednie, $f(x')$ – rozwiązanie nowo wygenerowane, t – aktualna temperatura algorytmu dla aktualnej epoki, a – to **liczba pseudolosowa** (o rozkładzie równomiernym) z przedziału $< 0; 1 >$ [13].

Taki zabieg, pozwala uchronić algorytm przed trwaniem w miejscu lokalnego minimum. Umożliwia to eksplorację dalszej przestrzeni rozwiązań i pozwala zachować różnorodność prowadzącą do znajdowania coraz to lepszych rozwiązań. Generacja nowego rozwiązania polega na wyborze rozwiązania powstałego na wskutek operacji opisanych w punkcie 2.3.2.2. Należy mieć na uwadze, że rozwiązanie to jest modyfikowane wraz z każdą kolejną epoką. Ogólny schemat działania algorytmu symulowanego wyżarzania, został przedstawiony na Rys. 2.11.

1. Wygeneruj początkowe rozwiązanie (**punkt startowy**).
2. Oblicz całkowity koszt dla początkowego rozwiązania (aktualne rozwiązanie).
3. Przyjmij za temperaturę bieżącą T , temperaturę początkową
4. **Powtarzaj**, dopóki temperatura bieżąca nie będzie równa temperaturze minimalnej (lub inny dodatkowy warunek zatrzymania nie zostanie spełniony):
 - a. **Powtarzaj**, do momentu osiągnięcia czasu trwania epoki:
 - i. Wybierz nowe rozwiązanie w sąsiedztwie obecnego i oblicz jego całkowity koszt
 - ii. Porównaj rozwiązania (ich całkowity koszt):
 - Jeżeli nowe rozwiązanie jest lepsze – zaakceptuj je
 - Jeżeli nowe rozwiązanie jest gorsze:
 - Wylosuj prawdopodobieństwo $a \in (0; 1)$
 - Oblicz prawdopodobieństwo akceptacji (opisane równaniem (2))
 - Zaakceptuj gorsze rozwiązanie wedle ustalonych zasad (równanie (3))
 - iii. Zaktualizuj czas trwania epoki (+1)
 - b. Spadek temperatury - przelicz bieżącą wartość temperatury: $T = T * \alpha$
5. Zwróć najlepsze znalezione rozwiązanie

Rys. 2.11 Pseudokod algorytmu symulowanego wyżarzania

2.3.5. Algorytm Przeszukiwania tabu

Jest to algorytm heurystyczny będący szerszą modyfikacją algorytmu lokalnego przeszukiwania [13]. Znaczącym ulepszeniem jest wyposażenie algorytmu w mechanizm pamięci, dla operacji jakie zostały wykonane na poprzednich wersjach rozwiązań. Pamięć operacji zakazanych (nazywana dalej *listą tabu*) zawiera dwie informacje:

- Modyfikację jaka została zakazana – tutaj podejście jest różne, można stosować technikę zapamiętywania całych zbiorów rozwiązań – niepolecane z uwagi na duże zużycie zasobów - lub można zapamiętać kluczowe informacje nt. operacji jakie zostały wykonane, aby dane (zakazane) rozwiązanie osiągnąć
- Czas trwania zakazu (inaczej: *czas trwania tabu*) – jest to po prostu zdefiniowanie długości trwania zakazu, wyrażane w iteracjach (np. czas trwania 20, oznaczać będzie brak wykonywania jakiejś wskazanej operacji przez 20 kolejnych iteracji algorytmu).

Dodanie takiego mechanizmu sprawiło, że algorytm przeszukiwania tabu, umożliwia przeszukiwanie nieco innych rejonów rozwiązań i umożliwia jego dłuższą pracę niż

algorytm przeszukiwania lokalnego. Algorytm przeszukiwania tabu posiada wiele dodatkowych mechanizmów jak np. stosowanie kryteriów aspiracji – tj. przyjmowanie rozwiązań pomimo znajdowania się na liście tabu jeżeli są one lepsze od rozwiązania obecnie najlepszego, czy też stosowanie listy kandydatów [13] – które polega na zawężeniu przeszukiwanego sąsiedztwa z uwagi na złożoność obliczeniową w przypadku dużych objętościowo problemów. W związku z zastosowaniem listy tabu, która jest de facto cykliczną i stałą strukturą (choć nie musi to być regułą – zależy to od implementacji) pojawia się pewne zagrożenie w postaci „błądzenia” algorytmu pośród odnalezionego minimum lokalnego (bądź globalnego) danego problemu. Mając powyższe na uwadze, algorytm posiada mechanizmy zakończenia w zależności od (najczęściej) całkowitej maksymalnej liczby iteracji algorytmu, bądź czasu (określonej liczby iteracji) w którym żadne rozwiązanie lepsze od aktualnie minimalnego, nie zostało dostarczone (Rys. 2.12).

1. Wygeneruj początkowe rozwiązanie (**punkt startowy**).
2. Oblicz całkowity koszt dla początkowego rozwiązania (aktualne rozwiązanie).
3. Lista zakazów tabu jest pusta.
4. **Powtarzaj**, dopóki warunek zatrzymania nie zostanie spełniony:
 - a. **Powtarzaj**, dopóki nie przeszukasz całego sąsiedztwa obecnego rozwiązania:
 - i. Wybierz aktualnie przeszukiwane rozwiązanie, oblicz jego całkowity koszt.
 - ii. Czy jest ono lepsze od najlepszego w przeszukiwanym sąsiedztwie?
(Jeśli tak – zaakceptuj je wstępnie, jeśli nie, kontynuuj dalej (tj. od punktu i)
 - iii. Czy wstępnie zaakceptowane rozwiązanie znajduje się na liście tabu?
 - Jeśli tak: odrzuć rozwiązanie, z wyjątkiem, w którym rozwiązanie jest lepsze od rozwiązania minimalnego (najlepszego).
 - Jeśli nie: zaakceptuj rozwiązanie lepsze w przestrzeni przeszukiwanego sąsiedztwa.
 - b. Porównaj rozwiązania (ich całkowity koszt):
 - i. Jeżeli rozwiązanie jest lepsze od najlepszego – zaakceptuj je
 - c. Uaktualnij listę tabu.
5. Zwróć najlepsze znalezione rozwiązanie.

Rys. 2.12 Pseudokod algorytmu przeszukiwania tabu

Algorytm w niniejszej formie nie posiada mechanizmu dywersyfikacji rozwiązań, mającej na celu przenoszenie przeszukiwania w nieodwiedzone wcześniej przestrzenie potencjalnych rozwiązań [13]. Takie mechanizmy występują głównie w znacznie bardziej zaawansowanym stopniu implementacji algorytmu przeszukiwania tabu. Na cele niniejszej pracy i przeprowadzanych badań, porównywana ma być skuteczność działania podstawowej wersji algorytmu przeszukiwania tabu, aby spróbować zaobserwować jaki wpływ na otrzymywane rozwiązania ma lista tabu w porównaniu do rozwiązań otrzymywanych za pomocą algorytmu przeszukiwania lokalnego.

2.3.6. Pozostałe algorytmy

Wymienione w punkcie 2.3.2 tj., algorytm mrówkowy i algorytm genetyczny znacząco odstają stopniem skomplikowania, szczególnie w fazie generacji nowych rozwiązań i ich późniejszej akceptacji. Oba algorytmy czerpią inspirację z obserwacji zjawisk natury, tak samo jak w przypadku algorytmu Symulowanego wyżarzania.

Algorytm mrówkowy (z ang. *Ant Colony Optimization*) został opracowany w latach dziewięćdziesiątych ubiegłego stulecia. Główną ideą algorytmu, jest wyznaczanie najkrótszych tras w grafie poprzez utrzymywanie odpowiednio wysokiego stężenia tzw. feromonów na przebytej aktualnie najlepszej kosztowo drodze. Związek feromonów jest ściśle inspirowany z natury. Mrówki w poszukiwaniu pożywienia podróżują dość losowo. W momencie, kiedy jedna z mrówek znajdzie pożywienie, wraca do kolonii wytwarzając przy tym feromony – tym samym oznacza ona trasę jako „przydatną” i widoczną dla innych mrówek. Każda inna mrówka podróżująca losowo, trafiając na ślad feromonów, zaczyna nią podążać. Jednak ślad feromonów po pewnym czasie zanika, zwłaszcza na długich dystansach [14], [16]. Logicznym więc jest, utrzymywanie się wysokiego stężenia feromonów na krótkich dystansach tras, co jest pożądane z punktu poszukiwania najkrótszych ścieżek w grafie. Po kilku cyklach takiego podążania mrówek, kosztowne trasy zostaną zapomniane (feromony wyparują), a ścieżka najczęściej uczęszczana będzie uznawana za najlepszą (bliską optimum) [16].

Algorytm genetyczny jest najczęściej wykorzystywanym algorytmem w dziedzinie problemów VRP [11] [14], z uwagi na swoją skuteczność znajdowania najkrótszych tras. Inspirowany mechanizmem ewolucyjnym ze świata zwierząt. Zakłada się dobieranie w parę najlepszych osobników, mających na celu wydanie najsilniejszego potomstwa (czyli w tym przypadku trasy o najlepszych dotychczas parametrach). Każda iteracja algorytmu, może nawiązywać do „pokolenia”, które w założeniu powinno być silniejsze od poprzedniego.

Strategia działania, polega na modyfikacji rozwiązań poprzez krzyżowanie najlepszych otrzymanych wyników (w drodze selekcji), a następnie nowopowstałe rozwiązanie zostaje

delikatnie zmodyfikowane w sposób losowy (następuje tzw. mutacja), co pozwala na osiągnięcie różnorodności „genetycznej” [14]. Algorytm działa do momentu wykrycia braku poprawy w kolejnych iteracjach. Wtedy uznaje się wynik za najlepszy z możliwych, zatem kombinacja odwiedzonych punktów dla każdego z pojazdów, może uchodzić za najlepszą (tj. o najmniejszym koszcie)

2.4. Problematyka VRP w ujęciu światowym

Problematyka marszrutyżacji pojazdów jest wysoce praktycznym tematem do badań optymalizacyjnych. Ilość prowadzonych prac naukowych w tych tematach rośnie z roku na rok, co sprawia że powstają coraz to nowe warianty VRP i jeszcze bardziej finezyjne sposoby rozwiązywania tych problemów.

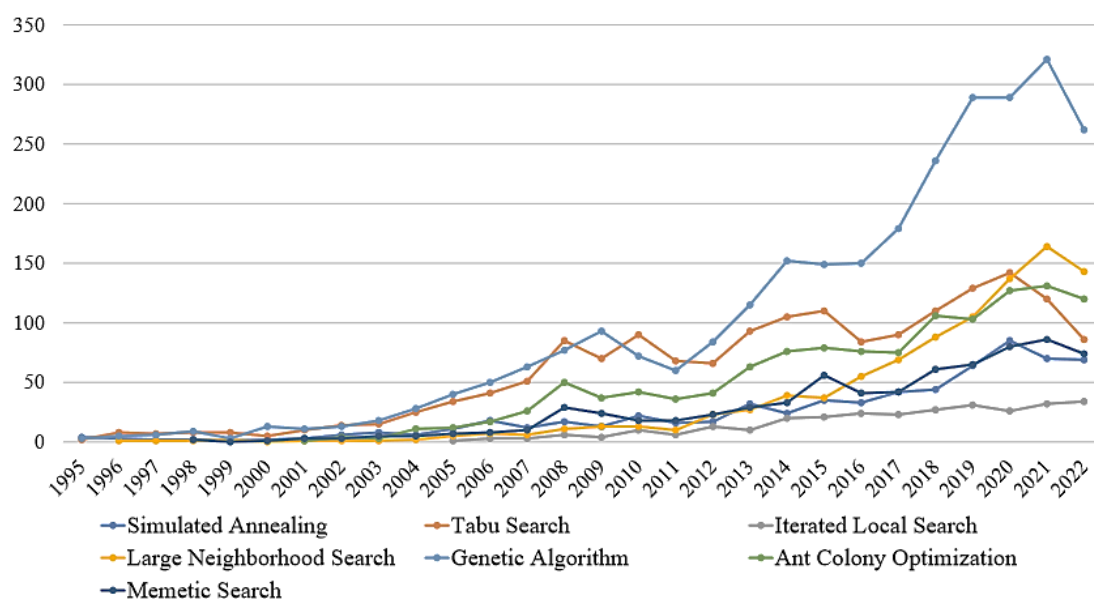
Istnieją liczne prace naukowe dotyczące podstawowych wariantów VRP, (np. Capacited VRP, Capacited VRP with Time Window), jednak ilość badań dot. Cargo Theft VRP i wpływu na jego rozwiązanie końcowe na skutek doboru algorytmu jest znikomy. Potwierdzeniem tej reguły jest fakt, iż nie istnieją de facto oficjalne zestawy danych testowych dedykowane do tego problemu trasowania pojazdów.

W 2013 roku, przeprowadzono badania klasyfikujące kraje świata pod kątem kradzieży towarów podczas transportu. Przedstawienie wyników oparto o pięciostopniową skalę o parametrach klasyfikujących ryzyko: niewielkie, podwyższone, umiarkowane, wysokie, ogromne [4]. W zestawieniu bardzo wysoko znalazły się takie kraje jak: Meksyk, RPA oraz Brazylia.

W związku z powyższym, główne prace naukowe tego wariantu były prowadzone w Brazylii, gdzie szczególnie wysoki odsetek kradzieży dotyczy dystrybucji farmaceutyków a straty poniesione w ramach jednego transportu mogą wynosić miliony (o ile nie setki milionów) złotych [4]. W 2018 roku przeprowadzono pierwsze próby optymalizacji pod kątem ryzyka kradzieży towaru, wyniki tych badań zostały przedstawione w artykule naukowym [4]. W pracy badawczej zaprezentowano definicje problemu, sens praktyczny badania (otrzymany model przetestowano u jednego z brazylijskich dystrybutorów farmaceutycznych), a do rozwiązania problemu skorzystano z algorytmu Symulowanego Wyżarzania, który autorzy dostosowali na potrzeby CTVRP.

Według danych opublikowanych w innym artykule naukowym [11], można zaobserwować trend wzrostowy dla wykorzystania algorytmów heurystycznych w rozwiązywaniu problemów optymalizacyjnych VRP, co wynika głównie z faktu rosnącego zainteresowania w wykorzystaniu praktycznym i rosnącymi w związku

z powyższymi publikacjami naukowymi [4], [10], [11]. Publikacja zawiera informacje dot. ilości publikacji naukowych, względem rodzaju algorytmu heurystycznego wykorzystanego do rozwiązania danego problemu. Dane te, nie zawierają jednak podziału na warianty VRP, ukazują więc zbiorczą ilość, która ma się nijak do badanego wariantu Cargo Theft VRP.



Rys. 2.13 Wykres przedstawiający ilość publikacji dot. problematyki VRP [11]

Mimo wszystko, od ponad dziesięciu lat następuje wyraźny wzrost zainteresowania tematyką marszrutyzacji pojazdów, co zbiega się również ze znaczącym skokiem wydajnościowym podzespołów komputerowych, a ten fakt, może mieć wpływ na popularyzację tej niszowej (jak dotąd) dziedziny nauki.

Do celów badawczych wybrano mniej popularne algorytmy – takie jak Symulowanego wyżarzania (z ang. Simulated Annealing), przeszukiwania Tabu (z ang. Tabu Search) oraz przeszukiwania lokalnego (z ang. Local Search). Decyzję motywuje się faktem różnorodności powyższych trzech algorytmów oraz stosunkowo dużej różnicy pod kątem publikacji, co może nasuwać pytania pod kątem zasadności użycia tych algorytmów w problematyce VRP i ich potencjalnych zalet lub wad.

Rozdział 3

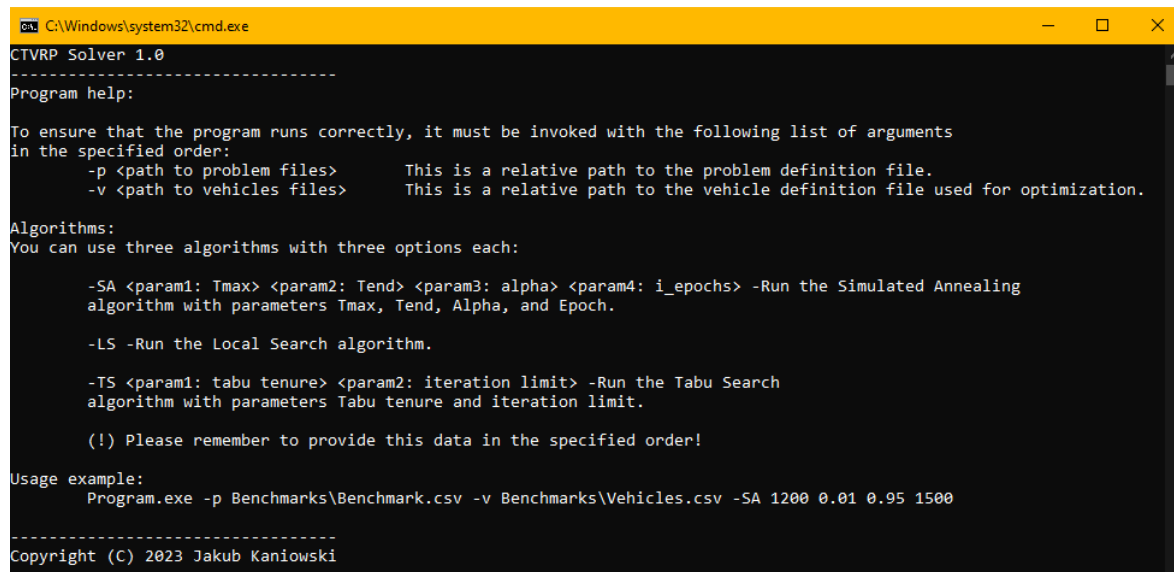
Przedmiot pracy

Niniejszy rozdział dotyczy zaproponowanego rozwiązania w postaci omówienia implementacji wykorzystanych algorytmów, sposobów generowania nowych rozwiązań problemu CTVRP oraz opisu funkcjonalnego programu wykorzystanego do badań eksperymentalnych.

3.1.1. Główny program – Solver

Na potrzeby niniejszej pracy magisterskiej, której celem jest implementacja kilku dowolnych algorytmów heurystycznych i ich badanie, wykonano program umożliwiający dalsze przeprowadzenie prac. Aplikacja została nazwana „Solver” od angielskiego słowa „*solve*” – „rozwiązać”. Naczelną funkcjonalnością programu jest właśnie rozwiązywanie problemu Cargo Theft VRP oraz zbieranie pozyskanych danych do plików tekstowych (w formacie tekstowym lub rozdzielanym przecinkami – CSV – z ang. *Comma Separated Values*). Aplikacja działa tylko na systemach 64-bitowych rodziny Windows, w trybie konsolowym (Rys. 3.1). W ramach swojej funkcjonalności, zaimplementowano trzy algorytmy możliwe do uruchomienia przez użytkownika:

- Algorytm Lokalnego przeszukiwania – w wersji stromej
- Algorytm Symulowanego wyżarzania
- Algorytm Przeszukiwania tabu



```

C:\Windows\system32\cmd.exe
CTVRP Solver 1.0
-----
Program help:

To ensure that the program runs correctly, it must be invoked with the following list of arguments
in the specified order:
    -p <path to problem files>      This is a relative path to the problem definition file.
    -v <path to vehicles files>     This is a relative path to the vehicle definition file used for optimization.

Algorithms:
You can use three algorithms with three options each:

    -SA <param1: Tmax> <param2: Tend> <param3: alpha> <param4: i_epochs> -Run the Simulated Annealing
    algorithm with parameters Tmax, Tend, Alpha, and Epoch.

    -LS -Run the Local Search algorithm.

    -TS <param1: tabu tenure> <param2: iteration limit> -Run the Tabu Search
    algorithm with parameters Tabu tenure and iteration limit.

    (!) Please remember to provide this data in the specified order!

Usage example:
    Program.exe -p Benchmarks\Benchmark.csv -v Benchmarks\Vehicles.csv -SA 1200 0.01 0.95 1500

-----
Copyright (C) 2023 Jakub Kaniowski

```

Rys. 3.1 Widok programu "Solver.exe" - sekcja pomocy

3.1.1.1. Specyfikacja aplikacji

Implementacja programu została oparta o język C++11 - z uwagi na wykorzystywanie wyrażeń lambda oraz pętli for opartej na zakresie danych (z ang. Ranged for loop). Ponadto wykorzystana została biblioteka standardowa STL oraz darmowa biblioteka Boost w wersji 1.8.3, objęta licencją Boost Software License. Biblioteka Boost umożliwia wykorzystanie gotowych narzędzi umożliwiających budowę grafów (wykorzystywanych w kontekście VRP) oraz zestaw funkcji umożliwiający prosty zapis/odczyt do pliku zarówno tekstowego jak i rozdzielanego przecinkami CSV.

Taki zestaw użytych technologii umożliwia wprowadzenie szerokich zmian w przyszłości, obejmujących rozbudowanie aplikacji o aspekty takie jak:

- wielowątkowość – umożliwiająca przyspieszenie i wykorzystanie pełnej mocy obliczeniowej procesora CPU (z ang. *Central Processing Unit*)
- integracji z procesorami o wysokiej wydajności obliczeniowej GPU (z ang. *Graphics Processing Unit*) – np. wykorzystania architektury Nvidia CUDA.
- wieloplatformowość - np. kompilacja programu dla systemów rodziny Unix (np. Linux)

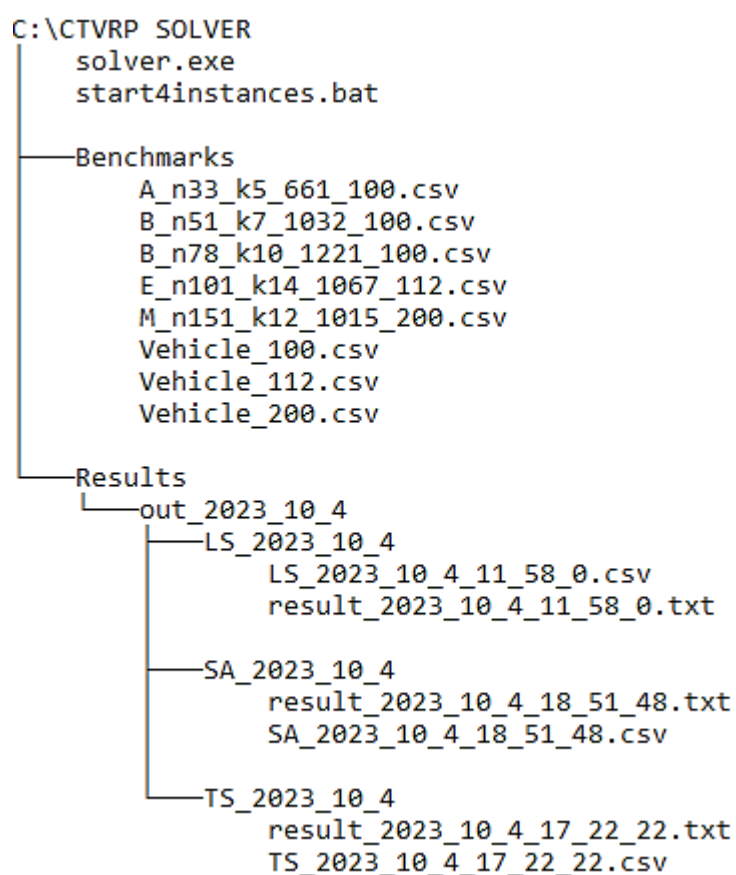
Język C++ mimo iż, uchodzi za trudny w implementacji odwdzięcza się dobrą wydajnością (stąd powszechnie się go stosuje w implementacji algorytmów) i szerokim polem do optymalizacji z uwagi na swoją niskopoziomowość i brak nadrzędnych programów służących do uruchamiania programu (jak np. interpretery języka – np. Interpreter języka Python, lub środowisko uruchomieniowe języka Java).

3.1.1.2. Funkcje aplikacji – podstawowe wykorzystanie

Ponieważ program wykorzystany do badań jest aplikacją konsolową, zostanie opisany krótki instruktaż jak korzystać z aplikacji. Program składa się z kilku funkcji, po wywołaniu

których w oknie konsoli pojawi się odpowiedni widok aplikacji, jednak zanim użytkownik przystąpi do obsługi aplikacji, należy wskazać właściwy folder w którym znajduje się program „solver.exe”. Należy mieć również na względzie lokalizacje wykorzystywanych zbiorów danych, które będą wykorzystywane przez program.

Całkowitą strukturę katalogów dostarczonego programu przedstawia Rys. 3.2, z którego wynika, iż program *solver.exe* znajduje się w katalogu nadrzędnym aplikacji, wraz z plikiem wsadowym *start4instances.bat* – który umożliwia uruchomienie czterech instancji programu *solver.exe* na raz, w odstępie kilkusekundowym – przy czym każda instancja programu jest uruchomiona w osobnym wątku procesora. Umożliwia to niezależną od siebie pracę czterech programów w jednym czasie co znacząco automatyzuje i przyspiesza proces zbierania danych. Plik wsadowy można modyfikować w zależności od konfiguracji sprzętowej środowiska.



Rys. 3.2 Struktura plików i katalogów głównego programu

Warto zwrócić uwagę na dwa katalogi przedstawione na Rys. 3.2 o nazwie „Benchmarks” – w którym mieszczą się pliki z danymi w formacie CSV oraz „Results”- w którym mieszczą się rozwiązania wygenerowane przez aplikację Rys. 3.2. i są prezentowane jako „zrzut” widoku z okna konsoli (pliki tekstowe) oraz logi zebrane podczas działania poszczególnych algorytmów.

Obecność wspomnianych katalogów w tej samej lokalizacji z programem *solver.exe* jest wymagana do poprawnego i niezakłóconego działania aplikacji.

Aby uruchomić aplikację *solver.exe* wystarczy wywołać program w oknie wiersza polecenia, jednak aplikacja zakończy się błędem z powodu niewystarczającej ilości parametrów (pokazane na Rys. 3.3 w zależności od spowodowanego błędu)

<pre>CTVRP Solver 1.0 ----- No input paths! Program closed.</pre>	<pre>CTVRP Solver 1.0 ----- Empty -p parameter. Program closed.</pre>
<pre>CTVRP Solver 1.0 ----- <[Algorithm]> : No enough input parameters! Program closed.</pre>	<pre>CTVRP Solver 1.0 ----- Empty -v parameter. Program closed.</pre>

Rys. 3.3 Błąd programu - brak odpowiednich parametrów wejściowych

Prawidłowa ilość parametrów podanych wraz z wywołaniem programu z wiersza linii poleceń to:

- **-p** <ścieżka do pliku z definicją problemu>
- **-v** <ścieżka do pliku z definicją pojazdu>

oraz w zależności jaki algorytm użytkownik chce uruchomić:

- **-LS** – uruchomiony zostanie algorytm **lokalnego przeszukiwania**
- **-SA** [Tmax] [Tend] [alpha] [maksymalna liczba iteracji algorytmu] – uruchomiony zostanie algorytm **symulowanego wyżarzania**, o wyżej wskazanych parametrach.
- **-TS** [czas tabu] [maksymalna liczba iteracji algorytmu] – uruchomiony zostanie algorytm **przeszukiwania tabu** o wyżej wskazanych parametrach.

Finalnie **polecenie wywołania programu** z użyciem algorytmu np. przeszukiwania tabu, wygląda następująco:

```
solver.exe -p Benchmarks\Benchmark_01.csv -v Benchmarks\Vehicles.csv -TS 10 5000
```

Program po poprawnym wywołaniu wczyta podane pliki dot. problemu oraz pojazdu. W przypadku błędu, zostanie wyświetlona odpowiednia informacja użytkownikowi. Przed zakończeniem działania programu (po skończonej optymalizacji) zostanie wyświetlone szczegółowe podsumowanie, które jest zapisywane w katalogu „Results” (Rys. 3.2) w pliku tekstowym o nazwie z przedrostkiem „result_” (Rys. 3.2), który zawiera dokładne odwzorowanie tego, co ukazało się użytkownikowi w oknie konsoli. Wszystkie dane zbierane podczas działania algorytmu – jak np. bieżąca iteracja, bieżący czas, czy inne

zmiany wartości zostaną zapisane w pliku CSV, z prefiksem odpowiadającym wykorzystanemu algorytmowi (Rys. 3.2).

3.1.1.3. Pliki wejściowe programu

Tabela 3.1 oraz Tabela 3.2 zawiera kolejno informacje nt. wykorzystywanej struktury danych plików wejściowych dla przygotowanych zbiorów testowych oraz strukturę przechowywania danych nt. wykorzystywanych pojazdów. Każdy z plików jest zapisywany w formacie CSV. Jest to jedyny dopuszczalny format i kolejność danych, która jest obsługiwana przez główną aplikację opisaną w punkcie 3.1.1.

Konfiguracja floty pojazdów została również przygotowana w zewnętrznym pliku CSV, o strukturze, którą przedstawia Tabela 3.2. Z uwagi na specyfikę niniejszego problemu należy mieć na uwadze, iż jest to flota homogeniczna dlatego, w tym pliku powinien znajdować się tylko i wyłącznie jeden pojazd. Program *solver.exe* sam dobiera wymaganą liczbę pojazdów do wskazanej instancji problemu.

Tabela 3.1 Struktura obsługiwanych zbiorów danych (przykład pliku wejściowego)

Nr. Węzła (Node)	Współrzędne		Zapotrzebowanie (Demand)	Ryzyko [%] (Risk)
	X	Y		
(Magazyn) 1	30	20	0	0
2	10	5	6	7
⋮	⋮	⋮	⋮	⋮
32	5	10	21	5

Tabela 3.2 Struktura danych dot. pojazdów (przykład pliku wejściowego)

Marka (Mark)	Typ (Type)	Ładowność (MaxCapacity)	Pojemność baku (TankCapacity)	Masa pojazdu (VehicleMass)	Spalanie (FuelConsumption)
VW	T4	100	75	1700	7

Jak można zauważyć, struktura danych dot. pojazdów zawiera nadmiarowe szczegółowe informacje nt. wykorzystywanego pojazdu. Aktualnie żadne inne informacje poza całkowitą ładownością pojazdu nie są uwzględniane na potrzeby niniejszej pracy. Użycie takiej struktury zostało podyktowane przygotowaniem programu *solver.exe* do szerszych analiz prowadzonych poza opisywaną pracą magisterską przez autora pracy w przyszłości.

3.1.1.4. Pliki wyjściowe programu

Wyniki każdej wykonanej instancji programu dzielą się na dwa rodzaje (Rys. 3.2):

- Plik z rozwiązaniem - opisowy – tekstowy, zawierający listę lokacji (inaczej – unikalne identyfikatory węzłów z zachowaną kolejnością) do odwiedzenia dla każdego z pojazdów wygenerowanych dla danego rozwiązania oraz ogólne informacje nt. przebiegu działania algorytmu (jak np. sprawdzenie spójności rozwiązania – tj. czy wszystkie lokalizacje zostały odwiedzone, a pojazdy nie są przeładowane, czy ilość wygenerowanych rozwiązań poprawnych, błędnych itp.). Nazwa tego rodzaju pliku rozpoczyna się od prefiksu „result_” (Rys. 3.2). Fragment wygenerowanego pliku przedstawiono na Rys. 3.4.

```
FOUND BEST:

V[02]:  0  21  1  31  29  3  16  20  32  0
V[05]:  0  23  6  24  2  0
V[03]:  0  13  8  7  22  28  18  0
V[04]:  0  12  10  30  25  27  4  0
V[01]:  0  11  19  14  26  5  17  9  15  0

BEST SOLUTION COST:      1847.87  found at: 0.597 s
Best Profit:              888.982
Best Profit [%]:          32.482%

Validation best solution:      OK
```

Rys. 3.4 Fragment pliku wyjściowego – tekstowego

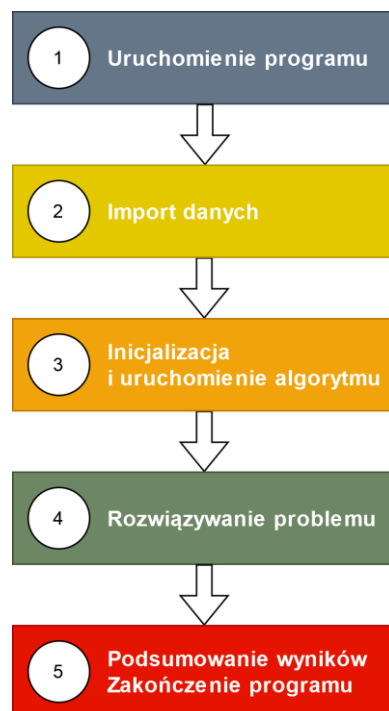
- Plik ze zmianami wartości podczas działania algorytmu – rozdzielany przecinkami CSV, zawierający czas oraz wartość rozwiązania bieżącego i najlepszego dla każdej iteracji algorytmu. Tabela 3.3 przedstawia strukturę pliku. Nazwa tego rodzaju pliku to „XX_RRRR_MM_DD.csv” (Rys. 3.2) gdzie XX to prefiks dotyczący użytego algorytmu (LS – przeszukiwanie lokalne, SA – symulowane wyżarzanie, TS – przeszukiwanie tabu), a reszta nazwy stanowi datę wykonania instancji programu w konwencji: rok, miesiąc, dzień.

Tabela 3.3 Struktura pliku CSV zawierającego wyniki działania algorytmów (przykład)

i	t	s	bs	T
Iteracja	Czas [s]	Koszt bieżący rozw.	Najlepszy koszt rozw.	Temperatura ¹
1	0.016	1875.50	1875.50	100.00
2	0.131	1813.20	1813.20	100.00
3	⋮	⋮	⋮	⋮
1000	336.520	685.32	683.85	0.01

3.1.2. Ogólna zasada działania programu Solver

Fazy działania programu *solver.exe* można wyodrębnić do następujących po sobie kilku etapach (Rys. 3.5):

Rys. 3.5 Ogólna sekwencja działania programu *solver.exe*

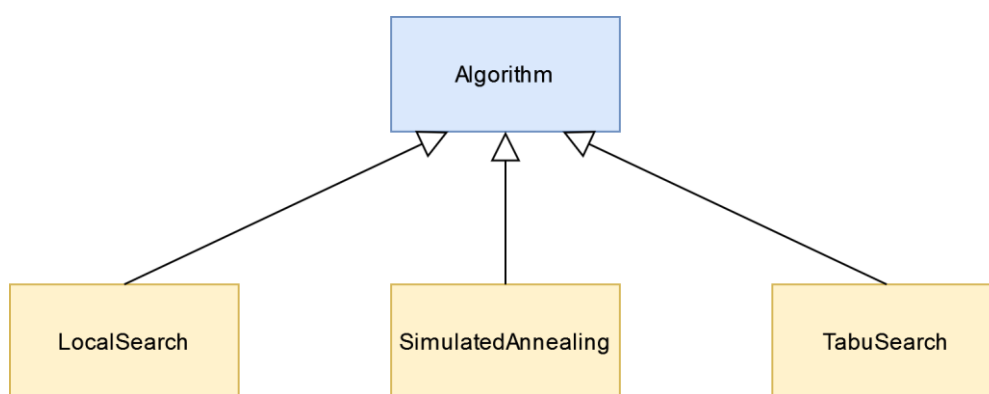
- Krok 1 – Uruchomienie aplikacji – wskazanie plików wejściowych. Walidacja wprowadzonych przez użytkownika danych (w przypadku błędów – zakończenie programu).

¹ Wartość temperatury dotyczy tylko zapisu działania algorytmu symulowanego wyżarzania

- Krok 2 – Import wskazanych przez użytkownika danych. Zbudowanie grafu wskazanego problemu (w przypadku błędów – zakończenie programu).
- Krok 3 – Inicjalizacja i uruchomienie algorytmu wskazanego przez użytkownika (podczas etapu I).
- Krok 4 – Rozwiązywanie problemu i jego jednoczesny zapis do plików wyjściowych (opisanych w punkcie 3.1.1.4).
- Krok 5 – Zakończenie algorytmu. Wyświetlenie na ekranie wiersza poleceń podsumowania działania programu dla użytkownika. Zamknięcie wszystkich plików wyjściowych. Koniec programu.

3.2. Szczegóły implementacyjne

Niniejszy program *solver.exe* zawiera implementacje trzech algorytmów wyszczególnionych w punkcie 2.3.3, 2.3.4 oraz 2.3.5. Algorytmy te zawierają pewne cechy wspólne i metody, które zostały umieszczone w klasie nadrzędnej nazwanej *Algorithm*. Klasa nadrzędna zawiera szczegóły dot. generacji nowych rozwiązań, jak i uogólnione narzędzia dot. weryfikacji poprawności otrzymywanych rozwiązań, dostępu do danych jak i logowania wyników do zewnętrznego pliku. W przypadku klas podrzędnych (dziedziczących po klasie *Algorithm*). Ogólna hierarchia wspomnianych klas została przedstawiona na Rys. 3.6, gdzie reprezentację klasy algorytmu przeszukiwania lokalnego nazwano *LocalSearch*, algorytmu symulowanego wyżarzania – *SimulatedAnnealing*, przeszukiwania tabu – *TabuSearch*.



Rys. 3.6 Hierarchia klas zaimplementowanych algorytmów

Reszta klas nie wymienionych w niniejszym rozdziale jest dostępna w elektronicznej dokumentacji sporządzonej automatycznie za pomocą narzędzia Doxygen. Dokumentacja ta została dołączona do materiałów dodatkowych załączonych do niniejszej pracy.

3.2.1. Sposoby generacji nowych rozwiązań

3.2.1.1. Generacja rozwiązania początkowego

Generowanie rozwiązania początkowego odbywa się w kroku trzecim programu *solver.exe* (Rys. 3.5) i ma to dokładnie miejsce w momencie wywołania metody odpowiedzialnej za uruchomienie wskazanego algorytmu. Pseudokod opisujący zaimplementowaną metodę generatora rozwiązań początkowych został przedstawiony na Rys. 3.7. Przyjmuje się, że import danych i zbudowanie grafu przebiegło pomyślnie.

1. Oblicz rozmiar rozwiązania (ilość wymaganych pojazdów)
2. Wybierz pierwszy pojazd do wytyczenia trasy. Stwórz listę lokalizacji L do odwiedzenia na podstawie danych z grafu.
3. **Powtarzaj**, dopóki każda z lokalizacji nie zostanie oznaczona jako „odwiedzona” i lista L nie będzie pusta:
 - Powtarzaj**, dopóki obecnie wybrany pojazd nie osiągnie swojej pełnej ładowności lub wybranie lokalizacji dla pojazdu nie będzie już możliwe (tj. brak odpowiednich lokalizacji o zapotrzebowaniu mniejszym niż pozostała ładowność) a lista L nie będzie pusta:
 - Wylosuj dowolną lokalizację z L
 - **Sprawdź**, czy zapotrzebowanie wylosowanej lokalizacji jest mniejsze bądź równe pozostałej ładowności pojazdu.
 - Jeżeli tak:
 - Przypisz lokalizację** do obecnie wybranego pojazdu.
 - Oznacz** lokalizację jako **odwiedzoną**
 - Usuń** lokalizację z listy L
 - Jeżeli nie:
 - Kontynuuj** – tj. przejdź do kolejnej iteracji pętli
 - Pojazd został maksymalnie załadowany, lub przypisanie pozostałych lokalizacji było niemożliwe – **stwórz nowy pojazd**, na wzór poprzedniego. **Wybierz nowy pojazd** jako aktualnie rozpatrywany.
4. Zweryfikuj poprawność rozwiązania:
 - Pojazdy nie mogą być przeładowane (Prawda).
 - Wszystkie lokalizacje odwiedzone (Prawda).

W przeciwnym przypadku wybierz rozwiązanie jako puste (null)/zwróć błąd
5. Zwróć otrzymane rozwiązanie.

Rys. 3.7 Pseudokod generacji rozwiązania początkowego

3.2.1.2. Generacja nowego rozwiązania na podstawie poprzedniego

Generowanie nowego rozwiązania odbywa się w kroku czwartym programu *solver.exe* (Rys. 3.5). Jest ono zależne od typu wykonywanego algorytmu heurystycznego, jednak posiada on wspólną zasadę działania, opisaną pseudokodem - Rys. 3.8. Opiera się na metodach modyfikacji rozwiązania opisanego w punkcie 2.3.2.2

1. Przyjmij przekazane w argumencie funkcji rozwiązanie początkowe. Przypisz do S
2. Wybierz dwa pojazdy i dwie lokalizacje (wybrane lokalizacje nie mogą być takie same i nie mogą być magazynem)
3. **Jeżeli:**
 - a. pojazd 1 jest taki sam jak wylosowany pojazd 2 – wykonaj metodę „exchange”
 - b. Jeżeli wylosowane pojazdy są różne:
 - i. **Sprawdź** czy możliwe jest wykonanie metody „move” – dla wylosowanych pojazdów:
Jeżeli tak:
 Ustanów tą metodę jako zaplanowaną. Przejdź do kroku 4.
Jeżeli nie:
Sprawdź czy możliwe jest wykonanie metody „swap” – dla wylosowanych lokalizacji i pojazdów:
Jeżeli tak:
 Ustanów tą metodę jako zaplanowaną. Przejdź do kroku 4
Jeżeli nie:
 Zwróć błąd – zakończ generację lub przejdź do kolejnej lokalizacji/pojazdu (zależy od wybranego algorytmu)
 - c. Wykonaj zaplanowaną operację modyfikacji rozwiązania S . Przypisz nowe rozwiązanie jako S'
4. Zweryfikuj poprawność rozwiązania S' :
 - Pojazdy nie mogą być przeładowane (Prawda).
 - Wszystkie lokalizacje odwiedzone (Prawda).
 W przeciwnym przypadku wybierz rozwiązanie S' jako puste (null)/zwróć błąd)
5. Zwróć otrzymane rozwiązanie S' .

Rys. 3.8 Pseudokod generacji nowego rozwiązania

3.2.1.3. Generatory liczb pseudolosowych wykorzystane w wykonanym programie

Program *solver.exe* zawiera elementy wymagające losowości. W związku z powyższym, wykorzystano metody generatorów liczb pseudolosowych o rozkładzie normalnym dostępne w ramach biblioteki standardowej języka C++ - *std::default_random_engine* w przypadku losowania liczb całkowitych – wykorzystywanych głównie w celu typowania lokalizacji i pojazdów do modyfikacji – jest to metoda charakteryzująca się szybkim i mało obciążającym mechanizmem generacji [17]. W przypadku liczb zmiennoprzecinkowych – wykorzystywanych do warunkowej akceptacji rozwiązania gorszego dla algorytmu symulowanego wyżarzania - zastosowano generator liczb pseudolosowych *Mersenne Twister* (*std::mt19937*), który bardzo dobrze przechodzi testy statystycznej losowości [17], przez co z powodzeniem jest stosowany w projektach naukowych.

3.2.2. Implementacja algorytmu przeszukiwania lokalnego

Algorytm przeszukiwania lokalnego został zaimplementowany dokładnie na podstawie zaprezentowanego w punkcie 2.3.3 pseudokodu. Algorytm przeszukiwania lokalnego jest algorytmem bezparametrowym, zatem jego inicjalizacja ogranicza się do przypisania odpowiednich pól klasy i dalszej iteracyjnej pracy algorytmu. Zgodnie z regułą – koniec algorytmu występuje w przypadku napotkania rozwiązania gorszego od obecnie dostarczanego. Generacja nowych rozwiązań odbywa się poprzez przeszukanie sąsiedztwa rozwiązania obecnego na zasadach opisanych w punkcie 2.3.3, zatem nie zachodzi czynnik losowy przedstawiony Rys. 3.8. Każda zmiana dokonywana jest w sposób iteracyjny – względem rozwiązania na zasadzie „każdy z każdym”.

Algorytm rozszerzono o mechanizm walidacji poprawności rozwiązania (dokładnie taki sam jak został zaprezentowany w punkcie 4 na Rys. 3.7 i Rys. 3.8). W przypadku wykrycia rozwiązania błędnego, iteracja algorytmu jest pomijana a zaproponowane nowe (błędne/uszkodzone) rozwiązanie nie jest brane pod uwagę, jednak wlicza się ono do statystyk generowanych rozwiązań, które są zapisywane w pliku wyjściowym przedstawionym na Rys. 3.4. Podsumowując – rozwiązania błędne nie wpływają na pracę algorytmu przeszukiwania lokalnego lecz informacja o ilości błędnych generacji jest przetwarzana przez program.

3.2.3. Implementacja algorytmu symulowanego wyżarzania

Algorytm symulowanego wyżarzania został zaimplementowany na podstawie zaprezentowanego w punkcie 2.3.4 pseudokodu. Proces inicjalizacji algorytmu polega na przypisaniu parametrów podanych przez użytkownika: temperatury początkowej, końcowej, długości epoki oraz współczynnika schładzania. Został rozszerzony o mechanizm

klasyfikacji rozwiązań i weryfikacji poprawności. Otrzymane nowo wygenerowane rozwiązanie, można zaklasyfikować jako:

- Rozwiązanie lepsze (od rozwiązania bieżącego) – w tym przypadku otrzymane rozwiązanie spełnia wszystkie kryteria ustanowione przez rozwiązywany problem VRP i zostaje przypisane jako rozwiązanie bieżące. Jest również porównywane z dotychczasowo zapisanym rozwiązaniem minimalnym. Przypisanie rozwiązania jako minimalnego następuje jeśli całkowity koszt rozpatrywanego rozwiązania jest mniejszy bądź równy dotychczasowemu znalezionemu minimum.
- Rozwiązanie gorsze (od rozwiązania bieżącego) - w tym przypadku stosowany jest mechanizm warunkowej akceptacji rozwiązania. Generowana jest pseudolosowa liczba zmiennoprzecinkowa (szczegóły generacji opisane w punkcie 3.2.1.3), która jest porównywana na zasadach opisanych w punkcie 2.3.4. Jeżeli rozwiązanie gorsze zostanie zaakceptowane, następuje przypisanie tego rozwiązania jako rozwiązanie bieżące. Rozwiązanie nie jest sprawdzane pod kątem otrzymywanego minimum. W przypadku kiedy rozwiązanie gorsze nie jest zaakceptowane, nowo wygenerowane rozwiązanie jest porzucane i następuje przejście do kolejnej iteracji w danej epoce algorytmu.
- Rozwiązanie błędne – w przypadku nie odwiedzenia wszystkich lokalizacji, lub przeładowania któregośkolwiek z pojazdów, rozwiązanie jest odrzucane a iteracja algorytmu pomijana – tzn. rozwiązanie błędne nie wpływa na długość epoki algorytmu.

Algorytm symulowanego wyżarzania kończy swoje działanie po osiągnięciu temperatury minimalnej. Nie zaimplementowano mechanizmu przedwczesnego kończenia algorytmu, z uwagi na chęć zaobserwowania zmian rozwiązania bieżącego do samego końca.

3.2.4. Implementacja algorytmu przeszukiwania tabu

Algorytm przeszukiwania tabu został zaimplementowany dokładnie na podstawie zaprezentowanego w punkcie 2.3.5 pseudokodu. Metoda generacji nowych rozwiązań została zmodyfikowana o możliwość zapamiętywania wykonanej ostatnio modyfikacji, tak aby można było umieszczać je w liście tabu. Ocena rozwiązania jest klasyfikowana w taki sam sposób jak w przypadku algorytmu symulowanego wyżarzania (punkt 3.2.3), z tym że w przypadku rozwiązania gorszego – jest ono po prostu pomijane, algorytm przechodzi do kolejnej iteracji, a w przypadku rozwiązania błędnego postanowiono umieszczać je na liście tabu, z uwagi na fakt, żeby nie dopuścić do nadmiernego powtarzania kroków prowadzących do błędnego rozwiązania. Algorytm został zaimplementowany w swojej podstawowej

konfiguracji wraz z wykorzystaniem kryteriów aspiracji. W obecnej formie algorytm przeszukiwania tabu jest formą rozwinięcia algorytmu przeszukiwania lokalnego. Szersza modyfikacja algorytmu przeszukiwania tabu o mechanizm dywersyfikacji czy też listy kandydatów jest możliwa w przyszłości.

Rozdział 4

Badania eksperymentalne

Niniejszy rozdział został poświęcony przeprowadzonym badaniom trzech algorytmów: Symulowanego wyżarzania, Lokalnego przeszukiwania i Przeszukiwania tabu. Badania zostały wykonane w oparciu o zaimplementowany program komputerowy, który został poddany odpowiedniej walidacji. Użycie go, pozwoliło na zebranie odpowiednich danych dot. działania każdego z algorytmów jak i wyników z przeprowadzonych optymalizacji. Uzyskane wyniki znacznie pomogą w ocenie i porównaniu każdego z wyżej wymienionych algorytmów.

4.1. Przedmiot badań

Przedmiotem badań jest analiza i porównanie działania algorytmów heurystycznych w kontekście problemów z dziedziny NP-trudnych jakim jest problem marszrutyzacji pojazdów w wariancie z ryzykiem kradzieży – Cargo Theft VRP.

Badanie polega na porównaniu algorytmów przeszukiwania lokalnego, symulowanego wyżarzania oraz przeszukiwania tabu pod względem wydajności, dokładności rozwiązań jak i również w kontekście zmiany poszczególnych parametrów algorytmów. Będą badane różne instancje problemu CTVRP, od niskiej złożoności do wysokiej, tak aby sprawdzić który z algorytmów radzi sobie z problemem CTVRP najlepiej i w jakich okolicznościach. Badanie ma znaczenie praktyczne w dziedzinie zarządzania logistycznego i optymalizacji tras pojazdów, jak i również przydatnych informacji nt. jakiego algorytmu użyć w danym przypadku.

4.2. Opis środowiska badawczego

Środowisko badawcze jakie jest wymagane do przeprowadzenia badań składa się z komputera ogólnego przeznaczenia z zainstalowanym 64-bitowym systemem operacyjnym Microsoft Windows (min. Windows 7), oprogramowania zaimplementowanego na potrzeby niniejszej pracy magisterskiej oraz dowolnego pakietu biurowego, zdolnego otworzyć pliki formatu CSV (z ang. *Comma Separated Values*). Przykładowe przygotowane środowisko zostało opisane w Tabela 4.1 oraz Tabela 4.2.

Tabela 4.1 Przygotowane środowisko badawcze – część sprzętowa

Procesor	Intel Core i7-11850H (max. 4.8GHz, 8 rdzeniowy, 16 wątkowy)
Pamięć RAM	32GB (SO-DIMM DDR4 3200 MHz)
Grafika	Nvidia RTX A2000 (4GB GDDR6 VRAM, 1215-1687Mhz, 128 bit)
Dysk	Samsung 980 PRO 1TB (zapis: 5000MB/s, odczyt: 7000MB/s)
System	Windows 10 Enterprise (64 bit, Wersja: 22H2)

Tabela 4.2 Przygotowane środowisko badawcze - część programowa

Główny program	„Solver.exe” + pliki wsadowe .bat (służące do zwielokrotnionego uruchomienia programu)
Pakiet biurowy (przetwarzanie wyników)	Microsoft Office 2013 Professional 64-bit
Program użyty do walidacji algorytmów	HeuristicLab Optimizer Wersja: 3.3.16 Do pobrania na stronie: https://dev.heuristiclab.com

4.3. Zbiór danych i założenia przyjęte do badań

Do przeprowadzenia badań, wykorzystano przygotowane testowe zbiory danych (z ang. *Benchmark*) powszechnie stosowane w dziedzinie problemów VRP, jednak z uwagi na stosunkowo nowy typ wariantu jakim jest Cargo Theft, ogólnodostępne zbiory testowe są niewystarczające – nie zawierają informacji o ryzyku kradzieży. Najbliżej do pożądanego zbioru danych, jest wariantowi Capacited VRP, który w przygotowanych zbiorach zawiera informacje o współrzędnych każdego punktu do odwiedzenia, a magazyn stanowi zawsze

pierwszy wierzchołek w grafie. Zapotrzebowanie każdego z węzłów jest definiowane obok powyższych danych. Jako podstawę do stworzenia danych testowych dla wariantu Cargo Theft VRP, posłużyły zbiory dostępne w Internecie [18], które przedstawia Tabela 4.3.

Tabela 4.3 Wykorzystane zbiory danych

Nazwa zbioru	Liczba miast	Liczba pojazdów	Ładowność pojazdu	Optymalna wartość
<i>Set A</i> : A-n33-k5	32 (+1 magazyn)	5	100	661
<i>Set B</i> : B-n51-k7	50 (+1 magazyn)	7	100	1032
<i>Set B</i> : B-n78-k10	77 (+1 magazyn)	10	100	1221
<i>Set E</i> : E-n101-k14	100 (+1 magazyn)	14	112	1067
<i>Set M</i> : M-n151-k12	150 (+1 magazyn)	12	200	1015

Powyższe zbiory danych zostały zaproponowane przez Augerat’a w roku 1995 (tak zwany „Set A” i „Set B”) oraz Christofides’a w roku 1969 i 1979 („Set E” i „Set M”). Do tego czasu, badaczom udało się odkryć minimum globalne tych problemów, dlatego taki zbiór danych dobrze nadaje się do walidacji zaimplementowanych algorytmów heurystycznych. Warto zwrócić uwagę na inne zbiory danych [18] w szczególności wygenerowane ok. 2020 roku ze względu na swoją objętość (powyżej 10000 miast). Takie zbiory nie posiadają jeszcze rozwiązania optymalnego, stąd ich skuteczność w ocenie zaimplementowanych algorytmów jest nikła i nie zostały one wzięte pod uwagę w dalszych badaniach. W dalszej części badań, zbiór przystosowany do wariantu Cargo Theft VRP, będzie oznaczany jako „**CTVRP X-nXX-kX**” - gdzie jako X oznaczono wartość specyficzną dla każdego z wariantów pierwotnych przedstawionych w Tabela 4.3.

4.3.1. Założenia

Implementacja algorytmów przedstawianych w literaturze [4] [13] została dostosowana do zasad zdefiniowanych dla wariantu CTVRP, tzn.:

- Koszt całkowity rozwiązania składa się z sumy kosztów danych rozwiązań częściowych (tj. suma kosztów rozwiązań częściowych dla wszystkich pojazdów wygenerowanych w ramach rozwiązania problemu) określonych równaniem (4).

$$S = \sum_{v=1}^{v_n} s_{cz}(v_n) \quad (4)$$

Gdzie, $s_{cz}(v_n)$ oznacza koszt rozwiązania częściowego dla danego pojazdu v_n wchodzącego w skład rozwiązania całkowitego (końcowego).

- Koszt rozwiązania cząstkowego - (tj. dla poszczególnego pojazdu v_n) określone równaniem (5) - składa się z sumy: kosztu podróży d_m (całkowity przebyty dystans przez pojazd) oraz średniego ryzyka kradzieży r_m dla wszystkich lokalizacji l_n , które pojazd musi odwiedzić [4].

$$s_{cz}(v_n) = \sum_{l=1}^{l_n} d_m + r_m \quad (5)$$

- Pojazdy mają swoją maksymalną ładowność, która nie może zostać przekroczona
- Pojazd musi rozpocząć i zakończyć podróż w punkcie początkowym jakim jest magazyn (koszty podróży „z” i „do” magazynu są uwzględniane)
- Magazyn jest definiowany jest tak samo, jak każda inna lokalizacja (położenie) lecz bez parametru ryzyka i zapotrzebowania (Tabela 3.1)
- Wszystkie lokalizacje odwiedzone tylko raz (nie dotyczy magazynu)

4.3.2. Przygotowanie zbioru danych dla wariantu Cargo Theft

Aby pozyskać dane testowe do wariantu Cargo Theft rozszerzono przedstawione zbiory testowe wariantu CVRP (Tabela 4.3) o informację na temat ryzyka panującego dla każdego wierzchołka grafu. Skorzystano w tym celu z generatora liczb pseudolosowych w przedziale $[0; 50]$ (na podstawie danych z opracowania naukowego bazującego na danych brazylijskiej policji [4] – nie odnotowano ryzyka na trasie wyższego niż 50% szans na utratę ładunku). Przygotowane dane zapisano w postaci zgodnej z opisaną strukturą w punkcie 3.1.1.3

4.4. Walidacja algorytmów

Po procesie tworzenia oprogramowania, należało sprawdzić poprawność działania zaimplementowanych algorytmów i zgodność z postawionym założeniami. Do tego celu posłużyły podstawowe zbiory danych dla wariantu CVRP, opisane w Tabeli 4.3. Aplikacja „*solver.exe*” została programowo pozbawiona możliwości przetwarzania i minimalizacji parametrów ryzyka kradzieży, co pozwoliło na uzyskanie pewnego rodzaju „odniesienia” względem referencyjnych danych z CVRP, jednak uznano, że taki zabieg jest niewystarczający, dlatego też wykorzystano program dotyczący badań nad heurystykami o nazwie „*HeuristicLab*”. Eksperyment przeprowadzono na zbiorze Augerata A-n33-k5, powtarzając wykonanie programu 128 razy.

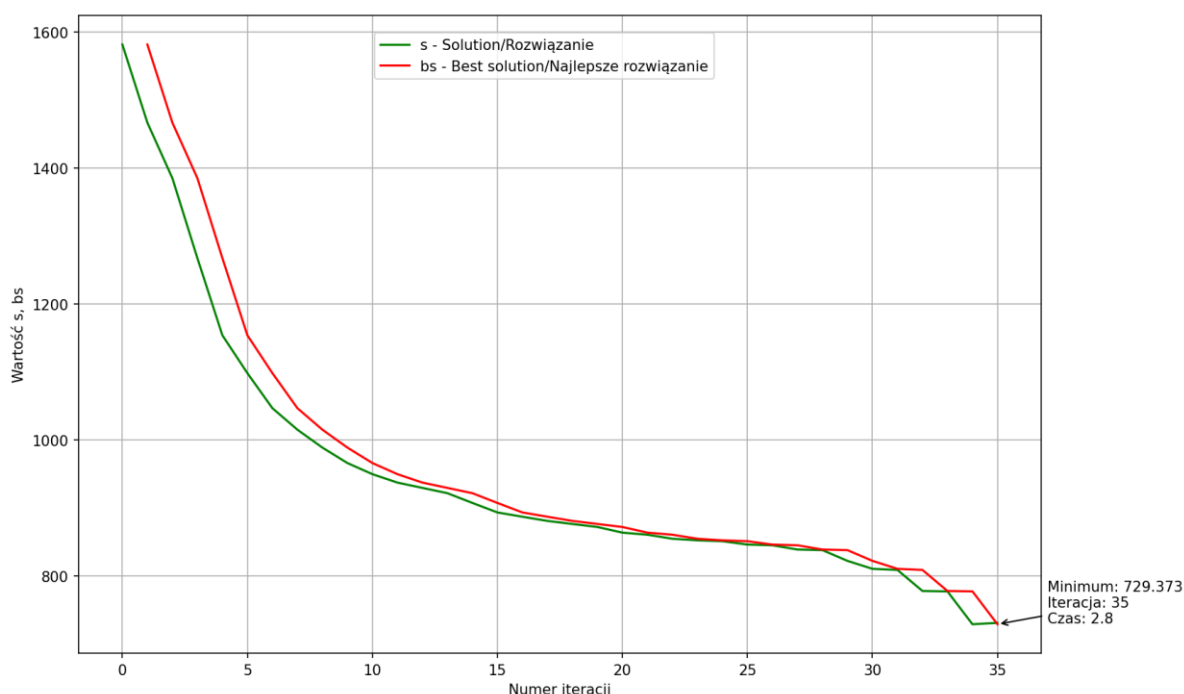
4.4.1. Program referencyjny HeuristicLab Optimizer

Program HeuristicLab Optimizer dostarcza kompleksowy zestaw narzędzi do badania problemów optymalizacyjnych - heurystycznych oraz ewolucyjnych. Umożliwia on rozwiązanie różnych rodzajów predefiniowanych problemów optymalizacyjnych, dowolnym spośród szerokiej listy algorytmów [19]. Program został opublikowany na zasadach wolnego oprogramowania w ramach licencji GNU General Public License w wersji 3, przez co jest to w pełni darmowy program. Aplikacja jest rozwijana od 2002 roku przez akademickie towarzystwo naukowe *Heuristic and Evolutionary Algorithm Laboratory* [19]. Zważając na długotrwałe utrzymanie się na rynku i ciągłe użytkowanie przez społeczność, uznaje się aplikację za wiarygodną w kontekście narzędzia referencyjnego dla stworzonego programu *solver.exe*. Zaimplementowane w programie algorytmy heurystyczne, nadają się do procesu walidacji rozwiązań aplikacji autorskiej. Program posiada predefiniowane zbiory danych opisane w punkcie 4.3.

4.4.2. Walidacja algorytmu lokalnego przeszukiwania.

- Losowy wynik programu „*solver*” (Rys. 4.1) - Otrzymane minimum: 729.373.
Iteracja 35. Czas: 2.8 s

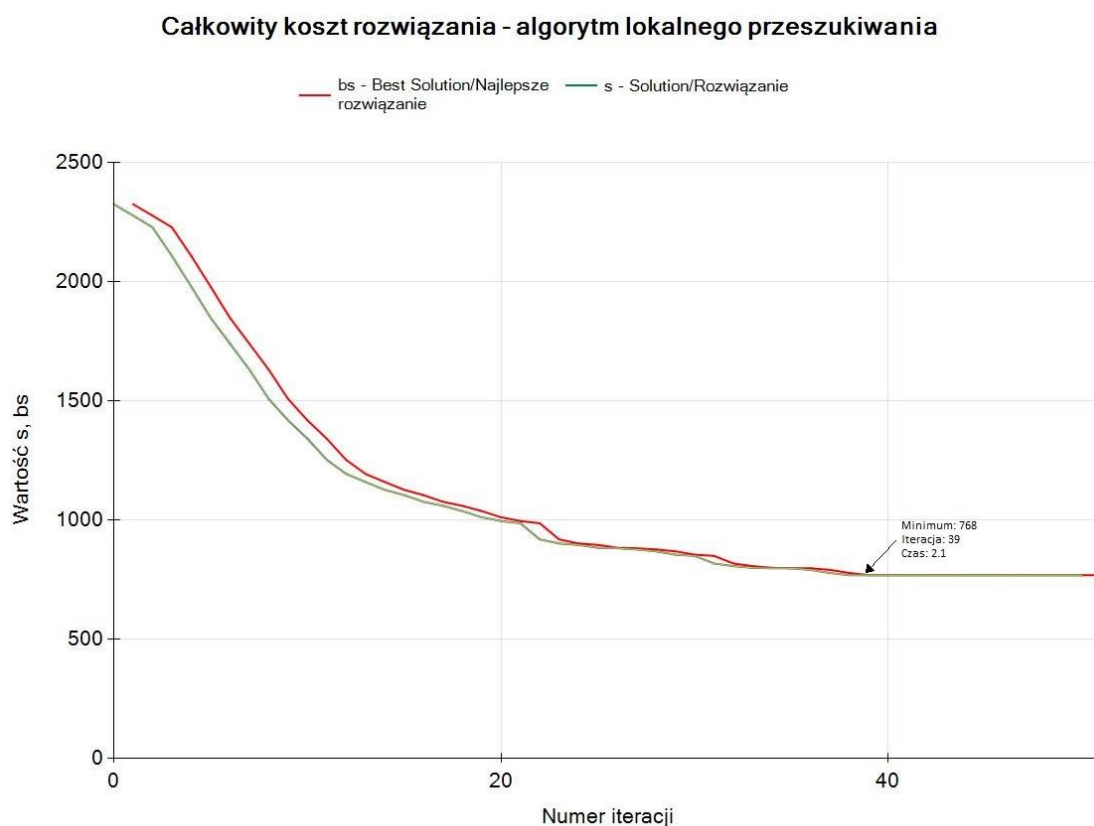
Całkowity koszt rozwiązania - algorytm lokalnego przeszukiwania



Rys. 4.1 Wykres całkowitych kosztów rozwiązania dla zbioru A-n33-k5. Rozwiązanie proponowane (*solver.exe*) – algorytm przeszukiwania lokalnego

Wykres przedstawia zmiany całkowitego kosztu rozwiązania bieżącego w stosunku do iteracji algorytmu. Wykres doskonale uwidacznia charakterystykę algorytmu przeszukiwania lokalnego, w którym musi ciągle następować poprawa wyniku (tj. w tym przypadku jego minimalizacja), aby algorytm nie skończył swojego działania. Przesunięcie pomiędzy linią ‘s’ a ‘bs’ wynika z momentu akceptacji rozwiązania bieżącego jako najlepsze.

- Losowy wynik programu „*HeuristicLab*” (Rys. 4.2) – Otrzymane minimum: 768.
Iteracja: 39.
Czas: 2.1 s



Rys. 4.2 Wykres całkowitych kosztów rozwiązania dla zbioru A-n33-k5. Rozwiązanie referencyjne (*HeuristicLab*) – algorytm przeszukiwania lokalnego

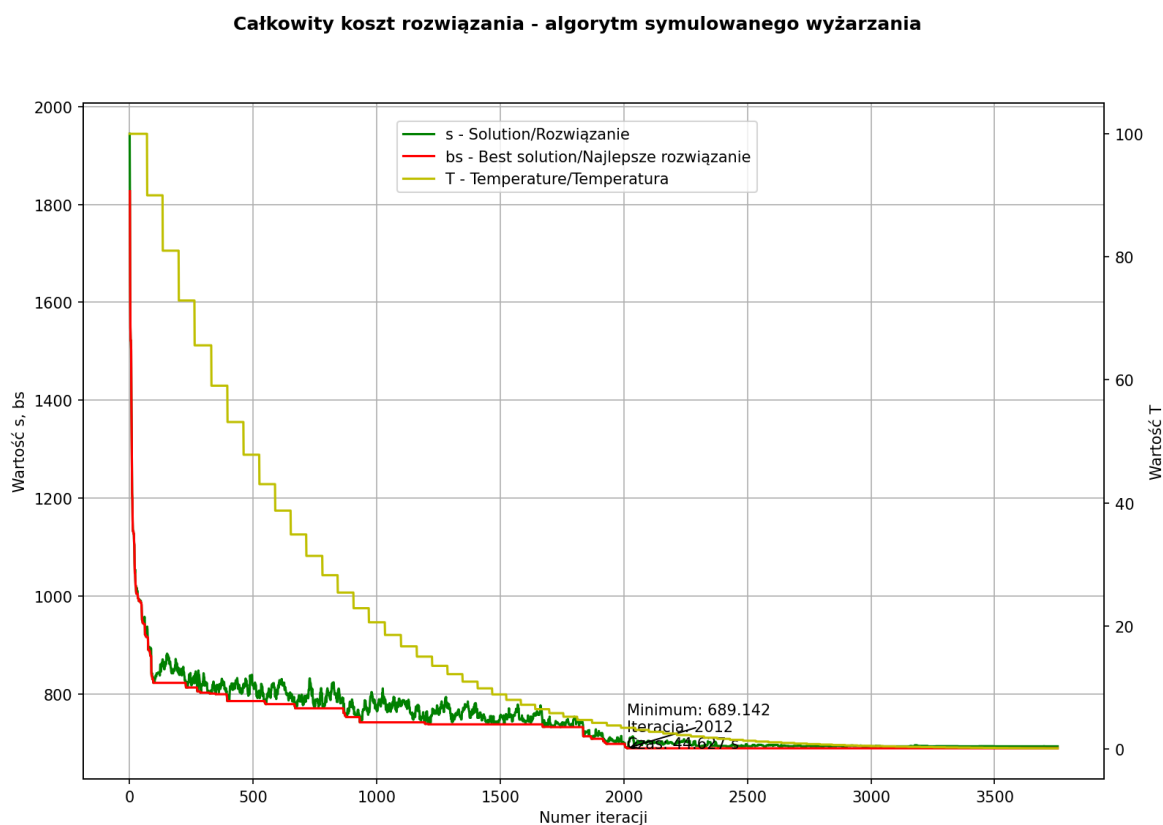
W przypadku działania programu *HeuristicLab* zauważono znacznie krótszy czas działania algorytmu. Wynika to ze zrównoleglenia metod generacji nowych rozwiązań w programie. Warto również zauważyć, że w przypadku tego programu, algorytm przeszukiwania lokalnego nie kończy się w momencie napotkania na rozwiązania gorsze. Jest to cecha aplikacji. Użytkownik może definiować maksymalną ilość iteracji. W przypadku programu *solver*, nie zaimplementowano algorytmów w sposób równoległy, stąd czas działania algorytmu jest dłuższy.

Wielokrotne próby wykazały, że zaimplementowany w programie *solver* algorytm przeszukiwania lokalnego działa poprawnie, zgodnie z oczekiwaniami i powtarzalnie w odniesieniu do programu „referencyjnego” jakim był *HeuristicLab*. W zaprezentowanym przykładzie (Rys. 4.1) otrzymane rozwiązanie minimalne okazało się lepsze od przykładu referencyjnego otrzymanego za pomocą programu *HeuristicLab* (Rys. 4.2). Warto nadmienić, iż w obydwu przypadkach nie osiągnięto minimum globalnego, lecz jest to wartość bliska temu minimum (wynosząca 661 dla zbioru *Set B-33-k5*).

4.4.3. Walidacja algorytmu symulowanego wyżarzania

Walidację przeprowadzono z wykorzystaniem parametrów opisanych w Tabeli 4.4.

- Losowy wynik programu „*solver*” (Rys. 4.3) - Otrzymane minimum: 689.142. Iteracja 2012. Czas: 44.6 s.

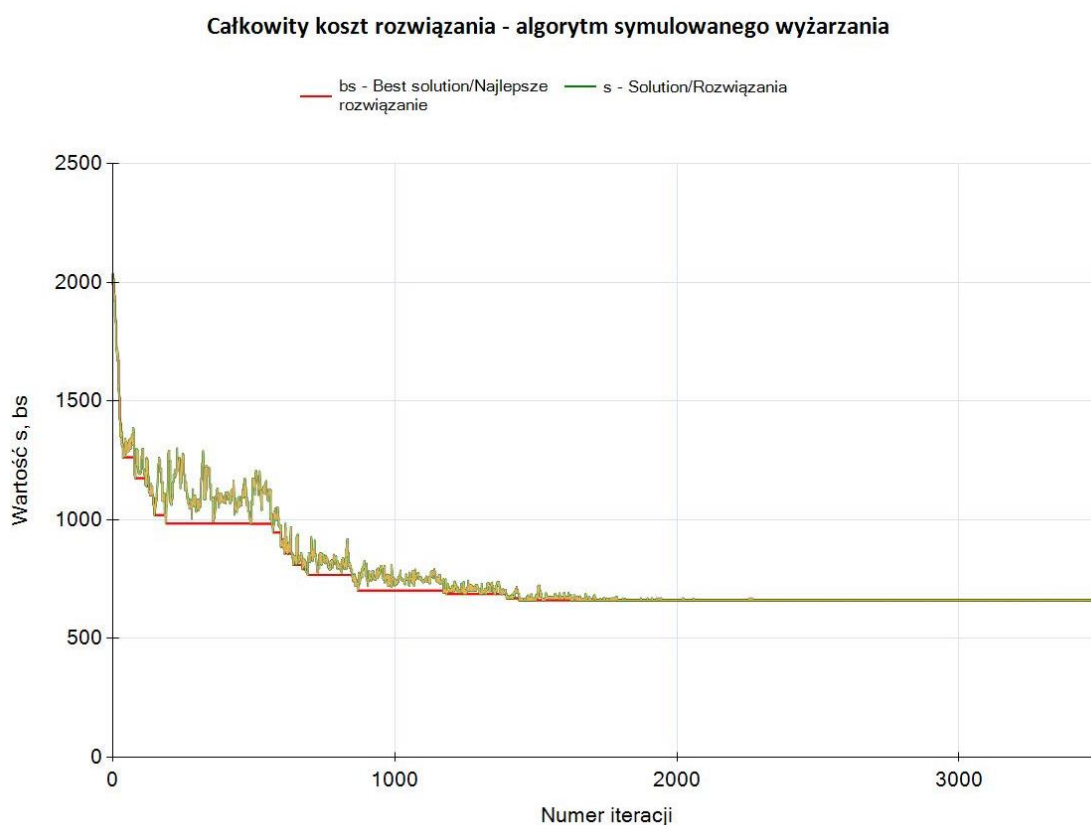


Rys. 4.3 Wykres całkowitych kosztów rozwiązania dla zbioru A-n33-k5. Rozwiązanie proponowane (*solver.exe*) – algorytm symulowanego wyżarzania

Otrzymane rozwiązania problemu A-n33-k5 w przypadku obydwu programów, są podobne. W rozwiązaniu otrzymanym za pomocą programu *HeuristicLab* (Rys. 4.4) osiągnięto minimum globalne. Rozwiązanie generowane przez program *solver.exe*

(Rys. 4.3) jest zbliżone do referencyjnego, w przypadku początku iteracji (dla wysokiej temperatury) optymalizacja zachodzi znacznie agresywniej niż w przypadku programu *HeuristicLab*, jednak te różnice wynikają z implementacji generatorów nowych rozwiązań (w przypadku *HeuristicLab* – analiza generatorów pod kątem implementacyjnym jest znacznie utrudniona). Element probabilistyczny algorytmu zachowuje się tak samo jak w przypadku referencyjnym – tzn. dla wysokich temperatur algorytmu, możliwa jest akceptacja dość niekorzystnych rozwiązań. Wraz ze spadkiem temperatury algorytmu, akceptacja rozwiązań gorszych jest znacznie bardziej rygorystyczna, a różnice pomiędzy wartością wylosowaną a aktualnym minimum są znacznie mniejsze od tych w wysokiej temperaturze. Podsumowując, w kontekście zasady działania stwierdza się, że algorytm symulowanego wyżarzania w programie *solver.exe* działa poprawnie uzyskując satysfakcjonujące wyniki.

- Losowy wynik programu „*HeuristicLab*” (Rys. 4.4) – Otrzymane minimum: 661. Iteracja: 1445. Czas: 40.1 s



Rys. 4.4 Wykres całkowitych kosztów rozwiązania dla zbioru A-n33-k5. Rozwiązanie referencyjne (*HeuristicLab*) – algorytm symulowanego wyżarzania

Tabela 4.4 Parametry algorytmu symulowanego wyżarzania wykorzystane do procesu walidacji

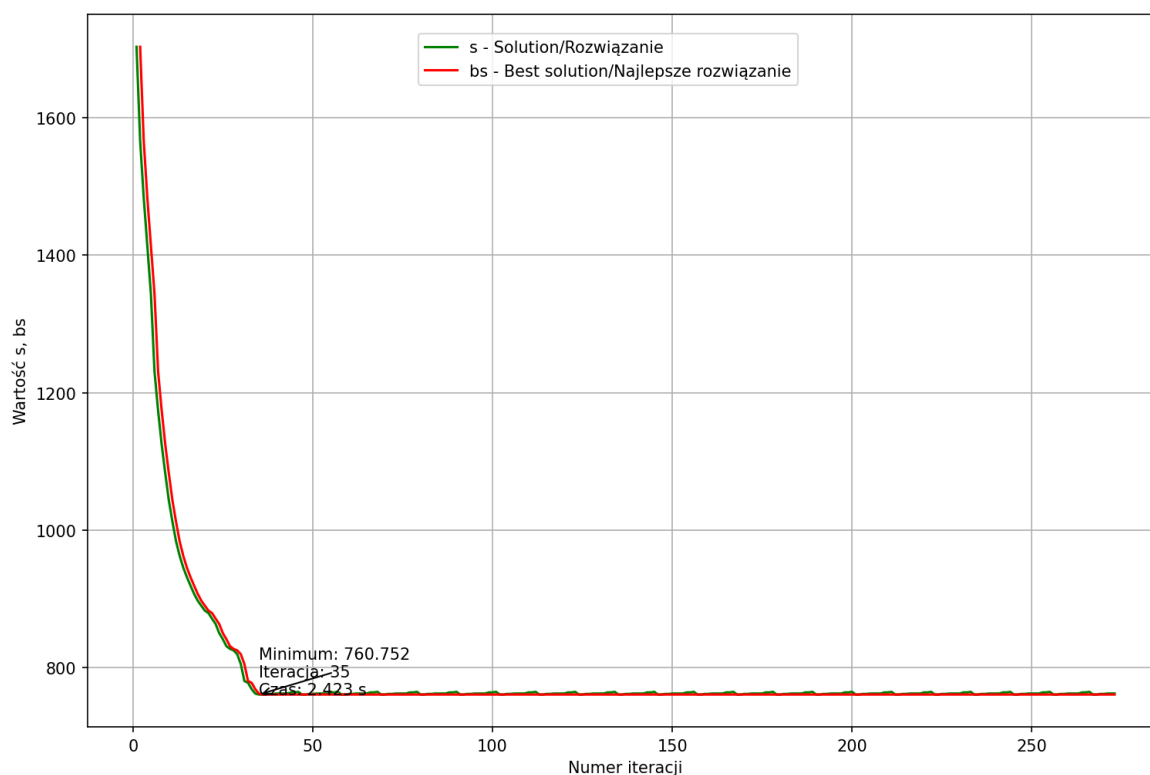
Parametry algorytmu	
Temperatura początkowa	100
Temperatura końcowa	0.1
Długość epoki (liczba iteracji pomiędzy spadkiem temperatury)	100
Współczynnik schładzania alfa	0.9

4.4.4. Walidacja algorytmu przeszukiwania tabu

Podczas procesu walidacji parametr **czasu trwania tabu** wynosił: **10** dla obu programów.

- Losowy wynik programu „*solver*” (Rys. 4.5) - Otrzymane minimum: 760.752.
Iteracja 35. Czas: 2.423 s

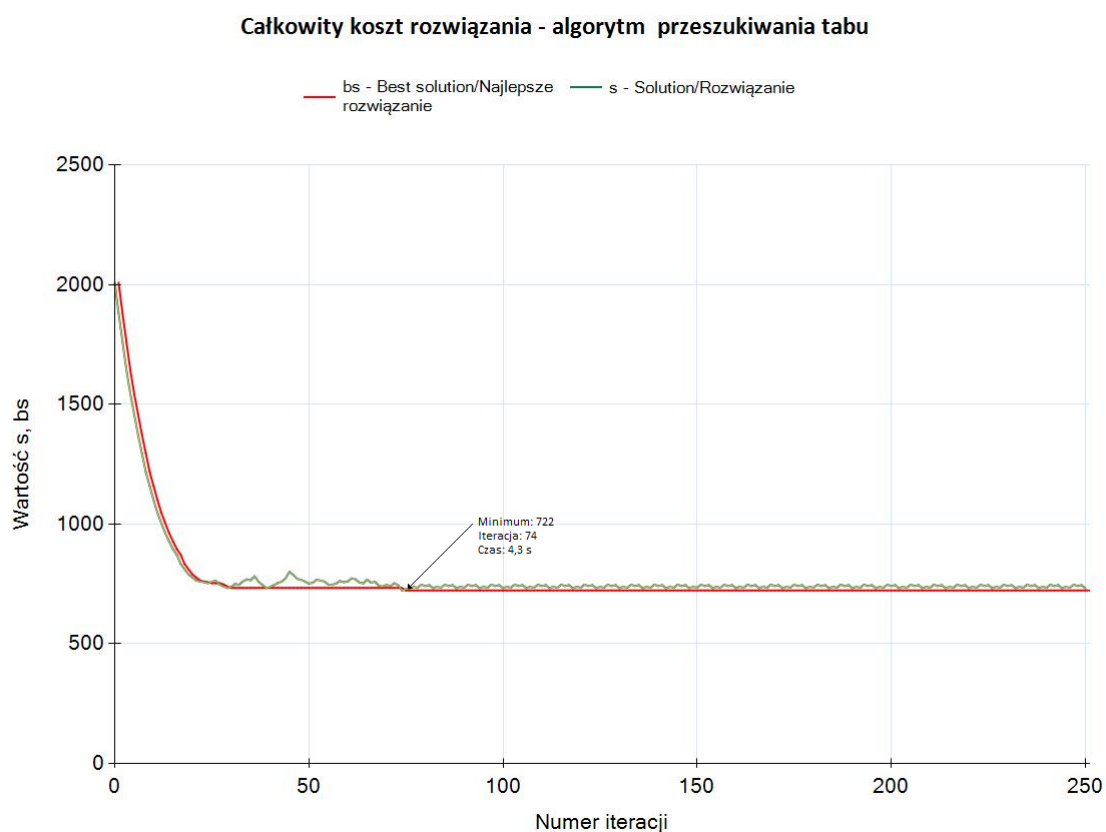
Całkowity koszt rozwiązania - algorytm przeszukiwania tabu



Rys. 4.5 Wykres całkowitych kosztów rozwiązania dla zbioru A-n33-k5. Rozwiązanie proponowane (*solver.exe*) – algorytm przeszukiwania tabu

Otrzymane rozwiązanie problemu A-n33-k5 wygenerowane zarówno przez program *solver.exe* (Rys. 4.5) jak i program *HeuristicLab* (Rys. 4.6) nie osiągnęło minimum globalnego. Rozwiązanie wygenerowane przez program *solver.exe* jest lepsze. Obydwa wykresy przedstawiają charakterystyczne zachowanie algorytmu przeszukiwania tabu, tzn. w końcowej fazie działania algorytmu - w którym to osiągnięto minimum lokalne - algorytm przeszukiwania tabu ma tendencję do „błądzenia” w związku z akceptowaniem lub odrzucaniem rozwiązań, które znajdują się na liście tabu. W przypadku programu *solver.exe* można zauważyć minimalne oscylacje wartości rozwiązania bieżącego pokrywające się z długością czasu trwania tabu. W kontekście rozwiązania referencyjnego, z programu *HeuristicLab* takie oscylacje również występują, są lepiej widoczne i swoją długością również pokrywają się z czasem trwania tabu. Na podstawie w/w opisu, stwierdza się, że algorytm przeszukiwania tabu działa poprawnie i zgodnie z oczekiwaniami.

- Losowy wynik programu „*HeuristicLab*” (Rys. 4.6) – Otrzymane minimum: 722. Iteracja: 74. Czas 4.3 s



Rys. 4.6 Wykres całkowitych kosztów rozwiązania dla zbioru A-n33-k5. Rozwiązanie referencyjne (*HeuristicLab*) – algorytm przeszukiwania tabu

4.5. Przeprowadzenie badań

Celem badań jest porównanie działania trzech algorytmów heurystycznych: przeszukiwania lokalnego, symulowanego wyżarzania oraz przeszukiwania tabu, w trzech seriach badań. Analiza algorytmów, odbywa się pod kątem jakościowym końcowych rozwiązań. Parametry podlegające ocenie:

- całkowity koszt rozwiązania,
- czas otrzymania najmniejszego (najlepszego) wyniku.

Przed przystąpieniem do właściwej fazy badawczej, należy sprawdzić jak dla algorytmu symulowanego wyżarzania wpływa zmiana współczynnika schładzania – tzw. alfa. Więcej szczegółów zostanie omówione w punkcie 4.5.1. Na podstawie uzyskanych wyników, zostanie wybrany jeden, wspólny współczynnik schładzania, który zostanie wykorzystywany w głównej fazie badawczej algorytmu (tzn. dla trzech serii badań - po **32 wywołania** programu każdy - współczynnik pozostanie bez zmian, po to, aby zaobserwować wpływ zmian innych parametrów algorytmu symulowanego wyżarzania na końcowy wynik optymalizacji).

W przypadku głównej fazy badawczej, algorytmy poddawane są zmianom parametrów w danej serii, gdzie **poszczególne serie badawcze składają się z 32 powtórzeń wykonania programu** głównego, opisanego w punkcie 3.1.1. Każda z serii badawczych, ma zostać przeprowadzona dla wszystkich zbiorów danych, które przedstawia Tabela 4.3, tak aby można było rozpatrzyć wpływ zmian objętości problemu na działanie algorytmów. Oznaczać to będzie, że analizie zostanie poddane 1440 próbek – tzn. instancji programu pokrywających wszystkie podane wcześniej kombinacje. Taka próba badawcza jest wystarczająca by na podstawie otrzymanych wyników przeanalizować charakterystykę działania wszystkich trzech algorytmów.

Wyniki przeprowadzonych badań zostały przedstawione w podrozdziale 4.6. Wartości poszczególnych wyników cząstkowych stanowią średnią ustanowioną z 32-krotnego wywołania programu.

4.5.1. Wyznaczanie współczynnika schładzania dla algorytmu symulowanego wyżarzania

Sięgając do literatury [4] [13], można spotkać stwierdzenie, iż współczynnik schładzania alfa, powinien mieścić się w przedziale $[0.8; 0.99]$. Nie istnieje konkretna metodyka wyznaczania wartości takiego współczynnika. Postanowiono więc przeprowadzić serię czterech eksperymentów, dla czterech pośrednich wartości parametru alfa (kolejno: 0.80, 0.85, 0.90, 0.95), mających na celu sprawdzenie, jak współczynnik schładzania będzie wpływał na końcowe rozwiązania problemu CTVRP. W tym celu, inne parametry algorytmu symulowanego wyżarzania nie mogą ulec zmianie. Badanie zostanie wykonane na jednym zestawie danych. Każda seria (nazywana także „eksperymentem”) składa się z wywołania programu 64 razy. Po tak przygotowanym badaniu, powinno być możliwe wyznaczenie odpowiedniego współczynnika (takiego, który może wpłynąć na końcowy wynik pozytywnie, tzn. zauważalnie zmniejszyć całkowity koszt). Tak przetestowany współczynnik, zostanie wykorzystany w głównej fazie badawczej.

Szczegółowe dane wejściowe algorytmu zostały opisane w punkcie 4.5.3, z kolei wyniki tych badań, zaprezentowano w punkcie 4.6.1.

4.5.2. Rodzaj algorytmu, a zestaw parametrów

Tabela 4.5 i Tabela 4.6 przedstawia kolejno wykorzystane parametry algorytmów - (symulowanego wyżarzania oraz przeszukiwania tabu. Algorytm lokalnego przeszukiwania jest bezparametrowy.

Tabela 4.5 Parametry algorytmu symulowanego wyżarzania

Zmienne parametry algorytmu		Wspólne parametry algorytmu
Seria I		Współczynnik schładzania
Temperatura początkowa	100	0.90
Temperatura końcowa	0.05	
Liczba iteracji w epoce	100	
Seria II		
Temperatura początkowa	1000	
Temperatura końcowa	0.05	
Liczba iteracji w epoce	100	
Seria III		
Temperatura początkowa	100	
Temperatura końcowa	0.05	
Liczba iteracji w epoce	1000	

Tabela 4.6 Parametry algorytmu przeszukiwania tabu

Seria badań	I	II	III
Czas trwania tabu	10	50	100
Maksymalna liczba iteracji	1000	5000	10000

4.5.3. Zmiana współczynnika alfa na końcowe wyniki działania algorytmu symulowanego wyżarzania

Na podstawie opisu z podrozdziału 4.5, przygotowano dane testowe do eksperymentu wyznaczenia współczynnika schładzania alfa, dla algorytmu symulowanego wyżarzania (Tabela 4.7). Wyznaczone zostały cztery przedziały wartości współczynnika schładzania. Pozostałe parametry pozostały bez zmian.

Tabela 4.7 Eksperymentalne parametry algorytmu symulowanego wyżarzania

Eksperyment	1	2	3	4
Temperatura początkowa	100	100	100	100
Temperatura końcowa	0.5	0.5	0.5	0.5
Liczba iteracji w epoce	100	100	100	100
Zbiór danych	A-n33-k5			
Współczynnik schładzania	0.80	0.85	0.90	0.95

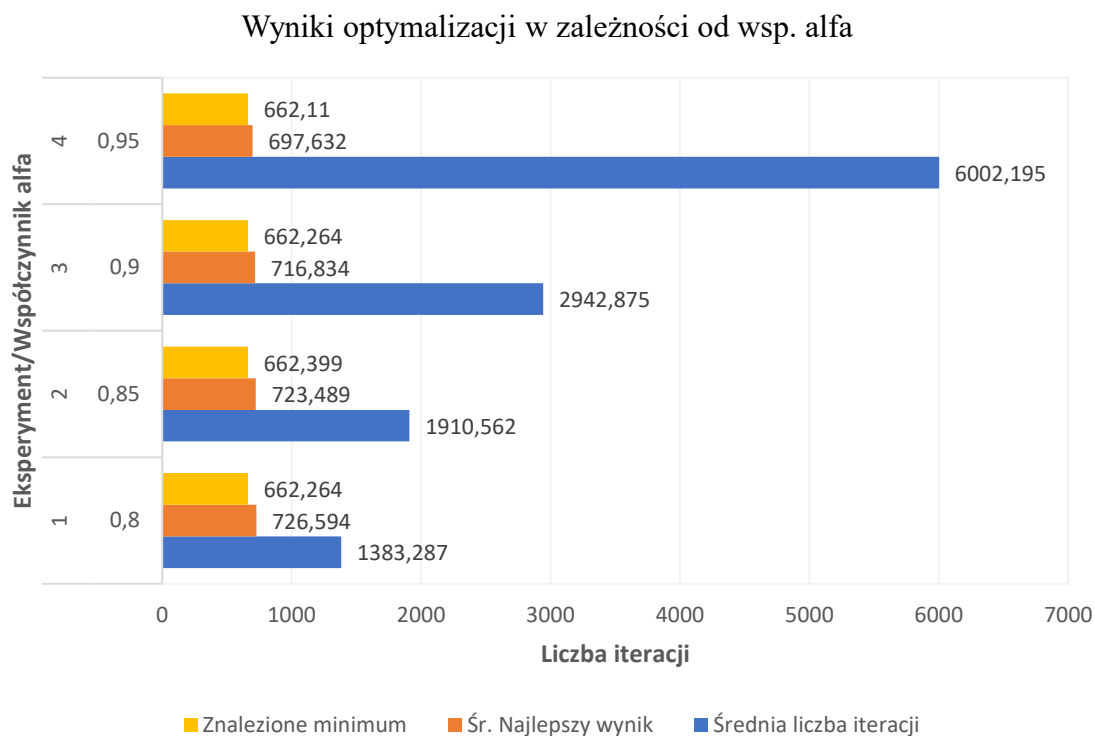
4.6. Uzyskane wyniki

4.6.1. Wyniki badania zależności zmian współczynnika alfa na końcowe wyniki działania algorytmu symulowanego wyżarzania

Przeprowadzono eksperyment, na podstawie założeń przedstawionych w punkcie 4.5.1 oraz danymi wejściowymi z Tabela 4.7. Otrzymano wyniki przedstawione w Tabela 4.8 oraz Rys. 4.7.

Tabela 4.8 Wyniki eksperymentu dot. współczynnika schładzania alfa

Eksperyment	1	2	3	4
Współczynnik schładzania	0.80	0.85	0.90	0.95
Średnia liczba iteracji	1383,287	1910,562	2942,875	6002,195
Śr. Najlepszy wynik	726,594	723,489	716,834	697,632
Znalezione minimum	662,264	662,399	662,264	662,110



Rys. 4.7 Wizualizacja wyników - badanie współczynnika alfa

Ponieważ inne parametry algorytmu jak temperatura początkowa, końcowa czy liczba iteracji były takie same dla każdego eksperymentu, zauważyć można wzrost liczby wykonanych iteracji wraz ze wzrostem wartości współczynnika alfa.

Poddając analizie otrzymane wyniki można stwierdzić, że wzrost wartości współczynnika alfa, bezpośrednio koreluje ze wzrostem ilości iteracji algorytmu, przy czym w otrzymaniu minimalnego wyniku nie wyróżnia się on na tle innych. Różnica występuje natomiast w średniej wartości rozwiązania. Im większy współczynnik schładzania, tym średnia wartość rozwiązania jest mniejsza (Tabela 4.8). Można uznać, że dla eksperymentu nr. 4 - współczynnik 0.95 - otrzymano najlepsze rezultaty, jednak konieczne należy zwrócić uwagę na wspomnianą już wcześniej liczbę wykonanych iteracji. Jest ona dwukrotnie wyższa, niż w przypadku wykorzystania współczynnika 0.90 (eksperyment nr. 3).

W związku z powyższym, dwukrotna liczba wykonanych iteracji jest zjawiskiem niepożądanym, biorąc pod uwagę nieznaczną poprawę wyników, dlatego też uznano, że wybrany zostanie współczynnik schładzania alfa, wykorzystany w eksperymencie trzecim (tj. 0.90).

4.6.2. Wyniki przeprowadzonych głównych badań

Jak wspomniano w podrozdziale 4.5, przeprowadzono badanie wpływu wykorzystanego algorytmu na końcowy wynik rozwiązania jednego z pięciu zbiorów danych. Badanie wykonano w trzech seriach, w których dla każdej zmieniano parametry algorytmów tak, aby sprawdzić zmianę w skuteczności pozyskiwanych wyników. W Tabeli 4.9, Tabeli 4.10 oraz Tabeli 4.11 przedstawiono rezultaty tychże badań. Spośród zaprezentowanych wyników można odczytać takie informacje jak:

- Typ – tj. typ algorytmu, który został uruchomiony w celach badań. Skrót LS oznacza algorytm lokalnego przeszukiwania, SA – symulowanego wyżarzania, natomiast skrót TS – oznacza algorytm przeszukiwania tabu.
- Ilość – tzn. ilość uruchomionych instancji programu (inaczej próbek)
- Średnia liczba wykonanych iteracji – jest to wartość średnia iteracji dla wszystkich 32 pomiarów w serii, podczas której algorytm znalazł rozwiązanie minimalne.
- Rozwiązanie początkowe – jest to wartość początkowa rozwiązania wygenerowana przez algorytm.
- Średnia wartość bs – jest to średnia wartość kosztu całkowitego, najlepszego rozwiązania dla wszystkich 32 pomiarów w serii podczas której, algorytm znalazł rozwiązanie minimalne.
- Minimum bs – jest to minimalna (najlepsza) wartość kosztu całkowitego rozwiązania znalezionego spośród wszystkich 32 pomiarów w serii.
- Średni czas odnalezienia bs – jest to średni czas, jaki był potrzebny do znalezienia rozwiązania minimalnego przez algorytm, dla 32 pomiarów w serii. Wyrażony w sekundach.
- Średni czas wykonywania algorytmu – jest to średni czas od początku do zakończenia działania algorytmu dla 32 pomiarów w serii. Wyrażony w sekundach.

4.6.3. Analiza wyników Serii I

Tabela 4.9 Wyniki badań - Seria I

Typ	Ilość	Średnia ilość iteracji	Rozwiązanie początkowe	Średnia wartość bs	Minimum bs	Średni czas odnalezienia bs [s]	Średni czas wykonywania algorytmu [s]
Zbiór danych: CTVRP A-n33-k5							
LS	32	27,28	2734,98	1776,04	1644,62	0,72	0,75
SA	32	4074,47	2884,61	1652,01	1596,26	11,93	20,92
TS	32	76,54	2788,18	1764,39	1666,12	0,94	2,35
Zbiór danych: CTVRP B-n51-k7							
LS	32	66,09	4804,55	2797,03	2422,72	3,01	4,24
SA	32	4313,97	5181,62	2581,99	2276,25	34,24	43,46
TS	32	124,34	4877,28	2790,16	2547,14	6,70	8,44
Zbiór danych: CTVRP B-n78-k10							
LS	32	104,88	4809,05	2033,56	1704,46	78,45	91,95
SA	32	4257,09	4937,79	1796,25	1648,98	81,26	90,44
TS	32	198,59	4748,85	1998,72	1752,63	104,43	209,47
Zbiór danych: CTVRP E-n101-k14							
LS	32	121,75	6417,81	4034,68	3876,98	291,16	297,89
SA	32	4054,63	6445,49	3876,15	3776,08	147,94	161,83
TS	32	222,63	6281,68	4038,52	3859,29	325,43	555,92
Zbiór danych: CTVRP M-n151-k12							
LS	32	212,53	9077,31	5474,72	5236,19	954,91	1035,88
SA	32	4216,41	9095,38	5200,73	5008,44	206,79	212,16
TS	32	361,66	9155,75	5439,69	5225,41	1759,18	3029,92

W Tabeli 4.9 można zauważyć, że wzrost objętości danych znacząco wpływa na czas działania algorytmu przeszukiwania lokalnego oraz przeszukiwania tabu. Im większa liczba miast do przetworzenia, tym czas działania tych dwóch algorytmów jest dłuższy. W przypadku małych zestawów danych (np. CTVRP A-n33-k5, lub CTVRP B-n51-k7) czas działania algorytmu przeszukiwania lokalnego jest najmniejszym spośród wszystkich trzech badanych. Można zwrócić uwagę na algorytm symulowanego wyżarzania, który dzięki akceptacji rozwiązań gorszych i tym samym przechodzeniu w inne rejony poszukiwań minimum, daje najlepsze z kolei rozwiązania – dla każdego rodzaju danych, jednak przy dość dużym koszcie czasowym, niezależnym od rozmiaru rozwiązywanego problemu. W przypadku zbiorów danych o znacznej objętości, algorytm symulowanego wyżarzania radzi sobie najlepiej, zarówno pod kątem dostarczania rozwiązań o najmniejszym koszcie, jak i czasowo – zarówno czas znalezienia minimalnego kosztu, jak i średniego czasu wykonywania algorytmu jest najmniejszy, często wielokrotnie mniejszy w stosunku do algorytmów przeszukiwania lokalnego lub tabu. Wartość średniego kosztu najlepszego rozwiązania dla algorytmu symulowanego wyżarzania – tj. Średnia wartość bs – jest lepsza

od wyników otrzymywanych przez algorytm przeszukiwania lokalnego i przeszukiwania tabu kolejno o: 4% oraz 5,81%.

4.6.4. Analiza wyników Serii II

Tabela 4.10 Wyniki badań - Seria II

Typ	Ilość	Średnia ilość iteracji	Rozwiązanie początkowe	Średnia wartość bs	Minimum bs	Średni czas odnalezienia bs [s]	Średni czas wykonywania algorytmu [s]
Zbiór danych: CTVRP A-n33-k5							
LS	32	33,44	2550,56	1748,65	1633,78	0,83	0,89
SA	32	5420,22	2878,65	1628,96	1596,26	21,35	34,03
TS	32	94,03	2601,95	1773,55	1652,98	0,75	2,48
Zbiór danych: CTVRP B-n51-k7							
LS	32	65,56	4710,30	2770,77	2549,70	2,88	4,05
SA	32	5721,38	5239,95	2529,12	2277,29	38,56	46,02
TS	32	125,63	4723,60	2763,74	2516,50	3,17	8,49
Zbiór danych: CTVRP B-n78-k10							
LS	32	106,16	4571,52	2010,82	1803,62	55,21	66,57
SA	32	5643,25	4960,97	1727,19	1650,79	82,29	90,00
TS	32	184,56	4686,64	2019,69	1845,65	73,42	143,00
Zbiór danych: CTVRP E-n101-k14							
LS	32	118,03	6209,58	4061,02	3878,19	211,15	221,79
SA	32	5382,69	6367,63	3839,89	3747,63	153,05	170,06
TS	32	218,97	6272,09	4049,94	3879,07	256,49	458,73
Zbiór danych: CTVRP M-n151-k12							
LS	32	205,88	9076,46	5487,21	5289,13	1565,93	1680,43
SA	32	5543,88	9202,59	5156,63	5023,19	252,95	261,55
TS	32	331,13	9178,74	5461,67	5265,93	1664,70	2653,46

W przeprowadzonej drugiej serii badań, algorytmy przeszukiwania tabu oraz symulowanego wyżarzania uległy zmianie:

- Wydłużono czas trwania tabu dla algorytmu przeszukiwania tabu do wartości 50 iteracji.
- Wydłużono czas działania algorytmu symulowanego wyżarzania poprzez zwiększenie temperatury początkowej z wartości 100 na 1000.

Zaobserwowano wyraźną poprawę w średniej wartości najlepszego rozwiązania dla algorytmu symulowanego wyżarzania. W odniesieniu do wyników otrzymywanych przez algorytm przeszukiwania lokalnego, uzyskano wyniki lepsze o średnio 6,46%, a w odniesieniu do algorytmu przeszukiwania tabu o 6,84%. Zauważalna jest również zwiększona średnia liczba iteracji dla tego algorytmu, przy czym – dla małych zestawów danych (CTVRP A-n33-k5 lub CTVRP B-n51-k7) jest to nieakceptowalnie dłuższy czas

w stosunku do algorytmu przeszukiwania lokalnego lub tabu Algorytm symulowanego wyżarzania działa wyraźnie krócej (dając jednocześnie lepszy rezultat) na dużych zestawach danych (CTVRP E-n101-k14, CTVRP M-n151-k12).

W przypadku modyfikacji parametrów algorytmu przeszukiwania tabu, nastąpiło dość ciekawe zjawisko, mianowicie zwiększenie czasu trwania tabu, doprowadziło do ogólnego pogorszenia średniego czasu wykonywania algorytmu (algorytm wykonywał się dłużej). Również nie uległy poprawie (zmniejszeniu) koszty rozwiązania – wartość średnia jak i minimum najlepszego rozwiązania pozostały na bardzo zbliżonym poziomie jak dla wyników przeprowadzonych podczas serii pierwszej (Tabela 4.9).

4.6.5. Analiza wyników Serii III

Tabela 4.11 Wyniki badań - Seria III

Typ	Ilość	Średnia ilość iteracji	Rozwiązanie początkowe	Średnia wartość bs	Minimum bs	Średni czas odnalezienia bs [s]	Średni czas wykonywania algorytmu [s]
Zbiór danych: CTVRP A-n33-k5							
LS	32	31,56	2660,03	1790,79	1662,23	0,76	0,84
SA	32	26780,00	2952,56	1642,96	1596,11	28,18	193,63
TS	32	93,81	2614,89	1764,15	1621,84	0,78	2,57
Zbiór danych: CTVRP B-n51-k7							
LS	32	66,66	4621,14	2753,02	2454,76	3,18	4,34
SA	32	28809,97	5429,03	2494,30	2223,40	120,60	257,85
TS	32	123,03	4760,90	2788,00	2548,70	3,23	8,45
Zbiór danych: CTVRP B-n78-k10							
LS	32	107,25	4520,21	1986,70	1825,88	61,54	71,11
SA	32	28525,84	4948,00	1706,04	1595,14	298,87	502,41
TS	32	185,97	4653,93	1972,92	1800,92	77,84	149,40
Zbiór danych: CTVRP E-n101-k14							
LS	32	115,78	6150,31	4049,38	3871,79	215,10	222,21
SA	32	27031,59	6404,03	3796,53	3709,30	465,22	884,57
TS	32	222,88	6144,24	4022,77	3878,15	262,03	457,63
Zbiór danych: CTVRP M-n151-k12							
LS	32	209,69	9025,97	5471,40	5191,62	1588,20	1703,19
SA	32	27761,19	9301,04	5040,04	4925,20	839,95	1208,36
TS	32	336,97	9010,85	5444,33	5277,10	1773,43	2746,80

W trzeciej serii badań (Tabela 4.11), dokonano modyfikacji algorytmów w następujący sposób:

- Dla algorytmu przeszukiwania tabu - wydłużono czas trwania tabu do wartości 100 iteracji.

- Dla algorytmu symulowanego wyżarzania, przywrócono temperaturę początkową do wartości 100, przy jednoczesnym zwiększeniu czasu trwania epoki z 100 do 1000 iteracji.

Zaobserwowano niekorzystny czas działania algorytmu symulowanego wyżarzania, dla zestawu danych o małej objętości. Cechuje się to tym, iż ten algorytm nie jest zależny od wielkości danych na jakich operuje. Wydłużony został czas trwania algorytmu w jednej epoce, tak więc poza zwiększoną ogólną średnią liczbę wykonanych iteracji dla algorytmu, widoczna jest również poprawa osiąganego minimum - Średniej wartości bs (Tabela 4.11) – średnio o 7.76% lepszy wynik niż analogiczne rezultaty uzyskane przez algorytm przeszukiwania lokalnego oraz średnio o 8,16% lepiej w odniesieniu do wyników uzyskiwanych za pomocą algorytmu przeszukiwania tabu.

4.6.6. Graficzna reprezentacja wyników – interpretacja

4.6.6.1. Algorytm przeszukiwania lokalnego

Na Rys. 4.8 dotyczącym algorytmu przeszukiwania lokalnego oraz Rys. 4.12 dotyczącym algorytmu przeszukiwania tabu, zauważalna jest wyraźna zależność czasu trwania algorytmu od wielkości przetwarzanych danych. Dzieje się tak dlatego, że obydwa algorytmy w swojej naturalnej formie (a taka została zaimplementowana), mają za zadanie przeszukiwać całe sąsiedztwo rozwiązania bieżącego, w poszukiwaniu wartości lepszych. Oznacza to tyle, iż w przypadku rozwiązań o wielkości rzędu 33 lokalizacji (czyli tzw. małych zbiorów danych), w każdej iteracji algorytmu, musi nastąpić przeszukanie sąsiedztwa o wymiarach 33×32 lokalizacji (magazyn brany jest pod uwagę tylko raz). Dokładnie tak samo odbywa się przeszukiwanie sąsiedztwa dla dużego zbioru danych z 151 lokalizacjami. W każdej iteracji, wymagane jest przeszukanie przestrzeni 151×150 możliwych potencjalnie lepszych rozwiązań. To skutkuje rosnącym czasem trwania algorytmu. Z uwagi na fakt, iż algorytm przeszukiwania lokalnego jest algorytmem bezparametrowym, można zauważyć, że na przestrzeni przeprowadzonych trzech serii badań, dostarczone przez niego wyniki nie różnią się znacząco od siebie - Rys. 4.9.

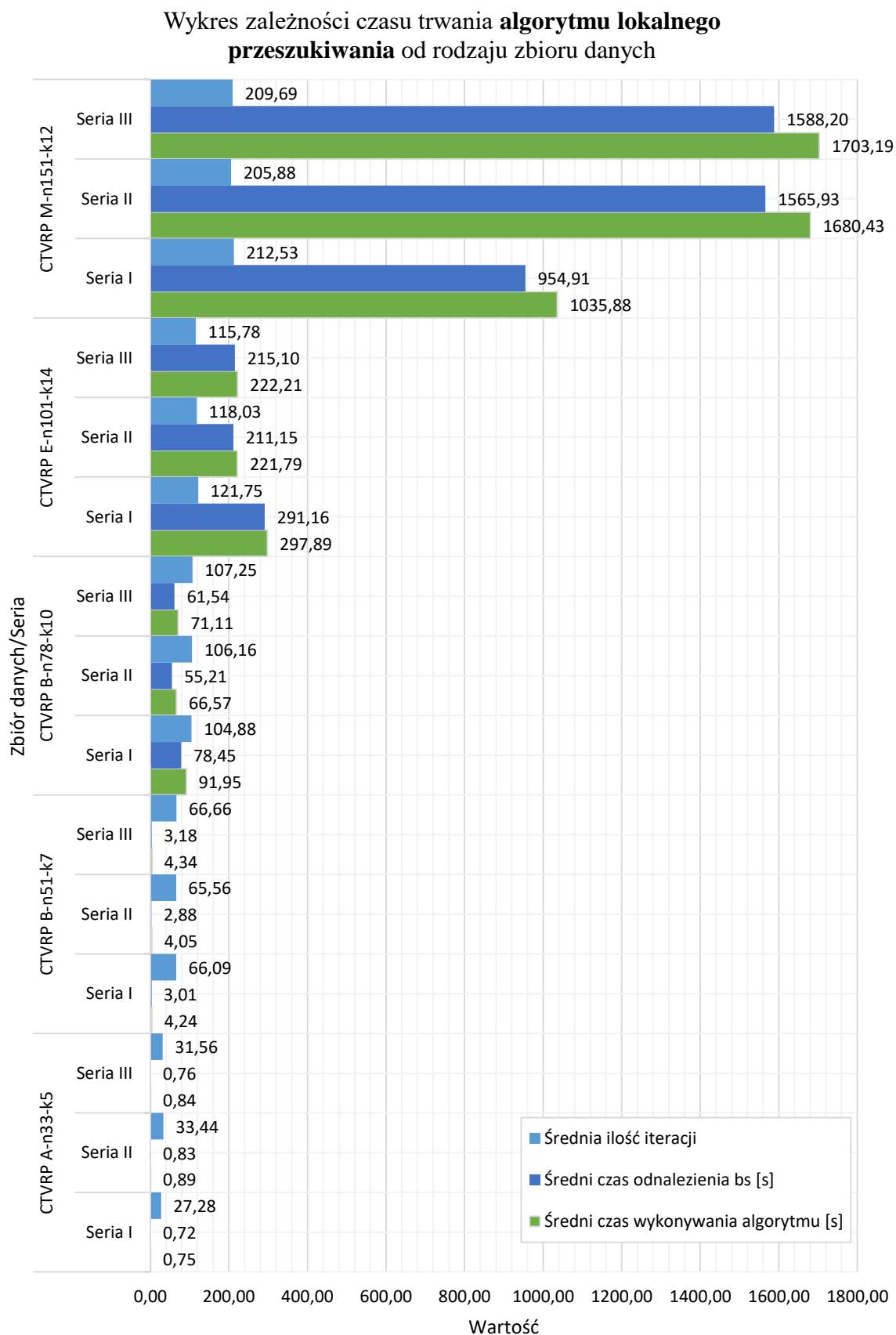
4.6.6.2. Algorytm symulowanego wyżarzania

Na Rys. 4.10 dla algorytmu symulowanego wyżarzania, mamy do czynienia z przeszukiwaniem losowej próbki sąsiedztwa, dlatego też czas działania nie jest tak bardzo uzależniony od rozmiaru sąsiedztwa jak w przypadku pozostałych algorytmów. Wadą tego rozwiązania jest natomiast fakt, że wymagane jest od użytkownika odpowiednie dobranie parametrów algorytmu w taki sposób, aby została przeszukana względnie duża przestrzeń rozwiązań, gdyż może zdarzyć się tak, że algorytm symulowanego wyżarzania nie będzie w stanie w ciągu zaledwie kilku iteracji (w sposób *de facto* losowy) dostarczyć rozwiązania równie dobrego co algorytmy o dłuższym czasie trwania.

Przedstawiając w sposób graficzny na Rys. 4.11 wyniki badań wartości średniej całkowitego kosztu rozwiązania, można zaobserwować, iż zwiększenie temperatury początkowej (Seria II) jak i zwiększenie czasu trwania epoki algorytmu (Seria III) wpłynęły pozytywnie na dostarczone rozwiązania przez algorytm. Rozpatrując zmianę względem serii referencyjnej – tj. Serii I, w przypadku zwiększenia temperatury początkowej – Seria II – uzyskane wyniki są średnio lepsze o ok. 1%, a w przypadku zwiększenia czasu trwania epoki – Seria III – wyniki średniej wartości bs były lepsze o ok. 3% (np. dla zbioru CTVRP M-n151-k12). Trzeba mieć na uwadze, że dla algorytmu symulowanego wyżarzania, generacja nowych rozwiązań powstaje wskutek pewnej losowości wyboru następnego sąsiada rozwiązania bieżącego dlatego też, można spodziewać się nieznacznego pogorszenia wyników, jak miało to miejsce na zbiorze danych CTVRP A-n33-k5.

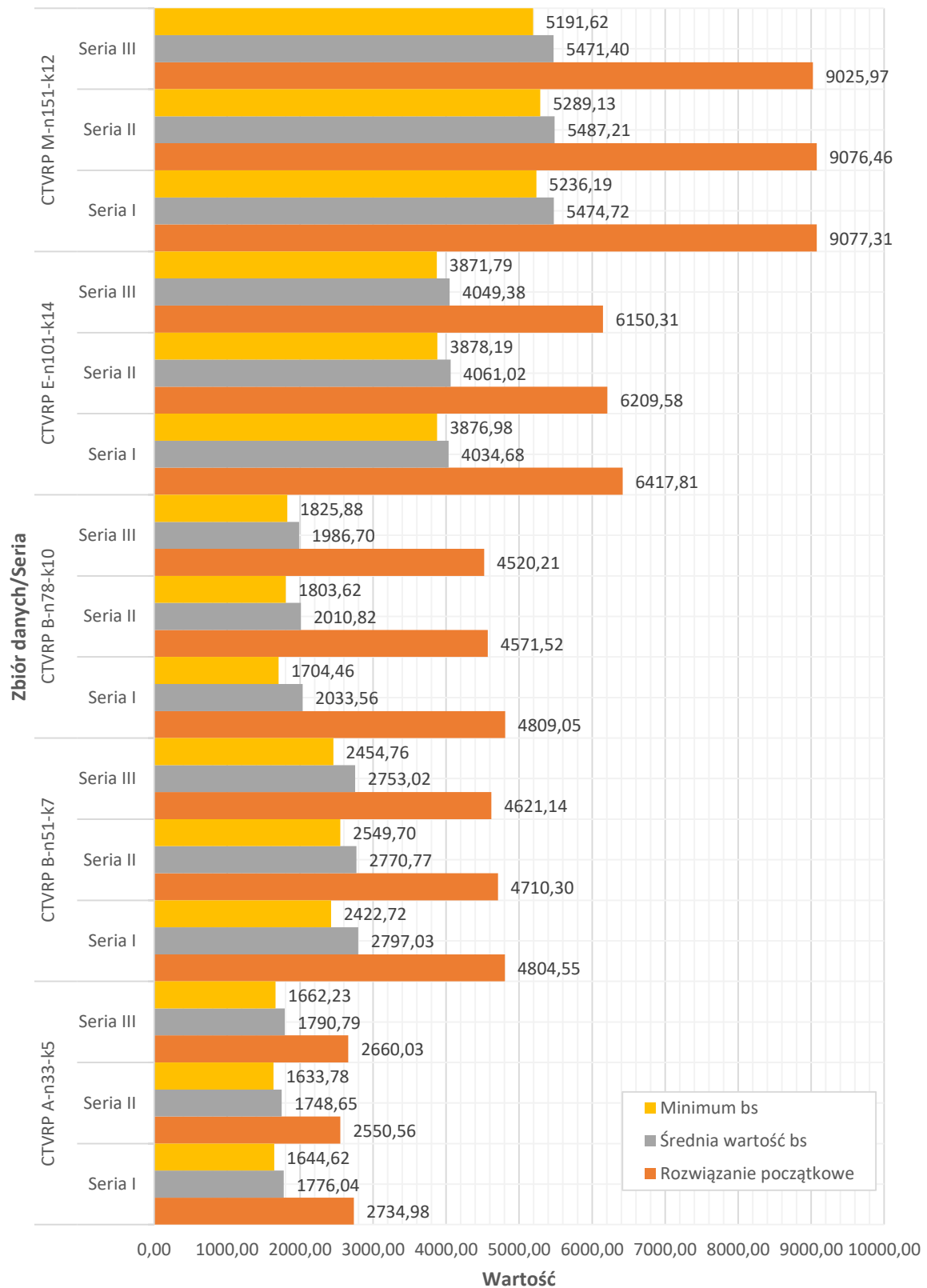
4.6.6.3. *Algorytm przeszukiwania tabu*

Analizując wykres przedstawiony na Rys. 4.12 można dostrzec zjawisko trudne do interpretacji tylko na podstawie Tabela 4.11. Dla algorytmu przeszukiwania tabu, w przypadku trzeciej serii badań, gdzie zwiększeniu uległ czas trwania zakazów, średni czas odnalezienia rozwiązania najlepszego uległ skróceniu – tylko dla danych o dużej objętości (CTVRP E-n101-k14 oraz CTVRP M-n151-k12). Dla pozostałych mniejszych zestawów danych, wyniki prezentują się podobnie co w przypadku pozostałych serii badań. Na podstawie Rys. 4.13 nie można stwierdzić, aby zwiększenie czasu trwania tabu wpływało znacząco (zarówno negatywnie jak i pozytywnie) na uzyskiwany koszt końcowy rozwiązania.

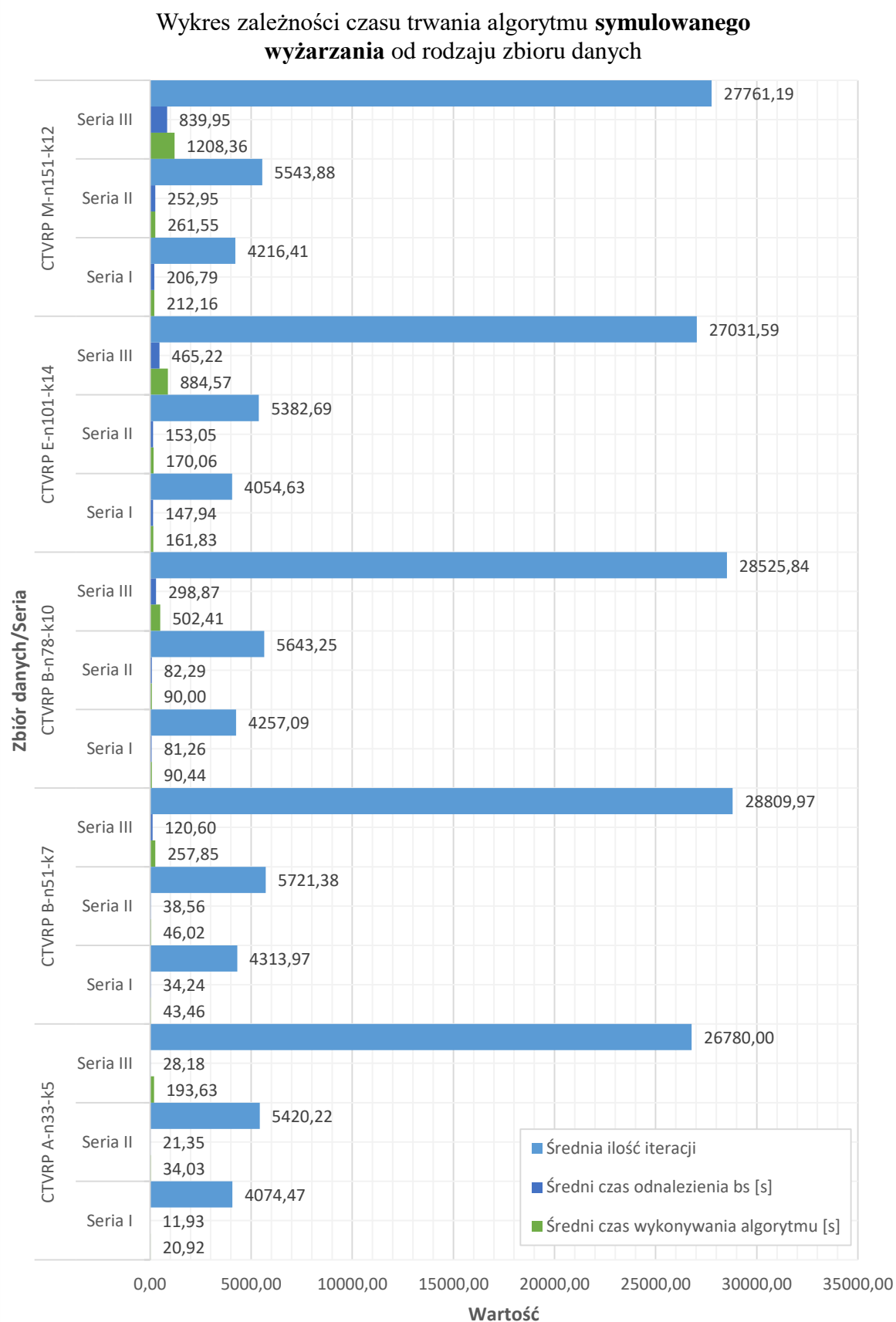


Rys. 4.8 Zależność czasu trwania algorytmu lokalnego przeszukiwania od rodzaju danych wejściowych

Wykres zależności otrzymanych rozwiązań od rodzaju zbioru danych dla algorytmu **lokalnego przeszukiwania**

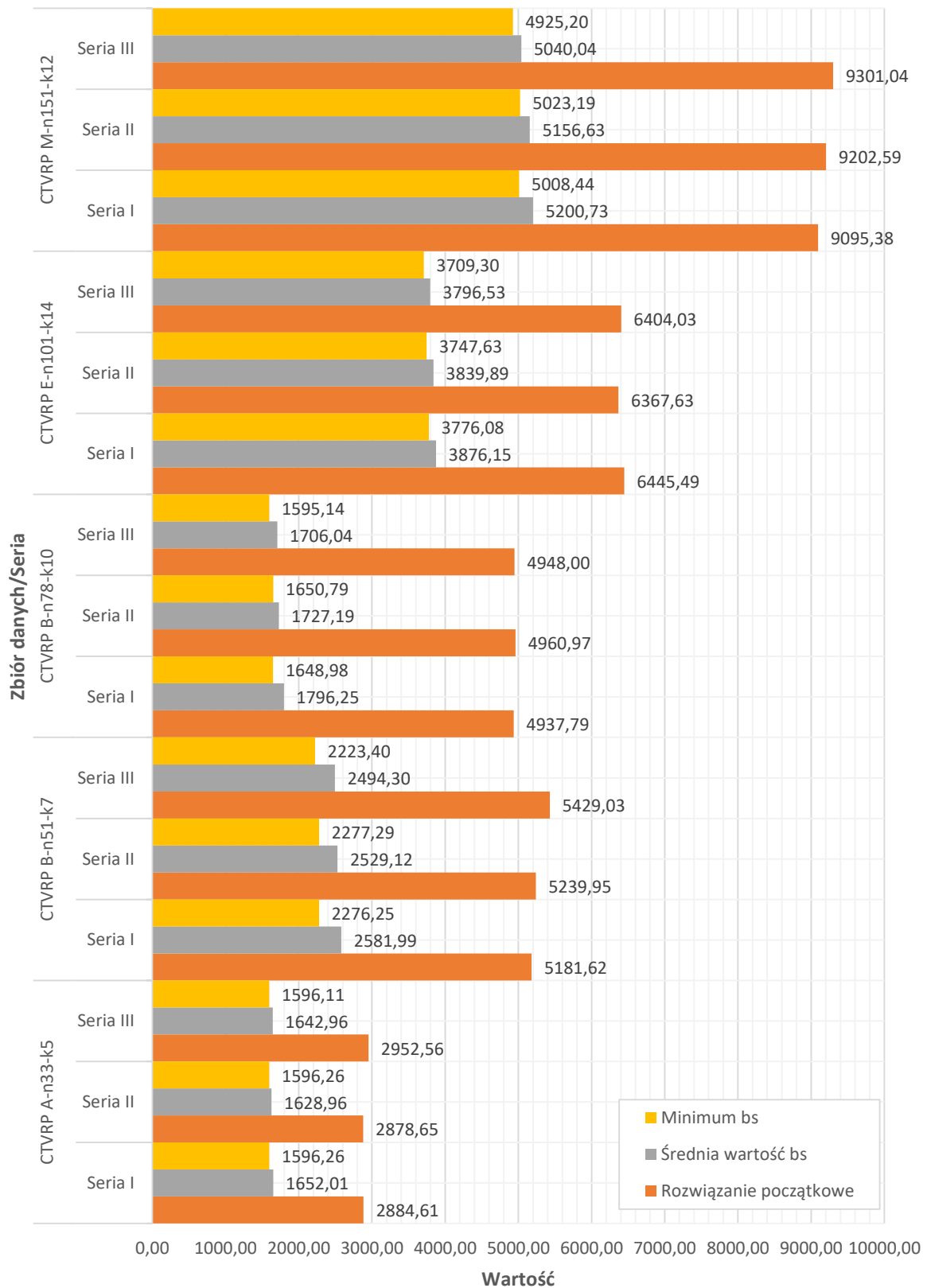


Rys. 4.9 Zależność otrzymanych rozwiązań od rodzaju danych wejściowych dla algorytmu lokalnego przeszukiwania



Rys. 4.10 Zależność czasu trwania algorytmu symulowanego wyżarzania od rodzaju danych wejściowych

Wykres zależności otrzymanych rozwiązań od rodzaju zbioru danych
dla **algorytmu symulowanego wyżarzania**

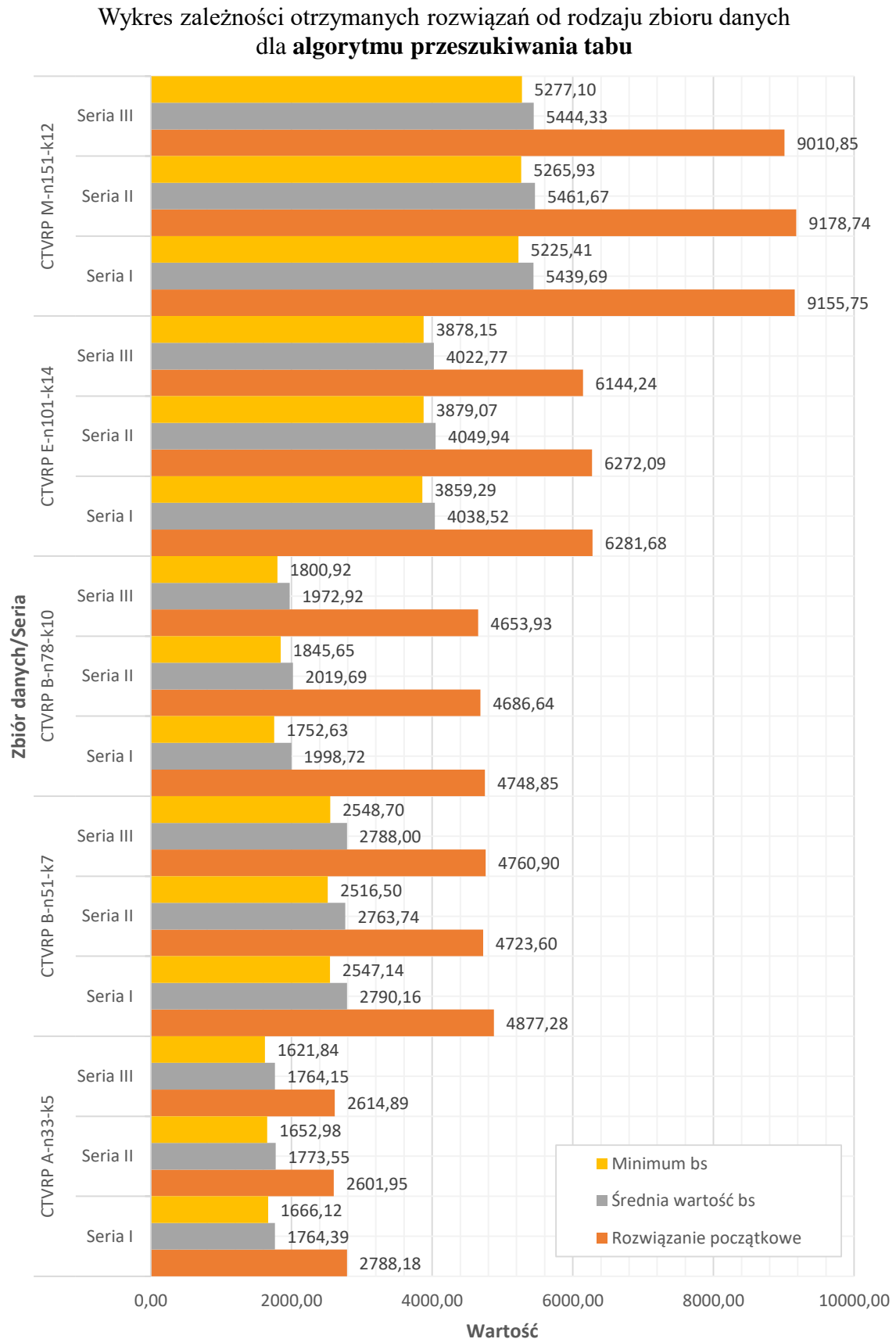


Rys. 4.11 Zależność otrzymanych rozwiązań od rodzaju danych wejściowych dla
algorytmu symulowanego wyżarzania

The chart displays performance metrics for three series (I, II, III) across five different data sets. The metrics are: Średnia ilość iteracji (Average number of iterations), Średni czas odnalezienia bs [s] (Average time to find the best solution [s]), and Średni czas wykonywania algorytmu [s] (Average algorithm execution time [s]).

Zbiór danych/Seria	Seria	Średnia ilość iteracji	Średni czas odnalezienia bs [s]	Średni czas wykonywania algorytmu [s]
CTVRP M-n151-k12	Seria III	336,97	1773,43	2746,80
	Seria II	331,13	1664,70	2653,46
	Seria I	361,66	1759,18	3029,92
CTVRP E-n101-k14	Seria III	222,88	262,03	457,63
	Seria II	218,97	256,49	458,73
	Seria I	222,63	325,43	555,92
CTVRP B-n78-k10	Seria III	185,97	77,84	149,40
	Seria II	184,56	73,42	143,00
	Seria I	198,59	104,43	209,47
CTVRP B-n51-k7	Seria III	123,03	3,23	8,45
	Seria II	125,63	3,17	8,49
	Seria I	124,34	6,70	8,44
CTVRP A-n33-k5	Seria III	93,81	0,78	2,57
	Seria II	94,03	0,75	2,48
	Seria I	76,54	0,94	2,35

Rys. 4.12 Zależność czasu trwania algorytmu przeszukiwania tabu od rodzaju danych wejściowych



Rys. 4.13 Zależność otrzymanych rozwiązań od rodzaju danych wejściowych dla algorytmu przeszukiwania tabu

4.6.7. Jakość optymalizacji

Interpretując uzyskane wyniki, postanowiono również sprawdzić badane algorytmy pod kątem uzyskiwanego zysku (Tabela 4.12). Zysk z przeprowadzonej optymalizacji można wyrazić wzorem (6).

$$Profit = \frac{(Cost_{init} - Cost_{best})}{Cost_{best}} * 100\% \quad (6)$$

Gdzie, $Cost_{best}$ oznacza całkowity koszt rozwiązania najlepszego, a $Cost_{init}$ oznacza całkowity koszt rozwiązania początkowego.

Tabela 4.12 Średnia uzyskana korzyść rozwiązania problemów CTVRP

Zbiór danych: CTVRP A-n33-k5			
Seria badań	Algorytm LS	Algorytm SA	Algorytm TS
Seria I	34,03%	40,01%	35,46%
Seria II	36,29%	41,31%	35,23%
Seria III	35,80%	40,49%	36,13%
Zbiór danych: CTVRP B-n51-k7			
Seria badań	Algorytm LS	Algorytm SA	Algorytm TS
Seria I	42,16%	46,04%	42,56%
Seria II	41,86%	48,09%	43,20%
Seria III	43,64%	49,12%	43,31%
Zbiór danych: CTVRP B-n78-k10			
Seria badań	Algorytm LS	Algorytm SA	Algorytm TS
Seria I	56,91%	61,42%	57,29%
Seria II	57,21%	62,82%	57,09%
Seria III	57,08%	63,43%	57,59%
Zbiór danych: CTVRP E-n101-k14			
Seria badań	Algorytm LS	Algorytm SA	Algorytm TS
Seria I	34,85%	37,98%	34,94%
Seria II	34,53%	38,14%	35,19%
Seria III	34,88%	38,75%	34,83%
Zbiór danych: CTVRP M-n151-k12			
Seria badań	Algorytm LS	Algorytm SA	Algorytm TS
Seria I	39,07%	41,63%	39,74%
Seria II	38,94%	42,44%	38,88%
Seria III	39,38%	43,81%	39,22%

Przedstawiona wcześniej Tabela 4.12 potwierdza opisane w punktach od 4.6.3 do 4.6.5 efekty wprowadzenia zmian do algorytmów symulowanego wyżarzania oraz przeszukiwania tabu na wartość końcowego kosztu rozwiązania. Można zauważyć, że każdy z algorytmów dostarcza rozwiązanie lepsze od pierwotnie wygenerowanego, zatem zaimplementowana część optymalizacyjna działa poprawnie.

Rozdział 5

Podsumowanie

W ramach niniejszej pracy przeprowadzono dogłębne porównanie trzech różnych heurystyk – algorytmu przeszukiwania lokalnego, symulowanego wyżarzania oraz przeszukiwania tabu dla problemu trasowania pojazdów w wariacie z występującym ryzykiem kradzieży ładunku – Cargo Theft VRP. Badanie miało na celu sprawdzenie jakości rozwiązań w zależności od każdego z trzech wybranych algorytmów. Proces badawczy przeszedł pomyślnie, a na podstawie pozyskanych trzech serii wyników dla każdego z pięciu różnorodnych zestawów danych, zaobserwowano, że algorytm symulowanego wyżarzania dostarczał najlepszych wyników w kontekście najniższego całkowitego kosztu rozwiązania, jednak nie można powiedzieć tego samego, w odniesieniu do niewielkich zestawów danych, w których bardzo dobrze radził sobie algorytm przeszukiwania lokalnego jak i przeszukiwania tabu. Obecny stopień zaawansowania dla zaimplementowanego algorytmu symulowanego wyżarzania nie uwzględnia automatycznego doboru parametrów temperatury, przez co arbitralny wybór parametrów początkowych, ma wpływ na czas działania algorytmu, który w przypadku małych zestawów danych, jest często zbyt duży, co jest nieopłacalne gdyby porównać go do algorytmu przeszukiwania lokalnego, który daje nieznacznie gorsze rozwiązania, przy bardzo krótkim czasie działania. W odniesieniu do algorytmu przeszukiwania tabu i uzyskanych przez niego wyników zaobserwowano, iż brak rozbudowania algorytmu w mechanizm dywersyfikacji czy listy kandydatów, skutecznie ogranicza jego możliwy potencjał sprawiając, iż jest to algorytm o bliźniaczym działaniu, co podstawowy algorytm przeszukiwania lokalnego - tak samo zależny od wielkości generowanego rozwiązania.

Podczas prac implementacyjnych i procesu testowania zaimplementowanych algorytmów dostrzeżono, iż bardzo duże znaczenie na końcowy efekt działania mają wykorzystywane generatory. Lepiej odbywająca się generacja/modyfikacja rozwiązań obecnych, przekładać się będzie na większą efektywność algorytmów, pod względem lepszego wyniku końcowego jak i krótszego czasu trwania obliczeń. Zastosowane

w algorytmach generatory, opierały się na wszystkich opisanych wcześniej metodach „shift-end”, „swap” oraz „exchange”, dzięki czemu podczas walidacji algorytmów otrzymywano wyniki zbliżone, a nawet lepsze w stosunku do algorytmów referencyjnych wykorzystywanych w programie *HeuristicLab*.

W przypadku wariantu Cargo Theft VRP, trzeba pamiętać, że nie posiada on dedykowanych zbiorów danych testowych, o znanym optimum, więc stwierdzenie przypadku, w którym osiągnięto minimum globalne w trakcie przeprowadzonych eksperymentów – nie jest możliwe.

Główną konkluzją autora, na podstawie przeprowadzonych badań jest stwierdzenie, iż w przypadku rozwiązywania problemu Cargo Theft VRP, poleca się stosować algorytm przeszukiwania lokalnego lub przeszukiwania tabu z ustalonym krótkim czasem trwania tabu dla przypadków, w których należy zoptymalizować trasy, dla nie więcej niż ok. 50 lokalizacji. W przeciwnym przypadku, koszt rozwiązania końcowego będzie gorszy i nieopłacalny z powodów długiego czasu oczekiwania na wynik końcowy. Algorytm przeszukiwania dla takich zbiorów danych charakteryzuje się nieporównywalnie niskim czasem otrzymania wyniku, z tym że to algorytm przeszukiwania tabu ma przewagę w kontekście otrzymywania rozwiązań stabilnych, bardziej zbliżonych do siebie. Algorytm symulowanego wyżarzania ma w tym przypadku przewagę, ponieważ przenosi swoje rejony poszukiwań w dalsze rejony (też i niekorzystne), dzięki czemu – jak wykazały badania – szansa na otrzymanie wyniku lepszego jest wyższa od pozostałych i to w krótszym czasie ze względu na swoją losowość i znacznie mniejszą zależność od rozmiaru problemu.

Bibliografia

- [1] H. A. Eiselt, Michel Gendreau, Gilbert Laporte, „Arc Routing Problems, Part I: The Chinese Postman Problem,” Czerwiec 1994. [Online]. Available: <https://pubsonline.informs.org/doi/epdf/10.1287/opre.43.2.231>. [Data uzyskania dostępu: 14 Sierpień 2023].
- [2] Ramser, G. B. Dantzig and J. H., „The truck dispatching problem,” w *The truck dispatching problem*, Management Science, 1959, pp. 80-91.
- [3] R.J. Wilson, Wprowadzenie do teorii grafów, Warszawa: Wydawnictwo Naukowe PWN, 2012.
- [4] Repolho, H.M., Marchesi, J.F., Júnior, O.S.S., „Cargo theft weighted vehicle routing problem: modeling and application to the pharmaceutical distribution sector,” Springer, 24 Maj 2018. [Online]. Available: <https://doi.org/10.1007/s00500-018-3250-6>. [Data uzyskania dostępu: 20 08 2023].
- [5] Lech Banachowski, Wojciech Rytter, Krzysztof Marian Diks, Algorytmy i struktury danych, Warszawa: Wydawnictwo Naukowe PWN, 2018.
- [6] Maciej M. Sysło, Narsingh De, Janusz S. Kowalik., Algorytmy optymalizacji dyskretnej z programami w języku Pascal.
- [7] J. Walaszek, „Algorytmy Struktury Danych,” 2023. [Online]. Available: https://eduinf.waw.pl/inf/alg/001_search/0139.php. [Data uzyskania dostępu: 3 Wrzesień 2023].
- [8] A. Eiselt, Michel Gendreau, Gilbert Laporte, „Pubsonline,” Czerwiec 1994. [Online]. Available: <https://pubsonline.informs.org/doi/epdf/10.1287/opre.43.2.231>. [Data uzyskania dostępu: 14 Sierpień 2023].
- [9] Weisstein, Eric W., „Eulerian Cycle - MathWorld,” 2023. [Online]. Available: <https://mathworld.wolfram.com/EulerianCycle.html>. [Data uzyskania dostępu: 15 Sierpień 2023].
- [10] Kris Braekers, Katrien Ramaekers, Inneke Van Nieuwenhuyse, „The vehicle routing problem: State of the art classification and review,” w *Computers & Industrial Engineering*, ScienceDirect - Elsevier, 2016, pp. 300-313.

- [11] Fei Liu, Chengyu Lu, Lin Gui, Qingfu Zhang, Xialiang Tong, Mingxuan Yuan, „Heuristics for Vehicle Routing Problem: A Survey and Recent Advances,” 1 Marzec 2023. [Online]. Available: <https://arxiv.org/pdf/2303.04147.pdf>. [Data uzyskania dostępu: 10 Wrzesień 2023].
- [12] Sun S, Duan Z, Xu Q, „School bus routing problem in the stochastic and time-dependent transportation network,” *PLoS One*, p. 13, Sierpień 2018.
- [13] Agnieszka Debudaj-Grabysz, Sebastian Deorowicz, Jacek Widuch, Algorytmy i struktury danych. Wybór zaawansowanych metod., Gliwice: Wydawnictwo Politechniki Śląskiej, 2012.
- [14] R. Jachimowski, Zastosowanie algorytmów heurystycznych do rozwiązywania problemu układania tras pojazdów, Warszawa: Politechnika Warszawska, 2015.
- [15] J. Pacyna, Metaloznawstwo. Wybrane zagadnienia., Kraków: UWND AGH, 2005.
- [16] M. Dorigo, G. Di Caro & L. M. Gambardella, „Artificial Life,” w *Ant Algorithms for Discrete Optimization*, 1999, p. 137–172.
- [17] Reference of C++, „cplusplus .com - reference <random>,” 2023. [Online]. Available: <https://cplusplus.com/reference/random/>. [Data uzyskania dostępu: 20 Październik 2023].
- [18] Augerat, Christofides, Eilon, „CVRPLIB - Capacited Vehicle Routing Problem Library,” [Online]. Available: <http://vrp.gallos.inf.puc-rio.br/index.php/en/>. [Data uzyskania dostępu: 1 Październik 2023].
- [19] S. Wagner, Heuristic Optimization Software Systems - Modeling of Heuristic Optimization Algorithms in the HeuristicLab Software Environment, Linz: Johannes Kepler University Linz, 2009.

Spis skrótów i symboli

<i>CPU</i>	Procesor umożliwiający wykonywanie obliczeń numerycznych (z ang. <i>Central Processing Unit</i>)
<i>CUDA</i>	Architektura procesorów wielordzeniowych opracowana przez firmę Nvidia (z ang. <i>Compute Unified Device Architecture</i>)
<i>CSV</i>	Format pliku tekstowego rozdzielanego przecinkami (z ang. <i>Comma-Separated Values</i>)
<i>EVRP</i>	Wariant problemu trasowania pojazdów elektrycznych uwzględniających ich możliwą najdłuższą trasę bez konieczności ładowania pojazdu (z ang. <i>Electric Vehicle Routing problem</i>)
<i>GPU</i>	Graficzny procesor wysokiej wydajności (z ang. <i>Graphical Processing Unit</i>)
<i>GVRP</i>	Wariant problemu trasowania pojazdów dotyczący aspektu emisji zanieczyszczeń posiadanej floty pojazdów
<i>VRP</i>	Klasyczny problem trasowania pojazdów (z ang. <i>Vehicle Routing Problem</i>)
<i>CVRP</i>	Wariant problemu VRP dotyczący ograniczeń w ładowności pojazdów (z ang. <i>Capacited Vehicle Routing Problem</i>)
<i>CTWVRP</i>	Wariant problemu VRP dotyczący z ograniczeniami ładowności pojazdów i tzw. oknami czasowymi dostaw (z ang. <i>Capacited with Time Window Vehicle Routing Problem</i>)
<i>CTVRP</i>	Wariant problemu VRP dotyczący zagadnienia kradzieży przewożonych towarów (z ang. <i>Cargo Theft Vehicle Routing Problem</i>)
<i>LS</i>	Skrót dot. algorytmu przeszukiwania lokalnego (z ang. <i>Local Search</i>)
<i>MDVRP</i>	Wariant problemu VRP z wieloma magazynami (z ang. <i>Multi Depot Vehicle Routing Problem</i>)
<i>SA</i>	Skrót dot. algorytmu symulowanego wyżarzania (z ang. <i>Simulated Annealing</i>)
<i>SBRP</i>	Wariant problemu marszrutyzacji autobusów szkolnych (z ang. <i>School Bus Routing Problem</i>)
<i>TS</i>	Skrót dot. algorytmu przeszukiwania tabu (z ang. <i>Tabu Search</i>)

Lista dodatkowych plików, uzupełniających tekst pracy

W systemie do pracy dołączono dodatkowe pliki zawierające:

- źródła programu (nie zawiera źródeł biblioteki Boost 1.8.3),
- dokumentacje programu wygenerowaną automatycznie,
- zbiory danych użyte w eksperymentach.

Spis rysunków

Rys. 2.1 Przykład grafu pełnego, skierowanego, z wagami.....	14
Rys. 2.2 Przykład graficzny dla problemu Komiwojażera.....	16
Rys. 2.3 Przykład reprezentacji graficznej (rozwiązanie) problemu VRP.....	18
Rys. 2.4 Struktura rozwiązania problemu VRP	21
Rys. 2.5 Ogólna zasada działania algorytmów heurystycznych	22
Rys. 2.6 Metoda „Swap”	23
Rys. 2.7 Metoda „Exchange”	23
Rys. 2.8 Metoda „Shift-End”	24
Rys. 2.9 Zachłanny algorytm przeszukiwania lokalnego.....	25
Rys. 2.10 Strony algorytm przeszukiwania lokalnego.....	26
Rys. 2.11 Pseudokod algorytmu symulowanego wyżarzania	28
Rys. 2.12 Pseudokod algorytmu przeszukiwania tabu	29
Rys. 2.13 Wykres przedstawiający ilość publikacji dot. problematyki VRP [10]	32
Rys. 3.1 Widok programu "Solver.exe" - sekcja pomocy.....	34
Rys. 3.2 Struktura plików i katalogów głównego programu.....	35
Rys. 3.3 Błąd programu - brak odpowiednich parametrów wejściowych	36
Rys. 3.4 Fragment pliku wyjściowego – tekstowego.....	38
Rys. 3.5 Ogólna sekwencja działania programu <i>solver.exe</i>	39
Rys. 3.6 Hierarchia klas zaimplementowanych algorytmów	40
Rys. 3.7 Pseudokod generacji rozwiązania początkowego	41
Rys. 3.8 Pseudokod generacji nowego rozwiązania	42
Rys. 4.1 Wykres całkowitych kosztów rozwiązania dla zbioru A-n33-k5. Rozwiązanie proponowane (<i>solver.exe</i>) – algorytm przeszukiwania lokalnego	50
Rys. 4.2 Wykres całkowitych kosztów rozwiązania dla zbioru A-n33-k5. Rozwiązanie referencyjne (<i>HeuristicLab</i>) – algorytm przeszukiwania lokalnego	51
Rys. 4.3 Wykres całkowitych kosztów rozwiązania dla zbioru A-n33-k5. Rozwiązanie proponowane (<i>solver.exe</i>) – algorytm symulowanego wyżarzania	52
Rys. 4.4 Wykres całkowitych kosztów rozwiązania dla zbioru A-n33-k5. Rozwiązanie referencyjne (<i>HeuristicLab</i>) – algorytm symulowanego wyżarzania	53
Rys. 4.5 Wykres całkowitych kosztów rozwiązania dla zbioru A-n33-k5. Rozwiązanie proponowane (<i>solver.exe</i>) – algorytm przeszukiwania tabu	54
Rys. 4.6 Wykres całkowitych kosztów rozwiązania dla zbioru A-n33-k5. Rozwiązanie referencyjne (<i>HeuristicLab</i>) – algorytm przeszukiwania tabu	55

Rys. 4.7 Wizualizacja wyników - badanie współczynnika alfa	59
Rys. 4.8 Zależność czasu trwania algorytmu lokalnego przeszukiwania od rodzaju danych wejściowych	66
Rys. 4.9 Zależność otrzymanych rozwiązań od rodzaju danych wejściowych dla algorytmu lokalnego przeszukiwania.....	67
Rys. 4.10 Zależność czasu trwania algorytmu symulowanego wyżarzania od rodzaju danych wejściowych.....	68
Rys. 4.11 Zależność otrzymanych rozwiązań od rodzaju danych wejściowych dla algorytmu symulowanego wyżarzania	69
Rys. 4.12 Zależność czasu trwania algorytmu przeszukiwania tabu od rodzaju danych wejściowych	70
Rys. 4.13 Zależność otrzymanych rozwiązań od rodzaju danych wejściowych dla algorytmu przeszukiwania tabu	71

Spis tabel

Tabela 3.1 Struktura obsługiwanych zbiorów danych (przykład pliku wejściowego)	37
Tabela 3.2 Struktura danych dot. pojazdów (przykład pliku wejściowego)	37
Tabela 3.3 Struktura pliku CSV zawierającego wyniki działania algorytmów (przykład). 39	
Tabela 4.1 Przygotowane środowisko badawcze – część sprzętowa	47
Tabela 4.2 Przygotowane środowisko badawcze - część programowa.....	47
Tabela 4.3 Wykorzystane zbiory danych	48
Tabela 4.4 Parametry algorytmu symulowanego wyżarzania wykorzystane do procesu walidacji	54
Tabela 4.5 Parametry algorytmu symulowanego wyżarzania.....	57
Tabela 4.6 Parametry algorytmu przeszukiwania tabu	58
Tabela 4.7 Eksperymentalne parametry algorytmu symulowanego wyżarzania	58
Tabela 4.8 Wyniki eksperymentu dot. współczynnika schładzania alfa.....	58
Tabela 4.9 Wyniki badań - Seria I.....	61
Tabela 4.10 Wyniki badań - Seria II	62
Tabela 4.11 Wyniki badań - Seria III.....	63
Tabela 4.12 Średnia uzyskana korzyść rozwiązania problemów CTVRP	72