Uniwersytet Śląski
Wydział Informatyki i Nauki o Materiałach
Instytut Informatyki

Rozprawa doktorska

Wojciech Bura

Wielokryterialne, mrowiskowe algorytmy optymalizacji w nawigacji samochodowej

Promotor: dr hab. inż. prof. UŚ Mariusz Boryczka

Sosnowiec, 2014 r.

Dziękuję mojej żonie i dzieciom za cierpliwość i wsparcie w trudnych momentach...

Spis treści

Rozdział 1

Wstęp

Rozwiązywanie złożonych problemów optymalizacji dyskretnej znajduje zastosowania praktyczne w wielu dziedzinach aktywności człowieka. Przykładem może być wyszukiwanie optymalnej drogi między dwoma punktami na mapie drogowej w nawigacji samochodowej z zastosowaniem wielu kryteriów oceny. Problem jest znany w literaturze jako bardziej ogólny, wielokryterialny problem najkrótszej drogi w grafie (ang. *multi-objective shortest path problem* – MOSP). Do rozwiązywania tego problemu stosowane są różne metody i podejścia. Między innymi stosowana jest metoda, która polega na wprowadzeniu tzw. kryterium zastępczego, będącego agregacją poszczególnych kryteriów oceny. Zastąpienie wielu kryteriów jednym kryterium zastępczym sprowadza złożony problem optymalizacji wielokryterialnej do optymalizacji jednokryterialnej. Zwykle umożliwia to skorzystanie z szybkich algorytmów deterministycznych, które wyznaczają jedno kompromisowe rozwiązanie. Metoda z kryterium zastępczym wymaga ustalenia wartości wag dla poszczególnych kryteriów przed rozpoczęciem procesu optymalizacji. W rezultacie wyniki otrzymywane za pomocą tej metody mogą się znacząco różnić w zależności od przyjętych wartości wag i zwykle konieczne jest powtórzenie obliczeń dla różnych wartości tych współczynników.

Innym bardzo popularnym podejściem do rozwiązywania problemów optymalizacji wielokryterialnej jest metoda Pareto. Przy tym podejściu wynikiem działania algorytmów optymalizacji wielokryterialnej jest zbiór rozwiązań niezdominowanych (paretooptymalnych), a nie pojedyncze (optymalne) rozwiązanie. Jednym z najbardziej znanych algorytmów do wyznaczania zbioru rozwiązań niezdominowanych dla problemu MOSP jest algorytm etykietowania (ang. *the labelling algorithm*). W literaturze występują dwie główne wersje tego algorytmu: Label Setting oraz Label Correcting. Ponieważ algorytmy deterministyczne wyznaczające pełne zbiory paretooptymalne charakteryzują się zwykle dużą złożonością obliczeniową (czasową oraz pamięciową), w literaturze znane są próby stosowania szybkich algorytmów wyznaczających przybliżone zbiory rozwiązań. Przykładem takich algorytmów mogą być algorytmy oparte na metaheurystyce optymalizacji mrowiskowej (ang. *ant colony optimization metaheuristic*).

Optymalizacja mrowiskowa (ang. *ant colony optimization* – ACO) jest paradygmatem związanym z tworzeniem algorytmów heurystycznych dla rozwiązywania problemów optymalizacji dyskretnej, które należą do licznego grona algorytmów inspirowanych przez naturę [99]. Jest on oparty na kolonii sztucznych mrówek, które współpracują i komunikują się za pośrednictwem sztucznych śladów feromonowych. Pierwszym algorytmem w tej klasie był system mrówkowy, który wywodzi się z badań w dziedzinie systemów

naśladujących rzeczywiste zachowania mrówek i został zaproponowany w 1991 r. przez M. Dorigo, V. Maniezzo i A. Colorniego jako algorytm rozwiązujący problem komiwojażera [44]. Mrówki podróżują w przestrzeni rozwiązań, która zwykle ma strukturę grafową. Następny punkt swojej drogi mrówki wybierają z prawdopodobieństwem zależącym od dwóch rodzajów informacji związanych z krawędzią:

- statycznej informacji heurystycznej, np. odległości między węzłami,
- śladu feromonowego, który zmienia się w trakcie obliczeń i jest środkiem „porozumiewania się" mrówek.

Ponieważ optymalizacja ACO jest wzorowana na zachowaniu kolonii rzeczywistych mrówek, dlatego też sztuczne mrówki mają wiele cech zaczerpniętych z ich obserwacji, np.:

- wybór odcinka drogi w zależności od wielkości śladu feromonowego,
- umiejętność znajdowania najkrótszej drogi między dwoma punktami,
- zdolność do kooperacji w celu osiągnięcia jak najlepszego wyniku,
- przemieszczanie się w sposób losowy.

W literaturze znanych jest niewiele wersji algorytmów optymalizacji ACO do rozwiązywania problemu MOSP w zastosowaniu do nawigacji samochodowej, m.in. zaproponowany został algorytm AVN (ang. *ant vehicle navigation*) [100–102]. Podstawowy algorytm znany z literatury znajduje optymalną drogę między dwoma punktami na mapie przy określeniu przez użytkownika preferencji odnośnie odległości, natężenia ruchu, liczby pasów ruchu, ryzyka kolizji, jakości oraz liczby skrzyżowań. Parametryzacja następuje przez ustalenie wartości współczynników występujących w kryteriach optymalizacji. Podczas obliczeń brany jest także pod uwagę czas rozpoczęcia podróży, ponieważ wagi, np. natężenie ruchu lub ryzyko kolizji, na poszczególnych odcinkach drogi mogą mieć różne wartości o różnej porze dnia i nocy. Niestety algorytm ten nie zawsze daje jakiekolwiek rozwiązanie postawionego problemu, w szczególności dla rozbudowanych danych rzeczywistych. W związku z tym w pracy [13] zaproponowano kilka jego modyfikacji oraz nową, znacznie zmienioną wersję algorytmu AVN nazwaną NAVN (ang. *new ant vehicle navigation*), która zapewnia uzyskanie prawidłowych wyników dla rzeczywistych danych.

Algorytm AVN do rozwiązywania wielokryterialnego problemu optymalizacyjnego wykorzystuje podejście z kryterium zastępczym. Jak wspomniano wcześniej, jest to jeden z możliwych sposobów rozwiązywania tego typu problemów lecz nie jest to sposób pozbawiony wad, choć w przypadku wyszukiwania najkrótszej drogi w grafie umożliwia zastosowanie szybkich algorytmów deterministycznych (np. algorytmu Dijkstry). Rezygnacja z kryterium zastępczego i zastosowanie podejścia polegającego na wyznaczaniu zbioru rozwiązań niezdominowanych znacznie zwiększa złożoność obliczeniową problemu i uzasadnia stosowanie metod przybliżonych. W pracach [21,22] zaproponowano algorytm MultiNAVN, który aproksymuje zbiór rozwiązań niezdominowanych.

W ostatnich czasach wzrasta zainteresowanie algorytmami, które umożliwiają stosowanie coraz bardziej popularnych i dostępnych systemów równoległych. Bariery technologiczne powodują, że trudne jest utrzymanie ciągłego wzrostu częstotliwości taktowania zegara pojedynczych procesorów. Wzrost mocy obliczeniowej (szybkości działania) systemów komputerowych osiągany jest przez równoległą pracę wielu tradycyjnych procesorów w modelu SMP (ang. *symmetric multiprocessing*) lub w modelu sieciowym oraz przez rozwój nowych architektur obliczeniowych, takich jak CUDA (ang. *compute unified device architecture*).

Znanych jest wiele prób zrównoleglenia algorytmów mrowiskowych, które dają różne, najczęściej dość słabe rezultaty. W trakcie prac nad algorytmem AVN opracowane zostały dwie wersje algorytmów równoległych: PAVN (ang. *parallel ant vehicle navigation*) dla modelu SMP oraz OCLAVN (ang. *opencl ant vehicle navigation*) dla architektury CUDA/OpenCL. Zaprezentowane w pracy [20] wyniki eksperymentów pokazują, że takie zrównoleglenie algorytmów mrowiskowych może poprawić ich szybkość działania przy zachowaniu porównywalnej jakości znajdowanych rozwiązań.

Teza rozprawy

Problem znajdowania optymalnej drogi w nawigacji samochodowej z uwzględnieniem wielu kryteriów może być rozwiązywany za pomocą algorytmów mrowiskowych na rzeczywistych danych nawigacyjnych skuteczniej niż za pomocą algorytmu Label Setting.

Cel rozprawy

Celem rozprawy jest udowodnienie prawdziwości postawionej tezy. Aby to osiągnąć, w rozprawie realizowane były następujące cele główne:

- Opracowanie i zaimplementowanie oryginalnego algorytmu mrowiskowego rozwiązującego problem znajdowania optymalnej drogi między dwoma punktami na mapie z wykorzystaniem wielu kryteriów i działającego w oparciu o zasady optymalizacji wielokryterialnej w sensie Pareto.
- Dokonanie weryfikacji eksperymentalnej działania algorytmu na rzeczywistych danych drogowych pochodzących z systemu OpenStreetMap, uzupełnionych o informacje o wypadkach i kolizjach uzyskanych z policyjnego systemu SEWiK (System Ewidencji Wypadków i Kolizji).

Celami dodatkowymi rozprawy były:

- Eksperymentalny dobór parametrów zaproponowanych dwóch wersji algorytmu mrowiskowego, tj. wersji z kryterium zastępczym oraz wersji z kryterium wybieranym losowo.
- Przeprowadzenie eksperymentów porównawczych wersji zaproponowanego algorytmu mrowiskowego.
- Wykonanie adaptacji zaproponowanego algorytmu do architektury CUDA oraz interfejsu programistycznego OpenCL.

Rozprawa składa się siedmiu rozdziałów opisujących aspekty teoretyczne poruszanych zagadnień wraz z przeglądem stanu badań dotyczących rozważanych zagadnień. Przedstawiono omówienie proponowanych algorytmów mrowiskowych i wyniki eksperymentów obliczeniowych potwierdzających zasadność ich opracowania. W rozdziale 2 opisane zostały najważniejsze zagadnienia związane z problematyką optymalizacji wielokryterialnej. Zdefiniowano podstawowe pojęcia oraz wyjaśniono najbardziej znane podejścia i algorytmy do rozwiązywania tego typu problemów. Szczególną uwagę poświęcono optymalizacji

wielokryterialnej w sensie Pareto. Rozdział 3 zawiera formalne sformułowanie problemu MOSP oraz przegląd literatury dotyczący algorytmów do rozwiązywania wielokryterialnego problemu wyszukiwania optymalnej drogi w grafie. Szerzej wyjaśniono zasadę działania deterministycznego algorytmu Label Setting, który wyznacza pełny zbiór rozwiązań niezdominowanych (paretooptymalnych) dla problemu MOSP, charakteryzujący się dużą złożonością obliczeniową. Rozdział 4 poświęcono algorytmom optymalizacji mrowiskowej (ACO). Zaprezentowano różne wersje tych algorytmów z uwzględnieniem wersji do rozwiązywania problemów optymalizacji wielokryterialnej (MOACO) oraz przedstawiono przykłady zastosowań algorytmów opartych na metaheurystyce mrowiskowej. Rozdziały 5, 6 i 7 zawierają opis oryginalnego dorobku autora rozprawy. W rozdziale 5 opisane zostały wielokryterialne, mrowiskowe algorytmy nawigacji samochodowej (AVN) znane z literatury, oraz dwa algorytmy opracowane przez autora rozprawy: udoskonalony algorytm AVN (NAVN) oraz algorytm mrowiskowej optymalizacji wielokryterialnej w sensie Pareto (MultiNAVN). Rozdział 6 przedstawia wyniki badań eksperymentalnych przeprowadzonych przez autora rozprawy na rzeczywistych danych kartograficznych pobranych z systemu OpenStreetMap, które zostały uzupełnione o informacje o wypadkach i kolizjach z systemu SEWiK. W rozdziale 7 opisana została adaptacja algorytmu NAVN dla architektury CUDA/OpenCL OCLAVN. Opisano w nim sposób przekształcenia algorytmu sekwencyjnego w wersję równoległą działającą w modelu SIMD (ang. *single instruction, multiple data*). Zaprezentowane zostały wyniki eksperymentów na danych rzeczywistych OpenStreetMap, ale bez informacji z systemu SEWiK.

Podziękowania

Autor dziękuje pracownikom Instytutu Informatyki Uniwersytetu Śląskiego za okazaną pomoc, wsparcie i cenne uwagi. Komendantowi Miejskiemu Policji w Katowicach za udostępnienie danych o wypadkach i kolizjach z systemu SEWiK, które umożliwiły przeprowadzenie eksperymentów na danych rzeczywistych. Szczególne podziękowania autor składa promotorowi Panu dr hab. inż. prof. UŚ Mariuszowi Boryczce, za życzliwą pomoc, cenne rady i liczne wskazówki udzielone w trakcie realizacji niniejszej rozprawy, bez których by ona nie powstała.

Rozdział 2

Optymalizacja wielokryterialna

Zadanie optymalizacji można rozumieć potocznie, jako dążenie do osiągnięcia stanu idealnego, spełniającego określone kryteria oceny. Rozwiązywanie problemów optymalizacyjnych z pojedynczym kryterium oceny to optymalizacja jednokryterialna. W rzeczywistości znacznie częściej występują problemy optymalizacyjne z większą liczbą kryteriów oceny rozwiązania. Dla większej liczby kryteriów często może dochodzić do sprzeczności między kryteriami, co oznacza, że poszukiwane rozwiązanie stanowi swego rodzaju kompromis pomiędzy nimi.

Optymalizacja wielokryterialna jest bez wątpienia bardzo ważnym tematem badań zarówno dla naukowców jak i inżynierów, nie tylko ze względu na wielokryterialny charakter większości rzeczywistych problemów, ale także dlatego, że jest jeszcze wiele otwartych zagadnień w tym obszarze. W przeciwieństwie do optymalizacji jednokryterialnej, nie ma nawet powszechnie przyjętej definicji „optimum", co sprawia, że trudno nawet porównać wyniki jednej metody z drugą, bo zazwyczaj decyzja o tym, co jest „najlepszym" rozwiązaniem i tak należy do czynnika ludzkiego – decydenta.

Formalnie optymalizację wielokryterialną można sformułować następująco [46]:
Niech $X = \{x_l, l = 1, 2, ..., N\}$ będzie wektorem niezależnych zmiennych decyzyjnych. Niech $F = \{f_i, i = 1, 2, ..., M\}$ będzie zbiorem kryteriów (funkcji), względem których oceniane są rozwiązania w poszukiwaniu kompromisu. Niech dane będą ograniczenia nałożone na wartości rozwiązań:

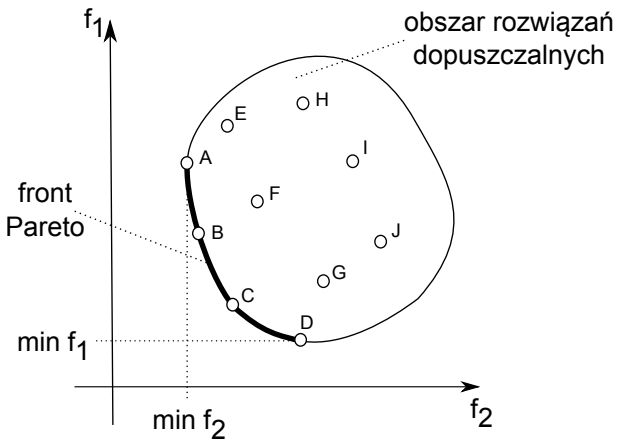
- nierównościowe $G = \{g_k, k = 1, 2, ..., K\}$, gdzie: $g_k(X) \leqslant 0$,
- równościowe $H = \{h_j, j = 1, 2, ..., J\}$, gdzie: $h_j(X) = 0$.

Celem optymalizacji wielokryterialnej jest znalezienie rozwiązania, które zachowuje ograniczenia oraz spełnia warunek, który w przypadku minimalizacji przyjmuje postać:

$$\min F(X) = \{f_1(X), f_2(X), ..., f_l(X)\}. \tag{2.1}$$

W przypadku gdy wymagana jest maksymalizacja pewnej funkcji f'_l można ją zastąpić funkcją f_l wprowadzając funkcję pomocniczą według zależności:

$$\min f_l(X) = -\max f'_l(-X). \tag{2.2}$$



Rysunek 2.1: Optymalizacja w sensie Pareto

2.1 Optymalizacja w sensie Pareto

Francusko-włoski ekonomista V. Pareto w roku 1896 [96] sformułował zasadę optymalizacji wielokryterialnej w zagadnieniach ekonomicznych, którą później nazwano optymalizacją w sensie Pareto. Według zasad tej metody możliwe rozwiązania zadania optymalizacji klasyfikuje się jako rozwiązania zdominowane i niezdominowane (paretooptymalne). Rozwiązanie niezdominowane można zdefiniować następująco [82]:

Definicja 2.1 (Rozwiązanie niezdominowane) *Rozwiązanie $x' \in X$ jest niezdominowane (paretooptymalne), jeżeli nie istnieje inne rozwiązanie $x \in X$, takie że $F(x) \leqslant F(x')$, oraz dla co najmniej jednego kryterium spełniona jest zależność $f_l(x) < f_l(x')$.*

Paretooptymalność można zdefiniować formalnie [23]:

Definicja 2.2 (Paretooptymalność) *Rozwiązanie $x \in X$ jest optymalne w sensie Pareto, jeżeli nie istnieje inne rozwiązanie $x' \in X$, dla którego wektor*

$$v = F(x') = (f_1(x'), \ldots, f_M(x'))$$

dominuje wektor

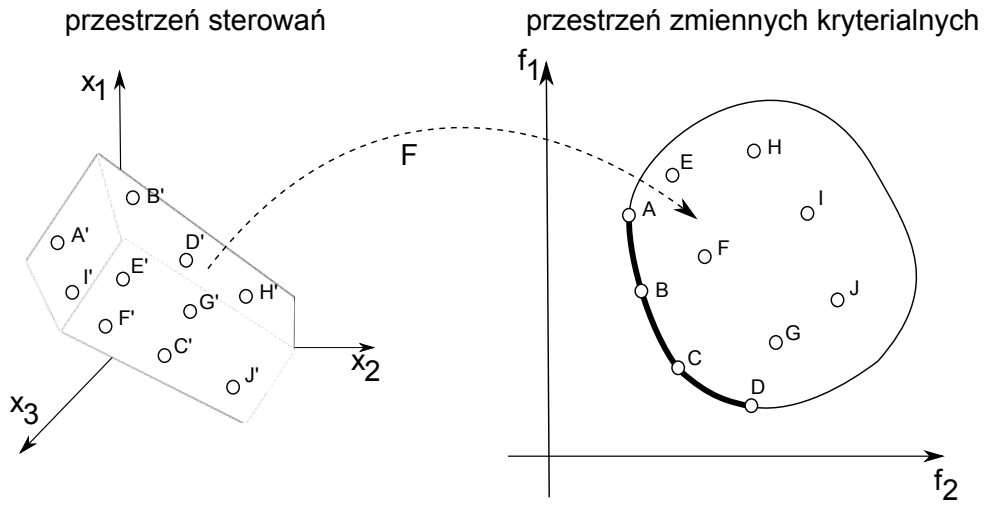
$$u = F(x) = (f_1(x), \ldots, f_M(x)).$$

W potocznym rozumieniu obie definicje mówią, że rozwiązanie x jest optymalne w sensie Pareto jeśli nie istnieje rozwiązanie x', dla którego jakieś kryterium ma mniejszą wartość, bez powodowania jednoczesnego wzrostu wartości dla co najmniej jednego innego kryterium (zakładając minimalizację).

Istnieje kilka innych definicji, które uzupełniają formalny opis podejścia Pareto do rozwiązywania problemów optymalizacji wielokryterialnej [23]:

Definicja 2.3 (Dominacja w sensie Pareto) *Wektor $u = (u_1, \ldots, u_M)$ dominuje wektor $v = (v_1, \ldots, v_M)$ (co oznacza się $u \preceq v$) wtedy i tylko wtedy gdy:*

$$\forall i \in \{1, \ldots, M\}, u_i \leqslant v_i \wedge \exists i \in \{1, \ldots, M\} : u_i < v_i.$$



Rysunek 2.2: Relacja między przestrzenią sterowań a przestrzenią zmiennych kryterialnych

Jedno rozwiązanie optymalne w sensie Pareto występuje tylko wtedy, gdy wszystkie optima cząstkowe znajdują się w tym samym punkcie; jest to wtedy również rozwiązanie optymalne całego problemu. Na ogół rozwiązań paretooptymalnych jest wiele, w skrajnym przypadku każde rozwiązanie może być takim rozwiązaniem. Rysunek 2.1 przedstawia przykład optymalizacji w sensie Pareto dla problemu minimalizacji dwóch kryteriów (f_1 i f_2). Punkty od A do J reprezentują rozwiązania należące do obszaru rozwiązań dopuszczalnych. Punkty A, B, C i D to rozwiązania niezdominowane (paretooptymalne) [23], które tworzą tzw. front Pareto.

Definicja 2.4 (Zbiór rozwiązań paretooptymalnych) *Dla danego problemu optymalizacji wielokryterialnej opisanego przez funkcję $F(x)$, zbiór rozwiązań paretooptymalnych $\mathcal{P}^*$ jest zdefiniowany jako:*

$$\mathcal{P}^* = \{x \in X | \nexists x' \in X F(x') \preceq F(x)\}. \tag{2.3}$$

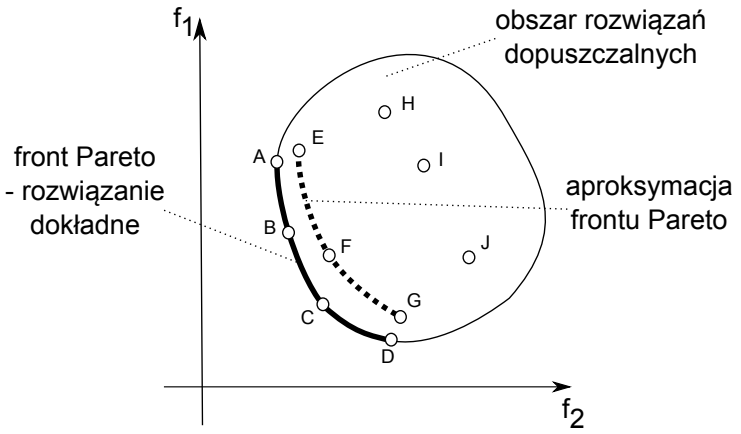
Zbiorowi rozwiązań paretooptymalnych w przestrzeni zmiennych decyzyjnych (przestrzeni sterowań) odpowiada zbiór wartości zmiennych kryterialnych nazywany frontem Pareto (rys. 2.2), który można zdefiniować formalnie jako [23]:

Definicja 2.5 (Front Pareto) *Dla danego problemu optymalizacji wielokryterialnej opisanego przez funkcję $F(x)$ i zbiór rozwiązań paretooptymalnych $\mathcal{P}^*$, front Pareto $\mathcal{PF}^*$ jest zdefiniowany jako:*

$$\mathcal{PF}^* = \{u = F(x) | x \in \mathcal{P}^*\}. \tag{2.4}$$

Po wyznaczeniu zbioru rozwiązań niezdominowanych należy określić metodę wyboru końcowego rozwiązania. Przykładami takich metod są:

- metoda dialogowa,
- metakryterium,
- hierarchia celów.



Rysunek 2.3: Aproksymacja frontu Pareto

Techniki używane do wyznaczania zbioru rozwiązań niezdominowanych można podzielić na dokładne i heurystyczne. Metody dokładne umożliwiają znalezienie zbioru rozwiązań optymalnych w sensie Pareto dla danego problemu optymalizacyjnego, lecz nie nadają się do większości praktycznych zastosowań ze względu na bardzo wysoki koszt obliczeniowy algorytmów. Z kolei metody heurystyczne nie gwarantują znalezienia kompletnego zbioru rozwiązań niezdominowanych, ale dzięki nim można znacznie ograniczyć czas poszukiwania takich rozwiązań. Przykładami metod heurystycznych mogą być:

- algorytmy genetyczne,
- symulowane wyżarzanie,
- tabu-search,
- algorytmy mrowiskowe.

2.2 Aproksymacja frontu Pareto

Zadanie algorytmu aproksymującego front Pareto polega na wyznaczeniu zbioru rozwiązań niezdominowanych, który przybliża zbiór optymalny w sensie Pareto w sposób możliwie najdokładniejszy. Metody dokładne, wyznaczające pełny zbiór rozwiązań pareto-optymalnych, mają zwykle wysoką złożoność obliczeniową (czasową), rzędu $O(2^n)$ [57,70]. Wysoka złożoność obliczeniowa tradycyjnych algorytmów zachęca do stosowania metod heurystycznych, np. algorytmów genetycznych (VEGA, SPEA). Przykładem takiej metody mogą być również algorytmy mrowiskowe.

Jeżeli istnieje możliwość wyznaczenia kompletnego zbioru Pareto metodą dokładną, to można oceniać jakość wyniku aproksymacji jako odległość między dwoma zbiorami (np. metryka Hausdorffa) (rys. 2.3). Wielkość ta powinna być minimalizowana. Pożądany jest jednolity rozkład rozwiązań, a więc zbiór rozproszony, a nie skupiony w obrębie niewielkiego zakresu wartości kryteriów, oraz możliwie szeroki front Pareto, tak aby dostarczany był możliwie liczny zbiór rozwiązań dla każdego kryterium.

Metryka Hausdorffa, zwana inaczej odstępem Hausdorffa, jest to odległość pomiędzy zwartymi podzbiorami przestrzeni metrycznej zupełnej X i może być zdefiniowana formalnie [45, 72, 86]:

Definicja 2.6 (Metryka Hausdorffa) *Niech (X, d) będzie dowolną przestrzenią metryczną zupełną, a $H(X)$ przestrzenią, której elementami są zwarte i niepuste podzbiory przestrzeni X. Niech A i B będą elementami przestrzeni $H(X)$, a x, y elementami przestrzeni X, przy czym $x \in A, y \in B$. Wyrażenia:*

$$\delta(x, B) = \min\{d(x, y) \colon y \in B\} \tag{2.5}$$

$$\delta(y, A) = \min\{d(x, y) \colon x \in A\} \tag{2.6}$$

oznaczają odpowiednio odstęp punktu x od zbioru B i odstęp punktu y od zbioru A. Z kolei wyrażenia:

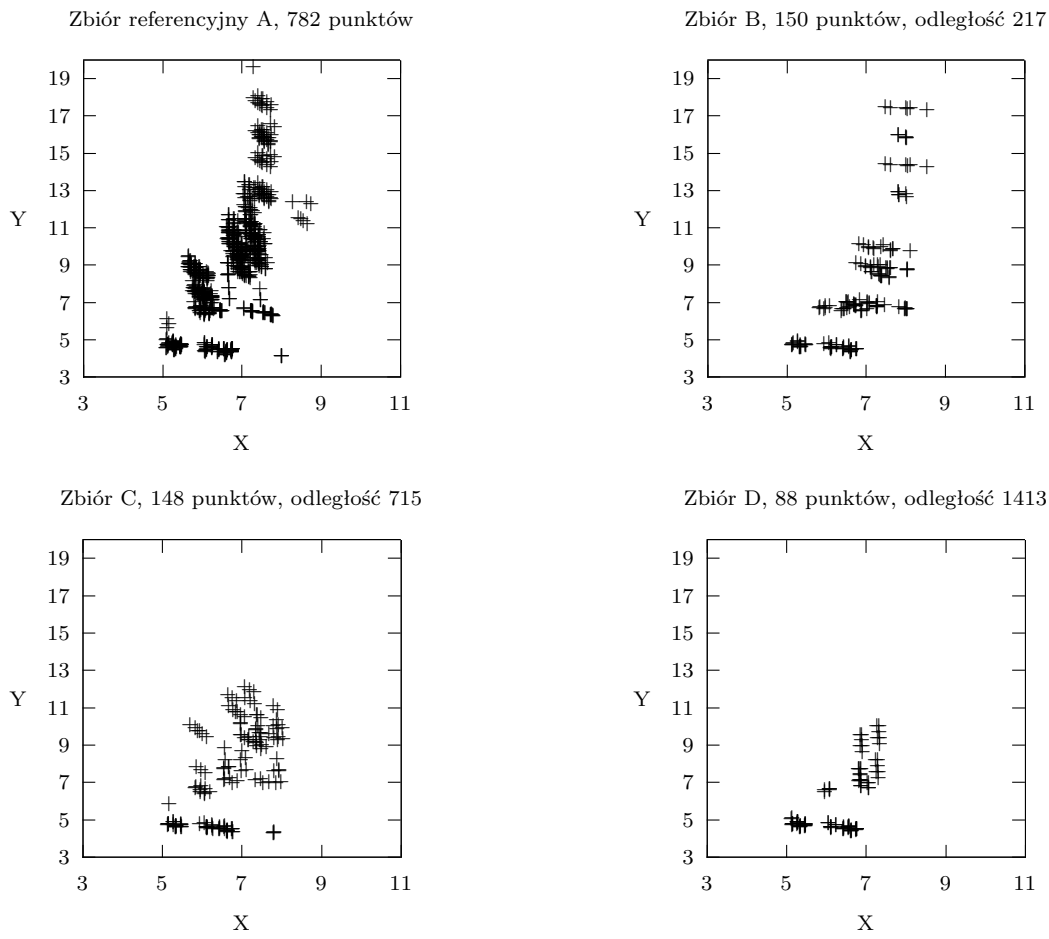
$$\delta(A, B) = \max\{\delta(x, B) \colon x \in A\} \tag{2.7}$$

$$\delta(B, A) = \max\{\delta(y, A) \colon y \in B\} \tag{2.8}$$

oznaczają odpowiednio odstęp zbioru A od zbioru B i odstęp zbioru B od zbioru A. Metryką Hausdorffa nazywa się funkcję $h \colon H(X) \times H(X) \to [0; \infty)$ określoną wzorem:

$$h(A, B) = \max\{\delta(A, B), \delta(B, A)\} \tag{2.9}$$

W celu zaprezentowania możliwości wykorzystania metryki Hausdorffa do oceny jakości aproksymacji zbiorów rozwiązań niezdominowanych, na rys. 2.4 zaprezentowane zostały cztery różne zbiory punktów: A, B, C i D. Zbiór A pełni rolę zbioru referencyjnego składającego się z 782 punktów. Pozostałe zbiory: B, C i D są różnymi zbiorami punktów, dla których obliczona została wartość metryki Hausdorffa, jako odległość od zbioru referencyjnego A. Zbiór B składa się ze 150 punktów, a wartość metryki Hausdorffa wynosi 217. Dla zbioru C, który składa się ze 148 punktów, wartość metryki wynosi 715. Dla ostatniego zbioru D, o liczności 88 punktów, metryka Hausdorffa wynosi 1413. Można zaobserwować, że im niższa jest wartość obliczonej metryki, tym większe jest podobieństwo między porównywanymi zbiorami. Ta własność metryki Hausdorffa została wykorzystana do oceny jakości aproksymacji zbiorów rozwiązań niezdominowanych podczas przeprowadzonych eksperymentów obliczeniowych w kolejnych rozdziałach rozprawy.



Rysunek 2.4: Referencyjny zbiór punktów i trzy zbiory o różnej odległości od zbioru referencyjnego

2.3 Inne podejścia do rozwiązywania optymalizacji wielokryterialnej

W literaturze znanych jest wiele metod rozwiązywania zadań optymalizacji wielokryterialnej, które łączy wspólne podejście do problemu polegające na zamianie wielu niezależnych kryteriów na jedną złożoną funkcję agregującą. Funkcja ta odgrywa rolę kryterium zastępczego i pozwala sprowadzić problem optymalizacji wielokryterialnej do problemu optymalizacji z jednym kryterium.

Innymi sposobami radzenia sobie ze złożonością problemu optymalizacji wielokryterialnej są próby ograniczania przestrzeni przeszukiwania lub rozpatrywanie poszczególnych kryteriów osobno.

W kolejnych podrozdziałach zaprezentowane są najbardziej popularne metody nie korzystające z podejścia zaprezentowanego przez Vilfredo Pareto.

2.3.1 Metoda kryteriów ważonych

Często stosowaną metodą rozwiązywania wielokryterialnych problemów optymalizacyjnych jest metoda ważonych kryteriów (ang. *weighted objectives method*). Polega ona na sprowadzeniu optymalizacji wielokryterialnej do jednokryterialnej przez wprowadzenie kryterium zastępczego, będącego sumą ważoną poszczególnych kryteriów:

$$Z = \sum_{i=0}^{M} w_i \cdot f_i(X), \tag{2.10}$$

gdzie: w_i – współczynnik ważności i-tego kryterium, spełniający następujące warunki:

$$0 \leqslant w_i \leqslant 1, \tag{2.11}$$

$$\sum_{i=0}^{M} w_i = 1. \tag{2.12}$$

Uzyskaną w ten sposób funkcję celu optymalizuje się przy użyciu standardowych metod optymalizacji z jedną funkcją celu.

Ponieważ wyniki uzyskiwane za pomocą tej metody mogą się bardzo różnić w zależności od zmian współczynników sterujących ważnością poszczególnych kryteriów; zwykle konieczne jest powtórzenie obliczeń dla wielu różnych wartości w_i lub ustalenie priorytetów przed obliczeniami. Wybór końcowego rozwiązania zależy od intuicji operatora lub właściwego doboru współczynników. Należy zwrócić uwagę, że współczynniki nie oddają w sposób proporcjonalny względnej ważności poszczególnych kryteriów w sytuacji, gdy poszczególne funkcje oceny wyrażone są w innych jednostkach lub dotyczą różnych zakresów wartości.

2.3.2 Metoda optymalizacji hierarchicznej

Metoda optymalizacji hierarchicznej (ang. *hierarchical optimization method*) polega na optymalizacji (jednokryterialnej) wykonywanej kolejno względem każdego kryterium. Kolejne kroki tej metody wyglądają następująco:

- Uszeregowanie kryteriów w porządku od najważniejszego f_1 do najmniej ważnego f_M.
- Znalezienie rozwiązania optymalnego X^1 względem kryterium f_1 i przy pierwotnych ograniczeniach.
- Poszukiwanie rozwiązań optymalnych $X^i, i = 2, 3, \ldots, M$ względem pozostałych kryteriów, przy wprowadzeniu dodatkowych ograniczeń:

$$f_{i-1}(X) \leqslant (1 \pm \frac{\varepsilon_{i-1}}{100}) \cdot f_{i-1}(X^{i-1}), \tag{2.13}$$

gdzie ε_i jest procentową wartością wariancji dozwoloną dla funkcji kryterialnej f_i. Wartość ta steruje wpływem optimum wyznaczonego w poprzednim kroku na optimum wyznaczane w kroku bieżącym. Jeżeli wartość wariancji przyjmuje wartość zero, wtedy taką metodę nazywa się leksykograficzną (ang. *lexicographic method*).

2.3.3 Metoda ograniczonych kryteriów

Kolejna opisywana metoda – metoda ograniczonych kryteriów (ang. *trade-off method, ε-constraint method*) – polega na ograniczeniu przestrzeni dopuszczalnych rozwiązań poprzez ustalenie poziomów wartości, jakie mogą przyjmować poszczególne kryteria. Problem optymalizacji wielokryterialnej jest rozwiązywany jako optymalizacja jednokryterialna względem wybranego kryterium f_r przy zwiększonej o $M - 1$ liczbie ograniczeń wynikających z pozostałych kryteriów. Formalnie można to przedstawić następująco:

$$\begin{aligned} & f_r(X) \to \text{MIN} \\ & f_i(X) \leqslant \varepsilon_i; i = 1, \ldots, M; i \neq r \\ & g_k(X) \leqslant 0; k = 1, \ldots, K \\ & h_j(X) = 0; j = 1, \ldots, J, \end{aligned} \tag{2.14}$$

gdzie ε_i jest ustalonym z góry współczynnikiem ograniczającym kryterium f_i.

2.3.4 Metoda programowania celów

Metoda programowania celów (ang. *goal programming*) charakteryzuje się tym, że poszczególne kryteria są traktowane jako cele, które należy osiągnąć, lub jako wartości ograniczające, których kryteria nie mogą przekroczyć. Oznacza to, że na wartości poszczególnych kryteriów mogą być nałożone warunki typu: większy lub równy, mniejszy lub równy, równy. Problem optymalizacji wielokryterialnej w ujęciu metody programowania celów można rozważyć na przykładzie dwóch funkcji: f_1 i f_2. Przyjmując, że pierwszy cel, a więc kryterium f_1, będzie mniejsze lub równe z_1, a drugie kryterium f_2 będzie równe z_2, wtedy opis formalny metody programowania celów można zapisać następująco:

$$\begin{aligned} & (w_1^+ d_1^+ + w_2^+ d_2^+ + w_2^- + d_2^-) \to \text{MIN} \\ & f_1(X) - d_1^+ \leqslant z_1 \\ & f_2(X) - d_2^+ + d_2^- = z_2, \end{aligned} \tag{2.15}$$

gdzie w_i są współczynnikami kary odpowiadającymi odchyłkom wartości poszczególnych kryteriów d_i.

Rozdział 3

Wielokryterialny problem wyszukiwania najkrótszej drogi w grafie

Tradycyjnie zadanie znalezienia najkrótszej drogi między dwoma węzłami w grafie było rozważane jako problem optymalizacji jednokryterialnej. Przy tym podejściu zakłada się, że z poszczególnymi krawędziami grafu związane są pewne wartości (np. odległość euklidesowa, czas podróży, koszt itd.), a celem optymalizacji jest znalezienie takiej drogi, dla której całkowita suma wartości dla krawędzi wchodzących w skład rozwiązania jest minimalna lub maksymalna. Znane są skuteczne algorytmy deterministyczne rozwiązujące ten problem. Przykładem mogą być algorytmy Bellmana-Forda, Dijkstry, Floyda-Warshalla, Johnsona czy algorytm A* [3, 5, 26, 37, 47, 48, 73, 119].

Sieci komunikacyjne i mapy drogowe są zazwyczaj reprezentowane w systemach komputerowych jako grafy skierowane, a taka reprezentacja wydaje się naturalna. W konsekwencji, rzeczywisty problem wyznaczenia optymalnej drogi między dwoma punktami na mapie jest rozpatrywany jako problem wyszukiwania najkrótszej drogi w grafie (ang. *shortest path problem*) [3]. Okazuje się, jednak, że wiele rzeczywistych problemów optymalizacyjnych można przekształcić lub sprowadzić do jednego z problemów występujących w teorii grafów. Przykładem może być algorytm Viterbiego [49], oparty na programowaniu dynamicznym, który jest powszechnie stosowany w systemach rozpoznawania mowy i pisma odręcznego oraz w systemach komunikacyjnych. Pomimo osobliwej nazwy, algorytm ten w zasadzie rozwiązuje problem wyszukiwania najkrótszej drogi w skierowanych grafach acyklicznych (typu DAG – ang. *directed acyclic graph*) [3]. Poszukiwanie sposobu na sformułowanie problemu w postaci grafu jest zazwyczaj bardzo dobrym kierunkiem prowadzącym do znalezienia skutecznej metody rozwiązania [108].

W rzeczywistych zastosowaniach często okazuje się, że jednokryterialna funkcja oceny rozwiązań jest niewystarczająca do odpowiedniego scharakteryzowania problemu. Droga uznawana za optymalną to niekoniecznie droga najkrótsza. W procesie optymalizacji znajdowane rozwiązania oceniane są pod kątem wielu kryteriów jakości, np. czasu przejazdu, liczby skrzyżowań, natężenia ruchu, w przypadku zastosowań cywilnych [13,14,19–22,100–102], lub np. zdolności do kamuflażu w zastosowaniach militarnych [115]. Innym przykładem zastosowania metod rozwiązywania wielokryterialnego problemu poszukiwania najkrótszej drogi są sieci komputerowe. Wybór drogi dla połączeń pakietowej transmisji danych najczęściej zależy od jakości usług (QoS) (ang. *quality of service*) [32],

gdzie brane są pod uwagę takie kryteria jak: liczba utraconych pakietów, czas opóźnienia, liczba przeskoków między routerami, natężenie ruchu.

W procesie optymalizacji dąży się do wyznaczenia jednego rozwiązania kompromisowego lub zbioru rozwiązań niezdominowanych.

3.1 Sformułowanie problemu

Poszukiwanie optymalnej drogi między dwoma punktami na mapie to bardziej ogólny wielokryterialny problem wyznaczenia najkrótszej drogi w grafie, znany w literaturze jako MOSP (ang. *multi-objective shortest path*). Formalnie problem ten można sformułować następująco [94]:

Sformułowanie 3.1 (Problem MOSP) *Niech $G = (V, E)$ oznacza graf skierowany, gdzie V jest zbiorem wierzchołków (węzłów), a E zbiorem krawędzi. Niech $c : E \to [\mathbb{R}^+]^m$ będzie m-wymiarowym wektorem funkcji kosztu. Z każdą krawędzią $e \in E$ jest związany wektor kosztu $c(e)$. Dany jest wierzchołek początkowy s i wierzchołek końcowy d. Droga p jest sekwencją wierzchołków i krawędzi z s do d. Wektor kosztu $C(p)$ dla drogi p jest sumą kosztów jej krawędzi: $C(p) = \sum_{e \in p} c(e)$. Niech $P(s, d)$ oznacza zbiór wszystkich możliwych dróg z wierzchołka s do wierzchołka d w grafie G. Zakładając, że wszystkie kryteria są minimalizowane, droga $p \in P(s, d)$ dominuje drogę $q \in P(s, d)$ jeżeli $\forall_{i \in m} C_i(p) \leqslant C_i(q)$, co jest zapisywane jako $p \preceq q$. Droga p jest paretooptymalna, jeżeli nie jest zdominowana przez inną drogę należącą do zbioru $P(s, d)$. Rozwiązaniem problemu MOSP jest zbiór niezdominowanych rozwiązań (dróg) $\mathbb{P} \in P(s, d)$ w odniesieniu do c.*

Problem ten należy do klasy problemów NP-zupełnych poprzez transformację z dyskretnego problemu plecakowego (ang. *0-1 knapsack problem*) [57, 70], co zachęca do stosowania metod heurystycznych, które działają szybko, choć nie gwarantują, że znalezione zostanie rozwiązanie optymalne lub kompletne, gdy poszukiwany jest zbiór rozwiązań. Przykładem takiej metody mogą być algorytmy mrowiskowe.

3.2 Przegląd algorytmów

Jedne z pierwszych i usystematyzowanych wyników badań nad dwukryterialnym problemem najkrótszej drogi w grafie (ang. *bicriteria shortest path problem*) zostały opublikowane przez Hansena w pracy [70]. Później Climaco i Martins [91], Martins [84], Hartley [71] zaproponowali dokładny algorytm etykietowania wierzchołków oparty na zasadzie programowania dynamicznego, służący do wyznaczania pełnych zbiorów dróg paretooptymalnych. Ponieważ problem jest NP-zupełny, algorytmy te nie mają złożoności wielomianowej, co skutkuje tym, że nawet umiarkowanych rozmiarów grafy mogą wymagać znacznego wysiłku obliczeniowego i obszernych zasobów pamięciowych, ponieważ zbiór wszystkich niezdominowanych dróg będących rozwiązaniem problemu może być bardzo liczny. Wielomianowe metody generowania prawie wszystkich paretooptymalnych dróg zaproponował Warburton w pracy [118]. Innym podejściem do generowania podzbioru dróg paretooptymalnych jest traktowanie części kryteriów jako ograniczeń. Sancho w pracy [103] rozważył problem z trzema kryteriami, ale dwa z nich traktował jako ograniczenia dla algorytmu opartego na programowaniu dynamicznym. Okazało się jednak, że niektóre z wyznaczonych dróg mogą nie być paretooptymalne w przypadku bez ograniczeń.

Cox [31] opracował wielokryterialny algorytm do wyznaczania zbioru paretooptymalnych dróg dla transportu niebezpiecznych materiałów radioaktywnych. Kryteriami oceny były między innymi czas podróży oraz gęstość zaludnienia. Martins i Santos w pracy [83] wykorzystali uogólnioną zasadę optymalności dla klasycznego problemu najkrótszej drogi w grafie i zaproponowali wielokryterialny algorytm etykietowania (ang. *the labelling algorithm*). Opracowali dwie wersje algorytmu nazwane przez nich: Label Setting oraz Label Correcting. Gandibleux i inni w pracy [54] zaproponowali modyfikację oryginalnego algorytmu Martinsa polegającą na wprowadzeniu ujednoliconego podejścia do kryteriów podlegających minimalizacji oraz maksymalizacji podczas wyznaczania zbioru rozwiązań paretooptymalnych. Ponieważ jest to rozszerzenie algorytmu Martinsa, generowanie wszystkich niezdominowanych dróg pozostaje zadaniem trudnym do wykonania w czasie wielomianowym. Jednak wyniki ich badań wykazują wysoką skuteczność algorytmu dla wielu przypadków testowych, zarówno pod względem złożoności pamięciowej jaki i złożoności obliczeniowej. Guerriero i Musmanno [66] przebadali pod kątem czasu wykonania kilka wersji algorytmów etykietowania, które wyznaczają rozwiązania paretooptymalne dla problemu MOSP. Eksperymenty przeprowadzono przy użyciu losowych grafów, a wyniki wskazują, że strategia wyboru etykiet (ang. *label selection*) jest na ogół szybsza niż strategia wyboru węzła (ang. *node selection*) oraz że równolegle przetwarzanie jest niezbędne przy projektowaniu efektywnych metod rozwiązywania problemu MOSP. Algorytmy etykietowania stosowane były do rozwiązywania różnych problemów komunikacyjnych, między innymi do optymalizacji połączeń autobusowych [120–122]. W pracy [93] opisanych i przetestowanych zostało 27 wariantów algorytmu etykietowania dla wielokryterialnego problemu najkrótszej drogi. Rozważane były różne techniki etykietowania (setting i correcting) z różnymi kombinacjami struktur danych. Eksperymenty obliczeniowe na dużej liczbie zestawów danych pokazały, że wersja algorytmu wykorzystująca kolejkę FIFO do wyboru etykiet była najbardziej sprawna.

Chociaż niektórzy badacze koncentrują się na rozwijaniu algorytmów dostarczających rozwiązania dokładne i pełne, inni badacze skupiają się na metodach bardziej efektywnych w sensie czasu wykonania. Tsaggouris i Zaroliagis [117] zaproponowali metodę aproksymacji działającą w czasie wielomianowym – FPTAS (ang. *fully polynomial time approximation scheme*) – dla wielokryterialnego problemu wyszukiwania najkrótszej drogi w grafie oraz ogólnej metody dla każdego problemu optymalizacji wielokryterialnej z nieliniowymi funkcjami celu. Zaproponowany algorytm opiera się na procesie iteracyjnym i różni się od wcześniejszych metod zastosowaniem technik zaokrąglania i skalowania na wartościach poszczególnych kosztów krawędzi grafu. Metoda przypomina algorytm Bellmana-Forda, ale implementuje zbiory etykiet jako tablice o wielomianowym rozmiarze poprzez rozluźnienie wymogów ścisłej paretooptymalności.

Granat i Guerriero [63] zaproponowali interaktywną procedurę rozwiązywania problemu MOSP bazującą na algorytmie etykietowania z punktem referencyjnym (ang. *reference point labelling algorithm*). Algorytm przekształca problem wielokryterialny do jednokryterialnego i był testowany na losowych zestawach danych, a jego wyniki oceniano głównie na podstawie czasu wykonania. Autorzy uważają, że zaproponowana przez nich interaktywna metoda jest zachęcająca głównie ze względu na fakt, że nie wymaga generowania pełnego zbioru rozwiązań paretooptymalnych (co pozwala uniknąć problemu wysokiej złożoności obliczeniowej). Podobnie, w pracy [30], zaproponowano interaktywny sposób wyznaczania paretooptymalnych dróg dla dwukryterialnego problemu MOSP, który stosuje skuteczny algorytm k-najkrótszych dróg (ang. *k-shortest path algorithm*). Algorytm był testowany

eksperymentalnie w celach porównawczych z innymi algorytmami do rozwiązywania problemu MOSP na 39 grafach, a autorzy doszli do wniosku, że ich algorytm działa lepiej pod względem czasu wykonania.

W literaturze znane są także rozwiązania oparte na wielokryterialnych algorytmach ewolucyjnych – MOEA (ang. *multi-objective evolutionary algorithms*). Gen i Lin [76] wykorzystali wielokryterialny, hybrydowy algorytm genetyczny w celu poprawy rozwiązań dla dwukryterialnego problemu projektowania sieci. Optymalizacja dotyczyła minimalizacji kosztu i maksymalizacji przepływu. W pracy przedstawiono metodę poprawy efektywności algorytmu genetycznego przez połączenie z lokalnym przeszukiwaniem oraz logiką rozmytą (ang. *fuzzy logic*). W pracy [75] zaprezentowano wielokryterialny algorytm optymalizujący sieć pod kątem minimalizacji opóźnień oraz kosztów transmisji przy zadanych ograniczeniach na przepływ oraz niezawodność. Autorzy przebadali algorytm genetyczny PCGA (ang. *pareto converging genetic algorithm*) do optymalizacji sieci różnej wielkości i uznali, że algorytm jest bardziej skalowalny niż inne tradycyjne podejścia. Jednym z wniosków autorów pracy jest stwierdzenie, że główną zaletą algorytmów genetycznych w zastosowaniach optymalizacji wielokryterialnej jest różnorodność rozwiązań dostarczanych w wielomianowym czasie. Mooney i Winstanley [89] zastosowali elitarny algorytm genetyczny (ang. *elitist genetic algorithm*) do rozwiązywania problemu MOSP w dziedzinie systemów informacji geograficznej (GIS). Przedmiotem eksperymentów było porównanie czasu wykonania algorytmu genetycznego oraz zmodyfikowanego algorytmu Dijkstry dla kilku sztucznych i rzeczywistych map drogowych. Rezultaty pokazały, że algorytm genetyczny jest konkurencyjny dla podejścia ze zmodyfikowanym algorytmem Dijkstry.

W pracy [94] Pangilinan przedstawił wyniki badań nad algorytmem ewolucyjnym w zastosowaniu do problemu MOSP i opisał jego zachowanie pod względem różnorodności rozwiązań, złożoności obliczeniowej i optymalności rozwiązań. Wyniki pokazują, że wielokryterialny algorytm ewolucyjny może znaleźć różne rozwiązania w czasie wielomianowym i może być alternatywą, gdy inne metody nie mogą być zastosowane ze względu na złożoność problemu.

Tarapata w pracy [116] przedstawił wybrane metody do rozwiązywania wielokryterialnych problemów poszukiwania najkrótszej drogi oraz sklasyfikował podstawowe odmiany tego problemu. Przeanalizował złożoność klasycznych metod rozwiązywania problemów MOSP oraz zaprezentował wyniki eksperymentów z przedstawionymi algorytmami na grafach reprezentujących informacje o topologii terenu.

Znane są również przykłady zastosowania metaheurystyki mrowiskowej do rozwiązywania wielokryterialnych problemów związanych z poszukiwaniem najkrótszej drogi w grafie. Więcej informacji na temat systemów mrowiskowych oraz ich zastosowań przedstawiono w rozdziale 4.

3.3 Algorytm Label Setting

Można stwierdzić, że idea algorytmu Label Setting Martinsa jest rozwinięciem algorytmu Dijkstry dla wielokryterialnego problemu poszukiwania najkrótszej drogi w grafie o nieujemnych wagach. W każdej iteracji, dla każdego wierzchołka są określane dwa różniące się zestawy etykiet: stałe i tymczasowe. Algorytm wybiera etykietę minimalną, według leksykograficznego porządku, ze wszystkich zestawów tymczasowych etykiet, konwertuje ją na stałą etykietę i propaguje informacje w niej zawarte do wszystkich tymczasowych

Algorytm 1 Pseudokod algorytmu LABEL SETTING

Dane wejściowe: $G = (V, E)$ oraz macierz kosztów C dla wszystkich krawędzi $(i, j) \in E$.
Dane wyjściowe: Zbiór wszystkich dróg paretooptymalnych z wierzchołka s do wszystkich wierzchołków $i \in V \setminus \{s\}$.

// l_i jest etykietą wierzchołka i,
// lt_i jest listą tymczasowych etykiet wierzchołka i,
// lp_i jest listą stałych etykiet wierzchołka i,
// $z^p_{q,h}$ jest p-tą wartością kryterium stałej etykiety wierzchołka q na pozycji h,
// $\preceq$ jest relacją dominacji (jeżeli $z' \preceq z$ to znaczy, że z jest zdominowane przez z').

$lt_i, lp_i \leftarrow \emptyset, \forall_i \in V$ ▷ Inicjacja
$lt_s \leftarrow \{[0, \ldots, 0, \perp, \perp]\}$

while $\bigcup_{i \in V} lt_i \neq \emptyset$ **do** ▷ Iteracja

    // Znajdź najmniejszą leksykograficzne etykietę z $lt_i, \forall_i \in V$
    $l_q \leftarrow \min lex\{\bigcup_{i \in V} lt_i\}$

    // Przenieś wybraną etykietę do listy etykiet stałych
    $lt_q \leftarrow lt_q \setminus \{l_q\}; lp_q \leftarrow lp_q \cup \{l_q\}$

    // Zapamiętaj pozycję etykiety l_q na liście lp_q
    $h \leftarrow |lp_q|$

    // Ustal etykiety dla następników q
    for all $j \in V | (q, j) \in E$ **do**

        // Oblicz etykietę l_j, aktualną etykietę wierzchołka j
        $l_j \leftarrow [z^1_{q,h} + c^1(q, j), \ldots, z^K_{q,h} + c^K(q, j), q, h]$

        // Sprawdź, czy nie istnieje etykieta dla wierzchołka j, która dominuje l_j
        if $\nexists l'_j \in \{lt_j \cup lp_j\} | l'_j \preceq l_j$ **then**

            // Zapamiętaj etykietę l_j wierzchołka j jako tymczasową
            $lt_j \leftarrow lt_j \cup \{l_j\}$

            // Usuń tymczasowe etykiety wierzchołka j zdominowane przez l_j
            $lt_j \leftarrow lt_j \setminus \{l'_j \in lt_j | l_j \preceq l'_j\}$
        end if
    end for
end while

etykiet swoich następników. Procedura zatrzymuje się, gdy nie ma już więcej tymczasowych etykiet. Informacje zawarte w i-tej etykiecie dla wybranego wierzchołka mogą być reprezentowane jako wektor:

$$l_i = [z^1, \ldots, z^K, j, h],$$

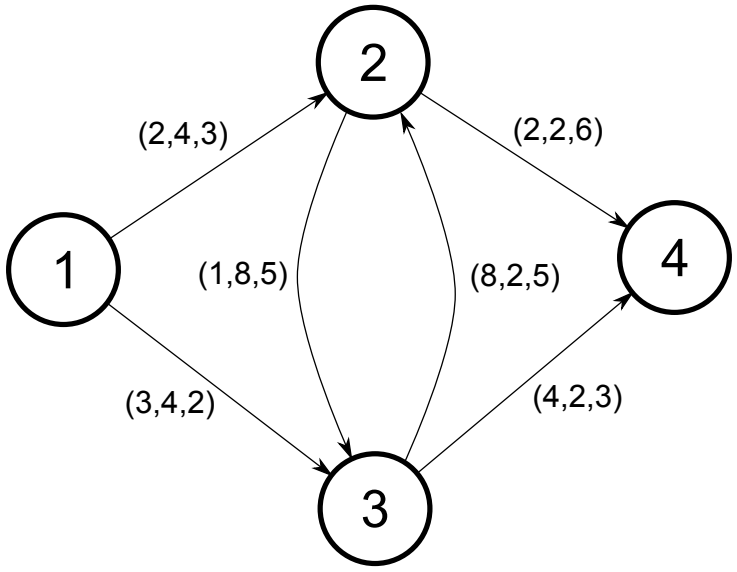
gdzie:

$(z^1, \ldots, z^K)$ – jest wektorem wartości kosztów dla drogi z wierzchołka źródłowego do bieżącego, która jest opisywana przez etykietę,

j – poprzedni wierzchołek na drodze,

h – pozycja poprzedniej etykiety na liście etykiet wierzchołka j.

Algorytm Martinsa jest przedstawiony w postaci pseudokodu jako Algorytm 1. Przykład z rys. 3.1 ilustruje działanie algorytmu dla problemu z trzema kryteriami. Zostaną wyznaczone wszystkie najkrótsze (paretooptymalne) drogi od wierzchołka $s = 1$ do wierzchołków $t = \{2, 3, 4\}$ dla grafu przedstawionego na rys. 3.1. Wszystkie kryteria będą minimalizowane.



Rysunek 3.1: Przykład grafu dla problemu z trzema kryteriami

1. Tymczasowa etykieta $[0, 0, 0, \perp, \perp]$ jest przypisana do wierzchołka $y = 1$. Jako pierwsza etykieta jest zamieniana na stałą. Następniki wierzchołka 1 są oznakowane i otrzymują dwie tymczasowe etykiety: $[2, 4, 3, 1, 1]$ wierzchołek 2 i $[3, 4, 2, 1, 1]$ – wierzchołek 3.

2. Z wszystkich tymczasowych etykiet, $[2, 4, 3, 1, 1]$ jest leksykograficznie najmniejsza i dlatego zostaje wybrana i zamieniona na stałą. Następniki wierzchołka 2 są oznakowane i otrzymują dwie nowe tymczasowe etykiety: $[3, 12, 8, 2, 1]$ – wierzchołek 3 i $[4, 6, 9, 2, 1]$ – wierzchołek 4. Nowa tymczasowa etykieta wierzchołka 3 jest zdominowana przez poprzednią $[3, 4, 2, 1, 1]$ i w związku z tym zostaje usunięta.

3. Spośród obecnie istniejących etykiet tymczasowych, $[3, 4, 2, 1, 1]$ jest leksykograficznie najmniejsza. Po wybraniu tej etykiety staje się ona etykietą stałą, a następniki wierzchołka 3 są oznakowane i otrzymują dwie tymczasowe etykiety: $[11, 6, 7, 3, 1]$ – wierzchołek 2 i $[7, 6, 5, 3, 1]$ – wierzchołek 4. Nowa etykieta wierzchołka 2 jest zdominowana przez etykietę $[2, 4, 3, 1, 1]$ i zostaje usunięta.

4. W następnym kroku wybierana jest etykieta $[4, 6, 9, 2, 1]$ i zamieniana na stałą. Wierzchołek 4 nie posiada następnika, dlatego nie są tworzone nowe etykiety tymczasowe.

5. Ostatnia etykieta tymczasowa pozostająca na liście, $[7, 6, 5, 3, 1]$, zostaje wybrana i staje się etykietą stałą. Ponieważ wierzchołek 4 nie ma następnika, nie jest tworzona nowa etykieta tymczasowa. Lista etykiet jest pusta, a więc algorytm kończy działania.

Tabela 3.1: Najkrótsze drogi dla opisywanego problemu z trzema kryteriami

Z węzła $s = 1$ do	Drogi	Wartości kryteriów
$t = 2$	$1 \rightarrow 2$	(2,4,3)
$t = 3$	$1 \rightarrow 3$	(3,4,2)
$t = 4$	$1 \rightarrow 2 \rightarrow 4$	(4,6,9)
	$1 \rightarrow 3 \rightarrow 4$	(7,6,5)

Każdej stałej etykiecie odpowiada unikalna droga. Aby określić jedną z tych dróg, należy wybrać stałą etykietę dla wierzchołka docelowego t i za pomocą wartości j i h ustalić etykietę odpowiadającą za poprzedni krok na drodze. Kroki należy powtarzać, aż do dojścia do początkowego węzła s. Tabela 3.1 zawiera wszystkie drogi paretooptymalne wychodzące z węzła $s = 1$.

Rozdział 4

Algorytmy optymalizacji mrowiskowej

Algorytmy optymalizacji mrowiskowej (ang. *ant colony optimization*) po raz pierwszy zostały zaproponowane do rozwiązywania problemów optymalizacji kombinatorycznej przez M. Dorigo w pierwszej połowie lat dziewięćdziesiątych ubiegłego wieku [40, 44] jako algorytmy mrówkowe (ang. *ant system*). Rozwój tych algorytmów został zainspirowany obserwacją kolonii rzeczywistych mrówek. Mrówki są owadami społecznymi, żyją w koloniach, a ich zachowanie jest podporządkowane przetrwaniu kolonii, a nie pojedynczego osobnika. Bezpośrednią inspiracją było zachowane mrówek podczas żerowania, a w szczególności sposób, w jaki mrówki potrafią znaleźć najkrótszą drogę pomiędzy źródłem pożywienia a mrowiskiem.

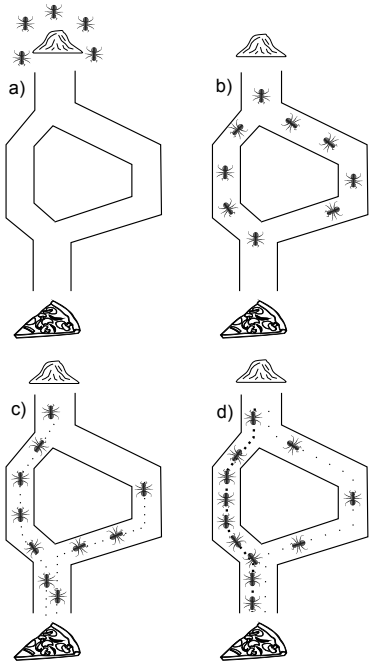
Podczas poszukiwania pożywienia, mrówki początkowo zwiedzają okolicę mrowiska w sposób całkowicie przypadkowy. Poruszając się pozostawiają na ziemi chemiczny ślad feromonowy, który potrafią wyczuć inne osobniki. Mrówki wybierając drogę, mają tendencję do wyboru dróg oznaczonych silniejszym stężeniem feromonu. Jak tylko mrówka znajdzie źródło żywności, ocenia ilość i jakość pożywienia, a część znalezionego pokarmu przenosi do mrowiska. Podczas drogi powrotnej ilość feromonu, który mrówka pozostawia na ziemi, może zależeć od ilości i jakości odnalezionej żywności. Ślad feromonowy prowadzi inne osobniki do odnalezionego pokarmu. Deneubourg i inni w pracach [35,60] wykazali, że komunikacja między mrówkami za pośrednictwem śladu feromonowego (zjawisko nazwane stygmergią [64]) pozwala mrówkom na znalezienie najkrótszej drogi od mrowiska do źródła żywności. Rys. 4.1 ilustruje sposób w jaki ślad feromonowy pozwala na wyznaczenie przez mrówki krótszej drogi od znalezionego pożywienia do mrowiska. Na początkowym etapie poszukiwań mrówki losowo wybierają drogę, ponieważ nie został jeszcze nałożony ślad feromonowy. W sytuacji gdy pierwsze mrówki odnajdą pożywienie, rozpoczynają powrót do mrowiska, jednocześnie nakładając na drodze feromon. Średnio więcej mrówek w jednostce czasu przejdzie krótszą drogą, więc, biorąc pod uwagę jego częściowe odparowanie, na krótszej drodze zostanie mocniejszy ślad feromonowy. Od tego momentu kolejne mrówki wyruszające z mrowiska w poszukiwaniu pokarmu chętniej wybierają drogę oznaczoną mocniejszym śladem feromonowym. Po przejściu kolejnych mrówek intensywność śladu na krótszej drodze jest coraz większa i w konsekwencji większość mrówek wybiera tę drogę.

Algorytm 2 Pseudokod ogólnego algorytmu ACO

1: **Dane wejściowe:** graf G, macierz kosztów C, liczba cykli N_c, liczba iteracji N_i, liczba mrówek N_m
2: **Dane wyjściowe:** najlepsze rozwiązanie T^+

3: **function** ACO(G, C, N_c, N_i, N_m)
4: INICJUJ
5: **for** $I_c \leftarrow 1$ **to** N_c **do** ▷ Dla każdego cyklu
6: USTAWMRÓWKI
7: **for** $I \leftarrow 1$ **to** N_i **do** ▷ Dla każdej iteracji
8: **for** $k \leftarrow 1$ **to** N_m **do** ▷ Dla każdej mrówki
9: WYZNACZKRAWĘDŹ ▷ Wybór krawędzi zgodnie z regułą przejścia
10: DODAJKRAWĘDŹDOROZWIĄZANIA
11: AKTUALIZUJFEROMONLOKALNIE ▷ Krok opcjonalny
12: **end for**
13: **end for**
14: OCEŃMRÓWKI ▷ Oceń mrówki oraz aktualizuj T^+
15: ODPARUJFEROMON ▷ Krok opcjonalny
16: AKTUALIZUJFEROMONGLOBALNIE ▷ Krok opcjonalny
17: **end for**
18: **return** T^+ ▷ Zwróć najlepsze rozwiązanie
19: **end function**

Ogólna postać algorytmów optymalizacji mrowiskowej zaprezentowana jest w postaci pseudokodu jako Algorytm 2, a w literaturze zaproponowano kilka głównych rodzajów algorytmów wykorzystujących metaheurystykę ACO. Wśród algorytmów ACO dla NP-trudnych problemów optymalizacyjnych są to: system mrówkowy (ang. *ant system*) [43], system mrowiskowy (ang. *ant colony system*) [42], system mrówkowy Max-Min (ang. *max-min ant system*) [113,114], system mrówkowy z rankingiem (ang. *rank-based ant system*) [18], system mrówkowy najlepszy-najgorszy (ang. *best-worst ant system*) [24]. Poszczególne algorytmy zostaną krótko opisane w dalszej części rozdziału.



Rysunek 4.1: Mrówki poszukujące pożywienia

4.1 System mrówkowy

System mrówkowy (ang. *ant system*) [40,44] został zaproponowany przez Dorigo, Maniezzo i Colorniego w 1991 roku i był pierwszym algorytmem ACO. Podstawą do opracowania algorytmu był model matematyczny funkcjonowania kolonii mrówek [61], który autorzy wykorzystali do rozwiązywania problemu TSP (ang. *travelling sale-*

sman problem) [104].

Problem komiwojażera jest to zadanie optymalizacyjne, które polega na znalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważonym. Nazwa pochodzi od najbardziej popularnej ilustracji problemu, przedstawiającej go z punktu widzenia wędrownego sprzedawcy (komiwojażera): dane jest n miast, które komiwojażer ma odwiedzić, oraz odległość pomiędzy każdą parą miast. Celem jest znalezienie najkrótszej drogi łączącej wszystkie miasta zaczynającej się i kończącej się w określonym mieście. Każde miasto powinno być odwiedzone tylko raz. Problem ten został sformułowany po raz pierwszy w 1930 roku i jest jednym z najintensywniej badanych problemów optymalizacji [95]. Jako jeden z problemów NP-trudnych jest bardzo często używany jako punkt odniesienia w porównaniach różnych algorytmów optymalizacyjnych.

Początkowo zostały zaproponowane trzy różne warianty algorytmu mrówkowego: feromon stały (ang. *ant-density*), feromon średni (ang. *ant-quantity*) i feromon cykliczny (ang. *ant-cycle*), różniące się sposobem, w jaki aktualizowany jest ślad feromonowy. W dwóch pierwszych wariantach mrówki nakładają ślad feromonowy podczas budowania rozwiązania (po każdym przejściu między wierzchołkami grafu), z tą różnicą, że w przypadku feromonu średniego nakładana ilość zależy od heurystycznej informacji związanej z przejściem z węzła i do węzła j ($\eta_{i,j}$ – widoczność miasta j z miasta i, czyli odwrotność odległości pomiędzy tymi miastami). W przypadku ostatniej wersji (feromon cykliczny) ślad feromonowy jest aktualizowany przez mrówkę po zbudowaniu rozwiązania. Ponieważ użycie tego wariantu dawało najlepsze wyniki podczas eksperymentów, to jest on bezpośrednio kojarzony z systemem mrówkowym.

W systemie mrówkowym rozwiązania są budowane w następujący sposób [25]. Przed każdym krokiem mrówki k z węzła i do węzła j wyznaczane jest prawdopodobieństwo przejścia według wzoru:

$$p_{i,j}^k(t) = \begin{cases} \frac{[\tau_{i,j}(t)]^\alpha \cdot [\eta_{i,j}]^\beta}{\sum_{r \in J_i^k} [\tau_{i,r}(t)]^\alpha \cdot [\eta_{i,r}]^\beta}, & \text{jeśli } j \in J_i^k \\ 0, & \text{w przeciwnym razie,} \end{cases} \tag{4.1}$$

gdzie:

J_i^k – zbiór decyzji, jakie mrówka k może podjąć będąc w węźle i,

$\tau_{i,j}$ – wartość nagrody, czyli stopień użyteczności branej pod uwagę decyzji (ilość feromonu),

$\eta_{i,j}$ – wartość heurystycznie oszacowanej jakości przejścia z węzła i do węzła j (np. widoczność miasta j z miasta i dla problemu TSP),

α – parametr określający ważność wartości $\tau_{i,j}$,

β – parametr określający ważność wartości $\eta_{i,j}$.

Każda mrówka k przechowuje sekwencję swoich kroków w pamięci L_k, która została nazwana listą *TABU*. Struktura ta stanowi rejestr odwiedzonych przez mrówkę węzłów (miast dla problemu TSP) w bieżącym cyklu działania algorytmu oraz może być wykorzystana do odtworzenia pełnej drogi, którą przeszła mrówka. Należy zwrócić uwagę, że definicja i znaczenie listy *TABU* w przypadku algorytmów mrówkowych są odmienne niż w przypadku innej znanej metody optymalizacyjnej Tabu Search.

Rola parametrów α i β jest następująca: jeżeli $\alpha = 0$ to większe prawdopodobieństwo wyboru mają krawędzie o lepszej informacji heurystycznej, dzięki czemu algorytm zbliżony jest do klasycznego algorytmu zachłannego. Jednakże, gdy $\beta = 0$, to tylko ślad feromonowy ma czynny udział w konstrukcji rozwiązania, co może prowadzić do szybkiej stagnacji w rejonie optimów lokalnych. Stąd też istnieje potrzeba ustanowienia właściwej równowagi pomiędzy wpływem informacji heurystycznej, a wpływem śladu feromonowego.

Po pełnym cyklu algorytmu, gdy wszystkie mrówki zbudowały swoje rozwiązania, następuje faza aktualizacji śladu feromonowego. W pierwszej kolejności następuje odparowanie śladu z każdej krawędzi według następującego wzoru:

$$\tau_{i,j}(t+1) = (1-\rho) \cdot \tau_{i,j}(t), \tag{4.2}$$

gdzie:

ρ – współczynnik wyparowania śladu, $0 \leqslant \rho \leqslant 1$.

W dalszej kolejności, każda mrówka odtwarza swoją drogę L_k i nakłada feromon na każdej przebytej krawędzi:

$$\tau_{i,j}(t+1) = \tau_{i,j}(t) + \Delta\tau_{i,j}^{k}, \forall e_{i,j} \in S_k, \tag{4.3}$$

gdzie:

$\Delta\tau_{i,j}^{k} = f(C(S_k))$ – ilość nałożonego feromonu, zależna od jakości rozwiązania S_k mrówki k.

Twórcy systemu mrówkowego zaproponowali również, zazwyczaj bardziej skuteczną, rozszerzoną wersję tego algorytmu o oryginalnej nazwie AS-elitist [43]. W tej wersji algorytmu wprowadzono dodatkowo wzmocnienie śladu feromonowego na krawędziach należących do najlepszego rozwiązania znalezionego dotychczas, a wielkość zmiany feromonu uzależniono od jakości tego rozwiązania:

$$\tau_{i,j}(t+1) = \tau_{i,j}(t) + K^{+} \cdot f(C(S^{+})), \forall e_{i,j} \in S^{+}, \tag{4.4}$$

gdzie:

K^{+} – liczba mrówek należących do „elity” (ang. *elitist ant*),

S^{+} – rozwiązanie dotychczas najlepsze.

Analiza przeprowadzona przez Dorigo, Maniezzo i Colorniego [44] na przykładzie problemu TSP wykazała, że czasowa złożoność obliczeniowa algorytmu opartego na systemie mrówkowym jest rzędu:

$$T(c, m, n) = O(c \cdot (m \cdot n + n^3)) \tag{4.5}$$

gdzie:

c – liczba iteracji algorytmu,

m – liczba mrówek,

n – liczba miast.

Eksperymentalnie wykazali, że optymalna liczba mrówek wynosi $m = s_1 \cdot n$, gdzie s_1 jest niewielką stałą zbliżoną do jedności. W konsekwencji równanie (4.5) można zapisać:

$$T(c, n) = O(c \cdot n^3). \tag{4.6}$$

4.2 System mrowiskowy

Ulepszoną wersją systemu mrówkowego jest system mrowiskowy (ang. *ant colony system*). Do podstawowych ulepszeń zalicza się wprowadzenie dodatkowych elementów, takich jak wybór pomiędzy eksploracją a eksploatacją oraz modyfikację sposobu odkładania śladu feromonowego [41].

W systemie mrowiskowym wirtualna mrówka będąc w konkretnej chwili czasu i na określonym etapie rozwiązania problemu, podejmuje decyzję o wyborze następnego kroku na podstawie zmodyfikowanej reguły przejścia. Generuje w tym celu losową liczbę q, $0 \leqslant q \leqslant 1$. Jeśli $q \leqslant q_0$ (q_0 – zadany parametr algorytmu), to wybierana jest „najlepsza" dostępna decyzja (deterministycznie – eksploatacja), w przeciwnym przypadku decyzja jest podejmowana losowo (eksploracja), biorąc pod uwagę prawdopodobieństwa wyliczane zgodnie ze wzorem (4.7) [41].

$$j = \begin{cases} \arg\max\limits_{r \in J_i^k} \{[\tau_{i,r}(t)] \cdot [\eta_{i,r}]^\beta\}, & \text{jeśli } q \leqslant q_0 \text{ (eksploatacja)} \\ S, & \text{w przeciwnym razie (eksploracja),} \end{cases} \tag{4.7}$$

gdzie:

$\tau_{i,j}$ – wartość nagrody, czyli stopień użyteczności branej pod uwagę decyzji (gęstość feromonu),

$\eta_{i,j}$ – wartość heurystycznie oszacowanej jakości przejścia ze stanu i do stanu j (np. widoczność miasta j z miasta i dla problemu TSP),

β – parametr określający ważność wartości $\eta_{i,j}$,

S – kolejna decyzja (kolejne miasto dla problemu TSP) wylosowana z zastosowaniem prawdopodobieństw:

$$p_{i,j}^k(t) = \begin{cases} \frac{\tau_{i,j}(t) \cdot [\eta_{i,j}]^\beta}{\sum\limits_{r \in J_i^k} \tau_{i,r}(t) \cdot [\eta_{i,r}]^\beta}, & \text{jeśli } j \in J_i^k \\ 0, & \text{w przeciwnym razie,} \end{cases} \tag{4.8}$$

gdzie J_i^k jest zbiorem decyzji, jakie mrówka k może podjąć będąc w stanie i.

Po wykonaniu każdego kroku mrówka nakłada ślad feromonowy na wybranej przez siebie krawędzi. Jest to aktualizacja lokalna śladu feromonowego, związana również z jego częściowym odparowaniem. Odbywa się ona zgodnie ze wzorem (4.9):

$$\tau_{i,j}(t+1) = (1 - \rho) \cdot \tau_{i,j}(t) + \rho \cdot \tau_0, \tag{4.9}$$

gdzie:

ρ – współczynnik wyparowania śladu feromonowego, $0 \leqslant \rho \leqslant 1$,

τ_0 – wartość równa śladowi początkowemu.

Jednocześnie, aktualnie wybrany węzeł (do którego prowadzi wybrana krawędź) jest dodawany do listy *TABU*, która zawiera listę węzłów odwiedzonych już przez mrówkę w danym cyklu.

Po wykonaniu przez mrówki pełnego cyklu następuje globalne uaktualnianie śladu feromonowego. Ślad jest nakładany na krawędzie należące do najlepszej znalezionej trasy, a poziom feromonu zmieniany jest z zastosowaniem reguły opisanej wzorem (4.10):

$$\tau_{i,j}(t+n) = (1-\gamma) \cdot \tau_{i,j}(t) + \gamma \cdot \frac{1}{L^+}, \tag{4.10}$$

gdzie decyzje przejścia z i do j należą do najlepszego rozwiązania, γ oznacza szybkość wyparowania feromonu, a L^+ jest długością najkrótszej drogi (najlepszego rozwiązania).

4.3 System mrówkowy Max-Min

System mrówkowy Max-Min [112–114] (MMAS – ang. *max-min ant system*) został opracowany przez Stützle i Hoosa w 1996 roku i jest jednym z najlepszych rozszerzeń systemu mrówkowego. Modyfikacji uległy następujące elementy oryginalnego systemu mrówkowego:

- Zastosowana została aktualizacja śladu feromonowego po pełnym cyklu (offline), po zbudowaniu rozwiązań przez wszystkie mrówki. W pierwszej kolejności ślad feromonowy jest odparowywany według formuły:

  $$\tau_{i,j}(t+1) = (1-\rho) \cdot \tau_{i,j}(t), \tag{4.11}$$

  a następnie jest nakładany zgodnie ze wzorem:

  $$\tau_{i,j}(t+1) = \tau_{i,j}(t) + f(C(S^+)), \forall e_{i,j} \in S^+, \tag{4.12}$$

  gdzie:

  S^+ – najlepsze rozwiązanie w cyklu lub rozwiązanie dotychczas najlepsze (globalnie).

  Eksperymenty wykazały, że najlepsze wyniki uzyskuje się przez stopniowe zwiększenie częstotliwości wyboru globalnie najlepszego rozwiązania jako podstawy do aktualizacji śladu feromonowego [113, 114].

- Gęstość śladu feromonowego może przyjmować wartości z przedziału $[\tau_{min}, \tau_{max}]$. Szansa na wejście algorytmu w fazę stagnacji jest przez to ograniczona, dzięki zwiększeniu szansy wybrania poszczególnych połączeń między węzłami.

- W miejsce inicjowania śladu feromonowego niewielką ilością feromonu, w przypadku algorytmu MMAS, ślad inicjowany jest szacowaną wartością maksymalną. Szacowanie może polegać na wstępnym wyznaczeniu rozwiązania za pomocą algorytmu zachłannego i uwzględnieniu ograniczenia τ_{max}.

4.4 Algorytm mrówkowy z rankingiem

Kolejnym algorytmem mrówkowym jest algorytm z rankingiem (ang. *rank-based ant system*) zaproponowany w 1997 roku przez Bullnheimera, Hartla i Straussa [18]. Opiera się na pomyśle wprowadzenia listy rankingowej podczas aktualizacji śladu feromonowego, która, podobnie jak w przypadku MMAS, jest wykonywana po pełnym cyklu według zasad:

- m mrówek jest oznaczonych rangą według malejącego porządku wyznaczonego przez jakość rozwiązań poszczególnych mrówek: $(S'_1, \ldots, S'_m)$, gdzie S'_1 jest najlepszym rozwiązaniem w bieżącym cyklu.
- Ślad feromonowy jest nakładany na krawędzie odwiedzone przez $\sigma - 1$ najlepszych mrówek. Ilość nałożonego feromonu bezpośrednio zależy od pozycji mrówki na liście rankingowej oraz od jakości jej rozwiązania.
- Na krawędziach należących do drogi odpowiadającej globalnie najlepszemu rozwiązaniu nakładana jest dodatkowa ilość feromonu, która zależy od jakości tego rozwiązania. Ta dawka feromonu jest uznawana za najbardziej znaczącą i dlatego otrzymuje wagę równą σ.

Aktualizację feromonu można przedstawić formalnie za pomocą następującej reguły, która jest wykonywana po fazie odparowania śladu feromonowego:

$$\tau_{i,j}(t+1) = \tau_{i,j}(t) + \sigma \cdot \Delta\tau^{+}_{i,j} + \Delta\tau^{rank}_{i,j}, \tag{4.13}$$

gdzie:

- $\Delta\tau^{+}_{i,j} = \begin{cases} f(C(S^+)), & \text{jeśli } e_{i,j} \in S^+ \\ 0, & \text{w przeciwnym razie,} \end{cases}$
- $\Delta\tau^{rank}_{i,j} = \begin{cases} \sum\limits_{\mu=1}^{\sigma-1} (\sigma - \mu) \cdot f(C(S'_\mu)), & \text{jeśli } e_{i,j} \in S'_\mu \\ 0, & \text{w przeciwnym razie,} \end{cases}$
- σ – liczba najlepszych mrówek,
- μ – indeks rozwiązania mrówki na liście rankingowej,
- S^+ – rozwiązanie najlepsze w cyklu,
- S'_μ – rozwiązanie o indeksie μ na liście rankingowej,
- C – funkcja określająca ilość feromonu nakładanego na krawędzi w zależności od jakości rozwiązania.

4.5 Algorytm mrówkowy „najlepszy-najgorszy"

System mrówkowy „najlepszy-najgorszy" (BWAS – ang. *best-worst ant system*) został zaproponowany w 1999 roku przez Cordona [24], a jego cechą szczególną jest wykorzystanie elementów obliczeń ewolucyjnych. Algorytm jest kolejnym rozszerzeniem systemu mrówkowego i wykorzystuje standardową regułę przejścia oraz odparowania śladu feromonowego. Podobnie jak w przypadku algorytmu MMAS, BWAS wykorzystuje przeszukiwanie lokalne w celu polepszenia rozwiązań wyznaczanych przez mrówki. Następujące elementy są kluczowe dla systemu mrówkowego BWAS:

- Reguła aktualizacji śladu feromonowego wzmacnia ślad na drodze najlepszego rozwiązania (S^+) oraz osłabia ślad na krawędziach należących do najgorszego rozwiązania (S^-) w bieżącym cyklu algorytmu. Można to zapisać formalnie:

$$\tau_{i,j}(t+1) = \begin{cases} \tau_{i,j}(t) + \rho \cdot f(C(S^+)), & \forall e_{i,j} \in S^+ \\ (1-\rho) \cdot \tau_{i,j}(t), & \forall (e_{i,j} \in S^- \bigwedge e_{i,j} \notin S^+) \end{cases} \tag{4.14}$$

- Wprowadzono mutację śladu feromonowego. Została ona wprowadzona po to, aby dodatkowo różnicować proces poszukiwania rozwiązań. W tym celu jedna z krawędzi wychodzących z każdego wierzchołka jest poddawana mutacji z prawdopodobieństwem P_m. Oryginalny algorytm stosuje operator mutacji, który dodaje lub odejmuje taką samą wartość feromonu. Zakres zmiany jest określony jako $mut(t, \tau_m)$ i jest zależny od średniej wartości śladu feromonowego dla najlepszego rozwiązania S^+. Parametr τ_m ma małą wartość w początkowych iteracjach algorytmu, kiedy prawdopodobieństwo stagnacji jest małe, a następnie stopniowo rośnie, aby zrównoważyć ryzyko stagnacji na późniejszych etapach działania algorytmu:

$$\tau'_{i,j} = \begin{cases} \tau_{i,j} + mut(t, \tau_m), & \text{jeśli } a = 0 \\ \tau_{i,j} - mut(t, \tau_m), & \text{jeśli } a = 1, \end{cases} \tag{4.15}$$

  gdzie a jest losową wartością ze zbioru $\{0, 1\}$.

- Algorytm BWAS, podobnie jak niektóre inne algorytmy z rodziny ACO, zakłada możliwość powtórnej inicjacji śladu feromonowego w sytuacji, gdy algorytm wpadnie w stagnację. Realizowane jest to przez ustawienie śladu na wszystkich krawędziach na wartość początkową τ_0.

4.6 Wielokryterialna optymalizacja mrowiskowa

Algorytmy oparte na metaheurystyce mrowiskowej można zastosować także do rozwiązywania problemów optymalizacji wielokryterialnej. W pracy [56] Garcia-Martinez i inni zaproponowali podział mrowiskowych algorytmów optymalizacji wielokryterialnej (MOACO – ang. *multi-objective ant colony optimization*) według dwóch kryteriów:

- wykorzystanie jednego lub wielu śladów feromonowych,
- wykorzystanie jednej lub wielu macierzy z informacją heurystyczną.

Część klasyfikowanych w ten sposób algorytmów oparta jest na podejściu Pareto i w wyniku działania dostarcza zbiór rozwiązań niezdominowanych (aproksymację pełnego zbioru Pareto), inne zwracają jedno rozwiązanie. Szkielet algorytmu mrowiskowego uwzględniającego wiele kryteriów, który poszukuje zbioru rozwiązań aproksymujących front Pareto, przedstawiony został w postaci pseudokodu jako Algorytm 3.

Algorytm 3 Pseudokod algorytmu mrowiskowego uwzględniającego wiele kryteriów

```
1: Dane wejściowe: graf G, macierz kosztów C, liczba cykli N_c, liczba iteracji N_i, liczba mrówek N_m
2: Dane wyjściowe: przybliżony zbiór rozwiązań paretooptymalnych ℙ

3: function MultiACO(G, C, N_c, N_i, N_m)
4:     InicjujFeromon
5:     InicjujInformacjęHeurystyczną
6:     InicjujZbiórPareto
7:     for I_c ← 1 to N_c do                          ▷ Dla każdego cyklu
8:         BudujRozwiązaniaDlaMrówek(N_i, N_m)
9:         AktualizujZbiórPareto                      ▷ Aktualizuj zbiór rozwiązań ℙ
10:        AktualizujFeromon
11:    end for
12:    return ℙ                                       ▷ Zwróć przybliżony zbiór Pareto
13: end function
```

W procedurze BudujRozwiązaniaDlaMrówek mrówki tworzą rozwiązania na podstawie poprzedniego stanu śladu feromonowego, przyjętej strategii wyboru kolejnego odwiedzanego węzła oraz informacji heurystycznej związanej z instancją problemu. Po zakończeniu budowy rozwiązań przez wszystkie mrówki zbiór rozwiązań niezdominowanych (zbiór Pareto) jest aktualizowany w procedurze AktualizujZbiórPareto. Następnie w procedurze AktualizujFeromon modyfikowany jest ślad feromonowy. Część pozostawionego śladu feromonowego jest zwykle odparowywana, a część zmieniana na podstawie otrzymanych rozwiązań, biorąc pod uwagę ich jakość.

W literaturze znaleźć można wiele różnych implementacji algorytmów mrowiskowych rozwiązujących problemy wielokryterialne. Zwykle różnią się między sobą strategią wyboru kolejnego węzła oraz sposobem aktualizacji śladu feromonowego, tak samo jak różnią się odpowiadające im wersje jednokryterialne (AS, ACS, MMAS), na bazie których zostały opracowane. Oczywiście różnice występują także w niektórych cechach algorytmów związanych z obsługą wielu kryteriów optymalizacji. Zostały one szerzej opisane poniżej.

Wiele kolonii. W podejściu z wieloma koloniami cała populacja mrówek jest dzielona na niezależne grupy zwane koloniami. Każda kolonia tworzy rozwiązania w sposób niezależny, na podstawie własnego śladu feromonowego i informacji heurystycznej, specjalizując się w ramach wybranego fragmentu frontu Pareto. Kolonie mogą ze sobą współpracować poprzez: (i) wymianę rozwiązań przez wspólny rejestr rozwiązań niezdominowanych; (ii) wymianę rozwiązań w celu aktualizacji śladu feromonowego w ramach kolonii.

Ślad feromonowy i informacja heurystyczna. Istnieją dwa podstawowe modele wykorzystywane do opisywania śladu feromonowego i informacji heurystycznej:

wykorzystanie pojedynczej macierzy lub wielu macierzy. W przypadku gdy wykorzystywanych jest wiele macierzy, zwykle każda z nich koresponduje z innym kryterium optymalizacji. W przypadku użycia pojedynczej macierzy krok konstrukcji rozwiązania jest wykonywany w sposób bardzo podobny do jednokryterialnych algorytmów ACO. W tym przypadku ślad feromonowy związany z każdym kryterium powinien być połączony w jeden. To samo dotyczy informacji heurystycznej.

Agregacja śladu feromonowego i informacji heurystycznej. W przypadku gdy jest używanych wiele niezależnych macierzy konieczne może być przyjęcie jednej z metod agregacji danych związanych z poszczególnymi kryteriami. W literaturze można spotkać trzy podstawowe takie metody:

- sumę ważoną, gdzie macierze są agregowane jako wyrażenie $\sum_{m=1}^{M} \lambda_m \cdot \tau_{ij}^m$, w którym M oznacza liczbę kryteriów, a λ_m oznacza wagę przydzieloną wybranemu kryterium;
- ważony produkt, gdzie macierze są agregowane jako wyrażenie $\prod_{m=1}^{M} (\tau_{ij}^m)^{\lambda_m}$, w którym M i λ_m mają takie same znaczenie jak w przypadku sumy ważonej;
- wybór losowy, gdzie przed każdym krokiem losowo wybierane jest kryterium, które jest optymalizowane.

W przypadku gdy wykorzystywana jest metoda sumy ważonej lub ważonego produktu można wyróżnić dwie strategie ustalania wartości wag: dynamiczną, gdzie wagi dla poszczególnych kryteriów mogą się zmieniać na różnych etapach konstruowania rozwiązań, oraz statyczną, gdzie wagi mają ustalone wartości na stałe w trakcie działania algorytmu.

Aktualizacja śladu feromonowego. Ten element algorytmów MOACO może być realizowany na wiele różnych sposobów. Gdy wykorzystywana jest pojedyncza macierz feromonowa aktualizacja jest zwykle wykonywana w taki sam sposób jak w przypadku optymalizacji jednokryterialnej, na przykład wzmacniając ślad na drodze najlepszego rozwiązania w iteracji lub najlepszego rozwiązania znalezionego dotychczas. Podobnie można postąpić w przypadku mrowiskowej optymalizacji wielokryterialnej, ale mając na uwadze oddzielne kryteria oceny, na przykład wzmacniając ślad dla najlepszego rozwiązania dla danego kryterium. Innym podejściem jest utrzymywanie zbioru rozwiązań niezdominowanych i wzmacnianie śladu w obrębie tych rozwiązań.

Tabela 4.1: Możliwe wartości cech wielokryterialnych algorytmów mrowiskowych

Cecha	Wartości
Kolonia	pojedyncza, wiele
Macierz feromonu	pojedyncza, wiele
Macierz heurystyki	pojedyncza, wiele
Agregacja	produkt ważony, suma ważona, losowa
Wagi	dynamiczne, statyczne
Aktualizacja feromonu	niezdominowane, najlepsze dla kryterium, najlepsze, wszystkie
Zbiór Pareto	offline, online, brak

Zbiór Pareto. Dla algorytmów MOACO opartych na podejściu Pareto wymagane jest utrzymywanie zbioru rozwiązań niezdominowanych w trakcie całego przebiegu algorytmu. W wielu przypadkach ślad feromonowy jest aktualizowany na podstawie rozwiązań będących elementami tego zbioru. W literaturze [2] można spotkać dwie metody obsługi zbioru rozwiązań niezdominowanych. W pierwszej metodzie (*offline*) rozwiązania są zapisywane w zbiorze, ale nie są wykorzystywane na dalszym etapie konstruowania rozwiązań. Zbiór ten jest wykorzystywany do aktualizacji śladu feromonowego, a po zakończeniu działania algorytmu jest zwracany jako końcowe rozwiązanie. W drugiej metodzie (*online*) rozwiązania w zbiorze Pareto są połączone z procedurą aktualizacji feromonu. Za każdym razem gdy zbiór rozwiązań niezdominowanych zostanie poprawiony procedura jest o tym powiadamiana. Umożliwia to ciągłą aktualizację śladu feromonowego, który odpowiada aktualnej zawartości zbioru Pareto. Istnieją także metody, w których zbiór Pareto nie jest używany do aktualizacji śladu feromonowego, ale jest zwracany jako ostateczne rozwiązanie.

W tabeli 4.1 zaprezentowano zbiory możliwych wartości dla poszczególnych cech wielokryterialnych algorytmów mrowiskowych.

4.7 Zastosowania algorytmów mrowiskowych

Opisane w tym rozdziale algorytmy optymalizacji mrowiskowej znalazły wiele zastosowań do rozwiązywania teoretycznych oraz rzeczywistych problemów optymalizacyjnych, które mogą być reprezentowane za pomocą grafu. Duża przydatność metaheurystyki ACO w tej dziedzinie przyczyniła się do szybkiego rozwoju tej metody. Aktualnie znanych jest wiele zastosowań systemów mrowiskowych do rozwiązywania popularnych problemów optymalizacyjnych, co przedstawia tab. 4.2 [9].

Poza tym znanych jest bardzo wiele zastosowań branżowych, między innymi są to problemy optymalizacji dystrybucji wody [123], sieci komputerowych [27, 36] oraz wiele problemów związanych z transportem. Przykładami tych ostatnich problemów mogą być: optymalizacja sieci połączeń portów kontenerowych [69], czy planowanie ruchu pojazdów w zaawansowanych systemach logistycznych [52],

W przypadku sieci komputerowych istotna jest optymalizacja dróg pakietów od węzła źródłowego do docelowego. Optymalizacja sprowadza się do minimalizacji liczby przeskoków pakietów między urządzeniami oraz maksymalizacji jakości połączeń [27, 36].

Ogólne podejście do optymalizacji wielokryterialnej z zastosowaniem metaheurystyki mrowiskowej oraz przegląd różnych opracowanych algorytmów przedstawili w swoich pracach między innymi: Ghezail [58], Lopez [77] oraz Angelo [2].

W pracy [88] Moncayo i Zhang zaproponowali algorytm optymalizacji Pareto oparty na metaheurystyce mrowiskowej (ang. *pareto ant colony optimization*) do wyznaczania optymalnego łańcucha dostaw dla rodziny produktów. Produkty zawierają skomplikowane hierarchie podzespołów i komponentów. Łańcuch dostaw może obejmować wielu dostawców, oferujących te same podzespoły oraz różne zakłady produkcyjne. Zadaniem algorytmu jest minimalizacja całkowitych kosztów łańcucha dostaw przy dotrzymaniu wymaganych czasów realizacji.

Tematyką związaną z wielokryterialnym problemem poszukiwania najkrótszej drogi zajmowali się między innymi Ghoseiri i Nadjari, którzy w pracy [59] zaproponowali mrowiskowy algorytm rozwiązujący dwukryterialny problem najkrótszej ścieżki. Autorzy

Tabela 4.2: Wybrane przykłady zastosowań algorytmów optymalizacji mrowiskowej

Zastosowanie	Autorzy	Pozycje
Problem komiwojażera	Dorigo, Maniezzo, Colorni	[39, 43, 44]
	Dorigo, Gambardella	[42]
	Stützle, Hoos	[114]
Problem kwadratowego przydziału	Maniezzo	[79]
	Maniezzo, Colorni	[81]
	Stützle, Hoos	[114]
Problem szeregowania	Stützle	[111]
	den Besten, Stützle, Dorigo	[34]
	Gagné, Price, Gravel	[50]
	Merkle, Middendorf, Schmeck	[85]
	Blum, Sampels	[10, 11]
Problem marszrutowania	Gambardella, Taillard, Agazzi	[53]
	Reimann, Doerner, Hartl	[98]
Problem układania harmonogramów	Socha, Sampels, Manfrin	[109]
Problem pakowania zbioru	Gandibleux, Delorme, T'Kindt	[55]
Problem kolorowania grafów	Costa, Hertz	[29]
Problem najkrótszego wspólnego nadłańcucha	Michel, Middendorf	[87]
Problem sekwencyjnego porządkowania	Gambardella, Dorigo	[51]
Problem spełniania ograniczeń	Solnon	[110]
Problemy pozyskiwania wiedzy	Parpinelli, Lopes, Freitas	[97]
Problem maksymalnej kliki	Bui, Rizzo Jr	[17]
Problem ścieżek krawędziowo rozłącznych	Blesa, Blum	[8]
Projektowanie układów scalonych	Alupoaei, Katkoori	[1]
Projektowanie sieci komunikacyjnych	Maniezzo, Boschetti, Jelasity	[80]
Bioinformatyka	Shmygelska, Aguirre-Hernández, Hoos	[105]
	Moss, Johnson	[90]
	Karpenko, Shi, Dai	[74]
	Shmygelska, Hoos	[106]
Zastosowania w przemyśle	Bautista, Pereira	[4]
	Silva, Runkler, Sousa, Palm	[107]
	Gottlieb, Puchta, Solnon	[62]
	Corry, Kozan	[28]
Problemy wielokryterialne	Guntsch, Middendorf	[68]
	Lopéz-Ibánez, Paquete, Stützle	[78]
	Doerner, Gutjahr, Hartl, Strauss, Stummer	[38]
Problemy dynamiczne i stochastyczne	Guntsch, Middendorf	[67]
	Bianchi, Gambardella, Dorigo	[7]
Problemy związane z muzyką	Guéret, Monmarché, Slimane	[65]

przeprowadzili eksperymenty na różnych zestawach danych i porównywali wyniki działania algorytmu mrowiskowego z wynikami dostarczonymi przez algorytm etykietowania. Wyniki eksperymentów pokazały, że dla dużych zestawów danych algorytm jest w stanie dostarczyć dobrej jakości zbiory rozwiązań niezdominowanych w czasie znacznie krótszym niż algorytm etykietowania.

W pracy [6] Berrichi i inni zaprezentowali dwukryterialny algorytm mrowiskowy do rozwiązywania problemu optymalizacji harmonogramowania produkcji i utrzymania (ang. *joint production and maintenance scheduling problem*). Celem jest tutaj znalezienie kompromisowego rozwiązania, które w sposób optymalny przydziela zadania produkcyjne oraz planuje prewencyjne przeglądy utrzymaniowe. Wyniki przeprowadzonych eksperymentów pokazały, że zaproponowany algorytm daje lepsze efekty niż dwa znane wielokryterialne algorytmy genetyczne: SPEA 2 i NSGA II.

Przykłady z literatury stanowiły zachętę dla autora niniejszej rozprawy do rozwoju własnych algorytmów mrowiskowych rozwiązujących złożone, wielokryterialne problemy optymalizacyjne.

Rozdział 5

Mrowiskowe algorytmy nawigacji samochodowej

W ostatnich latach ma miejsce lawinowy wzrost popularności i rozwoju systemów nawigacji samochodowej. Jest to spowodowane przede wszystkim coraz większą dostępnością przenośnych urządzeń elektronicznych wyposażonych w moduł GPS, które mogą być zastosowane do tego celu. Do tej grupy zaliczyć można urządzenia dedykowane, takie jak PND (ang. *portable navigation device*) jak również urządzenia bardziej uniwersalne, takie jak PDA (ang. *personal digital assistant*) i nowoczesne telefony komórkowe. Wykorzystywane oprogramowanie łączy w sobie zazwyczaj kilka funkcji, z których najważniejsze to: wyszukiwanie miejsc i adresów, wyznaczanie optymalnej drogi oraz kierowanie użytkownika do celu za pomocą wizualnych i akustycznych wskazówek.

Przedmiotem niniejszej rozprawy jest analiza możliwości wykorzystania algorytmów mrowiskowych w procesie wyszukiwania optymalnej drogi między dwoma wybranymi punktami na mapie, przy czym chodzi o drogę optymalną względem większej liczby kryteriów, a nie tylko drogę najkrótszą.

Prosty algorytm mrowiskowy realizujący to zadanie zaproponowano w pracach [100–102]. Algorytm ten opiera się na podstawowej wersji algorytmu mrowiskowego opisanego np. w pracach [25, 42]. Znajduje on optymalną drogę między dwoma punktami na mapie, przy określeniu przez użytkownika preferencji dotyczących odległości, intensywności ruchu, liczby pasów ruchu, ryzyka kolizji, jakości oraz liczby skrzyżowań. Parametryzacja następuje przez ustalenie wartości współczynników odpowiadających poszczególnym kryteriom optymalności. Podczas obliczeń brany jest także pod uwagę czas rozpoczęcia podróży z uwagi na to, że wagi poszczególnych odcinków drogi mogą mieć różne wartości o różnej porze dnia i nocy. Niestety algorytm AVN zaproponowany w literaturze [100–102] działa tylko dla niewielkich zestawów danych, w szczególności nie działa dla rzeczywistych danych kartograficznych. Problemem okazała się zasada blokowania mrówek w węzłach grafu bez krawędzi wyjściowych po których mrówka mogłaby kontynuować eksplorację przestrzeni przeszukiwania rozwiązań. Duży rozmiar rzeczywistych danych kartograficznych jest przyczyną, że zwykle dochodzi do zablokowania wszystkich mrówek przed znalezieniem jakiegokolwiek rozwiązania. W związku z tym w niniejszej rozprawie zaproponowano jego udoskonalone wersje.

Zadania realizowane przez algorytm AVN można rozwiązać jako bardziej ogólny problem – wielokryterialny problem poszukiwania najkrótszej drogi w grafie (ang. *multi-objective shortest path*), który został sformułowany w rozdziale 3.

5.1 Mrowiskowy algorytm nawigacji samochodowej – AVN

Algorytm AVN (ang. *ant vehicle navigation*), o pseudokodzie oznaczonym jako Algorytm 4, wyszukuje optymalną drogę między dwoma punktami na mapie, która odpowiada preferencjom podanym przez użytkownika. Zbiór parametrów użytkownika składa się ze współczynników określających ważność odległości, liczby pasów, liczby skrzyżowań, natężenia ruchu na drodze, bezpieczeństwa i jakości w proponowanej drodze. Algorytm oparty jest na systemie mrówkowym zaproponowanym przez Dorigo, Maniezzo i Colorniego [25]. Danymi wejściowymi algorytmu, z którymi wywoływana jest procedura AVN, są:

- G – graf $G = (V, E)$ reprezentujący mapę drogową,
- C – macierz kosztów dla wszystkich krawędzi $(i, j) \in E$ oraz wszystkich kryteriów $l \in M$,
- s – węzeł początkowy,
- d – węzeł docelowy,
- N_c – liczba cykli algorytmu,
- N_i – liczba iteracji,
- N_m – liczba mrówek,
- Λ – zbiór parametrów wagowych sterujących ważnością poszczególnych kryteriów $\Lambda = \{\lambda_l, \forall_l \in M\}$.

Dane wyjściowe algorytmu to:

- T^+ – najlepsze znalezione rozwiązanie.

W trakcie działania algorytm wykorzystuje następujące dane pomocnicze:

- ψ^+ – całkowity koszt najlepszego rozwiązania,
- ψ_k – wektor kosztów rozwiązań wyznaczonych przez poszczególne mrówki,
- χ – średni koszt tras wyznaczonych przez mrówki w bieżącym cyklu algorytmu,
- v_k – wektor aktualnych pozycji (węzłów) mrówek,
- a_k – wektor znaczników aktywności mrówki,
- $TABU_k$ – wektor zbiorów $TABU$ dla poszczególnych mrówek,
- i – indeks pomocniczy wierzchołka,
- j – indeks pomocniczy wierzchołka,
- k – indeks mrówki wykorzystywany w iteracjach,

- I – indeks iteracji algorytmu,
- I_c – indeks cyklu algorytmu,
- $\tau_{i,j}$ – tablica śladu feromonowego,
- p^k – wektor prawdopodobieństw wyboru krawędzi dla poszczególnych mrówek,
- N – zbiór mrówek uznanych w bieżącym cyklu algorytmu za najlepsze,
- K – zbiór mrówek uznanych w bieżącym cyklu algorytmu za najgorsze.

Zakłada się, że dane pomocnicze mają charakter zmiennych globalnych. Na początku działania algorytmu (wiersze od 5 do 7) następuje ustalenie wartości parametrów i zmiennych algorytmu oraz inicjacja śladu feromonowego wartością początkową τ_0. W dalszej kolejności powtarzana jest N_c razy główna pętla algorytmu odpowiadająca kolejnym cyklom wykonania. Na początku każdego cyklu wszystkie mrówki są ustawiane w punkcie startowym i inicjowane są zmienne związane ze stanem każdej mrówki (wiersz 9). W trakcie działania algorytmu, każda mrówka może być aktywna lub nieaktywna. Mrówka jest aktywna dopóty, dopóki nie dotrze do punktu docelowego lub nie zostanie zablokowana. Mrówka zostaje zablokowana w momencie, gdy nie ma już możliwej do wybrania drogi (krawędzi) umożliwiającej kontynuowanie podróży. Następnie powtarzane są kolejne iteracje algorytmu odpowiadające kolejnym krokom mrówek (wiersze od 11 do 29). Pętla powtarzana jest do momentu, gdy istnieje co najmniej jedna aktywna mrówka oraz nie przekroczono wartości parametru ograniczającego liczbę iteracji N_i. Podczas każdej iteracji dla każdej aktywnej mrówki sprawdzany jest warunek dotarcia do celu (wiersz 14). Jeżeli warunek jest spełniony, to mrówka przestaje być aktywna, w przeciwnym razie wykonywane są funkcje WyznaczPrawdop oraz WybierzKrawędź mające na celu ustalić kolejny węzeł na drodze mrówki. Jeżeli nie można ustalić kolejnego węzła to oznacza to, że mrówka jest zablokowana i przestaje być aktywna. W przeciwnym razie wykonywany jest ruch bieżącej mrówki (wiersz 22). Wybrany przez mrówkę węzeł jest dodawany do listy $TABU_k$, tak aby nie był już więcej brany pod uwagę przy obliczaniu prawdopodobieństwa.

WyznaczPrawdop – Wyznaczenie prawdopodobieństw wyboru poszczególnych krawędzi wychodzących z bieżącego węzła. Dla wskazanej mrówki obliczane jest prawdopodobieństwo wyboru każdej możliwej krawędzi wychodzącej z aktualnego położenia mrówki. Prawdopodobieństwo wyboru krawędzi z węzła i do węzła j dla mrówki k jest obliczane według wzoru:

$$p_{i,j}^k = \begin{cases} \frac{\tau_{i,j}^{\alpha} \prod_{l \in M} \xi_{i,j,l}^{-2\lambda_l}}{\sum_{h \notin TABU_k} \tau_{i,h}^{\alpha} \prod_{l \in M} \xi_{(i,h,l)}^{-2\lambda_l}}, & \text{jeśli } j \notin TABU_k, \\ 0, & \text{w przeciwnym razie,} \end{cases} \tag{5.1}$$

gdzie:

$\tau_{i,j}$ – wartość śladu feromonowego na krawędzi (i, j),

α – parametr wagowy sterujący ważnością $\tau_{i,j}$,

$TABU_k$ – zbiór węzłów już odwiedzonych przez mrówkę k,

Algorytm 4 Algorytm AVN [100–102]

Dane wejściowe: G, C, s, d, N_c, N_i, N_m, Λ
Dane wyjściowe: rozwiązanie T^+
Dane pomocnicze: ψ^+, ψ_k, χ, v_k, a_k, $TABU_k$, i, j, k, I, I_c, $\tau_{i,j}$, p^k, N, K

function AVN(G, C, s, d, N_c, N_i, N_m, Λ)
    $\tau_{i,j} \leftarrow \tau_0, \forall (i,j) \in E$ ▷ Inicjuj
    $T^+ \leftarrow \emptyset$
    $\psi^+ \leftarrow \infty$
    for $I_c \leftarrow 1$ **to** N_c **do**
        $v_k \leftarrow s, a_k \leftarrow true, TABU_k \leftarrow \emptyset, \forall k \in \{1, \ldots, N_m\}$ ▷ Ustaw mrówki
        $I \leftarrow 1$
        while $I \leqslant N_i \wedge$ istnieje aktywna mrówka **do**
            for $k \leftarrow 1$ **to** N_m **do** ▷ Dla każdej mrówki
                if $a_k = true$ **then** ▷ Jeżeli mrówka jest aktywna
                    if $v_k = d$ **then**
                        $a_k \leftarrow false$ ▷ Mrówka dotarła do celu
                    else
                        $p^k \leftarrow$ WyznaczPrawdop(v_k, $TABU_k$)
                        $j \leftarrow$ WybierzKrawędź(p^k)
                        if $j = 0$ **then**
                            $a_k \leftarrow false$ ▷ Mrówka zablokowana
                        else
                            $v_k \leftarrow j$ ▷ Wykonaj ruch
                            $TABU_k \leftarrow TABU_k \cup \{j\}$ ▷ Aktualizuj listę TABU
                        end if
                    end if
                end if
            end for
            $I \leftarrow I + 1$
        end while
        OceńMrówki ▷ Oceń mrówki oraz aktualizuj T^+
        NagrodźNajlepszeMrówki
        UkarajNajgorszeMrówki
        $\tau_{i,j} \leftarrow \rho \cdot \tau_{i,j}, \forall (i,j) \in E$ ▷ Odparuj feromon
    end for
    return T^+ ▷ Zwróć najlepsze rozwiązanie
end function

function WyznaczPrawdop(i, $TABU_k$)
    for $j \leftarrow 1$ **to** liczba krawędzi wychodzących z i **do**
        $p^k_{i,j} \leftarrow$ wynik obliczeń według wzoru (5.1)
    end for
    return p^k
end function

Algorytm 4 Algorytm AVN (c.d.)

function WybierzKrawędź(p^k)
    $q \leftarrow$ liczba losowa $\in \langle 0, 1 \rangle$
    return krawędź wybrana zgodnie z wzorem (5.7)
end function

procedure OceńMrówki
    for $k \leftarrow 1$ **to** N_m **do** ▷ Dla każdej mrówki
        if $v_k = d$ **then** ▷ Jeżeli mrówka dotarła do celu
            $\psi_k \leftarrow$ całkowity koszt trasy
            if $\psi_k < \psi^+$ **then** ▷ Czy koszt mniejszy od najlepszego?
                $T^+ \leftarrow TABU_k$
                $\psi^+ \leftarrow \psi_k$
            end if
        end if
    end for
    $\chi \leftarrow$ średni koszt trasy
    for $k \leftarrow 1$ **to** N_m **do**
        if $v_k = d \wedge \psi_k < \chi$ **then** ▷ Czy koszt mniejszy od średniej?
            $N \leftarrow N \cup \{k\}$
        else
            $K \leftarrow K \cup \{k\}$
        end if
    end for
end procedure

procedure NagrodźNajlepszeMrówki
    for all $k \in N$ **do**
        $\tau_{i,j} \leftarrow \tau_{i,j} + \frac{\omega_a}{\psi_{i,j}}, \forall (i,j) \in TABU_k$
    end for
end procedure

procedure UkarajNajgorszeMrówki
    for all $k \in K$ **do**
        $\tau_{i,j} \leftarrow \tau_{i,j} \cdot \omega_p, \forall (i,j) \in TABU_k$
    end for
end procedure

M – zbiór kryteriów oceny rozwiązań,

$\xi(i,j,l)$ – wartość funkcji kosztu dla kryterium l i krawędzi (i,j),

λ_l – parametr wagowy należący do zbioru Λ sterujący ważnością kryterium l.

Funkcje kosztu dla poszczególnych kryteriów dla krawędzi i,j to [100]:

$\xi(i,j,O)$ – odległość,

$\xi(i,j,S)$ – liczba pasów ruchu,

$\xi(i,j,N)$ – natężenie ruchu,

$\xi(i,j,R)$ – ryzyko,

$\xi(i,j,J)$ – jakość.

Funkcje kosztu są określone następującymi wzorami:

$$\xi(i,j,O) = d(i,j), \tag{5.2}$$

gdzie $d(i,j)$ jest odległością między węzłami i oraz j,

$$\xi(i,j,S) = \frac{\max_S}{w(i,j)}, \tag{5.3}$$

gdzie $\max_S$ jest największą szerokością drogi, a $w(i,j)$ jest szerokością odcinka drogi między węzłem i a j,

$$\xi(i,j,N) = n(i,j,t), \tag{5.4}$$

gdzie $n(i,j,t)$ jest wartością intensywności ruchu na odcinku drogi (i,j) w czasie t,

$$\xi(i,j,R) = r(i,j,t), \tag{5.5}$$

gdzie $r(i,j,t)$ jest statystyczną liczbą wypadków na odcinku drogi (i,j) w czasie t,

$$\xi(i,j,J) = \frac{\max_J}{q(i,j)}, \tag{5.6}$$

gdzie $\max_J$ jest największą wartością miary jakości drogi, a $q(i,j)$ jest miarą jakości (atrakcyjności) odcinka między węzłem i a j.

WybierzKrawędź – Wybór krawędzi na podstawie obliczonych prawdopodobieństw. Mrówka k na podstawie prawdopodobieństw obliczonych w procedurze WyznaczPrawdop podejmuje decyzję o wyborze krawędzi prowadzącej do kolejnego węzła. W tym celu liczba losowa q z zakresu $\langle 0,1\rangle$ jest porównywana z wartością parametru Q z przedziału $\langle 0,1\rangle$, tak aby wybrać między eksploatacją już uzyskanych rozwiązań, a eksploracją nowych rozwiązań:

$$j = \begin{cases} \arg\max\limits_{h \in J_i^k} \{p_{i,h}^k\}, & \text{jeśli } q \leqslant Q \text{ (eksploatacja)}, \\ S, & \text{w przeciwnym razie (eksploracja)}. \end{cases} \tag{5.7}$$

OceńMrówki – Ocena rozwiązań uzyskanych przez poszczególne mrówki. Dla każdej mrówki, która dotarła do celu obliczany jest całkowity koszt ψ drogi. Dla wyników wszystkich mrówek obliczany jest średni koszt rozwiązania χ. Jeżeli $\psi_k < \chi$, to mrówka jest dodawana do listy nagradzanych mrówek N (wiersz 60), w przeciwnym razie – do listy mrówek karanych K (wiersz 62):

$$k \in \begin{cases} N, & \text{jeśli } \psi_k < \chi, \\ K, & \text{w przeciwnym razie.} \end{cases} \tag{5.8}$$

Dodatkowo, jeżeli w bieżącym cyklu znalezione zostało rozwiązanie o koszcie całkowitym ψ mniejszym od kosztu dotychczas najlepszego rozwiązania T^+, to rozwiązanie to zostaje zapamiętane jako najlepsze (wiersze 51-52). W obliczaniu całkowitego kosztu rozwiązania brany jest pod uwagę dodatkowy składnik równy kosztowi związanemu z liczbą skrzyżowań, a więc węzłów na drodze mrówki:

$$\xi_c^k = rozmiar(TABU_k). \tag{5.9}$$

Z opisu algorytmu AVN nie wynika bezpośrednio, że w obliczeniach kosztu całkowitego uwzględniane są parametry preferencji użytkownika λ_c.

NagrodźNajlepszeMrówki i UkarajNajgorszeMrówki – Nagrodzenie najlepszych rozwiązań (wiersze 67-68) oraz ukaranie rozwiązań o jakości poniżej przeciętnej (wiersze 72-73). Na końcu każdego cyklu każda mrówka k aktualizuje ślad feromonowy na poszczególnych krawędziach (i, j) według zasady:

$$\tau_{i,j}(t) = \begin{cases} \tau_{i,j}(t-1) + \frac{\omega_a}{\psi_{i,j}}, & \text{jeśli } k \in N, \\ \tau_{i,j}(t-1) \cdot \omega_p, & \text{jeśli } k \in K, \end{cases} \tag{5.10}$$

gdzie $\omega_a > 1$ jest współczynnikiem nagrody, a $\omega_p < 1$ jest współczynnikiem kary.

Ostatni krok algorytmu w bieżącym cyklu (wiersz 33), to globalne wyparowanie śladu feromonowego według wzoru:

$$\tau_{i,j}(t) = \rho \cdot \tau_{i,j}(t-1), \tag{5.11}$$

gdzie ρ to współczynnik wyparowania $(0 < \rho < 1)$.

Na końcu, po wykonaniu N_c cykli algorytmu, dotychczas najlepsze rozwiązanie T^+ jest zwracane jako końcowy rezultat.

5.2 Udoskonalona wersja mrowiskowego algorytmu nawigacji samochodowej – NAVN

W trakcie wstępnych badań eksperymentalnych okazało się, że algorytm AVN nie jest w stanie uzyskać rozwiązania dla większych zestawów danych wejściowych. Jest to spowodowane zastosowaniem w algorytmie reguły blokowania mrówek, które znalazły się w miejscu, z którego nie ma wyjścia. W efekcie stosowania tej reguły, bardzo szybko wszystkie mrówki stają się nieaktywne co powoduje przerwanie bieżącego cyklu pracy algorytmu. W rezultacie algorytm nie jest w stanie uzyskać żadnego rozwiązania. Kolejną wadą algorytmu AVN jest sposób określania współczynników λ_l preferencji użytkownika.

Współczynniki te stanowią jednocześnie preferencje użytkownika i spełniają funkcję normalizacyjną. W rezultacie określenie tych wartości nie jest łatwe, a co ważniejsze – nie jest wygodne dla użytkownika.

Aby poprawić jakość i szybkość działania algorytmu AVN zaproponowany został nowy algorytm: NAVN (ang. *new ant vehicle navigation*) [13, 15]. W algorytmie NAVN usuwanie zablokowanych mrówek zostało zastąpione przez powroty ze „ślepych zaułków". Ślad feromonowy jest aktualizowany lokalnie – kiedy mrówka wędruje, oraz globalnie – po każdym cyklu, na drodze najlepszego rozwiązania. Dodatkowo, mrówka w trakcie cofania się znacznie redukuje ślad feromonowy na krawędzi, która prowadziła do węzła bez wyjścia. W ten sposób redukowane jest prawdopodobieństwo wybrania tej krawędzi przez inne mrówki. Te dwie zmiany miały decydujący wpływ na zdolność wyznaczania rozwiązań na rzeczywistych mapach przez nowy algorytm.

Kolejnym ulepszonym elementem algorytmu AVN jest sposób ustalania i uwzględniania współczynników preferencji użytkownika. W algorytmie AVN brakuje elementów zapewniających wzajemną porównywalność poszczególnych parametrów preferencji. W algorytmie NAVN wprowadzone zostały współczynniki normalizacyjne, które obliczone na początku działania algorytmu są następnie używane w obliczeniach funkcji kosztu i prawdopodobieństwa wyboru kolejnej krawędzi. Ponieważ współczynniki normalizacyjne zależą tylko od danych na mapie, ich wartości mogą być także wyliczone przed wykonaniem algorytmu, tj. na etapie przygotowywania mapy.

Pseudokod mrowiskowego algorytmu nawigacji samochodowej NAVN został przedstawiony jako Algorytm 5. Poniżej opisano jego poszczególne elementy. Danymi wejściowymi algorytmu są:

- G – graf $G = (V, E)$ reprezentujący mapę drogową,
- C – macierz kosztów dla wszystkich krawędzi $(i, j) \in E$ oraz wszystkich kryteriów $l \in M$,
- s – węzeł początkowy,
- d – węzeł docelowy,
- N_c – liczba cykli algorytmu,
- N_i – liczba iteracji,
- N_m – liczba mrówek,
- Λ – zbiór parametrów wagowych sterujących ważnością poszczególnych kryteriów $\Lambda = \{\lambda_l, \forall_l \in M\}$.

Dane wyjściowe algorytmu to:

- T^+ – najlepsze znalezione rozwiązanie.

W trakcie działania algorytm wykorzystuje następujące dane pomocnicze:

- ψ^+ – całkowity koszt najlepszego rozwiązania,
- ψ_k – wektor kosztów rozwiązań wyznaczonych przez poszczególne mrówki,

- θ^+ – liczba krawędzi należących do najlepszego rozwiązania,
- η_l – wektor współczynników normalizacyjnych,
- v_k – wektor aktualnych pozycji (węzłów) mrówek,
- a_k – wektor znaczników aktywności mrówki,
- $TABU_k$ – wektor zbiorów $TABU$ dla poszczególnych mrówek,
- $blindEdges_k$ – wektor zbiorów ślepych krawędzi dla każdej mrówki,
- i – indeks pomocniczy wierzchołka,
- j – indeks pomocniczy wierzchołka,
- k – indeks mrówki wykorzystywany w iteracjach,
- I – indeks iteracji algorytmu,
- I_c – indeks cyklu algorytmu,
- $\tau_{i,j}$ – tablica śladu feromonowego,
- p^k – wektor prawdopodobieństw wyboru krawędzi dla poszczególnych mrówek,
- Q – parametr mający wpływ na wybór między eksploatacją, a eksploracją rozwiązań.

PrzygotójNormalizację – Przygotowanie współczynników normalizacyjnych. W tym kroku dla wszystkich składników funkcji kosztu (odległość, liczba pasów, natężenie ruchu, ryzyko, jakość oraz liczba skrzyżowań) obliczany jest współczynnik normalizacji η_l. Dzięki temu parametry preferencji użytkownika λ_l mogą przyjmować wartości z przedziału $[0, 1]$ i nie muszą być dostosowywane do konkretnych danych mapy.

Na potrzeby obliczeń wyznaczane są maksymalne i minimalne wartości wag dla wszystkich krawędzi grafu:

$\max_O$ – maksymalna długość odcinka,

$\max_S$ – maksymalna szerokość (liczba pasów ruchu),

$\min_S$ – minimalna szerokość,

$\max_N$ – maksymalna wartość natężenia ruchu,

$\max_R$ – maksymalna wartość ryzyka rozumiana jako poziom zagrożenia wypadkiem,

$\max_J$ – maksymalna wartość jakości,

$\min_J$ – minimalna wartość jakości,

$\max_M = \max(\max_O, \max_S, \max_N, \max_R, \max_J)$.

Algorytm 5 Algorytm NAVN

Dane wejściowe: G, C, s, d, N_c, N_i, N_m, Λ
Dane wyjściowe: rozwiązanie T^+
Dane pomocnicze: ψ^+, ψ_k, θ^+, η_l, v_k, a_k, $TABU_k$, $blindEdges_k$, i, j, k, I, I_c, $\tau_{i,j}$, p^k, Q

function NAVN(G, C, s, d, N_c, N_i, N_m, Λ)
    PrzygotujNormalizację(G, C, η)
    $\tau_{i,j} \leftarrow \tau_0, \forall (i,j) \in E$ ▷ Inicjuj
    $T^+ \leftarrow \emptyset$
    $\psi^+ \leftarrow \infty$
    for $I_{cykl} \leftarrow 1$ **to** N_c **do**
        $v_k \leftarrow s, a_k \leftarrow true, TABU_k \leftarrow \emptyset, \forall k \in \{1, \ldots, N_m\}$ ▷ Ustaw mrówki
        $I \leftarrow 1$
        while $I \leqslant N_i \wedge$ istnieje aktywna mrówka **do**
            for $k \leftarrow 1$ **to** N_m **do** ▷ Dla każdej mrówki
                if $a_k = true$ **then** ▷ Jeżeli mrówka jest aktywna
                    if $v_k = d$ **then**
                        $a_k \leftarrow false$ ▷ Mrówka dotarła do celu
                    else
                        $p^k \leftarrow$ WyznaczPrawdop(v_k, $TABU_k$)
                        $j \leftarrow$ WybierzKrawędź(p^k)
                        if $j = 0$ **then**
                            CofnijMrówkę(k, i, j) ▷ Mrówka zablokowana
                        else
                            $v_k \leftarrow j$ ▷ Wykonaj ruch
                            $TABU_k \leftarrow TABU_k \cup \{j\}$ ▷ Aktualizuj listę TABU
                        end if
                    end if
                end if
            end for
            $I \leftarrow I + 1$
        end while
        OceńMrówki ▷ Oceń mrówki oraz aktualizuj T^+
        NagrodźNajlepszeRozwiązanie
        UkarajNajgorszeMrówki
        $Q \leftarrow Q \cdot \varphi$ ▷ Zmodyfikuj Q
    end for
    return T^+ ▷ Zwróć najlepsze rozwiązanie
end function

procedure PrzygotujNormalizację(**in** G, **in** C, **out** η)
    for all $l \in M$ **do**
        $\eta_l \leftarrow$ wylicz według wzorów (5.12 - 5.17)
    end for
end procedure

Algorytm 5 Algorytm NAVN (c.d)

```
43: function WyznaczPrawdop(i, TABU_k)
44:     for j ← 1 to liczba krawędzi wychodzących z i do
45:         p^k_{i,j} ← wynik obliczeń według wzoru (5.18)
46:     end for
47:     return p^k
48: end function

49: function WybierzKrawędź(p^k)
50:     q ← liczba losowa ∈ ⟨0, 1⟩
51:     j ← krawędź wybrana zgodnie z wzorem (5.7)
52:     τ_{i,j} ← (1 − ρ) · τ_{i,j} + ρ · τ_0          ▷ Wzmocnij feromon na krawędzi (i, j)
53:     return j
54: end function

55: procedure CofnijMrówkę(in k, in i, in j)
56:     TABU_k ← TABU_k \ {i}                          ▷ Usuń węzeł i z TABU_k
57:     blindEdges_k ← blindEdges_k ∪ {i, j}           ▷ Dodaj krawędź (i, j) do blindEdges_k
58:     τ_{i,j} ← τ_{i,j} · ω_b                        ▷ Redukuj feromon na krawędzi (i, j)
59: end procedure

60: procedure OceńMrówki
61:     for k ← 1 to N_m do                            ▷ Dla każdej mrówki
62:         if v_k = d then                            ▷ Jeżeli mrówka dotarła do celu
63:             ψ_k ← całkowity koszt trasy
64:             if ψ_k < ψ^+ then                      ▷ Czy koszt mniejszy od najlepszego?
65:                 T^+ ← TABU_k
66:                 ψ^+ ← ψ_k
67:             end if
68:         end if
69:     end for
70: end procedure

71: procedure NagrodźNajlepszeRozwiązanie
72:     τ_{i,j} ← (1 − ρ) · τ_{i,j} + 2 · ρ · θ^+/ψ^+, ∀(i, j) ∈ T^+   ▷ Wzmocnij feromon
73: end procedure

74: procedure UkarajNajgorszeMrówki
75:     for k ← 1 to N_m do                            ▷ Dla każdej mrówki
76:         if v_k ≠ d then                            ▷ Jeżeli mrówka nie dotarła do celu
77:             τ_{i,j} ← τ_{i,j} · ω_p, ∀(i, j) ∈ TABU_k
78:         end if
79:     end for
80: end procedure
```

Poszczególne współczynniki normalizacyjne obliczane są następująco:

$$\eta_O = \frac{\max_M}{\max_O} \tag{5.12}$$

$$\eta_S = \frac{\max_M}{\max_S / \min_S} \tag{5.13}$$

$$\eta_N = \frac{\max_M}{\max_N} \tag{5.14}$$

$$\eta_R = \frac{\max_M}{\max_R} \tag{5.15}$$

$$\eta_J = \frac{\max_M}{\max_J / \min_J} \tag{5.16}$$

$$\eta_C = \max_M \tag{5.17}$$

WyznaczPrawdop – Wyznaczenie prawdopodobieństw wyboru poszczególnych krawędzi wychodzących z bieżącego węzła. Dla wskazanej mrówki obliczane jest prawdopodobieństwo wyboru każdej możliwej krawędzi wychodzącej z aktualnego położenia mrówki. Obliczanie składników tablicy prawdopodobieństw jest bardziej podobne do klasycznego algorytmu mrowiskowego, w stosunku do analogicznej procedury w algorytmie AVN. Tak samo jak w algorytmie AVN wykorzystywane jest kryterium zastępcze w formie zaagregowanej informacji heurystycznej, ale w procesie obliczania prawdopodobieństwa uwzględniane są współczynniki normalizacyjne η. Obliczenia w tym kroku wyrażone są wzorem:

$$p_{i,j}^k = \begin{cases} \frac{\tau_{i,j}^{\alpha} \cdot (\frac{1}{\psi_{i,j}})^{\beta}}{\sum_{h \notin TABU_k} \tau_{i,h}^{\alpha} \cdot (\frac{1}{\psi_{i,h}})^{\beta}}, & j \notin TABU_k \wedge (i,j) \notin blindEdges_k, \\ 0, & \text{w przeciwnym razie,} \end{cases} \tag{5.18}$$

gdzie:

$\tau_{i,j}$ – wartość śladu feromonowego na krawędzi (i, j),

α – współczynnik wagowy śladu feromonowego $\tau_{i,j}$,

β – współczynnik wagowy informacji heurystycznej,

$\psi_{i,j}$ – zagregowany koszt krawędzi (i, j) obliczany według wzoru:

$$\psi_{i,j} = \sum_{l \in M} \xi(i,j,l) \cdot \lambda_l, \tag{5.19}$$

gdzie:

$\xi(i, j, l)$ – wartość funkcji kosztu dla kryterium l dla krawędzi (i, j),

λ_l – współczynnik wagowy ważności kryterium l określony przez użytkownika w przedziale $0 < \lambda_l < 1$.

WybierzKrawędź – Wybór krawędzi na podstawie obliczonego prawdopodobieństwa. Wybór krawędzi wykonywany jest w taki sam sposób, jak w algorytmie AVN, ale po wybraniu krawędzi mrówka pozostawia ślad feromonowy, aktualizując go według wzoru:

$$\tau_{i,j}(t) = (1-\rho)\cdot\tau_{i,j}(t-1) + \rho\cdot\tau_0, \tag{5.20}$$

gdzie:

ρ – współczynnik wyparowania śladu feromonowego z przedziału $[0,1]$,

τ_0 – wartość śladu początkowego.

CofnijMrówkę – Wycofanie się mrówki z miejsca bez wyjścia. Jeżeli mrówka jest zablokowana, a więc z węzła, w którym aktualnie się znajduje nie ma wyjścia, to cofa się o jeden krok. Krawędź, po której się cofa, dodawana jest do listy $blindEdges_k$, zawierającej ślepe krawędzie, których mrówka już nie wybierze w bieżącym cyklu działania algorytmu. Jednocześnie ślad feromonowy na krawędzi jest aktualizowany według wzoru:

$$\tau_{i,j}(t) = \tau_{i,j}(t-1)\cdot\omega_b, \tag{5.21}$$

gdzie ω_b to współczynnik zmniejszenia feromonu na ślepej krawędzi.

OceńMrówki – Ocena rozwiązań uzyskanych przez poszczególne mrówki. Rozwiązania w postaci dróg znalezione przez mrówki są oceniane. Jeżeli rozwiązanie mrówki ma mniejszy koszt zagregowany od dotychczas najlepszego znalezionego, to jest ono zapisywane jako najlepsze. Podobnie jak w algorytmie AVN, w koszcie całkowitym ujęty zostaje koszt wynikający z liczby skrzyżowań, liczony według wzoru:

$$\xi_c^k = rozmiar(TABU_k)\cdot\eta_c \tag{5.22}$$

NagrodźNajlepszeRozwiązanie – Nagrodzenie najlepszego rozwiązania. Dotychczas najlepsze rozwiązanie jest nagradzane na podstawie globalnej reguły aktualizacji śladu feromonowego wyrażonej wzorem:

$$\tau_{i,j}(t) = (1-\rho)\cdot\tau_{i,j}(t-1) + 2\cdot\rho\cdot\frac{\theta^+}{\psi^+}, \tag{5.23}$$

gdzie:

ρ – współczynnik wyparowania śladu feromonowego, ten sam, który jest używany w lokalnej regule wyparowania śladu feromonowego w kroku WybierzKrawędź,

θ^+ – liczba krawędzi należących do drogi najlepszego rozwiązania,

ψ^+ – koszt najlepszego rozwiązania.

UkarajNajgorszeMrówki – Ukaranie mrówek, którym nie udało się zbudować rozwiązania. W tym kroku karane są mrówki, którym nie udało się znaleźć rozwiązania w zadanej liczbie iteracji algorytmu. Dla wszystkich mrówek, które nie dotarły do punktu docelowego, aktualizowany jest ślad feromonowy na krawędziach należących do ich dróg:

$$\tau_{i,j}(t) = \tau_{i,j}(t-1)\cdot\omega_p, \tag{5.24}$$

gdzie $0 < \omega_p < 1$ jest współczynnikiem kary.

Ostatni krok algorytmu w bieżącym cyklu (wiersz 34), to zmodyfikowanie wartości parametru Q. Wartość parametru Q zostaje zmniejszona, wykorzystując współczynnik zmiany φ. Nowa wartość parametru jest obliczana według wzoru:

$$Q(t) = Q(t-1) \cdot \varphi, \tag{5.25}$$

gdzie $0 < \varphi < 1$ jest współczynnikiem zmiany wartości parametru Q. W ten sposób stopniowo, w kolejnych cyklach pracy algorytmu jest zmniejszana eksploracja na rzecz eksploatacji dotychczasowych rozwiązań.

Wyniki eksperymentów pokazały, że algorytm NAVN znajduje poprawne rozwiązanie dla dużej, rzeczywistej mapy, dla której algorytm AVN nie był w stanie znaleźć żadnego rozwiązania. Należy dodać, że rozwiązania znajdowane przez algorytm NAVN, niejednokrotnie zostały uznane przez kierowców (użytkowników) znających drogi, jako najlepsze.

5.3 Aproksymacyjny algorytm MultiNAVN

Dotychczas zaprezentowane algorytmy AVN i NAVN wyznaczały jedno rozwiązanie na podstawie preferencji użytkownika określanych przed rozpoczęciem działania algorytmów. Złożony problem optymalizacji wielokryterialnej był więc już na wstępie sprowadzany do problemu optymalizacji jednokryterialnej przez wprowadzenie kryterium zastępczego.

Metody sprowadzania optymalizacji wielokryterialnej do problemu z jednym kryterium są często zbyt uproszczone i niezbyt nadają się do rozwiązywania rzeczywistych problemów. Ponadto pojawia się trudność z doborem wartości wag dla poszczególnych kryteriów oraz to, że algorytmy zwykle zwracają tylko jedno rozwiązanie, a nie zbiór rozwiązań niezdominowanych (paretooptymalnych).

Ogólny algorytm mrowiskowy poszukujący rozwiązań niezdominowanych, który został przedstawiony w podrozdziale 4.6 może zostać zastosowany do rozwiązywania wielu wielokryterialnych problemów kombinatorycznych. W dalszym ciągu zaproponujemy modyfikację algorytmu NAVN, tak aby poszukiwał przybliżonego zbioru rozwiązań, który jest jak najbardziej zbliżony do pełnego zbioru rozwiązań paretooptymalnych. Jako miara podobieństwa zbioru przybliżonego do pełnego zbioru referencyjnego przyjęta została odległość pomiędzy dwoma zbiorami (metryka Hausdorffa), która została opisana w rozdziale 2.2.

Zaproponowany algorytm zamiast jednego rozwiązania zwraca zbiór rozwiązań. W stosunku do wersji poprzedniej wprowadzono elementy związane z wyznaczaniem zbioru rozwiązań paretooptymalnych. Możliwe są różne warianty realizacji funkcji WyznaczPrawdop w zależności od konstrukcji tablicy feromonowej. Funkcja NagrodźNajlepszeRozwiązania może brać pod uwagę rozwiązania ze zbioru Pareto lub najlepsze w danej iteracji algorytmu. Na potrzeby eksperymentów obliczeniowych z algorytmem MultiNAVN wyróżnione zostały dwie wersje algorytmu, które opiszemy w kolejnych podrozdziałach.

5.3.1 Algorytm MultiNAVN-Z z kryterium zastępczym

Wersja algorytmu MultiNAVN, nazwana MultiNAVN-Z, wykorzystuje zaagregowaną informację heurystyczną, według której liczone jest prawdopodobieństwo wyboru

krawędzi. Procedura wyznaczająca prawdopodobieństwa wyboru krawędzi wychodzących z bieżącego węzła WyznaczPrawdop ma taką samą postać, jak w algorytmie NAVN. Należy jednak zwrócić uwagę na fakt, że zbiór współczynników Λ nie jest już traktowany jako jeden z parametrów algorytmu. Wartości poszczególnych współczynników są ustalone wewnętrznie w sposób mający zapewnić równowagę między poszczególnymi kryteriami w trakcie obliczania kryterium zastępczego.

Celem algorytmu MultiNAVN jest wyznaczanie zbioru rozwiązań paretooptymalnych. Dopiero w kolejnej fazie procesu optymalizacji wielokryterialnej wybierane jest rozwiązanie kompromisowe. W związku z tym dla algorytmu nie są istotne współczynniki określające preferencje użytkownika. Kluczową zmianą jest wprowadzenie elementów związanych z wyznaczaniem zbioru rozwiązań paretooptymalnych, które przedstawiono poniżej. Pseudokod algorytmu MultiNAVN-Z został zaprezentowany jako Algorytm 6. Danymi wejściowymi algorytmu są:

- G – graf $G = (V, E)$ reprezentujący mapę drogową,
- C – macierz kosztów dla wszystkich krawędzi $(i, j) \in E$ oraz wszystkich kryteriów $l \in M$,
- s – węzeł początkowy,
- d – węzeł docelowy,
- N_c – liczba cykli algorytmu,
- N_i – liczba iteracji,
- N_m – liczba mrówek.

Dane wyjściowe algorytmu to:

- $\mathbb{P}$ – przybliżony zbiór rozwiązań paretooptymalnych.

W trakcie działania algorytm wykorzystuje następujące dane pomocnicze:

- ψ^+ – całkowity koszt (kryterium zastępczego) najlepszego rozwiązania,
- ψ_k – wektor kosztów rozwiązań wyznaczonych przez poszczególne mrówki,
- T^+ – najlepsze znalezione rozwiązanie (o najniższej wartości kryterium zastępczego),
- T – rozwiązanie pomocnicze wykorzystywane w iteracjach,
- T_k – rozwiązanie wyznaczone przez mrówkę w bieżącym cyklu algorytmu,
- θ^+ – liczba krawędzi należących do najlepszego rozwiązania,
- v_k – wektor aktualnych pozycji (węzłów) mrówek,
- a_k – wektor znaczników aktywności mrówki,
- $TABU_k$ – wektor zbiorów $TABU$ dla poszczególnych mrówek,

- $blindEdges_k$ – wektor zbiorów ślepych krawędzi dla każdej mrówki,
- i – indeks pomocniczy wierzchołka,
- j – indeks pomocniczy wierzchołka,
- k – indeks mrówki wykorzystywany w iteracjach,
- I – indeks iteracji algorytmu,
- I_c – indeks cyklu algorytmu,
- $\tau_{i,j}$ – tablica śladu feromonowego,
- p^k – wektor prawdopodobieństw wyboru krawędzi dla poszczególnych mrówek,
- Q – parametr mający wpływ na wybór między eksploatacją, a eksploracją rozwiązań.

AktualizujZbiórPareto – Procedura aktualizuje zbiór Pareto przez porównanie rozwiązania utworzonego przez mrówkę z rozwiązaniami dotychczas znalezionymi i uznanymi za paretooptymalne. Jeżeli nowe rozwiązanie nie jest zdominowane przez żadne z rozwiązań dotychczasowych, to dodawane jest ono do listy rozwiązań niezdominowanych. Rozwiązania zdominowane przez nowe rozwiązanie są z tej listy usuwane.

Na końcu algorytmu (wiersz 38) zwracany jest zbiór rozwiązań dotychczas uznanych za paretooptymalne.

5.3.2 Algorytm MultiNAVN-L z kryterium wybieranym losowo

W przypadku wersji algorytmu MultiNAVN, nazwanej MultiNAVN-L, w każdym kroku mrówka wybiera w sposób losowy kryterium, według którego liczone jest prawdopodobieństwo wyboru krawędzi. Informacja o śladzie feromonowym pozostaje wspólna dla wszystkich kryteriów. Pseudokod algorytmu MultiNAVN-Z został zaprezentowany jako Algorytm 7. Danymi wejściowymi algorytmu są:

- G – graf $G = (V, E)$ reprezentujący mapę drogową,
- C – macierz kosztów dla wszystkich krawędzi $(i, j) \in E$ oraz wszystkich kryteriów $l \in M$,
- s – węzeł początkowy,
- d – węzeł docelowy,
- N_c – liczba cykli algorytmu,
- N_i – liczba iteracji,
- N_m – liczba mrówek.

Dane wyjściowe algorytmu to:

- $\mathbb{P}$ – przybliżony zbiór rozwiązań paretooptymalnych.

Algorytm 6 Algorytm MultiNAVN-Z

Dane wejściowe: G, C, s, d, N_c, N_i, N_m
Dane wyjściowe: przybliżony zbiór rozwiązań paretooptymalnych $\mathbb{P}$
Dane pomocnicze: ψ^+, ψ_k, T^+, T, T_k, θ^+, v_k, a_k, $TABU_k$, $blindEdges_k$, i, j, k, I, I_c, $\tau_{i,j}$, p^k, Q

function MultiNAVN-Z(G, C, s, d, N_c, N_i, N_m)
    PrzygotujNormalizację(G, C, η)
    $\tau_{i,j} \leftarrow \tau_0, \forall (i,j) \in E$ ▷ Inicjuj
    $T^+ \leftarrow \emptyset$
    $\mathbb{P} \leftarrow \emptyset$ ▷ Inicjuj zbiór Pareto
    $\psi^+ \leftarrow \infty$
    for $I_{cykl} \leftarrow 1$ **to** N_c **do**
        $v_k \leftarrow s, a_k \leftarrow true, TABU_k \leftarrow \emptyset, \forall k \in \{1, \ldots, N_m\}$ ▷ Ustaw mrówki
        $I \leftarrow 1$
        while $I \leqslant N_i \wedge$ istnieje aktywna mrówka **do**
            for $k \leftarrow 1$ **to** N_m **do** ▷ Dla każdej mrówki
                if $a_k = true$ **then** ▷ Jeżeli mrówka jest aktywna
                    if $v_k = d$ **then**
                        $a_k \leftarrow false$ ▷ Mrówka dotarła do celu
                    else
                        $p^k \leftarrow$ WyznaczPrawdop(v_k, $TABU_k$)
                        $j \leftarrow$ WybierzKrawędź(p^k)
                        if $j = 0$ **then**
                            CofnijMrówkę(k, i, j) ▷ Mrówka zablokowana
                        else
                            $v_k \leftarrow j$ ▷ Wykonaj ruch
                            $TABU_k \leftarrow TABU_k \cup \{j\}$ ▷ Aktualizuj listę TABU
                        end if
                    end if
                end if
            end for
            $I \leftarrow I + 1$
        end while
        OceńMrówki ▷ Oceń mrówki oraz aktualizuj T^+
        AktualizujZbiórPareto
        NagrodźNajlepszeRozwiązania
        UkarajNajgorszeMrówki
        $Q \leftarrow Q \cdot \varphi$ ▷ Zmodyfikuj Q
    end for
    return $\mathbb{P}$ ▷ Zwróć przybliżony zbiór Pareto
end function

procedure PrzygotujNormalizację(**in** G, **in** C, **out** η)
    for all $l \in M$ **do**
        $\eta_l \leftarrow$ wylicz według wzorów (5.12 - 5.17)
    end for
end procedure

Algorytm 6 Algorytm MultiNAVN-Z (c.d.)

```
45: function WyznaczPrawdop(i, TABU_k)
46:     for j ← 1 to liczba krawędzi wychodzących z i do
47:         p^k_{i,j} ← wynik obliczeń według wzoru (5.18)
48:     end for
49:     return p^k
50: end function

51: function WybierzKrawędź(p^k)
52:     q ← liczba losowa ∈ ⟨0, 1⟩
53:     j ← krawędź wybrana zgodnie z wzorem (5.7)
54:     τ_{i,j} ← (1 − ρ) · τ_{i,j} + ρ · τ_0          ▷ Wzmocnij feromon na krawędzi (i, j)
55:     return j
56: end function

57: procedure CofnijMrówkę(in k, in i, in j)
58:     TABU_k ← TABU_k \ {i}                          ▷ Usuń węzeł i z TABU_k
59:     blindEdges_k ← blindEdges_k ∪ {i, j}           ▷ Dodaj krawędź (i, j) do blindEdges_k
60:     τ_{i,j} ← τ_{i,j} · ω_b                        ▷ Redukuj feromon na krawędzi (i, j)
61: end procedure

62: procedure OceńMrówki
63:     for k ← 1 to N_m do                            ▷ Dla każdej mrówki
64:         if v_k = d then                            ▷ Jeżeli mrówka dotarła do celu
65:             ψ_k ← wartość kryterium zastępczego dla trasy
66:             if ψ_k < ψ^+ then                      ▷ Czy koszt mniejszy od najlepszego?
67:                 T^+ ← TABU_k
68:                 ψ^+ ← ψ_k
69:             end if
70:         end if
71:     end for
72: end procedure

73: procedure AktualizujZbiórPareto
74:     for k ← 1 to N_m do                            ▷ Dla każdej mrówki
75:         if v_k = d then                            ▷ Jeżeli mrówka dotarła do celu
76:             // Czy w zbiorze ℙ nie istnieją rozwiązania dominujące T_k?
77:             if ∄T ∈ ℙ|T ⪯ T_k then
78:                 ℙ ← ℙ ∪ {T_k}                      ▷ Dodaj rozwiązanie do zbioru ℙ
79:                 ℙ ← ℙ \ {T ∈ ℙ|T_k ⪯ T}            ▷ Usuń rozwiązania zdominowane
80:             end if
81:         end if
82:     end for
83: end procedure
```

Algorytm 6 Algorytm MultiNAVN-Z (c.d.)

84: **procedure** NagrodźNajlepszeRozwiązania
85: $\tau_{i,j} \leftarrow (1-\rho) \cdot \tau_{i,j} + 2 \cdot \rho \cdot \frac{\theta^+}{\psi^+}, \forall (i,j) \in T^+$ ▷ Wzmocnij feromon
86: **end procedure**

87: **procedure** UkarajNajgorszeMrówki
88: **for** $k \leftarrow 1$ **to** N_m **do** ▷ Dla każdej mrówki
89: **if** $v_k \neq d$ **then** ▷ Jeżeli mrówka nie dotarła do celu
90: $\tau_{i,j} \leftarrow \tau_{i,j} \cdot \omega_p, \forall (i,j) \in TABU_k$
91: **end if**
92: **end for**
93: **end procedure**

W trakcie działania algorytm wykorzystuje następujące dane pomocnicze:

- ψ_T – całkowity koszt rozwiązania pomocniczego,
- T – rozwiązanie pomocnicze wykorzystywane w iteracjach,
- T_k – rozwiązanie wyznaczone przez mrówkę w bieżącym cyklu algorytmu,
- θ_T – liczba krawędzi należących do rozwiązania pomocniczego,
- v_k – wektor aktualnych pozycji (węzłów) mrówek,
- a_k – wektor znaczników aktywności mrówki,
- $TABU_k$ – wektor zbiorów $TABU$ dla poszczególnych mrówek,
- $blindEdges_k$ – wektor zbiorów ślepych krawędzi dla każdej mrówki,
- i – indeks pomocniczy wierzchołka,
- j – indeks pomocniczy wierzchołka,
- k – indeks mrówki wykorzystywany w iteracjach,
- I – indeks iteracji algorytmu,
- I_c – indeks cyklu algorytmu,
- $\tau_{i,j}$ – tablica śladu feromonowego,
- p^k – wektor prawdopodobieństw wyboru krawędzi dla poszczególnych mrówek,
- Q – parametr mający wpływ na wybór między eksploatacją, a eksploracją rozwiązań.

W stosunku do algorytmu MultiNAVN-Z zmiany dotyczą definicji procedury WyznaczPrawdop, która w zmienionej formie została opisana poniżej.

WyznaczPrawdop - Wyznaczenie prawdopodobieństwa wyboru krawędzi wychodzących z bieżącego węzła. W przypadku tej wersji algorytmu nie jest używana zaagregowana wartość kosztu dla poszczególnych krawędzi, lecz w zamian podczas każdego kroku

Algorytm 7 Algorytm MultiNAVN-L

Dane wejściowe: G, C, s, d, N_c, N_i, N_m
Dane wyjściowe: przybliżony zbiór rozwiązań paretooptymalnych $\mathbb{P}$
Dane pomocnicze: ψ_T, T, T_k, θ_T, v_k, a_k, $TABU_k$, $blindEdges_k$, i, j, k, I, I_c, $\tau_{i,j}$, p^k, Q

function MultiNAVN-L(G, C, s, d, N_c, N_i, N_m)
    $\tau_{i,j} \leftarrow \tau_0, \forall (i,j) \in E$ ▷ Inicjuj
    $\mathbb{P} \leftarrow \emptyset$ ▷ Inicjuj zbiór Pareto
    $\psi^+ \leftarrow \infty$
    for $I_{cykl} \leftarrow 1$ **to** N_c **do**
        $v_k \leftarrow s, a_k \leftarrow true, TABU_k \leftarrow \emptyset, \forall k \in \{1, \ldots, N_m\}$ ▷ Ustaw mrówki
        $I \leftarrow 1$
        while $I \leqslant N_i \wedge$ istnieje aktywna mrówka **do**
            for $k \leftarrow 1$ **to** N_m **do** ▷ Dla każdej mrówki
                if $a_k = true$ **then** ▷ Jeżeli mrówka jest aktywna
                    if $v_k = d$ **then**
                        $a_k \leftarrow false$ ▷ Mrówka dotarła do celu
                    else
                        $p^k \leftarrow$ WyznaczPrawdop(v_k, $TABU_k$)
                        $j \leftarrow$ WybierzKrawędź(p^k)
                        if $j = 0$ **then**
                            CofnijMrówkę(k, i, j) ▷ Mrówka zablokowana
                        else
                            $v_k \leftarrow j$ ▷ Wykonaj ruch
                            $TABU_k \leftarrow TABU_k \cup \{j\}$ ▷ Aktualizuj listę TABU
                        end if
                    end if
                end if
            end for
            $I \leftarrow I + 1$
        end while
        AktualizujZbiórPareto
        NagrodźNajlepszeRozwiązania
        UkarajNajgorszeMrówki
        $Q \leftarrow Q \cdot \varphi$ ▷ Zmodyfikuj Q
    end for
    return $\mathbb{P}$ ▷ Zwróć przybliżony zbiór Pareto
end function

function WyznaczPrawdop(i, $TABU_k$)
    $l \leftarrow$ losowy numer kryterium
    for $j \leftarrow 1$ **to** liczba krawędzi wychodzących z i **do**
        $p^k_{i,j} \leftarrow$ wynik obliczeń według wzoru (5.26)
    end for
    return p^k
end function

Algorytm 7 Algorytm MultiNAVN-L (c.d.)

function WybierzKrawędź(p^k)
  $q \leftarrow$ liczba losowa $\in \langle 0, 1\rangle$
  $j \leftarrow$ krawędź wybrana zgodnie z wzorem (5.7)
  $\tau_{i,j} \leftarrow (1-\rho)\cdot\tau_{i,j} + \rho\cdot\tau_0$ ▷ Wzmocnij feromon na krawędzi (i,j)
  return j
end function

procedure CofnijMrówkę(**in** k, **in** i, **in** j)
  $TABU_k \leftarrow TABU_k \setminus \{i\}$ ▷ Usuń węzeł i z $TABU_k$
  $blindEdges_k \leftarrow blindEdges_k \cup \{i,j\}$ ▷ Dodaj krawędź (i,j) do $blindEdges_k$
  $\tau_{i,j} \leftarrow \tau_{i,j}\cdot\omega_b$ ▷ Redukuj feromon na krawędzi (i,j)
end procedure

procedure AktualizujZbiórPareto
  for $k \leftarrow 1$ **to** N_m **do** ▷ Dla każdej mrówki
    if $v_k = d$ **then** ▷ Jeżeli mrówka dotarła do celu
      // Czy w zbiorze $\mathbb{P}$ nie istnieją rozwiązania dominujące T_k?
      if $\nexists T \in \mathbb{P} | T \preceq T_k$ **then**
        $\mathbb{P} \leftarrow \mathbb{P} \cup \{T_k\}$ ▷ Dodaj rozwiązanie do zbioru $\mathbb{P}$
        $\mathbb{P} \leftarrow \mathbb{P} \setminus \{T \in \mathbb{P} | T_k \preceq T\}$ ▷ Usuń rozwiązania zdominowane
      end if
    end if
  end for
end procedure

procedure NagrodźNajlepszeRozwiązania
  for all $T \in \mathbb{P}$ **do** ▷ Dla każdego rozwiązania w zbiorze $\mathbb{P}$
    $\tau_{i,j} \leftarrow (1-\rho)\cdot\tau_{i,j} + 2\cdot\rho\cdot\frac{\theta_T}{\psi_T}, \forall(i,j) \in T$ ▷ Wzmocnij feromon
  end for
end procedure

procedure UkarajNajgorszeMrówki
  for $k \leftarrow 1$ **to** N_m **do** ▷ Dla każdej mrówki
    if $v_k \neq d$ **then** ▷ Jeżeli mrówka nie dotarła do celu
      $\tau_{i,j} \leftarrow \tau_{i,j}\cdot\omega_p, \forall(i,j) \in TABU_k$
    end if
  end for
end procedure

mrówka w sposób losowy wybiera kryterium l oraz koszt związany z tym kryterium, na podstawie którego wyznacza prawdopodobieństwo wyboru krawędzi:

$$p_{i,j}^{k} = \begin{cases} \frac{\tau_{i,j}^{\alpha} \cdot (\frac{1}{\xi(i,j,l)})^{\beta}}{\sum_{h \notin TABU_k} \tau_{i,h}^{\alpha} \cdot (\frac{1}{\xi(i,h,l)})^{\beta}}, & j \in TABU_k, \\ 0, & \text{w przeciwnym razie,} \end{cases} \tag{5.26}$$

gdzie:

l – kryterium wybrane w sposób losowy przed rozpoczęciem obliczeń prawdopodobieństwa,

$\tau_{i,j}$ – wartość śladu feromonowego na krawędzi (i, j),

α – współczynnik wagowy śladu feromonowego $\tau_{i,j}$,

β – współczynnik wagowy informacji heurystycznej,

$\xi(i, j, l)$ – wartość funkcji kosztu dla kryterium l dla krawędzi z (i, j).

Ponieważ algorytm MultiNAVN-L nie wykorzystuje współczynników λ_l sterujących ważnością poszczególnych kryteriów oceny rozwiązań (preferencje użytkownika) pominąć można krok PrzygotójNormalizację, którego zadaniem jest obliczenie wartości współczynników normalizacyjnych. Procedura OceńMrówki występująca w poprzednich wersjach algorytmu NAVN również nie jest wykorzystywana w algorytmie MultiNAVN-L ponieważ nie jest wyznaczane rozwiązanie „najlepsze" oparte o kryterium zastępcze. Pozostałe elementy algorytmu występujące w wersji podstawowej MultiNAVN nie zostały zmodyfikowane i są takie same, jak w wersji MultiNAVN-Z.

5.4 Weryfikacja poprawności algorytmów MultiNAVN-Z oraz MultiNAVN-L

Weryfikacja poprawności algorytmu polega na udowodnieniu, że zachodzą pożądane relacje między jego danymi wejściowymi i wyjściowymi. Poprawność algorytmu sekwencyjnego wyrażana jest za pomocą zdania postaci:

$$\{p\}A\{q\}, \tag{5.27}$$

gdzie A jest algorytmem, p warunkiem wstępnym (ang. *precondition*), a q warunkiem ostatecznym (ang. *postcondition*) [33]. Warunek wstępny określa warunki, jakie powinny spełniać dane wejściowe przed rozpoczęciem działania algorytmu, a warunek ostateczny precyzuje pożądane warunki jakie powinny spełniać wyniki jego działania. Zdanie $\{p\}A\{q\}$ jest prawdziwe w sensie częściowej poprawności (ang. *partial correctness*), jeżeli każde kończące się wykonanie algorytmu A z danymi wejściowymi spełniającymi warunek p kończy się z danymi wyjściowymi spełniającymi warunek q. Zdanie $\{p\}A\{q\}$ jest prawdziwe w sensie pełnej poprawności (ang. *total correctness*), jeżeli każde wykonanie algorytmu A z danymi wejściowymi spełniającymi warunek p się kończy, a dane wyjściowe spełniają warunek q. Kończenie się działania algorytmu dla wszystkich danych wejściowych spełniających warunek wstępny p zwane jest warunkiem stopu (ang. *stop condition*). Jego udowodnienie sprowadza się do wykazania, że obliczenia wszystkich instrukcji iteracyjnych (pętli) w algorytmie zawsze się kończą. Dowód pełnej poprawności algorytmu polega więc na wykazaniu częściowej poprawności oraz spełnienia warunku stopu [33].

W przypadku algorytmów MultiNAVN warunek wstępny, który powinny spełniać dane wejściowe jest iloczynem następujących warunków częściowych:

- G – graf $G = (V, E)$ reprezentujący mapę drogową, będący grafem skierowanym,
- C – macierz kosztów: $C(i, j, l) > 0, \forall(i, j) \in E \wedge \forall l \in M$,
- s – węzeł początkowy: $s \in V$,
- d – węzeł docelowy: $d \in V$,
- N_c – liczba cykli algorytmu: $N_c > 0$,
- N_i – liczba iteracji: $N_i > 0$,
- N_m – liczba mrówek: $N_m > 0$.

Rezultatem działania algorytmu jest przybliżony zbiór rozwiązań paretooptymalnych $\mathbb{P}$, który w szczególnym przypadku może być zbiorem pustym w sytuacji, gdy nie zostało znalezione jakiekolwiek rozwiązanie. A zatem warunek ostateczny, który powinny spełniać dane zwracane przez algorytmy MultiNAVN jest iloczynem następujących warunków częściowych:

- Jeżeli algorytm wyznaczył co najmniej jedno rozwiązanie:
  - $\forall T \in \mathbb{P}, T$ jest drogą w grafie G od wierzchołka s do d,

– $\forall T \in \mathbb{P}$ spełniony jest warunek $\nexists T' \in \mathbb{P} | T' \preceq T$.

- Jeżeli algorytm nie wyznaczył rozwiązań [1], to $\mathbb{P} = \emptyset$.

Dodawanie rozwiązań do zbioru $\mathbb{P}$ jest realizowane w procedurze AktualizujZbiórPareto, która ma taką samą postać dla obu wersji algorytmu MultiNAVN, dlatego można ją przeanalizować na podstawie pseudokodu Algorytm 6. Ponieważ rozwiązania T_k dodawane do zbioru $\mathbb{P}$ są tworzone na podstawie zawartości poszczególnych tablic $TABU_k$ to oznacza, że każde rozwiązanie należące do zbioru $\mathbb{P}$ jest poprawną drogą od wierzchołka s do d. Warunek sprawdzany w wierszu 75 gwarantuje, że do zbioru $\mathbb{P}$ trafić mogą jedynie rozwiązania wyznaczone przez mrówki, które dotarły do węzła docelowego d. W instrukcjach w wierszach 76-79 sprawdza się, czy rozwiązanie wyznaczone przez mrówkę nie jest zdominowane przez jakiekolwiek rozwiązanie znajdujące się w zbiorze $\mathbb{P}$. Jeżeli nie jest, to rozwiązanie jest dodawane do zbioru, a wszystkie inne rozwiązania, które są przez nie zdominowane są usuwane ze zbioru wynikowego. Dowodzi to częściowej poprawności algorytmów MultiNAVN, a więc tego, że każde kończące się wykonanie algorytmu z danymi wejściowymi spełniającymi warunek wstępny kończy się z danymi wyjściowymi spełniającymi warunek końcowy.

W dalszej kolejności należy przeanalizować obliczenia instrukcji iteracyjnych algorytmów MultiNAVN tak, aby wykazać spełnienie warunku stopu. Dwie pętle w głównej części algorytmu decydują o jego zakończeniu. Pierwsza pętla obejmująca zakres wierszy 10-37 jest wykonywana dokładnie tyle razy, ile wynosi zadana liczba cykli powtórzeń algorytmu N_c. A zatem po wykonaniu N_c cykli ta pętla się zakończy. Druga pętla pomiędzy wierszami 13-31 jest wykonywana dopóty, dopóki spełniony jest warunek: $I \leqslant N_i \wedge$ *istnieje aktywna mrówka*. Pętla jest przerywana, gdy wszystkie mrówki dotrą do węzła docelowego, lub zostanie przekroczone ograniczenie na liczbę iteracji N_i. A zatem także w przypadku tej pętli istnieje gwarancja, że algorytm zawsze zakończy działanie. Pozostałe pętle występujące w algorytmach MultiNAVN nie budzą wątpliwości i nie stanowią przeszkody w zakończeniu działania analizowanych algorytmów. W ten sposób wykazaliśmy pełną poprawność algorytmów MultiNAVN-Z i MultiNAVN-L.

Do przeanalizowania pozostaje kwestia jaka wartość N_i dla poprawnych danych wejściowych gwarantuje wyznaczenie rozwiązań w każdym cyklu algorytmu, a w konsekwencji spowoduje, że zwrócony zostanie niepusty zbiór rozwiązań paretooptymalnych $\mathbb{P}$. Okazuje się, że odpowiedź na to pytanie jest stosunkowo prosta i wynika bezpośrednio z konstrukcji algorytmu. Aby dotrzeć do węzła docelowego każda mrówka w najgorszym przypadku musi przejść przez każdą krawędź grafu G dwa razy. Raz idąc do przodu, a drugi raz cofając się z miejsca bez drogi wyjściowej. Stąd wniosek, że wartość $N_i = 2|E|$ gwarantuje, że dla poprawnych danych wejściowych, algorytm zwróci niepusty zbiór rozwiązań paretooptymalnych $\mathbb{P}$.

5.5 Analiza złożoności obliczeniowej algorytmu MultiNAVN-Z

Analiza pesymistycznej złożoności czasowej dla algorytmu MultiNAVN-Z została przedstawiona w postaci komentarzy do pseudokodu Algorytm 8. Dla każdej operacji

[1] Taki przypadek może wystąpić gdy $N_i < 2|E|$ i zostało to szerzej wyjaśnione w dalszej części analizy.

algorytmu oszacowana została pesymistyczna złożoność czasowa i zapisana w komentarzach w poszczególnych wierszach algorytmu. Całkowita złożoność czasowa algorytmu (wzór (5.28)) została wyrażona jako funkcja następujących parametrów:

- $G = (V, E)$ – graf reprezentujący mapę,
- M – zbiór kryteriów oceny rozwiązań,

i wyraża się następującym ogólnym wzorem:

$$\begin{aligned} T(|V|, |E|, |M|, \triangle(G)) &= |E|s_1 + N_c \cdot \{N_m s_2 + N_i N_m \triangle(G) s_3 \\ &\quad + N_m s_4 + N_m |M| s_5 + |V| s_6 + N_m |V| s_7\} \\ &= O(|V| + |E| + |M| + \triangle(G)), \end{aligned} \tag{5.28}$$

gdzie $|E|s_1$, $N_m s_2$, $N_i N_m \triangle(G) s_3$, $N_m s_4$, $N_m |M| s_5$, $|V| s_6$, $N_m |V| s_7$, są kosztami operacji, odpowiednio, w wierszach 6-9, 11-12, 13-31, 32, 33, 34, 35, dla pewnych stałych s_1, s_2, s_3, s_4, s_5, s_6, s_7. Czas wykonania funkcji PrzygotójNormalizację w wierszu 5 nie został uwzględniony, ponieważ normalizacja może zostać wykonana jednorazowo przed wykonaniem algorytmu jako element przygotowania danych wejściowych.

Pesymistyczna złożoność czasowa stanowi górne ograniczenie czasu działania algorytmu, zwykle występujące dla szczególnych zbiorów danych wejściowych. Zadaniem algorytmów MultiNAVN jest optymalizacja tras na mapach drogowych, które są reprezentowane jako struktury grafowe. Przypadek pesymistyczny dla tego typu danych wejściowych występuje dla grafu pełnego. Wówczas, jeżeli $|V|$ oznaczymy przez n to $|E| = \frac{n(n-1)}{2} = O(n^2)$, a $\triangle_{pes}(G) = n - 1 = O(n)$. Liczbę kryteriów oceny rozwiązań w zastosowaniach w nawigacji samochodowej możemy uznać za ograniczoną od góry pewną stałą. Przyjmując powyższe założenia wzór (5.28) określający pesymistyczną, czasową złożoność algorytmu MultiNAVN-Z przyjmuje następującą postać:

$$T_{pes}(n) = O(n + n^2 + const. + n) = O(n^2). \tag{5.29}$$

Średnia liczba krawędzi wchodzących i wychodzących z wierzchołka grafu dla rzeczywistych danych z systemu OpenStreetMap wynosi ok. 2,2, mapy drogowe są bowiem opisane grafami rzadkimi. Z tego względu stopień grafu $\triangle(G)$ występujący w szacowaniu dla wierszy 19, 20, 45, 46, 51, 53 można zastąpić średnim stopniem wierzchołka $\triangle_{śr}(G) := \text{avg}\{\deg(v) : \forall v \in V(G)\} = O(1)$. Na potrzeby eksperymentów z algorytmami MultiNAVN przyjęto, że maksymalna liczba iteracji w każdym cyklu jest równa liczbie wierzchołków w grafie $N_i = |V|$. W konsekwencji, korzystając z wzoru (5.28), średnia złożoność czasową algorytmu MultiNAVN-Z w zastosowaniach w nawigacji samochodowej może być oszacowana jako:

$$\begin{aligned} T_{śr}(|V|, |E|, |M|) &= |E|s_1 + N_c \cdot \{N_m s_2 + |V| N_m \triangle_{śr}(G) s_3 \\ &\quad + N_m s_4 + N_m |M| s_5 + |V| s_6 + N_m |V| s_7\} \\ &= O(|E| + |V| + |M|). \end{aligned} \tag{5.30}$$

Analogicznie jak w przypadku złożoności pesymistycznej, jeżeli $|V|$ oznaczymy przez n to liczbę krawędzi możemy oszacować jako $|E| \approx 2n$. Przyjmując, że liczba kryteriów oceny rozwiązań jest wielkością stałą, średnią złożoność czasową algorytmu MultiNAVN-Z można wyrazić jako:

$$T_{śr}(n) = O(n + n + const.) = O(n). \tag{5.31}$$

Pamięciowa złożoność obliczeniowa algorytmu MultiNAVN-Z jest równa liczbie komórek pamięci potrzebnych do przechowywania danych podczas realizacji tego algorytmu w funkcji $|V|$, $|E|$, $|M|$. W przypadku algorytmu MultiNAVN-Z można wyróżnić następujące struktury pamięci i ich szacowaną, pesymistyczną złożoność pamięciową:

- $\mathbb{P}$ – $O(N_m N_c(|V| + |M|))$,
- ψ^+ – $O(1)$,
- ψ_k – $O(N_m)$,
- T^+ – $O(|V| + |M|)$,
- T – $O(|V| + |M|)$,
- T_k – $O(N_m(|V| + |M|))$,
- θ^+ – $O(1)$,
- v_k – $O(N_m)$,
- a_k – $O(N_m)$,
- $TABU_k$ – $O(N_m|V|)$,
- $blindEdges_k$ – $O(N_m|E|)$,
- i – $O(1)$,
- j – $O(1)$,
- k – $O(1)$,
- I – $O(1)$,
- I_c – $O(1)$,
- $\tau_{i,j}$ – $O(|E|)$,
- p^k – $O(N_m \triangle (G))$,
- Q – $O(1)$.

Pamięciowa złożoność obliczeniowa algorytmu MultiNAVN-Z zależy od liczności zbiorów V, E i M, liczby mrówek N_m oraz liczby cykli algorytmu N_c i można ją wyrazić następująco:

$$\begin{aligned} S(|V|,|E|,|M|,\triangle(G)) &= N_m N_c(|V|+|M|) + N_m + |V| + |M| + |V| + |M| \\ &\quad + N_m(|V|+|M|) + N_m + N_m + N_m|V| \\ &\quad + N_m|E| + |E| + N_m \triangle(G) \\ &= O(|V|+|E|+|M|+\triangle(G)). \end{aligned} \tag{5.32}$$

Przyjmując takie same założenia, które przyjęto w przypadku analizy czasowej złożoności obliczeniowej algorytmu MultiNAVN-Z, pesymistyczna i średnia pamięciowa złożoność obliczeniowa tego algorytmu może być oszacowana jako:

$$S_{pes}(n) = O(n + n^2 + const. + n) = O(n^2) \tag{5.33}$$

$$S_{śr}(n) = O(n + n + const. + const.) = O(n). \tag{5.34}$$

5.6 Analiza złożoności obliczeniowej algorytmu MultiNAVN-L

Analiza pesymistycznej złożoności czasowej dla algorytmu MultiNAVN-L została przeprowadzona w sposób analogiczny, jak w przypadku algorytmu MultiNAVN-Z i jest przedstawiona w postaci komentarzy do pseudokodu Algorytm 9. Różnica w złożoności obliczeniowej algorytmu MultiNAVN-L wynika z braku procedury OceńMrówki oraz innego sposobu realizacji procedury NagrodźNajlepszeRozwiązania. Całkowita złożoność czasowa algorytmu (wzór (5.35)) została wyrażona jako funkcja następujących parametrów:

- $G = (V, E)$ – graf reprezentujący mapę,
- M – zbiór kryteriów oceny rozwiązań,

i wyraża się następującym ogólnym wzorem:

$$\begin{aligned} T(|V|,|E|,|M|,\triangle(G)) &= |E|s_1 + N_c \cdot \{N_m s_2 + N_i N_m \triangle(G) s_3 \\ &\quad + N_m|M|s_4 + N_c N_m|V|s_5 + N_m|V|s_6\} \\ &= O(|V|+|E|+|M|+\triangle(G)), \end{aligned} \tag{5.35}$$

gdzie $|E|s_1$, $N_m s_2$, $N_i N_m \triangle(G) s_3$, $N_m|M|s_4$, $N_c N_m|V|s_5$, $N_m|V|s_6$, są kosztami operacji, odpowiednio, w wierszach 5-7, 9-10, 11-29, 30, 31, 32, dla pewnych stałych s_1, s_2, s_3, s_4, s_5, s_6.

Algorytm 8 Analiza pesymistycznej złożoności czasowej algorytmu MultiNAVN-Z

Dane wejściowe: G, C, s, d, N_c, N_i, N_m
Dane wyjściowe: przybliżony zbiór rozwiązań paretooptymalnych $\mathbb{P}$
Dane pomocnicze: ψ^+, ψ_k, T^+, T, T_k, θ^+, v_k, a_k, $TABU_k$, $blindEdges_k$, i, j, k, I, I_c, $\tau_{i,j}$, p^k, Q

function MultiNAVN-Z(G, C, s, d, N_c, N_i, N_m)
    PrzygotujNormalizację(G, C, η) ▷ –
    $\tau_{i,j} \leftarrow \tau_0, \forall (i,j) \in E$ ▷ $O(|E|)$
    $T^+ \leftarrow \emptyset$ ▷ $O(1)$
    $\mathbb{P} \leftarrow \emptyset$ ▷ $O(1)$
    $\psi^+ \leftarrow \infty$ ▷ $O(1)$
    for $I_{cykl} \leftarrow 1$ **to** N_c **do** ▷ $\times N_c$
        $v_k \leftarrow s, a_k \leftarrow true, TABU_k \leftarrow \emptyset, \forall k \in \{1, \ldots, N_m\}$ ▷ $O(N_m)$
        $I \leftarrow 1$ ▷ $O(1)$
        while $I \leqslant N_i \wedge$ istnieje aktywna mrówka **do** ▷ $\times N_i$
            for $k \leftarrow 1$ **to** N_m **do** ▷ $\times N_m$
                if $a_k = true$ **then** ▷ $O(1)$
                    if $v_k = d$ **then** ▷ $O(1)$
                        $a_k \leftarrow false$ ▷ $O(1)$
                    else
                        $p^k \leftarrow$ WyznaczPrawdop(v_k, $TABU_k$) ▷ $O(\triangle(G))$
                        $j \leftarrow$ WybierzKrawędź(p^k) ▷ $O(\triangle(G))$
                        if $j = 0$ **then** ▷ $O(1)$
                            CofnijMrówkę(k, i, j) ▷ $O(1)$
                        else
                            $v_k \leftarrow j$ ▷ $O(1)$
                            $TABU_k \leftarrow TABU_k \cup \{j\}$ ▷ $O(1)$
                        end if
                    end if
                end if
            end for
            $I \leftarrow I + 1$
        end while
        OceńMrówki ▷ $O(N_m)$
        AktualizujZbiórPareto ▷ $O(N_m|M|)$
        NagrodźNajlepszeRozwiązania ▷ $O(|V|)$
        UkarajNajgorszeMrówki ▷ $O(N_m|V|)$
        $Q \leftarrow Q \cdot \varphi$ ▷ $O(1)$
    end for
    return $\mathbb{P}$ ▷ Zwróć przybliżony zbiór Pareto
end function

procedure PrzygotujNormalizację(**in** G, **in** C, **out** η)
    for all $l \in M$ **do**
        $\eta_l \leftarrow$ wylicz według wzorów ((5.12) - (5.17))
    end for
end procedure

Algorytm 8 Analiza pesymistycznej złożoności czasowej algorytmu MultiNAVN-Z (c.d)

45: **function** WyznaczPrawdop(i, $TABU_k$) ▷ $O(\triangle(G))$
46: **for** $j \leftarrow 1$ **to** liczba krawędzi wychodzących z i **do** ▷ $\times \triangle(G)$
47: $p^k_{i,j} \leftarrow$ wynik obliczeń według wzoru ((5.18)) ▷ $O(1)$
48: **end for**
49: **return** p^k ▷ $O(1)$
50: **end function**

51: **function** WybierzKrawędź(p^k) ▷ $O(\triangle(G))$
52: $q \leftarrow$ liczba losowa $\in \langle 0, 1 \rangle$ ▷ $O(1)$
53: $j \leftarrow$ krawędź wybrana zgodnie z wzorem ((5.7)) ▷ $O(\triangle(G))$
54: $\tau_{i,j} \leftarrow (1 - \rho) \cdot \tau_{i,j} + \rho \cdot \tau_0$ ▷ $O(1)$
55: **return** j ▷ $O(1)$
56: **end function**

57: **procedure** CofnijMrówkę(**in** k, **in** i, **in** j) ▷ $O(1)$
58: $TABU_k \leftarrow TABU_k \setminus \{i\}$ ▷ $O(1)$
59: $blindEdges_k \leftarrow blindEdges_k \cup \{i, j\}$ ▷ $O(1)$
60: $\tau_{i,j} \leftarrow \tau_{i,j} \cdot \omega_b$ ▷ $O(1)$
61: **end procedure**

62: **procedure** OceńMrówki ▷ $O(N_m)$
63: **for** $k \leftarrow 1$ **to** N_m **do** ▷ $\times N_m$
64: **if** $v_k = d$ **then** ▷ Jeżeli mrówka dotarła do celu ▷ $O(1)$
65: $\psi_k \leftarrow$ wartość kryterium zastępczego dla trasy ▷ $O(1)$
66: **if** $\psi_k < \psi^+$ **then** ▷ $O(1)$
67: $T^+ \leftarrow TABU_k$ ▷ $O(1)$
68: $\psi^+ \leftarrow \psi_k$ ▷ $O(1)$
69: **end if**
70: **end if**
71: **end for**
72: **end procedure**

73: **procedure** AktualizujZbiórPareto ▷ $O(N_m|M|)$
74: **for** $k \leftarrow 1$ **to** N_m **do** ▷ $\times N_m$
75: **if** $v_k = d$ **then** ▷ Jeżeli mrówka dotarła do celu ▷ $O(1)$
76: // Czy w zbiorze $\mathbb{P}$ nie istnieją rozwiązania dominujące T_k?
77: **if** $\nexists T \in \mathbb{P} | T \preceq T_k$ **then** ▷ $O(|M|)$
78: $\mathbb{P} \leftarrow \mathbb{P} \cup \{T_k\}$ ▷ $O(1)$
79: $\mathbb{P} \leftarrow \mathbb{P} \setminus \{T \in \mathbb{P} | T_k \preceq T\}$ ▷ $O(|M|)$
80: **end if**
81: **end if**
82: **end for**
83: **end procedure**

Algorytm 8 Analiza pesymistycznej złożoności czasowej algorytmu MultiNAVN-Z (c.d)

84: **procedure** NagrodźNajlepszeRozwiązania ▷ $O(|V|)$
85: $\tau_{i,j} \leftarrow (1-\rho)\cdot\tau_{i,j} + 2\cdot\rho\cdot\frac{\theta^+}{\psi^+}, \forall(i,j)\in T^+$ ▷ $O(|V|)$
86: **end procedure**

87: **procedure** UkarajNajgorszeMrówki ▷ $O(N_m|V|)$
88: **for** $k \leftarrow 1$ **to** N_m **do** ▷ $\times N_m$
89: **if** $v_k \neq d$ **then** ▷ $O(1)$
90: $\tau_{i,j} \leftarrow \tau_{i,j}\cdot\omega_p, \forall(i,j)\in TABU_k$ ▷ $O(|V|)$
91: **end if**
92: **end for**
93: **end procedure**

Przyjmując te same założenia, co w przypadku analizy złożoności obliczeniowej dla algorytmu MultiNAVN-Z (wzór (5.29)) pesymistyczną czasową złożoność obliczeniową algorytmu MultiNAVN-L w zastosowaniach w nawigacji samochodowej można oszacować jako:

$$T_{pes}(n) = O(n + n^2 + const. + n) = O(n^2). \tag{5.36}$$

W konsekwencji średnia złożoność obliczeniowa algorytmu MultiNAVN-L w zastosowaniach w nawigacji samochodowej może być wyrażona następująco:

$$\begin{aligned} T_{śr}(|V|,|E|,|M|) &= |E|s_1 + N_c\cdot\{N_m s_2 + |V|N_m \triangle (G)_{śr}(G)s_3 \\ &\quad + N_m|M|s_4 + N_c N_m|V|s_5 + N_m|V|s_6\} \\ &= O(|E| + |V| + |M|). \end{aligned} \tag{5.37}$$

Podobnie jak w przypadku analizy MultiNAVN-Z, jeżeli $|V|$ oznaczymy przez n to liczbę krawędzi możemy oszacować jako $|E| \approx 2n$. Uznając liczbę kryteriów oceny rozwiązań za wielkość stałą, wówczas średnią złożoność czasową algorytmu MultiNAVN-L można wyrazić (5.38) jako:

$$T_{śr}(n) = O(n + n + const.) = O(n). \tag{5.38}$$

W przypadku algorytmu MultiNAVN-L można wyróżnić następujące struktury pamięci i ich szacowaną, pesymistyczną złożoność pamięciową w funkcji $|V|$, $|E|$, $|M|$:

- $\mathbb{P}$ – $O(N_m N_c(|V| + |M|))$,
- ψ_T – $O(1)$,
- T – $O(|V| + |M|)$,
- T_k – $O(N_m(|V| + |M|))$,
- θ_T – $O(1)$,

- v_k – $O(N_m)$,
- a_k – $O(N_m)$,
- $TABU_k$ – $O(N_m|V|)$,
- $blindEdges_k$ – $O(N_m|E|)$,
- i – $O(1)$,
- j – $O(1)$,
- k – $O(1)$,
- I – $O(1)$,
- I_c – $O(1)$,
- $\tau_{i,j}$ – $O(|E|)$,
- p^k – $O(N_m \triangle (G))$,
- Q – $O(1)$.

Można zauważyć, że podobnie jak w przypadku algorytmu MultiNAVN-Z pesymistyczna, pamięciowa złożoność obliczeniowa algorytmu MultiNAVN-L zależy od liczności zbiorów V, E i M, stopnia grafu G, liczby mrówek N_m oraz liczby cykli algorytmu N_c i można ją oszacować jako:

$$\begin{aligned} S(|V|,|E|,|M|,\triangle(G)) &= N_m N_c(|V|+|M|)+|V|+|M|+N_m(|V|+|M|) \\ &\quad + N_m + N_m + N_m|V| + N_m|E| + |E| + N_m \triangle (G) \\ &= O(|V|+|E|+|M|+\triangle(G)). \end{aligned} \tag{5.39}$$

Przyjmując takie same założenia, które przyjęto w przypadku analizy czasowej złożoności obliczeniowej algorytmów to pesymistyczna i średnia pamięciowa złożoność obliczeniowa tego algorytmu może być oszacowana jako:

$$S_{pes}(n) = O(n + n^2 + const. + n) = O(n^2) \tag{5.40}$$

$$S_{śr}(n) = O(n + n + const. + const.) = O(n). \tag{5.41}$$

Złożoności czasowe i pamięciowe obu algorytmów są niskiego rzędu – liniowe lub kwadratowe – ale mogą być one obarczone dużą stałą proporcjonalności. Przykładem mogą być wartości parametrów wykorzystane podczas eksperymentów porównawczych: $N_c = 500$ i $N_m = 80$. W efekcie stała proporcjonalności $N_c \cdot N_m$, która występuje we wzorach ogólnych na złożoność obliczeniową przyjmuje wartość 40000 i jest większa niż liczba węzłów w grafie reprezentującym mapę drogową. Jednocześnie należy mieć na uwadze, że niższa złożoność obliczeniowa algorytmów MultiNAVN-Z i MultiNAVN-L wynika z tego, że oba algorytmy wyznaczają rozwiązania przybliżone, których jakość może być mierzona odległością (metryka Hausdorffa) od pełnych zbiorów rozwiązań.

Algorytm 9 Analiza pesymistycznej złożoności czasowej algorytmu MultiNAVN-L

1: **Dane wejściowe:** G, C, s, d, N_c, N_i, N_m
2: **Dane wyjściowe:** przybliżony zbiór rozwiązań paretooptymalnych $\mathbb{P}$
3: **Dane pomocnicze:** ψ_T, T, T_k, θ_T, v_k, a_k, $TABU_k$, $blindEdges_k$, i, j, k, I, I_c, $\tau_{i,j}$, p^k, Q

4: **function** MultiNAVN-L(G, C, s, d, N_c, N_i, N_m)
5: $\tau_{i,j} \leftarrow \tau_0, \forall (i,j) \in E$ ▷ $O(|E|)$
6: $\mathbb{P} \leftarrow \emptyset$ ▷ $O(1)$
7: $\psi^+ \leftarrow \infty$ ▷ $O(1)$
8: **for** $I_{cykl} \leftarrow 1$ **to** N_c **do** ▷ $\times N_c$
9: $v_k \leftarrow s, a_k \leftarrow true, TABU_k \leftarrow \emptyset, \forall k \in \{1, \ldots, N_m\}$ ▷ $O(N_m)$
10: $I \leftarrow 1$ ▷ $O(1)$
11: **while** $I \leqslant N_i \wedge$ istnieje aktywna mrówka **do** ▷ $\times N_i$
12: **for** $k \leftarrow 1$ **to** N_m **do** ▷ $\times N_m$
13: **if** $a_k = true$ **then** ▷ $O(1)$
14: **if** $v_k = d$ **then** ▷ $O(1)$
15: $a_k \leftarrow false$ ▷ $O(1)$
16: **else**
17: $p^k \leftarrow$ WyznaczPrawdop(v_k, $TABU_k$) ▷ $O(\triangle(G))$
18: $j \leftarrow$ WybierzKrawędź(p^k) ▷ $O(\triangle(G))$
19: **if** $j = 0$ **then** ▷ $O(1)$
20: CofnijMrówkę(k, i, j) ▷ $O(1)$
21: **else**
22: $v_k \leftarrow j$ ▷ $O(1)$
23: $TABU_k \leftarrow TABU_k \cup \{j\}$ ▷ $O(1)$
24: **end if**
25: **end if**
26: **end if**
27: **end for**
28: $I \leftarrow I + 1$
29: **end while**
30: AktualizujZbiórPareto ▷ $O(N_m|M|)$
31: NagrodźNajlepszeRozwiązania ▷ $O(N_c N_m|V|)$
32: UkarajNajgorszeMrówki ▷ $O(N_m|V|)$
33: $Q \leftarrow Q \cdot \varphi$ ▷ $O(1)$
34: **end for**
35: **return** $\mathbb{P}$ ▷ Zwróć przybliżony zbiór Pareto
36: **end function**

37: **function** WyznaczPrawdop(i, $TABU_k$) ▷ $O(\triangle(G))$
38: $l \leftarrow$ losowy numer kryterium ▷ $O(1)$
39: **for** $j \leftarrow 1$ **to** liczba krawędzi wychodzących z i **do** ▷ $\times \triangle(G)$
40: $p^k_{i,j} \leftarrow$ wynik obliczeń według wzoru ((5.18)) ▷ $O(1)$
41: **end for**
42: **return** p^k ▷ $O(1)$
43: **end function**

Algorytm 9 Analiza pesymistycznej złożoności czasowej algorytmu MultiNAVN-L (c.d)

```
44: function WybierzKrawędź(p^k)                                   ▷ O(△(G))
45:     q ← liczba losowa ∈ ⟨0, 1⟩                                  ▷ O(1)
46:     j ← krawędź wybrana zgodnie z wzorem ((5.7))                ▷ O(△(G))
47:     τ_{i,j} ← (1 − ρ) · τ_{i,j} + ρ · τ_0                       ▷ O(1)
48:     return j                                                    ▷ O(1)
49: end function

50: procedure CofnijMrówkę(in k, in i, in j)                       ▷ O(1)
51:     TABU_k ← TABU_k \ {i}                                       ▷ O(1)
52:     blindEdges_k ← blindEdges_k ∪ {i, j}                        ▷ O(1)
53:     τ_{i,j} ← τ_{i,j} · ω_b                                     ▷ O(1)
54: end procedure

55: procedure AktualizujZbiórPareto                                ▷ O(N_m|M|)
56:     for k ← 1 to N_m do                                         ▷ ×N_m
57:         if v_k = d then          ▷ Jeżeli mrówka dotarła do celu ▷ O(1)
58:             // Czy w zbiorze ℙ nie istnieją rozwiązania dominujące T_k?
59:             if ∄T ∈ ℙ|T ⪯ T_k then                              ▷ O(|M|)
60:                 ℙ ← ℙ ∪ {T_k}                                   ▷ O(1)
61:                 ℙ ← ℙ \ {T ∈ ℙ|T_k ⪯ T}                         ▷ O(|M|)
62:             end if
63:         end if
64:     end for
65: end procedure

66: procedure NagrodźNajlepszeRozwiązania                          ▷ O(N_c N_m|V|)
67:     for all T ∈ ℙ do                                            ▷ ×N_c × N_m
68:         τ_{i,j} ← (1 − ρ) · τ_{i,j} + 2 · ρ · θ_T/ψ_T, ∀(i, j) ∈ T   ▷ O(|V|)
69:     end for
70: end procedure

71: procedure UkarajNajgorszeMrówki                                ▷ O(N_m|V|)
72:     for k ← 1 to N_m do                                         ▷ ×N_m
73:         if v_k ≠ d then                                         ▷ O(1)
74:             τ_{i,j} ← τ_{i,j} · ω_p, ∀(i, j) ∈ TABU_k           ▷ O(|V|)
75:         end if
76:     end for
77: end procedure
```

Rozdział 6

Badania eksperymentalne

Celem przeprowadzonych badań eksperymentalnych było wykazanie możliwości zastosowania opracowanych algorytmów mrowiskowych do wyznaczania optymalnych dróg dla nawigacji samochodowej. Wyniki uzyskane przez wybrane algorytmy mrowiskowe z rodziny AVN były porównywane z wynikami uzyskanymi przez algorytm Label Setting, który jest najbardziej znanym algorytmem deterministycznym rozwiązującym problem MOSP.

Analiza porównawcza umożliwiła określenie różnic w czasie działania poszczególnych algorytmów oraz ich wymagań co do pamięci operacyjnej komputera. Pełniejsze zbiory rozwiązań paretooptymalnych, będące wynikiem działania algorytmu Label Setting, dały dodatkowo możliwość oszacowania jakości aproksymowanych rozwiązań uzyskiwanych przez algorytmy mrowiskowe.

Wszystkie eksperymenty były przeprowadzone na danych rzeczywistych pobranych z systemu OpenStreetMap (w skrócie OSM) uzupełnionych na ograniczonym obszarze Polski o dane o wypadkach i kolizjach drogowych uzyskanych dzięki uprzejmości Policji. Wykorzystane źródła danych umożliwiły otrzymanie zbiorów rozwiązań paretooptymalnych dla następujących kryteriów oceny: długość drogi, liczba pasów ruchu, liczba skrzyżowań oraz bezpieczeństwo.

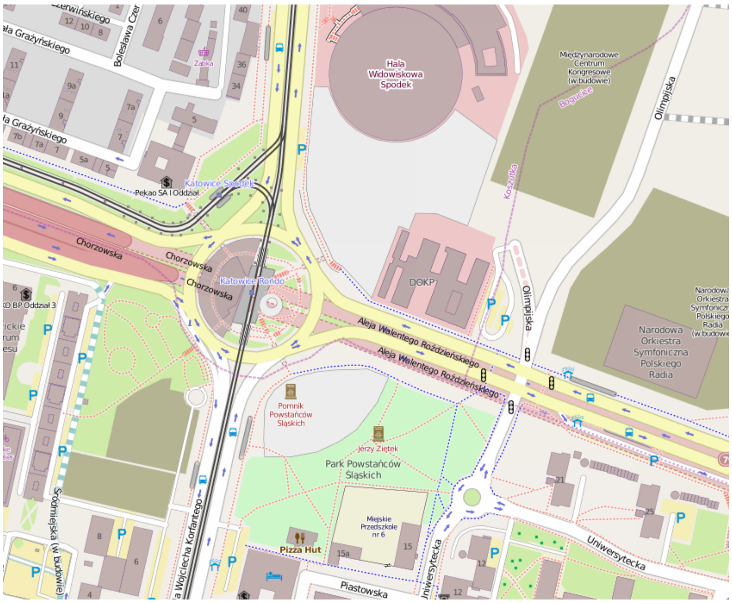
6.1 Źródła i metoda pozyskania danych doświadczalnych

Algorytmy wspomagające proces nawigacji samochodowej powinny być skuteczne dla rzeczywistych danych kartograficznych i ewidencyjnych opisujących przestrzeń, w której poruszają się pojazdy. Dlatego istotnym zagadnieniem przed przystąpieniem do przeprowadzenia eksperymentów było pozyskanie rzeczywistych danych. Zdecydowano się na wykorzystanie informacji kartograficznych z systemu OpenStreetMap uzupełnionych o dane o wypadkach i kolizjach na drodze z systemu SEWiK, którego operatorem jest Policja.

6.1.1 Dane kartograficzne z systemu OpenStreetMap

System OpenStreetMap jest internetowym systemem kartograficznym tworzonym przez społeczność wolontariuszy z całego świata na zasadach zbliżonych do Wikipedii. Dzięki temu szczegółowość map oraz szybkość reakcji na zmiany jest zbliżona, a niejednokrotnie

lepsza w stosunku do rozwiązań komercyjnych, a dane pochodzące z systemu mogą być wykorzystywane nieodpłatnie. Przykładowa mapa dla obszaru centrum miasta Katowic została zaprezentowana na rys. 6.1. Istnieje między innymi możliwość pobrania kompletnej bazy danych o zasięgu globalnym lub jej wybranych fragmentów, np. tylko Europy, tylko Polski. Treść bazy danych może być reprezentowana w wielu różnych formatach, takich jak: pliki XML, pliki binarne, pliki JSON, relacyjna baza danych. Najbardziej rozbudowane możliwości daje wykorzystanie relacyjnej bazy danych z rozszerzeniami GIS. W przypadku OSM zalecany jest silnik bazodanowy PostgreSQL z rozszerzeniem PostGIS.



Rysunek 6.1: Przykładowa mapa systemu OSM dla obszaru centrum Katowic

Schemat danych OSM

Dane kartograficzne o zasięgu globalnym dostępne w systemie OSM są bardzo obszerne i zróżnicowane. Ponieważ system OpenStreetMap jest rozwijany na zasadach otwartego oprogramowania (ang. *open source*) dostępna jest szczegółowa dokumentacja schematu bazy danych, która jest wykorzystywana przez różne narzędzia rozwijane w ramach projektu OpenStreetMap.

Dane kartograficzne OSM zbudowane są z następujących elementów:

- `nodes` – węzły,
- `ways, way_nodes` – łamane,
- `relations, relation_members` – relacje, składowe relacji.

Węzły opisywane są za pomocą podstawowych atrybutów:

- `id` – unikalny identyfikator numeryczny,
- `lat` – szerokość geograficzna,

- `lon` – długość geograficzna.

Uzupełnieniem podstawowych danych jest rozszerzalny system znaczników (ang. *tags*), który umożliwia określenie typów obiektów, ich nazw, alternatywnych oznaczeń oraz innych właściwości. Znaczniki są reprezentowane przez trzy tablice:

- `node_tags` – znaczniki dla węzłów,
- `way_tags` – znaczniki dla łamanych,
- `relation_tags` – znaczniki dla relacji.

Poszczególne elementy, a więc węzły, łamane i relacje, mogą mieć dowolną liczbę par „klucz (key) = wartość (value)". Obie składowe są dowolnymi tekstami, ale istnieje konwencja dla typowych zastosowań. Znaczniki umożliwiają rozszerzanie danych OSM o nowe elementy nieobjęte konwencją. Poniżej pokazano przykłady znaczników i ich wartości:

- `highway=motorway` – autostrada,
- `oneway=yes` – ulica jednokierunkowa,
- `name=Adama Mickiewicza` – nazwa ulicy,
- `building=church` – obszar określający położenie kościoła.

Dane przechowywane w podstawowych tablicach i rozszerzone o znaczniki mogą być wykorzystane m.in. do nawigacji samochodowej. W szczególności możliwe jest uzyskanie danych o:

- sieci połączeń drogowych,
- rodzajach dróg – autostrada, droga krajowa, osiedlowa, ścieżka dla pieszych,
- dozwolonych kierunkach ruchu,
- sygnalizacji świetlnej,
- nazwach ulic i miejscowości.

Dzięki temu możliwe było skorzystanie z danych systemu OSM do przeprowadzenia eksperymentów obliczeniowych z mrowiskowymi algorytmami nawigacji samochodowej. W dalszych podrozdziałach opisane są sposoby ustalania danych dla poszczególnych kryteriów oceny rozwiązań dotyczących długości drogi, szerokości drogi i liczby skrzyżowań.

Kryterium długości drogi

Długość drogi obliczana jest jako suma długości odcinków ją tworzących. Odległość między poszczególnymi węzłami jest obliczana z wykorzystaniem formuły:

$$d = 2 \cdot r \cdot \arcsin \sqrt{\text{haversin}(\phi_A - \phi_B) + \cos \phi_A \cdot \cos \phi_B \cdot \text{haversin}(\lambda_B - \lambda_A)}, \qquad (6.1)$$

gdzie:

$\text{haversin}(\theta) = \sin^2\left(\frac{\theta}{2}\right)$,

d – odległość między dwoma punktami,

r – promień sfery (w tym przypadku Ziemi),

ϕ_A, ϕ_B – szerokości geograficzne punktów A i B,

λ_A, λ_B – długości geograficzne punktów A i B.

Kryterium szerokości drogi

Poszczególnym odcinkom drogi przydzielana jest waga odpowiadająca szerokości drogi, którą reprezentują. Waga ustalana jest na podstawie znacznika `highway` zdefiniowanego dla drogi (`way_tags`) według schematu:

- `motorway` – 20 (autostrady),
- `trunk` – 15 (drogi szybkiego ruchu, międzymiastowe),
- `primary` – 10 (drogi główne, zwykle dwupasmowe),
- `secondary` – 5 (drogi z jednym pasem),
- `tertiary` – 5 (drogi lokalnego ruchu),
- `unclassified` – 2 (drogi wąskie, często z jednym pasem dla obu kierunków),
- `road` – 2 (inne),
- `residential` – 2 (drogi osiedlowe).

Zauważmy, że im szersza jest droga, tym większa waga ustalana jest na odcinku, który jest opisany znacznikiem.

Kryterium liczby skrzyżowań

Poszczególnym węzłom drogi przydzielany jest znacznik oznaczający, że w miejscu węzła znajduje się skrzyżowanie z sygnalizacją świetlną. Decyzja jest podejmowana na podstawie wartości znaczników `highway` i `crossing` zdefiniowanych dla węzła (`node_tags`). Sygnalizacja oznaczana jest wartościami:

- `traffic_signals`,
- `traffic_lights`.

6.1.2 Dane o wypadkach i kolizjach z systemu SEWiK

Komenda Stołeczna Policji oraz komendy powiatowe korzystają z Systemu Ewidencji Wypadków i Kolizji o skróconej nazwie SEWiK. W bazach danych systemu są rejestrowane dane o zdarzeniach na drogach, m.in.: rodzaj zdarzenia, miejsce, czas i uczestnicy zdarzenia. Dostęp do danych systemu mają także podmioty spoza Policji na pisemny wniosek kierowany do Komendanta Powiatowego Policji. Dzięki uprzejmości przedstawicieli Komendy Miejskiej Policji w Katowicach uzyskano dane o zdarzeniach, które miały miejsce na drogach Katowic w latach 2010-2011. W dalszej części przedstawione zostaną sposoby integracji tych danych z systemem OSM i metody ich wykorzystania w eksperymentach z mrowiskowymi algorytmami nawigacji samochodowej.

Integracja z systemem OSM

Pierwszym etapem wykorzystania danych pochodzących z systemu SEWiK była integracja danych o zdarzeniach z mapą drogową OSM. Aby to było możliwe, konieczne było określenie lokalizacji zdarzeń, wyrażone jako węzeł lub łamana systemu OSM. Specyfika danych spowodowała, że nie był to proces całkowicie automatyczny. Na rys. 6.2 widać niejednoznaczny sposób określania miejsca wystąpienia zdarzenia.

Statystyka ogólna wg miesiąca roku. OKRES OD : '2011/01/01', DO : '2011/12/31

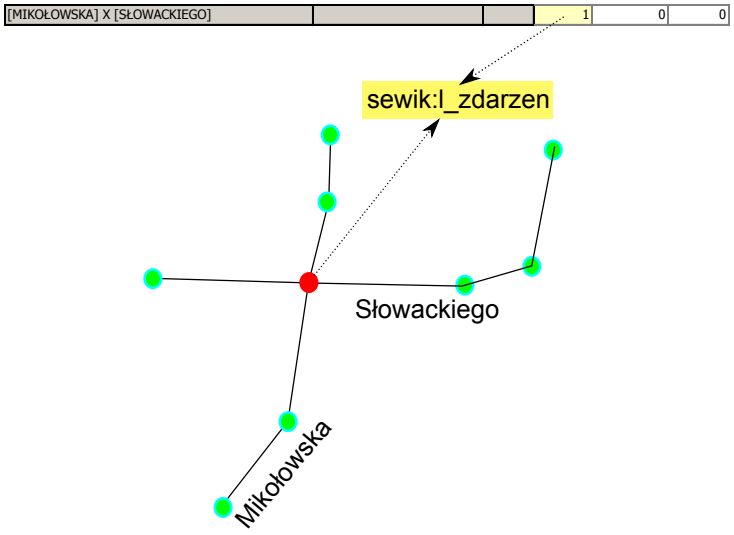
Rok zdarzenia: <Wszystkie> Komendy Miejskie/Powiatowe:KA KMP Katowice

					Liczba Zdarzen	Liczba Wypadkow	Liczba Zabitych	Liczba Rannych	Liczba Kolizji
Ogółem					5093	306	12	401	4787
	2011/01/01				3	0	0	0	3
		[REBACZY] X [FILAROWA]			1	0	0	0	1
		[SŁONECZNA] X [KORFANTEGO]			1	0	0	0	1
		NULL			1	0	0	0	1
			SOWIŃSKIEGO		1	0	0	0	1
	2011/01/02				1	0	0	0	1
		NULL			1	0	0	0	1
			PANEWNICKA		1	0	0	0	1
	2011/01/03				8	0	0	0	8
		[MIKOŁOWSKA] X [SŁOWACKIEGO]			1	0	0	0	1
		[RONDO GEN. ZIĘTKA] X [AL.ROŹDZIEŃSKIEGO]			1	0	0	0	1
		NULL			6	0	0	0	6
			AGNIESZKI		1	0	0	0	1
			LOTNISKO		1	0	0	0	1
			ROŹDZIEŃSKIEGO		1	0	0	0	1
			SAMSONOWICZA		1	0	0	0	1
			WOJCIECHA		1	0	0	0	1
			NULL		1	0	0	0	1
				K86	1	0	0	0	1

Rysunek 6.2: Przykład danych z systemu SEWiK przedstawiających niejednoznaczne określenie miejsca wystąpienia zdarzenia

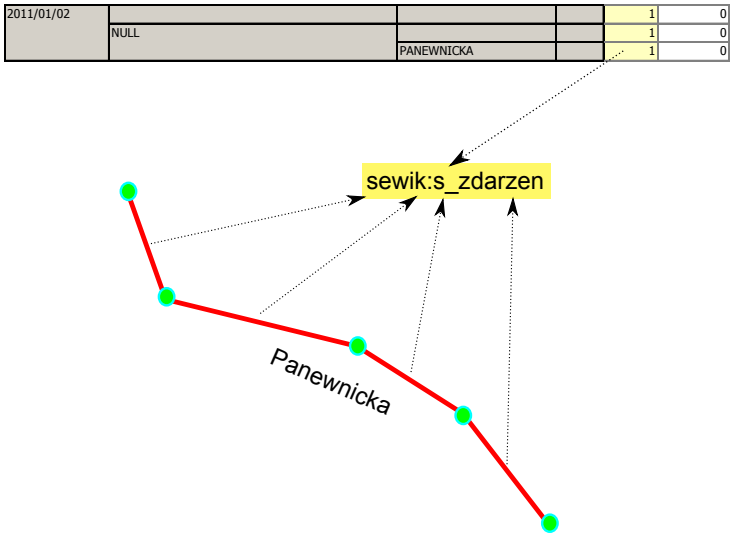
W systemie SEWiK przyjęto określać lokalizację zdarzenia na dwa sposoby, w zależności od tego, czy zdarzenie miało miejsce na skrzyżowaniu, czy też nie. W pierwszym przypadku miejsce oznaczane jest jako przecięcie się dwóch nazwanych ulic lub dróg, a w drugim oznaczana jest pojedyncza droga bez dokładnego sprecyzowania, na którym odcinku tej drogi doszło do kolizji. Oba przypadki są nieco inaczej integrowane z danymi OSM.

Dla pierwszego przypadku wyszukiwany jest węzeł OSM i tworzony jest znacznik `sewik:l_zdarzen`, którego wartością jest liczba zdarzeń w węźle. Przedstawione jest to na rys. 6.3.



Rysunek 6.3: Schemat wprowadzania do bazy OSM zdarzeń dla węzła

W drugim przypadku lokalizowana jest łamana lub zbiór łamanych, które reprezentują drogę, na której doszło do zdarzenia, i dla nich tworzony jest znacznik `sewik:s_zdarzen` zawierający wartość odpowiadającą średniej liczbie zdarzeń na metr długości drogi. Przedstawia to schemat na rys. 6.4.



Rysunek 6.4: Schemat wprowadzania do bazy OSM zdarzeń dla łamanej

Kryterium bezpieczeństwa

Na podstawie wartości znaczników `sewik:l_zdarzen` dla węzłów i `sewik:s_zdarzen` dla łamanych, możliwe stało się wykorzystanie danych z systemu SEWiK jako kryterium bezpieczeństwa dla przeprowadzonych eksperymentów obliczeniowych z algorytmami MultiNAVN. Składnik heurystyczny b (reprezentujący bezpieczeństwo) podczas oblicza-

nia prawdopodobieństwa przejścia mrówki z węzła A do węzła B jest obliczany według formuły:

$$b = k + d \cdot S + \frac{L}{d}, \tag{6.2}$$

gdzie:

k – minimalna wartość składnika,

d – odległość między dwoma węzłami A i B,

S – średnia liczba zdarzeń na metr drogi, której elementem jest krawędź z węzła A do węzła B (`sewik:s_zdarzen`),

L – liczba zdarzeń w węźle B (`sewik:l_zdarzen`).

6.2 Implementacja badanych algorytmów

Wszystkie wersje porównywanych algorytmów zostały zaimplementowane w języku Java i wykonywane były na komputerze wyposażonym w dwurdzeniowy procesor Intel Core i5 2,5 GHz, pracującym pod kontrolą systemu Windows 8 64-bit z zainstalowanym środowiskiem uruchomieniowym Java JRE 1.7.0 build 21 64-bit. Komputer był wyposażony w 6 GB pamięci operacyjnej RAM, przy czym dla programów Java zastały udostępnione 4 GB.

Algorytm Label Setting jest implementacją opartą na publikacjach Martinsa [83] i Gandibleux [54]. Przy badaniu kolejnych wierzchołków dodana została reguła sprawdzania, czy wierzchołek należy już do drogi związanej ze sprawdzaną etykietą. Dzięki temu możliwe stało się wyznaczanie dróg dla grafów uzyskanych na podstawie danych OSM, które zawierają cykle.

Algorytmy MultiNAVN-L i MultiNAVN-Z są dwoma wersjami algorytmu mrowiskowej nawigacji samochodowej MultiNAVN, który wyznacza aproksymację zbioru rozwiązań paretooptymalnych dla problemu MOSP. Oba algorytmy różnią się sposobem uwzględnienia wielokryterialnej informacji heurystycznej podczas obliczania prawdopodobieństwa wyboru krawędzi przez poruszającą się mrówkę. Algorytm MultiNAVN-L w każdym kolejnym kroku wybiera kryterium w sposób losowy, a MultiNAVN-Z wykorzystuje kryterium zastępcze.

6.3 Wyznaczenie wartości parametrów algorytmów mrowiskowych

Algorytmy mrowiskowe wymagają ustalenia (strojenia) wartości kilku parametrów, które mają wpływ na skuteczność algorytmów w rozwiązywaniu wybranego problemu lub konkretnego zestawu danych. Parametry dostrajano w zakresach:

- $\tau_0 \in \{0{,}000001,\ 0{,}00001,\ 0{,}0001,\ 0{,}001,\ 0{,}01\}$,
- $\alpha \in \{0{,}5,\ 1,\ 2\}$,

- $\beta \in \{0{,}5,\ 1,\ 2,\ 3,\ 5\}$,
- $q_0 \in \{0{,}5,\ 0{,}7,\ 0{,}8,\ 0{,}95\}$,
- $\rho \in \{0{,}1,\ 0{,}2,\ 0{,}3,\ 0{,}4,\ 0{,}5\}$,
- $\omega_b \in \{0{,}05,\ 0{,}1,\ 0{,}15,\ 0{,}20,\ 0{,}25,\ 0{,}30\}$,
- $\omega_p \in \{0{,}80,\ 0{,}85,\ 0{,}90,\ 0{,}95,\ 0{,}99\}$.

Wyznaczenie wartości parametrów algorytmów mrowiskowych wykonano zmieniając wartości kolejnych parametrów w zadanym zakresie, przy ustalonych wartościach pozostałych parametrów [12, 44]. Wartość parametru, w przypadku której algorytm mrowiskowy uzyskiwał rozwiązania o najlepszej jakości, każdorazowo była ustalana jako obowiązująca dla dalszych testów. Jako kryterium oceny jakości przyjęto wartość metryki Hausdorffa oraz liczność wynikowego zbioru rozwiązań. Operacje te wykonywano kolejno dla wszystkich badanych parametrów. Przyjęto następujące wartości początkowe parametrów:

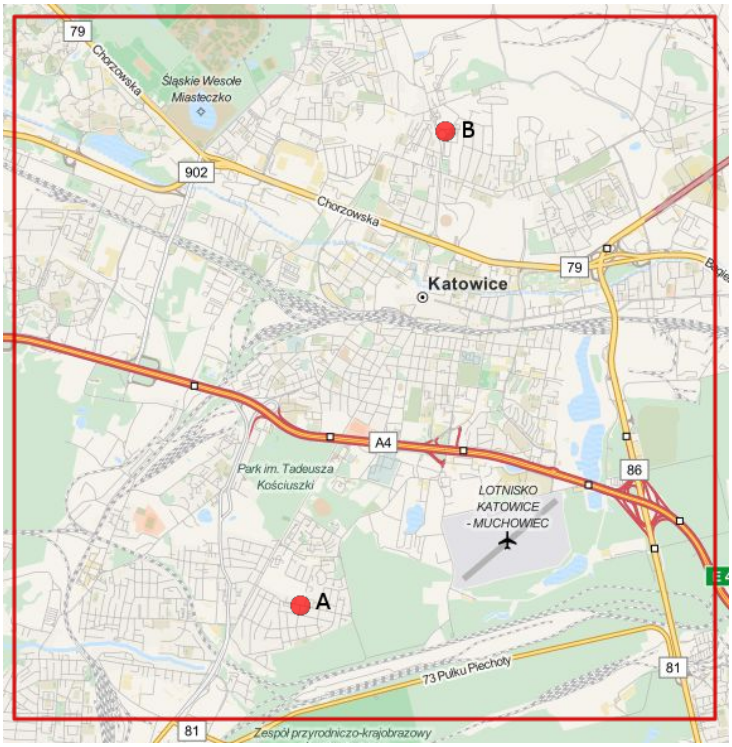
- $\tau_0 = 0{,}00001$,
- $\alpha = 1$,
- $\beta = 2$,
- $q_0 = 0{,}95$,
- $\rho = 0{,}2$,
- $\omega_b = 0{,}15$,
- $\omega_p = 0{,}95$.

Wartości parametrów: liczba mrówek oraz liczba iteracji algorytmów mrowiskowych były stałe, i zostały ustalone odpowiednio na $N_m = 32$ oraz $N_{cykl} = 100$.

Eksperymenty przeprowadzone zostały na mapie, dla której dostępne były dane o wypadkach i kolizjach z systemu SEWiK, tak aby uwzględniane były cztery kryteria optymalizacyjne. Ze względu na dużą złożoność obliczeniową i pamięciową algorytmu Label Setting, którego wyniki były podstawą do oceny jakości rozwiązań dostarczonych przez algorytmy mrowiskowe, powierzchnia mapy została ograniczona do obszaru centrum miasta Katowic (tabela 6.1). W celach referencyjnych mapa została oznaczona skrótem KAT-CEN.

Tabela 6.1: Mapa KAT-CEN użyta do wyznaczania wartości parametrów algorytmów mrowiskowych

Długość geograficzna	18,9939°E - 19,0490°E
Szerokość geograficzna	50,2294°N - 50,2740°N
Liczba węzłów	2514
Liczba krawędzi	4462
Liczba węzłów z sygnalizacją	38



Rysunek 6.5: Punkty na mapie KAT-CEN wykorzystane jako miejsca początkowe i końcowe dróg wykorzystanych do wyznaczania wartości parametrów algorytmów mrowiskowych

Tabela 6.2: Wyniki działania algorytmu Label Setting na mapie KAT-CEN

Czas [ms]	75179
Liczba rozwiązań Pareto	1392
Zajętość pamięci [kB]	119285

Jako punkt startowy (A) wybrano węzeł OSM o identyfikatorze 732709037, a punkt docelowy (B) o identyfikatorze 384912139. Oba punkty leżą w obszarze miasta Katowic.

W tabeli 6.2 zaprezentowano podsumowanie wyników działania algorytmu Label Setting. Dokładny zbiór rozwiązań paretooptymalnych będący wynikiem działania algorytmu umożliwiał obliczenie odległości między tym zbiorem, a przybliżonym zbiorem rozwiązań paretooptymalnych wyznaczanym przez algorytmy MultiNAVN-L i MultiNAVN-Z. Jako miarę odległości między zbiorami przyjęto metrykę Hausdorffa, która została opisana w podrozdziale 2.2.

Ze względu na stochastyczny charakter testowanych algorytmów, dla każdego zestawu wartości parametrów powtórzono eksperyment 30 razy. Mierzono następujące wielkości: czas trwania eksperymentu, liczebność zbioru i metrykę Hausdorffa dla zbioru rozwiązań paretooptymalnych zwróconych przez algorytm oraz przyrost zaalokowanej pamięci operacyjnej w trakcie eksperymentu. Dla poszczególnych zbiorów mierzonych wielkości obliczona została średnia arytmetyczna oraz odchylenie standardowe. W celu wyeliminowania grubych błędów, z każdego zbioru usunięto wartości skrajne. Dodatkowo zastosowano zasadę 3σ i odrzucano wyniki oddalone od wartości oczekiwanej o 3 wartości odchylenia standardowego σ.

6.3.1 Algorytm MultiNAVN-Z z kryterium zastępczym

Pierwsze eksperymenty mające na celu wyznaczenie wartości parametrów algorytmów mrowiskowych zostały przeprowadzone dla wersji algorytmu MultiNAVN-Z, która wykorzystuje kryterium zastępcze.

Pierwszym badanym parametrem była początkowa wartość śladu feromonowego τ_0. Rezultaty tych eksperymentów prezentują tab. 6.3 oraz rys. 6.6. Najlepsze wyniki uzyskano dla $\tau_0 = 0,0001$ i taką wartość przyjęto do dalszych eksperymentów. Dla tej wartości parametru τ uzyskano zdecydowanie największą średnią liczność zbioru rozwiązań niezdominowanych oraz stosunkowo niską wartość metryki Hausdorffa.

Tabela 6.3: Wyniki działania algorytmu MultiNAVN-Z na mapie KAT-CEN podczas wyznaczania wartości parametru τ_0

τ_0	**Czas [ms]**	**Liczba rozwiązań**	**Metryka H.**	**Pamięć [kB]**
0,000001	455,29	36,75	3395,05	5329,82
0,00001	456,21	43,93	1904,34	5501,68
0,0001	**535,79**	**52,00**	**1664,97**	**5542,25**
0,001	527,61	29,64	1323,89	5742,82
0,01	765,75	9,14	2551,12	7638,29

Następnie analizowano różne wartości parametru α oznaczającego wpływ śladu feromonowego na wynik reguły przejścia. Rezultaty analizy prezentują tab. 6.4 oraz rys. 6.7. Najlepsze wyniki uzyskano dla $\alpha = 1$ i taką wartość przyjęto do dalszych eksperymentów. Podobnie jak w przypadku wyznaczania wartości parametru τ, wybrana została wartość parametru, dla której uzyskano największą średnią liczność zbioru rozwiązań niezdominowanych oraz niską wartość metryki Hausdorffa.

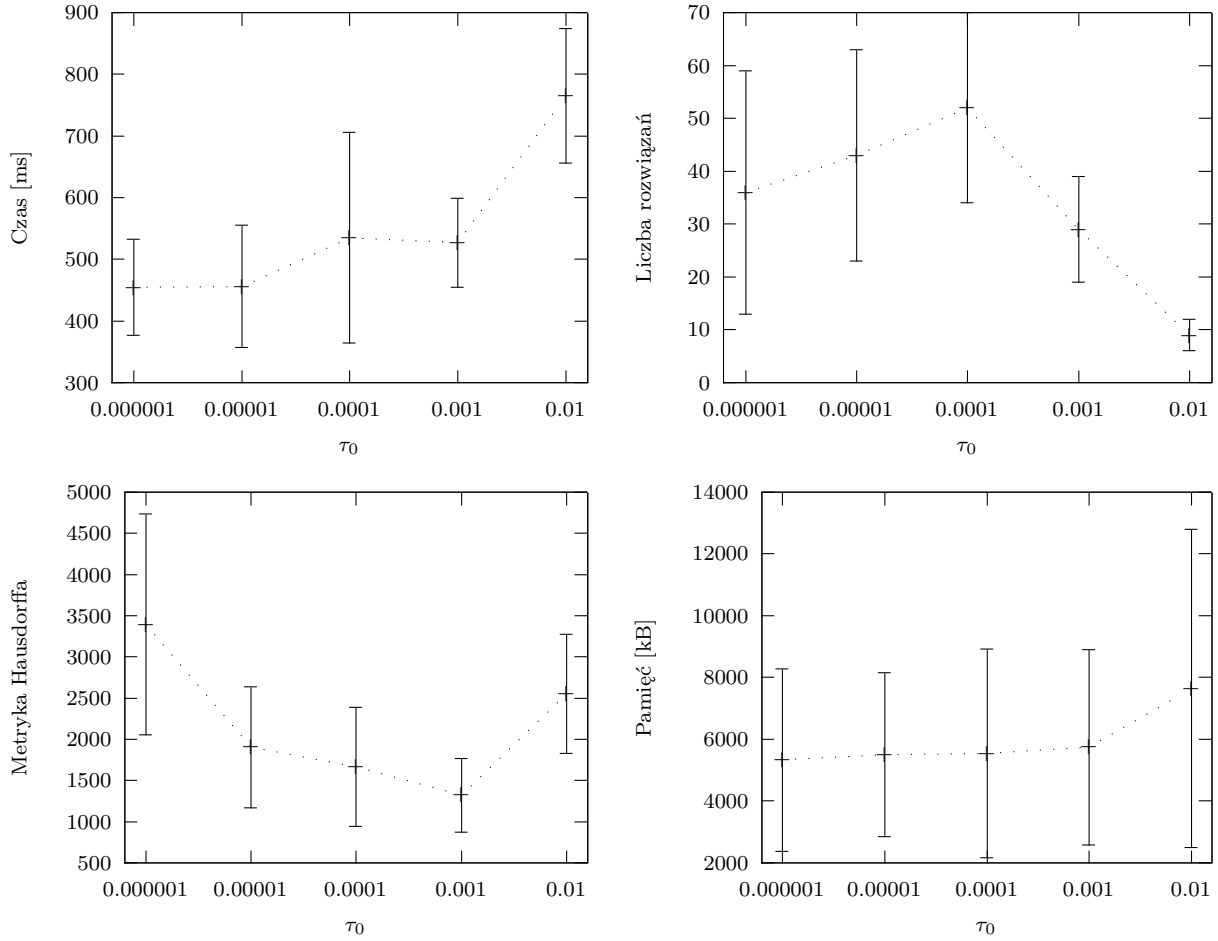
Tabela 6.4: Wyniki działania algorytmu MultiNAVN-Z na mapie KAT-CEN podczas wyznaczania wartości parametru α

α	**Czas [ms]**	**Liczba rozwiązań**	**Metryka H.**	**Pamięć [kB]**
0,5	966,93	35,61	1343,49	10094,11
1	**472,54**	**44,32**	**1487,54**	**5280,54**
2	386,21	30,61	2408,21	6560,43

Taką samą procedurę zastosowano do analizy parametru β, który określa wpływ informacji heurystycznej (widoczności) na wynik reguły przejścia. Rezultaty analizy prezentują tab. 6.5 oraz rys. 6.8. Zarówno największa średnia liczność zbioru rozwiązań niezdominowanych, jak i najniższa wartość metryki Hausdorffa została uzyskana dla wartości parametru $\beta = 0,5$ i taką wartość przyjęto do dalszych eksperymentów.

Kolejnym rozpatrywanym parametrem był parametr q_0, który decyduje o wyborze między eksploatacją, a eksploracją. Rezultaty analizy prezentują tab. 6.6 oraz rys. 6.9. Podobnie jak w przypadku wyznaczania wartości parametru β, zarówno największa średnia liczność zbioru rozwiązań niezdominowanych, jak i najniższa wartość metryki Hausdorffa została uzyskana dla tej samej wartości parametru $q_0 = 0,95$ i taką wartość przyjęto do dalszych eksperymentów.

Następnym badanym parametrem był współczynnik wyparowywania śladu feromonowego ρ. Rezultaty analizy prezentują tab. 6.7 oraz rys. 6.10. Dla wartości parametru

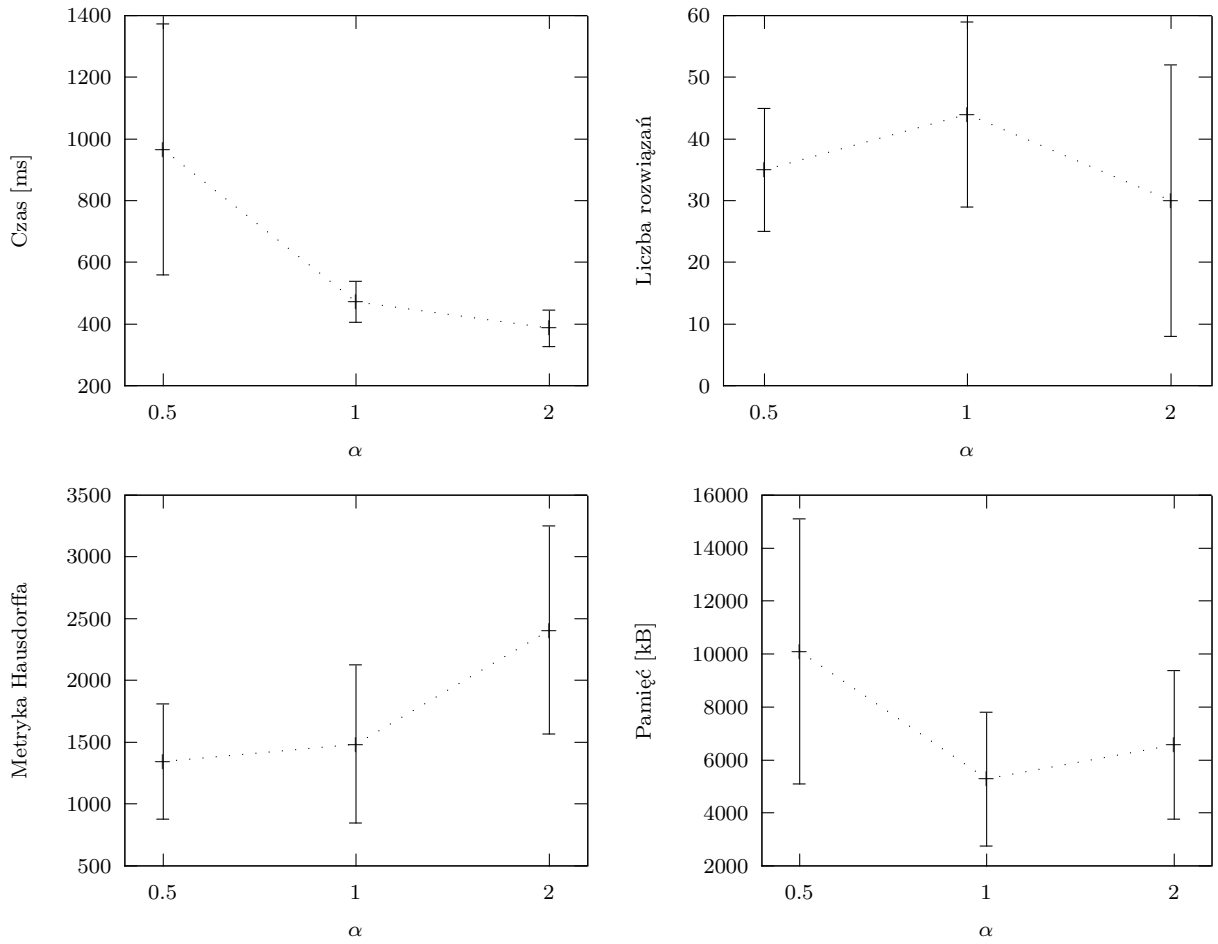


Rysunek 6.6: Wyniki działania algorytmu MultiNAVN-Z na mapie KAT-CEN podczas wyznaczania wartości parametru τ_0

$\rho = 0,2$ uzyskano najmniejszą wartość metryki Hausdorffa oraz prawie najwyższą średnią liczność zbioru rozwiązań niezdominowanych. Dlatego taką wartość parametru ρ przyjęto do dalszych eksperymentów.

Kolejnym badanym parametrem był współczynnik zmiany feromonu na ślepej krawędzi ω_b. Rezultaty analizy prezentują tab. 6.8 oraz rys. 6.11. Dla wartości parametru $\omega_b = 0,2$ uzyskano najwyższą średnią liczność zbioru rozwiązań niezdominowanych oraz prawie najniższą wartość metryki Hausdorffa i dlatego do dalszych eksperymentów przyjęto taką wartość.

Ostatnim badanym parametrem był współczynnik kary ω_p. Rezultaty analizy prezentują tab. 6.9 oraz rys. 6.12. Dla wartości parametru $\omega_p = 0,95$ uzyskano drugą w kolejności średnią liczność zbioru rozwiązań niezdominowanych oraz drugą w kolejności średnią metrykę Hausdorffa. Dlatego taka wartość parametru została uznana za optymalną i przyjęta do dalszych eksperymentów.



Rysunek 6.7: Wyniki działania algorytmu MultiNAVN-Z na mapie KAT-CEN podczas wyznaczania wartości parametru α

Podsumowując, w wyniku dostrajania wartości parametrów algorytmu MultiNAVN-Z ustalono następujące ich wartości:

- $\tau_0 = 0,0001$,
- $\alpha = 1$,
- $\beta = 0,5$,
- $q_0 = 0,95$,
- $\rho = 0,2$,
- $\omega_b = 0,2$,
- $\omega_p = 0,95$.

Tabela 6.5: Wyniki działania algorytmu MultiNAVN-Z na mapie KAT-CEN podczas wyznaczania wartości parametru β

β	**Czas [ms]**	**Liczba rozwiązań**	**Metryka H.**	**Pamięć [kB]**
0,5	**402,32**	**78,50**	**1243,46**	**7418,36**
1	436,14	67,18	1474,37	4948,86
2	482,50	56,39	1297,32	6400,61
3	774,11	41,25	1567,94	5309,11
5	951,57	19,07	1629,12	5147,32

Tabela 6.6: Wyniki działania algorytmu MultiNAVN-Z na mapie KAT-CEN podczas wyznaczania wartości parametru q_0

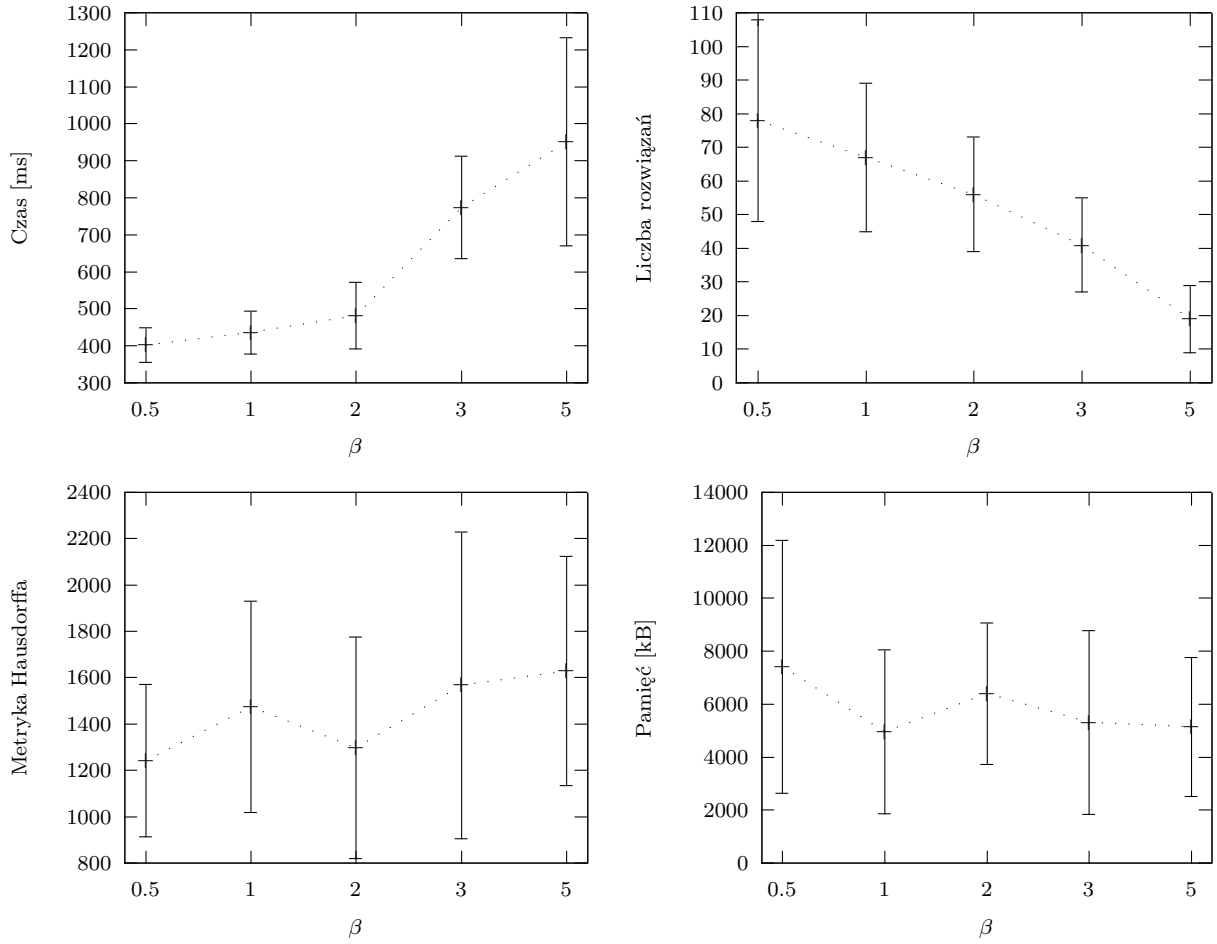
q_0	**Czas [ms]**	**Liczba rozwiązań**	**Metryka H.**	**Pamięć [kB]**
0,5	400,68	55,96	1787,58	8877,54
0,7	387,54	57,00	1509,24	5667,25
0,8	383,29	61,25	1459,27	3887,04
0,95	**421,82**	**77,36**	**1365,64**	**5403,79**

Tabela 6.7: Wyniki działania algorytmu MultiNAVN-Z na mapie KAT-CEN podczas wyznaczania wartości parametru ρ

ρ	**Czas [ms]**	**Liczba rozwiązań**	**Metryka H.**	**Pamięć [kB]**
0,1	384,54	64,54	1451,24	6353,79
0,2	**390,68**	**75,32**	**1411,24**	**5142,75**
0,3	433,11	76,32	1527,71	4833,50
0,4	472,21	67,96	1385,64	3752,54
0,5	565,43	75,14	1421,55	4861,96

Tabela 6.8: Wyniki działania algorytmu MultiNAVN-Z na mapie KAT-CEN podczas wyznaczania wartości parametru ω_b

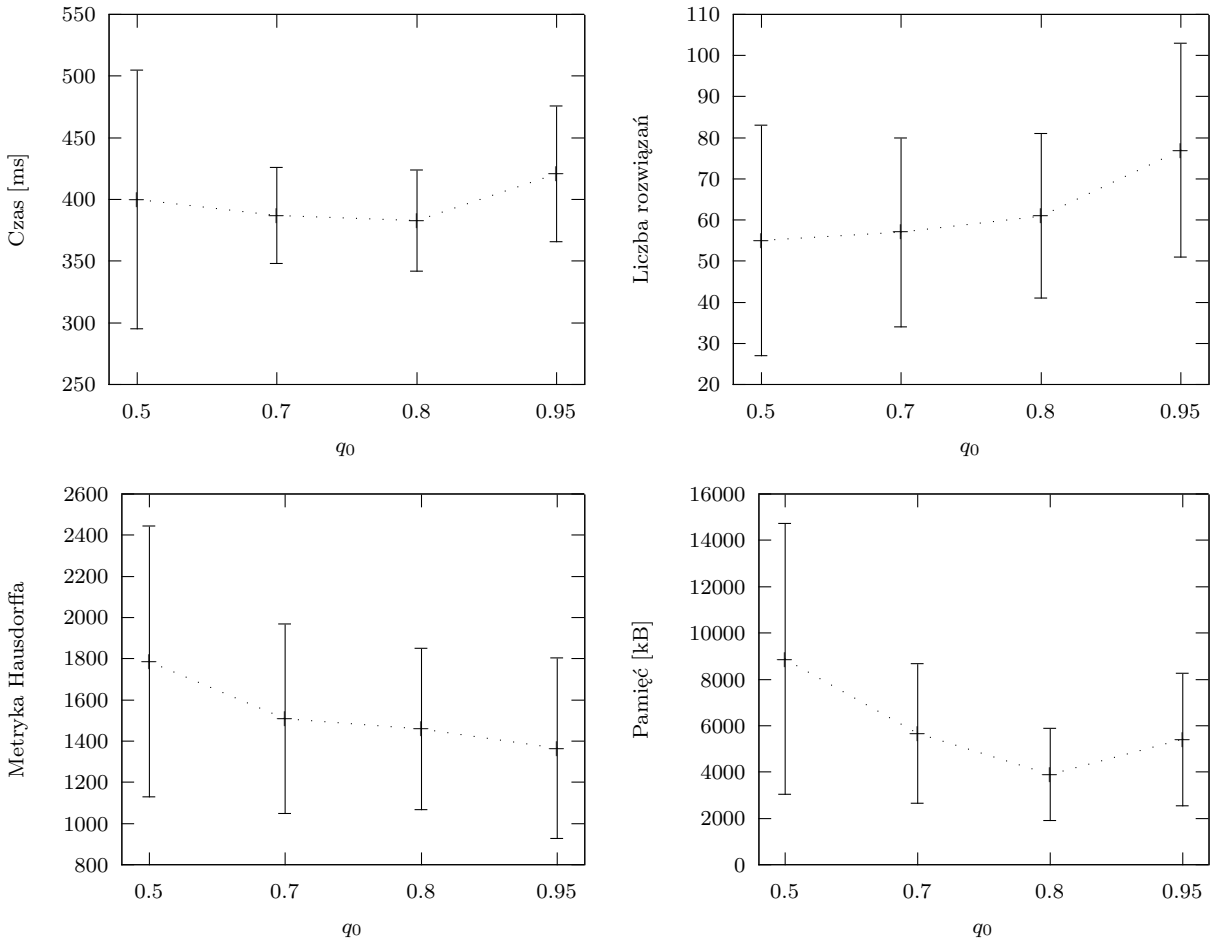
ω_b	**Czas [ms]**	**Liczba rozwiązań**	**Metryka H.**	**Pamięć [kB]**
0,05	415,79	64,39	1745,79	5775,93
0,1	388,57	70,82	1472,07	5263,75
0,15	421,18	73,89	1501,28	4622,18
0,20	**409,11**	**83,32**	**1385,87**	**5586,86**
0,25	444,29	76,11	1399,31	4720,21
0,30	534,79	76,36	1321,41	4902,39



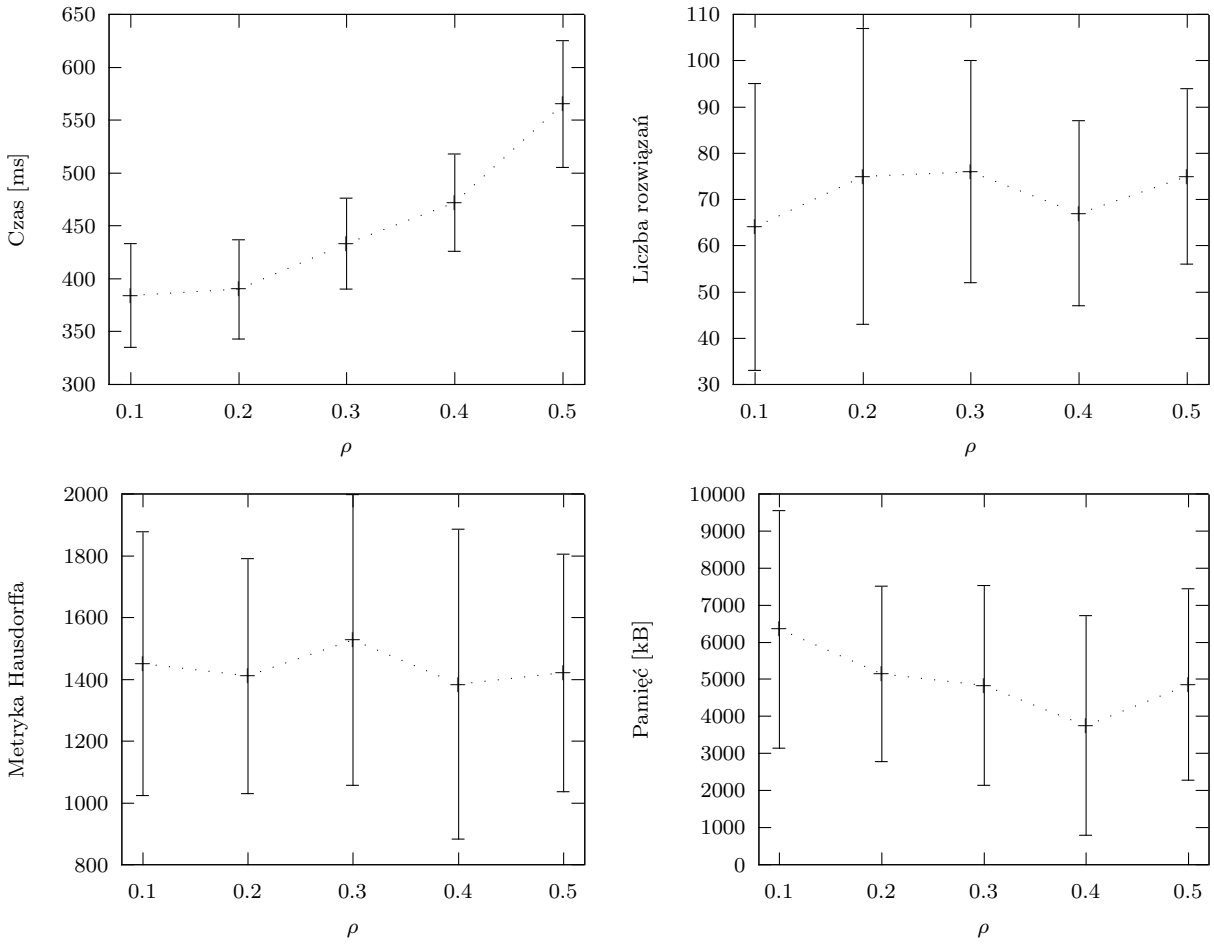
Rysunek 6.8: Wyniki działania algorytmu MultiNAVN-Z na mapie KAT-CEN podczas wyznaczania wartości parametru β

Tabela 6.9: Wyniki działania algorytmu MultiNAVN-Z na mapie KAT-CEN podczas wyznaczania wartości parametru ω_p

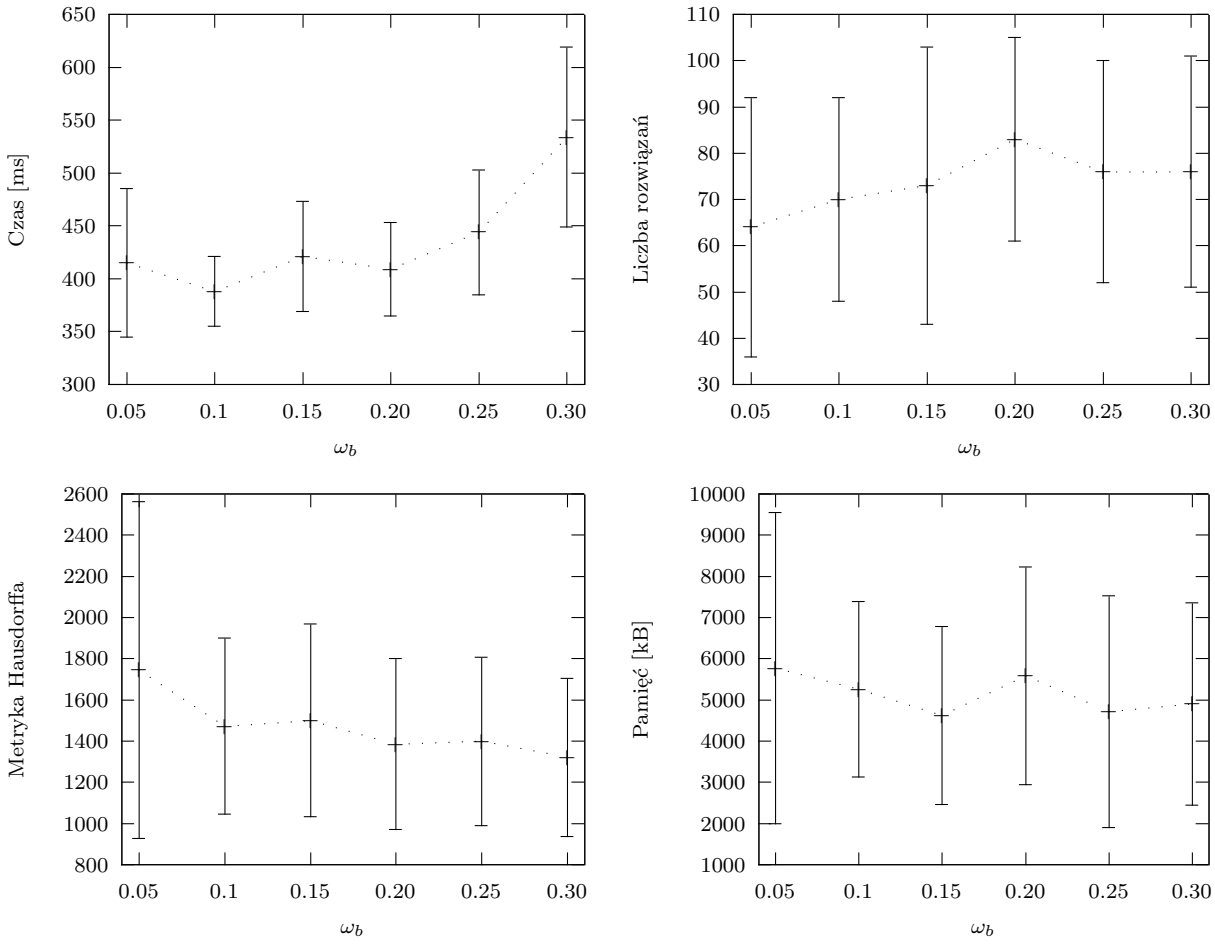
ω_p	**Czas [ms]**	**Liczba rozwiązań**	**Metryka H.**	**Pamięć [kB]**
0,80	492,32	28,07	1479,38	7220,54
0,85	456,57	49,86	1277,71	7901,71
0,90	458,18	84,00	1592,95	5247,39
0,95	**475,50**	**63,07**	**1337,63**	**6157,82**
0,99	480,61	60,93	1393,94	6851,86



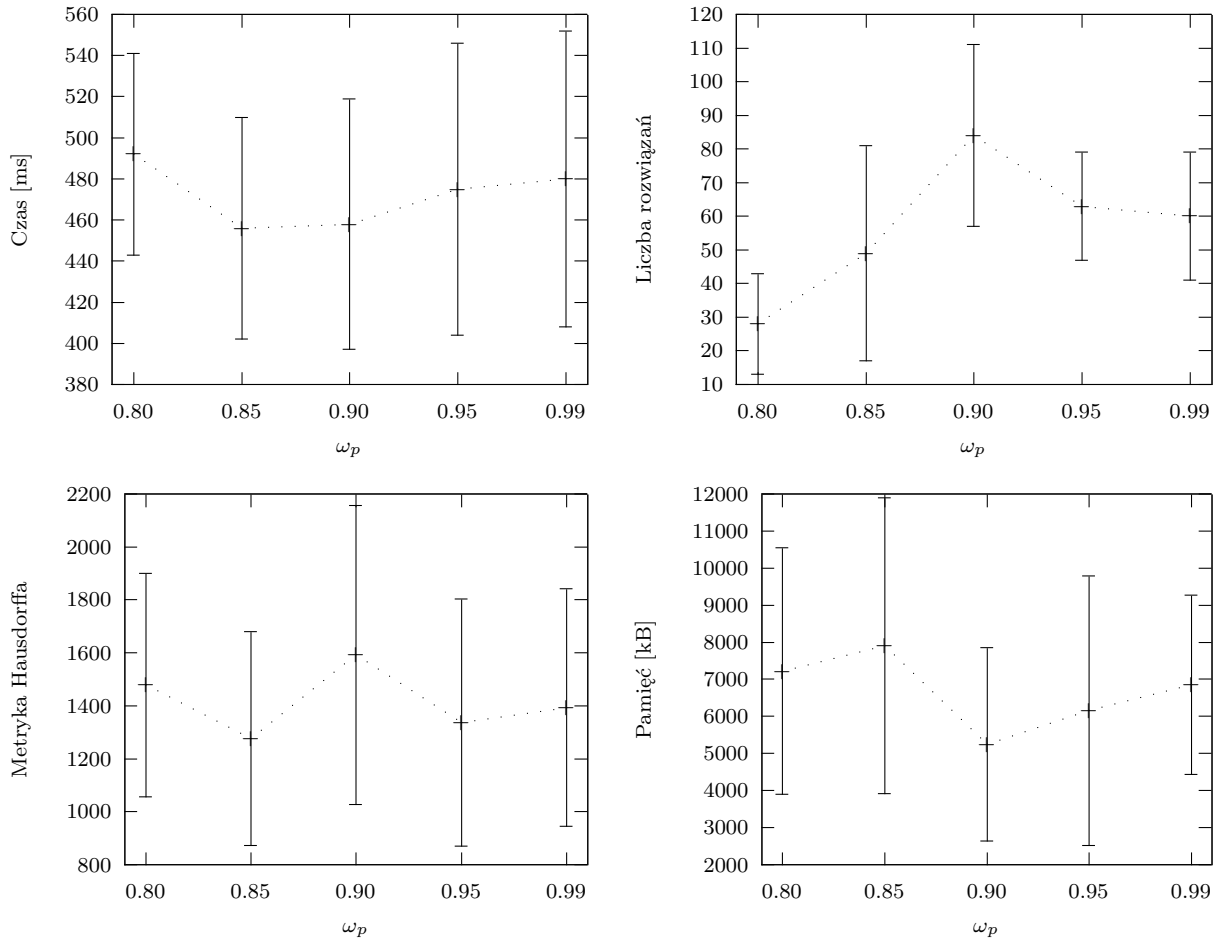
Rysunek 6.9: Wyniki działania algorytmu MultiNAVN-Z na mapie KAT-CEN podczas wyznaczania wartości parametru q_0



Rysunek 6.10: Wyniki działania algorytmu MultiNAVN-Z na mapie KAT-CEN podczas wyznaczania wartości parametru ρ



Rysunek 6.11: Wyniki działania algorytmu MultiNAVN-Z na mapie KAT-CEN podczas wyznaczania wartości parametru ω_b



Rysunek 6.12: Wyniki działania algorytmu MultiNAVN-Z na mapie KAT-CEN podczas wyznaczania wartości parametru ω_p

6.3.2 Algorytm MultiNAVN-L z kryterium wybieranym losowo

W drugiej kolejności, wyznaczanie wartości parametrów algorytmów mrowiskowych, przeprowadzono dla wersji algorytmu MultiNAVN-L, która w sposób losowy wybiera kryterium optymalizacji podczas obliczania prawdopodobieństwa przejścia mrówki między węzłami.

Pierwszym badanym parametrem była początkowa wartość śladu feromonowego τ_0. Rezultaty tych eksperymentów prezentują tab. 6.10 oraz rys. 6.13. Najlepsze wyniki uzyskano dla $\tau_0 = 0,001$ i taką wartość przyjęto do dalszych eksperymentów. Dla tej wartości parametru τ uzyskano największą średnią liczność zbioru rozwiązań niezdominowanych oraz drugą w kolejności wartość metryki Hausdorffa.

Tabela 6.10: Wyniki działania algorytmu MultiNAVN-L na mapie KAT-CEN podczas wyznaczania wartości parametru τ_0

τ_0	**Czas [ms]**	**Liczba rozwiązań**	**Metryka H.**	**Pamięć [kB]**
0,000001	918,04	37,32	3817,48	9369,43
0,00001	850,14	51,71	2586,51	9126,96
0,0001	1300,57	54,36	1978,08	11849,32
0,001	**1174,18**	**65,43**	**1452,85**	**10620,50**
0,01	1269,18	48,04	1347,31	9253,29

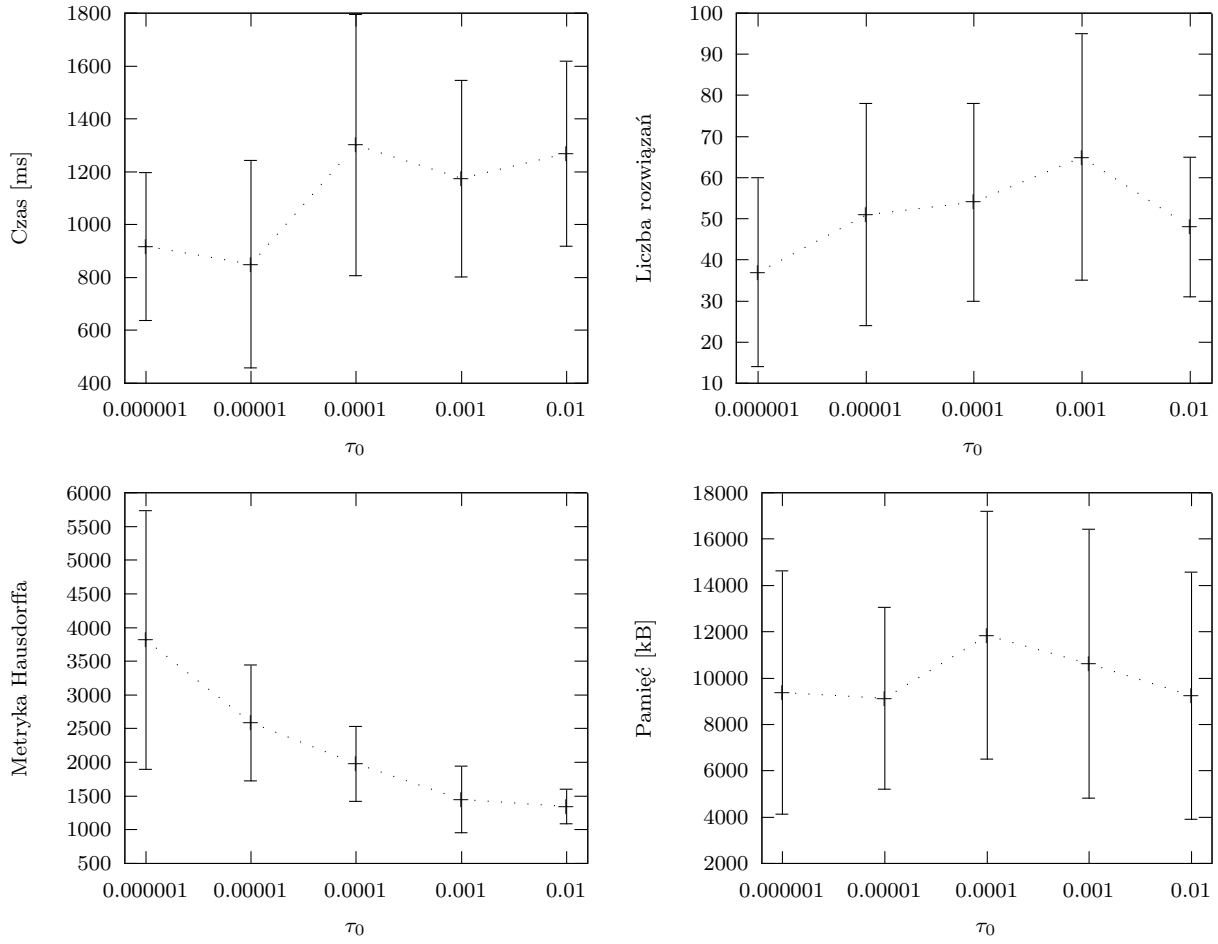
Następnie analizowano różne wartości parametru α oznaczającego wpływ śladu feromonowego na wynik reguły przejścia. Rezultaty analizy prezentują tab. 6.11 oraz rys. 6.14. Najlepsze wyniki uzyskano dla $\alpha = 1$ i taką wartość przyjęto do dalszych eksperymentów. Podobnie jak w przypadku wyznaczania wartości parametru τ wybrana została wartość parametru dla której uzyskano największą średnią liczność zbioru rozwiązań niezdominowanych oraz niską wartość metryki Hausdorffa.

Tabela 6.11: Wyniki działania algorytmu MultiNAVN-L na mapie KAT-CEN podczas wyznaczania wartości parametru α

α	**Czas [ms]**	**Liczba rozwiązań**	**Metryka H.**	**Pamięć [kB]**
0,5	2392,86	33,50	1384,83	17278,46
1	**1282,00**	**64,21**	**1478,65**	**9631,93**
2	542,00	39,25	2387,70	7843,54

Taką samą procedurę zastosowano do analizy parametru β, który określa wpływ informacji heurystycznej (widoczności) na wynik reguły przejścia. Rezultaty analizy prezentują tab. 6.12 oraz rys. 6.15. Zarówno największa średnia liczność zbioru rozwiązań niezdominowanych jak i najniższa wartość metryki Hausdorffa została uzyskana dla wartości parametru $\beta = 1$ i taką wartość przyjęto do dalszych eksperymentów.

Kolejnym rozpatrywanym parametrem był parametr q_0, który decyduje o wyborze między eksploatacją, a eksploracją. Rezultaty analizy prezentują tab. 6.13 oraz rys. 6.16. Dla wartości parametru $q_0 = 0,8$ uzyskano największą średnią liczność zbioru rozwiązań niezdominowanych oraz drugą w kolejności wartość metryki Hausdorffa. Dlatego taką wartość przyjęto do dalszych eksperymentów.



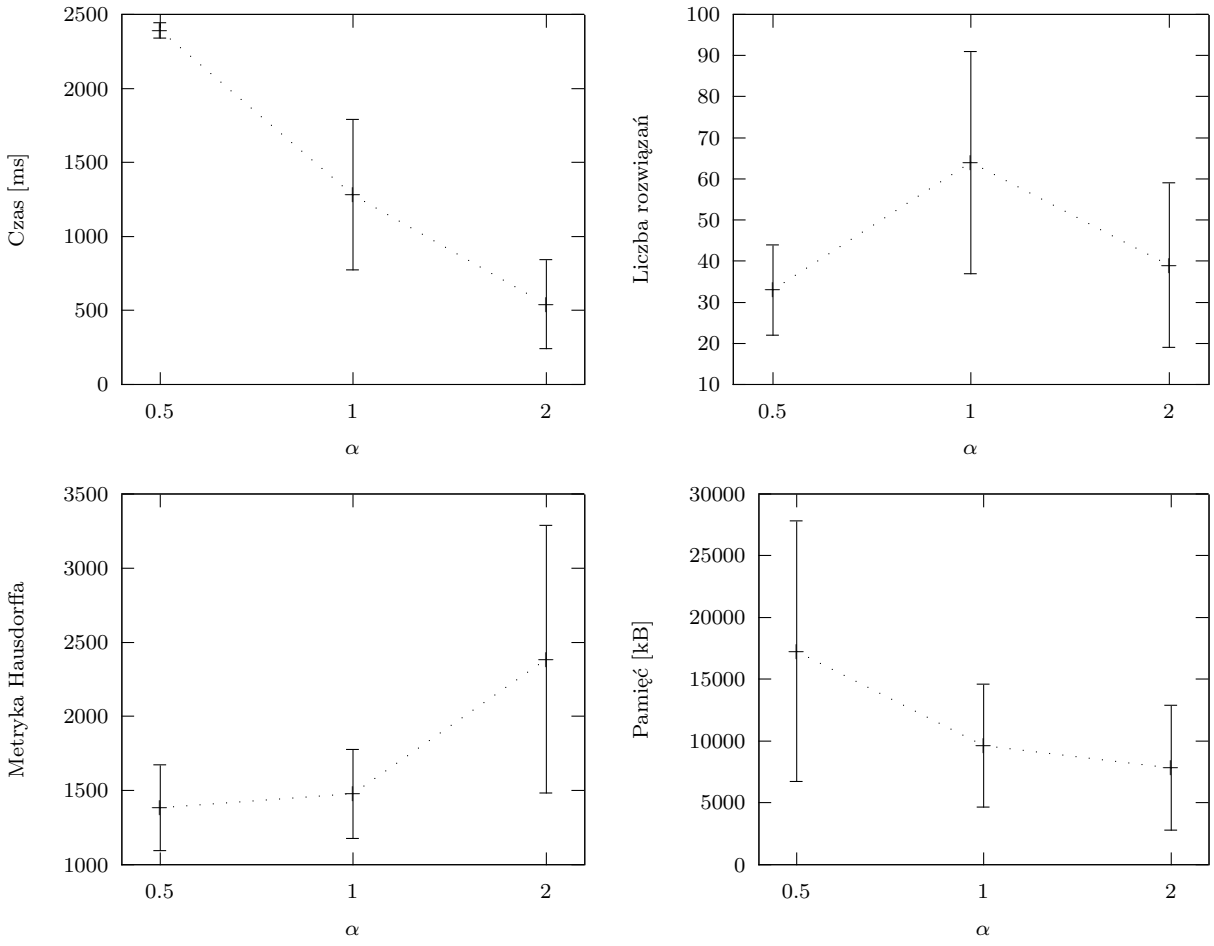
Rysunek 6.13: Wyniki działania algorytmu MultiNAVN-L na mapie KAT-CEN podczas wyznaczania wartości parametru τ_0

Tabela 6.12: Wyniki działania algorytmu MultiNAVN-L na mapie KAT-CEN podczas wyznaczania wartości parametru β

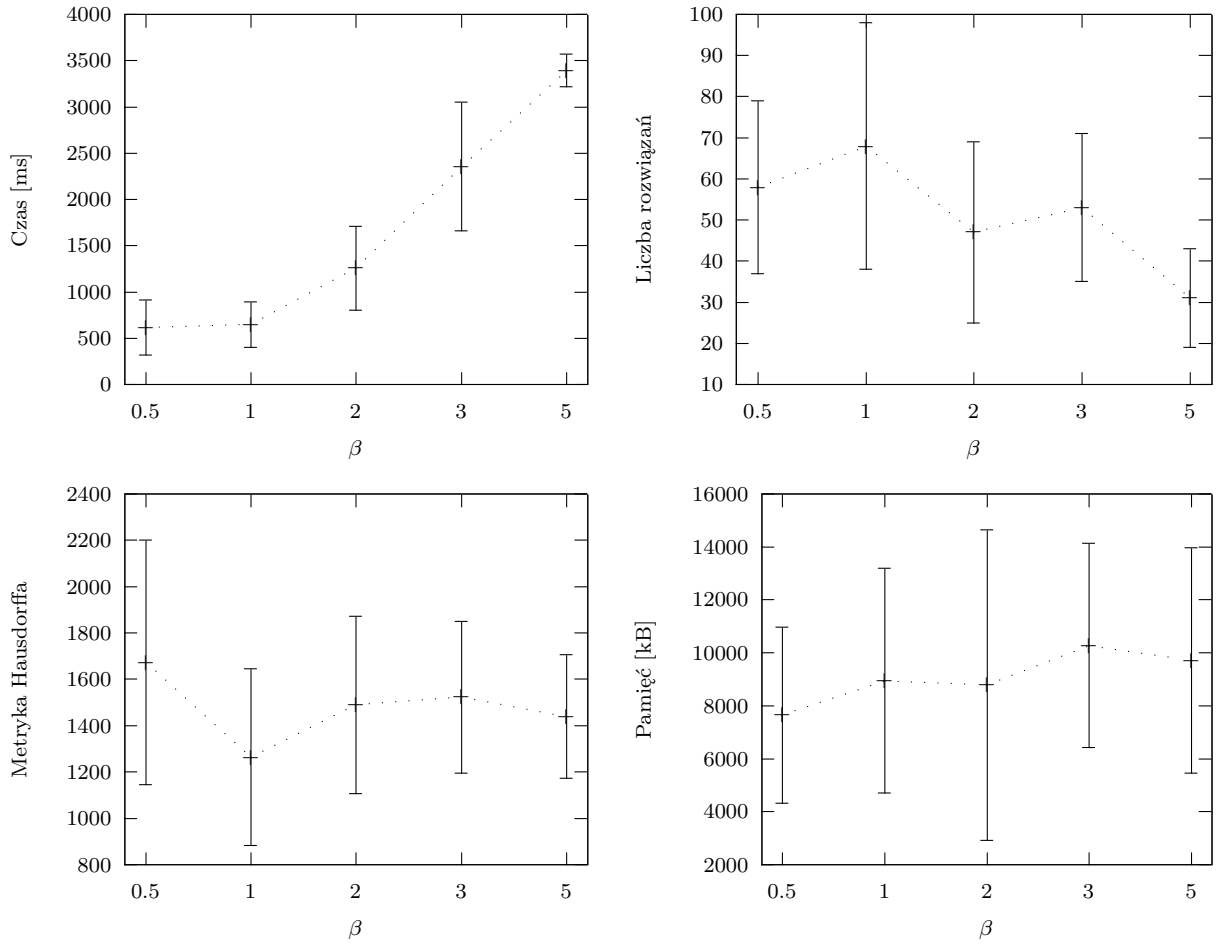
β	**Czas [ms]**	**Liczba rozwiązań**	**Metryka H.**	**Pamięć [kB]**
0,5	616,71	58,46	1674,18	7650,32
1	**649,21**	**68,75**	**1264,22**	**8944,36**
2	1257,07	47,89	1490,67	8787,43
3	2356,43	53,07	1523,38	10279,86
5	3392,79	31,18	1439,45	9722,82

Tabela 6.13: Wyniki działania algorytmu MultiNAVN-L na mapie KAT-CEN podczas wyznaczania wartości parametru q_0

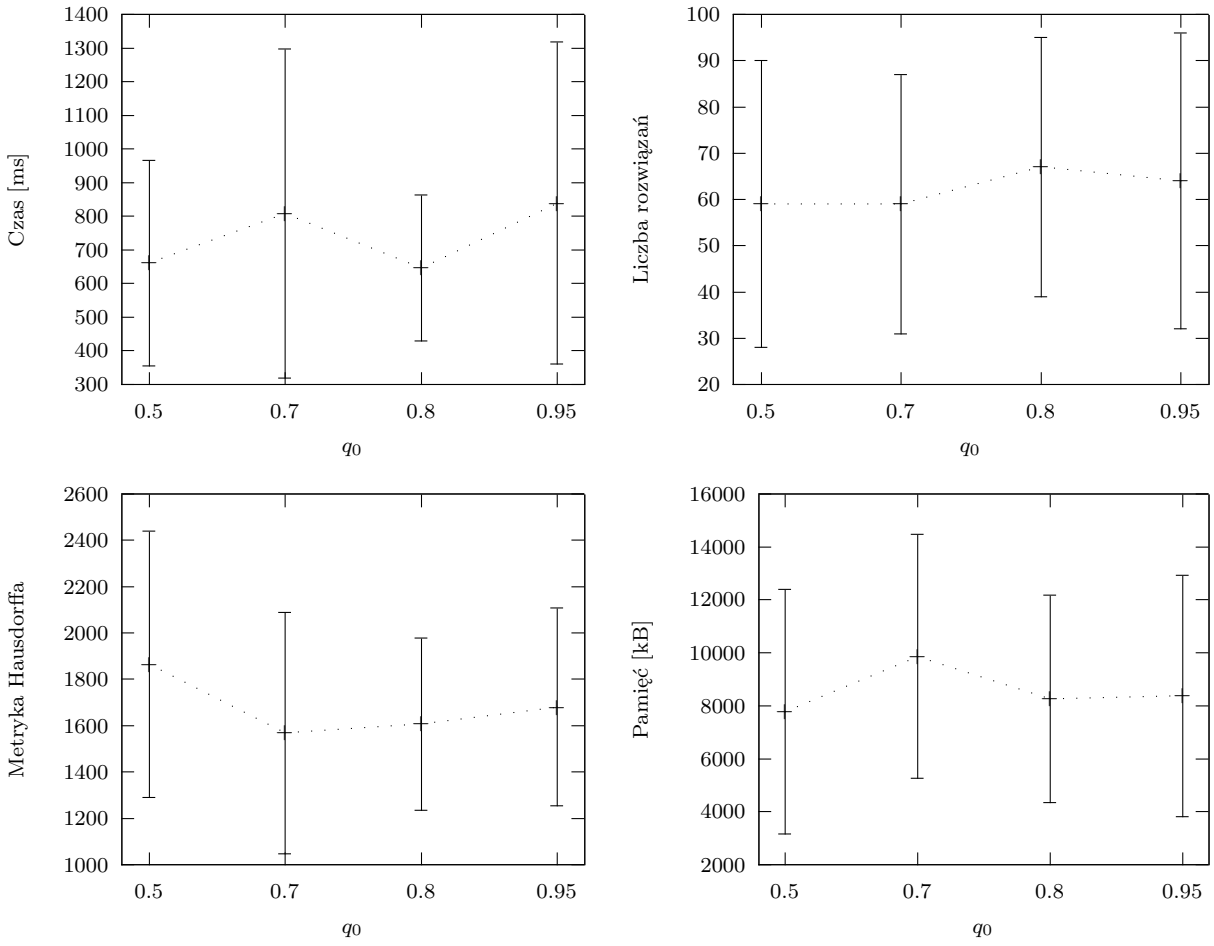
q_0	**Czas [ms]**	**Liczba rozwiązań**	**Metryka H.**	**Pamięć [kB]**
0,5	661,00	59,64	1866,32	7785,54
0,7	808,86	59,96	1568,49	9864,36
0,8	**647,25**	**67,21**	**1607,79**	**8271,82**
0,95	839,43	64,68	1681,54	8373,18



Rysunek 6.14: Wyniki działania algorytmu MultiNAVN-L na mapie KAT-CEN podczas wyznaczania wartości parametru α



Rysunek 6.15: Wyniki działania algorytmu MultiNAVN-L na mapie KAT-CEN podczas wyznaczania wartości parametru β



Rysunek 6.16: Wyniki działania algorytmu MultiNAVN-L na mapie KAT-CEN podczas wyznaczania wartości parametru q_0

Następnym badanym parametrem był współczynnik wyparowywania śladu feromonowego ρ. Rezultaty analizy prezentują tab. 6.14 oraz rys. 6.17. Dla wartości parametru $\rho = 0,3$ uzyskano najniższą wartość metryki Hausdorffa oraz prawie najwyższą średnią liczność zbioru rozwiązań niezdominowanych. Dlatego taką wartość parametru ρ przyjęto do dalszych eksperymentów.

Tabela 6.14: Wyniki działania algorytmu MultiNAVN-L na mapie KAT-CEN podczas wyznaczania wartości parametru ρ

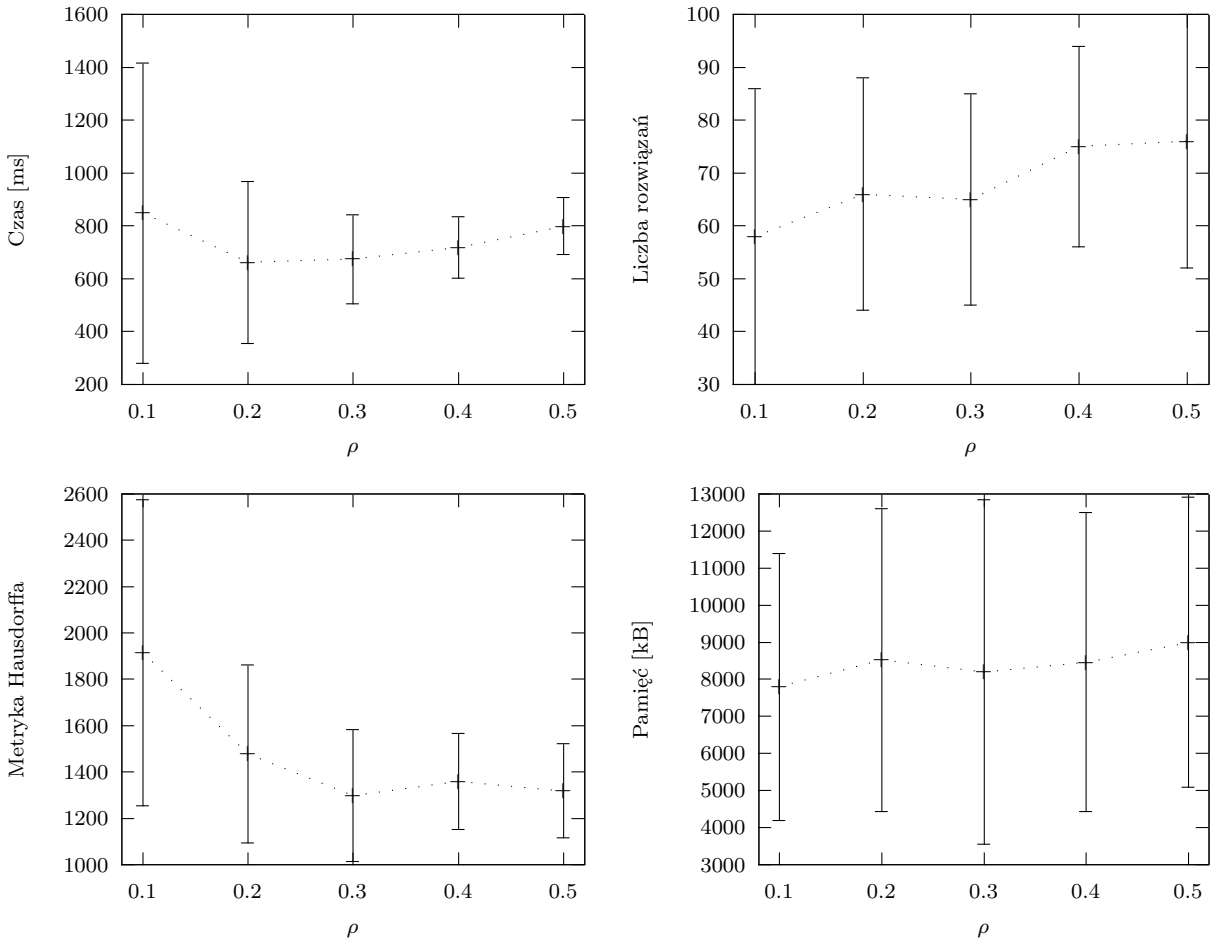
ρ	**Czas [ms]**	**Liczba rozwiązań**	**Metryka H.**	**Pamięć [kB]**
0,1	848,57	58,29	1914,35	7793,36
0,2	662,21	66,07	1478,46	8523,93
0,3	**674,79**	**65,79**	**1299,25**	**8199,11**
0,4	719,04	75,93	1360,69	8461,64
0,5	799,14	76,93	1319,74	8999,54

Kolejnym badanym parametrem był współczynnik zmiany feromonu na ślepej krawędzi, ω_b. Rezultaty analizy prezentują tab. 6.15 oraz rys. 6.18. Dla wartości parametru $\omega_b = 0,2$ uzyskano najwyższą średnią liczność zbioru rozwiązań niezdominowanych oraz prawie najniższą wartość metryki Hausdorffa i dlatego do dalszych eksperymentów przyjęto taką wartość.

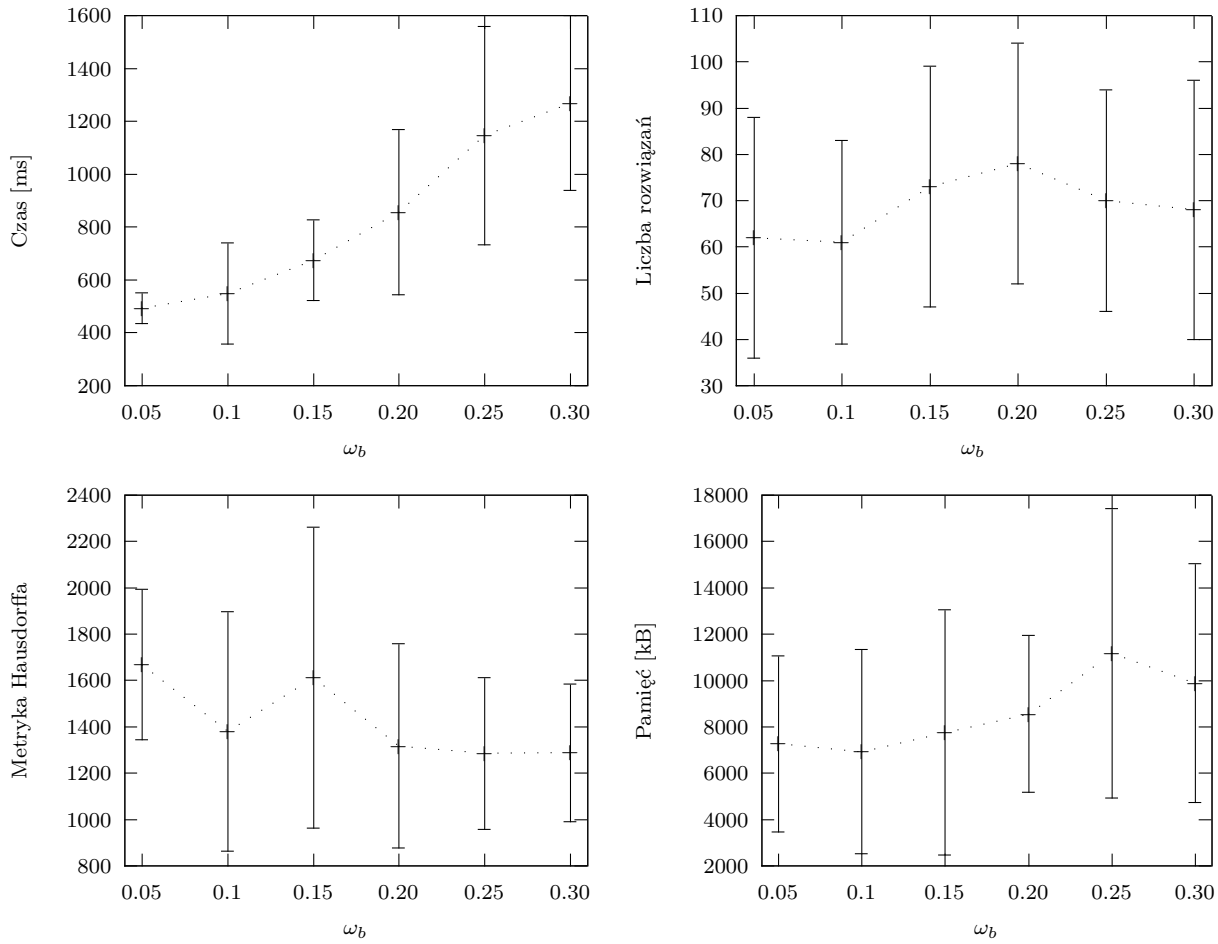
Tabela 6.15: Wyniki działania algorytmu MultiNAVN-L na mapie KAT-CEN podczas wyznaczania wartości parametru ω_b

ω_b	**Czas [ms]**	**Liczba rozwiązań**	**Metryka H.**	**Pamięć [kB]**
0,05	493,93	62,86	1669,38	7272,36
0,1	548,21	61,11	1381,32	6927,75
0,15	675,04	73,04	1613,27	7759,43
0,20	**856,07**	**78,36**	**1317,35**	**8554,25**
0,25	1147,04	70,43	1285,61	11180,04
0,30	1269,11	68,54	1288,86	9889,25

Ostatnim badanym parametrem był współczynnikiem kary ω_p. Rezultaty analizy prezentują tab. 6.16 oraz rys. 6.19. Dla wartości parametru $\omega_p = 0,95$ uzyskano drugą w kolejności średnią liczność zbioru rozwiązań niezdominowanych oraz najniższą średnią metrykę Hausdorffa. Dlatego taka wartość parametru została uznana za optymalną i przyjęta do dalszych eksperymentów.



Rysunek 6.17: Wyniki działania algorytmu MultiNAVN-L na mapie KAT-CEN podczas wyznaczania wartości parametru ρ



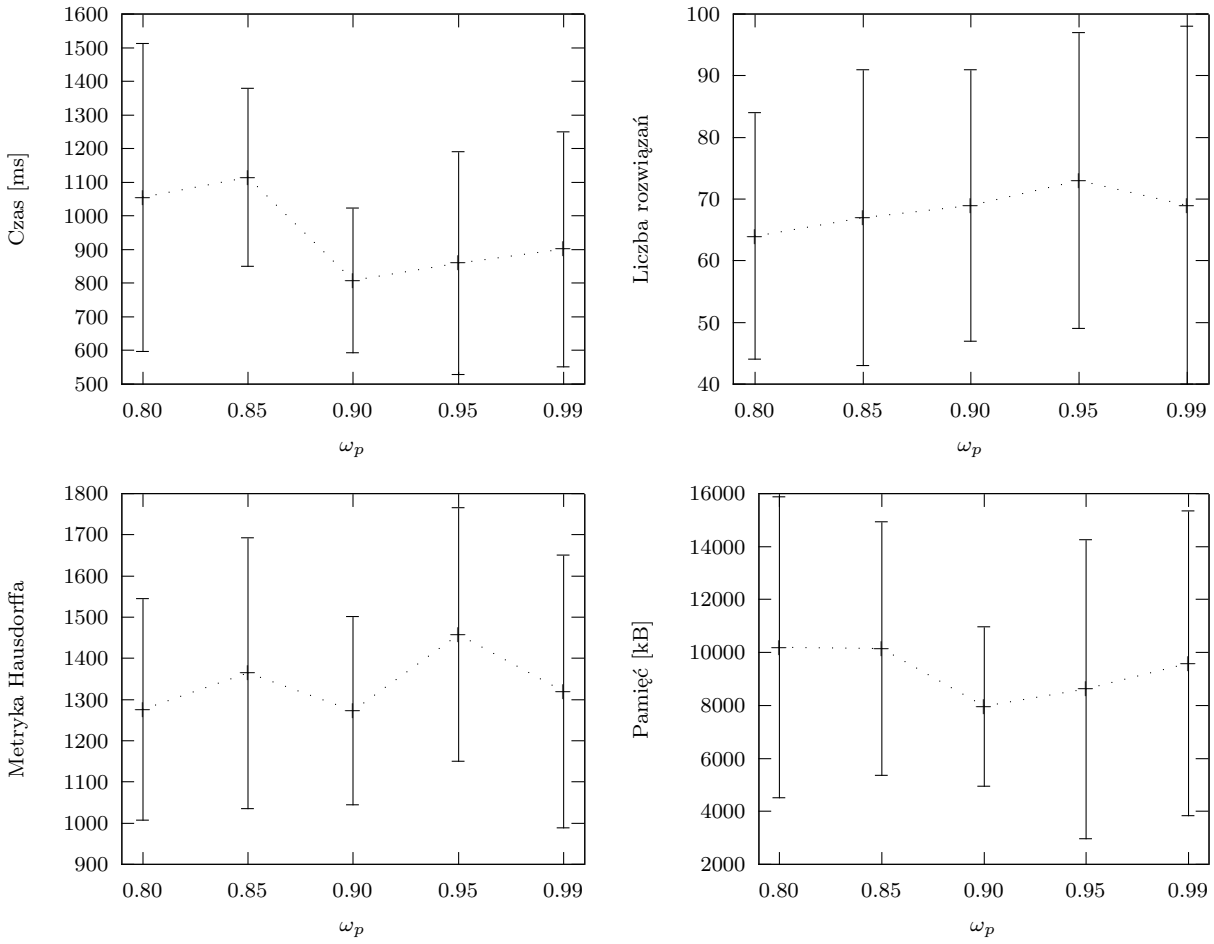
Rysunek 6.18: Wyniki działania algorytmu MultiNAVN-L na mapie KAT-CEN podczas wyznaczania wartości parametru ω_b

Podsumowując, w wyniku dostrajania wartości parametrów algorytmu MultiNAVN-L ustalono następujące ich wartości:

- $\tau_0 = 0,001$,
- $\alpha = 1$,
- $\beta = 1$,
- $q_0 = 0,8$,
- $\rho = 0,3$,
- $\omega_b = 0,2$,
- $\omega_p = 0,9$.

Tabela 6.16: Wyniki działania algorytmu MultiNAVN-L na mapie KAT-CEN podczas wyznaczania wartości parametru ω_p

ω_p	**Czas [ms]**	**Liczba rozwiązań**	**Metryka H.**	**Pamięć [kB]**
0,80	1055,25	64,04	1276,78	10199,29
0,85	1115,89	67,29	1364,06	10142,68
0,90	**808,79**	**69,50**	**1273,80**	**7961,14**
0,95	859,75	73,61	1458,18	8618,68
0,99	901,14	69,46	1320,23	9594,86



Rysunek 6.19: Wyniki działania algorytmu MultiNAVN-L na mapie KAT-CEN podczas wyznaczania wartości parametru ω_p

6.4 Dobór liczby mrówek

W poprzednim rozdziale zostały opisane eksperymenty w rezultacie których wyznaczone zostały wartości parametrów algorytmów mrowiskowych MultiNAVN-Z i MultiNAVN-L. Wartości tych parametrów zostały przyjęte w dalszych etapach eksperymentów.

Kolejnym etapem było ustalenie optymalnej liczby mrówek dla której algorytmy konstruują rozwiązania możliwie najlepszej jakości. Eksperymenty, których celem było ustalenie optymalnej liczby mrówek były wykonywane ze zwiększoną liczbą iteracji $N_{cykl} = 500$, tak aby zbyt mała wartość tego parametru nie ograniczała testowanego zakresu liczby agentów. Zbyt mała liczba iteracji mogłaby spowodować niską efektywność algorytmów dla wybranych wartości liczby mrówek.

Tabela 6.17: Wyniki działania algorytmu MultiNAVN-Z na mapie KAT-CEN podczas wyznaczania wartości parametru N_m

N_m	**Czas [ms]**	**Liczba rozwiązań**	**Metryka H.**	**Pamięć [kB]**
10	525,25	21,61	1828,17	5827,96
20	1097,04	55,79	1672,81	3869,64
40	2055,04	102,50	1375,76	4779,14
80	**3937,64**	**132,54**	**1456,02**	**10962,14**
160	8562,96	89,14	1362,60	20460,07
320	20425,18	72,50	1334,44	39725,82
640	39757,50	65,00	1311,87	306558,57

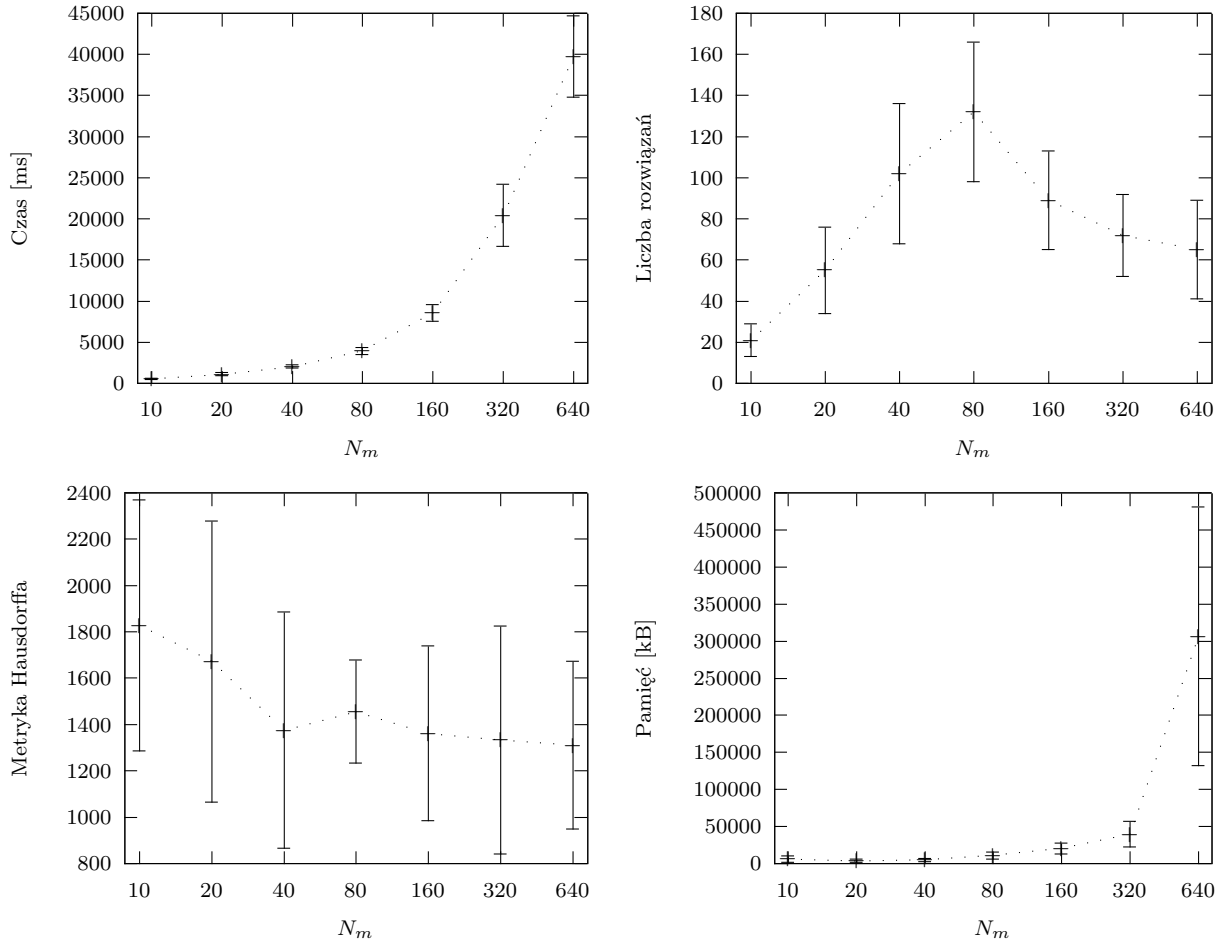
Tabela 6.18: Wyniki działania algorytmu MultiNAVN-L na mapie KAT-CEN podczas wyznaczania wartości parametru N_m

N_m	**Czas [ms]**	**Liczba rozwiązań**	**Metryka H.**	**Pamięć [kB]**
10	1634,68	28,86	2746,20	5797,04
20	2509,89	72,71	1661,98	7428,04
40	2551,21	109,21	1246,48	7858,07
80	**4976,79**	**147,36**	**1091,92**	**14160,29**
160	12456,50	74,36	1370,17	28574,75
320	25368,07	62,04	1555,61	328458,93
640	53021,25	66,68	1912,22	405615,79

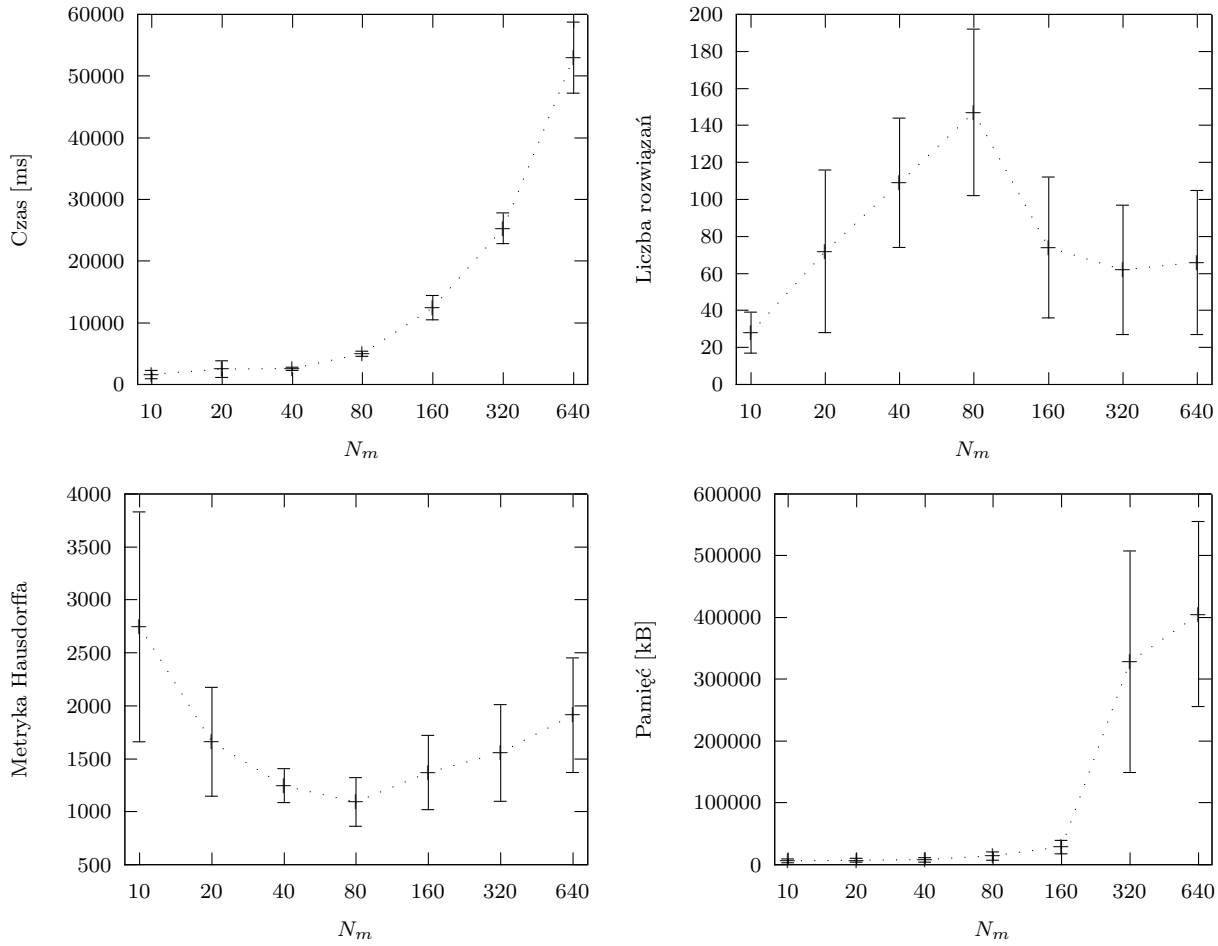
Podobnie jak w przypadku wcześniejszych eksperymentów, także w tym przypadku wykorzystano mapę oznaczoną jako KAT-CEN, a dla każdej testowanej wartości liczby agentów-mrówek eksperyment powtórzono 30 razy. Mierzono następujące wielkości: czas trwania eksperymentu, liczebność zbioru i metrykę Hausdorffa dla zbioru rozwiązań paretooptymalnych zwróconych przez algorytm oraz przyrost zaalokowanej pamięci operacyjnej w trakcie eksperymentu. Dla poszczególnych zbiorów z mierzonych wielkości obliczona została średnia arytmetyczna oraz odchylenie standardowe. W celu wyeliminowania grubych błędów z każdego zbioru usunięto wartości skrajne. Dodatkowo zastosowano zasadę 3σ i odrzucano obserwacje oddalone od wartości oczekiwanej o 3 wartości odchylenia standardowego.

W tab. 6.17 oraz na rys. 6.20 zaprezentowano wyniki eksperymentów dobierania liczby agentów dla algorytmu MultiNAVN-Z. Największy średni rozmiar zbioru rozwiązań paretooptymalnych (132,54) przy jednocześnie niskiej średniej wartości metryki Hausdorffa (1456,02) został uzyskany dla liczby mrówek $N_m = 80$ i taka wartość została przyjęta jako obowiązująca dla dalszych eksperymentów.

W tab. 6.18 oraz na rys. 6.21 zaprezentowano wyniki eksperymentów dobierania liczby agentów dla algorytmu MultiNAVN-L. Okazało się, że podobnie jak w przypadku algorytmu MultiNAVN-Z, największy średni rozmiar zbioru rozwiązań paretooptymalnych (147,36) przy jednocześnie najniższej średniej wartości metryki Hausdorffa (1091,92) został uzyskany dla liczby mrówek $N_m = 80$ i także w tym przypadku taka wartość została przyjęta jako obowiązująca dla dalszych eksperymentów.



Rysunek 6.20: Wyniki działania algorytmu MultiNAVN-Z na mapie KAT-CEN podczas wyznaczania wartości parametru N_m



Rysunek 6.21: Wyniki działania algorytmu MultiNAVN-L na mapie KAT-CEN podczas wyznaczania wartości parametru N_m

6.5 Porównanie efektywności badanych wersji algorytmów

Badania przedstawione w tym podrozdziale mają na celu stwierdzenie, czy różnice w konstrukcji poszczególnych wersji algorytmu MultiNAVN mają wpływ na jakość uzyskiwanych przez nie wyników oraz na czasy wyznaczania rozwiązań.

6.5.1 Statystyczna metoda porównywania wyników

Algorytmy optymalizacji mrowiskowej, do których zalicza się algorytm MultiNAVN, w swojej konstrukcji opierają się na elementach stochastycznych. Stochastyczny charakter badanych algorytmów powoduje, że każde wykonanie algorytmu na tym samym zestawie danych wejściowych oraz z wykorzystaniem tych samych wartości parametrów może dać inne rozwiązania. W związku z tym nie będzie ważny wynik pojedynczego wykonania badanego algorytmu lecz wyniki uśrednione (wartości oczekiwane oznaczane jako μ) będące wynikiem wielu kolejnych wykonań algorytmu na tych samych danych i w takich samych warunkach. W celu stwierdzenia, która z wersji badanych algorytmów (A_1, A_2) daje wyniki lepszej jakości lub daje je w krótszym czasie, można sprawdzić za pomocą metod statystycznych, badając czy poszczególne wartości oczekiwane (μ_1, μ_2) dla tych wersji algorytmów są istotnie różne. W rozprawie zastosowano test dla porównania *dwóch wartości średnich*, zwany również testem t Welcha.

Na podstawie dwóch niezależnych prób losowych o licznościach n_1 i n_2 należy sprawdzić hipotezę zerową $H_0 : \mu_1 - \mu_2 = D_0$, jako hipotezę alternatywną przyjmując hipotezę $H_1 : \mu_1 - \mu_2 \neq D_0$. Dla dużej liczebności n_1 i n_2 dwóch prób statystycznych (co najmniej kilkadziesiąt elementów), estymator $\bar{x}_1 - \bar{x}_2$ ma w przybliżeniu rozkład normalny. Po obliczeniu wartości średnich $\bar{x}_1$ i $\bar{x}_2$ oraz odchyleń standardowych s_1 i s_2 należy obliczyć wartość statystyki Z korzystając ze wzoru [92]:

$$Z = \frac{(\bar{x}_1 - \bar{x}_2) - D_0}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}, \tag{6.3}$$

a następnie dla poziomu ufności α odczytać z tablicy rozkładu normalnego $N(0, 1)$ wartość krytyczną statystyki Z_α. Gdy $|Z| \geqslant Z_\alpha$ hipotezę zerową należy odrzucić, a gdy $|Z| < Z_\alpha$ – nie ma podstaw do odrzucenia hipotezy zerowej. Najczęściej stosowane są wartości poziomu ufności $\alpha = 0{,}05$ oraz $\alpha = 0{,}01$.

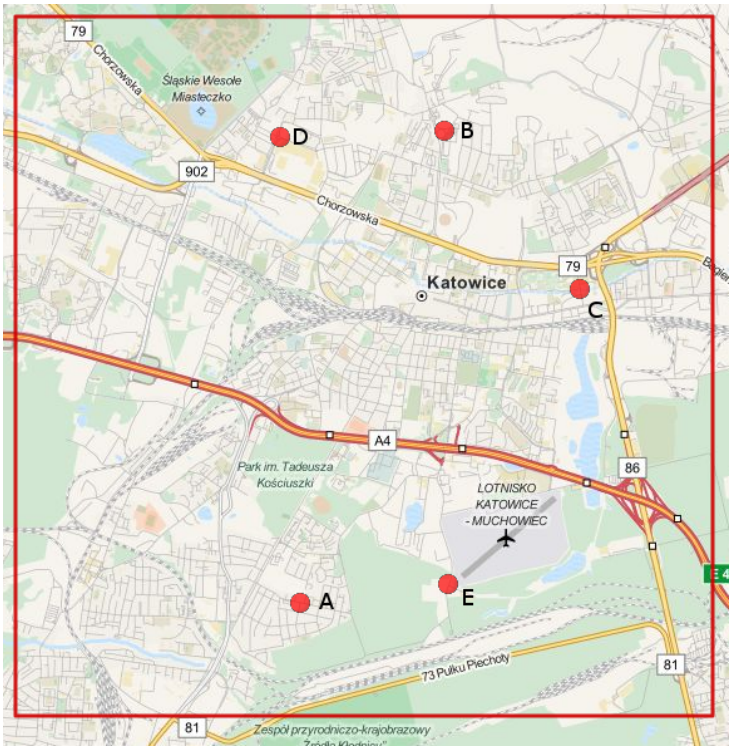
6.5.2 Analiza porównawcza wyników uzyskanych na mapie KATCEN

Celem eksperymentu jest sprawdzenie, która wersja mrowiskowego algorytmu nawigacji samochodowej (MultiNAVN-Z lub MultiNAVN-L), uzyskuje rozwiązania o lepszej jakości. Podobnie jak w doświadczeniach z podrozdziału 6.3 podstawową miarą jakości będzie średnia odległość zbioru rozwiązań paretooptymalnych wyznaczonych przez badany algorytm aproksymujący od pełnego zbioru rozwiązań wyznaczonego przez algorytm Label Setting. Dodatkowo porównane zostały liczności wyznaczonych zbiorów rozwiązań, zużycie pamięci oraz czas działania. Do eksperymentów wykorzystano mapę oznaczoną

jako KAT-CEN, której parametry zostały zaprezentowane wcześniej w tabeli 6.1. W obrębie tej mapy wytypowano 5 różnych punktów, które zostały przedstawione w tabeli 6.19 i na rysunku 6.22. Kolumny tej tabeli przedstawiają kolejno: identyfikator węzła OSM dla wybranego punktu, jego współrzędne geograficzne oraz opisowe położenie punktu. Z punktów tych utworzono 10 różnych dróg, które zostały zaprezentowane w tabeli 6.20, wraz z licznościami pełnych zbiorów rozwiązań paretooptymalnych wyznaczonych przez algorytm Label Setting.

Tabela 6.19: Punkty na mapie KAT-CEN wykorzystane jako miejsca początkowe i końcowe dróg wykorzystanych do analizy porównawczej

Punkt OSM	**Współrzędne geograficzne**	**Opis miejsca**
732709037 (A)	50.2326°N 19.0055°E	ul.Drozdów
384912139 (B)	50.2739°N 19.0253°E	Pętla Słoneczna
475040742 (C)	50.2600°N 19.0438°E	ul.Bogucicka na wysokości Bulwarów Rawy
324291561 (D)	50.2733°N 19.0028°E	Rondo księdza Józefa Kani
837939423 (E)	50.2342°N 19.0258°E	ul.Francuska, początek Doliny Trzech Stawów



Rysunek 6.22: Punkty na mapie KAT-CEN wykorzystane jako miejsca początkowe i końcowe dróg wykorzystanych do analizy porównawczej

Wartości parametrów testowanych algorytmów mrowiskowych zostały ustalone na podstawie eksperymentów opisanych w podrozdziale 6.3. Dla obu wersji algorytmu Mul-

Tabela 6.20: drogi na mapie KAT-CEN użyte do analizy porównawczej

Nr drogi	Punkt startowy	Punkt docelowy	Liczba rozwiązań Pareto
1	732709037 (A)	384912139 (B)	1392
2	732709037 (A)	475040742 (C)	782
3	324291561 (D)	837939423 (E)	594
4	324291561 (D)	732709037 (A)	655
5	384912139 (B)	837939423 (E)	415
6	384912139 (B)	732709037 (A)	1466
7	837939423 (E)	324291561 (D)	596
8	837939423 (E)	475040742 (C)	51
9	475040742 (C)	324291561 (D)	51
10	475040742 (C)	837939423 (E)	13

tiNAVN przyjęto te same, uśrednione wartości parametrów, które zostały przedstawione w tabeli 6.21.

Tabela 6.21: Wartości parametrów algorytmów mrowiskowych użyte do analizy porównawczej

Parametr	Wartość
τ_0	0,0005
α	1
β	0,75
q_0	0,88
ρ	0,25
ω_b	0,2
ω_p	0,92
N_m	80
N_{cykl}	500

Dla każdej drogi i obu wersji algorytmu MultiNAVN wykonano 40 kolejnych powtórzeń eksperymentu, tak aby możliwe było statystyczne porównanie jakości otrzymywanych rozwiązań. Dla poszczególnych dróg i wersji algorytmów obliczone zostały średnie wartości metryki Hausdorffa ($\bar{x}_z$ i $\bar{x}_l$) dla otrzymywanych zbiorów oraz odchylenia standardowe z próby (s_z i s_l). Następnie korzystając ze wzoru 6.3 obliczona została wartość statystyki Z, na podstawie której sprawdzono hipotezę zerową $H_0 : \bar{x}_z - \bar{x}_l = D_0$, przyjmując wartość $D_0 = 0$ oraz wartość poziomu ufności $\alpha = 0{,}01$. Dla wybranego poziomu ufności wartość krytyczna odczytana z tablic rozkładu normalnego $N(0, 1)$ wynosi $Z_\alpha = 2{,}33$. We wszystkich tabelach prezentujących wyniki eksperymentów porównawczych wartości bezwzględne statystyki Z, dla których nie jest spełniona hipoteza zerowa oznaczono pogrubioną czcionką.

W tabeli 6.22 zaprezentowane są wyniki analizy porównawczej dwóch wersji algorytmu MulitNAVN: MultiNAVN-Z i MultiNAVN-L pod względem wartości metryki Hausdorffa. Hipoteza zerowa została odrzucona dla czterech z dziesięciu przebadanych dróg. Dla wszystkich tych przypadków średnia wartość metryki Hausdorffa została osiągnięta przez algorytm MultiNAVN-L. Wyniki wskazują na przewagę wersji MultiNAVN-L

pod względem jakości otrzymywanych zbiorów rozwiązań, której miarą jest podobieństwo do zbioru referencyjnego.

Tabela 6.22: Porównanie metryk Hausdorffa zbiorów rozwiązań wyznaczonych przez algorytmy MultiNAVN-Z i MultiNAVN-L na mapie KAT-CEN

	MultiNAVN-Z		MultiNAVN-L		
Nr drogi	$\bar{x}_z$	s_z	$\bar{x}_l$	s_l	$\|Z\|$
1	1214,95	367,19	1190,54	237,42	0,353
2	799,77	344,54	812,50	219,58	0,197
3	813,67	291,58	1048,17	634,77	2,123
4	2388,78	930,97	**1639,51**	871,04	**3,717**
5	754,22	251,36	685,52	246,83	1,233
6	1726,94	595,56	**1379,36**	275,54	**3,350**
7	2071,77	788,19	2162,84	708,29	0,544
8	2840,67	804,27	3068,18	374,85	1,622
9	949,12	490,47	**476,72**	296,60	**5,213**
10	277,66	334,37	**128,98**	50,51	**2,781**

W ten sam sposób zostały porównane średnie liczności rozwiązań niezdominowanych wyznaczonych przez obie wersje algorytmu. Wyniki zostały przedstawione w tabeli 6.23 i jeszcze bardziej jednoznacznie wskazują na lepszą efektywność algorytmu MultiNAVN-L. Można to stwierdzić na podstawie tego, że hipoteza zerowa została odrzucona dla wszystkich z wyjątkiem jednej przebadanej drogi i jednocześnie dla wszystkich przypadków większa średnia liczność zbiorów rozwiązań wyznaczona została przez wersję MultiNAVN-L.

Tabela 6.23: Porównanie liczności zbiorów rozwiązań wyznaczonych przez algorytmy MultiNAVN-Z i MultiNAVN-L na mapie KAT-CEN

	MultiNAVN-Z		MultiNAVN-L		
Nr drogi	$\bar{y}_z$	s_z	$\bar{y}_l$	s_l	$\|Z\|$
1	41,39	14,29	**176,42**	47,85	**17,101**
2	61,37	16,97	**134,55**	36,76	**11,431**
3	31,16	12,16	**67,08**	30,06	**7,006**
4	27,05	12,36	**79,11**	34,61	**8,959**
5	21,39	9,49	**44,18**	12,72	**9,082**
6	39,55	17,40	**97,42**	29,71	**10,630**
7	47,11	11,24	**106,66**	39,71	**9,126**
8	19,71	8,01	20,18	5,17	0,312
9	12,11	4,23	**29,66**	9,85	**10,354**
10	6,79	1,83	**9,53**	1,70	**6,938**

Przewaga wersji algorytmu MutiNAVN-L, tak wyraźna w przypadku porównania jakości rozwiązania, została przełamana w przypadku porównania średnich czasów wykonania, które zostały zaprezentowane w tabeli 6.24. W tym przypadku hipoteza zerowa została odrzucona dla ośmiu z dziesięciu dróg, ale w każdym z tych przypadków krótsze

czasy działania uzyskał algorytm MutiNAVN-Z. Obie wersje algorytmu były uruchamiane z tymi samymi wartościami parametrów (liczba mrówek, liczba iteracji), które mają bezpośredni wpływ na czasy wykonania. Można stąd wyciągnąć wniosek, że przewaga wersji MutiNAVN-Z pod względem czasu działania algorytmów wynika z faktu, że wersja MultiNAVN-L mocniej akcentuje eksplorację przestrzeni rozwiązań niż eksploatację dotychczas wyznaczonych rozwiązań.

Tabela 6.24: Porównanie czasu wykonania algorytmów MultiNAVN-Z i MultiNAVN-L na mapie KAT-CEN

	MultiNAVN-Z		MultiNAVN-L		
Nr drogi	$\bar{z}_z$ [ms]	s_z	$\bar{z}_l$ [ms]	s_l	$\|Z\|$
1	6027,24	850,66	6165,39	358,31	0,947
2	4899,74	747,10	4837,08	309,35	0,490
3	**5458,13**	1057,88	10479,00	7534,36	**4,174**
4	**5621,66**	1018,02	7978,45	3800,77	**3,788**
5	**5498,37**	1154,46	7230,95	2805,16	**3,612**
6	**5860,55**	716,07	7152,84	2083,75	**3,709**
7	**5101,76**	445,27	8099,71	1784,89	**10,307**
8	**2405,79**	295,13	3379,58	302,66	**14,569**
9	**2727,42**	306,16	9622,58	5366,39	**8,113**
10	**2385,63**	129,86	10940,79	6281,24	**8,612**

Tabela 6.25: Porównanie zapotrzebowania na pamięć operacyjną oraz czasu działania algorytmów MultiNAVN-Z, MultiNAVN-L oraz Label Setting na mapie KAT-CEN

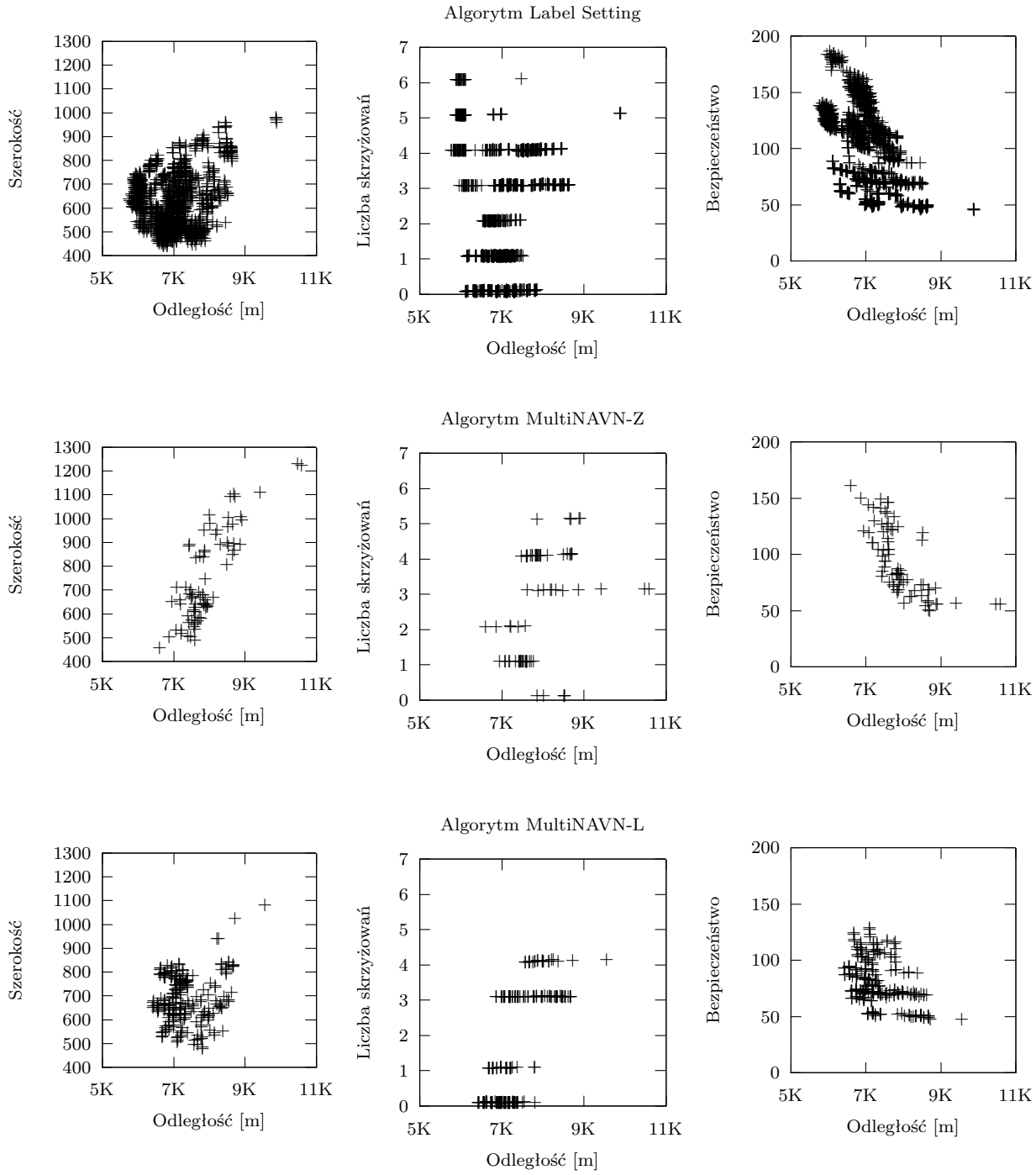
	MultiNAVN-Z		MultiNAVN-L		Label Setting	
Nr drogi	$\bar{m}_z$ [kB]	$\bar{z}_z$ [ms]	$\bar{m}_l$ [kB]	$\bar{z}_l$ [ms]	$\bar{m}_{ls}$ [kB]	$\bar{z}_{ls}$ [ms]
1	10618	6027	11602	6165	122865	74856
2	5831	4900	6579	4837	122874	73876
3	9736	5458	14060	10479	56842	14924
4	9515	5622	12322	7978	57171	14913
5	9448	5498	11948	7231	81895	38993
6	8864	5861	11924	7153	83627	38864
7	8302	5102	10015	8100	36701	5495
8	5634	2406	6468	3380	35753	5490
9	7922	2727	13148	9623	18620	1275
10	8658	2386	13471	10941	18127	1262

Tabela 6.25 prezentuje porównanie zapotrzebowania na pamięć operacyjną oraz czasu działania algorytmów MultiNAVN oraz Label Setting. Można w niej zobaczyć, że w przypadku większości dróg algorytm Label Setting działał zdecydowanie wolniej oraz wymagał do działania większej ilości pamięci operacyjnej.

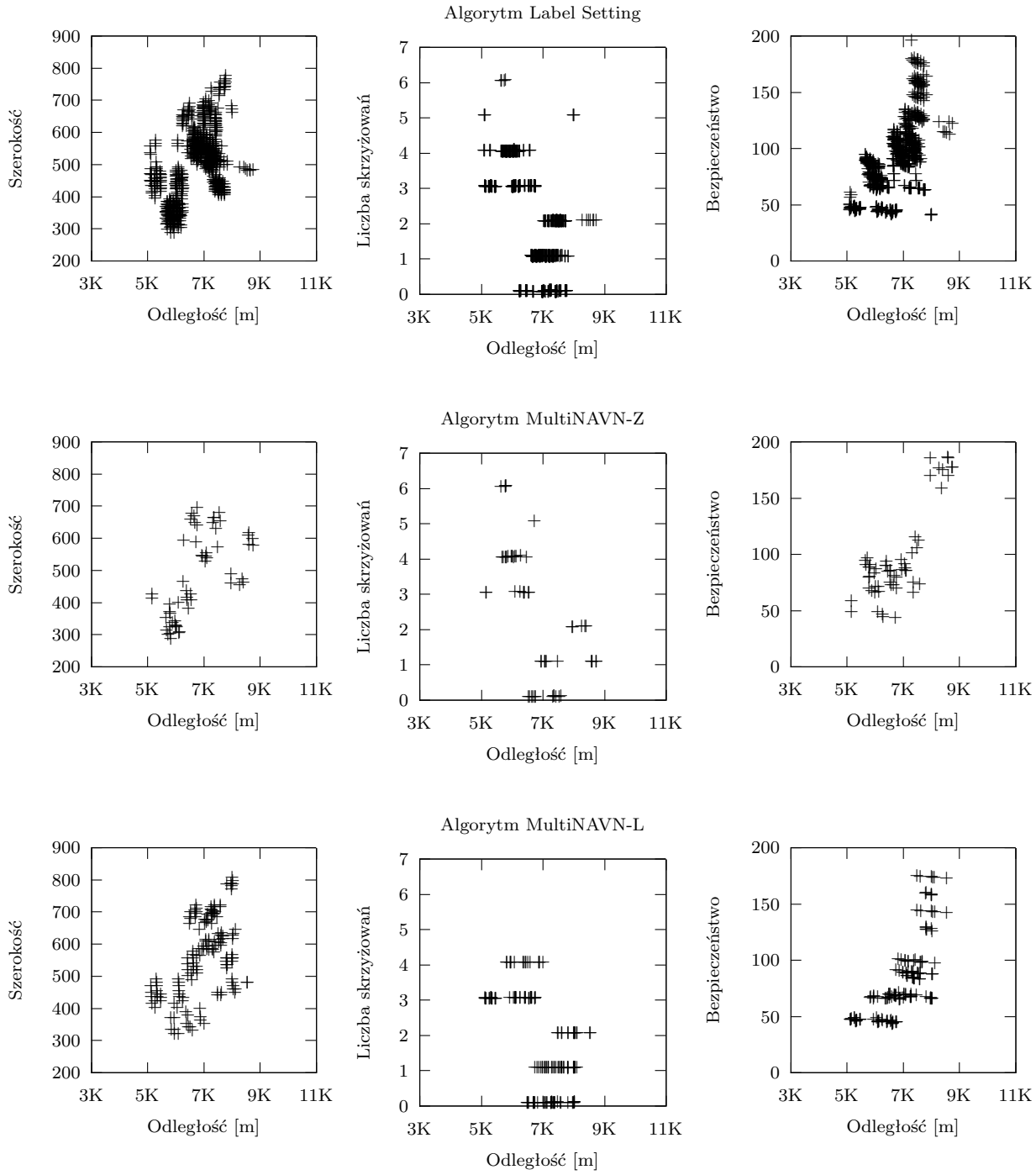
Na rysunkach od 6.23 do 6.32 zaprezentowano pełne zbiory rozwiązań, które zostały wyznaczone przez algorytm Label Setting, oraz wybrane aproksymacje zbiorów rozwiązań niezdominowanych wyznaczone przez algorytmy MultiNAVN-Z i MultiNAVN-L.

Wybrane zbiory rozwiązań charakteryzują się najmniejszą wartością metryki Hausdorffa spośród wszystkich uruchomień algorytmów. Dzięki temu możliwe jest porównanie jakości przybliżonych zbiorów rozwiązań dostarczanych przez algorytmy mrowiskowe z pełnymi zbiorami referencyjnymi.

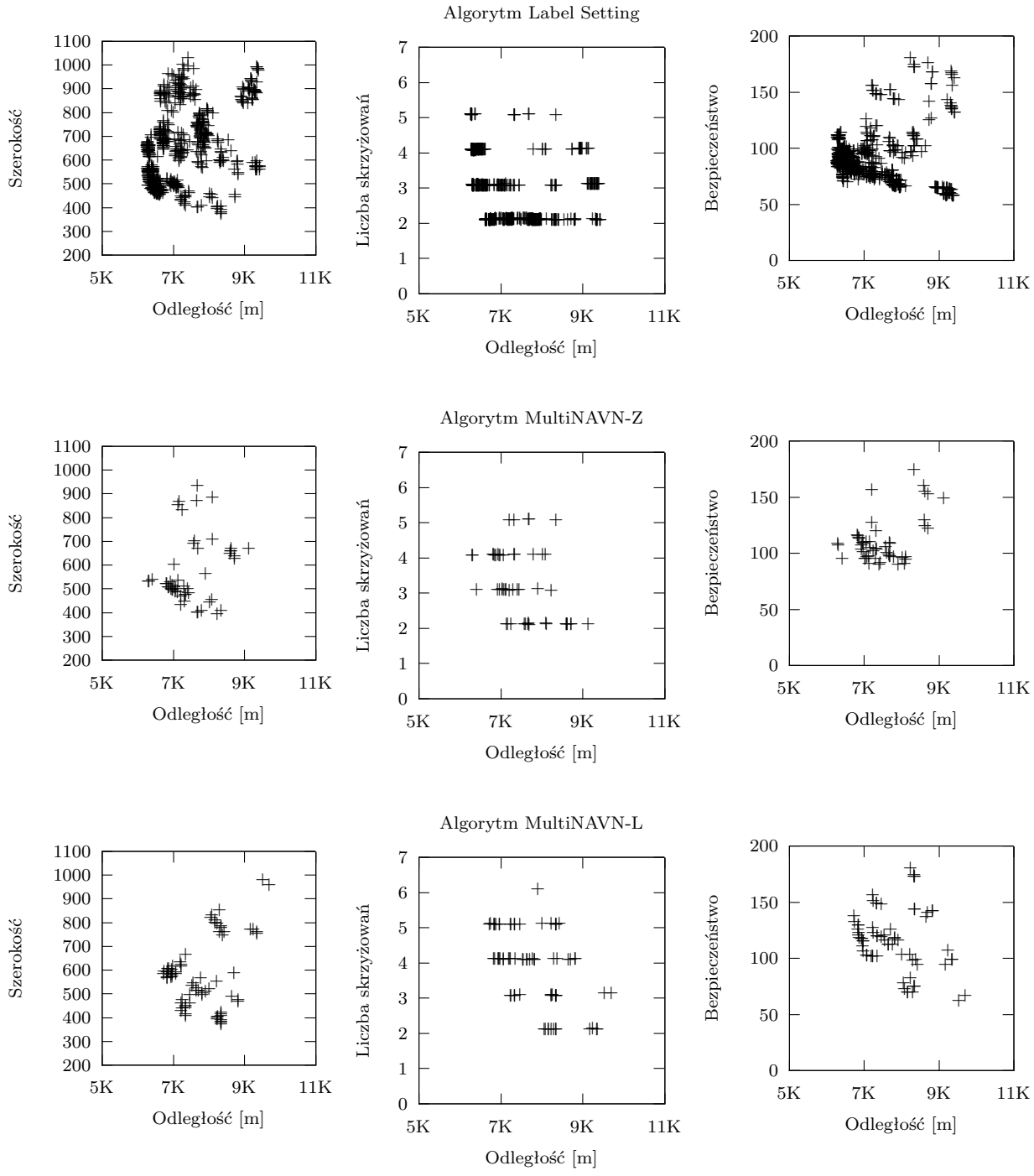
Na podstawie wyników analizy porównawczej przeprowadzonej na mapie KAT-CEN można stwierdzić, że wersja algorytmu MultiNAVN-L, wyznacza rozwiązania lepszej jakości, charakteryzujące się w stosunku do wersji MultiNAVN-Z średnio niższą wartością metryki Hausdorffa oraz wyższą licznością zbioru rozwiązań niezdominowanych. Jednocześnie należy zaznaczyć, że wersja MultiNAVN-L potrzebuje więcej czasu na wyznaczenie rozwiązań, ze względu na bardziej eksploracyjną charakterystykę obliczeń w stosunku do wersji MultiNAVN-Z.



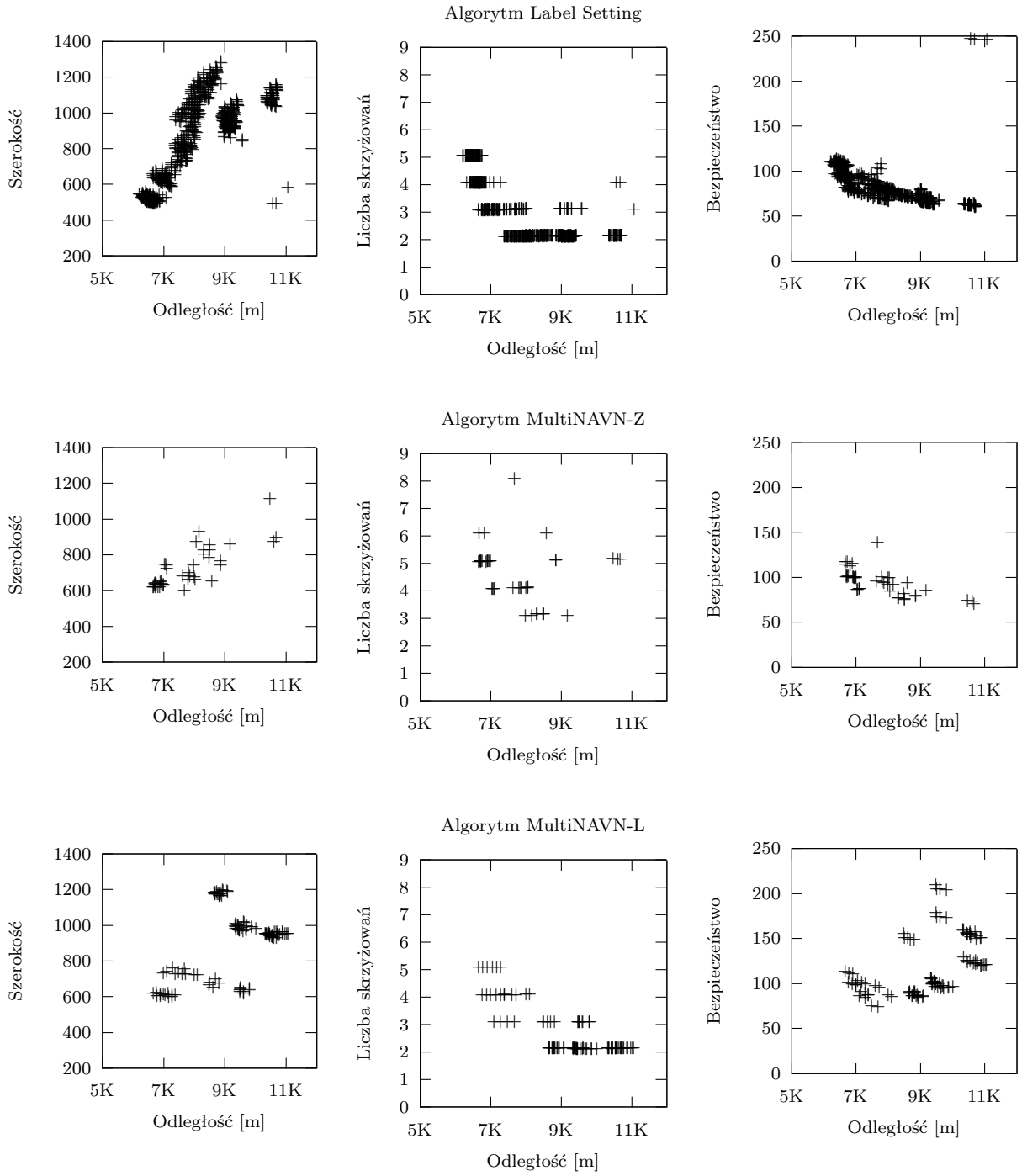
Rysunek 6.23: Zbiory rozwiązań wyznaczone przez porównywane algorytmy na mapie KAT-CEN dla drogi nr 01



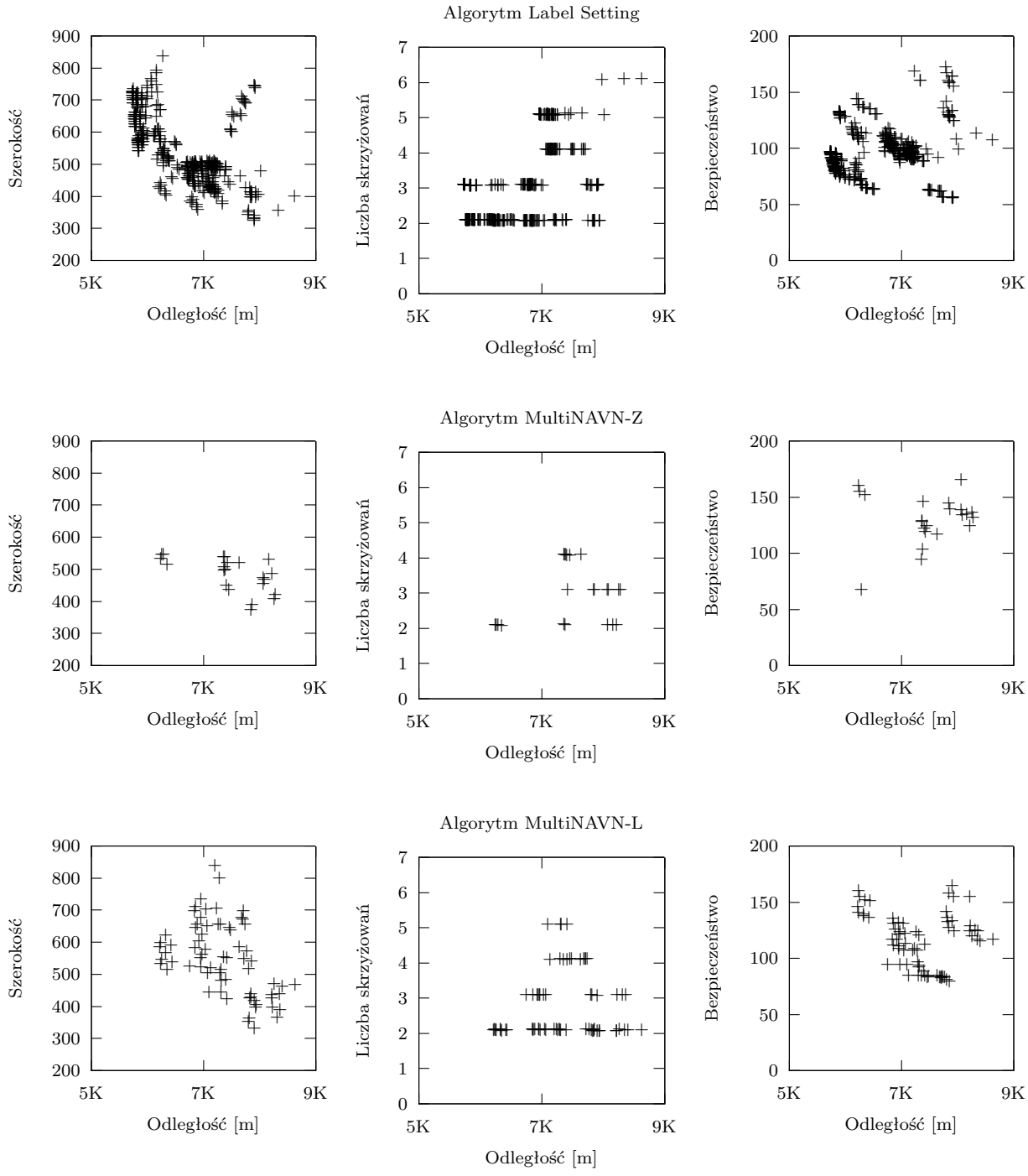
Rysunek 6.24: Zbiory rozwiązań wyznaczone przez porównywane algorytmy na mapie KAT-CEN dla drogi nr 02



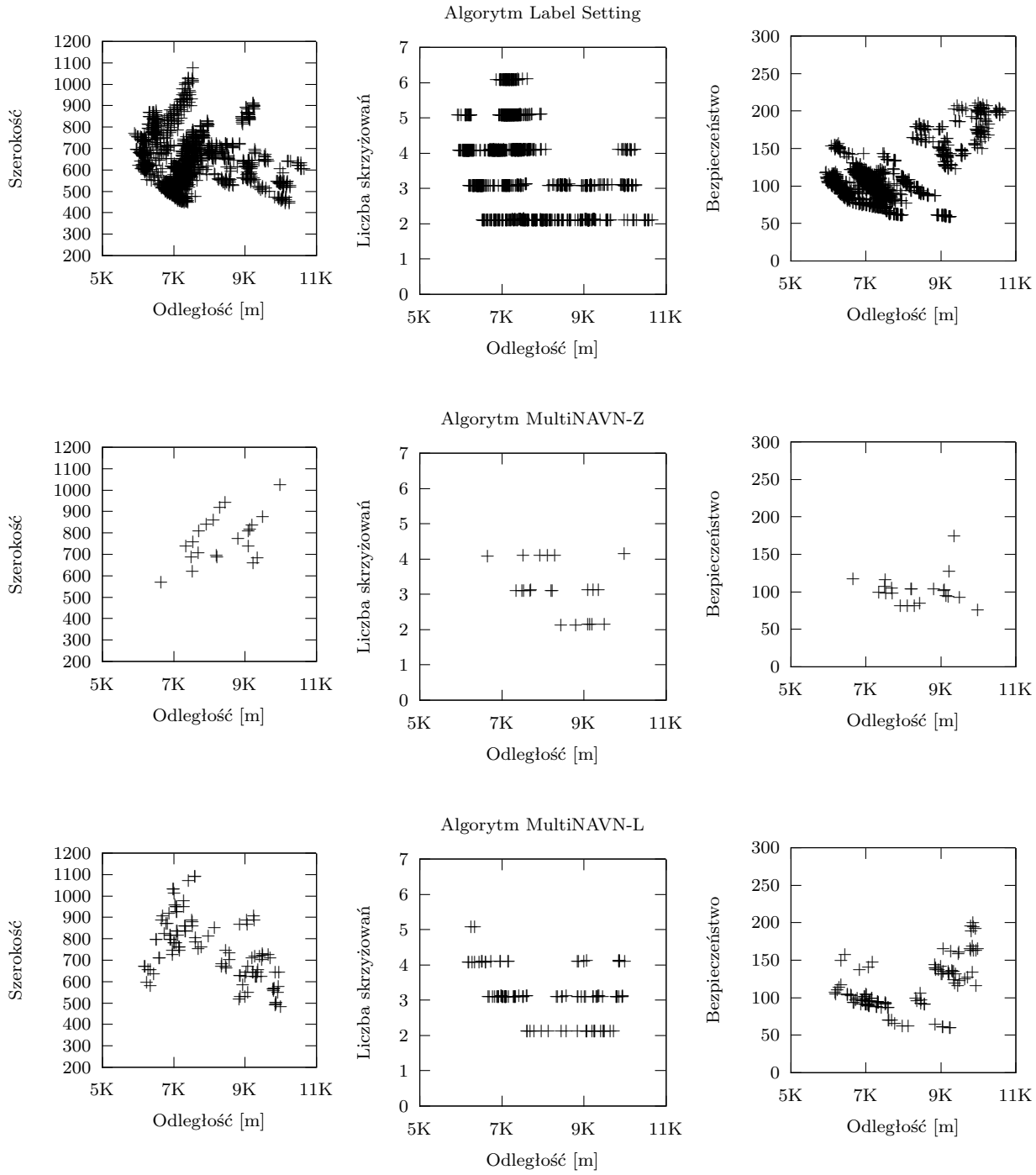
Rysunek 6.25: Zbiory rozwiązań wyznaczone przez porównywane algorytmy na mapie KAT-CEN dla drogi nr 03



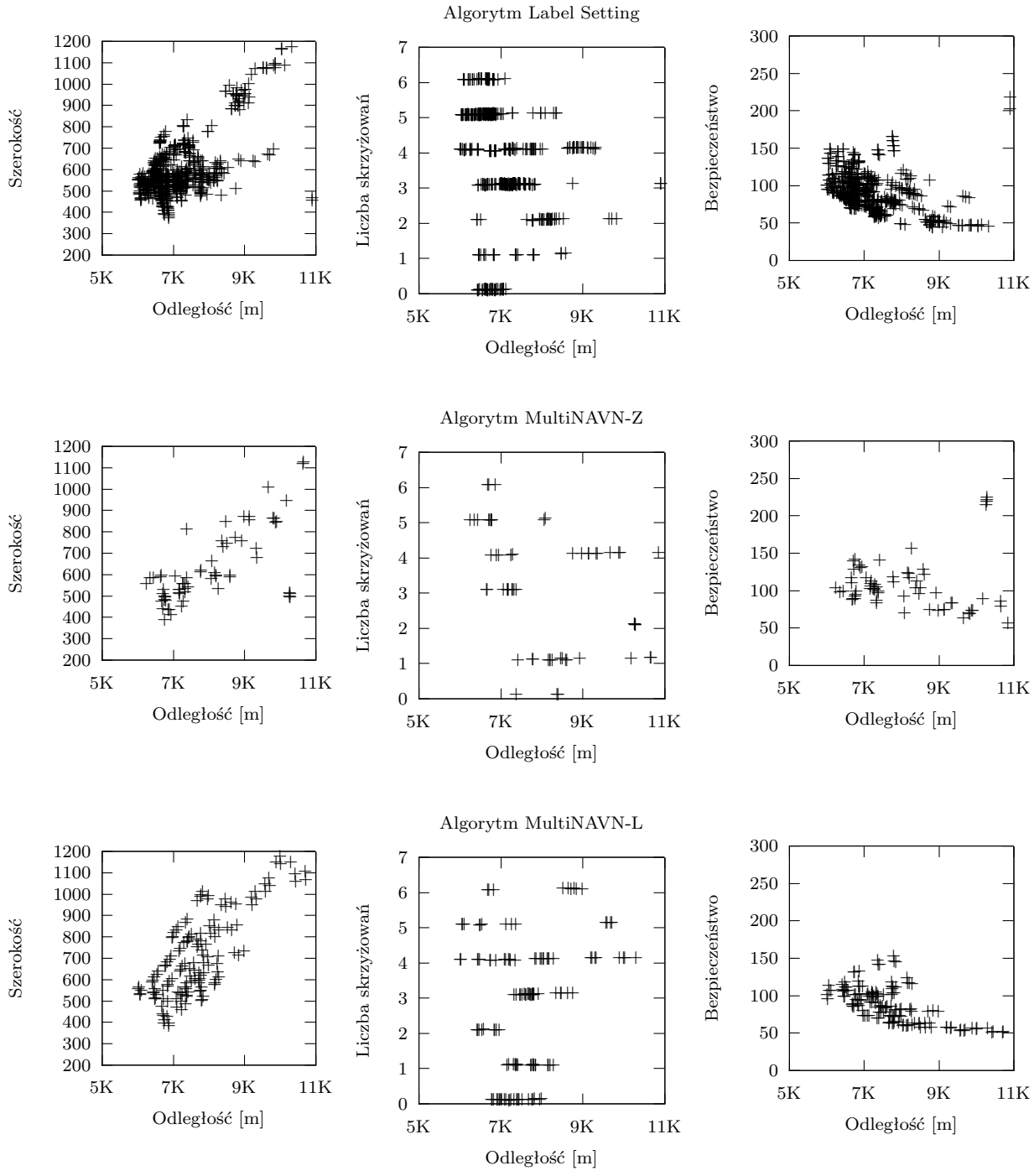
Rysunek 6.26: Zbiory rozwiązań wyznaczone przez porównywane algorytmy na mapie KAT-CEN dla drogi nr 04



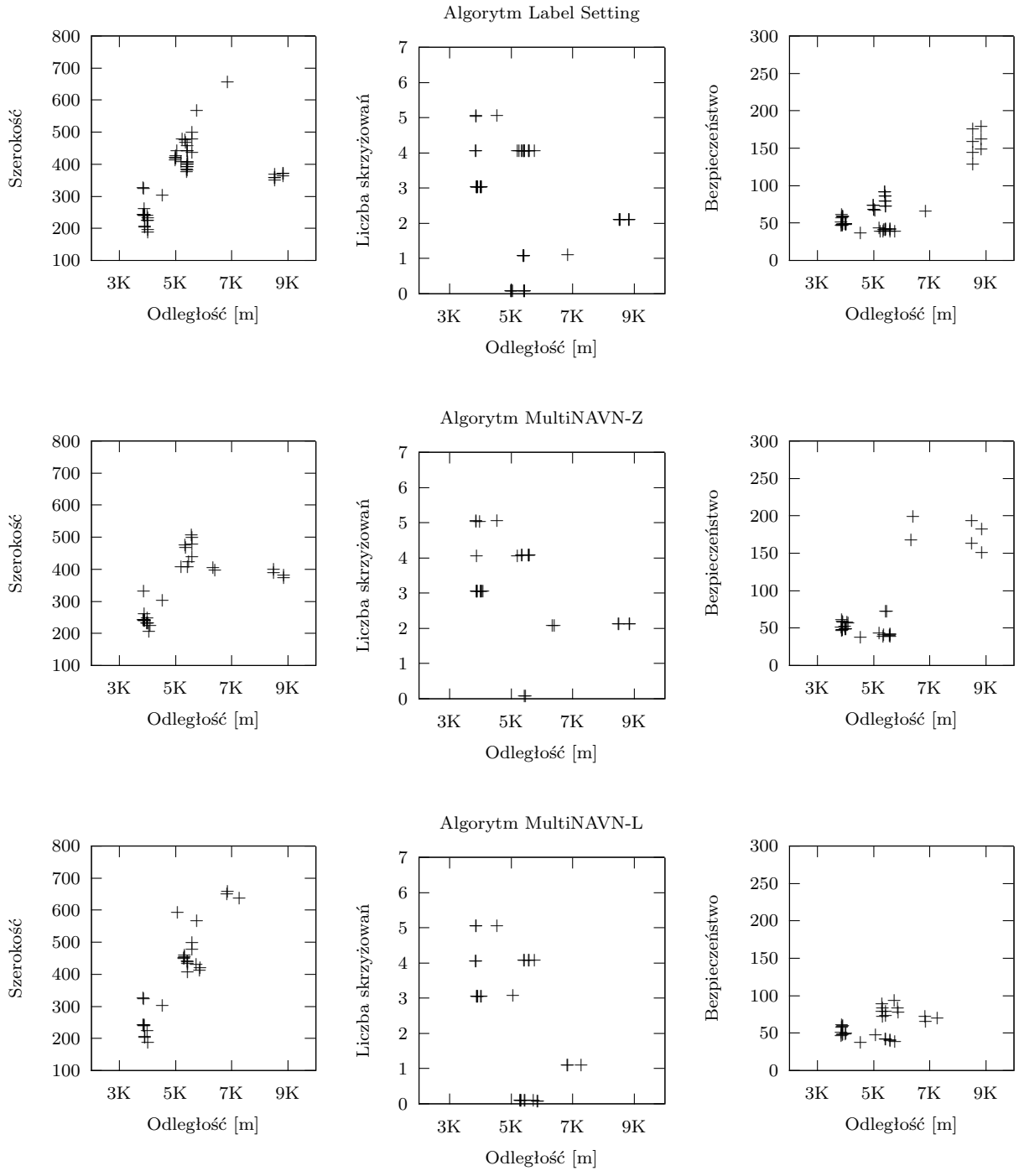
Rysunek 6.27: Zbiory rozwiązań wyznaczone przez porównywane algorytmy na mapie KAT-CEN dla drogi nr 05



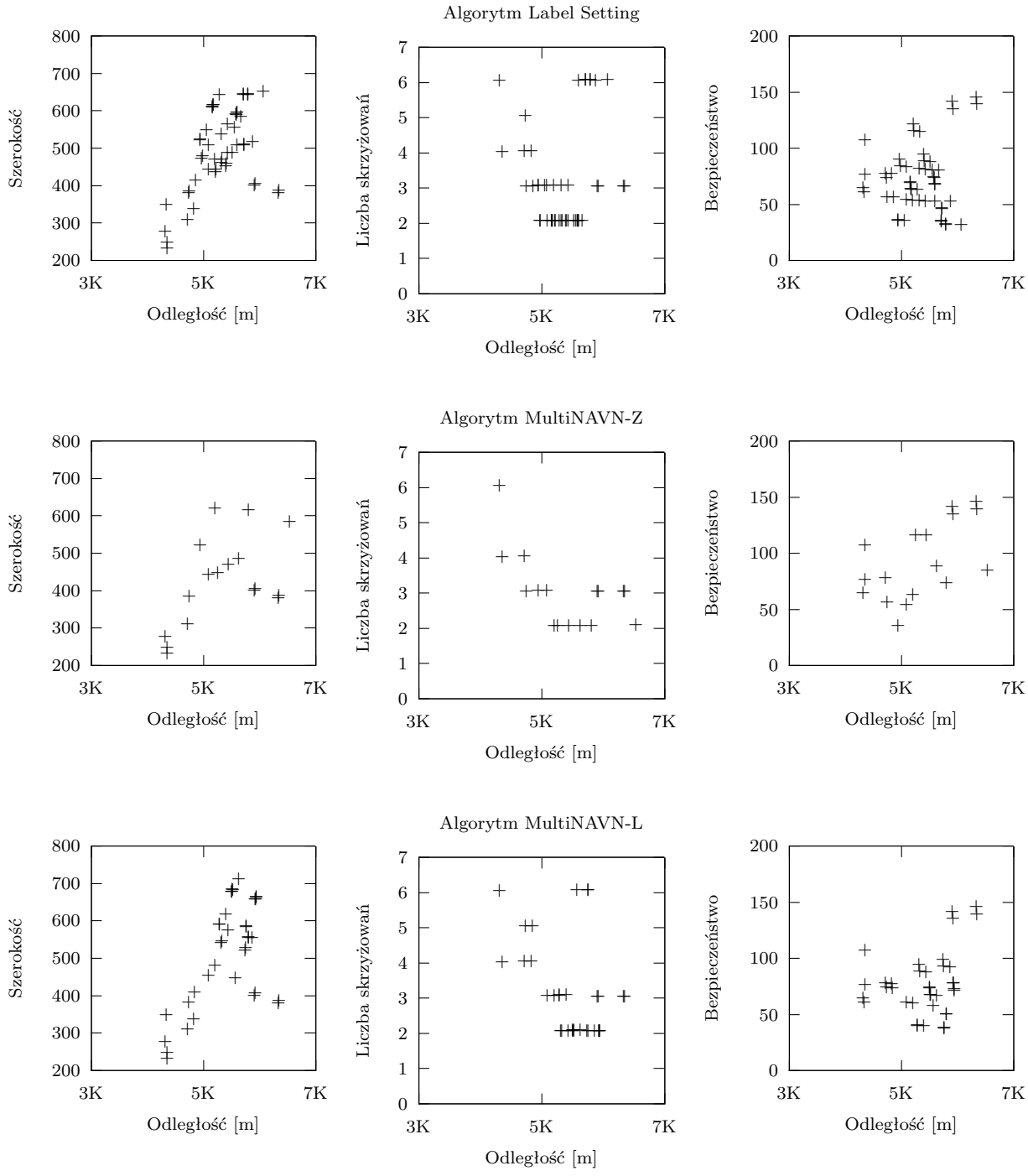
Rysunek 6.28: Zbiory rozwiązań wyznaczone przez porównywane algorytmy na mapie KAT-CEN dla drogi nr 06



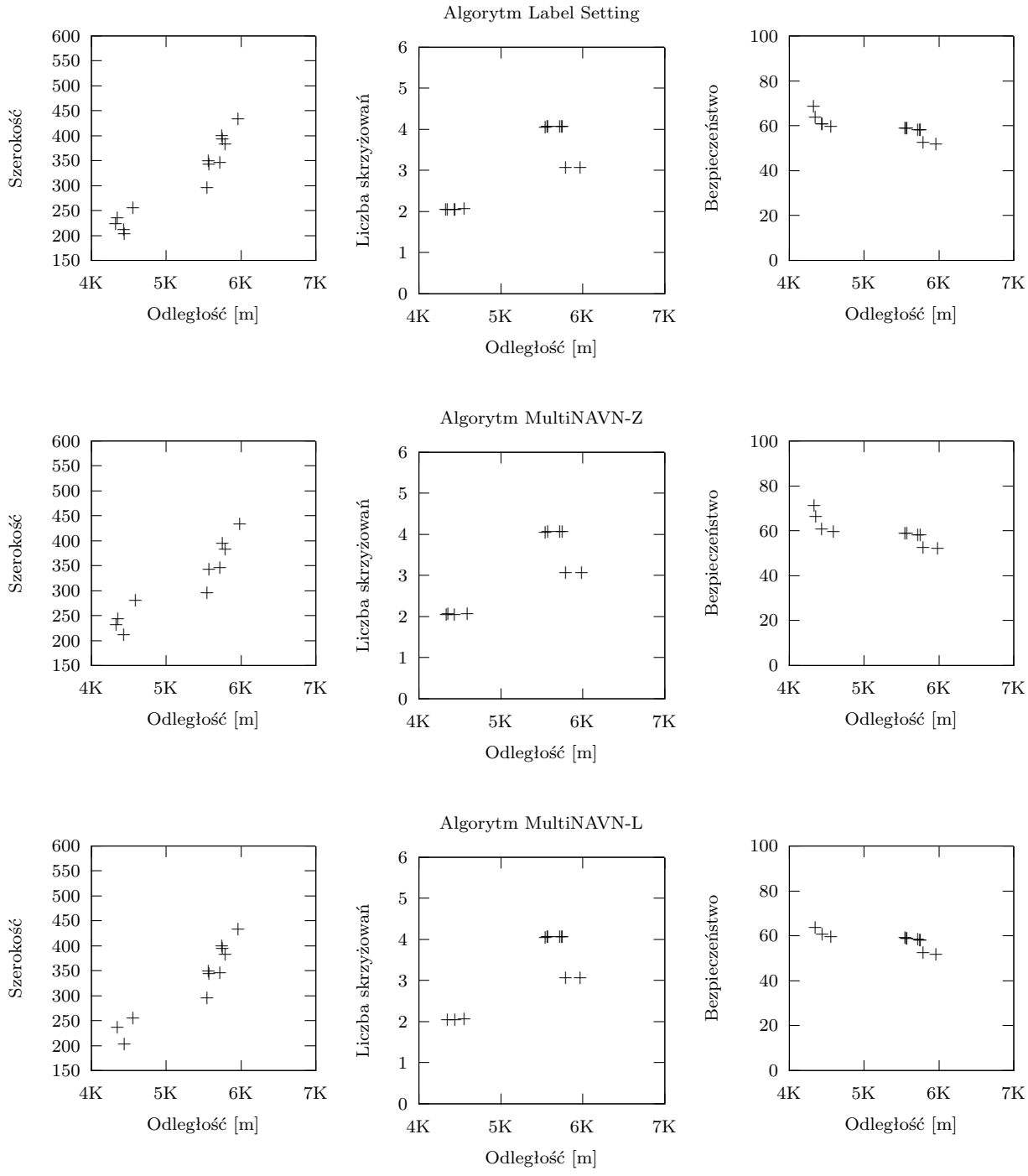
Rysunek 6.29: Zbiory rozwiązań wyznaczone przez porównywane algorytmy na mapie KAT-CEN dla drogi nr 07



Rysunek 6.30: Zbiory rozwiązań wyznaczone przez porównywane algorytmy na mapie KAT-CEN dla drogi nr 08



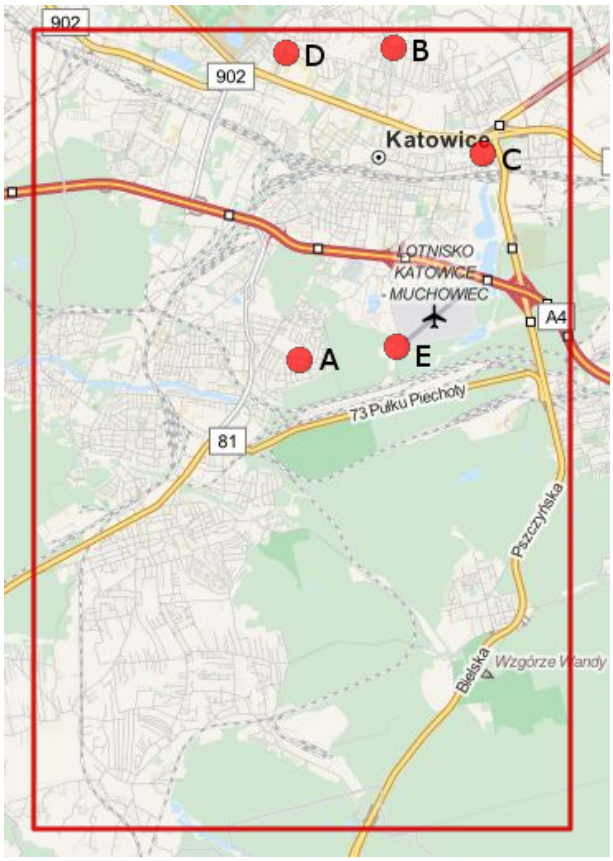
Rysunek 6.31: Zbiory rozwiązań wyznaczone przez porównywane algorytmy na mapie KAT-CEN dla drogi nr 09



Rysunek 6.32: Zbiory rozwiązań wyznaczone przez porównywane algorytmy na mapie KAT-CEN dla drogi nr 10

6.5.3 Analiza porównawcza wyników uzyskanych na mapie KAT-MID

W pierwszej części analizy porównawczej wykorzystano mapę, której wielkość wyrażona liczbą węzłów i krawędzi, pozwalała na uzyskanie rozwiązań dokładanych za pomocą algorytmu Label Setting. Dzięki temu możliwa była ocena jakości rozwiązań przybliżonych wyznaczanych przez algorytmy mrowiskowe. Średnie czasy działania poszczególnych algorytmów oraz zapotrzebowania przez nie na pamięć operacyjną podczas prób na mapie KAT-CEN, mogą poddawać w wątpliwość celowość stosowania niedeterministycznych algorytmów aproksymacyjnych. Zestawienie wyników czasu działania oraz wykorzystania pamięci operacyjnej przez poszczególne algorytmy zostało przedstawione w tabeli 6.25. W związku z tym w dalszej części eksperymentów przeprowadzono próby z większą mapą, której parametry zostały przedstawione w tabeli 6.26 oraz na rysunku 6.33 . W celach referencyjnych mapa została oznaczona skrótem KAT-MID. Liczba węzłów oraz liczba krawędzi grafu, który reprezentuje mapę KAT-MID jest kilka razy większa niż w przypadku mapy KAT-CEN. Wykorzystane zostały te same punkty na mapie (tabela 6.19) oraz drogi (tabela 6.20), które były użyte podczas eksperymentów na mapie KAT-CEN.



Rysunek 6.33: Punkty na mapie KAT-MID wykorzystane jako miejsca początkowe i końcowe dróg wykorzystanych do analizy porównawczej

W pierwszej kolejności przeprowadzone zostały badania eksperymentalne z algorytmem Label Setting mające na celu uzyskanie dokładnych, pełnych zbiorów rozwiązań paretooptymalnych, dla wszystkich dziesięciu dróg, tak aby możliwa była ocena jakości

Tabela 6.26: Mapa KAT-MID użyta w drugim etapie analizy porównawczej

Długość geograficzna	18,9504°E - 19,0620°E
Szerokość geograficzna	50,1700°N - 50,2764°N
Liczba węzłów	7640
Liczba krawędzi	14411
Liczba węzłów z sygnalizacją	69

aproksymacji wyznaczanych przez algorytmy mrowiskowe MultiNAVN-Z i MultiNAVN-L. Okazało się, że na dostępnych zasobach sprzętowo programowych nie udało się wyznaczyć pełnych rozwiązań dla wszystkich dziesięciu dróg, a jedynie dla czterech z nich. Powodem okazały się bardzo duże wymagania na ilość pamięci operacyjnej RAM. Dostępne 4 GB pamięci dla programów Java nie było wystarczające dla sześciu z dziesięciu dróg będących przedmiotem eksperymentów. Wyniki dla pozostałych czterech dróg zostały przedstawione w tabeli 6.27.

Tabela 6.27: Wyniki osiągnięte przez algorytm Label Setting mapie KAT-MID

Droga	**Czas [ms]**	**Liczba rozwiązań**	**Pamięć [kB]**
1	1675808	2207	1123590
2	1203168	867	1169710
7	1970573	783	1096557
8	1902541	65	1092799

W związku z powyższym dalsze eksperymenty porównawcze obu wersji algorytmu MultiNAVN: MultiNAVN-Z i MultiNAVN-L zostały ograniczone do dróg 1, 2, 7 i 8, dla których dostępne były referencyjne, pełne zbiory rozwiązań paretooptymalnych. Wszystkie wartości parametrów algorytmów mrowiskowych były takie same jak podczas eksperymentów na mapie KAT-CEN. Zastosowano również taką samą metodę statystycznego porównywania danych eksperymentalnych. We wszystkich tabelach prezentujących wyniki eksperymentów porównawczych wartości bezwzględne statystyki Z dla, których nie jest spełniona hipoteza zerowa oznaczone zostały pogrubioną czcionką.

W tabeli 6.28 zaprezentowane są wyniki eksperymentów porównawczych dwóch wersji algorytmu MulitNAVN w zakresie wartości metryki Hausdorffa. Hipoteza zerowa została odrzucona w przypadku drogi nr 7, dla której niższa średnia wartość metryki Hausdorffa została osiągnięta przez algorytm MultiNAVN-L. Wskazuje to na przewagę tej wersji algorytmu w zakresie podobieństwa wyznaczanego zbioru rozwiązań do zbioru referencyjnego wyznaczonego przez algorytm Label Setting.

Wyniki porównania średnich liczności rozwiązań niezdominowanych wyznaczonych przez obie wersje algorytmu, zostały przedstawione w tabeli 6.29. Podobnie jak w przypadku prób wykonanych na mapie KAT-CEN, także w przypadku mapy KAT-MID algorytm MultiNAVN-L wykazał znacznie lepszą sprawność mierzoną średnią licznością zbiorów rozwiązań niezdominowanych. Hipoteza zerowa została odrzucona dla trzech z czterech dróg i dla wszystkich tych przypadków lepsza okazała się wersja MultiNAVN-L.

Biorąc pod uwagę średnie czasy działania porównywanych wersji algorytmów, algorytm MultiNAVN-Z wykazał się lepszymi rezultatami dla wszystkich badanych dróg. Przewaga algorytmu MultiNAVN-Z była nawet bardziej wyraźna niż w przypadku prób

Tabela 6.28: Porównanie metryk Hausdorffa zbiorów rozwiązań wyznaczonych przez algorytmy MultiNAVN-Z i MultiNAVN-L na mapie KAT-MID

	MultiNAVN-Z		MultiNAVN-L		
Nr drogi	$\bar{x}_z$	s_z	$\bar{x}_l$	s_l	$\|Z\|$
1	1436,95	369,31	1719,41	892,45	1,850
2	1023,41	349,87	856,07	362,40	2,101
7	4354,86	975,25	**3152,33**	977,01	**5,509**
8	3400,82	927,73	3000,03	1217,92	1,656

Tabela 6.29: Porównanie liczności zbiorów rozwiązań wyznaczonych przez algorytmy MultiNAVN-Z i MultiNAVN-L na mapie KAT-MID

	MultiNAVN-Z		MultiNAVN-L		
Nr drogi	$\bar{y}_z$	s_z	$\bar{y}_l$	s_l	$\|Z\|$
1	61,95	22,31	**212,08**	70,24	**12,884**
2	56,76	19,56	**99,26**	39,66	**6,078**
7	40,50	13,46	**117,87**	46,96	**10,017**
8	15,63	3,91	16,45	5,23	0,794

z mapą KAT-CEN, co zostało zaprezentowane w tabeli 6.30. Hipoteza zerowa została odrzucona dla wszystkich czterech dróg i dla każdej z nich krótsze czasy działania uzyskała wersja algorytmu MultiNAVN-Z.

Tabela 6.30: Porównanie czasu wykonania algorytmów MultiNAVN-Z i MultiNAVN-L na mapie KAT-MID

	MultiNAVN-Z		MultiNAVN-L		
Nr drogi	$\bar{z}_z$ [ms]	s_z	$\bar{z}_l$ [ms]	s_l	$\|Z\|$
1	**8600,55**	1359,06	28043,08	16784,30	**7,302**
2	**5152,71**	591,91	10156,82	2212,16	**13,821**
7	**7506,08**	1328,90	13758,21	2315,98	**14,809**
8	**3351,82**	305,96	8261,32	1478,52	**20,565**

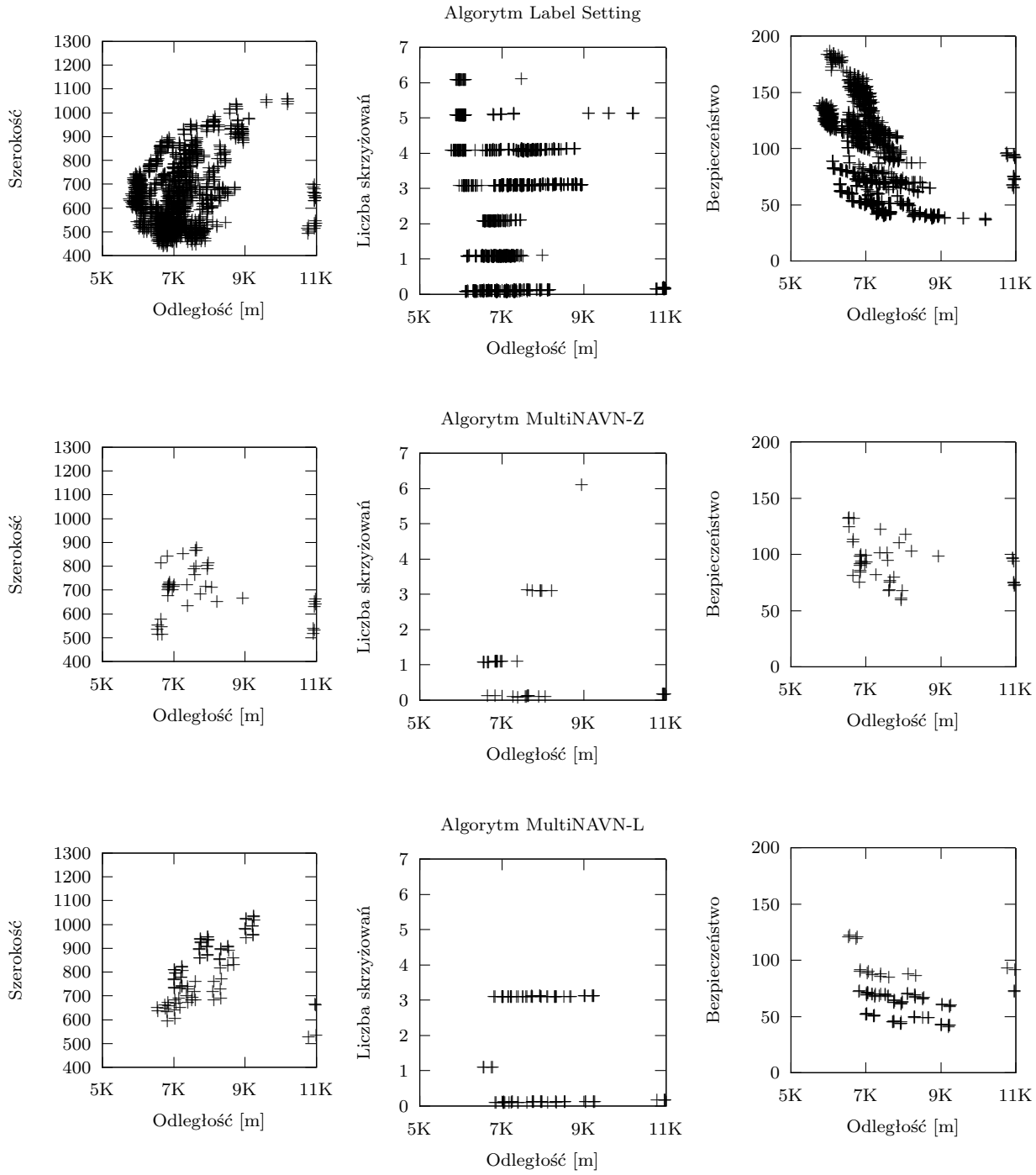
W tabeli 6.31 zaprezentowane zostało zestawienie średnich czasów działania oraz konsumpcji pamięci operacyjnej badanych algorytmów: MultiNAVN-Z, MultiNAVN-L oraz Label Setting. Porównując te wyniki z wynikami podsumowującymi pierwszy etap eksperymentów (tabela 6.25) można wyraźnie zaobserwować nieliniowy wzrost konsumpcji pamięci operacyjnej oraz nieliniowy wzrost czasu wykonania algorytmu Label Setting. W przypadku wszystkich czterech dróg, które były badane na mapie KAT-MID, algorytm Label Setting działał zdecydowanie wolniej niż algorytmy mrowiskowe, ale wymagał do działania mniejszej ilości pamięci operacyjnej. Należy jednak zwrócić uwagę, na fakt, że w przypadku pozostałych sześciu dróg, nie udało się wyznaczyć za pomocą algorytmu Label Setting pełnego zbioru rozwiązań niezdominowanych ze względu na przekroczenie rozmiaru pamięci operacyjnej dostępnej na komputerze użytym do przeprowadzenia eksperymentów. Duże zapotrzebowanie na pamięć operacyjną algorytmu Label Setting było przyczyną wykorzystania w drugim etapie eksperymentów porównawczych

tylko czterech dróg z pośród dziesięciu użytych podczas pierwszego etapu. Fakt ten uzasadnia stosowanie metod przybliżonych do rozwiązywania wielokryterialnego problemu najkrótszej drogi.

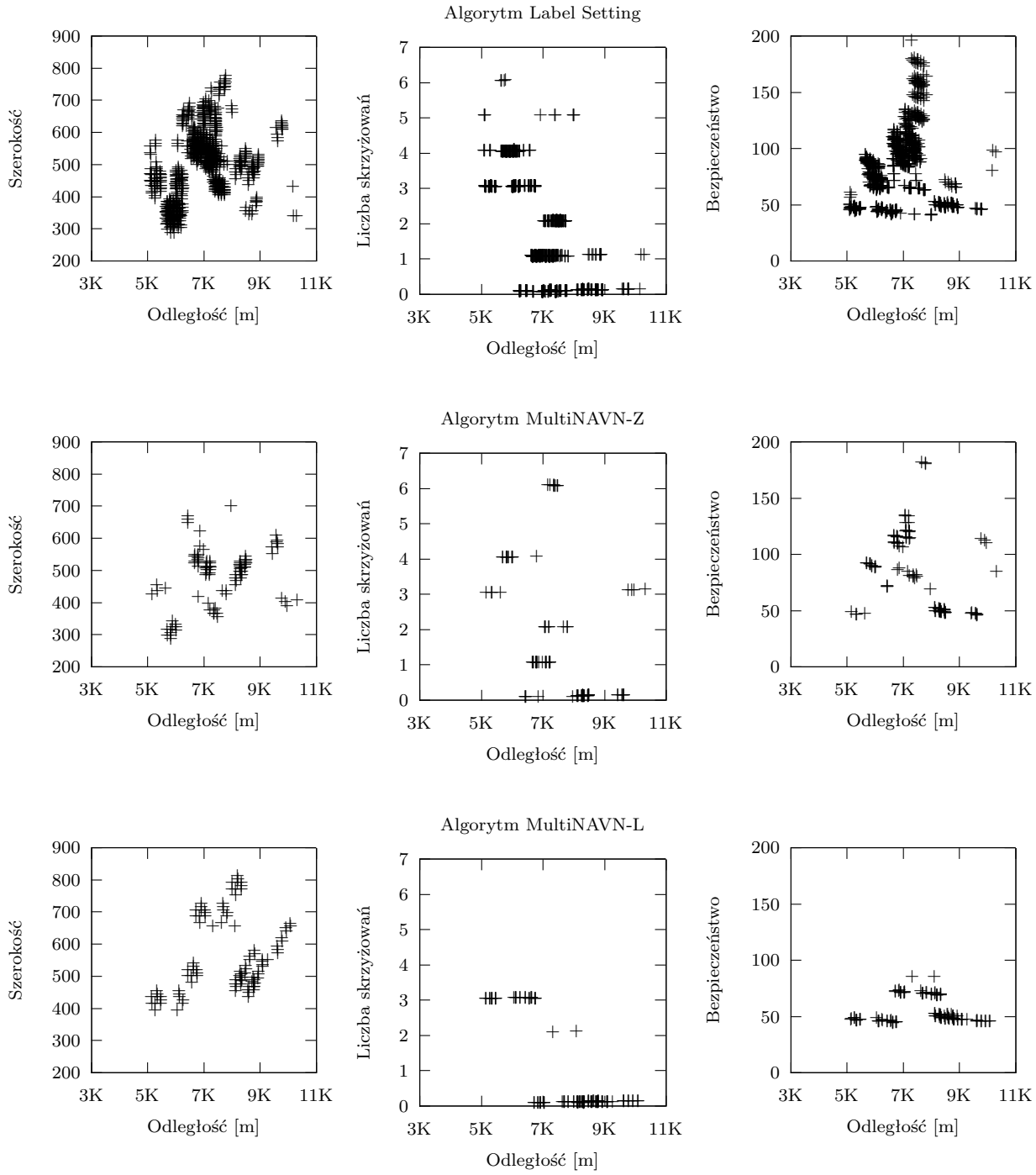
Tabela 6.31: Porównanie konsumpcji pamięci operacyjnej oraz czasu działania algorytmów MultiNAVN-Z, MultiNAVN-L oraz Label Setting na mapie KAT-MID

	MultiNAVN-Z		MultiNAVN-L		Label Setting	
Nr drogi	$\bar{m}_z$ [kB]	$\bar{z}_z$ [ms]	$\bar{m}_l$ [kB]	$\bar{z}_l$ [ms]	$\bar{m}_{ls}$ [kB]	$\bar{z}_{ls}$ [ms]
1	92199	8601	164330	28043	122865	74856
2	119333	5153	310322	10157	122874	73876
7	134074	7506	240255	13758	56842	14924
8	175730	3352	339084	8261	57171	14913

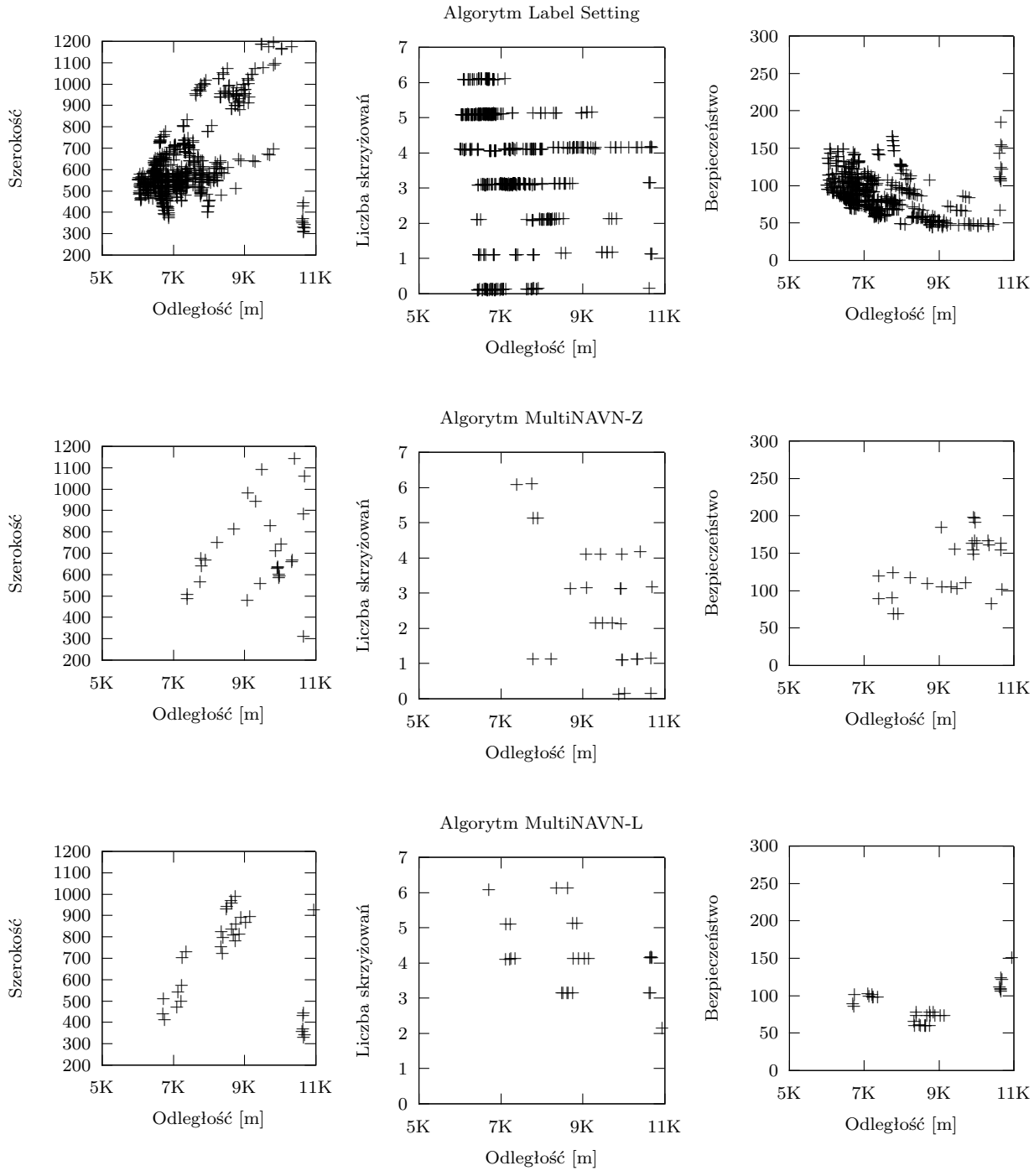
Na rysunkach od 6.34 do 6.37 zaprezentowano pełne zbiory rozwiązań, które zostały wyznaczone przez algorytm Label Setting dla czterech dróg (1,2,7 i 8) oraz wybrane aproksymacje zbiorów rozwiązań niezdominowanych wyznaczone przez algorytmy MultiNAVN-Z i MultiNAVN-L. Podobnie jak to miało miejsce w przypadku wcześniejszych eksperymentów, przeprowadzonych na mapie KAT-CEN, wybrane rozwiązania charakteryzują się najmniejszą metryką Hausdorffa spośród wszystkich uruchomień algorytmów.



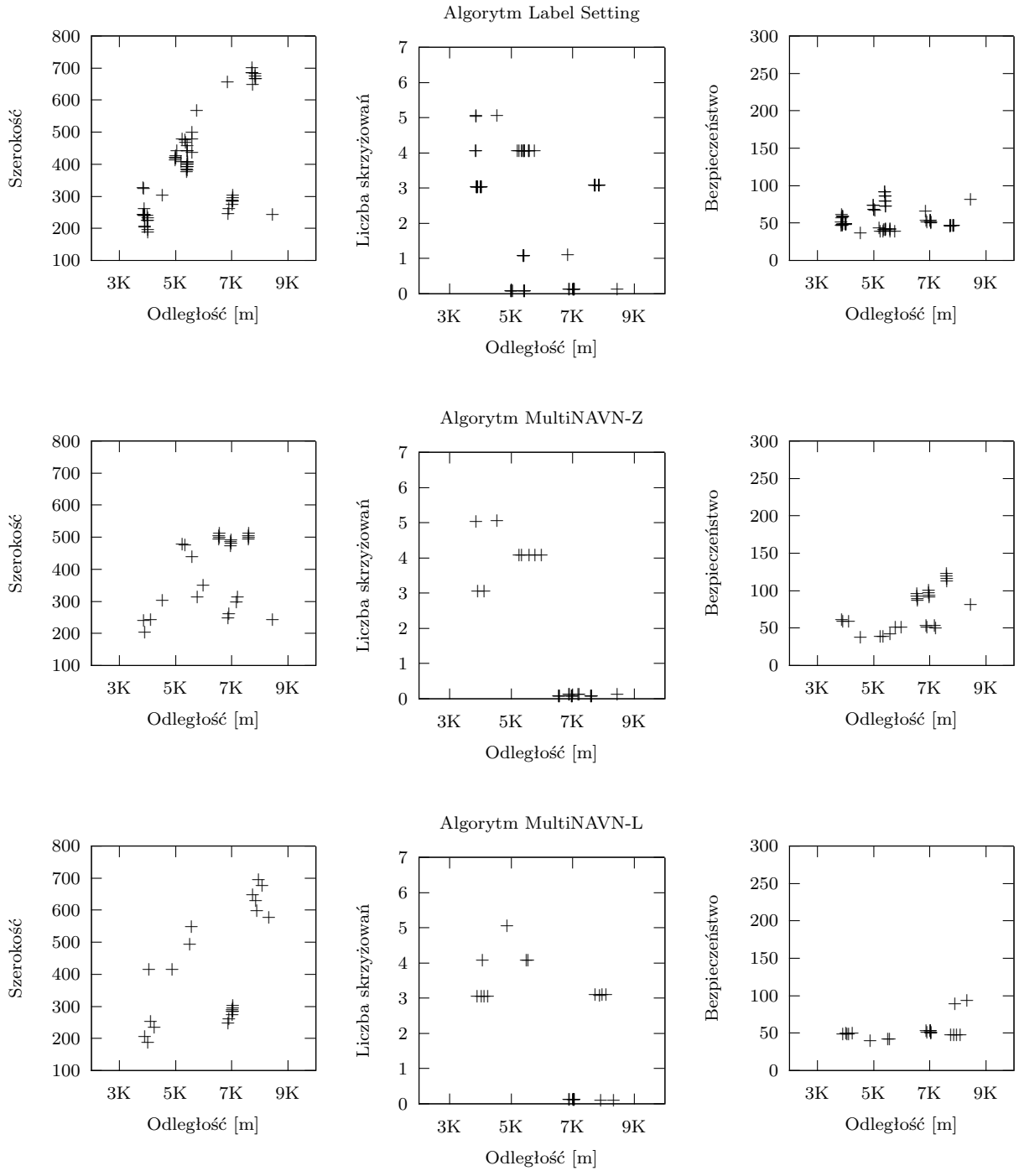
Rysunek 6.34: Zbiory rozwiązań wyznaczone przez porównywane algorytmy na mapie KAT-MID dla drogi nr 01



Rysunek 6.35: Zbiory rozwiązań wyznaczone przez porównywane algorytmy na mapie KAT-MID dla drogi nr 02



Rysunek 6.36: Zbiory rozwiązań wyznaczone przez porównywane algorytmy na mapie KAT-MID dla drogi nr 07



Rysunek 6.37: Zbiory rozwiązań wyznaczone przez porównywane algorytmy na mapie KAT-MID dla drogi nr 08

Wyniki eksperymentów porównawczych przeprowadzonych na mapie KAT-MID potwierdzają wcześniejsze wnioski, że wersja algorytmu MultiNAVN-L, wyznacza rozwiązania lepszej jakości, charakteryzujące się w stosunku do wersji MultiNAVN-Z średnio niższą metryką Hausdorffa oraz wyższą licznością zbioru rozwiązań niezdominowanych. Podobnie jak to miało miejsce w przypadku prób wykonanych na mapie KAT-CEN wersja algorytmu MultiNAVN-Z działała szybciej, niż wersja MultiNAVN-L ze względu na mniej eksploracyjną, a bardziej zachłanną charakterystykę obliczeń.

Eksperymenty z większą mapą wykazały również celowość stosowania opisywanych algorytmów aproksymujących w sytuacji gdy metody dokładne, takie jak algorytm Label Setting, nie są w stanie dostarczyć pełnego zbioru rozwiązań niezdominowanych ze względu na ograniczenia pamięciowe lub czasowe.

Rozdział 7

Wersja równoległa algorytmu OCLAVN

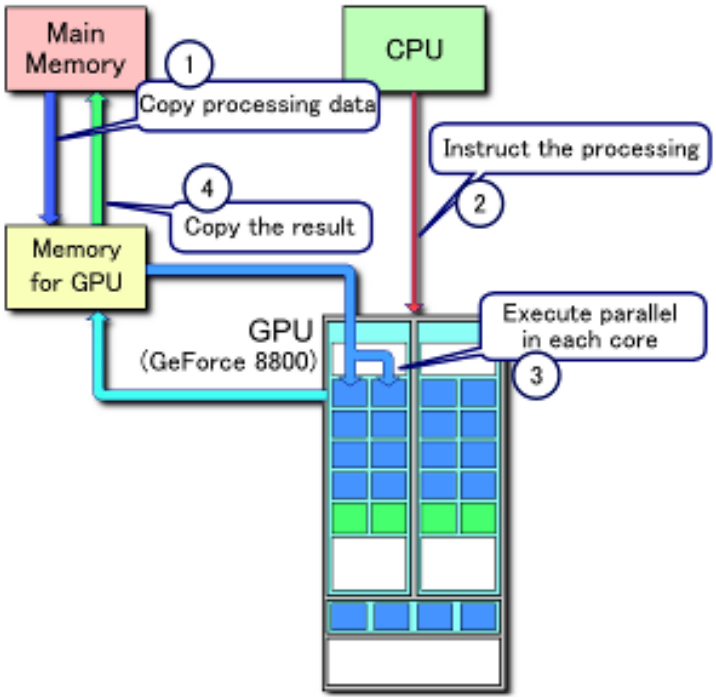
Dalsze prace związane z ulepszeniem równoległej wersji algorytmu PAVN [14] polegały na wykorzystaniu potencjału obliczeniowego nowoczesnych procesorów graficznych GPU (ang. *graphics processing unit*) o architekturze CUDA (ang. *compute unified device architecture*) firmy nVIDIA. W rezultacie przygotowany został algorytm, zaimplementowany z wykorzystaniem interfejsu programistycznego OpenCL i nazwany OCLAVN (ang. *opencl ant vehicle navigation*) [20].

7.1 Architektura CUDA

Architektura CUDA, przedstawiona na rys. 7.1, została opracowana przez firmę nVIDIA do obliczeń równoległych na procesorach graficznych (GPU) oraz na dedykowanych akceleratorach (np. Tesla). To sprawia, że możliwe jest wykorzystanie GPU do obliczeń ogólnego przeznaczenia wykonywanych do tej pory wyłącznie na jednostkach CPU.

W przeciwieństwie do tradycyjnej architektury CPU, procesor graficzny GPU wymaga podziału zadań na wiele wątków, które są realizowane stosunkowo powoli, ale w skali globalnej wyznaczają rozwiązanie szybciej niż tradycyjne, sekwencyjne procesory ogólnego przeznaczenia. Pełny potencjał takiej architektury może być wykorzystany w przypadku algorytmów, które mogą być podzielone na mniejsze fragmenty wykonywane równolegle.

Aby możliwe było wykonanie algorytmu OCLAVN, także na sprzęcie innych producentów, zdecydowano się na wykorzystanie interfejsu programistycznego OpenCL (ang. *open computing language*). OpenCL to interfejs programistyczny (ang. *application program interface* – API) niskiego poziomu dla komputerów o budowie heterogenicznej, w tym systemów wykorzystujących elementy architektury CUDA. Wykorzystując interfejs OpenCL programiści mogą tworzyć programy przy użyciu języka programowania opartego na standardzie C99, które następnie są wykonywane na procesorze GPU.



Rysunek 7.1: Schemat architektury CUDA

7.2 Implementacja algorytmu NAVN w architekturze CUDA

W trakcie przygotowywania równoległej wersji algorytmu NAVN musiały zostać wzięte pod uwagę różne specyficzne czynniki związane z docelową architekturą CUDA oraz interfejsem programistycznym OpenCL. W rezultacie została opracowana nowa wersja algorytmu NAVN – OCLAVN. Poniżej opisane zostały zagadnienia, które musiały zostać uwzględnione i rozwiązane w opracowanej wersji równoległej algorytmu NAVN.

Optymalizacja transmisji danych między komputerem macierzystym a procesorem GPU. Obliczenia w architekturze CUDA są wykonywane w heterogenicznym środowisku. Aplikacja użytkownika jest podzielona na dwie części: kod systemu macierzystego oraz kod dla GPU. Obydwie części muszą się ze sobą komunikować w celu wykonania zadania. W programach opartych na interfejsie OpenCL, dane muszą być przesyłane z komputera macierzystego do procesora GPU i w kierunku odwrotnym. Przesyły danych między pamięcią CPU a GPU mogą zmniejszyć sprawność działania, więc ich liczba powinna być ograniczona do minimum. Dane powinny być przechowywane w procesorze GPU tak długo, jak to jest możliwe. Wersja OpenCL algorytmu NAVN spełnia te wymagania. Prawie wszystkie operacje wykonywane są w procesorze GPU a dane są przechowywane w pamięci GPU.

Minimalizacja wykorzystania pamięci globalnej. Specyfikacja interfejsu OpenCL opisuje trzy rodzaje pamięci: globalna, lokalna i prywatna. Różnią się one dostępnością, pojemnością i czasem dostępu. Pamięć globalna ma największą pojemność i jest dostępna dla każdego wątku, ale równocześnie jest zdecydowanie najwolniejsza. Pamięć lokalna może być współużytkowana przez wątki w tej samej grupie, a pamięć prywatna jest dostępna

jedynie dla bieżącego wątku i nie jest widoczna dla innych wątków. W wielu miejscach była konieczna zmiana sposobu działania algorytmu NAVN tak aby zminimalizować konieczność dostępu do pamięci globalnej. Na przykład, informacje o stanie mrówek i rozwiązaniach budowanych przez mrówki mieszczą się w pamięci lokalnej w celu zwiększenia szybkości działania algorytmu.

Eliminacja konieczności synchronizacji. W trakcie obliczeń równoległych zdarzają się sytuacje, gdy dwa (lub więcej) równoległych wątków musi zmodyfikować to samo miejsce w pamięci globalnej. Do rozwiązywania tego typu problemów mogą zostać użyte różne metody synchronizacji równoległych wątków, na przykład: operacje aktualizacji atomowej lub semafory. Interfejs programistyczny OpenCL wspiera operacje atomowe na zmiennych całkowitych w pamięci globalnej. Dzięki temu jest możliwe wprowadzenie i zastosowanie semaforów, aby uniknąć jednoczesnego dostępu do tych samych komórek pamięci przez równoległe wątki. Niestety, operacje atomowe z udziałem pamięci globalnej spowalniają pracę procesora GPU, dlatego postanowiono zmienić algorytm NAVN w taki sposób, że synchronizacja wątków nie jest konieczna. W algorytmie NAVN istnieją dwa główne źródła konfliktów w dostępie do pamięci: aktualizacja on-line śladu feromonowego w procedurze WybierzKrawędź oraz cała procedura UkarajPrzegraneMrówki. Z tego powodu procedury te zostały usunięte z ostatecznej wersji algorytmu OCLAVN, a ślad feromonowy jest aktualizowany w procedurach CofnijMrówkę i NagrodźNajlepszeRozwiązanie.

Optymalizacje kodu dla architektury SIMD. Aby uzyskać możliwie najlepszą szybkość wykonywania algorytmu wątki powinny być wykonywane w grupach składających się z co najmniej 32 elementów (ang. *warp size*), a łączna liczba wątków powinna sięgać kilku tysięcy. Instrukcje warunkowe i w konsekwencji rożne ścieżki wykonania programu nie mają dużego wpływu na szybkość działania pod warunkiem, że wszystkie wątki w grupie wykonują tę samą ścieżkę realizacji (ang. *single instruction multiple data*). W różnych miejscach algorytm NAVN został zmieniony w taki sposób, aby jak najczęściej uzyskać tę samą ścieżkę wykonania wszystkich wątków w grupie.

Inne zagadnienia. Aby zapewnić maksymalną szybkość obliczeń na procesorze GPU o architekturze CUDA, algorytm OCLAVN używa liczb zmiennopozycyjnych o pojedynczej precyzji. Z tego samego powodu używane są operacje arytmetyczne, które wykorzystują bezpośrednio możliwości sprzętowe procesora GPU. Operacje te są szybsze, ale zapewniają nieco mniejszą dokładność nie mającą jednak znaczącego wpływu na sprawność badanych algorytmów.

Pseudokod algorytmu oznaczono jako Algorytm 10. Procedura UruchomMrówki jest wykonywana przez procesory na urządzeniu GPU. W pseudokodzie algorytmu OCLANV, dodatkowego komentarza wymagają następujące słowa kluczowe: WRITE, READ, KERNEL oraz BARRIER

WRITE – realizuje transfer danych z pamięci komputera macierzystego, do globalnej pamięci procesora GPU.

READ – transfer danych z globalnej pamięci na urządzeniu, do pamięci w komputerze macierzystym.

KERNEL – inicjuje wykonanie wskazanej procedury w procesorze GPU za pomocą wskazanej liczby wątków (ang. *work items*) podzielonych na grupy (ang. *work groups*).

BARRIER – umożliwia synchronizację wątków przez zatrzymanie wykonywania i oczekiwanie na wszystkie wątki.

Algorytm 10 Algorytm OCLAVN

```
1: PrzygotójNormalizację
2: Inicjuj
3: InicjujPamięćUrządzeń

4: for all urządzenie do
5:     write MapaPrzeszukiwania
6: end for

7: for I_cykl ← 1 to N_cykl do
8:     for all procesor GPU do
9:         kernel UruchomMrówki for N_m · RozmiarWarpa threads in N_m groups
10:    end for

11:    for all urządzenie do
12:        read RozwiązaniaMrówek
13:    end for

14:    for all urządzenie do
15:        OceńMrówki
16:    end for

17:    Modyfikuj q_0

18:    for all urządzenie do
19:        write ZmienioneParametry
20:    end for

21:    if NajlepszeRozwiązanie się zmieniło then
22:        for all urządzenie do
23:            write NajlepszeRozwiązanie
24:        end for
25:    end if

26:    for all urządzenie do
27:        kernel NagrodźNajlepszeRozwiązanie for LiczbaKrokówNajlepszegoRozwiązania threads
28:    end for
29: end for

30: ZwolnijPamięćNaUrządzeniach

31: return NajlepszeRozwiązanie
```

Algorytm 10 Algorytm OCLAVN (c.d.)

```
32: procedure UruchomMrówki

33:     IdWątku ← GetLocalThreadId
34:     NumerMrówki ← GetGroupId

35:     if IdWątku = 0 then
36:         KopiujGlobalneParametryDoPamięciLokalnej
37:         InicjujLokalnąMrówkę
38:     end if

39:     barrier

40:     InicjujListęTABU

41:     for all iteracja do
42:         if mrówka jest aktywna then
43:             WyznaczPrawdopodobieństwa
44:             if IdWątku = 0 then
45:                 if mrówka nie ma możliwości ruchu then
46:                     CofnijMrówkę
47:                 else
48:                     WybierzKrawędź
49:                     AktualizujListęTABU
50:                 end if
51:             end if
52:         end if

53:         barrier

54:     end for

55:     KopiujLokalnąMrówkęDoPamięciGlobalnej
56: end procedure
```

Tabela 7.1: Wartości parametrów algorytmów mrowiskowych użyte podczas eksperymentów z algorytmem OCLAVN

Algorytm	α	β	ω_p	ω_b	ρ	Q	φ	τ_0	N_m	N_{cykl}
NAVN	1.0	2.0	0.9	0.1	0.2	0.9	0.995	1/44500	30	50
OCLAVN	0.1	1.0	0.9	0.1	0.1	0.9	0.995	1/44500	30	50

Tabela 7.2: Parametry techniczne wybranych procesorów GPU firmy nVIDIA

GPU	**Liczba multiprocesorów**	**Liczba rdzeni CUDA**	**Poziom zdolności obliczeniowej**
GF 8400 GS	1	8	1.1
GF 9500 GT	4	32	1.1
Quadro NVS 320M	4	32	1.1
GF GTX 295	2x30	2x240	1.3

7.3 Badania eksperymentalne

Dane użyte do badań eksperymentalnych z algorytmem OCLAVN zostały uzyskane z systemu OpenStreetMap (OSM) dla miasta Katowice. Wykorzystany obszar mapy składał się z 31044 węzłów i 68934 krawędzi, a średnia liczba krawędzi przypadająca na węzeł wynosiła 2,2. Jako punkt startowy przyjęto węzeł o identyfikatorze 383783583 (przedmieścia Katowic), a jako punkt docelowy 384912139 (centrum Katowic). Szczegółowe informacje o wartościach pozostałych parametrów porównywanych algorytmów mrowiskowych zaprezentowane zostały w tab. 7.1.

Zmiany w oryginalnym algorytmie NAVN wprowadzone w celu dostosowania go do architektury CUDA spowodowały, że konieczne było zastosowanie innych wartości parametrów dla wersji sekwencyjnej i równoległej OCLAVN. Wartości tych parametrów zostały wyznaczone w trakcie wstępnych prób wykonania poszczególnych algorytmów. Eksperymenty porównawcze były przeprowadzone w komputerze z systemem Microsoft Windows XP oraz, w przypadku karty GF GTX 295, w komputerze z systemem Windows 7. Wszystkie aplikacje zostały napisane w języku C++ i skompilowane za pomocą narzędzia Microsoft Visual C++ 2008. Wersja algorytmu wykorzystująca interfejs OpenCL została zbudowana z wykorzystaniem bibliotek nVIDIA GPU Computing Toolkit 3.2.

Algorytm OCLAVN został wykonany na różnych procesorach GPU firmy nVIDIA, wyposażonych w architekturę CUDA. Parametry techniczne poszczególnych procesorów [1] zostały przedstawione w tab. 7.2. Eksperymenty pozwoliły zaobserwować jak możliwości sprzętowe poszczególnych procesorów GPU wpływają na rezultaty osiągane przez algorytm OCLAVN.

Tab. 7.3 prezentuje wybrane wyniki eksperymentów na różnych platformach. Zaprezentowane wyniki to średnie wartości uzyskane z 10 wykonań każdego algorytmu. Średnia jakość wyznaczanych rozwiązań (rozumiana jako wartość kosztu) jest bardzo podobna. Algorytm OCLAVN wykonany w procesorze GF GTX 295 uzyskał najlepszy rezultat

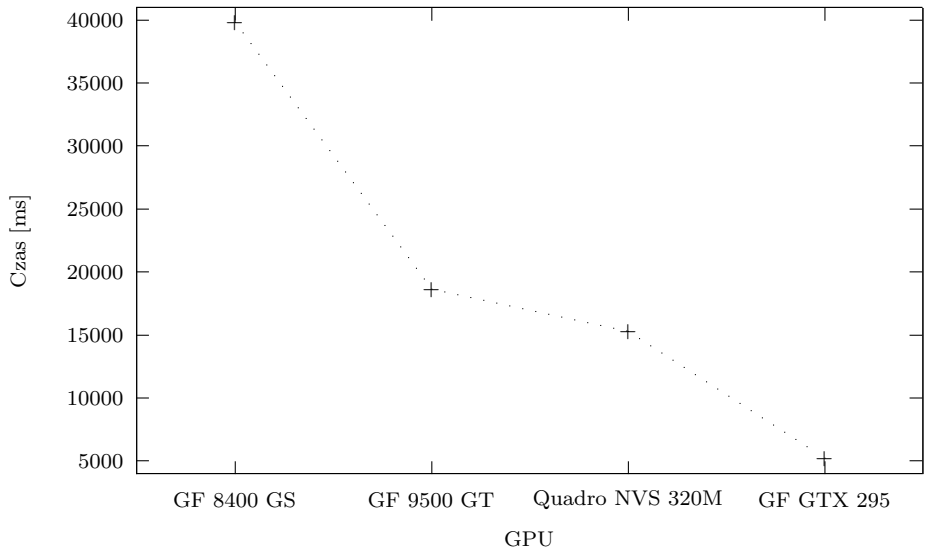
[1] W przypadku procesora GF GTX 295 zostało wykorzystane jedno z dostępnych urządzeń.

Tabela 7.3: Wybrane rezultaty eksperymentów z wersją OCLAVN

CPU	**GPU**	**Algorytm**	**Koszt**	**Czas [ms]**
Intel Core2 Quad 2,4 GHz	GF 8400 GS	OCLNAVN	15702	39842
Intel Core2 Quad 2,4 GHz	GF 9500 GT	OCLNAVN	15152	18618
Intel Core2 Duo 2,5 GHz	Quadro NVS 320M	OCLNAVN	14939	15270
Intel Core I5 2,67 GHz	GF GTX 295	OCLNAVN	15538	**5232**
Intel Core I5 2,67 GHz	-	NAVN	15651	6424

pod względem czasu działania, lepszy niż uzyskany przez sekwencyjną wersję algorytmu NAVN.

Na rys. 7.2 zaprezentowane są czasy działania algorytmu OCLAVN w różnych procesorach GPU. Na podstawie wyników można stwierdzić, że szybkość działania algorytmu zwiększa się wraz ze wzrostem liczby multiprocesorów. Nie jest to przyspieszenie liniowe, ponieważ algorytm OCLAVN nie jest całkowicie przystosowany do wymogów obliczeń w modelu SIMD. Struktury danych reprezentujące przestrzeń przeszukiwania są zlokalizowane w pamięci globalnej procesora GPU, dlatego ograniczenia szybkości działania związane z tym typem pamięci stały się ważnym czynnikiem wpływającym na całkowitą szybkość algorytmu OCLAVN.



Rysunek 7.2: Wyniki działania algorytmu OCLAVN w różnych procesorach GPU

7.4 Wnioski z badań eksperymentalnych

Zaproponowana została równoległa wersja algorytmu OCLAVN wykorzystująca architekturę CUDA, która jest dostępna na nowoczesnych procesorach GPU. Przeprowadzone eksperymenty wykazały zdolność algorytmu do wyznaczania rozwiązań dla danych

rzeczywistych. Czasy działania uzyskane przez algorytm OCLAVN były krótsze niż uzyskane przez wersję sekwencyjną NAVN. Ważnym wnioskiem z eksperymentów jest to, że algorytm OCLAVN jest dobrze przystosowany do wykorzystania potencjału nowoczesnych procesorów GPU, ponieważ czas działania zmniejsza się wraz ze wzrostem liczby multiprocesorów dostępnych w urządzeniu.

Rozdział 8

Podsumowanie

W rozprawie zaprezentowano zagadnienia związane z wyznaczaniem optymalnej drogi w nawigacji samochodowej z wykorzystaniem wielu kryteriów oceny rozwiązań. Na wstępie zostały omówione różne, klasyczne podejścia do rozwiązywania problemów optymalizacji wielokryterialnej, ze szczególnym uwzględnieniem podejścia zaproponowanego przez włoskiego ekonomistę Vilfredo Pareto. W metodzie tej zakłada się, że w procesie optymalizacji wielokryterialnej bardzo rzadko możliwe jest wyznaczenie jednego rozwiązania, które jest optymalne z punktu widzenia każdego kryterium oceny równocześnie. W związku z tym wynikiem optymalizacji wielokryterialnej metodą Pareto jest najczęściej zbiór rozwiązań niezdominowanych. Każde rozwiązanie, które należy do tego zbioru charakteryzuje się tym, że nie da się już polepszyć żadnego z kryterium oceny bez pogorszenia pozostałych.

W dalszej części rozprawy zdefiniowano wielokryterialny problem wyszukiwania najkrótszej drogi w grafie (ang. *multi-objective shortest path*) oraz dokonano przeglądu różnych metod rozwiązywania tego problemu. Charakteryzują się one dużą złożonością obliczeniową, co zachęca do stosowania algorytmów wyznaczających rozwiązania przybliżone. Są wśród nich algorytmy optymalizacji mrowiskowej, które zostały zaprezentowane w rozdziale 4. Następnie przedstawiono wielokryterialny algorytm mrowiskowej optymalizacji w nawigacji samochodowej znany z literatury jako AVN oraz zaprezentowano oryginalne, udoskonalone algorytmy NAVN i MultiNAVN będące efektami prac nad niniejszą rozprawą. Zaproponowano dwa nowe podejścia: z kryterium zastępczym (MultiNAVN-Z) oraz losowym wyborem kryterium (MultiNAVN-L). Za pomocą wielu eksperymentów wykazano, że zaproponowane algorytmy z powodzeniem wyznaczają drogi dla rzeczywistych map drogowych, przy czym wyniki są porównywalne, a nierzadko lepsze od rezultatów uzyskiwanych za pomocą innych algorytmów.

W celu realizacji postawionych w rozprawie celów:

- W rozdziale 3 przeanalizowano wielokryterialny problem wyszukiwania najkrótszej drogi w grafie oraz różne podejścia do rozwiązywania problemu.
- Zaprezentowano nową wersję algorytmu mrowiskowej nawigacji samochodowej NAVN (rozdział 5.2), która w stosunku do algorytmu znanego z literatury umożliwia wyznaczanie rozwiązań dla dużych, rzeczywistych map drogowych. Kluczową zmianą było wprowadzenie powrotu zablokowanych mrówek oraz dodatkowego zmniejszania śladu feromonowego na krawędziach, po których następuje cofanie się mrówki. Równocześnie nowy algorytm zwraca przybliżony zbiór rozwiązań niezdominowanych, a nie jedno rozwiązanie kompromisowe.

- Zaproponowano dwie wersje algorytmu MultiNAVN: z kryterium zastępczym oraz losowym wyborem kryterium, które zostały poddane eksperymentom na danych rzeczywistych. Wykorzystano cztery kryteria oceny rozwiązań: długość drogi, szerokość drogi, liczba skrzyżowań oraz bezpieczeństwo. Dane dotyczące trzech pierwszych kryteriów pozyskano z bazy danych systemu OpenStreetMap, a jako kryterium bezpieczeństwa wykorzystano informacje o wypadkach i kolizjach z rejestru Policji (rozdział 6.1). Wyniki eksperymentów przeanalizowano pod kątem następujących aspektów:
  - wpływ wartości parametrów algorytmów mrowiskowych ($\alpha, \beta, \tau_0, q_0, \rho$) na jakość otrzymanych wyników (rozdział 6.3),
  - możliwości uzyskiwania aproksymacji możliwie najbardziej zbliżonych do pełnego zbioru rozwiązań przez poszczególne wersje algorytmu NAVN: z kryterium zastępczym oraz losowym wyborem kryterium (rozdział 6.5),
  - porównanie wyników z klasycznym algorytmem Label Setting.
- Zaimplementowano oryginalne, zaproponowane w rozprawie algorytmy mrowiskowe oraz algorytmy referencyjne jak również narzędzia pomocnicze wykorzystywane podczas eksperymentów. Oprogramowanie zostało przygotowane w języku programowania Java, a zawartość plików źródłowych liczyła w sumie ponad 5 tysięcy linii kodu.
- Opracowano narzędzia umożliwiające wykorzystanie rzeczywistych danych kartograficznych z systemu OpenStreetMap i uzupełnienie tych danych informacjami o wypadkach i kolizjach z systemu SEWiK. Narzędzia zostały wykonane w postaci skryptów oraz wbudowanych procedur motoru bazy danych PostgreSQL.
- W rozdziałach 6.3 i 6.4 opisano badania eksperymentalne, których celem było wyznaczenie parametrów algorytmów mrowiskowych (τ_0, α, β, q_0, ρ, ω_b, ω_p) oraz liczby mrówek (N_m) optymalnych dla wykorzystywanych zestawów danych wejściowych.
- Na podstawie przeprowadzonych i przedstawionych w rozdziale 6.5 analiz porównawczych badanych algorytmów MultiNAVN-L i MultiNAVN-Z, można stwierdzić, że:
  - wersja algorytmu z losowym wyborem kryterium oceny wymaga większej liczby mrówek, ale zapewnia przy tym lepszą jakość wyznaczanych zbiorów rozwiązań. Charakteryzują się one zarówno większą licznością zbiorów, jak i mniejszą odległością (mierzoną metryką Hausdorffa) od pełnych zbiorów rozwiązań wyznaczanych przez algorytmy dokładne,
  - wersja algorytmu z zastosowaniem kryterium zastępczego wyznacza mniej liczne zbiory rozwiązań o większej odległości od zbiorów pełnych (o większej wartości metryki Hausdorffa), ale ponieważ wymaga zastosowania mniejszej liczby mrówek, wyznacza rozwiązania w krótszym czasie.
- Zaproponowane algorytmy mrowiskowe MultiNAVN-Z i MultiNAVN-L charakteryzują się znacznie szerszym zakresem zastosowania na rzeczywistych mapach drogowych w porównaniu do klasycznego algorytmu Label Setting, którego zakres

zastosowań jest ograniczony z uwagi na dużą złożoność obliczeniową (tabele 6.25 i 6.31).

- W rozdziale 7 zaprezentowano udaną adaptację zaproponowanego algorytmu dla architektury CUDA oraz interfejsu programistycznego OpenCL, a także przeprowadzono eksperymenty na rzeczywistych mapach drogowych. Algorytm OCLAVN oraz wymagane narzędzia do przeprowadzenia eksperymentów zostały zaimplementowane w języku programowania C/C++, a pełne źródła liczyły ponad 1000 linii kodu. Na podstawie badań stwierdzono, że algorytm OCLAVN jest dobrze przystosowany do wykorzystania potencjału nowoczesnych procesorów GPU.

Podsumowując, wszystkie cele postawione na początku rozprawy zostały zrealizowane. Umożliwiło to potwierdzenie tezy rozprawy mówiącej, że problem znajdowania optymalnej drogi w nawigacji samochodowej z uwzględnieniem wielu kryteriów może być rozwiązywany za pomocą algorytmów mrowiskowych na rzeczywistych danych nawigacyjnych skuteczniej niż za pomocą algorytmu Label Setting.

Niezależnie od wykazanych możliwości zaproponowanych algorytmów mrowiskowej nawigacji samochodowej badania algorytmów trwają. Można zasygnalizować następujące kierunki dalszych badań:

- Zbadanie efektywności algorytmu działającego z wykorzystaniem niezależnych kast mrówek [16] optymalizujących poszczególne kryteria.

- Algorytmy NAVN, które są zrealizowane zgodnie z podejściem Pareto wyznaczają zbiór rozwiązań niezdominowanych. W wielu zastosowaniach praktycznych konieczne jest wyznaczenie jednego rozwiązania, zgodnego np. z preferencjami użytkownika lub hierarchią ważności kryteriów. Możliwe są badania w kierunku określenia zakresu stosowania różnych metod wyboru rozwiązania końcowego spośród zbioru rozwiązań otrzymanych jako wynik działania algorytmu optymalizacyjnego.

- Zastosowanie zaproponowanych algorytmów do rozwiązywania innych, pokrewnych problemów optymalizacji wielokryterialnej.

Problematyka związana z nawigacją samochodową jest w ostatnich czasach bardzo popularna, m.in. ze względu na szybki rozwój przenośnych urządzeń wykorzystujących różne sposoby geopozycjonowania. W związku z tym na szczególne podkreślenie zasługuje fakt rozwinięcia nowych efektywnych metod rozwiązywania tego problemu, które potrafią wyznaczyć rozwiązania dla rzeczywistych map drogowych z wykorzystaniem wielu kryteriów.

Bibliografia

[1] S. Alupoaei and S. Katkoori. Ant colony system application to macrocell overlap removal. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 12(10):1118–1123, 2004.

[2] J. S. Angelo and H. J. Barbosa. On ant colony optimization algorithms for multiobjective problems. *ANT COLONY OPTIMIZATION METHODS AND APPLICATIONS*, page 53, 2011.

[3] J. Bang-Jensen and G. Z. Gutin. *Digraphs: theory, algorithms and applications*. Springer, 2009.

[4] J. Bautista and J. Pereira. Ant algorithms for assembly line balancing. In *Ant Algorithms*, pages 65–75. Springer, 2002.

[5] R. Bellman. On a routing problem. Technical report, DTIC Document, 1956.

[6] A. Berrichi, F. Yalaoui, L. Amodeo, and M. Mezghiche. Bi-objective ant colony optimization approach to optimize production and maintenance scheduling. *Computers & Operations Research*, 37(9):1584–1596, 2010.

[7] L. Bianchi, L. M. Gambardella, and M. Dorigo. An ant colony optimization approach to the probabilistic traveling salesman problem. In *Parallel Problem Solving from Nature—PPSN VII*, pages 883–892. Springer, 2002.

[8] M. Blesa and C. Blum. Ant colony optimization for the maximum edge-disjoint paths problem. In *Applications of Evolutionary Computing*, pages 160–169. Springer, 2004.

[9] C. Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life reviews*, 2(4):353–373, 2005.

[10] C. Blum. Beam-aco—hybridizing ant colony optimization with beam search: An application to open shop scheduling. *Computers & Operations Research*, 32(6):1565–1591, 2005.

[11] C. Blum and M. Sampels. An ant colony optimization algorithm for shop scheduling problems. *Journal of Mathematical Modelling and Algorithms*, 3(3):285–308, 2004.

[12] M. Boryczka. *Programowanie mrowiskowe w procesie aproksymacji funkcji*. Wydawnictwo Uniwersytetu Śląskiego, 2006.

[13] M. Boryczka and W. Bura. *System mrowiskowy w nawigacji samochodowej*, volume 1 of *Systemy Wspomagania Decyzji*, section XX, pages 219–237. Uniwersytet Śląski, 2010.

[14] M. Boryczka and W. Bura. *System mrowiskowy w nawigacji samochodowej - wersja równoległa*, volume 1 of *Systemy Wspomagania Decyzji*, section XVI, pages 169–183. Uniwersytet Śląski, 2011.

[15] M. Boryczka and W. Bura. Ant colony optimization for the multi-criteria vehicle navigation problem. In *Agent-Based Optimization*, pages 29–53. Springer, 2013.

[16] U. Boryczka. *Algorytmy optymalizacji mrowiskowej*. Wydawnictwo Uniwersytetu Śląskiego, 2006.

[17] T. N. Bui and J. R. Rizzo Jr. Finding maximum cliques with distributed ants. In *Genetic and Evolutionary Computation–GECCO 2004*, pages 24–35. Springer, 2004.

[18] B. Bullnheimer, R. F. Hartl, and C. Strauss. A new rank based version of the ant system. a computational study. 1997.

[19] W. Bura and M. Boryczka. Ant colony system in ambulance navigation. *Journal of Medical Informatics & Technologies*, 16:115–124, oct 2010.

[20] W. Bura and M. Boryczka. The parallel ant vehicle navigation system with CUDA technology. In P. Jędrzejowicz, N. Nguyen, and K. Hoang, editors, *Computational Collective Intelligence. Technologies and Applications*, volume 6923 of *Lecture Notes in Computer Science*, pages 505–514. Springer Berlin Heidelberg, 2011.

[21] W. Bura and M. Boryczka. Ant colony optimization for the pareto front approximation in vehicle navigation. *Computational Collective Intelligence. Technologies and Applications*, pages 493–502, 2012.

[22] W. Bura and M. Boryczka. *Optymalizacja mrowiskowa w aproksymacji frontu Pareto*, volume 1 of *Systemy Wspomagania Decyzji*, section XIX, pages 223–235. Uniwersytet Śląski, 2012.

[23] C. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007.

[24] O. Cordon, I. F. de Viana, F. Herrera, and L. Moreno. A new aco model integrating evolutionary computation concepts: The best-worst ant system. 2000.

[25] O. Cordon, F. Herrera, and T. Stützle. A review on the ant colony optimization metaheuristic: Basis, models and new trends. *Mathware & Soft Computing*, 9:141–175, 2002.

[26] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Wprowadzenie do algorytmów*. Wydawnictwa Naukowo-Techniczne, 1997.

[27] F. Correia and T. Vazao. Simple ant routing algorithm. In *Information Networking, 2008. ICOIN 2008. International Conference on*, pages 1–8. IEEE, 2008.

[28] P. Corry and E. Kozan. Ant colony optimisation for machine layout problems. *Computational optimization and applications*, 28(3):287–310, 2004.

[29] D. Costa and A. Hertz. Ants can colour graphs. *Journal of the operational Research Society*, 48(3):295–305, 1997.

[30] J. M. Coutinho-Rodrigues, J. Clımaco, and J. R. Current. An interactive bi-objective shortest path approach: searching for unsupported nondominated solutions. *Computers & Operations Research*, 26(8):789–798, 1999.

[31] R. G. Cox. Routing and scheduling of hazardous materials shipments: algorithmic approaches to managing spent nuclear fuel transport. Technical report, Cornell Univ., Ithaca, NY (USA), 1984.

[32] J. Craveirinha, R. Girao-Silva, J. Clımaco, and L. Martins. A hierarchical multiobjective routing model for mpls networks with two service classes–analysis and resolution approach. *Res. Rep*, 5, 2007.

[33] Z. Czech. *Wprowadzenie do obliczeń równoległych*. Wydawnictwo Naukowe PWN, 2010.

[34] M. Den Besten, T. Stützle, and M. Dorigo. Ant colony optimization for the total weighted tardiness problem. In *Parallel Problem Solving from Nature PPSN VI*, pages 611–620. Springer, 2000.

[35] J.-L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of insect behavior*, 3(2):159–168, 1990.

[36] G. Di Caro and M. Dorigo. Mobile agents for adaptive routing. In *System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on*, volume 7, pages 74–83. IEEE, 1998.

[37] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[38] K. Doerner, W. J. Gutjahr, R. F. Hartl, C. Strauss, and C. Stummer. Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals of Operations Research*, 131(1-4):79–99, 2004.

[39] M. Dorigo. Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano, Italy*, 1992.

[40] M. Dorigo and L. Gambardella. A study of some properties of Ant-Q. *Parallel Problem Solving from Nature—PPSN IV*, pages 656–665, 1996.

[41] M. Dorigo and L. Gambardella. Ant colonies for the travelling salesman problem. *BioSystems*, 43(2):73–81, 1997.

[42] M. Dorigo and L. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*, 1(1):53–66, 1997.

[43] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(1):29–41, 1996.

[44] M. Dorigo, V. Maniezzo, A. Colorni, et al. Positive feedback as a search strategy. *Unpublished manuscript*, 1991.

[45] R. Engelking. *Biblioteka matematyczna*. PWN, 1975.

[46] H. Eschenauer, J. Koski, and A. Osyczka. Multicriteria design optimization, 1990.

[47] R. W. Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[48] D. Ford and D. R. Fulkerson. *Flows in networks*. Princeton university press, 2010.

[49] G. D. Forney Jr. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[50] C. Gagné, W. Price, and M. Gravel. Comparing an aco algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times. *Journal of the Operational Research Society*, pages 895–906, 2002.

[51] L. M. Gambardella and M. Dorigo. An ant colony system hybridized with a new local search for the sequential ordering problem. *INFORMS Journal on Computing*, 12(3):237–255, 2000.

[52] L. M. Gambardella, A. E. Rizzoli, F. Oliverio, N. Casagrande, A. V. Donati, R. Montemanni, and E. Lucibello. Ant colony optimization for vehicle routing in advanced logistics systems. In *Proceedings of the International Workshop on Modelling and Applied Simulation (MAS 2003)*, pages 3–9, 2003.

[53] L. M. Gambardella, É. Taillard, and G. Agazzi. Macs-vrptw: A multiple colony system for vehicle routing problems with time windows. In *New ideas in optimization*. Citeseer, 1999.

[54] X. Gandibleux, F. Beugnies, and S. Randriamasy. Martins' algorithm revisited for multi-objective shortest path problems with a maxmin cost function. *4OR*, 4(1):47–59, 2006.

[55] X. Gandibleux, X. Delorme, and V. T'Kindt. An ant colony optimisation algorithm for the set packing problem. In *Ant Colony Optimization and Swarm Intelligence*, pages 49–60. Springer, 2004.

[56] C. García-Martínez, O. Cordón, and F. Herrera. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria tsp. *European Journal of Operational Research*, 180(1):116–148, 2007.

[57] M. Gary and D. Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

[58] F. Ghezail, H. Pierreval, and S. Hajri-Gabouj. Multi objective optimization using ant colonies. In *Complex Systems and Self-organization Modelling*, pages 65–70. Springer, 2009.

[59] K. Ghoseiri and B. Nadjari. An ant colony optimization algorithm for the bi-objective shortest path problem. *Applied Soft Computing*, 10(4):1237–1246, 2010.

[60] S. Goss, S. Aron, J.-L. Deneubourg, and J. M. Pasteels. Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, 76(12):579–581, 1989.

[61] S. Goss, R. Beckers, J.-L. Deneubourg, S. Aron, J. Pasteels, and R. N. Hughes. How trail laying and trail following can solve foraging problems for ant colonies. *Behavioural Mechanisms of Food Selection*, pages 661–678, 1990.

[62] J. Gottlieb, M. Puchta, and C. Solnon. A study of greedy, local search, and ant colony optimization approaches for car sequencing problems. In *Applications of evolutionary computing*, pages 246–257. Springer, 2003.

[63] J. Granat and F. Guerriero. The interactive analysis of the multicriteria shortest path problem by the reference point method. *European Journal of Operational Research*, 151(1):103–118, 2003.

[64] P.-P. Grassé. La reconstruction du nid et les coordinations interindividuelles chezbellicositermes natalensis etcubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insectes sociaux*, 6(1):41–80, 1959.

[65] C. Guéret, N. Monmarché, and M. Slimane. Ants can play music. In *Ant Colony Optimization and Swarm Intelligence*, pages 310–317. Springer, 2004.

[66] F. Guerriero and R. Musmanno. Label correcting methods to solve multicriteria shortest path problems. *Journal of Optimization Theory and Applications*, 111(3):589–613, 2001.

[67] M. Guntsch and M. Middendorf. Pheromone modification strategies for ant algorithms applied to dynamic tsp. In *Applications of Evolutionary Computing*, pages 213–222. Springer, 2001.

[68] M. Guntsch and M. Middendorf. Solving multi-criteria optimization problems with population-based aco. In *Evolutionary Multi-Criterion Optimization*, pages 464–478. Springer, 2003.

[69] Z. Guo, X. Song, and P. Zhang. The aco algorithm for container transportation network of seaports. In *Proceedings of the Eastern Asia Society for Transportation Studies*, volume 5, pages 581–591, 2005.

[70] P. Hansen. Bicriterion path problems. *Multiple Criteria Decision Making: Theory and Applications*, 177:109–127, 1980.

[71] R. Hartley. Vector optimal routing by dynamic programming. *Mathematics of multiobjective optimization*, 289:215–224, 1985.

[72] K. Jacek. Fraktale i chaos. *Wydawnictwo Naukowo-Techniczne*, 2007.

[73] D. B. Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM (JACM)*, 24(1):1–13, 1977.

[74] O. Karpenko, J. Shi, and Y. Dai. Prediction of mhc class ii binders using the ant colony search strategy. *Artificial Intelligence in Medicine*, 35(1):147–156, 2005.

[75] R. Kumar and N. Banerjee. Multicriteria network design using evolutionary algorithm. In *Genetic and Evolutionary Computation—GECCO 2003*, pages 2179–2190. Springer, 2003.

[76] L. Lin and M. Gen. Multiobjective genetic algorithm for solving network design problem. *Faji Shisutemu Shinpojiumu Koen Ronbunshu (CD-ROM)*, 20:290–291, 2004.

[77] M. López-Ibánez. *Multi-objective ant colony optimization*. PhD thesis, Diploma thesis, Intellectics Group, Computer Science Department, Technische Universität Darmstadt, Germany, 2004.

[78] M. López-Ibánez, L. Paquete, and T. Stützle. On the design of aco for the biobjective quadratic assignment problem. In *Ant Colony Optimization and Swarm Intelligence*, pages 214–225. Springer, 2004.

[79] V. Maniezzo. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS Journal on Computing*, 11(4):358–369, 1999.

[80] V. Maniezzo, M. Boschetti, and M. Jelasity. An ant approach to membership overlay design. In *Ant Colony Optimization and Swarm Intelligence*, pages 37–48. Springer, 2004.

[81] V. Maniezzo and A. Colorni. The ant system applied to the quadratic assignment problem. *Knowledge and Data Engineering, IEEE Transactions on*, 11(5):769–778, 1999.

[82] R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[83] E. d. Q. V. Martins and J. Santos. The labeling algorithm for the multiobjective shortest path problem. *Departamento de Matematica, Universidade de Coimbra, Portugal, Tech. Rep. TR-99/005*, 1999.

[84] E. Q. V. Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2):236–245, 1984.

[85] D. Merkle, M. Middendorf, and H. Schmeck. Ant colony optimization for resource-constrained project scheduling. *Evolutionary Computation, IEEE Transactions on*, 6(4):333–346, 2002.

[86] B. Michael. Fractals everywhere, 1988.

[87] R. Michel and M. Middendorf. An island model based ant system with lookahead for the shortest supersequence problem. In *Parallel Problem Solving from Nature—PPSN V*, pages 692–701. Springer, 1998.

[88] L. A. Moncayo-Martínez and D. Z. Zhang. Multi-objective ant colony optimisation: A meta-heuristic approach to supply chain design. *International Journal of Production Economics*, 131(1):407–420, 2011.

[89] P. Mooney and A. Winstanley. An evolutionary algorithm for multicriteria path optimization problems. *International Journal of Geographical Information Science*, 20(4):401–423, 2006.

[90] J. Moss and C. G. Johnson. An ant colony algorithm for multiple sequence alignment in bioinformatics. In *Artificial Neural Nets and Genetic Algorithms*, pages 182–186. Springer, 2003.

[91] J. C. Namorado Climaco and E. Queiros Vieira Martins. A bicriterion shortest path algorithm. *European Journal of Operational Research*, 11(4):399–404, 1982.

[92] R. Ott and M. Longnecker. *An introduction to statistical methods and data analysis*. Cengage Learning, 2008.

[93] J. M. Paixão and J. L. Santos. Labeling methods for the general case of the multi-objective shortest path problem–a computational study. In *Computational Intelligence and Decision Making*, pages 489–502. Springer, 2013.

[94] J. Pangilinan and G. Janssens. Evolutionary algorithms for the multi-objective shortest path problem. *World Academy of Science, Engineering and Technology 25*, 2007.

[95] C. H. Papadimitriou. *Złożoność obliczeniowa*. Wydawnictwa Naukowo-Techniczne, 2002.

[96] V. Pareto. *Cours d'economie politique*. F. Rouge, Lausanne, Switzerland, 1896.

[97] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas. Data mining with an ant colony optimization algorithm. *Evolutionary Computation, IEEE Transactions on*, 6(4):321–332, 2002.

[98] M. Reimann, K. Doerner, and R. F. Hartl. D-ants: savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, 31(4):563–591, 2004.

[99] G. Rozenberg, T. Bck, and J. N. Kok. *Handbook of natural computing*. Springer Publishing Company, Incorporated, 2011.

[100] H. Salehinejad and F. Farrahi-Moghaddam. An ant based algorithm approach to vehicle navigation. In *Proceedings of the First Joint Congress on Fuzzy and Intelligent Systems*, 2007.

[101] H. Salehinejad, F. Pouladi, and S. Talebi. A new route selection system: multiparameter ant algorithm based vehicle navigation approach. In *Computational Intelligence for Modelling Control & Automation, 2008 International Conference on*, pages 1089–1094. IEEE, 2008.

[102] H. Salehinejad and S. Talebi. A new ant algorithm based vehicle navigation system: a wireless networking approach. In *Telecommunications, 2008. IST 2008. International Symposium on*, pages 36–41. IEEE, 2008.

[103] N. Sancho. A new type of multi-objective routing problem. *Engineering optimization*, 14(2):115–119, 1988.

[104] D. Shmoys, J. Lenstra, A. R. Kan, and E. Lawler. *The traveling salesman problem*, volume 12. Wiley, 1985.

[105] A. Shmygelska, R. Aguirre-Hernandez, and H. H. Hoos. An ant colony optimization algorithm for the 2d hp protein folding problem. In *Ant Algorithms*, pages 40–52. Springer, 2002.

[106] A. Shmygelska and H. H. Hoos. An ant colony optimisation algorithm for the 2d and 3d hydrophobic polar protein folding problem. *BMC bioinformatics*, 6(1):30, 2005.

[107] C. A. Silva, T. A. Runkler, J. M. Sousa, and R. Palm. Ant colonies as logistic processes optimizers. In *Ant Algorithms*, pages 76–87. Springer, 2002.

[108] S. S. Skiena. The algorithm design manual. 2008.

[109] K. Socha, M. Sampels, and M. Manfrin. Ant algorithms for the university course timetabling problem with regard to the state-of-the-art. In *Applications of evolutionary computing*, pages 334–345. Springer, 2003.

[110] C. Solnon. Ants can solve constraint satisfaction problems. *Evolutionary Computation, IEEE Transactions on*, 6(4):347–357, 2002.

[111] T. Stützle et al. An ant approach to the flow shop problem. In *Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98)*, volume 3, pages 1560–1564, 1998.

[112] T. Stützle and H. Hoos. Improving the ant system: A detailed report on the max-min ant system. 1996.

[113] T. Stützle and H. Hoos. Max-min ant system and local search for the traveling salesman problem. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 309–314. IEEE, 1997.

[114] T. Stützle and H. H. Hoos. Max–min ant system. *Future generation computer systems*, 16(8):889–914, 2000.

[115] Z. Tarapata. Military route planning in battlefield simulation: effectiveness problems and potential solutions. *Journal of Telecommunications and Information technology*, pages 47–56, 2003.

[116] Z. Tarapata. Selected multicriteria shortest path problems: An analysis of complexity, models and adaptation of standard algorithms. *International Journal of Applied Mathematics and Computer Science*, 17(2):269–287, 2007.

[117] G. Tsaggouris and C. Zaroliagis. Multiobjective optimization: Improved fptas for shortest paths and non-linear objectives with applications. *Theory of Computing Systems*, 45(1):162–186, 2009.

[118] A. Warburton. Approximation of pareto optima in multiple-objective, shortest-path problems. *Operations Research*, 35(1):70–79, 1987.

[119] S. Warshall. A theorem on boolean matrices. *Journal of the ACM (JACM)*, 9(1):11–12, 1962.

[120] J. Widuch. Algorytmy optymalizacji wielokryterialnej w problemach komunikacyjnych. *Rozprawa doktorska, Politechnika Śląska, Gliwice*, 2008.

[121] J. Widuch. A label correcting algorithm for the bus routing problem. *Fundamenta Informaticae*, 118(3):305–326, 2012.

[122] J. Widuch. A label correcting algorithm with storing partial solutions to solving the bus routing problem. *Informatica*, 24(3):461–484, 2013.

[123] A. C. Zecchin, A. R. Simpson, H. R. Maier, and J. B. Nixon. Parametric study for an ant algorithm applied to water distribution system optimization. *Evolutionary Computation, IEEE Transactions on*, 9(2):175–191, 2005.

Spis algorytmów

Spis rysunków

Spis tabel