

POLITECHNIKA BIAŁOSTOCKA

WYDZIAŁ INFORMATYKI

PRACA DYPLOMOWA INŻYNIERSKA

**TEMAT: TEMAT PRACY INŻYNIERSKIEJ /
MAGISTERSKIEJ.**

WYKONAWCA: GAL ANONIM

.....
podpis

PROMOTOR: DR INŻ. DOKTOR INŻYNIER

.....
podpis

BIAŁYSTOK 2021 r.

Załącznik nr 2 do „Zasad postępowania przy przygotowaniu i obronie pracy dyplomowej na PB”

Karta dyplomowa

POLITECHNIKA BIAŁOSTOCKA Wydział..... 	Studia..... stacjonarne/niestacjonarne studia I stopnia/studia II stopnia	Nr albumu studenta..... Rok akademicki..... Kierunek studiów..... Specjalność.....
Imiona i nazwisko studenta		
TEMAT PRACY DYPLOMOWEJ: 		
Zakres pracy: 1. 2. 3. 4.		
Slowa kluczowe (max 5):		
TO JEST SKAN		
..... Imiona i nazwisko, stopień/tytuł promotora - podpis		
Data wydania tematu pracy dyplomowej - podpis promotora	Regulaminowy termin złożenia pracy dyplomowej	Data złożenia pracy dyplomowej - potwierdzenie dziekanatu
..... Ocena promotora Podpis promotora
..... Imiona i nazwisko, stopień/tytuł recenzenta Ocena recenzenta Podpis recenzenta

Subject of diploma thesis

Application of selected algorithms to optimize routes of municipal economy vehicles.

Summary

Nowadays waste generated by residents is one of the problems that government have to deal with in Poland. Segregation and storage is not the only problem that can be encountered. Another difficulty that can be distinguished is their transport. The authors of the thesis proposed to conduct research to optimize vehicles routes acquired from the city of Białystok. Paweł Stypułkowski presented proposals for solving the problem using a genetic algorithm. On the other hand, Kamil Łętowski explored the topic of route optimization using the ant algorithm in more detail. Where the final optimization solutions were presented by Przemysław Noskowicz with greedy algorithms. After the theoretical introduction, the data sets on which the algorithms worked were presented. Next, the results that have been achieved with a particular solution are shown. There was also a list and comparison of the best achievements.

Załącznik nr 4 do „Zasad postępowania przy przygotowaniu i obronie pracy dyplomowej na PB”
Białystok, dnia 05.01.2020 r.

Gal Anonim

Imiona i nazwisko studenta

12345

Nr albumu

informatyka, stacjonarne

Kierunek i forma studiów

dr inż. Doktor Inżynier

Promotor pracy dyplomowej

OŚWIADCZENIE

Przedkładając w roku akademickim 2019/2020 Promotorowi **dr inż. Doktor Inżynier** pracę dyplomową pt.: **Temat pracy**, dalej zwaną pracą dyplomową, **oświadczam, że:**

- 1) praca dyplomowa stanowi wynik samodzielnej pracy twórczej;
- 2) wykorzystując w pracy dyplomowej materiały źródłowe, w tym w szczególności: monografie, artykuły naukowe, zestawienia zawierające wyniki badań (opublikowane, jak i nieopublikowane), materiały ze stron internetowych, w przypisach wskazywałem/am ich autora, tytuł, miejsce i rok publikacji oraz stronę, z której pochodzą powoływane fragmenty, ponadto w pracy dyplomowej zamieściłem/am bibliografię;
- 3) praca dyplomowa nie zawiera żadnych danych, informacji i materiałów, których publikacja nie jest prawnie dozwolona;
- 4) praca dyplomowa dotychczas nie stanowiła podstawy nadania tytułu zawodowego, stopnia naukowego, tytułu naukowego oraz uzyskania innych kwalifikacji;
- 5) treść pracy dyplomowej przekazanej do dziekanatu Wydziału Informatyki jest jednakowa w wersji drukowanej oraz w formie elektronicznej;
- 6) jestem świadomy/a, że naruszenie praw autorskich podlega odpowiedzialności na podstawie przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz. U. z 2019 r. poz. 1231, późn. zm.), jednocześnie na podstawie przepisów ustawy z dnia 20 lipca 2018 roku Prawo o szkolnictwie wyższym i nauce (Dz. U. poz. 1668, z późn. zm.) stanowi przesłankę wszczęcia postępowania dyscyplinarnego oraz stwierdzenia nieważności postępowania w sprawie nadania tytułu zawodowego;
- 7) udzielam Politechnice Białostockiej nieodpłatnej, nieograniczonej terytorialnie i czasowo licencji wyłącznej na umieszczenie i przechowywanie elektronicznej wersji pracy dyplomowej w zbiorach systemu Archiwum Prac Dyplomowych Politechniki Białostockiej oraz jej zwielokrotniania i udostępniania w formie elektronicznej w zakresie koniecznym do weryfikacji autorstwa tej pracy i ochrony przed przywłaszczeniem jej autorstwa.

.....

czytelny podpis studenta

Spis treści

Streszczenie	5
Wstęp	11
1 Ogólny problem	13
1.1 Problem komiwojażera	13
1.2 Metody optymalizacji	15
2 Algorytm genetyczny - Paweł	21
2.1 Wprowadzenie do algorytmu	21
2.2 Parametry wejściowe	24
2.3 Metody krzyżowania	26
2.4 Metody mutacji	29
3 Algorytm mrówkowy - Kamil	31
3.1 Wprowadzenie do algorytmu	31
3.2 Opis działania algorytmu	32
3.3 Feromony	33
4 Algorytmy zachłanne - Przemysław	39
4.1 Algorytm najbliższego sąsiada	39
4.2 Algorytm najmniejszej krawędzi	41
4.3 Algorytm A*	44
4.4 Optymalizacja otrzymanych rozwiązań	45
5 Zbiór danych	49
6 Badania	51
6.1 Trasa I	52
6.2 Trasa II	68
6.3 Trasa III	82

Podsumowanie	97
Bibliografia	102
Spis tabel	104
Spis rysunków	107

Wstęp

Odpady generowane przez mieszkańców są jednym z problemów z jakimi władze muszą sobie radzić w Polsce. Nie jest to problem jedynie wielkich metropolii, ale również mniejszych miasteczek. Ilość produkowanych śmieci jest olbrzymia, a to również powoduje kolejne problemy. Segregacja oraz składowanie to nie jest jedyny problem z jakim można się spotkać. Kolejnym utrudnieniem jaki można wyróżnić jest ich transport.

W 2018 roku w Białymstoku został zorganizowany 24-godzinny Hackaton Miejski w Białostockim Parku Naukowo-Technologicznym. Organizatorzy tego przedsięwzięcia przygotowali dużą ilość merytorycznych materiałów na temat zbiórki odpadów komunalnych w mieście Białystok. Uczestnicy hackatonu zauważali wiele problemów z jakimi boryka się miasto. Jednak największą uwagę organizatorów przykuły różne anomalie.

Jedna z grup zauważała, że trasy jakie pokonują ciężarówki są nieoptimalne. Jako przykład została przedstawiona droga losowej ciężarówki. Można było zauważyc, że przebyta ścieżka przecina się w wielu miejscach. Na podstawie tylko jednego przykładu można dojść do wniosku, że problem jest globalny i może dotyczyć wszystkich tras. Organizatorzy przedsięwzięcia przyznali, że nie przykładaли do tego aspektu uwagi, ale może to być kluczowe w oszczędności czasu oraz pieniędzy.

Przy pomocy komputerów oraz odpowiednio napisanego programu można wyznaczyć ścieżki, które będą spełniać założenia biznesowe oraz będą odpowiednio optymalne. Wyzwaniem jest jedynie znalezienie odpowiedniego rozwiązania które nada prawidłowy kierunek przeprowadzanym optymalizacjom. Na przestrzeni lat zostało opracowanych wiele rozwiązań. Niestety, nie każde rozwiązanie jest w stanie poradzić sobie z konkretnymi danymi w ten sam sposób.

Celem pracy jest zastosowanie i porównanie algorytmów optymalizacji w celu znalezienia najkrótszych tras przejazdu ciężarówek z odpadami komunalnymi. Dla algorytmów zostaną znalezione takie parametry wejściowe aby wynik ich prac był jak najbardziej optymalny. Po przeprowadzeniu badań, rozwiązania zostaną porównane pod kątem optymalizacji oraz szybkości działania.

W rozdziale 2 Paweł Stypułkowski przedstawił algorytm genetyczny. W tym rozdziale można znaleźć informacje w jaki sposób działa to rozwiązanie. Dogłębnie zostały opisane

składowe tego algorytmu. Można się z tego dowiedzieć o parametrach wejściowych, metodach krzyżowania i metodach mutacji.

Kamil Łętowski w rozdziale 3 zaproponował optymalizację tras przy pomocy algorytmu mrówkowego. Oprósz wprowadzenia do algorytmu został zawarty opis działania. Autor swoją uwagę skupił również na bardzo ważnej składowej tego rozwiązania jakim są feromony wykorzystywane przez mrówki.

Algorytmy zachłanne to rozwiązanie jakie zaproponował Przemysław Noskowicz w rozdziale 4. Autor tego rozdziału opisuje kilka wybranych metod optymalizacji. Wśród nich można wymienić algorytm najbliższego sąsiada, algorytm najmniejszej krawędzi, algorytm a*. Na sam koniec za pomocą różnych metod algorytmy te zostały ponownie zoptymalizowane.

Po teoretycznym wstępnie zostały przedstawione zbiory danych na których pracowały algorytmy. W dalszej kolejności zostały pokazane wyniki jakie udało się osiągnąć poszczególnemu rozwiązaniu. Nie zabrakło również zestawienia oraz porównania najlepszych osiągnięć.

1. Ogólny problem

Badany problem jest w informatyce nazywany problemem komiwojażera. W tym rozdziale zostanie on omówiony oraz metody optymalizacji.

1.1 Problem komiwojażera

Problem komiwojażera (ang. travelling salesman problem - TSP) należy do rodziny problemów NP-trudnych. Znalezienie najlepszego rozwiązania jest trudne i fascynuje naukowców od wielu lat. Niektórzy poddają pod wątpliwość znalezienie efektywnego rozwiązania, czyli takiego którego czas działania jest maksymalnie wielomianowy. Aktualnie istnieje wiele rozwiązań tego problemu, a proponowane podejścia są bardzo interesujące. Niektóre z nich bazują na lokalnych przeszukiwaniach grafu, a inne opierają się na przykładach które występują w przyrodzie.

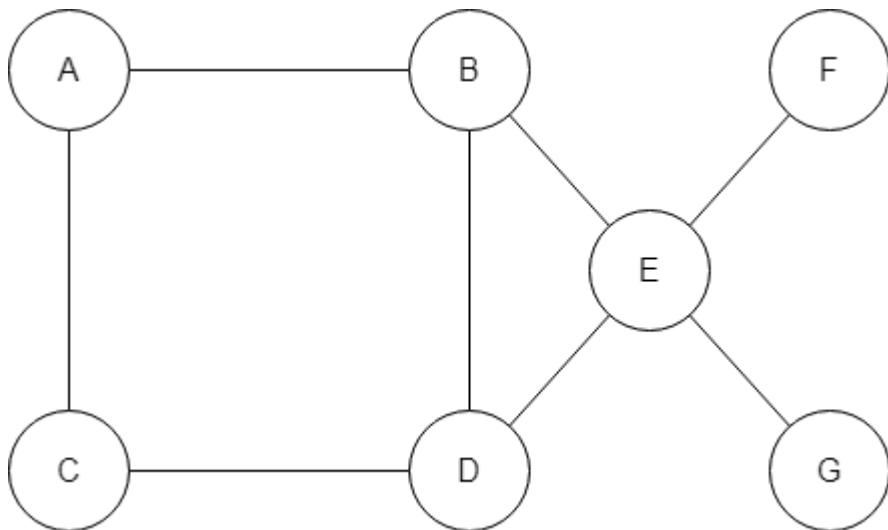
Podobnym problemem do TSP jest problem konika szachowego. Problem ten również należy do rodziny problemów NP-zupełnym. Już w XVIII wieku badania nad tym problemem rozpoczął Euler [5]. Rozwiązanie tego problemu polega na znalezieniu ścieżki jaką ma przebyć konik szachowy, tak aby odwiedzić każde pole na szachownicy tylko i wyłącznie raz. Skoczek porusza się po planszy zgodnie z określonym ruchem, a plansza szachowa może mieć różny rozmiar. Konik porusza się aż do momentu odwiedzenia wszystkich pól lub do momentu w którym nie ma możliwości odwiedzenia kolejnego pola.

Optymalizacja tras od zawsze jest obecna w historii ludzkości. Nawet takie trywialne problemy jak podróz pomiędzy 3 miejscowościami może zostać sklasyfikowany jako problem komiwojażera. Chociaż dokładne wskazanie na źródło problemu TSP nie jest znane, to już w 1832 roku w przewodniku dla podróżujących po Niemczech i Szwajcarii została zawarta informacja o optymalizacji trasy przejazdu. Nie ma tam zawartych żadnych teorii matematycznych w związku z czym nie można uznać tego dzieła za początek rozważań nad problemem komiwojażera.[15]

W XIX wieku William Hamilton stworzył fundamenty pod definicję TSP. W rozwiązaniu problemu komiwojażera należy znaleźć cykl w grafie. W skład takiego cyklu musi zostać zawarty każdy z wierzchołków. Każdy z wierzchołków może znajdować się w rozwiązaniu dokładnie tylko raz. Cykl który spełnia wymieniony warunek jest cyklem

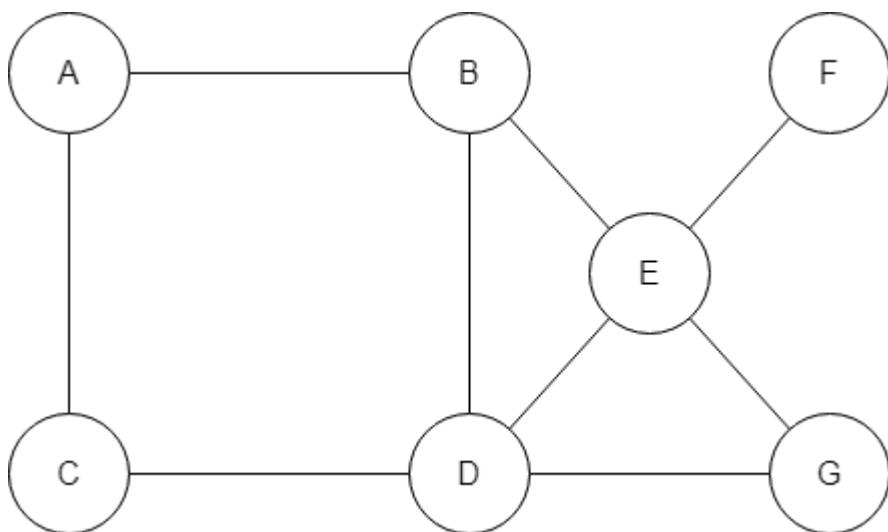
Hamiltona. Jeśli w grafie można wyróżnić cykl z opisanymi powyżej warunkami, to graf jest grafem Hamiltonowskim.

Na 1.1 został przedstawiony graf bez cyklu Hamiltona. W grafie tym nie można znaleźć takiego połączenia które zawiera wszystkie wierzchołki przechodząc przez każdy z nich dokładnie raz. Istnieje możliwość przejścia przez wszystkie wierzchołki jedynie po powtórnym odwiedzeniu przynajmniej jednego wierzchołka.



Rysunek 1.1: Graf bez cyklu Hamiltona.

Graf z 1.2 posiada połączenie krawędzi dzięki któremu można przejść po wszystkich wierzchołkach dokładnie raz. Takie przejście jest właśnie cyklem Hamiltona w związku z czym graf jest Hamiltonowski. Wyruszając przykładowo z punktu F możemy przejść kolejno do E - G - D - B - A - C. W ten sposób odwiedzimy wszystkie wierzchołki tylko raz.



Rysunek 1.2: Graf z cyklem Hamiltona.

W latach 30 XX wieku Merrill Meeks Flood rozpoczął rozważania nad optymalizacją przejazdu autobusów szkolnych. Działalność tą możemy uznać za początek pracy nad problemem TSP. Wraz z upływem czasu zainteresowanie problemem optymalizacyjnym narastało, a co za tym idzie powstawały nowe pomysły na algorytmy. Jednak żaden z pomysłów nie jest w stanie zaproponować dokładnego rozwiązania które jest w stanie przedstawić rezultat w czasie wyrażonym za pomocą wielomianu.

1.2 Metody optymalizacji

Jednym z proponowanych rozwiązań jest algorytm Helda Karpa który jest oparty na programowaniu dynamicznym. Złożoność pamięciowa tego algorytmu wynosi $\theta(2n^n)$, a czasowa $\theta(n^2 * 2^n)$. W algorytmie tym na każdym kroku wyznaczamy punkt który powinien być przedostatni na trasie. Aby wyznaczyć poprzednika należy skorzystać ze wzoru w którym poszukiwana jest najmniejsza wartość pomiędzy punktami.

Innym przykładem, który można wykorzystać do rozwiązywania problemu komiwojażera jest algorytm najbliższego sąsiada. Rozwiązanie to wykorzystuje strategię zachłanną. W algorytmie szukamy aktualnie najlepszego ruchu. W tym celu przeszukiwani są jedynie sąsiedzi którzy są najbliżej aktualnego punktu. Złożoność takiego algorytmu jest szacowana na $\theta(n^2)$.

Oprócz standardowych przeszukiwań zbiorów na przestrzeni lat pojawiły się propozycje które wprowadzają elementy losowości. Przykładem takiego rozwiązania mogą być algorytm genetyczny oraz algorytm mrówkowy. Należą one do grup algorytmów heurystycznych, czyli do takich, które nie dają gwarancji znalezienia dokładnego rozwiązania.

W pracy zostaną zbadane rozwiązania problemu za pomocą algorytmu genetycznego, mrówkowego oraz zachłannego. W pozostałych podrozdziałach zostaną opisane ich klasyczne wersje.

1.2.1 Algorytm genetyczny - Paweł

Algorytm genetyczny (ang. *GA - Genetic Algorithm*) polega na symulacji procesów genetycznych zachodzących w populacjach osobników, stosuje się je głównie przy zadaniach optymalizacyjnych. W przyrodzie większość gatunków od wieków, w tym przede wszystkim i człowiek, w kolejnych swoich pokoleniach się rozwinęło i dostosowało do otaczających warunków na świecie. Gdy jakiś osobnik urodzi się z cechą, która jest przydatna w

przetrwaniu, przekażę tę cechę kolejnym pokoleniom. Najsłabsze osobniki w populacji mają zarówno mniejsze szanse na przetrwanie oraz rozmnożenie, czyli przekazanie swoich cech potomkom. W przyrodzie zazwyczaj silniejsi wygrywają. W latach 50 XX wieku zaczęto symulować te procesy w informatyce. W latach 60 John Henry Holland zastosował algorytm genetyczny przy pracach nad systemami adaptacyjnymi, a w 1975 wydał książkę *Adaptation in Natural and Artificial Systems*, w której to opisał.[16]. Algorytm genetyczny poszukuje najlepsze rozwiązania, wśród populacji potencjalnych rozwiązań. Jest to główna cecha, która odróżnia go od tradycyjnych metod optymalizacji. Każde rozwiązanie ulega ocenie na podstawie jego dopasowania do problemu - funkcja przystosowania. Algorytm na populacji symuluje zjawiska ewolucyjne, krzyżując i mutując rozwiązania, stosując probabilistyczne reguły wyboru. W każdym takim nowym pokoleniu najsłabsi są usuwani, więc w kolejnych etapach populacja składa się z coraz to lepszych rozwiązań [13]. Kolejne pokolenia są generowane, aż zostanie spełniony warunek zakończenia. Może to być z góry ustalony czas trwania, ilość kolejnych pokoleń lub brak poprawy wśród nowych rozwiązań.

Algorytmy genetyczne i jego odmiany zrewolucjonizowały systemy informatyczne. Nie są one algorytmami, które wyliczają dokładne wyniki, ale przy odpowiedniej implementacji i ustaleniu parametrów wejściowych, pozwalają osiągnąć dobre rezultaty. Bardzo ważny jest czas znalezienia takiego rozwiązania. Jeśli rozwiązanie idealne obliczane jest przez algorytm tradycyjny w ciągu 24 godziny, a algorytm genetyczny w trakcie kilku minut znajdzie rozwiązanie będące w sąsiedztwie, swoją efektywnością wygra ten drugi. Użytkownik nie będzie chciał czekać całej doby na wynik swojego zapytania. Oczywiście, algorytmy genetyczne też mogą znaleźć najlepsze rozwiązanie. Kwestią ograniczenia jest zawsze czas.

Medycyna jest jedną z ważniejszych dziedzin, gdzie wykorzystuje się algorytmy genetyczne. Zbiory danych i przestrzeń przeszukiwań jest ogromna i złożona. Zazwyczaj w oparciu o te informacje, lekarz musi podjąć decyzję, czy np. nowotwór jest złośliwy, czy ma łagodny przebieg. Algorytm genetyczny pozwala wspomóc lekarza przy podejmowaniu takich decyzji, przetwarzając i analizując te ogromne zbiory. [9]. Kolejną gałęzią gospodarki, w których zastosowanie znajduje algorytm genetyczny, jest przemysł spożywczy, a konkretnie optymalizacja linii produkcyjnych. Za ich pomocą algorytmu genetycznego wyznacza się parametry takie jak temperatura, ciśnienie lub zapotrzebowanie mocy. Dzięki temu wszystkie procesy technologiczne zachodzące w maszynach i urządzeniach są wydajne

i zoptymalizowane[3]. Algorytmy genetyczne często stosuje się jako wskazanie punktów początkowych w innych metodach optymalizacyjnych. Poza podanymi przykładami, znajdują one zastosowanie praktycznie wszędzie: ekonomia, giełda, przemysł lotniczy, kombinatoryka, sieci komputerowe, zarządzanie łańcuchem dostaw, a także ustalanie czasu reklamy w telewizji [8].

1.2.2 Algorytm mrówkowy

Obserwacje nad zachowaniami w przyrodzie wielokrotnie miały wpływ na rozwój nowych rozwiązań. Tak jak w przypadku algorytmu genetycznego, tak i w przypadku algorytmu mrówkowego pomysł został zaczerpnięty z przyrody [12]. Dokładne działanie algorytmu mrówkowego wzoruje się na zachowaniu kolonii mrówek. Dzięki pracy zespołowej, owady te są w stanie wypracować optymalną ścieżkę między siedliskiem a znalezionym pokarmem.

Dla niejednego gatunku problematyczne mogłyby być odtworzenie przebytej ścieżki. Na początku należałyby zadać sobie pytanie, w jaki sposób te niewielkich rozmiarów owady są w stanie znaczco ułatwić sobie przetrwanie? Istotną rolę odgrywa tutaj wspomniana już praca zespołowa. To dzięki niej mrówki są w stanie optymalizować trasę. Innym ważnym czynnikiem determinującym poprawę ścieżki jest zapach, jaki zostawiają mrówki.

Zapach nie jest niczym innym jak feromonem wytwarzanym przez mrówki. Dzięki po-zostawionemu zapachowi mrówki identyfikują, w jaki sposób poruszali się ich poprzednicy w związku z czym odtworzenie trasy nie stanowiło już poważnego problemu. Przy kolejnych iteracjach, kolonia próbuje optymalizować aktualną trasę. W tym celu również wykorzystuje zapach pozostawiony w poprzednich przejściach. Ścieżka jest losowo zmieniana w celu optymalizacji. Jeśli modyfikacja przyniosła oczekiwany efekt, to trasa zostaje zmieniona.

Algorytm mrówkowy znajduje swoje zastosowanie w rozwiązywaniu innych problemów. Problem plecakowy jest jednym z takich przykładów. Pojawia się on najczęściej przy optymalnym zarządzaniu zasobami. Mamy dany zbiór przedmiotów, z czego każdy z nich posiada określony ciężar i wartość. Do plecaka musimy załadować przedmioty o jak największej wartości. Naszym ograniczeniem jest jednak łączny ciężar przedmiotów, które możemy udźwignąć.

Algorytm mrówkowy wygląda w tym przypadku podobnie. Na początku jest N agentów – mrówek. Każdy agent iteracyjnie poszukuje jak najlepszego rozwiązania. Po

każdej iteracji można wyróżnić trzy typy rozwiązań: rozwiązanie pośrednie, rozwiązanie częściowe lub stan. Agenci w celu znalezienia rozwiązania wykorzystują swoje naturalne umiejętności, czyli zostawiają na wszystkich obiektach w plecaku feromony. Dzięki lotności feromonów mrówki są w stanie identyfikować przedmioty, które korzystniej jest zabrać.

Krzysztof Schiff w artykule *Ant colony optimization algorithm for the 0-1 knapsack* [14] przedstawił rozwiązanie problemu plecakowego. Do wyboru najlepszego rozwiązania wykorzystane zostały trzy metody. Zgodnie z przyjętą przez autora konwencją metody mają odpowiednie nazwy: AKA1, AKA2 oraz AKA3. Za wybór najlepszego rozwiązania odpowiadają wzory:

$$AKA1 = \frac{z}{\frac{w}{V}}$$

$$AKA2 = \frac{z}{w^2}$$

$$AKA3 = \frac{z}{\frac{w}{C}}$$

gdzie:

- z jest zyskiem wybranego obiektu;
- w jest wagą wybranego obiektu;
- V jest aktualną ładownością plecaka;
- z jest zyskiem wybranego obiektu;
- C jest całkowitą wagą plecaka.

Problem komiwojażera i problem plecakowy są problemami kombinatorycznymi. Ich rozwiązanie polega na poszukiwaniu optymalnej ścieżki na grafie pełnym. Inna odmiana problemu jest kolorowania grafu. W tej wariacji problem jest od razu zadany na grafie. Dla wszystkich wierzchołków w grafie należy dobrać takie kolory, aby żadne dwa sąsiednie wierzchołki nie miały tego samego koloru [4].

Mrówki nie działają bezpośrednio na grafie początkowym, ponieważ graf ten nie musi być grafem pełnym. Należy stworzyć dla mrówek alternatywę podobną do oryginału z zachowaniem takiego samego zbioru wierzchołków ale z pełnymi krawędziami. Następnie należy dobrać numeryczne wartości odpowiadające konkretnym kolorom. Jeśli mrówka odwiedzi dany wierzchołek, to zostaje on pokolorowany na kolor, który nie został dotychczas użyty do kolorowania jego sąsiadów.

Tak jak w przypadku poprzednich algorytmów, wykorzystywane są zapachy pozostawiane przez mrówki. Liczba użytych unikalnych kolorów byłaby odwrotnie proporcjonalna do ilości feromonów. W efekcie czego heurystyka byłaby odwrotnie proporcjonalna do wykorzystanych kolorów po kolejnych iteracjach. Wynikiem tych operacji będzie rozwiązanie w którym, w grafie oryginalnym każdy wierzchołek będzie odwiedzany tylko raz.

Ostatni z przykładów wykorzystania algorytmu mrówkowego jest harmonogram produkcji. W porównaniu do poprzednich metod w tym algorytmie zachodzi pewna modyfikacja. Głównym problemem w harmonogramie produkcji jest znalezienie takiej kolejności przetwarzanych zadań, aby jak najszybciej je przetworzyć. Aby lepiej zobrazować tą sytuację należy sobie wyobrazić fabrykę w której znajdują się linie produkcyjne. Na linii są przetwarzane zadania w odpowiedniej kolejności oraz każde z zadań może zostać wykonane w różnym czasie[4].

W tej metodzie, podobnie jak w metodzie do rozwiązymania problemu plecakowego, należy stworzyć graf pełny z wierzchołkami odpowiadającymi konkretnym zadaniom. Następnie mrówki przechodzą przez wszystkie wierzchołki i zostawiają feromony. Czynnikiem decydującym o wyborze wierzchołków nadal jest związana z feromonami. Do rozwiązyania tego problemu nie jest brana pod uwagę liczba feromonów na krawędzi pomiędzy wierzchołkiem a jego sąsiadami. Wykorzystywana jest natomiast suma feromonów na wszystkich krawędziach do odwiedzanych wierzchołków z wierzchołkami już odwiedzonymi.

1.2.3 Algorytmy zachłanne

Kolejnym spojrzeniem na poruszany w pracy problem komiwojażera są algorytmy zachłanne (ang. *greedy algorithms*). Nie łatwo znaleźć dowód na to czy dla podanego problemu algorytm zachłanny znalazł poprawny wynik, jednak stosując się do pewnych zasad można określić, że dla danego problemu istnieje rozwiązanie zachłanne. Główną strategią jaką się kierują jak sama nazwa naprowadza jest dokonywanie wyborów, które w danej chwili wydają się najlepsze. Oznacza to, że dokonuje się wyborów optymalnych lokalnie w nadziei, że te wybory doprowadzą do rozwiązania globalnego w zadowalającym czasie. W odróżnieniu do strategii zastosowanej w programowaniu dynamicznym wybory podejmowane przez algorytmy zachłanne nie są uzależnione od wyborów przeszłych. Kolejnym kryterium stosowanym do ocenienia poprawności rozwiązania zachłannego jest własność optymalnej

pod struktury, mówiąca o tym, że optymalne rozwiązanie dla całego problemu istnieje jedynie przy optymalnym rozwiązaniu pod problemów. Dane kryterium jest również istotne w przypadku rozwiązywania problemów metodą programowania dynamicznego. Algorytmy zachłanne nie zawsze prowadzą jednak do optymalnych rozwiązań, jednakże dla większości problemów dają wystarczające rezultaty. Skorzystanie z algorytmów zachłannych często okazuje się niewystarczające. Aby uzyskać lepszy efekt i polepszyć zbudowane już trasy możemy wykorzystać algorytmy lokalnej optymalizacji (ang. *local search*). Użycie ich na zwróconych przez algorytmy zachłanne trasach powinno zminimalizować odległości między wierzchołkami w celu poprawienia otrzymanego rozwiązania. Dokładniejszy opis działania wybranych algorytmów zachłannych i metod optymalizujących został przedstawiony w podrozdziałach.

Już w latach pięćdziesiątych została zastosowana koncepcja algorytmów zachłannych do przeszukiwania grafów, gdzie Esdger Wybe Dijkstra dążąc do skrócenia tras w Amsterdamie, stolicy Holandii opracował technikę do generowania minimalnych drzew rozpinających. Również w tej samej dekadzie Robert Clay Prim oraz Joseph Bernard Kruskal opracowali strategie optymalizacji, które opierały się na minimalizacji kosztów dla ważonych tras. Do dzisiaj stworzone algorytmy są wykorzystywane w mapach geograficznych do wyznaczania najkrótszych ścieżek, do znajdowania OSPF (ang. *Open Shortest Path First*) w protokole routingu IP (ang. *Internet Protocol*) czy w sieciach telefonicznych.

2. Algorytm genetyczny - Paweł

W tym rozdziale zostanie przedstawiony sposób, w jaki zostanie zastosowany algorytm genetyczny przy wyznaczaniu najlepszych tras dla ciężarówek przewozu odpadów komunalnych.

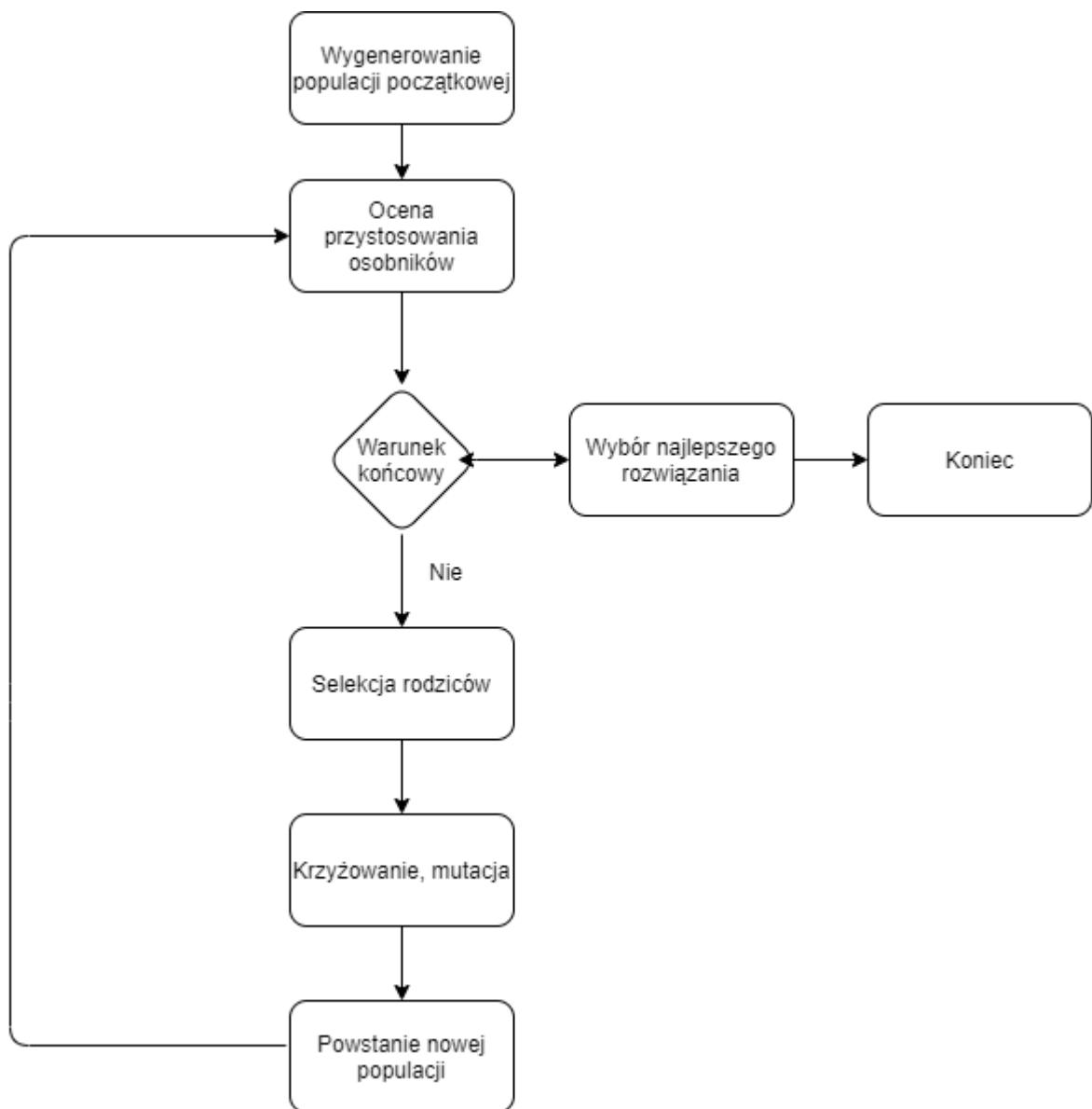
2.1 Wprowadzenie do algorytmu

Przed przejściem do omawiania algorytmu, należy wyjaśnić podstawowe pojęcia, które występują w algorytmie genetycznym:

- OSOBNIK – pojedyncze rozwiązanie problemu, zakodowane w postaci chromosomu,
- POPULACJA – zbiór osobników o stałej liczbie N w przekroju trwania całego algorytmu,
- GEN – przechowuje informację o dowolnej cenie osobnika. W zależności od sposobu kodowania może to być bit, dowolna cyfra lub znak,
- CHROMOSOM – składa się z uporządkowanego ciągu genów, przechowuje wszystkie cechy osobnika,
- GENOTYP – w przyrodzie może składać się z kilku chromosomów i określa skład osobnika. W algorytmach genetycznych przyjmuje się, że jest to pojedynczy chromosom,
- FUNKCJA PRZYSTOSOWANIA – funkcja za pomocą której ocenia się jakość osobnika[16].

Schemat blokowy klasycznego algorytmu genetycznego został pokazany na rysunku 2.1. Pierwszym krokiem jest wylosowanie populacji początkowej algorytmu. Wielkość populacji podczas trwania całego algorytmu jest stała N . Ważne jest, aby wszystkie osobniki były jak najbardziej zróżnicowane i wygenerowane losowo. Każdy z nich następnie musi zostać zakodowany do postaci chromosomów, które będą przechowywać w sobie informację o odwiedzanych punktach w postaci genów.

W populacji każdy osobnik musi zostać poddany ocenie funkcji przystosowania. Jej wynik determinuje jak dobre jest dane rozwiązanie. W klasycznym algorytmie dąży się do maksymalizacji tej funkcji. Określenie jak dana funkcja przystosowania będzie



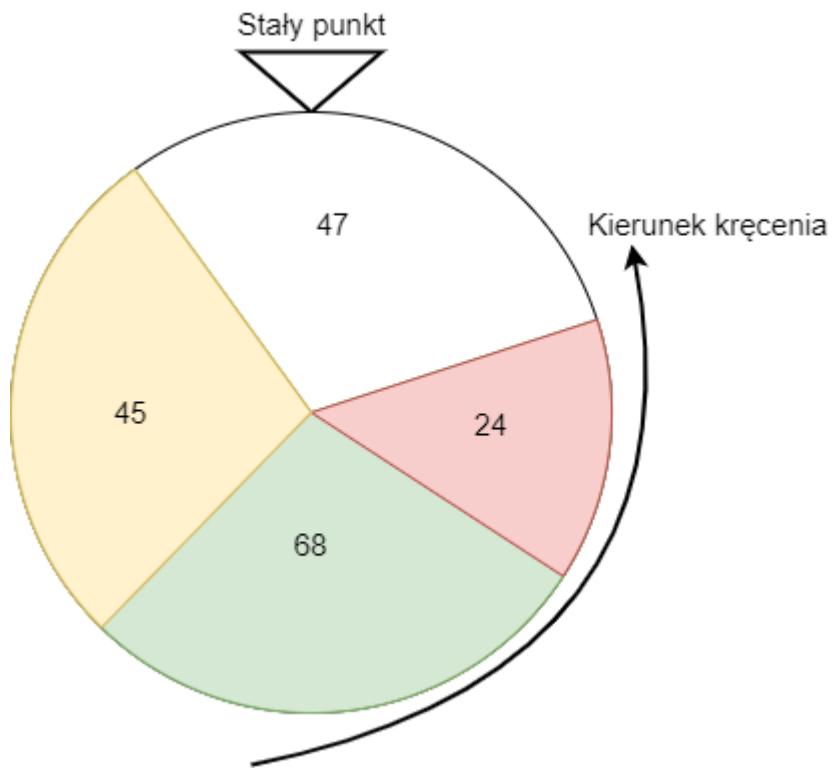
Rysunek 2.1: Schemat algorytmu genetycznego

wyglądać, jest to jedną z najważniejszych części algorytmu genetycznego. Jeśli zostanie źle zdefiniowana, znalezione osobnik może nie spełniać wymagań rozwiązania problemu.

Po ocenie osobników zostaje sprawdzony warunek końcowy algorytmu. W zależności od problemu jest zdefiniowany inny warunek. W klasycznych podejściach są trzy różne rodzaje warunków końcowych. Pierwszym popularnym warunkiem końcowym jest stała ilość iteracji algorytmu. Drugim warunkiem jest określenie z góry czasu trwania algorytmu, po upływie którego zostaje wybrany najlepszy osobnik. Ostatnim popularnym warunkiem jest wykonywanie algorytmu, do momentu, aż wyniki przestaną się poprawiać w kolejnych pokoleniach. Wybór w jaki sposób będzie wyglądać warunek końcowy zależy od wielu czynników. Jeśli ważny jest krótki czas, należy założyć pierwszy wariant. Jeśli natomiast

algorytm może szukać rozwiązania nawet kilka godzin, można przyjąć drugi wariant, łącząc go jednocześnie z trzecim.

Kolejnym krokiem algorytmu jest wyselekcjonowanie rodziców do reprodukcji. Polega ona na tym, że osobniki lepsze(mają większą wartość oceny przystosowania) mają większe szanse na pozostanie rodzicem i przekazanie swoich cech. [6]. Najpopularniejszymi metodami wyboru rodziców jest metoda ruletki oraz turniejowa.



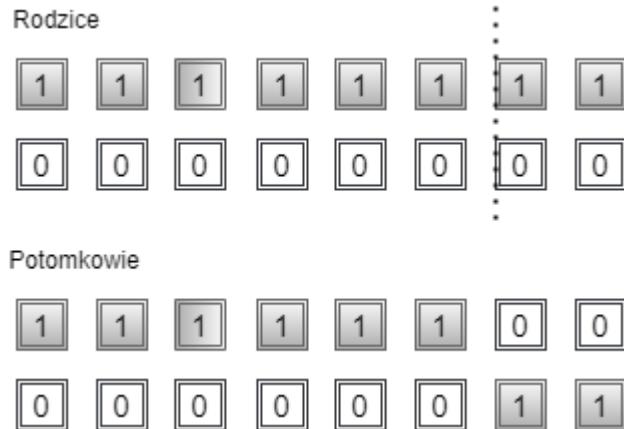
Rysunek 2.2: Metoda ruletki

Na rysunku 2.2 została zilustrowana metoda ruletki. Każdy z osobników dostaje wirtualny wycinek koła fortuny. Jego wielkość zależy od wartości funkcji prawdopodobieństwa. Przy każdym wyborze rodzica następuje zakręcenie koła i do reprodukcji zostaje wybrany osobnik na który będzie wskazywał stały punkt.

W metodzie turniejowej zostaje wybranych r osobników z populacji N . Z pośród nich zostaje wybrany zwycięzca(największa wartość funkcji przystosowania), który trafia do puli rodzicielskiej. Im większa jest ilość osobników r tym mniejsze szanse, że słabsze osobniki zostaną wybrane.

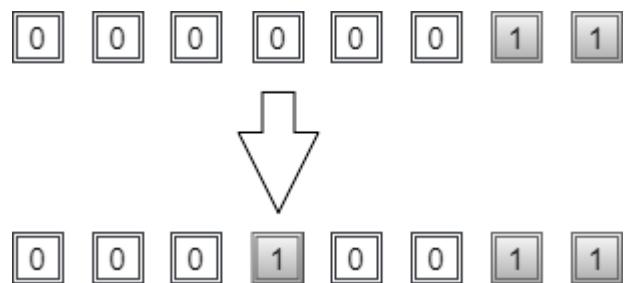
Wybrani rodzice zostają poddani operatorom genetycznym: krzyżowanie(ang. *crossover*) oraz mutacji(ang. *mutation*). Krzyżowanie polega na połączeniu części chromosomu jednego rodzica z częścią drugiego. Wynikiem takiego połączenia jest nowy osobnik.

Metody krzyżowania są różne, ale zawsze wykonywane muszą być w ten sam określony sposób. Wszystko zależy od metody zakodowania chromosomu oraz od tego czy kolejność genów i ich unikalność ma znaczenie. W klasycznym podejściu polega na rozcięciu w dowolnym miejscu genotypu u dwóch osobników oraz skrzyżowaniu ich ze sobą w tym punkcie (Rys. 2.3).



Rysunek 2.3: Klasyczne krzyżowanie

Następnie u nowego osobnika może z prawdopodobieństwem pm wystąpić mutacja. Jest to zmiana dowolnego pojedynczego lub ciągu genów na inny (Rys 2.4). Wartość pm w klasycznych algorytmach jest stosunkowo niska. Mutacja ma na celu delikatne zróżnicowanie osobników w celu przeszukania nowej przestrzeni rozwiązań. Natomiast gdyby zachodziła często, mogłaby powodować niszczenie dobrych rozwiązań. Po zasto-



Rysunek 2.4: Mutacja genotypu

sowaniu wszystkich operatorów genetycznych, nowa populacja jest poddawana ocenie przystosowania i jeśli wystąpił warunek końcowy, wybierane jest najlepsze rozwiązanie.

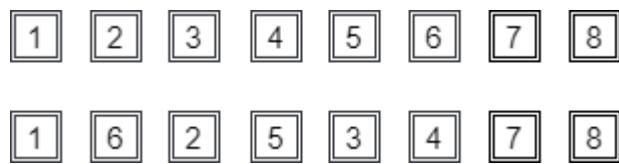
2.2 Parametry wejściowe

Przy problemie zoptymalizowania tras samochodów wywożących odpady komunalne zostaną zbadane takie parametry wejściowe jak: wielkość populacji, ilość iteracji (pokoleń),

czas trwania algorytmu, rodzaj krzyżowania, rodzaj mutacji oraz prawdopodobieństwo mutacji. Jeśli chodzi o metodę selekcji, to będzie to metoda turniejowa. Poza tymi zmiennymi ważne jest wyznaczenie funkcji przystosowania oraz zakodowania informacji o trasach do postaci chromosomu. W podrozdziałach 2.3 oraz 2.4 zostaną opisane trzy rodzaje mutacji oraz krzyżowań, które zostaną zbadane.

2.2.1 Chromosom

W algorytmie genetycznym, każdy osobnik z populacji reprezentuje jedno rozwiązanie. Jakość takiego rozwiązania jest zapisywana do zakodowanej postaci chromosomu. Chromosom z definicji jest to ciąg genów reprezentujący dane rozwiązanie. Z kolei gen przenosi informację o cechach. Możliwość osiągnięcia sukcesu w algorytmie genetycznym jest tylko wtedy, gdy odpowiednio zakoduje się cechy i ustali funkcję przystosowania. Do zakodowania badanego problemu zostanie użyta metody permutacyjna bez powtórzeń. Permutacja to jest dowolnie utworzony ciąg ze wszystkich elementów zbioru. Każdemu genowi przed zakodowaniem chromosomów zostanie przypisany unikalny indeks. Będzie on przechowywał informację, który punkt jest odwiedzany. Następnie dla każdego z N osobników zostanie zakodowany chromosom w postaci ciągu permutacyjnego. Kolejną warunkiem jaki musi spełnić chromosom to pierwszy i ostatni gen, nigdy nie może zmienić swojego miejsca. Punkt startowy zawsze musi zostać na miejscu pierwszym, jak również punkt rozładowkowy musi być na końcu. Na rysunku 2.5 zostały zilustrowane przykłady kodowania permutacyjnego bez powtórzeń, dla chromosomu o długości 8. W obu chromosomach pierwsze i ostatnie miejsca są takie same. W przyszłych podrozdziałach gen będzie oznaczał jeden z punktów załadowań (lub ostatni rozładowania) na trasie ciężarówki. Chromosom będzie oznaczał kolejność odwiedzania tych punktów przez pojazd.



Rysunek 2.5: Kodowanie chromosomów

2.2.2 Funkcja przystosowania

Algorytm genetyczny szuka osobnika z największą wartością funkcji przystosowania. W badanym problemie należy znaleźć najkrótszą trasę. W momencie, gdy długość takiej trasy zostanie odwrócona, to okaże się, że im większa wartość odwrotności tym krótsza trasa. Zatem wzór funkcji przystosowania to

$$fp = \frac{1}{s + 1}$$

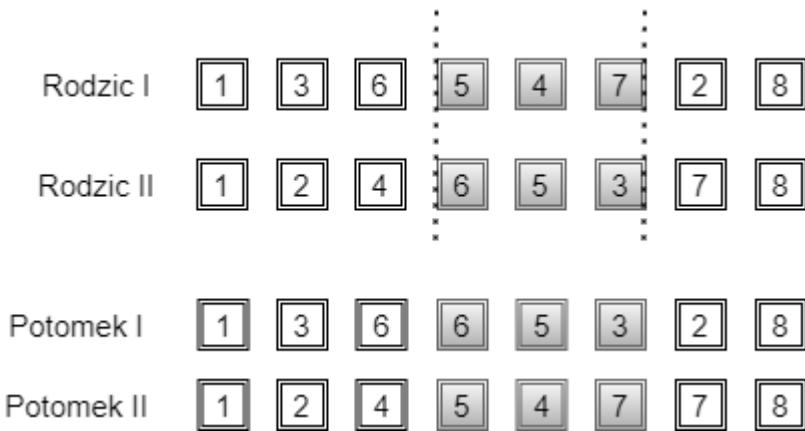
gdzie s - długość trasy, czyli suma odległości pomiędzy genami w chromosomie(1 - 2 - 3 - 4 - 5 - 6 - 7).

2.3 Metody krzyżowania

W pracy zostaną zbadane trzy rodzaje metod krzyżowania. Każde z nich charakteryzuje się czymś innym jeśli chodzi o liczbę potomków oraz porządek genów względem rodziców. Żadne z krzyżowań nie może zaburzyć dwóch warunków. Potomek musi posiadać strukturę permutacyjną oraz pierwszy i ostatni punkt nie mogą się przemieścić.

2.3.1 Krzyżowanie z częściowym odwzorowaniem - PMX

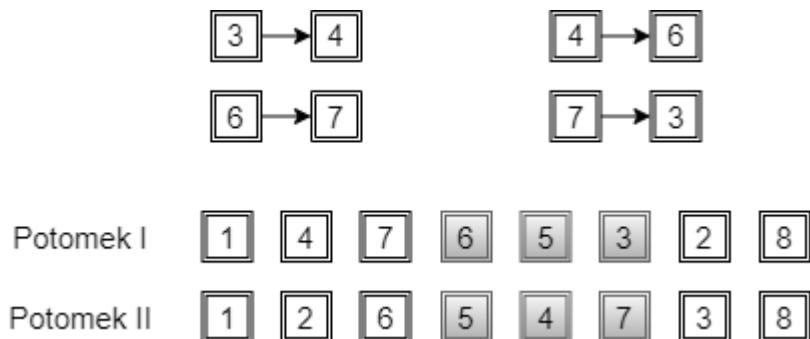
PMX[7] (ang. *Partially Mapped Crossover*) jest odmianą krzyżowania dwupunktowego, w którym powstaje dwójka potomstwa. Zakładając, że wyselekcyjnowano dwóch rodziców: 1 - 3 - 6 - 5 - 4 - 7 - 2 - 8 oraz 1 - 2 - 4 - 6 - 5 - 3 - 7 - 8 (Rys. 2.6). Losowane są



Rysunek 2.6: Krzyżowanie PMX - część 1

dwa dowolne punkty (nie uwzględnia się pierwszego i ostatniego), w których się je rozcina,

w opisywanym przypadku znajdują się one po trzecim i szóstym genie. Rozcięcia tworzą dwa segmenty 5 - 4 - 7 oraz 6 - 5 - 3, które zamieniają się ze sobą. W wyniku tej operacji powstały dwa chromosomy: 1 - 3 - 6 - 6 - 5 - 3 - 2 - 8 oraz 1 - 2 - 4 - 5 - 4 - 7 - 7 - 8. Oba z nich nie są jeszcze permutacjami. Należy teraz w powtórzonym genie, które znajdują się poza segmentami, zamienić na te geny, których brakuje. W tym celu następuje określenie mapowania pomiędzy genami, które następnie zamieniają się ze sobą w chromosomach (Rys. 2.7). W pierwszym dziecku należy zamienić odpowiednie geny: 3 -> 4 oraz 6 -> 7. W drugim

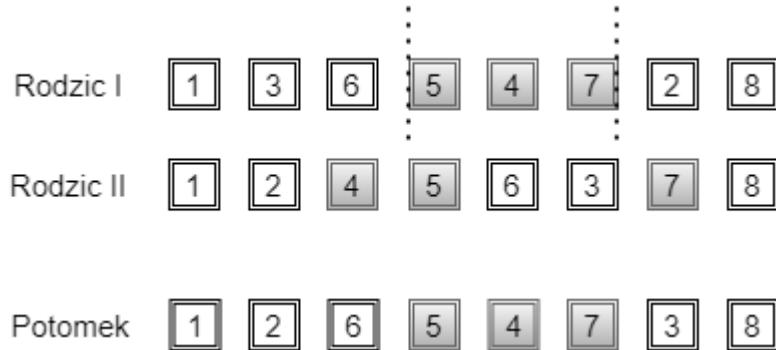


Rysunek 2.7: Krzyżowanie PMX - część 2

natomaiast zamienia się: 4 -> 6 oraz 7 -> 3. W wyniku tych zmian powstają dwa chromosomy: 1 - 4 - 7 - 6 - 5 - 3 - 2 - 8 oraz 1 - 2 - 6 - 5 - 4 - 7 - 3 - 8. Powstałe dzieci posiadają już strukturę permutacyjną oraz pierwszy i ostatni punkt się nie przemieściły.

2.3.2 Krzyżowanie z zachowaniem porządku - OX

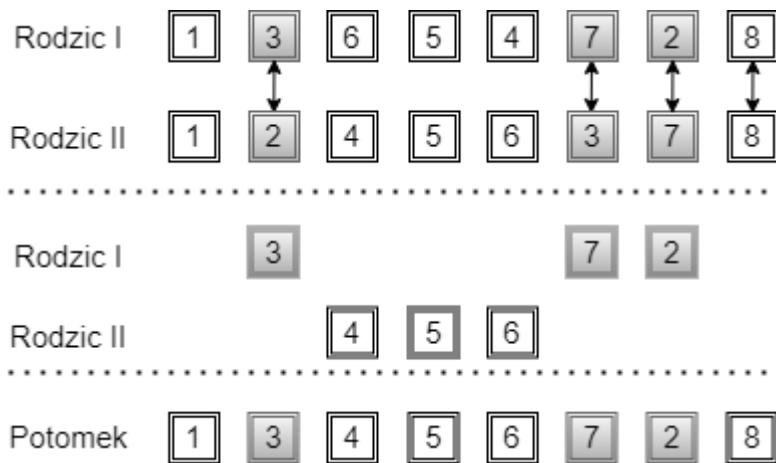
OX[11] (ang. *Order Crossover*) jest również odmianą krzyżowania dwupunktowego. W przeciwieństwie do PMX wynikiem będzie tylko jedno dziecko. Krzyżując ze sobą dwa chromosomy: 1 - 3 - 6 - 5 - 4 - 7 - 2 - 8 oraz 1 - 2 - 4 - 5 - 6 - 3 - 7 - 8. Pierwszym krokiem jest wylosowanie dwóch dowolnych punktów w których zostanie rozcięty pierwszy rodzic. Zakładając podobnie jak przy krzyżowaniu PMX, że są to punkty po trzecim i szóstym genie (Rys. 2.8). Oba punkty tworzą segment 5 - 4 - 7. Następnie w drugim rodzicu należy usunąć geny, które zostały z pierwszego rodzica wycięte. Ostatnim krokiem jest wstawienie wycinka 5 - 4 - 7 do drugiego rodzica, w to samo miejsce z jakiego zostały usunięte w pierwszym rodzicu. Powstał potomek 1 - 2 - 6 - 5 - 4 - 7 - 3 - 8, który spełnia założenia.



Rysunek 2.8: Krzyżowanie OX

2.3.3 Krzyżowanie cykliczne - CX

CX[10] (ang. *Cycle Crossover*) w przeciwieństwie do PX i OX nie polega na krzyżowaniu w dwóch określonych punktach. Aby wyznaczyć potomka, należy w dowolnym rodzicu znaleźć cykl permutacji, zaczynając od dowolnego miejsca pomiędzy pierwszym i ostatnim genem. Szukanie cyklu polega na kopiowaniu genów z jednego rodzica, według pozycji określonych przez rodzica drugiego. Na rysunku przedstawiono krzyżowanie CX dla dwóch chromosomów: 1 - 3 - 6 - 5 - 4 - 7 - 2 - 8 oraz 1 - 2 - 4 - 5 - 6 - 3 - 7 - 8. Losowany jest punkt startowy, inny niż 1 oraz 8. Na rysunku 2.9 został wylosowany



Rysunek 2.9: Krzyżowanie CX

drugi gen, czyli 3. Jego odpowiednikiem w drugim rodzicu jest 2. Należy znaleźć ten gen w pierwszym rodzicu. Znajduje się on na siódmym miejscu, jego odpowiednikiem jest gen 7. Ten gen jest na szóstym miejscu, a jego odpowiednikiem jest 3. W tym momencie został znaleziony cykl, ponieważ zaczynał się od 3 i nastąpiło zapętlenie. Znaleziony cykl to 3 - 2 - 7. Kolejnym krokiem jest wycięcie cyklu z chromosomu, w którym został wyznaczony. W ten sposób zostaje przepisane X - 3 - X - X - X - 7 - 2 - X. Aby skończyć krzyżowanie,

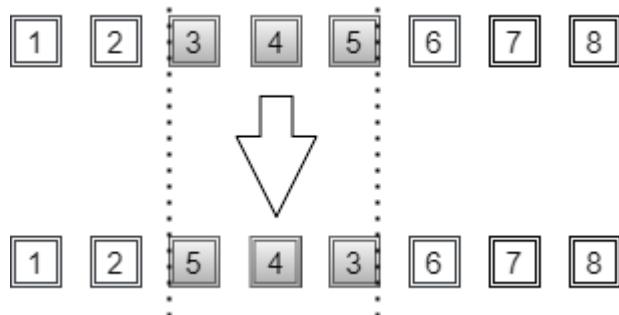
w X należy wpisać geny z drugiego rodzica, które nie wystąpiły w cyklu. W tak powstającym dziecku 1 - 3 - 4 - 5 - 6 - 7 - 2 - 8, wszystkie geny zajmują taką samą pozycję jak w którymś z rodziców, inaczej niż to było przy krzyżowaniu OX.

2.4 Metody mutacji

W pracy zostaną zbadane trzy różne rodzaje mutacji. Są to specjalne mutacje wykorzystywane przy strukturach permutacyjnych. Każda z nich zostanie również zbadana z różną wartością pm . Z reguły nie może być one duże, aby nie niszczyć potencjalnych rozwiązań. Powinno delikatnie wprowadzać różnorodność, aby była możliwość przeszukiwać nowe obszary. Poza tym bardzo ważne jest, aby zmiany powstałe w wyniku mutacji nie zaburzały struktury permutacyjnej chromosomu. Wszystkie mutacje zostaną opisane na tym samym chromosomie 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8.

2.4.1 Mutacja odwracająca

W mutacji odwracającej (ang. *revert mutation*) odwracającej wybierany jest dowolny podciąg genów z chromosomu, a następnie ich kolejność jest odwracana. Założmy, że wybrany podciąg to 3 - 4 - 5 (Rys. 2.10). W tym przypadku składa się on z trzech genów, więc po odwróceniu zamienią się ze sobą dwa geny, środkowy zostanie na tym samym miejscu. Po mutacji końcowy chromosom ma postać 1 - 2 - 5 - 4 - 3 - 6 - 7 - 8. Struktura permutacyjna nie została zachwiana.

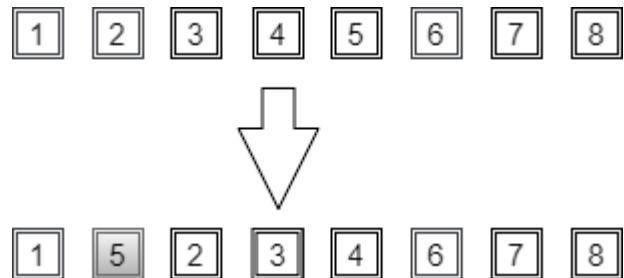


Rysunek 2.10: Mutacja odwracająca

2.4.2 Mutacja wstawiająca

W tej mutacji wstawiającej (ang. *insert mutation*) dowolny gen jest przestawia się w losowe miejsce. Jest to najprostsza mutacja, ale teoretycznie, może generować rozwiązaniami

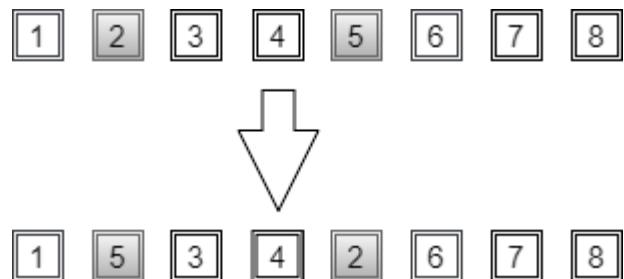
w całkowicie nowych przestrzeniach przeszukiwań. Punkt znajdujący się na końcu trasy, może znaleźć się na początku. Na rysunku 2.11 została wylosowany gen 5, który znajdował się na piątym miejscu, po mutacji znalazł się na drugim miejscu, w rezultacie chromosom po mutacji ma postać 1 - 5 - 2 - 3 - 4 - 6 - 7 - 8.



Rysunek 2.11: Mutacja wstawiająca

2.4.3 Mutacja zamieniająca

W mutacji zamieniającej (ang. *swap mutation*) zamienia się dwa dowolne geny miejscami. Jest to tak naprawdę, rozszerzona wersja mutacji wstawiającej. Losowane są dwa dowolne geny, na rysunku 2.12 są to 2 i 5, które ulegają zamianie miejscami. Powstały chromosom to 1 - 5 - 3 - 4 - 2 - 6 - 7 - 8.



Rysunek 2.12: Mutacja zamieniająca

3. Algorytm mrówkowy - Kamil

Słowo rozwój może być zestawiane z wieloma rzeczownikami. Wiele dziedzin ciągle się rozwija, powstają nowe udoskonalenia które wpływają na wiele dziedzin życia. Dzięki rozwojowi techniki ludzie są w stanie osiągać cele, które jeszcze nie dawno mogły być tylko marzeniami.

Wykorzystując pracę zespołową można w dokładniejszy oraz szybszy sposób udoskonalać rozwiązanie. Wysiłek wkładany przez każdą jednostkę jest bardzo istotny. Można naprawiać własne błędy oraz korygować innych uczestników zespołu. Kontrola oraz współpraca na każdym etapie jest kluczem do sukcesu.

Obserwując przyrodę możemy zauważać, w jaki sposób zwierzęta radzą sobie z różnymi problemami. Złożone grupy mogą być spokojniejsze o zdobycie pożywienia, czy też o przetrwanie w ciężkich warunkach. Praca zespołowa jest jedną z cech, której osobniki w grupie uczą się, nie będąc nawet tego do końca świadomym.

3.1 Wprowadzenie do algorytmu

Na przestrzeni czasu gatunki zwierząt żyjących na ziemi przystosowało się do panujących tu warunków. Jedną z takich gatunków są mrówki. Te niewielkich rozmiarów owady posiadają zdolności dzięki którym są w stanie dostosowywać się do otoczenia. Mrówki żyją w koloniach, w związku z czym wykorzystują pracę zespołową do rozwiązywania problemów, jakie codziennie napotykają na swojej drodze.

Aby zapewnić przetrwanie kolonii, mrówki potrzebują zapewnić sobie dostęp do pokarmu. Dziesiątki tysięcy mrówek mają swoje schronienie w mrowiskach. To tam trafia zdobyty przez nich pokarm. Wystarczy, aby jedna mrówka znalazła miejsce z pokarmem, to po powrocie do mrowiska inne osobniki są w stanie odtworzyć drogę do pożywienia. Na tym etapie należałoby się zastanowić, w jaki sposób mrówki są w stanie komunikować się między sobą?

Jednym z opisywanych rozwiązań do wyznaczania zoptymalizowanej trasy jest algorytm mrówkowy, inaczej nazywany ACO - (ang. *Ant Colony Optimization*). Jak sama nazwa wskazuje działanie algorytmu jest związane z mrówkami, a dokładnie z kolonią mrówek. Pomyśl na algorytm został zaproponowany na początku lat 90 XX wieku przez włoskiego badacza - Marco Dorigo [12].

Mrówki, dzięki pracy jaką wykonują, optymalizują przebytą drogę. Chodzi dokładnie o trasę jaką pokonują od swojego siedliska do miejsca, w którym znajduje się pożywienie. Trasę kształtuje cała kolonia mrówek, a nie pojedyncze przypadki. Mrówki przebywając każdą kolejną podróż wykształcają coraz to krótszą trasę.

3.2 Opis działania algorytmu

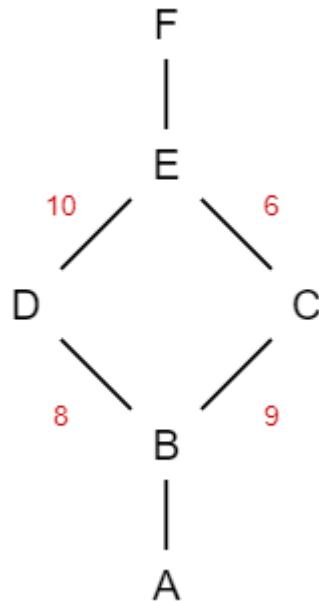
Mrówka w celu znalezienia pokarmu wyrusza z mrowiska bez obrania konkretnego kierunku. Losowo przesuwając się po terenie szuka pokarmu. Gdy już go znajdzie wraca do siedliska i informuje o tym fakcie pozostałe mrówki. Aby dostarczyć więcej pokarmu część kolonii mrówek wyrusza do miejsca spoczynku pożywienia. Chcąc uniknąć sytuacji w której zdobycz może zostać zabrana przez inne owady, mrówki muszą znaleźć jak najkrótszą trasę jaką mają do pokonania.

Mimo posiadania informacje o znalezionym pokarmie, każda mrówka może wyznaczyć na nowo lokalizację pożywienia. Czy mrówki poruszają się tą samą trasą przy każdym wyjściu z mrowiska? Aby trafić do miejsca, w którym znajduje się pokarm, wspomniane owady wykorzystują ślady pozostawione przez osobników, które już natrafiły na pożywienie. W ten sposób mrówka, natrafia na ślad poprzednika, który jest wskazówką do znalezienia poszukiwanego pokarmu. Wspomniany ślad nazywa się feromonem. To dzięki tej właściwości mrówki są w stanie lokalizować trasy prowadzące do pokarmu.

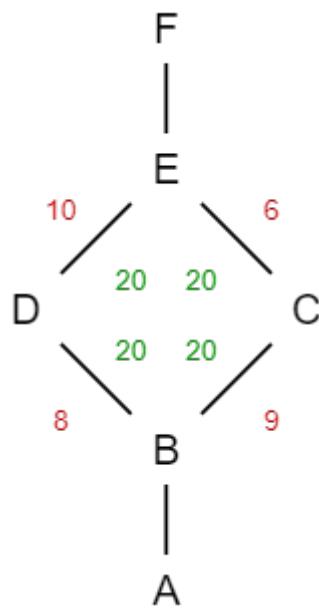
Na rysunku 3.1 został przedstawiony graf z wierzchołkami A-B-C-D-E-F. Nad krawędziami kolorem czerwonym zostały oznaczone wagи. Na początku założymy, że mamy do dyspozycji 80 agentów. Przez K pierwszych iteracji mrówki poruszały się losowo i powstał następujący podział:

Tak jak można to zauważyć na rysunku 3.2, po pierwszych iteracjach, przez obie ścieżki przechodzi taka sama liczba agentów. Dzieje się tak, ponieważ mrówki rozpoczynają pracę w sposób losowy. W dalszych krokach następują modyfikacje i agenci dążą do wyznaczenia optymalnej ścieżki.

Liczba agentów odwiedzających ścieżki zmienia się. Lepsza trasa zyskuje widoczną przewagę. W kolejnych iteracjach mrówki wykorzystują siłę feromonów. Krótsze ścieżki są częściej odwiedzane w związku z czym zapach na tych krawędziach jest silniejszy oraz podtrzymywany. Na dłuższych trasach zapach zanika i przestają one być atrakcyjne dla



Rysunek 3.1: Położenie ścieżek na przykładowym grafie

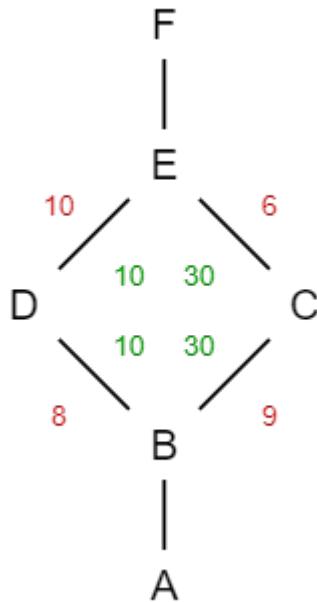


Rysunek 3.2: Rozkład agentów na grafie po N początkowych iteracjach

agentów. Opisana sytuacja prowadzi do wyznaczenia trasy która jest najatrakcyjniejsza do przebycia dla mrówek.

3.3 Feromony

Feromony posiadają cechę, która może się wydawać, że negatywnie wpływa na wyznaczanie ścieżki. Chodzi tutaj o ułatwianie się zostawionego zapachu. Na początku



Rysunek 3.3: Rozkład agentów w grafie po optymalizacji

zjawisko to może wydawać się niepożądany, ale w rzeczywistości ma duży wpływ na wyznaczenie krótszej trasy. Jeśli feromony nie straciłyby na swojej sile, to bardzo prawdopodobne, że pierwotna ścieżka mogłaby zostać uznana za bardziej atrakcyjną.

W jaki sposób wyznaczona zostaje najkrótsza ścieżka? Zapach feromonów jest podtrzymywany przez wędrujące mrówki. Z czasem owady te same zbaczają z drogi w celu poszukiwania alternatywnej trasy. Jeśli wybrana trasa jest krótsza od pozostałych to ślad jest podtrzymywany, a na innych zanika. Dzięki temu w sposób iteracyjny można wyróżnić trasę najkrótszą, a słabsze z czasem zostają odrzucone, ponieważ przestają być odwiedzane.

Feromony są istotnym czynnikiem w całym procesie. To dzięki nim trasa jest ulepszana. Zapach posiada jedną z charakterystyk, która na początku może wydawać się problematyczna. Wraz z upływem czasu siła zapachu słabnie aż do całkowitego zaniknięcia. Właściwość ta jest zaletą, a nie wadą. To dzięki pracy zespołowej zapach na najlepszej trasie jest podtrzymywany, a na słabszych zanika. Dzięki tej selekcji dłuższe trasy nie są brane pod uwagę w wyniku czego zostaje trasa najkorzystniejsza.

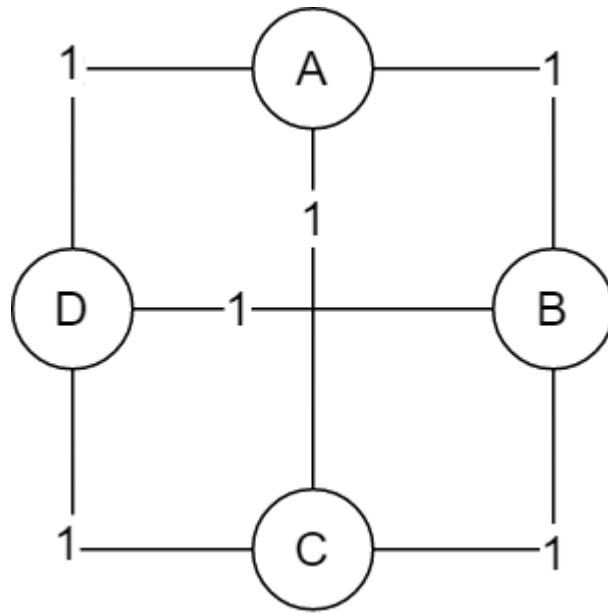
Do wyznaczenia optymalnej trasy potrzebne są długości jakie należy przebyć do przemieszczania się między punktami. W tabeli 3.1 przedstawione są przykładowe odległości.

Równie ważne jest wyznaczenie początkowych współczynników feromonów. Na początku nadajmy wszystkim krawędziom w grafie wartości równe 1, a współczynnik

Tabela 3.1: Wartości kosztów

	A	B	C	D
A	0	3	8	2
B	3	0	2	4
C	8	2	0	1
D	2	4	1	0

parowania niech wynosi 0.5. Jak zostały rozmieszczone wartości feromonów na grafie można zobaczyć na rysunku 3.4



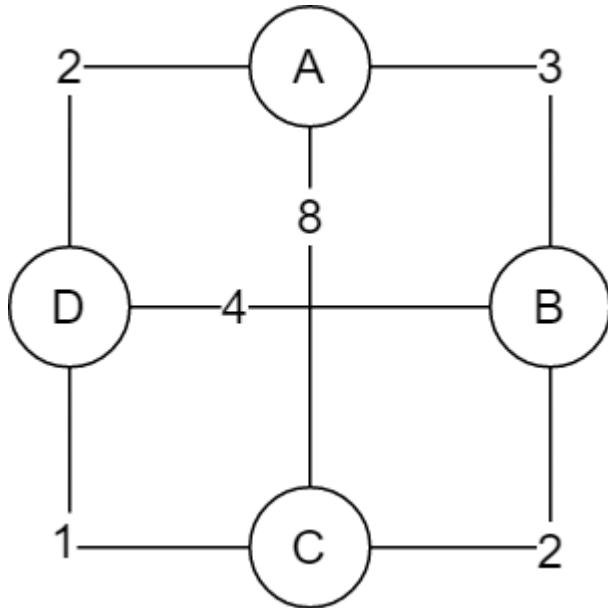
Rysunek 3.4: Graf z początkowymi wartościami feromonów

Na rysunku 3.5 został przedstawiony graf z wartościami feromonów jakie są początkowo umieszczone na krawędziach. Początkowe dane wyznaczmy w sposób losowy trasy dla dwóch agentów: L1 i L2. Na rysunku 3.6 została przedstawiona trasa agenta L1. Odwiedzone zostały wierzchołki w następującej kolejności: A, B, C, D, A.

Agent L2 wyznaczył następującą trasę: A, C, B, D, A. Trasa ta została przedstawiona na rysunku 3.7.

Mrówki odpowiednio pokonały dystans 8 i 16 punktów. Dzięki tej informacji można zaktualizować wartości feromonów na poszczególnych krawędziach. Do obliczeń wykorzystany jest wzór:

$$\tau_{i,j} = ((1-p)\tau_{i,j} + \sum_{k=1}^m \Delta\tau_{i,j}^k) \quad (3.1)$$



Rysunek 3.5: Graf z wagami

gdzie:

- p jest współczynnikiem parowania feromonów;
- i jest wierzchołkiem początkowym;
- j jest wierzchołkiem docelowym;
- k jest wagą między punktami;

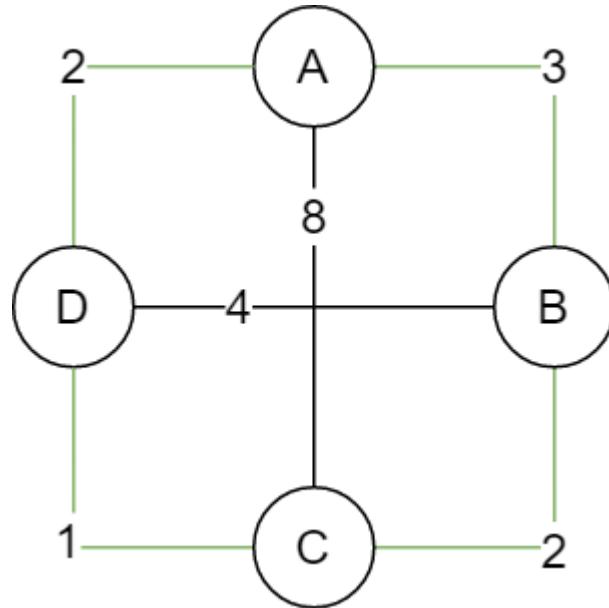
Krawędzie A-B i C-D zostały odwiedzona jedynie przez agenta L1 w wyniku czego wartość feromonu zostaje zmieniona na 10/16. Następnie krawędzie A-C i B-D zostają zaktualizowane na 9/16. Krawędzie A-D i B-C są odwiedzone dwukrotnie a wartość feromonów wynosi 11/16. Aktualne rozmieszczenie wartości feromonów zostało przedstawione na rysunku 3.8.

Ostatnią fazą algorytmu jest wyznaczenie prawdopodobieństwa z jakim kolejni agenci będą wybierać kolejny wierzchołek. W kolejnej iteracji agent L3 znajduje się w wierzchołku B. Do wyboru ma krawędzie A, C i D. Dla wszystkich możliwości wyliczane jest prawdopodobieństwo według wzoru

$$p_{i,j} = \frac{(\tau_{i,j})(\eta_{i,j})}{\sum(\tau_{i,j})(\eta_{i,j})} \quad (3.2)$$

gdzie:

- i jest wierzchołkiem początkowym;

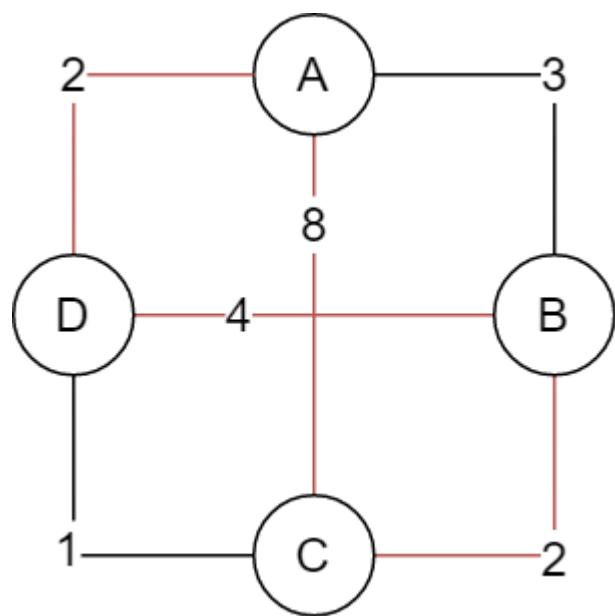


Rysunek 3.6: Trasa przebyta przez agenta L1

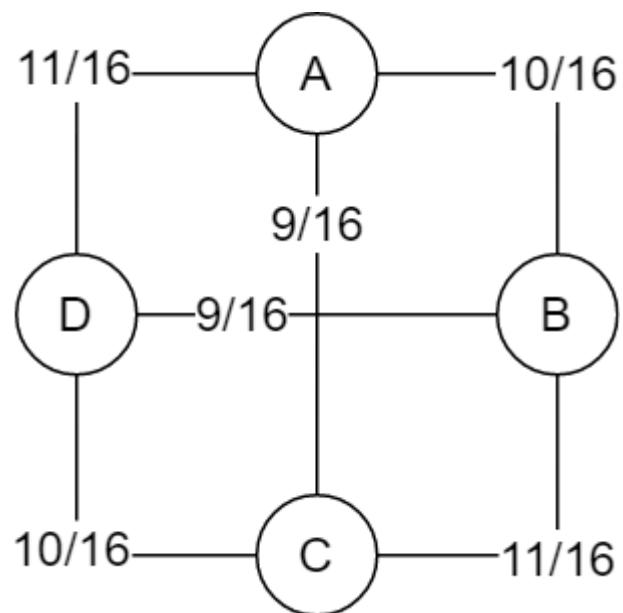
– j jest wierzchołkiem docelowym;

Prawdopodobieństwo przejścia z krawędzi B do krawędzi A wynosi około 20%, do krawędzi D 30%, a do krawędzi C około 50%.

Optymalna trasa zostanie wyznaczona po wykonaniu wielu iteracji. Liczba iteracji nie jest zdefiniowana i dla każdego przypadku może być różna. Sytuacja wygląda identycznie w przypadku wyboru wartości p . Ważne jest natomiast to, aby w trakcie działania algorytmu nie modyfikować tej wartości. Powinno ona być taka sama na każdym kroku algorytmu.



Rysunek 3.7: Trasa przebyta przez agenta L2



Rysunek 3.8: Graf z wartościami feromonów po modyfikacji

4. Algorytmy zachłanne - Przemysław

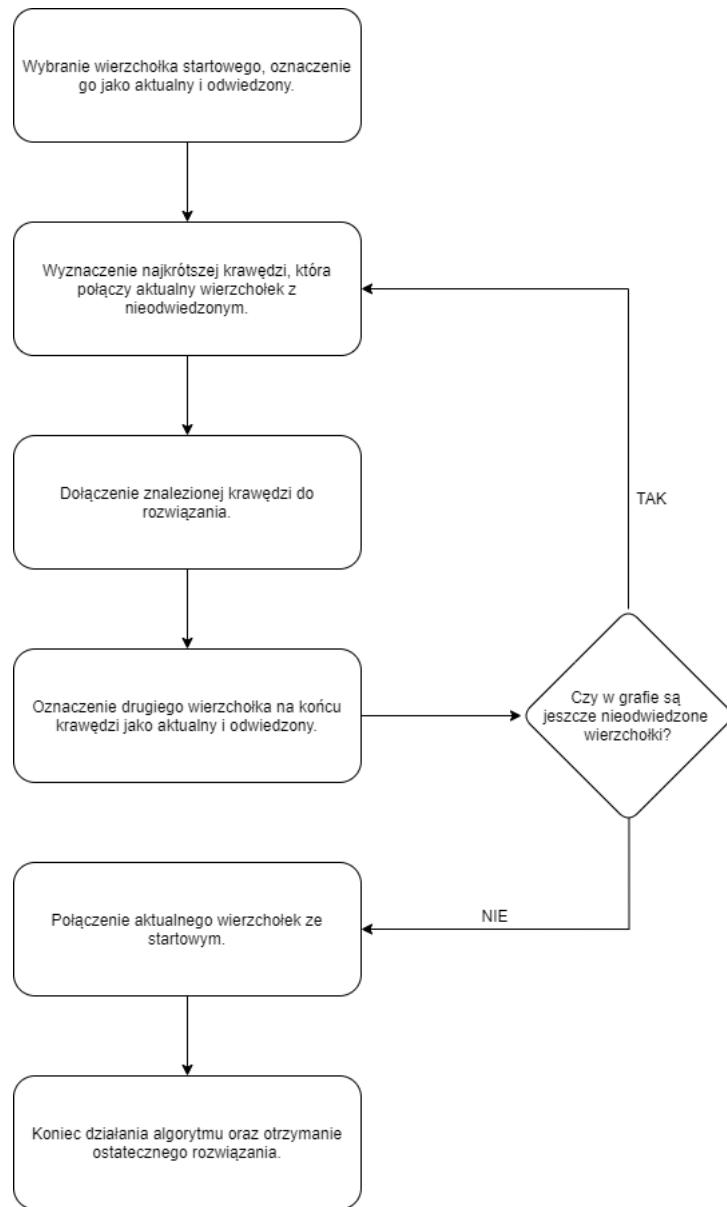
Istnieje wiele algorytmów zachłannych pozwalających otrzymać optymalne rozwiązanie dla problemu znalezienia najkrótszej trasy. W poniższym rozdziale opisane dokładniej zostało działanie algorytmu najbliższego sąsiada, algorytmu najmniejszej krawędzi oraz algorytmu A*. W celu zoptymalizowania otrzymanych rozwiązań zastosowany zostanie operator 2-opt oraz współczynnik błędu przy dokonywaniu wyborów.

4.1 Algorytm najbliższego sąsiada

Poniższy podrozdział przybliży działanie algorytmu najbliższego sąsiada. Wszystkie wymagane dla algorytmu operacje zostaną opisane krok po kroku oraz przedstawione na krótkim przykładzie. Jak sama nazwa wskazuje jest to algorytm polegający na odwiedzaniu, zaczynając od wybranego wierzchołka początkowego następnego wierzchołka jeszcze nieodwiedzonego znajdującego się najbliżej poprzednio odwiedzonego.

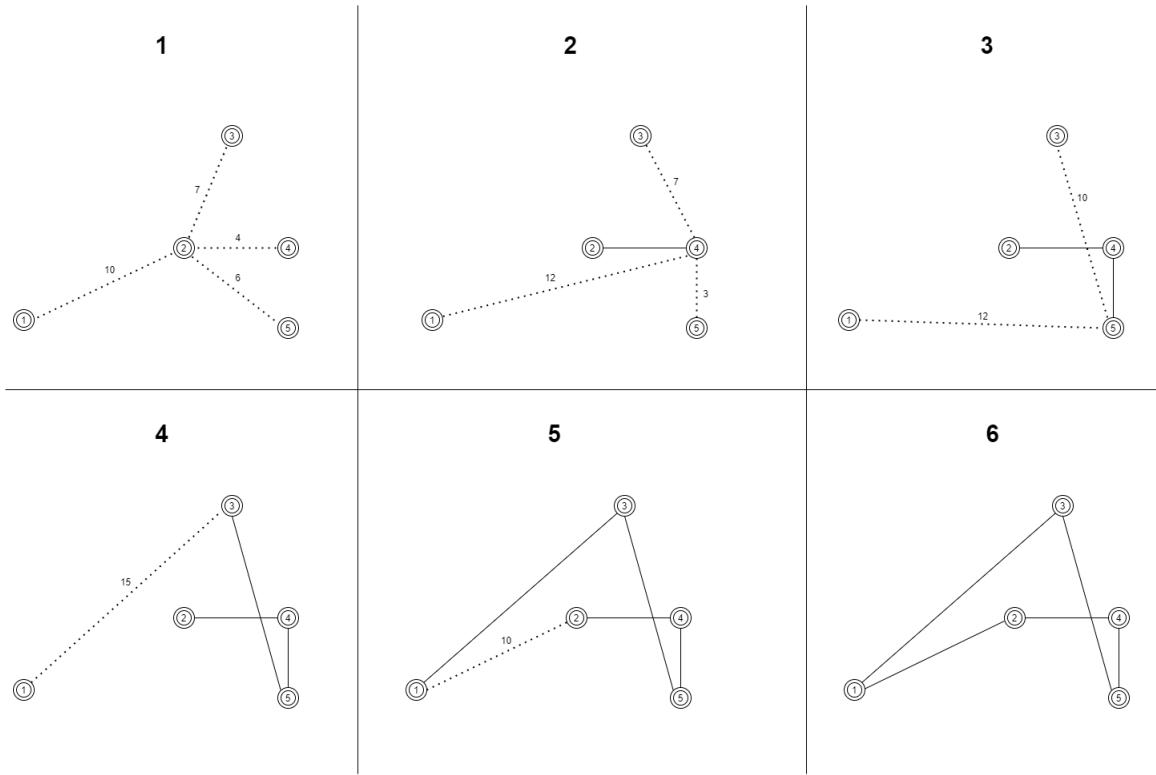
Na rysunku 4.1 został pokazany schemat blokowy algorytmu najbliższego sąsiada. Pierwszym krokiem jest wyznaczenie wyznaczenie wierzchołka startowego. Następnie dla aktualnego wierzchołka należy obliczyć najkrótszą krawędź łączącą aktualny wierzchołek spośród kolekcji wierzchołków nieodwiedzonych. Wybór najlepszej opcji połączenia dwóch wierzchołków należy dodać do rozwiązania, a drugi wierzchołek staje się aktualnym od którego będziemy wyznaczać kolejne odległości do nieodwiedzonych jeszcze wierzchołków. Czynności należy powtarzać do momentu odwiedzenia wszystkich wierzchołków w podanym grafie. Na samym końcu wystarczy jedynie połączyć ostatni wierzchołek z początkowym. Po wykonaniu wszystkich kroków algorytm najbliższego sąsiada powinien zwrócić optymalne dla niego rozwiązanie.

Aby lepiej zobrazować otrzymanie rozwiązania przez algorytm przedstawiono poniżej na rysunku 4.2 jego działanie na przykładzie. W podanym przypadku należy założyć że wierzchołek nr 2 jest wierzchołkiem startowym, więc należy ustawić go jako aktualny w danym momencie. Po obliczeniu wszystkich odległości prowadzonych do wierzchołków nieodwiedzonych wychodzi, że najkrótsza krawędź prowadzi do wierzchołka nr 4, więc należy dołączyć wybraną krawędź do rozwiązania i oznaczyć nowy aktualny wierzchołek, który staje się również odwiedzonym. W podanym grafie znajdują się nadal nieodwiedzone wierzchołki, dlatego algorytm powtarza krok nr 2 w celu znalezienia kolejnej najkrótszej



Rysunek 4.1: Schemat algorytmu najbliższego sąsiada

krawędzi. Kolejną najlepszą odnalezioną w danym momencie krawędzią, którą algorytm doda do rozwiązania będzie krawędź o wartości 3 łącząca aktualny wierzchołek z wierzchołkiem nr 5. Kroki 4, 5 oraz 6 przedstawiają kolejne powtarzalne iteracje algorytmu dochodząc w ostatnim kroku do utworzenia cyklu i tym zakończeniu działania algorytmu. W ten sposób otrzymano zachłanne rozwiązanie 2 - 4 - 5 - 3 - 1 - 2. Warto zaznaczyć, że dzięki oznaczaniu wierzchołków jako odwiedzone nie trzeba w żadnej iteracji martwić się o to czy dołączenie kolejnej krawędzi z nieodwiedzonym wierzchołkiem spowoduje utworzenie niepożądanego cyklu.

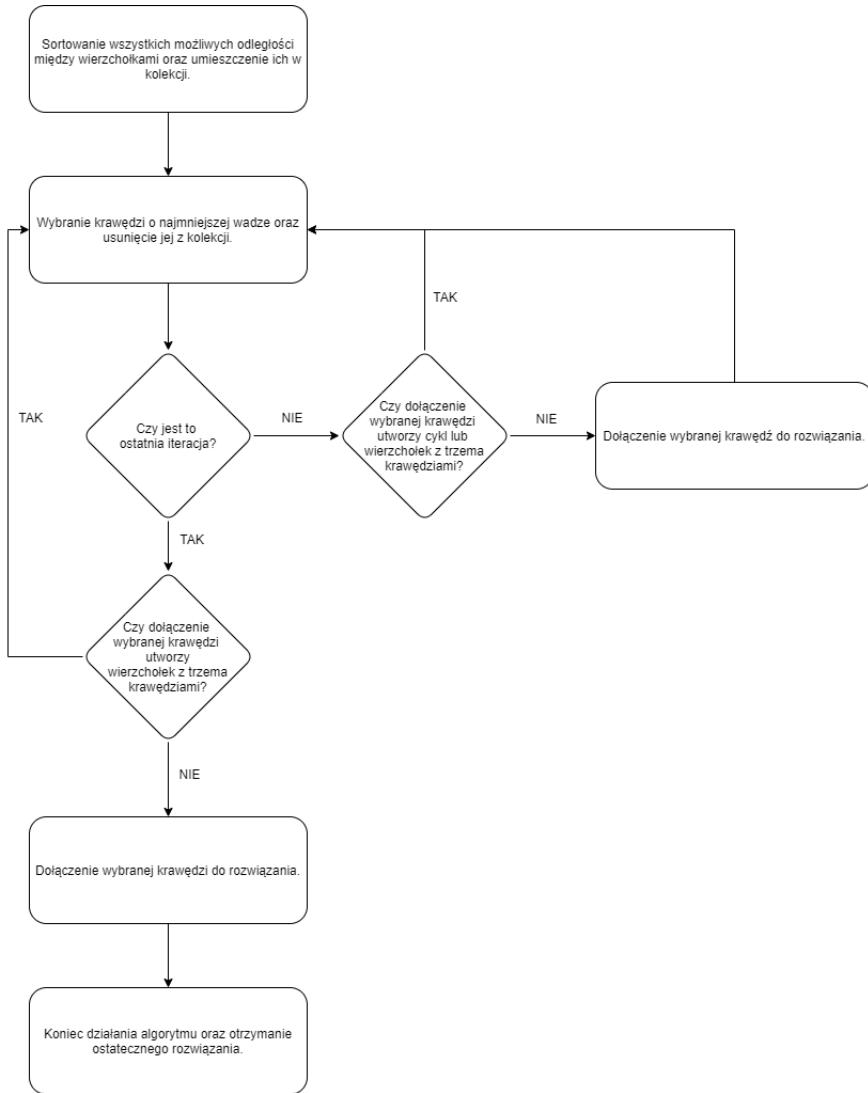


Rysunek 4.2: Algorytm najbliższego sąsiada

4.2 Algorytm najmniejszej krawędzi

Kolejny podrozdział algorytmów zachłannych został poświęcony dokładniejszemu opisie działania algorytmu najmniejszej krawędzi, który w swoim działaniu przypomina algorytm poszukujący minimalnego drzewa rozpinającego, poprzez dołączenie do aktualnego rozwiązania najkrótszych wśród dopuszczalnych krawędzi. Aby otrzymać zachłanne rozwiązanie przy wykorzystaniu algorytmu najmniejszej krawędzi należy wykonać następujące kroki przedstawione schemacie blokowym na rysunku 4.3.

Algorytm rozpoczyna swoje działanie od posortowania wag krawędzi między wszystkimi wierzchołkami oraz umieszcza podane odległości w kolekcji. Następnie wybierana jest zawsze krawędź o najmniejsze wartości oraz od razu usuwana jest z podanej kolekcji. Aby wybrane połączenie między wierzchołkami zostało dodane do rozwiązania musi spełniać warunek nie utworzenia cyklu oraz wierzchołka o trzech krawędziach, w przeciwnym wypadku połączenie jest pomijane oraz algorytm wybiera kolejną krawędź. Iteracje są powtarzane do momentu, aż liczba dodanych połączeń jest równa liczbie wszystkich wierzchołków. W przypadku ostatniej iteracji wyznaczona krawędź nie musi już spełniać

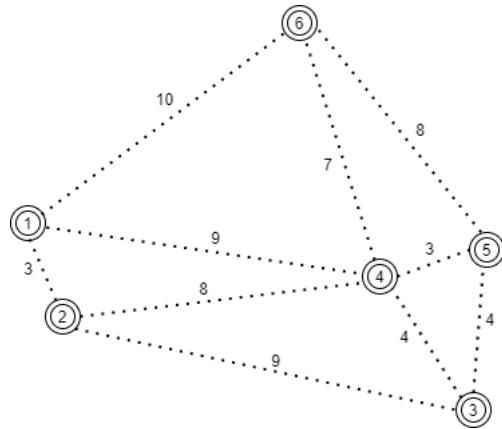


Rysunek 4.3: Schemat algorytmu najmniejszej krawędzi

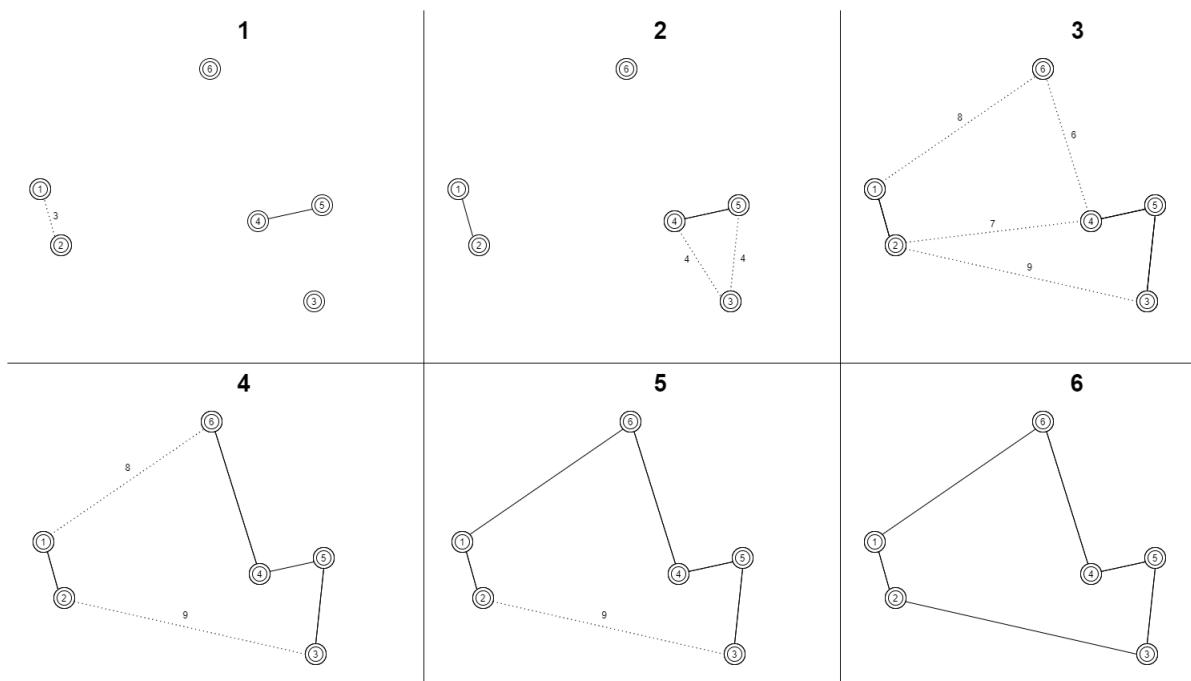
warunku z utworzeniem cyklu. Po zakończeniu działania algorytmu otrzymano optymalne rozwiązanie dla algorytmu najmniejszej krawędzi.

Działanie algorytmu przedstawione zostało na rysunku 4.4 przedstawiającym przykład ze sześcioma losowo rozmieszczonymi wierzchołkami. Natomiast wizualizacja kolejnych kroków algorytmu została przedstawiona na rysunku 4.5. Po posortowaniu wszystkich dostępnych krawędzi dla każdego wierzchołka można rozpoczęć działanie algorytmu.

Podczas pierwszej iteracji okazuje się, że są aktualnie dwie krawędzie z najmniejszą wagą o wartości 3 (1 - 2 oraz 4 - 5), nie ma więc znaczenia, która krawędź algorytm doda w pierwszej kolejności, ponieważ żadna z wybranych nie utworzy w tym momencie cyku. W przypadku drugiej iteracji algorytm wybiera krawędź o najmniejszej wagie, dołączając ją do aktualnego rozwiązania. Analogiczna sytuacja do pierwszej iteracji występuje w



Rysunek 4.4: Początkowe rozmieszczenie wierzchołków



Rysunek 4.5: Algorytm najmniejszej krawędzi

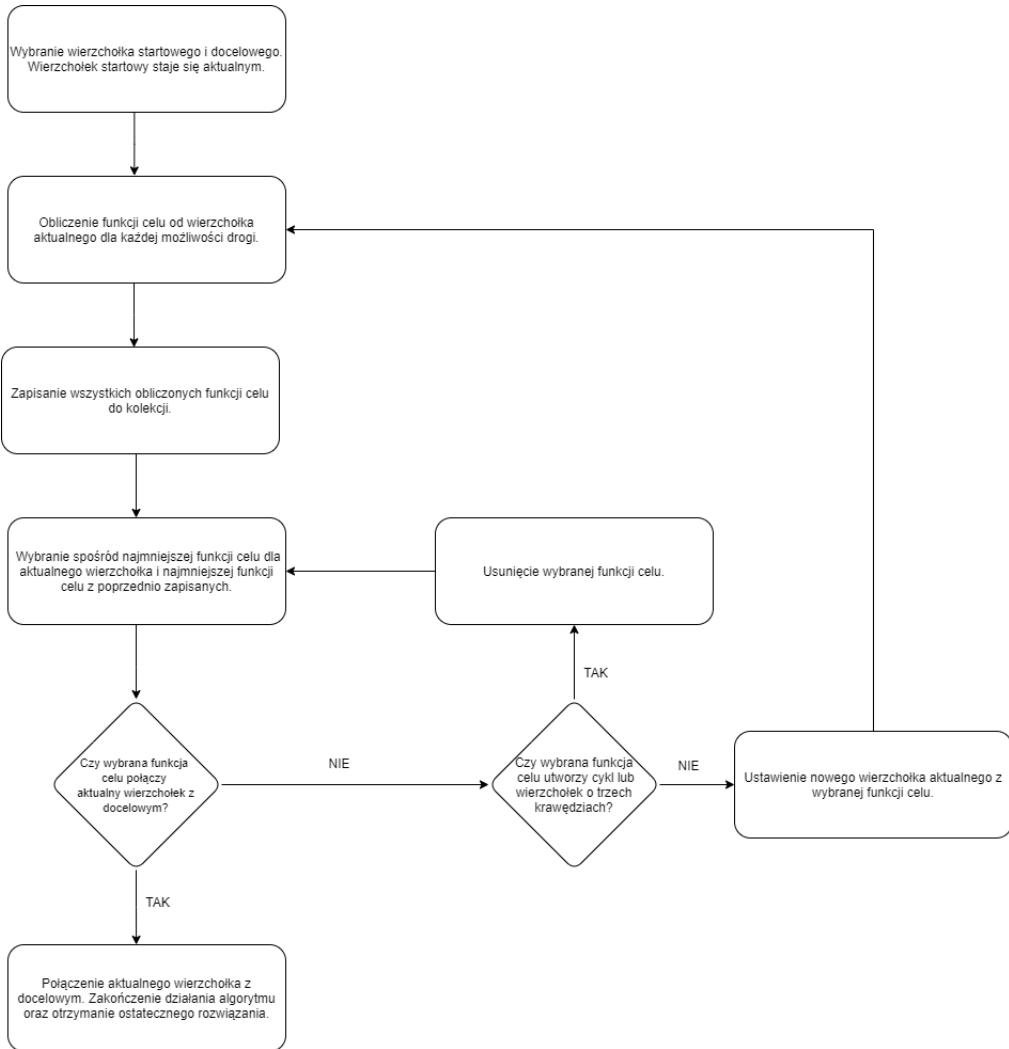
trzeciej iteracji, gdzie nie ma różnicy, która krawędź zostanie dołączona do rozwiązania. Czwarta iteracja pokazuje sytuację, w której nie można połączyć krawędzi 3 - 4, ponieważ utworzyłoby to niedozwolony w trakcie algorytmu cykl, dlatego tym razem wybierane jest inne połączenie o najmniejszej wadze (4 - 6). Kolejny krok przedstawia przypadek, gdzie nie jest możliwe połączenie krawędzi 2 - 4 (najmniejsza wartość - 7), ponieważ spowodowałoby dla wierzchołka nr 4 utworzenie trzech wychodzących z niego krawędzi, więc do rozwiązania dochodzi kolejna najmniejsza krawędź 1 - 6. W ostatnim kroku, pomimo dostępnych krawędzi z mniejszą wagą wybierana została krawędź 2 - 3, ponieważ tylko ona nie spowoduje utworzenia wierzchołka o trzech krawędziach. W takim wypadku kończąc działanie algorytmu otrzymano rozwiązanie 1 - 6 - 4 - 5 - 3 - 2 - 1.

4.3 Algorytm A*

Ostatnim omówionym rozwiązaniem do znajdowania najkrótszej ścieżki w grafie jest algorytm A*, w którym zawsze zostanie znalezione najkorzystniejsze zachłanne rozwiązanie. Strategia gwarantuje, że każdy wierzchołek zostanie odwiedzony, przy czym dokonuje w danym momencie najlepszych wyborów. Główną zasadą algorytmu A* jest minimalizacja funkcji kosztu $g(x)$ oraz funkcji heurystycznej $h(x)$ zdefiniowanej jako funkcja celu $f(x) = g(x) + h(x)$. Funkcja heurystyczna musi spełniać dwa wymagane warunki tj. warunek dopuszczalności i warunek monotoniczności. Warunek dopuszczalności polega na tym, aby funkcja heurystyczna za bardzo minimalizowała koszt, a przesadzała przy maksymalizacji zysku. Natomiast warunek monotoniczności mówi, że oszacowywanie wyniku musi być coraz mniej optymistyczne w momencie zbliżania się do rozwiązania. Jeśli przestrzeń przeszukiwań zawierać będzie ścieżki to można wówczas sprowadzić problem do problemu poszukiwania najkrótszej ścieżki w grafie. W danym przypadku funkcją heurystyczną będzie wówczas odległość w linii prostej między wierzchołkami. Zapewnia to w ten sposób, że taka funkcja jest nadmiernie optymistyczna.

Schemat blokowy działania algorytmu A* został przedstawiony na rysunku 4.6. Algorytm rozpoczyna swoje działanie od wyboru wierzchołka startowego oraz końcowego, gdzie wierzchołek startowy staje się aktualnym. Następnie dla aktualnego wierzchołka obliczana i zapisywana jest funkcja celu $f(x)$ dla każdego poszczególnego możliwego połączenia. Kolejnym krokiem jest wybranie spośród minimalnej funkcji celu dla aktualnego wierzchołka oraz minimalnej funkcji celu z poprzednio zapisanych (jeśli takowe istnieją). Jeśli wybrana funkcja celu nie łączy aktualnego wierzchołka z docelowym oraz nie utworzy cyklu lub wierzchołka o trzech krawędziach jest brana dalej pod uwagę i ustalany jest nowy aktualny wierzchołek. W przeciwnym wypadku wybrana funkcja celu nie jest brana pod uwagę i algorytm jeszcze raz porównuje obliczone poprzednio funkcje celu. Kroki algorytmu są powtarzane do momentu aż wybrana funkcja celu połączy aktualny wierzchołek z końcowym.

Lepsza wizualizacja działania algorytmu A* została przedstawiona na grafie umieszczonym na rysunku 4.7. Wierzchołkiem startowym będzie w tym wypadku wierzchołek numer 1, natomiast wierzchołkiem docelowym będzie punkt numer 7. Jak można zauważyć na rysunku 4.8 podczas pierwszej iteracji algorytm A* zdefiniował dwie funkcje celu, jednakże $f_1(2) = 4 + 5$ jest w danym momencie lepszym wyborem. W kolejnej iteracji

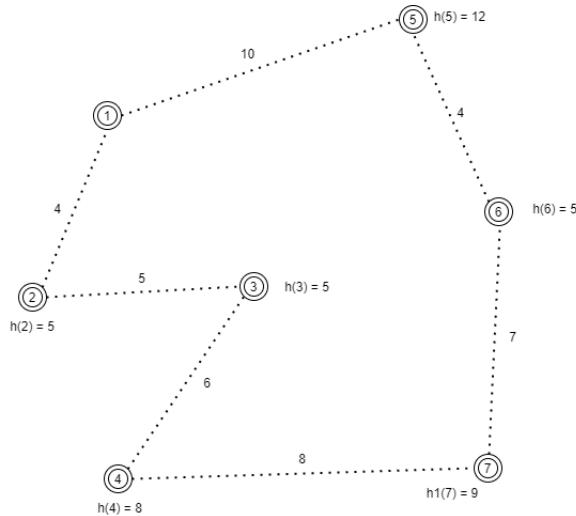


Rysunek 4.6: Schemat blokowy algorytmu A*

nadal rozwinięcie pierwszej funkcji celu daje lepszy rezultat, dlatego też nowym aktualnym wierzchołkiem staje się wierzchołek numer 3. Trzecia iteracja przedstawia sytuację, w której druga funkcja celu łącząca aktualny wierzchołek z wierzchołkiem numer 5 daje lepszy w danym momencie rezultat, dlatego wierzchołek numer 5 jest brany pod uwagę. Po zastosowaniu się do założeń algorytmu oraz wykonaniu wszystkich kroków otrzymujemy optymalne rozwiązanie łączące wierzchołek numer 1 z wierzchołkiem numer 7(1 - 5 - 6 - 7) o wartości 21.

4.4 Optymalizacja otrzymanych rozwiązań

W ostatnim podrozdziale skupimy się na sposobach optymalizacji otrzymanych przez algorytmy zachłanne rozwiązań. Dokładniej omówiona zostanie metoda 2-opt oraz zastosowanie współczynnika błędu zaproponowanego przez studenta.



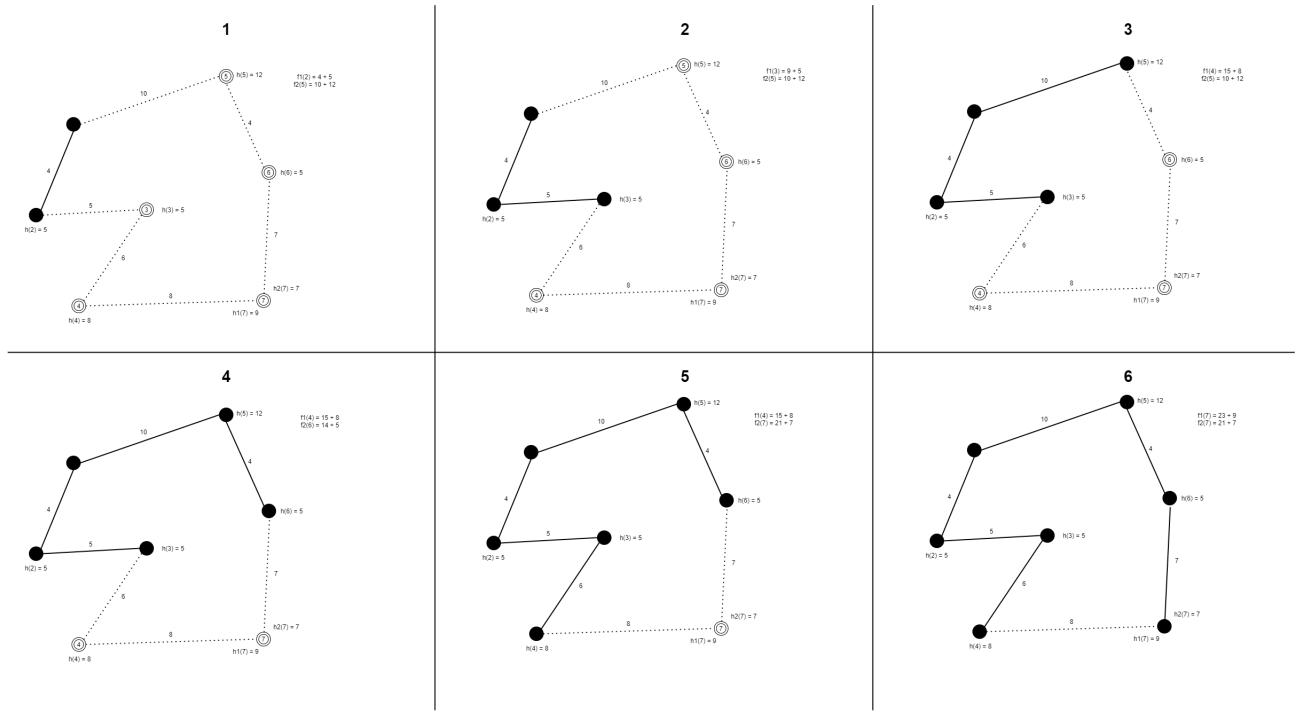
Rysunek 4.7: Graf użyty do przedstawienia działania algorytmu A*

4.4.1 Metoda 2-opt

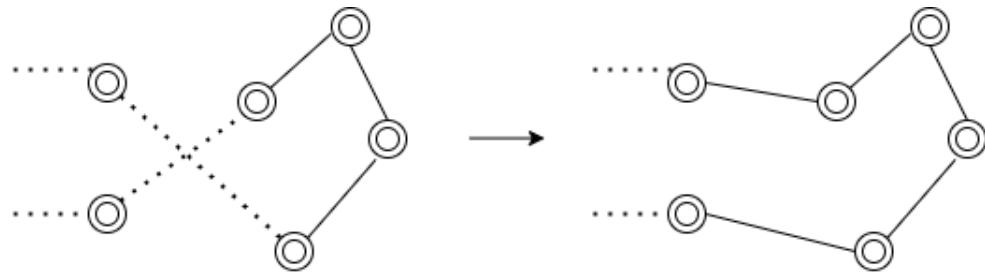
Metoda optymalizacyjna 2-opt polega na pozbyciu się z cyklu dwóch krawędzi w celu zastąpienia ich innymi krawędziami w taki sposób, aby otworzyć zupełnie inny cykl. Iteracje można powtarzać dla każdej pary krawędzi, oprócz tych sąsiadujących ze sobą, ponieważ ich zamienienie nie przyniosłoby żadnej modyfikacji. Metoda nie ma na celu zmiany położenia wierzchołków, jedynie kolejności ich odwiedzania. Po wykonaniu całej optymalizacji, należy sprawdzić która modyfikacja przyniosła najlepszy efekt skrócenia długości cyklu. W przypadku jeżeli żadna modyfikacja nie dała lepszego rozwiązania, nie należy modyfikować rozwiązania. Algorytm można wykonywać wielokrotnie, w ten sposób zostanie zrealizowane minimum lokalne. Dla lepszego przedstawienia działania metody 2-opt, należy spojrzeć na rysunek 5.4.

4.4.2 Propozycje optymalizacji

Aby rozważyć więcej tras zwracanych przez algorytmy zachłanne zaproponowano użycie współczynnika błędu przy dokonywaniu przez algorytm w danym momencie najlepszego dla niego wyboru. Współczynnik działa na zasadzie brania pod uwagę gorszego wyboru, aby zwiększyć liczbę przeglądanych odcinków. Ma to na celu możliwość znalezienia lepszego rozwiązania niż takiego jakie zwróciłby sam algorytm. Współczynnik błędu może być uruchamiany na każdym etapie budowania docelowej trasy. Zastosowanie współczynnika błędu może podnieść złożoność algorytmu nawet do $\theta(n^n)$, gdzie n jest



Rysunek 4.8: Poszczególne iteracje algorytmu A*



Rysunek 4.9: Metoda 2-opt

równe liczby wierzchołków. Stąd propozycją jest stosowanie współczynnika błędu tylko na wybranych trzech etapach budowania końcowej trasy.

5. Zbiór danych

W latach 2017-2018, miasto Białystok było podzielone na sektory w których odpowiednia firma była odpowiedzialna za wywóz śmieci. Model ten był praktykowany przez wiele lat. Każda z firm we własnym zakresie decydowała o trasach jakie mają pokonywać ciężarówki w celu zebrania odpadów komunalnych. Cały proces składał się z wielu kroków i budził wiele zastrzeżeń. W związku z tym w 19 października 2018 roku został zorganizowany hackathon podczas którego zespoły zmierzyły się z przedstawionymi problemami.

Drużyny do dyspozycji miały zestaw danych z 2017 roku. Dane te były gromadzone przez podwykonawców. Zbiory z kluczowymi informacjami znajdowały się w arkuszach programu Microsoft Excel. Każdy z dokumentów odpowiadał danemu miesiącowi. Niestety nie każdy arkusz składał się z tych samych kolumn. Jednym z największych problemów była odpowiednia agregacja danych. W wielu arkuszach brakowało kluczowych kolumn.

Przykładowy miesięczny raport zawierał informację z datą zdarzenia, krótkim opisem oraz identyfikatorem pojemnika na odpady. Ważną informacją był również nr rejestracyjny pojazdu oraz współrzędne pojazdu. Raporty zawierały wiele innych informacji takich jak frakcja lub rodzaj MGO. W miesięcznym raporcie znajdowało się kilkadziesiąt tysięcy rekordów z takimi informacjami.

Dzięki informacjom takim jak data załadunku kontenera, numerze rejestracyjnym pojazdu oraz współrzędnych pojazdu istnieje możliwość odtworzenia trasy przejazdu przykładowej ciężarówki. Kluczowe jest znalezienie punktu początkowego oraz punktu rozładunku. Następnie chronologiczne połączenie dat załadunku dla danej ciężarówki pozwoli na odtworzenie trasy jaka została przebyta.

Do obliczeń odległości pomiędzy kontenerami (punktami) zostało wykorzystane API udostępnione przez amerykańską spółkę Google LLC. Aby odpowiednio skorzystać z udostępnionej funkcjonalności należy przygotować zapytanie. Jako parametry wejściowe konieczne jest przekazanie współrzędnych dwóch punktów. W odpowiedzi API znajduje się informacja o odległości pomiędzy punktami. Wykorzystując udostępnione rozwiązanie można wyznaczyć odległości pomiędzy wszystkimi lokalizacjami zbiórki odpadów.

Analizując zbiory danych można zauważyc, że niektóre trasy zawierają więcej zdarzeń, inne mniej. Aby sprawdzić jak efektywny jest dany algorytm zostały wybrane trzy

trasy. Pierwsza trasa składa się z 111 punktów, druga z 150, a trzecia z 248. Dla każdej z wariacji została zmierzona droga jaką przebyła ciężarówka.

6. Badania

W tym rozdziale zostaną przedstawione wyniki optymalizacji dla tras przedstawionych w poprzednim rozdziale. Każda trasa została przedstawiona w oddzielnych podrozdziałach. W tabelach tła komórek z wartościami przebytego dystansu zostały zmienione w zależności od spełnianego warunku.

Algorytm genetyczny został zbadany dla trzech różnych wartości populacji N : 100, 1000 oraz 2000; czterech różnych wartości iteracji k : 100, 1000, 2000 oraz 4000. Dla każdej wariacji tych parametrów zostało zbadane krzyżowania PMX , OX oraz CX , z różnymi mutacjami: wstawiająca - *Insert*, zamieniająca - *Swap* oraz odwracająca - *Revert*. Współczynnik mutacji dla wszystkich uruchomień wynosi 0.1. Został zmierzony również czas dla każdego uruchomienia algorytmu. W tabelach kolorem jasno-żółtym są oznaczone wartości, które optymalizują trasę oryginalną, czyli są mniejsze. Kolorem żółtym zostały oznaczone komórki, które przedstawiają najlepszy wynik dla danej populacji i iteracji, pod warunkiem że był on lepszy od trasy oryginalnej. Kolorem zielonym został oznaczony w tabeli najlepszy globalne wyniki. Wszystkie krzyżowania i mutacje zostały zaimplementowane w .NET Core 3.1, a do samej obsługi algorytmu genetycznego, została wykorzystana biblioteka Accord.Genetic. Posiada ona interfejsy, pod które można połączyć własne metody mutacji, krzyżowania oraz funkcje celu.

Jednym z parametrów wejściowych algorytmu mrówkowego jest współczynnik parowania feromonów. To dzięki temu parametrowi wybierane są optymalne trasy, a te gorsze odrzucane. Ich wartości są wyrażane procentowo w zakresie od 1% do 99%. Dla każdej trasy została wybrana takie same wartości. Pomiar wyników został rozpoczęty od wartości 10%. W każdym korku wartość ta była zwiększana o kolejne 10% aż do 90%. Kolejnym istotnym parametrem jest ilość iteracji. Parametr ten oznacza ile razy mrówki mają wykonywać ulepszenie trasy. Do badań zostały dobrane trzy wartości: 50, 75, 100. Parametr ten ma bezpośredni wpływ na optymalizację ale również na czas jaki algorytm potrzebuje na wykonanie wszystkich operacji. W tabelach podobnie jak dla algorytmu genetycznego, kolorem jasno-żółtym oznaczono wyniki, które są lepsze od oryginalnej trasy. Kolorem żółtym najlepszy wynik dla danego współczynnika parowania feromonów, a kolorem zielonym najlepszy globalny wynik. Program obsługujący algorytm mrówkowy został stworzony za pomocą języka Python.

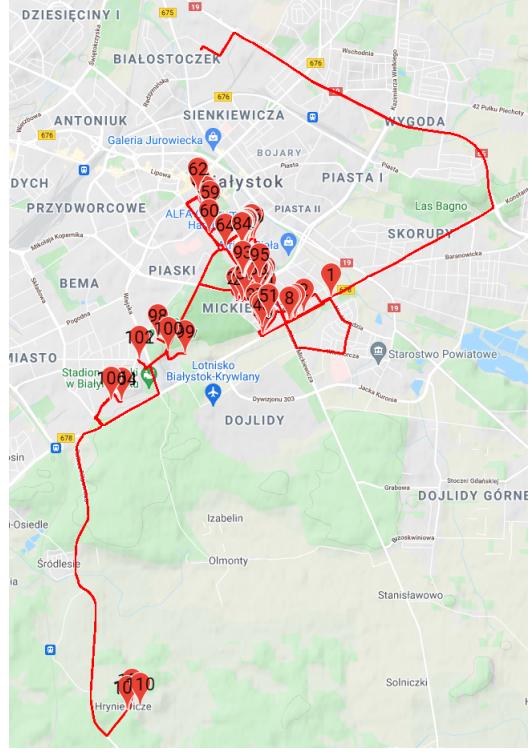
Przedstawicielami algorytmów zachłannych do wyznaczania tras był algorytm najbliższego sąsiada (ANS), algorytm najmniejszej krawędzi (ANK) oraz algorytm A*. Do wyznaczenia większej ilości tras w algorytmach zachłannych użyto współczynnika błędu w jednostce metrów: 0, 5, 10, 20, 30, 40, 50, 75 oraz 100. Współczynnik został zbadany na trzech różnych etapach algorytmu, aby zbadać większą ilość otrzymanych tras, wyłączany był po utworzeniu 100 000 tras do badań. Pierwsze badany etap obejmował uruchomienie współczynnika od początku działania danego algorytmu. Drugie badanie aktywowało współczynnik dopiero po utworzeniu $1/3$ całkowitej trasy. Ostatni eksperyment polegał na rozpoczęciu rozważania większej ilości tras po stworzeniu $2/3$ z całkowitej trasy. Następnie wszystkie otrzymane trasy zostały zoptymalizowane metodą 2-opt w celu znalezienia lepszego wyniku. Jeśli rezultat optymalizacji nie był lepszy, zostawał poprzedni wynik. W tabelach dotyczących wyników algorytmów zachłannych, kolorem żółtym zaznaczono najlepsze wyniki dla danego algorytmu, a kolorem zielonym najlepszy globalny wynik. Wszystkie algorytmy zostały zaimplementowane własnoręcznie w .NET Core.

6.1 Trasa I

Na rysunku 6.1 została przedstawiona oryginalna trasa ciężarówki na trasie pierwszej. Pokonała ona odległość 48897 metrów. Kierowca wyjechał z osiedla białostoczek i zaczął zbierać kontenery na osiedlach mickiewicza i centrum, na koniec odwiedził okolice stadionu miejskiego i pojechał na rozładunek do Hryniewicz.

6.1.1 Wyniki algorytmu genetycznego - Paweł

Algorytm wyznaczył trasę dla każdej możliwej kombinacji 10 razy. W tabelach dotyczących wyników algorytmu genetycznego, zostały przedstawione takie wyniki jak: średnia długość wyznaczonej trasy, najlepsza trasa z dziesięciu, średni czas wykonania oraz skuteczność algorytmu. W tabeli 6.1 zostały przedstawione średnie wyniki algorytmu genetycznego dla każdej kombinacji parametrów. Na pierwszy rzut oka widać, że wraz ze wzrostem parametru N osiągane były lepsze wyniki. Dla populacji 100 osobników, tylko dla krzyżowania CX i mutacji *Insert* przy 4000 iteracjach średni wynik zoptymalizował trasę. Dla $N = 1000$ zoptymalizowanych wyników jest już więcej. Od tej populacji zawsze najlepszy wynik dla poszczególnych wartości iteracji, został osiągnięty przez krzyżowanie CX i mutację *Insert*. W tej populacji udało się zoptymalizować trasę do 33843 metrów



Rysunek 6.1: Trasa I

przy 2000 iteracjach. W populacji 2000 osobników dla $k = 2000$ oraz krzyżowania CX i

Tabela 6.1: Trasa I - średnie wyniki algorytmu genetycznego dla poszczególnych parametrów wejściowych

N	k	PMX Insert	PMX Swap	PMX Revert	OX Insert	OX Swap	OX Revert	CX Insert	CX Swap	CX Revert
100	100	91502	96029	105868	102746	109380	115816	97015	112795	116291
100	1000	52959	67241	80988	58800	70722	93124	55599	67370	92538
100	2000	49747	61007	80468	52822	64938	89324	51134	61108	86920
100	4000	49368	59649	79494	50284	62967	98000	48571	57028	85739
1000	100	55008	67898	56269	64427	63525	69052	49959	53471	58104
1000	1000	37347	42081	49527	41809	46600	60650	33768	37948	50090
1000	2000	39158	40581	50985	40903	45041	64489	33843	38791	48072
1000	4000	37951	42112	50359	40036	47795	62245	34961	37842	49857
2000	100	51624	48902	49112	62336	61150	57542	40969	44643	42287
2000	1000	36492	39233	43081	38086	45761	53166	32571	35453	39076
2000	2000	36410	40379	40711	39554	41320	52030	32326	35353	40382
2000	4000	35406	38159	42454	39145	44137	51350	32667	34999	48295

mutacji *Insert*, został znaleziony najlepszy wynik globalny 32326 metrów. Również można zauważyc, że od $k = 1000$ dla wszystkich parametrów z wyjątkiem krzyżowania *OX* oraz mutacji *Revert* udało się zoptymalizować trasę. Porównując krzyżowania to *PMX* najlepiej sobie radziło do $N = 100$ oraz $k = 2000$, dla kolejnych populacji i iteracji, najlepsze wyniki osiągnęło krzyżowanie *CX*. Z mutacji najlepiej poradziła sobie *Insert*, a najgorzej *Revert*.

W tabeli 6.2 została przedstawiona procentowa ilość wyników lepszych od długości trasy oryginalnej, czyli skuteczność algorytmu genetycznego. Przy populacji 100 osobników, ani razu algorytm nie osiągnął 100% skuteczności. Pierwszy raz udało się to osiągnąć dla populacji 1000 osobników przy 1000 iteracjach. Krzyżowanie *PMX* oraz *CX* osiągnęło 100% skuteczności w połączeniu z każdą mutacją dla populacji 2000 osobników oraz $k = 2000$ i $k = 4000$. Krzyżowanie *OX* z mutacją *Revert* ani razu nie osiągnęło 100% skuteczności. Porównując mutacje, najsłabsza skuteczność osiągnęła *Revert*.

Tabela 6.2: Trasa I - skuteczność algorytmu genetycznego dla poszczególnych parametrów wejściowych

N	k	PMX Insert	PMX Swap	PMX Revert	OX Insert	OX Swap	OX Revert	CX Insert	CX Swap	CX Revert
100	100	0%	0%	0%	0%	0%	0%	0%	0%	0%
100	1000	30%	0%	0%	0%	0%	0%	10%	0%	0%
100	2000	50%	0%	0%	30%	0%	0%	30%	0%	0%
100	4000	40%	0%	0%	40%	0%	0%	50%	10%	0%
1000	100	10%	0%	0%	0%	0%	0%	80%	20%	10%
1000	1000	100%	100%	50%	100%	90%	0%	100%	100%	30%
1000	2000	100%	100%	30%	100%	90%	0%	100%	100%	60%
1000	4000	100%	100%	40%	100%	60%	0%	100%	100%	50%
2000	100	30%	40%	50%	0%	0%	0%	90%	80%	100%
2000	1000	100%	100%	90%	100%	70%	20%	100%	100%	90%
2000	2000	100%	100%	100%	100%	100%	40%	100%	100%	100%
2000	4000	100%	100%	100%	100%	100%	40%	100%	100%	100%

W tabeli 6.3 zostały pokazane najlepsze wyniki, czyli wartość minimalna z 10 wywołań algorytmu dla poszczególnych parametrów wejściowych. W populacji 100 osobników, tylko w przypadku dwóch krzyżowań *PMX*, *CX* oraz mutacji *Insert* zostały osiągnięte wyniki lepsze od oryginału, ale nie zostało to osiągnięte dla 100 iteracji. Dla tej populacji najlepszy wynik został znaleziony dla krzyżowania *PMX* oraz mutacji *Insert* i wyniósł 40964 metrów. W populacji 1000 osobników najlepiej już poradziło sobie krzyżowanie *CX* w połączeniu z mutacją *Insert*. Ta kombinacja dla 2000 iteracji znalazła najlepszy wynik w tej populacji 32344 metrów. Dla $N = 2000$ tylko krzyżowanie *OX* przy 100 iteracjach, ani razu nie osiągnęło wyniku lepszego od oryginału. Dla wszystkich innych kombinacji parametrów najlepszy wynik z 10 uruchomień algorytmu zoptymalizował trasę. Najlepszy globalny wynik został znaleziony dla $N = 2000$, $k = 100$, krzyżowania *CX* oraz mutacji *Insert*, wyniósł 30939 metrów. Od $N = 1000$ i $k = 4000$ zawsze najlepszy wynik wystąpił dla krzyżowania *CX* i mutacji *Insert*.

W tabeli 6.4 zostały przedstawione średnie czasy wywołań algorytmów dla poszczególnych kombinacji parametrów. Tylko w dwóch przypadkach najszybsze nie było

Tabela 6.3: Trasa I - najlepsze wyniki algorytmu genetycznego dla danych parametrów wejściowych

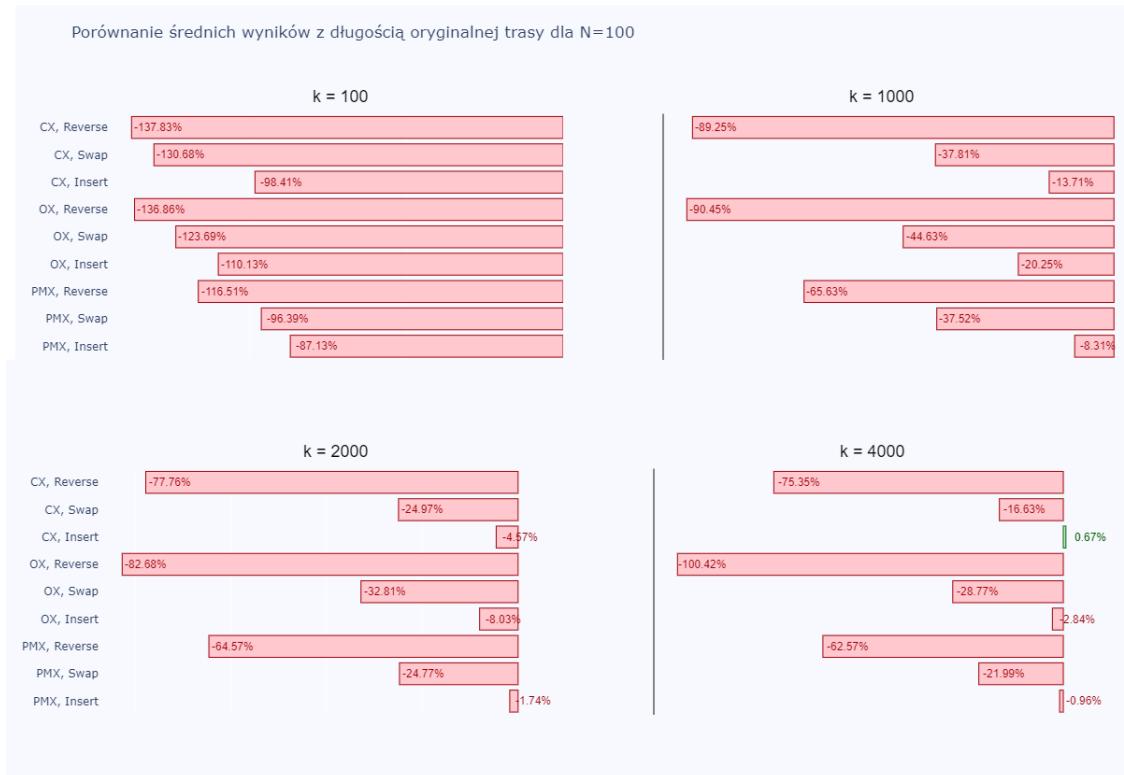
N	k	PMX Insert	PMX Swap	PMX Revert	OX Insert	OX Swap	OX Revert	CX Insert	CX Swap	CX Revert
100	100	80385	87226	81154	90764	99185	101137	72268	103408	92859
100	1000	45076	62481	67443	53948	60699	78382	45134	56918	79955
100	2000	40964	53316	70100	45500	55396	80719	43142	54430	79636
100	4000	44853	52397	69287	40628	56153	91440	41805	45953	72035
1000	100	45769	59523	50232	56634	58272	62965	40804	42391	41647
1000	1000	34630	36034	42016	39283	44320	53480	32433	35227	37569
1000	2000	35051	36704	45852	35709	41317	57478	32334	35576	39953
1000	4000	33268	37649	44333	35718	41883	49322	33089	34642	40220
2000	100	44493	42497	42138	56466	56884	52010	36349	38830	38461
2000	1000	33609	36100	37047	34801	38625	40817	30939	33284	36203
2000	2000	33257	36216	37967	34591	36458	47022	31118	32798	36203
2000	4000	31347	36834	40193	34269	37693	43242	31118	33245	35271

krzyżowanie *CX* oraz mutacja *Revert*. Najlepszy średni wynik optymalizacji pomiędzy populacjami 1000 osobników, a 2000 osobników poprawił się o około 1442 metrów (Tab. 6.1), ale czas trwania algorytmu dla tych parametrów: $N = 1000$, $k = 1000$, krzyżowanie *CX* i mutacja *Insert* a $N = 2000$, $k = 2000$, krzyżowanie *CX* i mutacja *Insert* wzrósł z 1.7 sekundy do 8.33 sekundy, ponad czterokrotnie. Najwolniej z krzyżowań radzi sobie krzyżowanie *PMX*, a najszybciej *CX*.

Tabela 6.4: Trasa I - średnie czasy wykonywania algorytmu dla danych parametrów wejściowych

N	k	PMX Insert	PMX Swap	PMX Revert	OX Insert	OX Swap	OX Revert	CX Insert	CX Swap	CX Revert
100	100	0,03	0,03	0,02	0,02	0,02	0,02	0,02	0,01	0,01
100	1000	0,19	0,19	0,18	0,15	0,15	0,14	0,15	0,12	0,12
100	2000	0,37	0,36	0,34	0,29	0,28	0,27	0,28	0,24	0,23
100	4000	0,70	0,70	0,71	0,60	0,58	0,55	0,53	0,49	0,47
1000	100	0,35	0,53	0,34	0,24	0,23	0,26	0,22	0,21	0,23
1000	1000	2,31	2,34	2,34	1,99	1,92	1,84	1,76	1,70	1,71
1000	2000	4,51	4,57	4,62	3,77	3,86	3,60	3,42	3,38	3,21
1000	4000	8,88	9,10	8,81	7,38	7,46	7,36	6,72	6,65	6,23
2000	100	0,80	0,80	0,79	0,59	0,58	0,57	0,61	0,55	0,54
2000	1000	5,64	5,58	5,37	4,48	4,44	4,32	4,12	3,96	3,88
2000	2000	10,22	10,28	10,23	9,02	8,96	8,79	8,33	8,02	7,86
2000	4000	20,77	20,85	20,83	17,74	17,98	17,99	16,06	15,74	7,26
2000	4000	20,77	20,85	20,83	17,74	17,98	17,99	16,06	15,74	15,26

Rysunki 6.2, 6.3 oraz 6.4 przedstawiają wykresy, które porównują o ile różni się rozwiązanie od długości trasy oryginalnej, kolejno dla $N = 100$, $N = 1000$ oraz $N = 2000$. W populacji 100 osobników (Rys. 6.2), tylko w jednym przypadku udało się zoptymalizować trasę o 0.67%, co jest znikomą wartością. W pozostałych przypadkach wyniki są gorsze o kilkanaście, a nawet kilkudziesiąt procent.

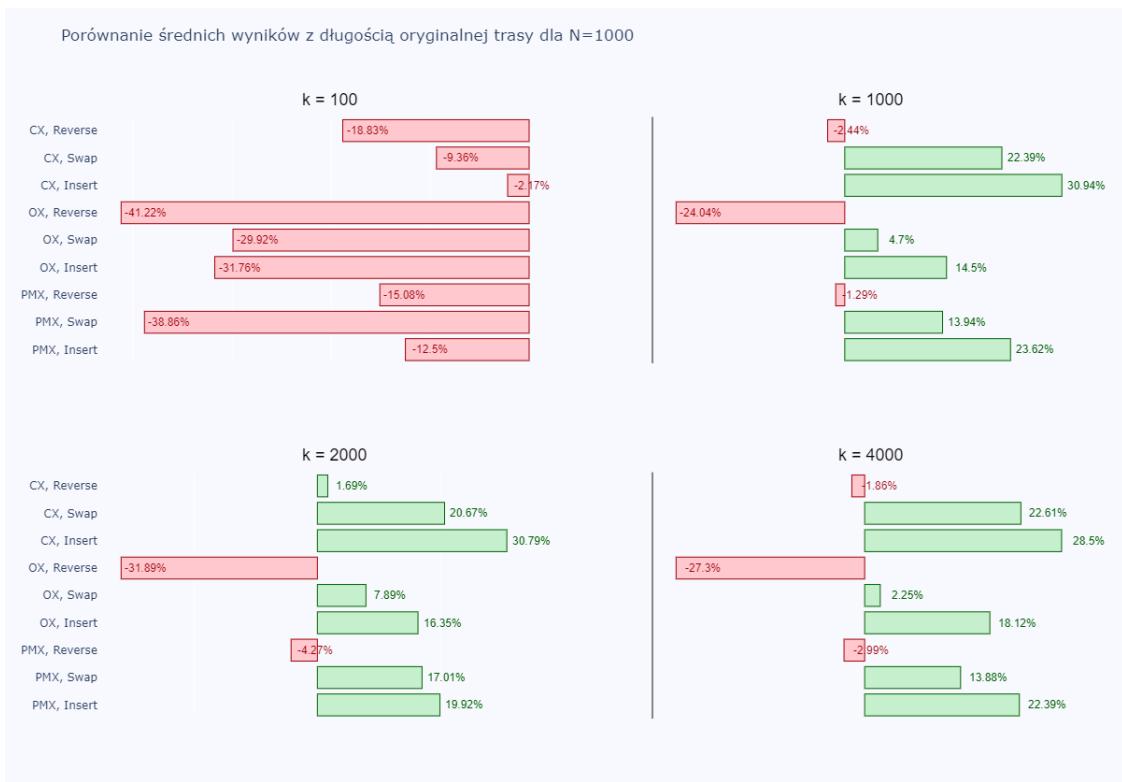


Rysunek 6.2: Trasa I - porównanie średnich wyników z długością oryginalnej trasy dla N=100

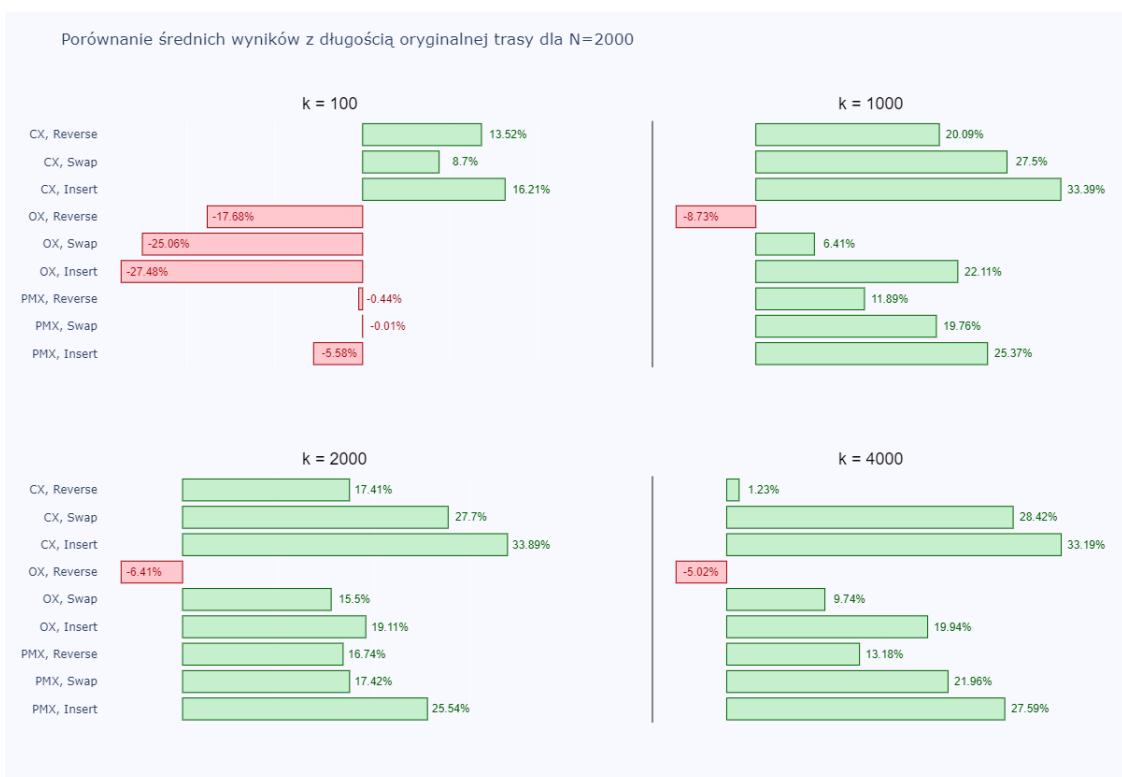
W populacji $N = 1000$ (Rys. 6.3) dla 100 iteracji algorytm znajdował trasę o kilkanaście procent gorsze. Dla 1000 iteracji udało się zoptymalizować trasę średnio już o 30.94% dla krzyżowania CX. To krzyżowanie dla mutacji Insert oraz Swap zoptymalizowało trasę średnio o ponad 20%. Krzyżowanie PMX oraz OX poradziło sobie bardzo podobnie w tej populacji. Pierwsze maksymalnie średnio zoptymalizowało trasę o 23.62%, a drugie o 18.12%.

W populacji $N = 2000$ (Rys. 6.4) krzyżowanie CX dla każdej mutacji zoptymalizowało trasę od 8.7% do 33.89%. Można zauważyć, że średni najlepszy wynik nie poprawiał się od $k = 1000$. Krzyżowanie PMX oraz OX poprawiło się o kilka procent porównując do poprzedniej populacji. Pierwsze maksymalnie średnio zoptymalizuje trasę o 27.59%, a drugie o 22.11%.

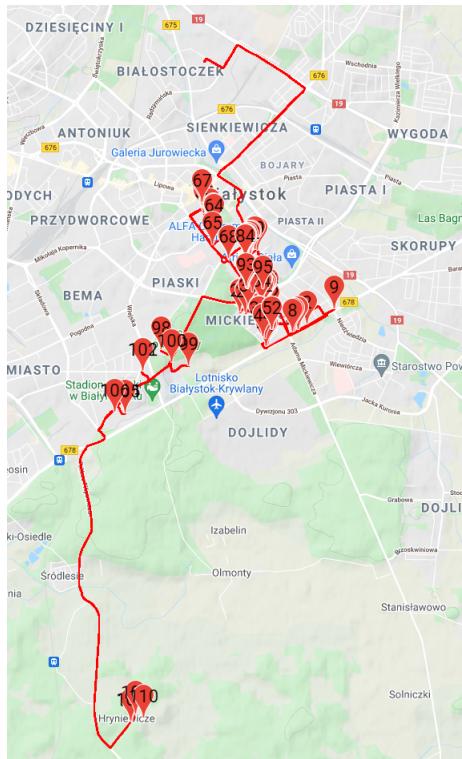
Na rysunku 6.5 przedstawiono najlepszą trasę jaką znalazł algorytm genetyczny. Od razu można zauważyć, że inny jest pierwszy odwiedzony kontener w stosunku do trasy oryginalnej. Ostatnie kontenery odwiedzane są w tej samej kolejności.



Rysunek 6.3: Trasa I - porównanie średnich wyników z długością oryginalnej trasy dla N=1000



Rysunek 6.4: Trasa I - porównanie średnich wyników z długością oryginalnej trasy dla N=2000



Rysunek 6.5: Trasa I - algorytm genetyczny.

6.1.2 Wyniki algorytmu mrówkowego - Kamil

Algorytm został uruchomiony dziesięciokrotnie dla każdej możliwej kombinacji parametrów wejściowych. W odpowiedzi program przedstawił w odpowiedniej kolejności punkty jakie zostały zawarte w rozwiązaniu, przebyty dystans oraz czas wykonania algorytmu. Z zebranych danych została wyliczona średnia, został wyznaczony najlepszy wynik oraz wyliczono sumę punktów, których łączny przebyty dystans jest krótszy od oryginału. Przy wykorzystaniu tych miar można wykonać porównanie z oryginalną trasą przebytą w rzeczywistości przez ciężarówkę.

Każda z tabel przedstawiająca wynik algorytmu mrówkowego składa się z takiej samej ilości kolumn oraz wierszy. W kolumnie z oznaczeniem k została zaprezentowana ilość przeprowadzonych iteracji. Wartości te zaczynają się od 50 i są zwiększane co 25 do 100. Każda kolumna przedstawia współczynnik parowania feromonów. Wartości te zaczynają się od 0.1 i są zwiększane co 0.1 do 0.9.

W tabeli numer 6.5 znajdują się średnie wyniki z dziesięciu pomiarów dla takich samych parametrów. Najlepsze wyniki występują dla współczynnika parowania znajdującego się w zakresie od 0.1 do 0.4 i oscylują poniżej progu 40000 metrów. Wyjątkiem jest jedynie współczynnik parowania równy 0.1 jeśli zostało wykonanych 50 iteracji. W tym przypadku

wartość jest większa od oryginalnej i wynosi 50543. Na tym przykładzie można świetnie zauważyc jak istotna jest ilość iteracji jaka zostanie wykonana. W kolejnych krokach ilość iteracji była zwiększana, a średnia odległość ulegała poprawie. Dla stu iteracji odnotowano najlepszą średnią równą 29479.

Tabela 6.5: Trasa I - średnie wyniki algorytmu mrówkowego dla poszczególnych parametrów wejściowych

k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
50	50543	30430	33453	35171	42787	41658	45557	44527	44370
75	32810	30993	33539	31485	44500	44022	45114	48684	46820
100	29479	34264	31697	39139	41934	47499	49393	47152	46382

Najlepsze wyniki algorytmu mrówkowego dla trasy numer 1 zostały przedstawione w tabeli 6.6. Spośród najlepszych wyników dla danej wariacji, żadna próba nie została sklasyfikowana jako gorsza od oryginalnej odległości. Najgorszy wynik wystąpiły dla najniższej wartości iteracji oraz najniższego współczynnika parowania. Ta trasa nadal jest lepsza od oryginalnej o 7 439 metrów. Najlepsza wartość jaką udało się osiągnąć to 27131. Jest to wartość jaka została osiągnięta dla 75 ilości iteracji oraz dla współczynnika parowania wynoszącego 0.4. W najlepszym wypadku trasa została ulepszona o około 55%.

Tabela 6.6: Trasa I - najlepsze wyniki algorytmu mrówkowego dla poszczególnych parametrów wejściowych

k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
50	41458	27288	27554	27257	32569	27297	30815	29160	32499
75	28137	27998	27583	27131	29459	29298	27332	36046	32471
100	27259	28194	27606	27490	27193	34429	37313	35244	31738

Istotnym parametrem jaki należy wziąć pod uwagę jest to, jak często algorytm jest w stanie poprawić oryginalną trasę. W tym celu została stworzona tabela numer 6.7. Możemy zauważyc z jaką skutecznością program jest w stanie poprawić trasę. Wyniki potwierdzają wcześniejsze wnioski. Najlepsze wyniki można zestawić ze współczynnikiem parowania feromonów z zakresu od 0.1 do 0.4. Co więcej dla ilości iteracji równej 75 dla każdego ze współczynników trasa została polepszona.

Oprócz samych prac nad optymalizacjami należy zwrócić uwagę z jaką szybkością algorytm jest w stanie przedstawić rozwiązanie. W tabeli 6.8 przedstawiony jest średni czas wykonania algorytmu. Oczywistym wydaje się, że wraz ze wzrostem ilości wykonanych iteracji wzrasta czas wykonania programu. Średnio dla programu z ilością iteracji równą

Tabela 6.7: Trasa I - skuteczność algorytmu mrówkowego

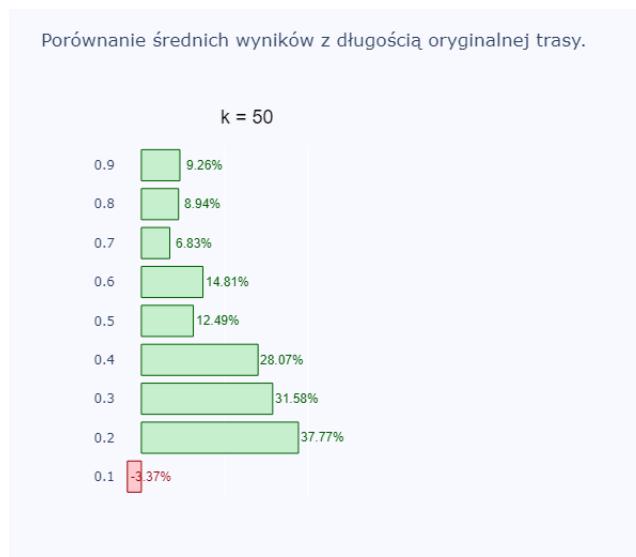
k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
50	50%	100%	100%	100%	80%	80%	60%	70%	80%
75	100%	100%	100%	100%	60%	40%	60%	40%	60%
100	100%	100%	100%	80%	70%	60%	20%	40%	70%

50 czas oczekiwania wynosił blisko 13 sekund. Dla 75 iteracji było to 21.732 sekundy, a dla 100 31.547. Zestawiając wyniki czasowe z procentowymi wartościami tras jakie zostały wyprodukowane przez program, można zauważyc, że ilość większa ilość iteracji nie musi oznaczać optymalizacji trasy.

Tabela 6.8: Trasa I - średnie wyniki czasu wykonania algorytmu mrówkowego dla parametru ilości iteracji

iteracje	50	75	100
czas	12.945	21.732	31.547

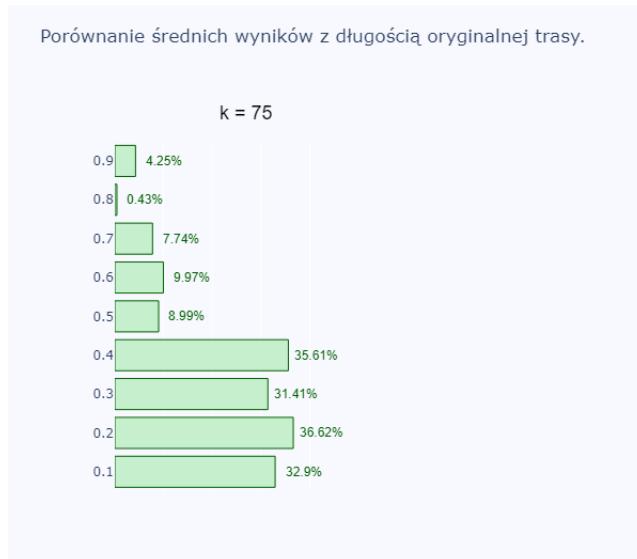
Na wykresie 6.6 przedstawione zostało procentowe porównanie średniej wartości dla każdego współczynnika parowania i liczbie iteracji równej 50. Z wykresu można odczytać, że algorytm mrówkowy w najgorszym wygenerował trasę o 3.37% gorszą od oryginalnej. Pozostałe przypadki pokazują jednak, że profit optymalizacji jest wyższy od tego spadku. Trasa została poprawiona o 37.77%.



Rysunek 6.6: Procentowy rozkład średnich wartości dla 50 iteracji

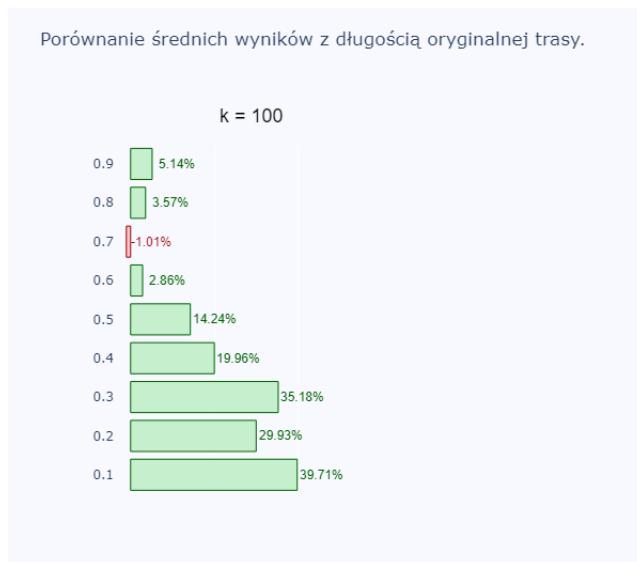
Na wykresie 6.7 zwiększoła została ilość iteracji do 75. Dzięki tej aktualizacji średnie wyniki są większe od pierwotnej długości trasy. Warte odnotowania jest również to, że

wcześniej najgorsza wartość jest teraz jedną z najlepszych. Różnica 25 iteracji przełożyła się na poprawę o 36.27%. Współczynniki z zakresu 0.2-0.4 w dalszym ciągu przekładają się na najwyższe średnie wyniki.



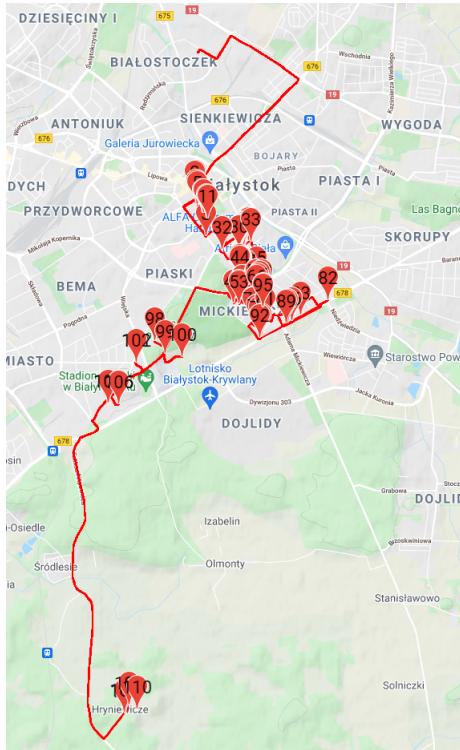
Rysunek 6.7: Procentowy rozkład średnich wartości dla 75 iteracji

Najwyższa średnia optymalizacja została odnotowana dla 100 iteracji oraz współczynnika parowania wynoszącego 0.1. Na uwagę zasługuje fakt, iż średnia długość trasy dlatego współczynnik przy 50 iteracjach był najgorszą średnią. Na wykresie 6.8 można zauważyć rozkład wszystkich współczynników.



Rysunek 6.8: Procentowy rozkład średnich wartości dla 100 iteracji

Na rysunku 6.9 przedstawiono najlepszą trasę jaką znalazł algorytm mrówkowy. Ciężarówka na tej trasie zaczyna odwiedzanie kontenerów od centrum, w przeciwieństwie do oryginalnej trasy.



Rysunek 6.9: Trasa I - algorytm mrówkowy

6.1.3 Wyniki algorytmów zachłannych - Przemysław

W tabeli 6.9 zostały przedstawione najlepsze wyniki algorytmów zachłannych dla każdej kombinacji współczynnika błędu uruchamianego na trzech etapach budowania całkowitej trasy. Algorytm A* wyznaczył najlepszą trasę o długości 30205 metrów oraz jako jedyny wyznaczył bardzo dużą ilość tras poniżej 31000 metrów. Najlepsza trasa otrzymana przez algorytm najbliższego sąsiada wyniosła 31895 metrów, natomiast algorytm najmniejszej krawędzi zwrócił trasę o długości 32015 metrów. Od razu można zauważyć, że po zastosowaniu współczynnika na pierwszym etapie trasy najgorzej wypadł algorytm najmniejszej krawędzi dając wyniki aż powyżej 40000 metrów. Natomiast algorytm najbliższego sąsiada oraz algorytm A* najsłabiej poradzili sobie tworząc dodatkowe trasy do badań na trzecim etapie działania.

Tabela 6.10 przedstawia procentową poprawę oryginalnej trasy nr 1. Wszystkie otrzymane trasy przez każdy z trzech algorytmów dały lepszy wynik od oryginalnej. Najlepszy

wynik otrzymany przez algorytm A* poprawił trasę aż o 38.23%. Na pierwszy rzut oka można zauważyć, że najwyższe optymalizacje zostały wygenerowane przez algorytm A*. Najgorsze optymalizacje otrzymano przy zastosowaniu algorytmu najmniejszej krawędzi w pierwszej fazie budowania tras, wszystkie wyniki były poniżej 20%. Nieco lepiej od poprzednika wypadł algorytm najbliższego sąsiada, jednak najbardziej optymalny wynik jest w obu zbliżony równy około 34%.

Tabela 6.9: Trasa I - wyniki algorytmów zachłannych

Błąd (m)	ANS-1	ANS-2	ANS-3	ANK-1	ANK-2	ANK-3	A*-1	A*-2	A*-3
0	37224	31895	38707	42343	32480	38497	31235	30906	36361
5	37224	32552	37925	42343	32479	39651	30906	30855	37143
10	37224	32552	38270	42335	32479	38496	30577	30855	36752
20	36840	32223	37879	40319	32479	38496	30577	30855	37143
30	36840	31895	38270	42335	33453	40421	31235	30205	36361
40	36840	32552	38270	42335	32479	38496	30577	30529	36317
50	37218	32552	37879	42335	32479	39651	30577	30854	36708
75	36842	32552	38660	40319	34103	38496	30899	30205	37098
100	36466	31895	38124	40319	32015	40421	31228	30205	37098

Tabela 6.10: Trasa I - optymalizacja wyników względem oryginalnej trasy

Błąd (m)	ANS-1	ANS-2	ANS-3	ANK-1	ANK-2	ANK-3	A*-1	A*-2	A*-3
0	23,87%	34,77%	20,84%	13,40%	33,57%	21,27%	36,12%	36,79%	25,64%
5	23,87%	33,43%	22,44%	13,40%	33,58%	18,91%	36,79%	36,90%	24,04%
10	23,87%	33,43%	21,73%	13,42%	33,58%	21,27%	37,47%	36,90%	24,84%
20	24,66%	34,10%	22,53%	17,54%	33,58%	21,27%	37,47%	36,90%	24,04%
30	24,66%	34,77%	21,73%	13,42%	31,58%	17,33%	36,12%	38,23%	25,64%
40	24,66%	33,43%	21,73%	13,42%	33,58%	21,27%	37,47%	37,56%	25,73%
50	23,88%	33,43%	22,53%	13,42%	33,58%	18,91%	37,47%	36,90%	24,93%
75	24,65%	33,43%	20,94%	17,54%	30,26%	21,27%	36,81%	38,23%	24,13%
100	25,42%	34,77%	22,03%	17,54%	34,53%	17,33%	36,14%	38,23%	24,13%

Zastosowanie metody 2-opt na utworzonych trasach znalazło nowy najlepszy wynik dla algorytmu A* testowanego w pierwszej fazie budowania tras. Jak można zauważyć w tabeli 6.11 po zastosowaniu optymalizacji nowa najlepsza trasa uzyskała wynik 29 965 metrów. Dzięki optymalizacji trasa wygenerowana przez algorytm najmniejszej krawędzi zmniejszyła się do 30855 metrów. Dla algorytmu najbliższego sąsiada optymalizacja nie przyniosła trasy o mniejszej długości od poprzednich. Nowa najlepsza optymalizacja wynosi 38.72% względem oryginalnej trasy, a najgorszy wynik to zaledwie 13.40% poprawy długości. Tabela 6.12 przedstawia wyniki optymalizacji względem trasy oryginalnej po zastosowaniu metody 2-opt.

Tabela 6.11: Trasa I - zastosowanie metody 2-opt na powstały trasach

Błąd (m)	ANS-1	ANS-2	ANS-3	ANK-1	ANK-2	ANK-3	A*-1	A*-2	A*-3
0	32751	31895	35997	42343	32480	38497	31235	30906	36361
5	32751	32552	37925	42343	30855	39651	30906	30205	36708
10	32751	32552	35997	40319	30855	38496	29965	30205	36708
20	32751	31895	37879	40319	30855	38496	29965	30205	36708
30	32751	31895	35997	40319	33453	38496	31235	30205	36361
40	32751	32552	35997	40319	30855	38496	29965	30529	36317
50	32751	32552	37879	40319	30855	39651	29965	30205	36708
75	32751	32552	35997	40319	34103	38496	30899	30205	36708
100	32751	31895	35997	40319	32015	38496	31228	30205	36708

Tabela 6.12: Trasa I - optymalizacja powstały tras przez metodę 2-opt

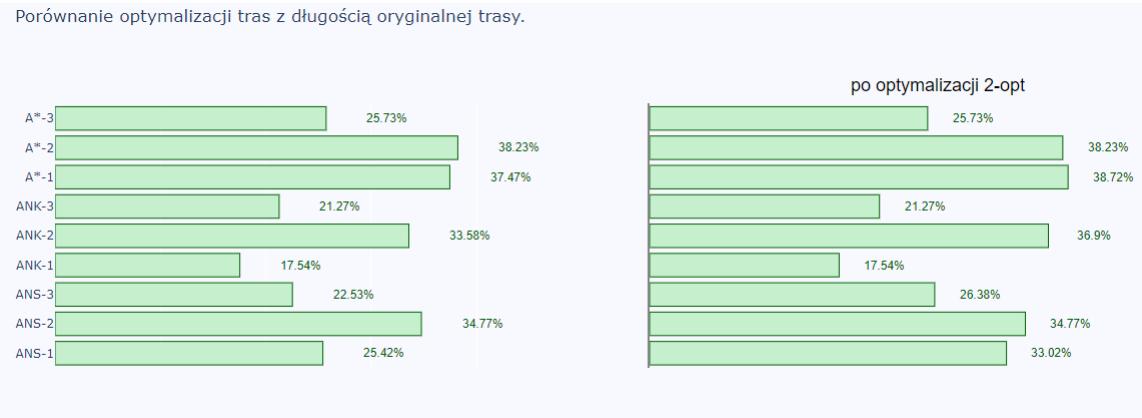
Błąd (m)	ANS-1	ANS-2	ANS-3	ANK-1	ANK-2	ANK-3	A*-1	A*-2	A*-3
0	33,02%	34,77%	26,38%	13,40%	33,57%	21,27%	36,12%	36,79%	25,64%
5	33,02%	33,43%	22,44%	13,40%	36,90%	18,91%	36,79%	38,23%	24,93%
10	33,02%	33,43%	26,38%	17,54%	36,90%	21,27%	38,72%	38,23%	24,93%
20	33,02%	34,77%	22,53%	17,54%	36,90%	21,27%	38,72%	38,23%	24,93%
30	33,02%	34,77%	26,38%	17,54%	31,58%	21,27%	36,12%	38,23%	25,64%
40	33,02%	33,43%	26,38%	17,54%	36,90%	21,27%	38,72%	37,56%	25,73%
50	33,02%	33,43%	22,53%	17,54%	36,90%	18,91%	38,72%	38,23%	24,93%
75	33,02%	33,43%	26,38%	17,54%	30,26%	21,27%	36,81%	38,23%	24,93%
100	33,02%	34,77%	26,38%	17,54%	34,53%	21,27%	36,14%	38,23%	24,93%

W tabeli 6.13 znajdują się czasy działania algorytmów. Dłuższy czas wykonywania nie oznacza lepszego wyniku dla algorytmów zachłannych. Najlepszą trasę udało się uzyskać w czasie 7.91 sekundy.

Tabela 6.13: Trasa I - czasy wykonywania algorytmów

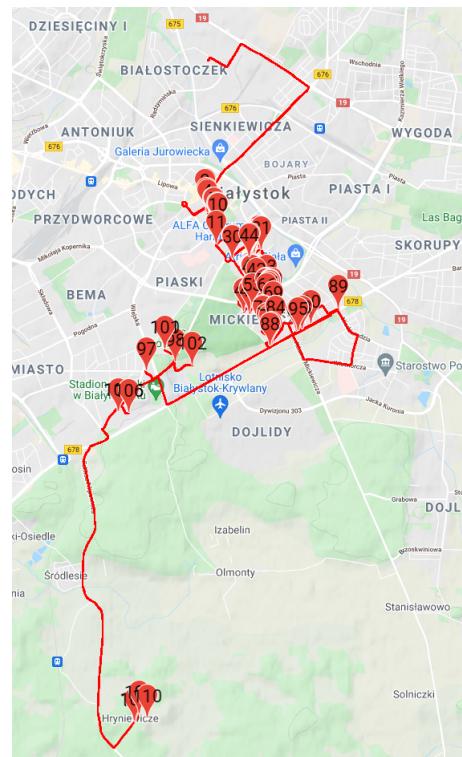
Błąd (m)	ANS-1	ANS-2	ANS-3	ANK-1	ANK-2	ANK-3	A*-1	A*-2	A*-3
0	3,18	0,33	0,31	0,27	0,35	0,3	2,04	0,33	0,3
5	2,92	0,31	0,34	0,31	1,83	4,42	4,25	3,59	0,3
10	1,56	0,31	12,14	1,39	2,07	11,75	7,91	3,9	0,32
20	16,27	0,32	14,1	2,61	2,13	13,85	8,03	7,84	0,33
30	26,95	0,32	22,51	6,42	2,09	19,54	8,2	8,22	0,36
40	43,77	0,31	23,73	13,27	2,2	19,56	8,29	17,18	2,27
50	15,74	0,33	23,29	14,32	2	20,05	8,29	23,84	9,16
75	25,51	0,33	24,11	14,75	2,2	20,81	34,28	5,22	11,01
100	35,77	0,31	24,76	21,13	11,57	21,32	13	6,1	11,39

Optymalizacja metodą 2-opt poprawiła wyniki we wszystkich zastosowanych algorytmach, co dobrze widać na rysunku 6.10. Algorytm najbliższego sąsiada poprawił dolny najlepszy wynik z 22.53% na 26.38%. Algorytm najmniejszej krawędzi natomiast zoptymalizował górny najlepszy wynik z 33.58% na 36.90%. Delikatnie lepiej wypadł także po optymalizacji algorytm A*, który swój górny wynik ulepszył z 38.23% na 38.72%.



Rysunek 6.10: Trasa I - porównanie optymalizacji algorytmów z zastosowanie metody 2-opt

Na rysunku 6.11 przedstawiono najlepszą trasę jaką znalazł algorytm zachłanny. Ciężarówka na tej trasie zaczyna odwiedzanie kontenerów od centrum, w przeciwnieństwie do oryginalnej trasy.



Rysunek 6.11: Trasa I - algorytm zachłanny

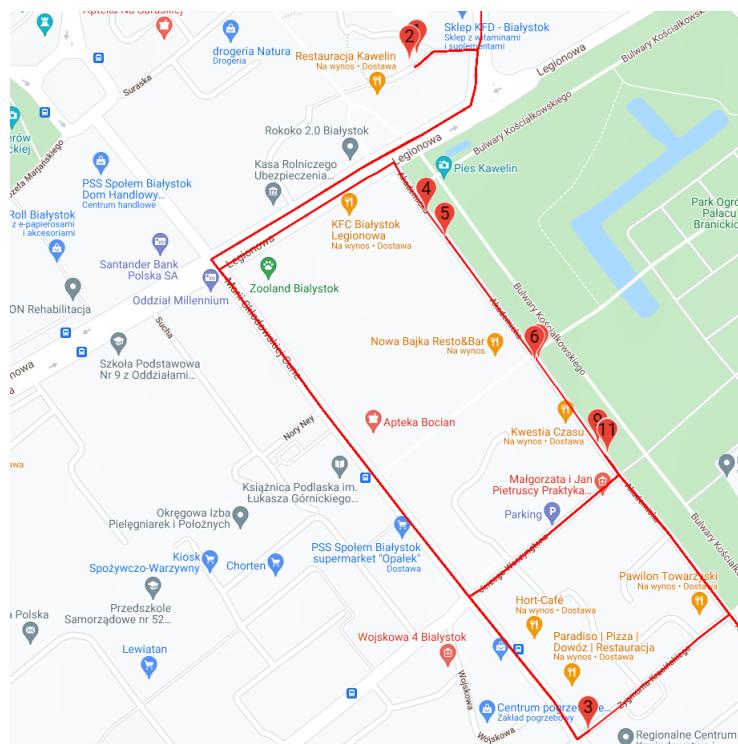
6.1.4 Porównanie algorytmów - wspólnie

Najlepszy wyniki algorytmów nie są takie same. Wszystkie z wyników są lepsze od oryginalnej trasy. Algorytm genetyczny znalazł najlepszy wynik równy 30939. Jest to wynik o 37% lepszy od pierwotnej trasy. Taki wynik udało się otrzymać w 8.33 sekundy. Algorytm

mrówkowy wygenerował trasę o łącznej długości 27131 metrów co jest lepszym wynikiem od poprzedniego algorytmu. Oryginalna trasa została poprawiona o 45%. Czas jaki był potrzebny na polepszenie ścieżki to 21.73 sekundy. Czas oczekiwania jest więc dłuższy od czasu jaki potrzebuje algorytm genetyczny. Najlepsza trasa, która uzyskały algorytmy zachłanne wynosiła 30205 metrów, a po optymalizacji metodą 2-opt trasa zmniejszyła się do 29965 metrów. Wynik ten jest o 39% lepszy od trasy początkowej. Wynik taki udało się otrzymać w 7.22 sekund.

Porównując dystans to najsłabiej sobie poradził algorytm genetyczny. Trasa ta została jednak wyznaczona w czasie około trzy razy krótszym niż algorytm mrówkowy, a wynik był gorszy o tylko 8%. Algorytm A* poradził sobie podobnie jak algorytm genetyczny w czasie o sekundę krótszym znalazły o 2% lepszą trasę.

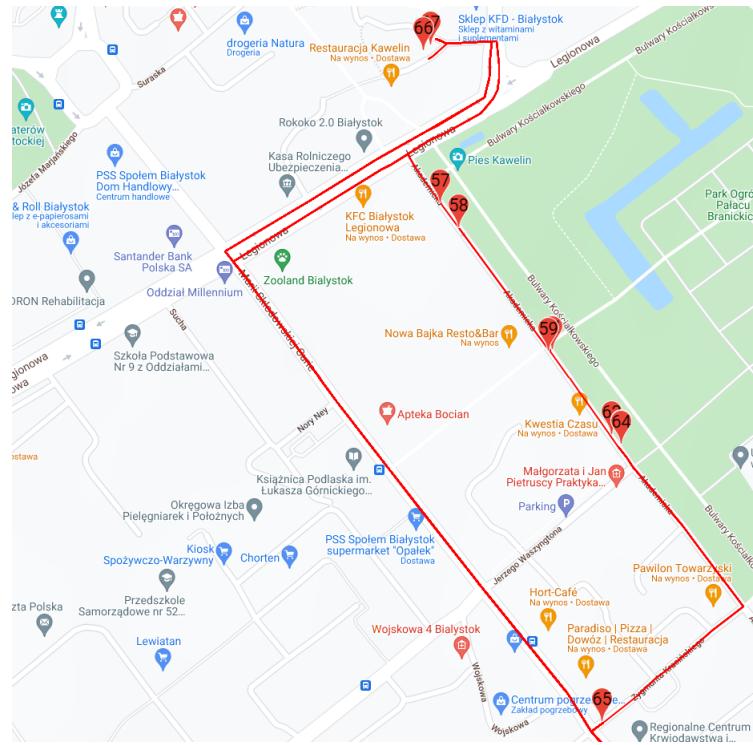
Na zdjęciach 6.13, 6.12, 6.14 zostały pokazane przykładowe różnice w znalezionych trasach przez algorytmy. Różnice te dotyczą kolejności odwiedzania punktów dla takiego



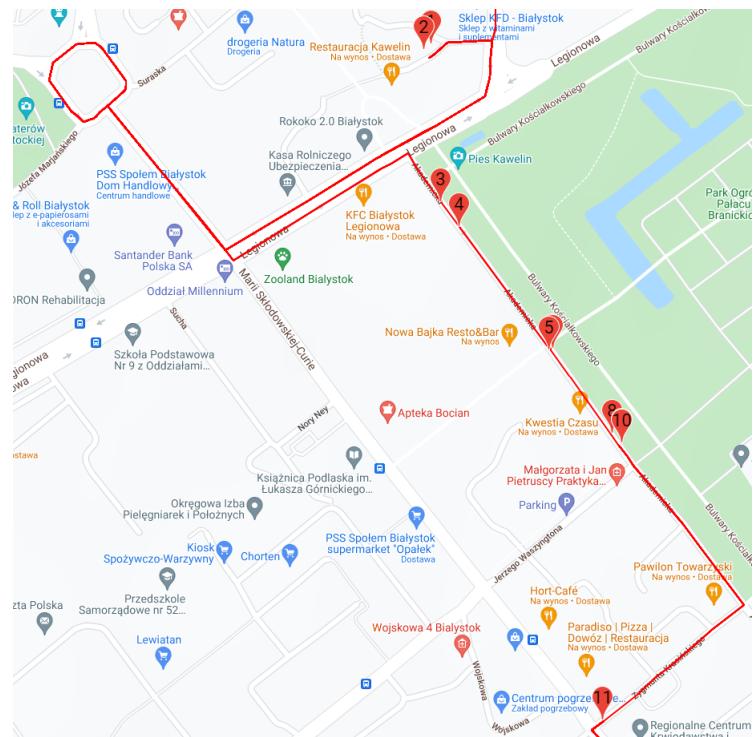
Rysunek 6.12: Fragment trasy algorytmu mrówkowego

samego wycinka trasy. Liczba znajdująca się w treści markera oznacza kolejność odwiedzania tych punktów przez algorytmy. Na rysunku 6.12 został pokazany fragment trasy wygenerowanej przez algorytm mrówkowy. Widać na pierwszy rzut oka, że jest to on początkiem trasy tego algorytmu, ponieważ są to początkowe numery. Porównując algorytm genetyczny 6.13 w tym samym miejscu, można zauważyć, że wierzchołki te są odwiedzane

dopiero w połowie trasy. Trasa dla algorytmu mrówkowego i algorytmu A* jest bardzo podobna. Jedyną różnicą na tym odcinku jest kolejność wybrania 3 punktu przez algorytm mrówkowy. W algorytmie A* punkt ten został odwiedzony jako 11.



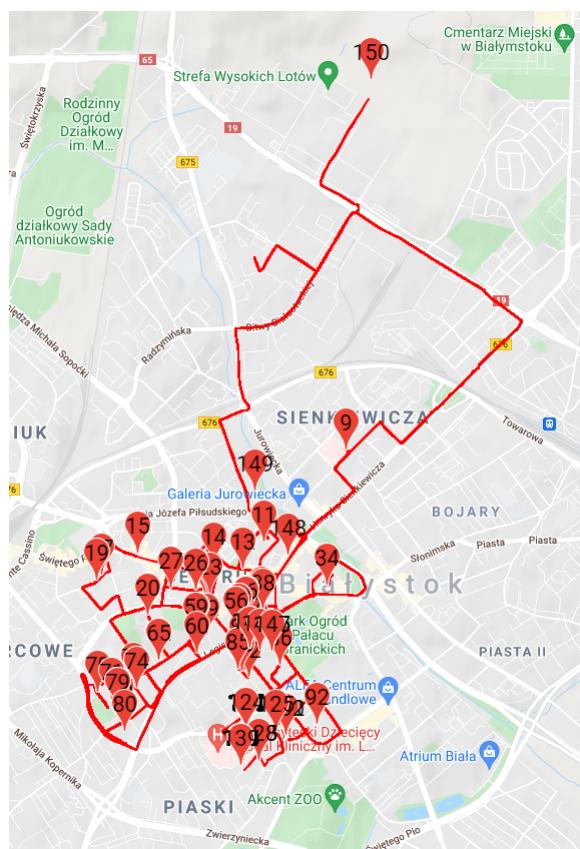
Rysunek 6.13: Fragment trasy algorytmu genetycznego



Rysunek 6.14: Fragment trasy algorytmu A*

6.2 Trasa II

Na rysunku 6.15 została przedstawiona oryginalna druga trasa ciężarówki, na której znajduje się 10 kontenerów. Trasa ta mimo większej ilości kontenerów, jest krótsza od poprzedniej. Jej długość wynosi 40402 metrów. Wszystkie punkty znajdują się na osiedlach w okolicy centrum. Ciężarówka często pokonywała kilka razy te same drogi jednokierunkowe. Po zebraniu wszystkich kontenerów, pojazd rozładował się na osiedlu wygoda.



Rysunek 6.15: Trasa II

6.2.1 Wyniki algorytmu genetycznego - Paweł

W tabeli 6.14 zostały przedstawione średnie wyniki algorytmu genetycznego dla każdej kombinacji parametrów. W przeciwieństwie do poprzedniej trasy, nie zawsze krzyżowanie *CX* z mutacją *Insert* optymalizowała najlepiej trasę. W populacji 2000 osobników oraz $k = 100$ najlepszy średni wynik osiągnęło krzyżowanie *CX* z mutacją *Revert*, a dla $k = 2000$ lub $k = 4000$ najlepszy wyniki były dla krzyżowania *CX* oraz mutacji *Swap*. W populacji $N = 100$, ani jedna kombinacji parametrów wejściowych, nie

osiągnęła średniego wyniku, który by optymalizował trasę. Dla populacji $N = 1000$ trasę udało się zoptymalizować do średnio 30385 metrów, jest to poprawa o 10017 metrów. Wynik ten osiągnęło krzyżowanie CX z mutacją $Insert$ w 1000 iteracji. Natomiast najlepszy wynik globalny 28196 metrów, osiągnięto dla populacji 2000 osobników oraz $k = 1000$, krzyżowanie CX z mutacją $Insert$. Porównując krzyżowania, znowu najlepsze okazało się krzyżowanie CX przed krzyżowaniem PMX . Natomiast w mutacjach zazwyczaj lepszy wynik był osiągany przez mutację $Insert$ z kilkoma wyjątkami.

Tabela 6.14: Trasa II - średnie wyniki algorytmu genetycznego dla poszczególnych parametrów wejściowych

N	k	PMX Insert	PMX Swap	PMX Revert	OX Insert	OX Swap	OX Revert	CX Insert	CX Swap	CX Revert
100	100	99575	107771	111797	113364	119707	120570	110318	116843	116749
100	1000	56105	70104	88449	60788	71531	93290	59452	70589	88734
100	2000	48407	61102	87698	50719	61516	92912	48634	61984	88546
100	4000	47154	54452	47154	48203	55866	92145	43488	52586	86276
1000	100	65248	67898	64703	78946	83318	81904	47854	46921	49695
1000	1000	39419	43935	55085	41294	46914	68463	30385	32616	43041
1000	2000	34882	40914	57467	39237	44605	65532	30631	31273	48346
1000	4000	36160	39848	55774	36414	44856	68215	30443	30720	48295
2000	100	58533	59876	60898	70291	71693	72095	38655	39129	37204
2000	1000	34939	38272	49381	35951	39779	62656	28196	29415	33519
2000	2000	32900	39068	48607	34474	41349	56907	28802	28506	34458
2000	4000	32626	36244	49132	36333	39164	57952	28808	28453	32683

W tabeli 6.15 została przedstawiona procentowa ilość wyników lepszych od długości trasy oryginalnej. Krzyżowanie PMX z mutacją $Insert$, osiągnęło czterokrotnie 100% skuteczności dla populacji 1000 oraz 2000 osobników, w obu dla $k = 2000$ lub $k = 4000$. Krzyżowanie OX z mutacją $Insert$ było bezbłędne dla $N = 2000$ oraz $k \geq 1000$. Krzyżowanie CX okazało się jedynym, które miało 100% skuteczności w połączeniu z innymi mutacjami niż $Insert$. Krzyżowanie OX jedynie w populacji 2000 osobników potrafiło osiągnąć 100% skuteczności. Mutacja $Revert$ nie poradziła sobie z krzyżowaniem OX , nie osiągnięto, ani jednego wyniku lepszego niż długość trasy oryginalnej. Natomiast krzyżowanie CX , aż w 14 różnych kombinacjach parametrów optymalizowało trasę za każdym razem.

W tabeli 6.16 pokazane zostały najlepsze wyniki jakie osiągnął algorytm genetyczny na trasie II dla poszczególnych parametrów wejściowych. W populacji $N = 100$ najlepszy wynik 37172 metrów, osiągnęło krzyżowanie CX z mutacją $Insert$. Krzyżowanie CX z każdą mutacją, osiągnęło najwięcej razy wynik lepszy od trasy oryginalnej. W populacji 1000 osobników najmniejszą odległość 27858 metrów, osiągnął algorytm przy parametrach:

Tabela 6.15: Trasa II - skuteczność algorytmu genetycznego dla poszczególnych parametrów wejściowych

N	k	PMX Insert	PMX Swap	PMX Revert	OX Insert	OX Swap	OX Revert	CX Insert	CX Swap	CX Revert
100	100	0%	0%	0%	0%	0%	0%	0%	0%	0%
100	1000	0%	0%	0%	0%	0%	0%	0%	0%	0%
100	2000	0%	0%	0%	0%	0%	0%	0%	0%	0%
100	4000	10%	0%	10%	0%	0%	0%	30%	0%	0%
1000	100	0%	0%	0%	0%	0%	0%	10%	10%	10%
1000	1000	60%	10%	0%	40%	0%	0%	100%	100%	30%
1000	2000	100%	40%	0%	60%	0%	0%	100%	100%	10%
1000	4000	100%	50%	0%	90%	20%	0%	100%	100%	30%
2000	100	0%	0%	0%	0%	0%	0%	90%	70%	70%
2000	1000	90%	60%	0%	100%	50%	0%	100%	100%	100%
2000	2000	100%	70%	10%	100%	40%	0%	100%	100%	80%
2000	4000	100%	90%	20%	100%	70%	0%	100%	100%	100%

$k = 2000$, krzyżowaniu CX oraz mutacji $Swap$. W populacji 2000 osobników zostało znalezione najlepsze globalne rozwiązanie 27381 metrów, dla mutacji CX oraz krzyżowania $Swap$. Najmniej wyników optymalizujących trasę znalazło krzyżowanie OX . Krzyżowanie PMX potrafiło dla każdej populacji przynajmniej raz znaleźć lepszą trasę niż oryginalną. Mutacja $Revert$ dwukrotnie znalazła najlepszy wynik, w populacji 1000 oraz 2000 osobników dla $k = 100$.

Tabela 6.16: Trasa II - najlepsze wyniki algorytmu genetycznego dla poszczególnych parametrów wejściowych

N	k	PMX	PMX	PMX	OX	OX	OX	CX	CX	CX
		Insert	Swap	Revert	Insert	Swap	Revert	Insert	Swap	Revert
100	100	89707	95824	100856	102788	110057	114491	104749	104765	101883
100	1000	44863	63868	76641	54433	59588	80783	53688	55723	75815
100	2000	42084	56000	79817	46908	54427	86609	43440	48834	80426
100	4000	40309	49246	40309	43816	49864	78281	37172	47158	78780
1000	100	53565	59523	59419	71181	75567	73827	40068	38220	33442
1000	1000	34352	38069	48381	34282	44087	62194	28947	29655	34089
1000	2000	32942	36491	48353	33013	40481	58856	28047	27858	35650
1000	4000	33021	33720	47438	29602	38640	57355	28442	28027	35271
2000	100	50007	53464	51193	61228	67171	59619	33775	35063	31852
2000	1000	29758	33420	43562	32178	33255	56070	27655	27910	29839
2000	2000	31390	35329	38151	30047	31229	48944	28178	27381	29839
2000	4000	30080	31724	38944	32360	35025	49517	28177	27925	29840

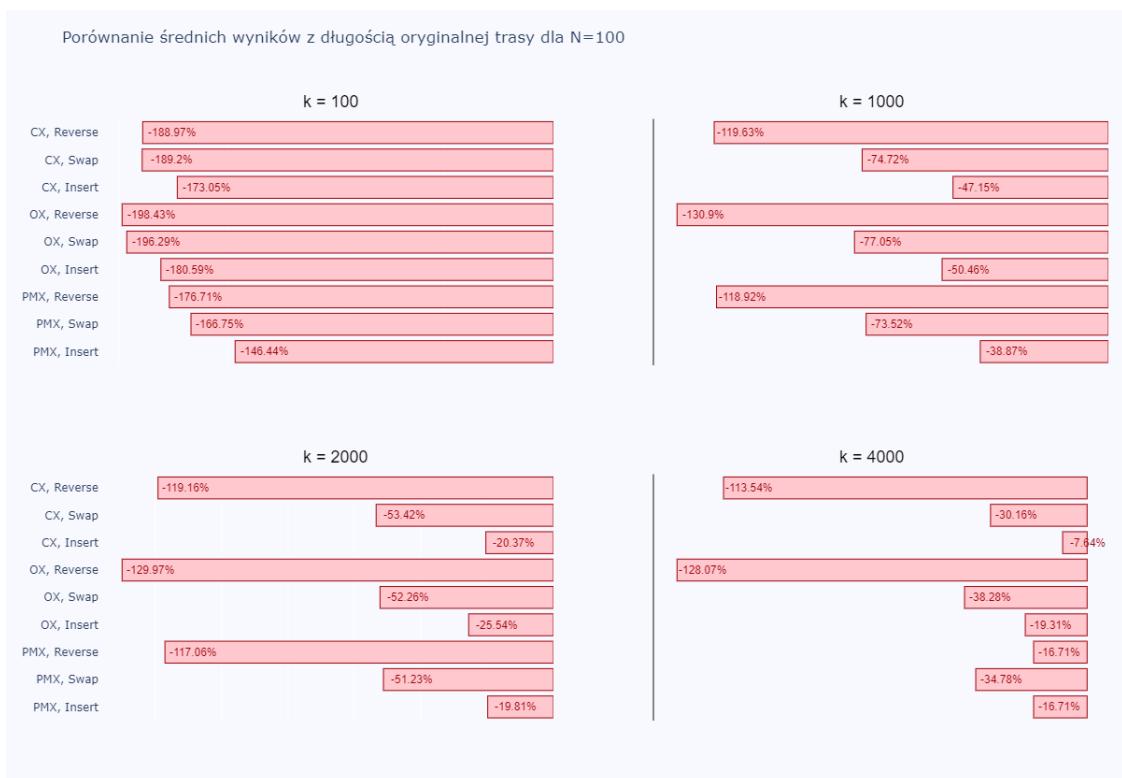
W tabeli 6.17 zostały pokazane średnie czasy trwania algorytmu genetycznego. Rozkład najlepszych średnich czasów jest bardzo podobny do trasy pierwszej. Najszybsze dla każdego N i k było krzyżowanie CX oraz mutacja $Revert$. Pomiędzy $N = 1000$, a $N=2000$ średnia najlepsza optymalizacja trasy poprawiła się o 477 metrów (Tab. 6.16), czas na znalezienie takiej trasy wzrósł ponad dwukrotnie z 4.17 sekund do 9.96 sekund. Najwolniej

sobie poradziło krzyżowanie *PMX*. Czasy wykonania wzrosły porównując je do trasy I (Tab. 6.4), spowodowane jest to tym, że chromosomy składają się ze 150 punktów, więc potrzeba wykonać więcej operacji przy krzyżowaniu oraz mutacji.

Tabela 6.17: Trasa II - średnie czasy wykonywania algorytmu dla poszczególnych parametrów wejściowych

N	k	PMX Insert	PMX Replace	PMX Revert	OX Insert	OX Replace	OX Revert	CX Insert	CX Replace	CX Revert
100	100	0,04	0,04	0,03	0,02	0,02	0,02	0,03	0,02	0,02
100	1000	0,27	0,26	0,24	0,20	0,19	0,18	0,24	0,16	0,15
100	2000	0,50	0,49	0,46	0,38	0,37	0,35	0,41	0,31	0,29
100	4000	0,95	0,95	0,95	0,75	0,72	0,69	0,74	0,60	0,59
1000	100	0,54	0,53	0,53	0,34	0,33	0,32	0,39	0,29	0,28
1000	1000	3,24	3,13	2,99	2,49	2,40	2,28	2,26	1,95	1,91
1000	2000	5,95	5,89	5,79	4,57	4,33	4,33	4,17	3,90	3,65
1000	4000	11,81	11,91	11,30	8,96	9,23	8,64	8,10	7,34	7,26
2000	100	1,17	1,19	1,12	0,76	0,75	0,74	0,84	0,66	0,64
2000	1000	7,17	7,55	6,98	5,56	5,51	5,23	5,11	4,56	4,45
2000	2000	13,61	13,61	13,48	10,71	11,13	10,64	9,96	9,03	8,93
2000	4000	27,25	27,60	27,11	21,06	21,97	21,21	18,63	17,79	17,43

Rysunki 6.16, 6.17 oraz 6.18 przedstawiają wykresy, które porównują o ile różni się rozwiązanie od długości trasy oryginalnej, kolejno dla $N = 100$, $N = 1000$ oraz $N = 2000$.

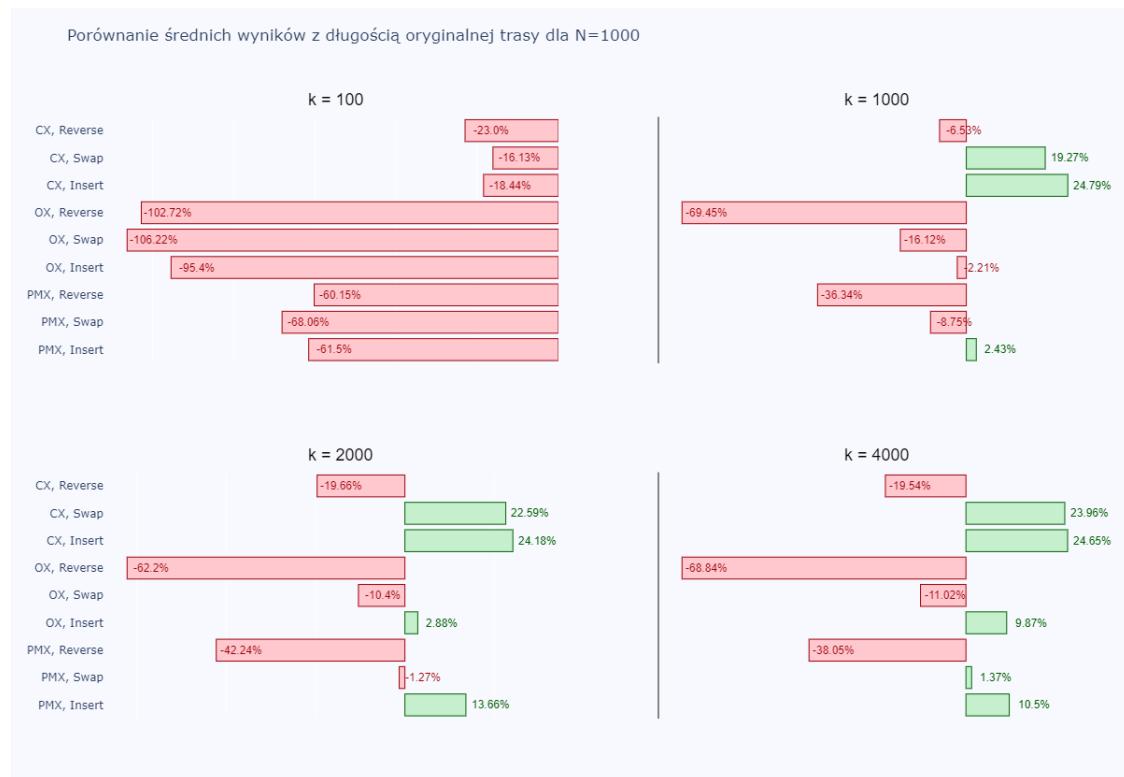


Rysunek 6.16: Trasa II - porównanie średnich wyników z długością oryginalnej trasy dla $N=100$

Dla $N = 100$ (Rys. 6.16) wszystkie kombinacje parametrów wejściowych nie potrafiły zoptymalizować rozwiązań. Dla $k = 100$ wygenerowane wyniki mają gorszą wartość od -195% do -150%. Przy $k = 4000$ wartości dla niektórych parametrów są średnio o gorsze o kilkanaście procent od trasy oryginalnej.

Dla $N = 1000$ (Rys. 6.17), nie udało się jedynie zoptymalizować tras dla $k = 100$, wszystkie wyniki są ujemne. Dla pozostałych wartości k , krzyżowanie CX z mutacjami *Insert* oraz *Swap*, średnio zoptymalizowało trasę od 19.27% do 24.65%. Krzyżowanie PMX dla tej populacji zoptymalizowało trasę w zależności od wartości k od 2.43% do 13.66%, a krzyżowanie OX od 2.88% do 9.87%.

Dla $N = 2000$ (Rys. 6.18) krzyżowanie CX zoptymalizowało trasę dla każdego k . Wynik dla mutacji *Swap* oraz *Insert* dla $k = 1000$ oraz $k = 2000$, waha się od 27.2% do 30.21%. Krzyżowanie PMX poprawiło wynik dla mutacji *Insert* do przedziału 13.52% - 19.25% w zależności od wartości parametru k . Natomiast krzyżowanie OX dla tej samej mutacji poprawiło się do przedziału 10.07%-14.67%. Inne mutacje dla tych krzyżowań wygenerowały słabsze wyniki.

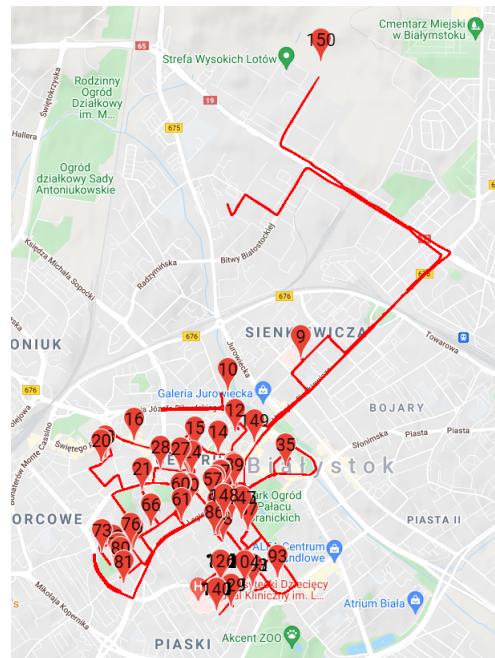


Rysunek 6.17: Trasa II - porównanie średnich wyników z długością oryginalnej trasy dla $N=1000$

Na rysunku 6.19 przedstawiono najlepszą trasę jaką znalazł algorytm genetyczny.



Rysunek 6.18: Trasa II - porównanie średnich wyników z długością oryginalnej trasy dla N=2000



Rysunek 6.19: Trasa II - algorytm genetyczny

6.2.2 Wyniki algorytmu mrówkowego - Kamil

Podobnie tak jak w przypadku trasy numer 1, algorytm został uruchomiony dziesięciokrotnie dla każdej możliwej wariacji parametrów wejściowych. Wynikiem jaki jest

generowany przez algorytm są punkty w których znajdują się kontenery z odpadami, przebyty dystans, a także czas jaki trzeba było czekać na wygenerowanie wyniku. Tabele z wyliczonymi wynikami również zostały przedstawione według takiego samego schematu. Wiersze tabeli odpowiadają ilościom przeprowadzonych iteracji, a w kolumnach zawarte są informacje o współczynniku parowania.

Najlepszy średni wynik został przedstawiony w tabeli 6.18 i wynosi 26447. Jest to wynik który został uzyskany dla współczynnika parowania równego 0.2 i liczbie iteracji równej 50. Wyniki dla takiego samego współczynnika ale innej liczby iteracji są na zbliżonym poziomie. Podobnie jak w przypadku poprzedniej trasy, najgorsza średnia została obliczona dla współczynnika parowania równego 0.1. Wraz ze wzrostem liczby iteracji następował progres. Ostatecznie dla 100 iteracji długość trasy jest zbliżona do najlepszych wyników. Ścieżka z największą średnią jest lepsza od oryginalnej i stanowi 78% jej długości.

Tabela 6.18: Trasa II - średnie wyniki algorytmu mrówkowego dla poszczególnych parametrów wejściowych

k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
50	35518	26447	27146	26449	28817	28061	28672	30477	29670
75	26937	26812	27660	26896	28351	28242	29070	29968	28910
100	26721	26532	26888	27358	28191	28116	29397	29738	28608

Algorytm mrówkowy bardzo dobrze poradził sobie z optymalizacją trasy numer 2. W tabeli numer 6.19 przedstawione zostały najlepsze wyniki algorytmu mrówkowego dla tej trasy. Każdy z wyników jest lepszy od oryginalnej trasy. Prawie wszystkie wartości są niższe od 30000 metrów. Dla drugiej trasy najkrótsza odległość jaką udało się uzyskać wynosi 24347 metrów.

Tabela 6.19: Trasa II - najlepsze wyniki algorytmu mrówkowego dla poszczególnych parametrów wejściowych

k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
50	31995	25023	25209	25119	26103	25560	25799	28107	26360
75	26037	25686	25330	24347	26291	25999	25912	27805	25776
100	25820	25343	24760	24404	26080	25136	25695	27485	26650

Dzięki tabeli 6.20 można zauważyć jakie optymalizacje wykonał algorytm mrówkowy. Wartością każdej komórki tabeli jest 100%. Świadczy to o tym, że każda z tras jaka została wygenerowana przez algorytm mrówkowy jest lepsza od oryginalnej. W tej sytuacji bez znaczenia jest liczba iteracji czy też współczynnik parowania.

Tabela 6.20: Trasa II - skuteczność algorytmu mrówkowego dla poszczególnych parametrów wejściowych

k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
50	100%	100%	100%	100%	100%	100%	100%	100%	100%
75	100%	100%	100%	100%	100%	100%	100%	100%	100%
100	100%	100%	100%	100%	100%	100%	100%	100%	100%

Dla najmniejszej ilości iteracji, zostają wygenerowane trasy krótsze od pierwotnej.

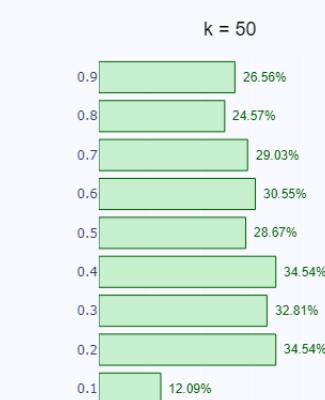
Na wykresie 6.20 zostały przedstawione średnie wyniki dla 50 iteracji. Można na tym wykresie zauważyć, że najdłuższe trasy były generowane dla najniższego współczynnika parowania. Najkrótsze trasy były generowane w przedziale od 0.2 do 0.4. Dla pozostałych współczynników wartości są większe od 24%.

Oprócz krótkich tras uwagę należy zwrócić na czas jaki program potrzebował na ich optymalizację. Średni czas oczekiwania dla najmniejszej wartości iteracji wynosił 28.325 sekund. Pomiar czasu trwania algorytmu przy 75 iteracjach trwał 42.689. Najdłużej program wykonywał się dla 100 iteracji. Czas ten został oszacowany na 60.281.

Tabela 6.21: Trasa II - średnie wyniki czasu wykonania algorytmu mrówkowego dla parametru ilości iteracji

iteracje	50	75	100
czas	28.325	42.689	60.281

Porównanie średnich wyników z długością oryginalnej trasy.



Rysunek 6.20: Procentowy rozkład średnich wartości dla 50 iteracji

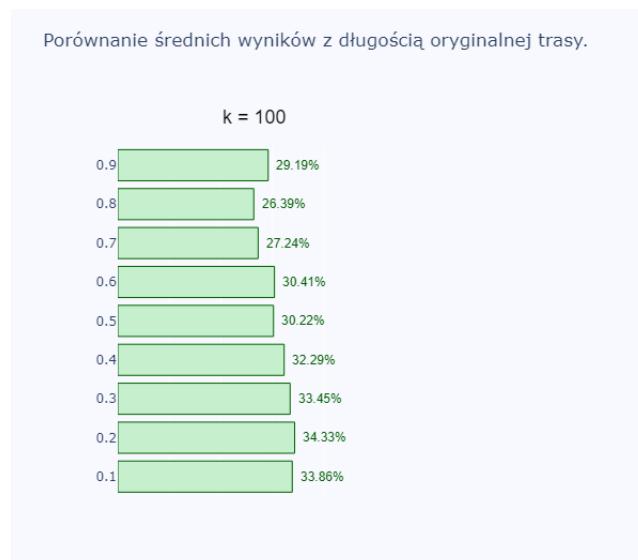
Po zwiększeniu liczby iteracji do 75, na wykresie 6.21 można zauważyć zmiany w średnich dystansach ścieżek. Na szczególną uwagę zasługuję współczynnik o wartości 0.1.

Jeśli liczba iteracji była zmniejszona o 25, to średnie wyniki były najgorsze. Po zwiększeniu iteracji o 25, sytuacja odwróciła się o 180 stopni. Średnia z tras dla współczynnika parowania feromonów równego 0.1 była jedną z najlepszych w całym zestawieniu. Pozostałe wyniki są co najmniej 25% lepsze od oryginalnego przebytego dystansu.



Rysunek 6.21: Procentowy rozkład średnich wartości dla 75 iteracji

Maksymalna liczba iteracji w niewielkim stopniu polepszyła wcześniejsze wyniki. Ponownie najkrótsze trasy zostały wygenerowane dla współczynników parowania z dolnego zakresu. Nie zmienia to faktu, iż wszystkie wyniki w dalszym ciągu były lepsze od badanej oryginalnej trasy.



Rysunek 6.22: Procentowy rozkład średnich wartości dla 100 iteracji

Na rysunku 6.23 przedstawiono najlepszą trasę jaką znalazł algorytm mrówkowy.



Rysunek 6.23: Trasa II - algorytm mrówkowy

6.2.3 Wyniki algorytmów zachłannych - Przemysław

W tabeli 6.22 zostały umieszczone najlepsze wartości wygenerowanych tras przez algorytmy zachłanne dla trasy nr 2 przy zastosowaniu współczynnika błędu uruchamianego na poszczególnych etapach budowania całkowitej trasy. Tak jak w poprzedniej trasie algorytm A* uzyskał najlepszy wynik równy 28757 metrów. Algorytm najbliższego sąsiada również doszedł do wyniku poniżej 30000 metrów, podczas etapu drugiego znalazł trasę o długości 29 800 metrów. Znów najgorszy okazał się algorytm najmniejszej krawędzi, który tym razem wyznaczył trasę 31745 metrów. Najwięcej najgorszych wyników zwrócił również algorytm najmniejszej krawędzi, gdzie inne algorytmy oddały wyniki bardzo przybliżone.

Tabela 6.22: Trasa II - wyniki algorytmów zachłannych

Błąd (m)	ANS-1	ANS-2	ANS-3	ANK-1	ANK-2	ANK-3	A*-1	A*-2	A*-3
0	30845	31765	31663	33416	32298	33944	30006	30483	28773
5	30222	31765	31986	32780	33267	32328	30066	29840	29083
10	30845	31123	31986	33416	33899	33944	30325	29813	28757
20	30845	31105	31340	33332	33892	32328	29687	30454	28757
30	30845	30798	31340	31745	33246	33944	30325	29813	29375
40	30845	30798	31663	32535	33246	33944	30006	30134	29066
50	30222	29800	31479	33053	33246	32328	30325	30454	29066
75	30845	30104	31800	32423	33246	33298	29687	29519	28757
100	30845	30104	31800	32423	32278	33298	30250	29519	29066

Tabela 6.23 przedstawia procentową poprawę oryginalnej trasy nr 2. Wszystkie wyznaczone trasy przez poszczególny z trzech algorytmów uzyskały lepszy wynik od trasy oryginalnej. Najlepszy wynik otrzymany przez algorytm A* poprawił trasę aż o 28.82%. Na pierwszy rzut oka można zauważyć, że najwyższe optymalizacje zostały wygenerowane znów przez algorytm A*. Najgorsze optymalizacje otrzymano przy zastosowaniu algorytmu najmniejszej krawędzi w trzeciej fazie budowania tras, gdzie wszystkie wyniki były poniżej 20%, a dolny wynik to tylko 15.98%. Algorytm najbliższego sąsiada wypadł tym razem umiarkowanie, dając najlepszy wynik równy 26.24%, który jest lepszy od niektórych tras wyznaczanych przez algorytm A*.

Tabela 6.23: Trasa II - optymalizacja wyników względem oryginalnej trasy

Błąd (m)	ANS-1	ANS-2	ANS-3	ANK-1	ANK-2	ANK-3	A*-1	A*-2	A*-3
0	23,65%	21,38%	21,63%	17,29%	20,06%	15,98%	25,73%	24,55%	28,78%
5	25,20%	21,38%	20,83%	18,87%	17,66%	19,98%	25,58%	26,14%	28,02%
10	23,65%	22,97%	20,83%	17,29%	16,10%	15,98%	24,94%	26,21%	28,82%
20	23,65%	23,01%	22,43%	17,50%	16,11%	19,98%	26,52%	24,62%	28,82%
30	23,65%	23,77%	22,43%	21,43%	17,71%	15,98%	24,94%	26,21%	27,29%
40	23,65%	23,77%	21,63%	19,47%	17,71%	15,98%	25,73%	25,41%	28,06%
50	25,20%	26,24%	22,09%	18,19%	17,71%	19,98%	24,94%	24,62%	28,06%
75	23,65%	25,49%	21,29%	19,75%	17,71%	17,58%	26,52%	26,94%	28,82%
100	23,65%	25,49%	21,29%	19,75%	20,11%	17,58%	25,13%	26,94%	28,06%

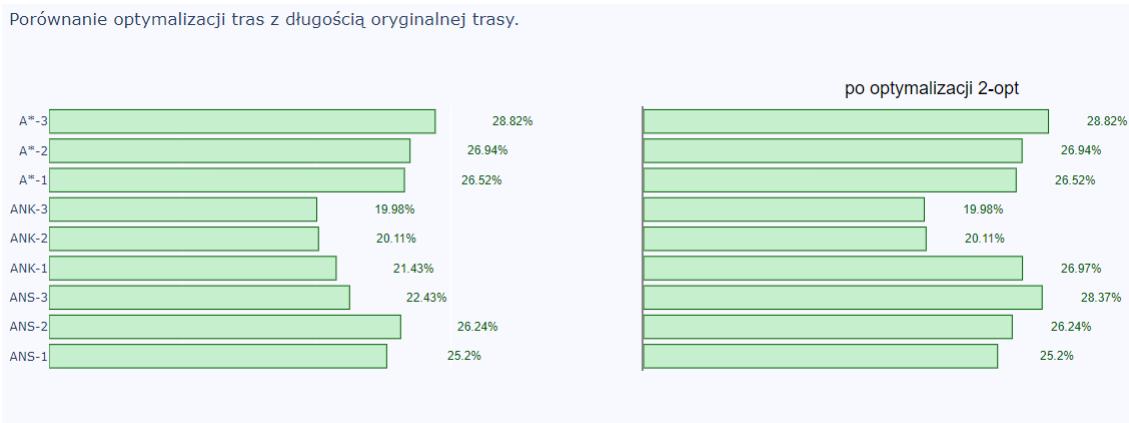
Tabela 6.24 przedstawia zoptymalizowane trasy metodą 2-opt. Optymalizacje nie przyniosły nowego najlepszego wyniku jednak poprawiły większość wcześniej budowanych tras. Tabela 6.25 przedstawia procentowe wyniki optymalizacji względem oryginalnej trasy, na podstawie której powstał rysunek 6.24. Dzięki optymalizacji 2-opt dla algorytmu najbliższego sąsiada otrzymano nową najlepszą trasę o 28.37% lepszą od trasy oryginalnej. Dla algorytmu najmniejszej krawędzi górnego wynik został poprawiony z 21.43% na 26.97%. Optymalizacja 2-opt dla algorytmu A* nie znalazła lepszego wyniku.

Tabela 6.24: Trasa II - zastosowanie metody 2-opt na powstałych trasach

Błąd (m)	ANS-1	ANS-2	ANS-3	ANK-1	ANK-2	ANK-3	A*-1	A*-2	A*-3
0	30222	30798	28938	32535	32298	33298	29687	30483	28773
5	30222	30798	31479	32780	33267	32328	30066	29840	29083
10	30222	30798	31479	32535	33899	33298	30325	29519	28757
20	30222	30798	28938	33332	33892	32328	29687	30454	28757
30	30222	30798	28938	29504	32298	33298	29687	29519	29066
40	30222	30798	28938	32535	32298	33298	29687	30134	29066
50	30222	29800	31479	33053	32298	32328	30325	30454	29066
75	30222	29800	31479	29504	32298	33298	29687	29519	28757
100	30222	29800	31479	29504	32278	33298	30250	29519	29066

Tabela 6.25: Trasa II - optymalizacja powstałych tras przez metodę 2-opt

Błąd (m)	ANS-1	ANS-2	ANS-3	ANK-1	ANK-2	ANK-3	A*-1	A*-2	A*-3
0	25,20%	23,77%	28,37%	19,47%	20,06%	17,58%	26,52%	24,55%	28,78%
5	25,20%	23,77%	22,09%	18,87%	17,66%	19,98%	25,58%	26,14%	28,02%
10	25,20%	23,77%	22,09%	19,47%	16,10%	17,58%	24,94%	26,94%	28,82%
20	25,20%	23,77%	28,37%	17,50%	16,11%	19,98%	26,52%	24,62%	28,82%
30	25,20%	23,77%	28,37%	26,97%	20,06%	17,58%	26,52%	26,94%	28,06%
40	25,20%	23,77%	28,37%	19,47%	20,06%	17,58%	26,52%	25,41%	28,06%
50	25,20%	26,24%	22,09%	18,19%	20,06%	19,98%	24,94%	24,62%	28,06%
75	25,20%	26,24%	22,09%	26,97%	20,06%	17,58%	26,52%	26,94%	28,82%
100	25,20%	26,24%	22,09%	26,97%	20,11%	17,58%	25,13%	26,94%	28,06%



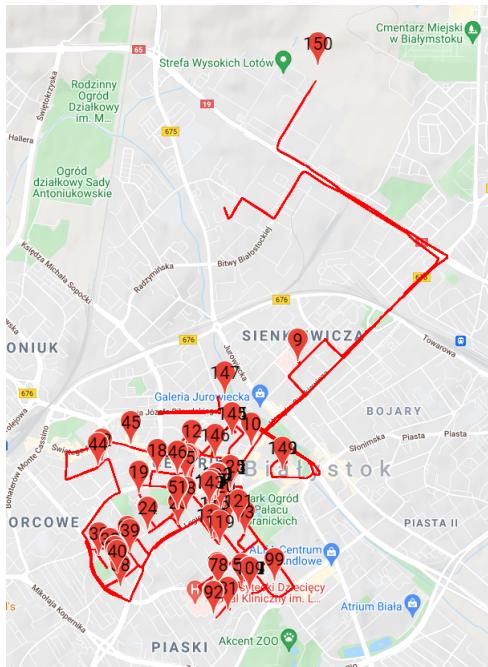
Rysunek 6.24: Trasa II - porównanie optymalizacji algorytmów z zastosowanie metody 2-opt

W tabeli 6.26 znajdują się czasy działania algorytmów. Najlepszą trasę udało się uzyskać w czasie 8.05 sekundy. W najgorszym wypadku algorytm wykonywał się ponad 70 sekund.

Tabela 6.26: Trasa II - czasy wykonywania algorytmów

Błąd (m)	ANS-1	ANS-2	ANS-3	ANK-1	ANK-2	ANK-3	A*-1	A*-2	A*-3
0	8,62	1,18	70,32	0,36	0,69	1,43	1,36	0,42	1,33
5	8,45	1,2	70,23	0,14	0,34	1,17	0,87	0,86	1,02
10	16,79	1,17	70,83	0,17	14,66	1,19	0,88	0,9	8,05
20	18,21	56,24	71	14,26	20,29	1,18	0,17	0,94	8,1
30	18,59	19,32	70,64	15,13	20,94	1,18	0,97	0,92	12,01
40	20,18	18,93	70,52	29,12	27,52	1,19	1,64	0,92	11,91
50	27,71	19,6	32,93	38,78	2,3	1,18	1,67	0,99	12,1
75	28,7	30,26	33,98	38,49	2,42	1,19	0,33	14,6	11,87
100	29,23	30,53	33,48	38,93	2,64	1,22	20,59	22,06	12,09

Na rysunku 6.25 przedstawiono najlepszą trasę jaką znalazły algorytmy zachłanne.



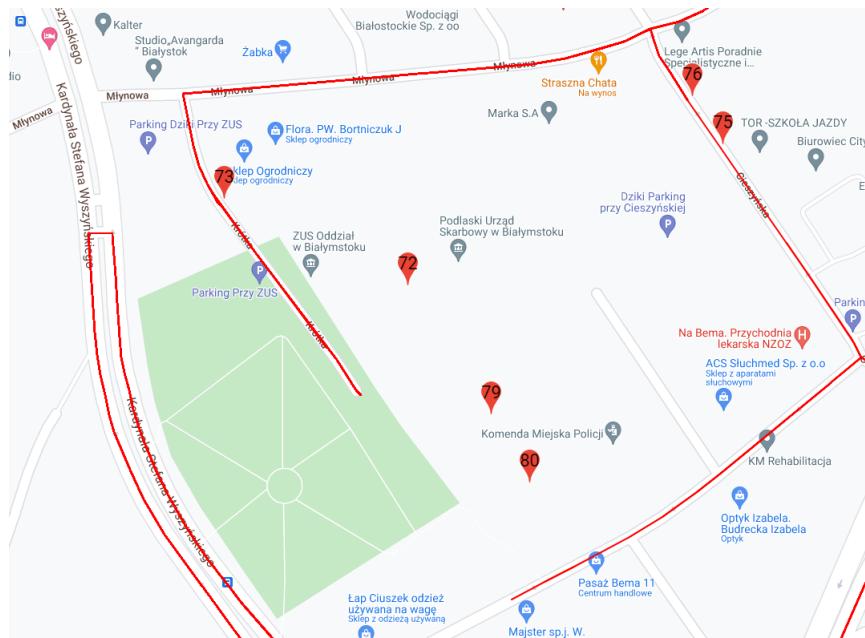
Rysunek 6.25: Trasa II - algorytm zachłanny

6.2.4 Porównanie algorytmów - wspólnie

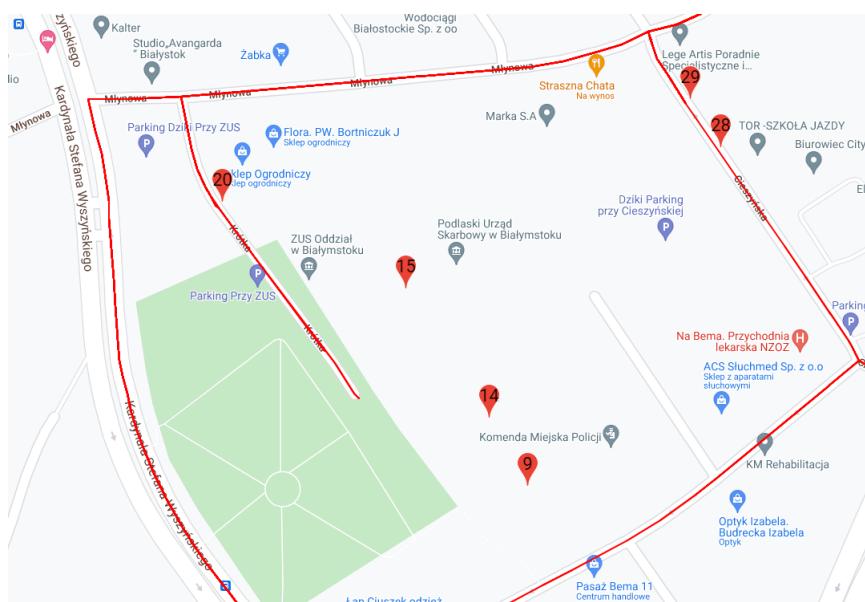
Podobnie jak w przypadku trasy numer 1, najlepsze wyniki algorytmów nie są takie same. Ponownie przeprowadzone optymalizacje przyniosły spodziewany efekt. Najlepszym wynikiem algorytmu genetycznego jest 27381. W stosunku do oryginalnej trasy jest to poprawa o 40%. Czas z jakim pracował program aby wygenerować taki wynik to 9.03 sekund. Ponownie najlepszy wynik został osiągnięty przez algorytm mrówkowy. Łączy przebyty dystans dla tego rozwiązania to 24347. Jest to o 46% wynik lepszy od pierwotnej trasy. Czas oczekiwania na ten rezultat wynosił 28.32 sekund. Natomiast algorytmy zachłanne, a dokładnie algorytm A*, znalazły trasę o długości 32 855 metrów, a przy zastosowaniu metody 2-opt wynik ten uległ poprawie do 28 757 metrów. Do wykonania wszystkich operacji potrzebne było 20.59 sekundy.

Najgorszy wynik został wygenerowany przez algorytm A*. Czas działania tego algorytmu również nie wyróżnia się spośród pozostałych rozwiązań. Lepszym wynik został wygenerowany przy użyciu algorytmu genetycznego. Wynik ten został osiągnięty w najkrótszym czasie spośród wszystkich trzech rozwiązań. Najlepszy wynik ponownie został wygenerowany przez algorytm mrówkowy. Czas wykonania programu również i w tym przypadku był najdłuższy.

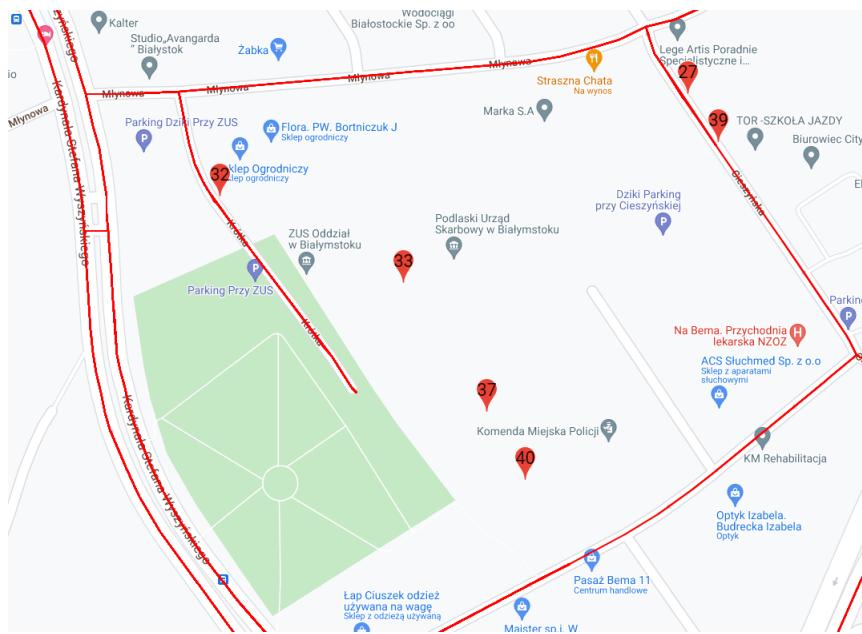
Na zdjęciach 6.26, 6.27, 6.28 zostały pokazane przykładowe różnice w wygenerowanych trasach przez algorytmy. Na przedstawionych fragmentach można zauważyc, że punkty były odwiedzane w różnych kolejnościach. Dodatkowo na uwagę zasługuje fakt, iż w każdym z rozwiązań pojazd w inny sposób pokonał ulicę Wyszyńskiego.



Rysunek 6.26: Fragment trasy algorytmu genetycznego



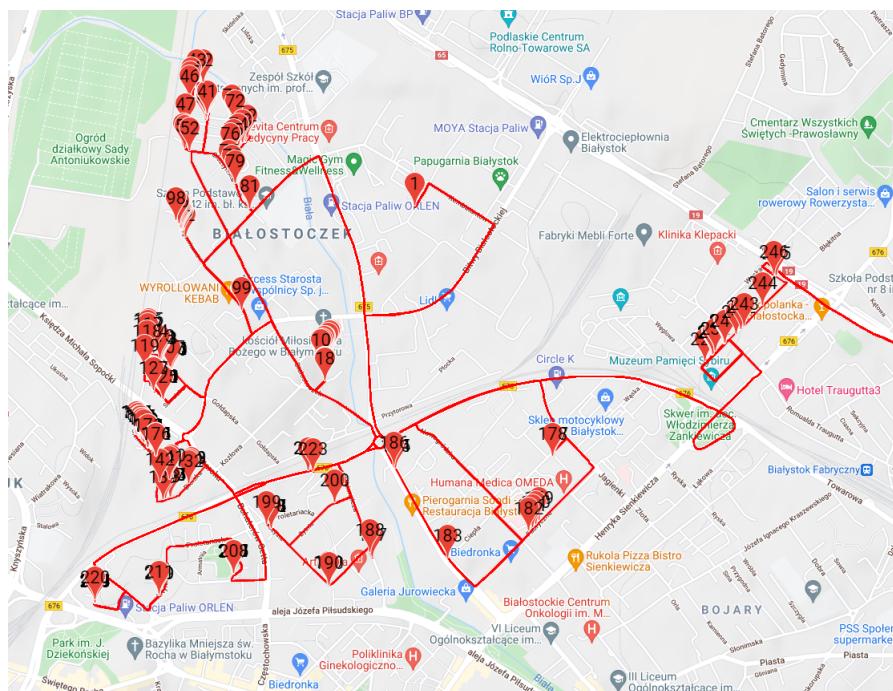
Rysunek 6.27: Fragment trasy algorytmu mrówkowego



Rysunek 6.28: Fragment trasy algorytmu A*

6.3 Trasa III

Na rysunku 6.29 została przedstawiona trasa III, która składa się z 248 punktów i ma długość 53478 metrów. Jest to zarazem najdłuższa jak i również posiada najwięcej kontenerów. Odwiedzane punkty znajdują się na osiedlach białostoczek oraz wygoda. Ciężarówka rozładowuje się w miejscowości Hryniowicze.



Rysunek 6.29: Trasa III

6.3.1 Wyniki algorytmu genetycznego - Paweł

W tabeli 6.27 zostały pokazane średnie wyniki algorytmu genetycznego dla trasy III. Krzyżowanie *OX* ani razu nie osiągnęło średniego wyniku lepszego od długości trasy oryginalnej. W populacji 100 osobników większość otrzymanych wyników ma wartość powyżej 100 kilometrów. Dopiero dla $N = 1000$, $k = 1000$, krzyżowania *CX* oraz mutacji *Insert* udało się osiągnąć średni lepszy wynik od wartości oryginalnej 43728 metrów, jest to poprawa długości o 9750 metrów. Najlepszy wynik dla $N = 1000$ wystąpił dla $k = 4000$, krzyżowania *CX* oraz mutacji *Insert*, wyniósł 40121 metrów. Dla tej populacji krzyżowanie *PMX* w jednym miejscu osiągnęło średni wynik lepszy niż długość trasy oryginalnej, dla $k = 4000$ oraz mutacji *Insert*. W populacji 2000 osobników wyniki rozłożyły się podobnie jak do poprzedniej. Najlepszy globalny wynik 34168 metrów został osiągnięty dla krzyżowania *CX*, mutacji *Insert* z $k = 4000$. Mutacja *Revert* nie osiągnęła ani jednego średniego wyniku optymalizującego trasę. Mutacja *Swap* osiągnęła takie wyniki jedynie dla krzyżowania *CX*.

Tabela 6.27: Trasa III - średnie wyniki algorytmu genetycznego dla danych parametrów wejściowych

N	k	PMX	PMX	PMX	OX	OX	OX	CX	CX	CX
		Insert	Swap	Revert	Insert	Swap	Revert	Insert	Swap	Revert
100	100	202200	215800	223134	226673	238350	244471	226996	238909	248130
100	1000	101445	119726	156991	108762	130635	168963	121076	140420	170028
100	2000	88501	112378	149298	95249	112897	159348	95405	110801	162912
100	4000	81056	97882	151635	80120	99671	163374	74624	97972	153400
1000	100	138988	141411	138102	159437	165202	161127	100709	114035	108668
1000	1000	60548	66745	93017	70666	76984	117566	43728	54573	85516
1000	2000	53736	68402	96311	59332	71183	117932	42545	50305	90144
1000	4000	49323	66474	96935	56877	69448	118314	40121	48213	86232
2000	100	126626	127066	125408	147299	148430	147276	64776	89803	81523
2000	1000	54123	64199	82598	57743	68314	103521	34365	39530	50624
2000	2000	45893	55922	82523	54700	63697	96266	36518	39200	59014
2000	4000	47206	57436	81653	54149	66321	96440	34168	37951	56645

W tabeli 6.28 została pokazana skuteczność algorytmu genetycznego. W populacji 100 osobników nie było ani jednego wyniku, lepszego od długości trasy oryginalnej. Podobnie dla $N = 1000$ i $k = 100$. Krzyżowanie *PMX* osiągnęło tylko raz skuteczność 100 procent dla $N = 2000$, $k = 2000$ oraz mutacji *Insert*. Mutacja *Revert* ani razu dla krzyżowania *PMX* oraz *OX* nie zoptymalizowała wyniku. *Revert* dopiero dla krzyżowania *CX* i $N = 2000$ miał skuteczność na poziomie 40% - 60%. Krzyżowanie *CX* z mutacją *Insert* uzyskało

100% skuteczności dla $N = 1000$ oraz $k = 1000$ i $k = 4000$. Mutacja *Swap* razem z *Insert* osiągnęła 100% skuteczności dla krzyżowania *CX* dla $N = 2000$ oraz $k \geq 1000$.

Tabela 6.28: Trasa III - skuteczność algorytmu genetycznego dla poszczególnych parametrów wejściowych

N	k	PMX Insert	PMX Swap	PMX Revert	OX Insert	OX Swap	OX Revert	CX Insert	CX Swap	CX Revert
100	100	0%	0%	0%	0%	0%	0%	0%	0%	0%
100	1000	0%	0%	0%	0%	0%	0%	0%	0%	0%
100	2000	0%	0%	0%	0%	0%	0%	0%	0%	0%
100	4000	0%	0%	0%	0%	0%	0%	0%	0%	0%
1000	100	0%	0%	0%	0%	0%	0%	0%	0%	0%
1000	1000	0%	0%	0%	0%	0%	0%	100%	40%	0%
1000	2000	50%	0%	0%	10%	0%	0%	90%	70%	0%
1000	4000	90%	0%	0%	30%	0%	0%	100%	70%	0%
2000	100	0%	0%	0%	0%	0%	0%	0%	0%	0%
2000	1000	40%	10%	0%	10%	0%	0%	100%	100%	60%
2000	2000	100%	30%	0%	80%	10%	0%	100%	100%	40%
2000	4000	90%	30%	0%	50%	0%	0%	100%	100%	30%

W tabeli 6.29 przedstawiono najlepsze wyniki jakie osiągnął algorytm genetyczny dla poszczególnych parametrów wejściowych. Algorytm genetyczny znalazł wyniki lepsze od długości trasy oryginalnej od $N = 1000$ i $k = 1000$. Najlepszy osiągnięty wynik dla $N = 1000$ to 34365 metrów, czyli optymalizacja o 19113 metrów. Dla tej populacji zdecydowanie krzyżowanie *OX* radzi sobie dużo lepiej niż inne. Algorytm genetyczny z krzyżowaniem *PMX* dla tej populacji osiągnął najlepszy wynik 43708 metrów. Najlepszy globalny wynik został osiągnięty dla $N = 2000$, $k = 1000$, krzyżowania *CX* oraz mutacji *Insert* i wyniósł 32372 metrów. Krzyżowanie *PMX* poprawiło się nieznacznie do 32326 metrów. Mimo, że mutacja *Revert* ani razu nie miała wyniku średniego, który by optymalizował trasę, to dla krzyżowania *CX* osiągnęła pojedyncze wyniki lepsze od trasy oryginalnej.

W tabeli 6.30 zostały pokazane średnie czasy trwania algorytmu genetycznego. Podobnie jak w poprzednich przypadkach, najszybsze okazało się krzyżowanie *CX*, a naj wolniejsze *PMX*. Pomiędzy $N = 1000$, a $N = 2000$ średnia najlepsza optymalizacja trasy poprawiła się około 5953 metrów (Tab. 6.27), czas na znalezienie takiej trasy wzrósł ponad dwukrotnie z 12.2 sekund do 26.81 sekund. Trasa III składa się z największej liczby punktów, więc z tego powodu czas trwania algorytmu jest dłuższy niż dla poprzednich tras.

Rysunki 6.30, 6.31 oraz 6.32 przedstawiają wykresy, które porównują o ile różni się rozwiązanie od długości trasy oryginalnej, kolejno dla $N = 100$, $N = 1000$ oraz $N = 2000$.

Tabela 6.29: Trasa III - najlepsze wyniki algorytmu genetycznego dla poszczególnych parametrów wejściowych

N	k	PMX Insert	PMX Swap	PMX Revert	OX Insert	OX Swap	OX Revert	CX Insert	CX Swap	CX Revert
100	100	178235	179169	210255	214392	220255	225128	199164	209900	228851
100	1000	92035	97541	145836	92447	119218	156940	108633	134528	147851
100	2000	75472	99019	133028	82103	100138	150643	82086	100030	146323
100	4000	66386	84653	128228	67843	87629	150569	64530	85814	128407
1000	100	130408	129829	113771	146677	146199	146851	80069	80657	91467
1000	1000	54442	57812	79987	65279	70631	98505	37536	44330	70152
1000	2000	43708	57296	67415	51213	57795	97761	36381	39559	64270
1000	4000	43938	57678	81284	47695	60624	102346	34365	38618	69499
2000	100	117579	114351	112117	133323	133203	140060	58239	66450	64827
2000	1000	46490	47507	72252	50359	60606	91474	32374	33408	41138
2000	2000	42326	44568	70413	45282	53114	81577	33070	36492	41724
2000	4000	43332	49969	69794	46970	56971	83286	32478	33962	41975

Tabela 6.30: Trasa III - średnie czasy wykonywania algorytmu dla poszczególnych parametrów wejściowych

N	k	PMX	PMX	PMX	OX	OX	OX	CX	CX	CX
		Insert	Swap	Revert	Insert	Swap	Revert	Insert	Swap	Revert
100	100	0,09	0,07	0,06	0,04	0,04	0,03	0,07	0,03	0,03
100	1000	0,44	0,43	0,35	0,32	0,31	0,26	0,48	0,25	0,22
100	2000	0,83	0,79	0,69	0,60	0,58	0,52	0,78	0,46	0,44
100	4000	1,51	1,48	1,35	1,12	1,11	1,03	1,28	0,89	0,84
1000	100	1,06	1,06	1,06	0,61	0,60	0,57	0,76	0,52	0,52
1000	1000	5,47	5,40	4,64	3,85	3,81	3,30	3,85	2,87	2,79
1000	2000	9,48	9,29	8,36	6,98	6,95	6,23	7,12	5,52	5,49
1000	4000	17,64	17,06	15,82	13,01	13,03	12,11	12,22	10,65	10,56
2000	100	2,61	2,53	2,46	1,49	1,52	1,44	1,89	1,30	1,32
2000	1000	13,40	13,30	11,46	9,34	9,24	7,90	8,25	7,07	6,93
2000	2000	21,82	21,51	20,35	16,03	16,04	14,76	14,59	13,04	12,60
2000	4000	38,35	38,67	37,01	30,41	30,20	28,53	26,81	25,30	24,47

W populacji 100 osobników (Rys. 6.30) wyniki nawet nie zbliżyły się do trasy oryginalnej. Dla małej ilości iteracji, wyniki były gorsze o ponad 300%. Przy $kk = 4000$ zbliżyły się do -40%.

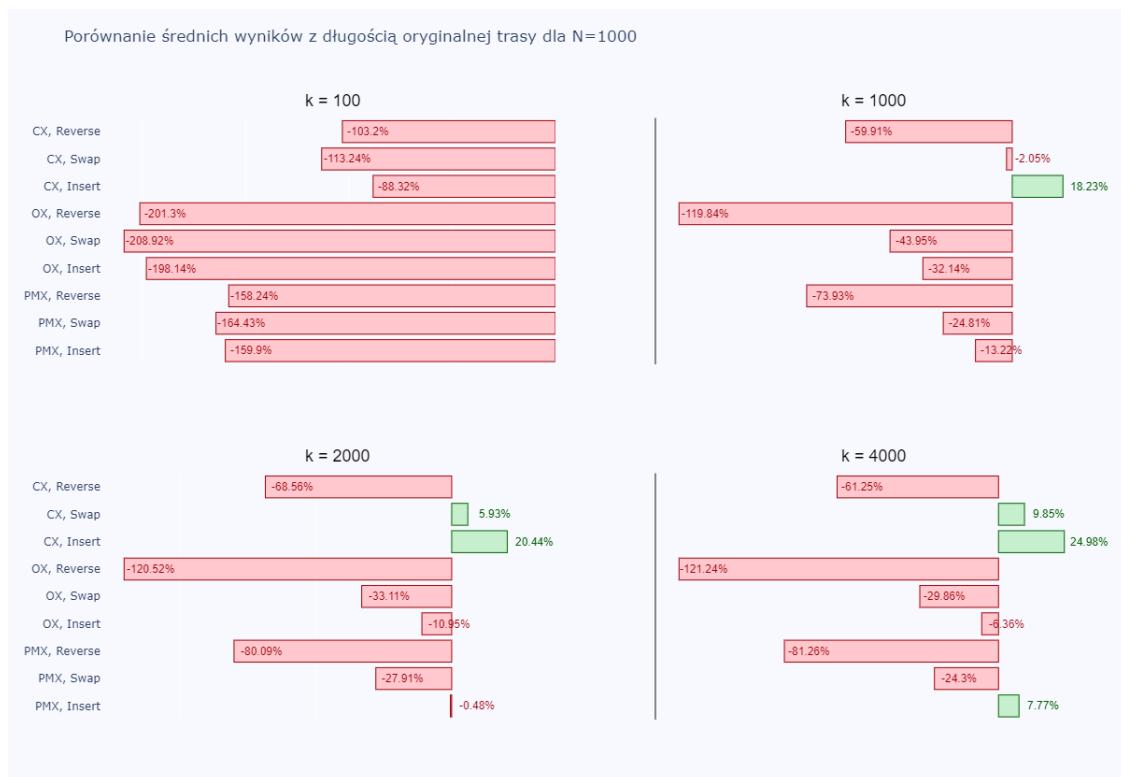
W populacji 1000 osobników (Rys. 6.31) krzyżowanie CX z mutacją $Insert$ średnio optymalizowało trasę od 18.23% do 24.98%. Krzyżowanie PMX optymalizuje trasę tylko dla $k = 4000$ z mutacją $Insert$ o 7.77%. Krzyżowanie OX osiągnęło najlepszy wynik -6%.

W populacji 2000 osobników (Rys. 6.32) krzyżowanie CX dla $k \geq 1000$ oraz mutacji $Insert$ i $Swap$, średnio zoptymalizowało trasę od 26.08% do 36.11%. Krzyżowanie PMX dla $k \geq 2000$ i mutacji $Insert$ średnio zoptymalizowało trasę od 11.73% do 14.18%. Krzyżowanie OX dla żadnej populacji nie potrafiło zoptymalizować tras.

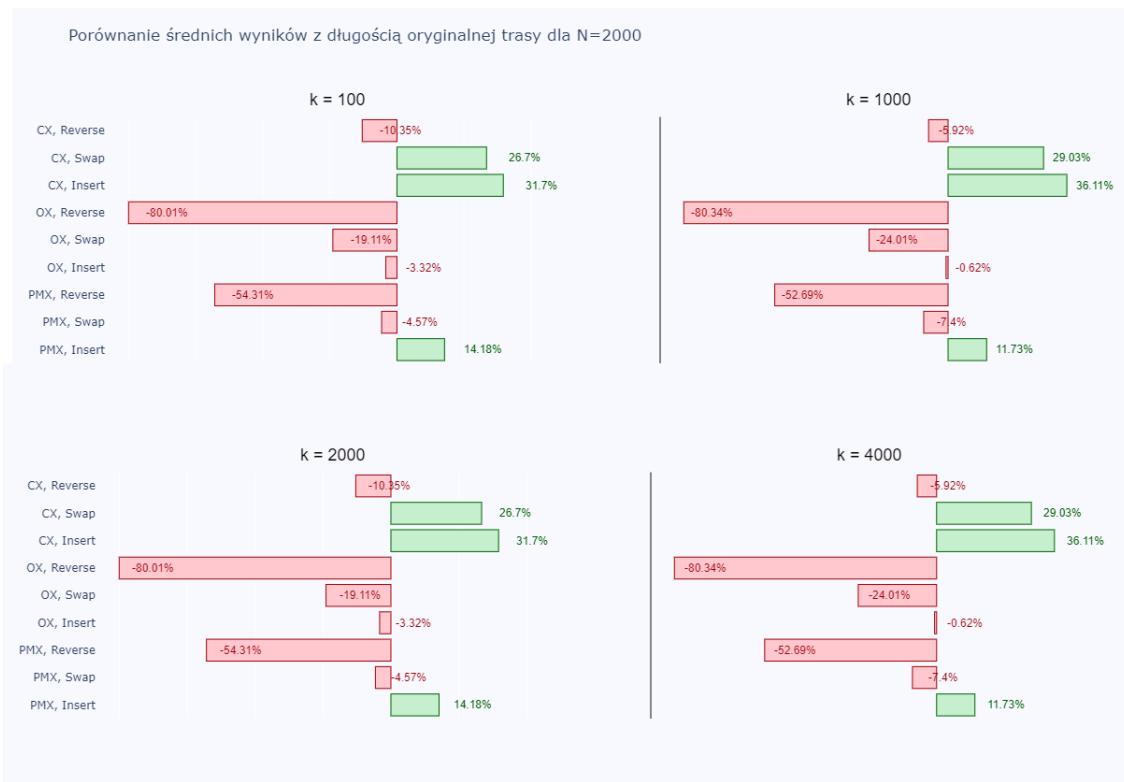
Na rysunku 6.33 przedstawiono najlepszą trasę jaką znalazła algorytm genetyczny.



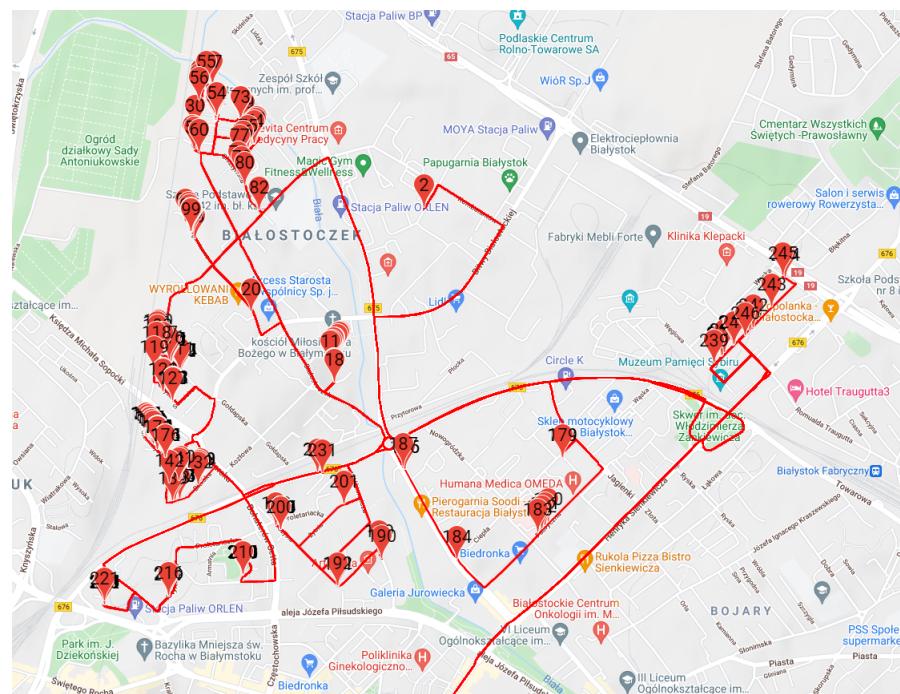
Rysunek 6.30: Trasa III - porównanie średnich wyników z długością oryginalnej trasy dla N=100



Rysunek 6.31: Trasa III - porównanie średnich wyników z długością oryginalnej trasy dla N=1000



Rysunek 6.32: Trasa III - porównanie średnich wyników z długością oryginalnej trasy dla N=2000



Rysunek 6.33: Trasa III - algorytm genetyczny

6.3.2 Wyniki algorytmu mrówkowego - Kamil

Na ostatniej badanej trasie, wyniki algorytmu mrówkowego również wyglądają dobrze. Analizując tabelę 6.31 łatwo zauważać, że wszystkie średnie wyniki były lepsze od pierwotnej długości. W najgorszym przypadku średni wynik ma wartość 39 797 metrów. Jest to wynik o 26% lepszy od oryginalnej trasy. Taki wynik został wygenerowany przy najniższej liczbie iteracji oraz dla współczynnika parowania wynoszącego 0.1. Dla tego samego współczynnika wraz ze wzrostem iteracji wynik ulega polepszeniu. Najlepszy wynik dla tego współczynnika wynosi 31935 metrów. Różnica między tą wartością a najlepszą wartością dla całej trasy jest niewielka i wynosi 359 metrów. Wartość 31 576 metrów jest o 41% lepszym wynikiem od trasy oryginalnej.

Tabela 6.31: Trasa III - średnie wyniki algorytmu mrówkowego dla poszczególnych parametrów wejściowych

k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
50	39797	31576	33115	34142	35197	35392	34947	35023	35678
75	32302	32777	34365	33947	34407	35280	35765	34893	36052
100	31935	32737	33768	33181	34423	34415	35571	36381	35566

W tabeli numer 6.32 przedstawione zostały najlepsze wyniki algorytmu mrówkowego dla tej trasy. Do wygenerowania najkrótszej trasy potrzeba było 75 iteracji. Współczynnik parowania dla tej trasy wynosi 0.1. Można zauważyć, jak ilość iteracji potrafi wpływać na wynik. Dla najniższej ilości iteracji wygenerowana trasa jest dużo dłuższa od pozostałych dwóch wariantów. Dla większości współczynników parowania sytuacja jest podobna. Wraz ze wzrostem liczby iteracji wyniki są lepsze. Dla wartości z zakresu od 0.6 do 0.9 trend ten jest zmieniony. Przykładowo dla współczynnika parowania równego 0.8, program utworzył najlepszą trasę przy 75 iteracjach. Najdłuższa trasa została natomiast odnotowana dla największej liczby iteracji.

Tabela 6.32: Trasa III - najlepsze wyniki algorytmu mrówkowego dla poszczególnych parametrów wejściowych

k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
50	38271	29339	30505	31383	31255	31141	30948	31304	32184
75	28172	28860	31254	30693	30415	32098	31372	29923	31230
100	28311	29084	31542	30573	30254	30428	29279	33067	30334

Skuteczność z jaką algorytm mrówkowy z optymalizował trasę jest najwyższa z możliwych. W tabeli 6.33 została przedstawiona procentowa wartość tras zoptymalizowanych

w stosunku do oryginalnej odległości. Wszystkie z prób okazały się lepsze. Dzięki temu można zauważyc, że niezależnie od ilości iteracji wygenerowany wynik da satysfakcjonujący wynik. Ma to znaczenie w kontekście czasu jaki potrzebuje program na ulepszenie trasy. Tak jak w przypadku poprzednich tras, algorytm mrówkowy potrzebuje wykonać wiele operacji. Do wykonania 50 prób polepszenia trasy potrzeba średnio 84.451 sekundy. Jeśli ilość prób została zwiększa do 75, to średni czas wzrósł do 131.146 sekund. Najdłużej trwało obliczanie trasy dla 100 iteracji. Czas ten wyniósł 176.874 sekund.

Tabela 6.33: Trasa III - skuteczność algorytmu mrówkowego

k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
50	100%	100%	100%	100%	100%	100%	100%	100%	100%
75	100%	100%	100%	100%	100%	100%	100%	100%	100%
100	100%	100%	100%	100%	100%	100%	100%	100%	100%

Tabela 6.34: Trasa III - średnie wyniki czasu wykonania algorytmu mrówkowego dla parametru ilości iteracji

iteracje	50	75	100
czas	84.451	131.146	176.874

Poza analizą najlepszych tras, na uwagę zasługuje porównanie oryginalnej trasy ze średnicą wyniki dla każdego ze współczynników. Na wykresie 6.34 można zauważyc, że dla najniższej badanej liczby iteracji średnia wszystkich wyników była większa o minimum 25%. Największą średnią uzyskaną podczas badań odnotowana dla współczynnika parowania równego 0.2. Średnia ta była większa o 40.96%.

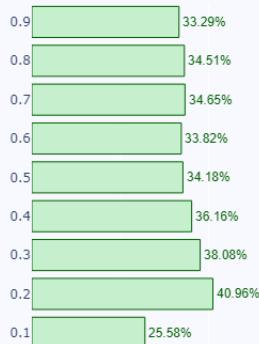
Na wykresie 6.35 zostały przedstawione średnie wyniki dla 75 iteracji. Wszystkie wartości są lepsze od początkowych. Najlepsza wartość została odnotowana dla współczynnika parowanej wynoszącej 0.1. Jest to największy progres względem mniejszej liczby iteracji.

Po zwiększeniu liczby iteracji wartości nadal utrzymywały się na wysokim poziomie. Wszystkie średnie były większe od pierwotnej trasy. Żadna średnia nie była mniejsza od 30%. Tak jak dla 75 iteracji, tak i dla 100, najlepsze średnie wyniki zostały odnotowane dla współczynnika wartości równej 0.1.

Na rysunku 6.39 przedstawiono najlepszą trasę jaką znalazła algorytm mrówkowy.

Porównanie średnich wyników z długością oryginalnej trasy.

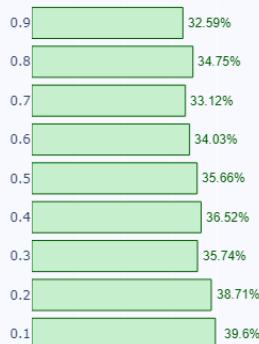
$k = 50$



Rysunek 6.34: Procentowy rozkład średnich wartości dla 50 iteracji

Porównanie średnich wyników z długością oryginalnej trasy.

$k = 75$

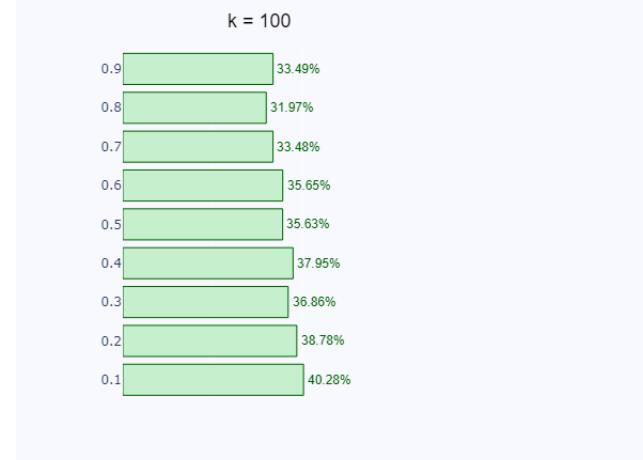


Rysunek 6.35: Procentowy rozkład średnich wartości dla 75 iteracji

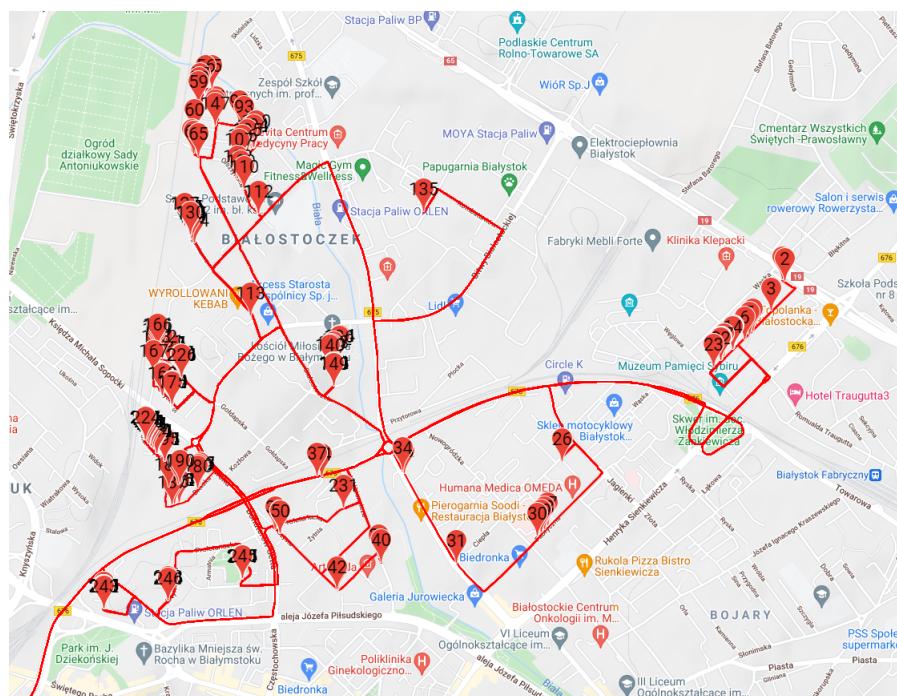
6.3.3 Wyniki algorytmów zachłannych - Przemysław

Tabela 6.35 przedstawia najlepsze wyniki uzyskane przez algorytmy zachłanne dla ostatniej badanej trasy. Jak w poprzednich badaniach tutaj również został zastosowany współczynnik błędu przy dokonywaniu wyborów na każdej z poszczególnych trzech faz tworzenia tras. Bez zmian najlepiej dalej zachowuje się algorytm A*, który dalej znajduje ścieżkę o najmniejszej wartości równą 32855 metrów. Tak jak w poprzednich badaniach najgorzej wypadł algorytm najmniejszej krawędzi, dla którego najlepsza trasa uzyskała wynik 34130 metrów co oznacza, że prawie wszystkie wyznaczone trasy przez algorytm A* zdobyły lepszy wynik. Algorytm najbliższego sąsiada tym razem poradził sobie tak samo

Porównanie średnich wyników z długością oryginalnej trasy.



Rysunek 6.36: Procentowy rozkład średnich wartości dla 100 iteracji



Rysunek 6.37: Trasa III - algorytm mrówkowy

dobrze jak algorytm A*, gdzie dobrze widać na tabeli 6.36 najlepsze wyniki są niemal równe wynoszące około 38.55% optymalizacji względem oryginalnej ścieżki.

Tabela 6.37 oraz 6.38 przedstawiają wyniki użycia metody 2-opt na powstały trasach. Po zastosowaniu rozwiązania 2-opt na zbudowanych wcześniej trasach okazuje się, że nowy zwycięzcą spośród badanych algorytmów stał się algorytm najbliższego sąsiada z nowym najlepszym wynikiem równym 31781 metrów co daje aż 40.57% poprawy ścieżki względem uzyskanej początkowo.

Tabela 6.35: Trasa III - wyniki algorytmów zachłannych

Błąd (m)	ANS-1	ANS-2	ANS-3	ANK-1	ANK-2	ANK-3	A*-1	A*-2	A*-3
0	33789	35566	35207	34130	37006	37719	33412	33410	33411
5	33447	35206	35567	35836	37724	35922	33771	33410	33770
10	33447	34839	34717	34130	37000	37717	34131	34129	33770
20	33106	35198	35183	34130	37000	35920	32856	33410	33770
30	33789	34839	34472	34130	37000	35920	32856	34129	33770
40	33106	34836	34472	35836	35922	36998	32856	33770	33770
50	33789	35555	32861	35836	37000	36998	33563	33402	33528
75	33106	35196	32861	35154	37718	35921	33208	33757	33406
100	33106	35555	32861	35836	37000	35921	32855	34122	33406

Tabela 6.36: Trasa III - optymalizacja wyników względem oryginalnej trasy

Błąd (m)	ANS-1	ANS-2	ANS-3	ANK-1	ANK-2	ANK-3	A*-1	A*-2	A*-3
0	36,82%	33,49%	34,17%	36,18%	30,80%	29,47%	37,52%	37,53%	37,52%
5	37,46%	34,17%	33,49%	32,99%	29,46%	32,83%	36,85%	37,53%	36,85%
10	37,46%	34,85%	35,08%	36,18%	30,81%	29,47%	36,18%	36,18%	36,85%
20	38,09%	34,18%	34,21%	36,18%	30,81%	32,83%	38,56%	37,53%	36,85%
30	36,82%	34,85%	35,54%	36,18%	30,81%	32,83%	38,56%	36,18%	36,85%
40	38,09%	34,86%	35,54%	32,99%	32,83%	30,82%	38,56%	36,85%	36,85%
50	36,82%	33,51%	38,55%	32,99%	30,81%	30,82%	37,24%	37,54%	37,31%
75	38,09%	34,19%	38,55%	34,26%	29,47%	32,83%	37,90%	36,88%	37,53%
100	24,81%	33,51%	38,55%	32,99%	30,81%	32,83%	38,56%	36,19%	37,53%

Rysunek 6.38 przedstawia porównanie optymalizacji tras z długością oryginalnej trasy.

Wszystkie wyniki są znakomite, jednakże zastosowanie metody 2-opt pozwoliło zwiększyć górny wynik algorytmu najbliższego sąsiada do 40.57%, a algorytmu A* do 38.75%. Algorytm najmniejszej krawędzi po zastosowaniu metody 2-opt nie otrzymał nowych lepszych wyników.

Tabela 6.37: Trasa III - zastosowanie metody 2-opt na powstałych trasach

Błąd (m)	ANS-1	ANS-2	ANS-3	ANK-1	ANK-2	ANK-3	A*-1	A*-2	A*-3
0	33447	35566	35207	34130	37006	37719	33208	33410	33411
5	33447	35206	35567	34130	37724	35922	33208	33410	32756
10	33447	34839	34717	34130	35922	37717	33208	33410	32756
20	31781	35198	35183	34130	35922	35920	32856	33410	32756
30	33447	34839	32861	34130	35922	35920	32856	33410	32756
40	31781	34836	32861	34130	35922	35920	32856	33770	32756
50	33447	34839	32861	34130	35922	35920	33208	33402	33528
75	31781	35196	32861	35154	35454	35921	33208	33757	33406
100	31781	34839	32861	34130	35922	35921	32855	33410	33406

Tabela 6.39 przedstawia czasy wykonywania algorytmów zachłannych dla trasy nr 3.

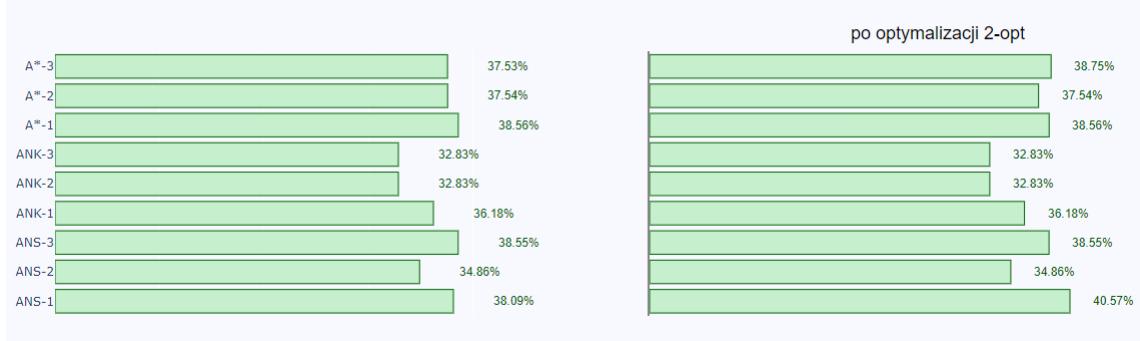
Najlepszy wynik otrzymaliśmy w 65.1 sekundy.

Na rysunku 6.25 przedstawiono najlepszą trasę jaką znalazł algorytm mrówkowy.

Tabela 6.38: Trasa III - optymalizacja powstałych tras przez metodę 2-opt

Błąd (m)	ANS-1	ANS-2	ANS-3	ANK-1	ANK-2	ANK-3	A*-1	A*-2	A*-3
0	37,46%	33,49%	34,17%	36,18%	30,80%	29,47%	37,90%	37,53%	37,52%
5	37,46%	34,17%	33,49%	36,18%	29,46%	32,83%	37,90%	37,53%	38,75%
10	37,46%	34,85%	35,08%	36,18%	32,83%	29,47%	37,90%	37,53%	38,75%
20	40,57%	34,18%	34,21%	36,18%	32,83%	32,83%	38,56%	37,53%	38,75%
30	37,46%	34,85%	38,55%	36,18%	32,83%	32,83%	38,56%	37,53%	38,75%
40	40,57%	34,86%	38,55%	36,18%	32,83%	32,83%	38,56%	36,85%	38,75%
50	37,46%	34,85%	38,55%	36,18%	32,83%	32,83%	37,90%	37,54%	37,31%
75	40,57%	34,19%	38,55%	34,26%	33,70%	32,83%	37,90%	36,85%	37,53%
100	40,57%	34,85%	38,55%	36,18%	32,83%	32,83%	38,56%	37,53%	37,53%

Porównanie optymalizacji tras z długością oryginalnej trasy.



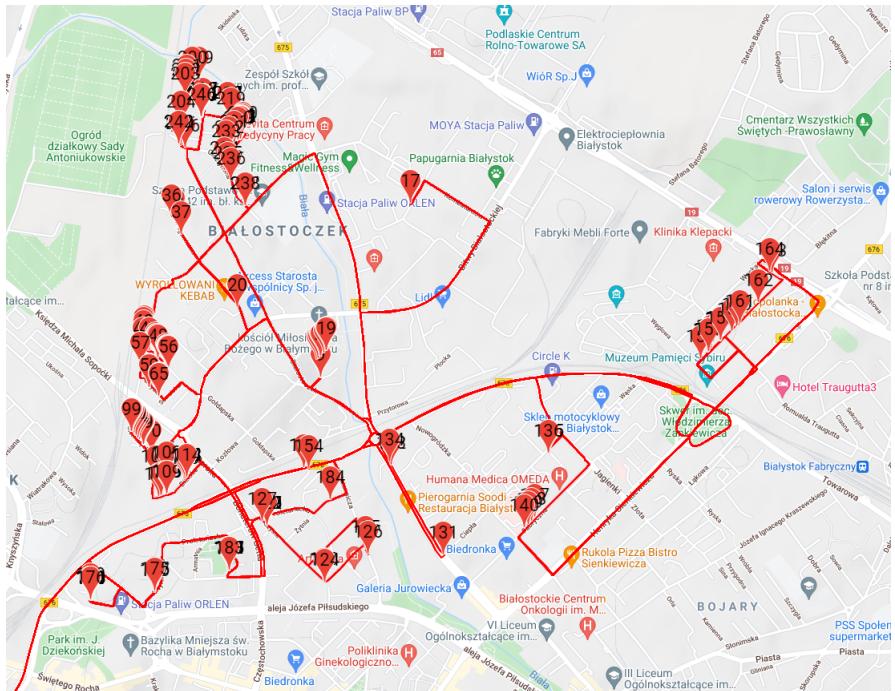
Rysunek 6.38: Trasa III - porównanie optymalizacji algorytmów z zastosowanie metody 2-opt

Tabela 6.39: Trasa III - czasy wykonywania algorytmów

Błąd (m)	ANS-1	ANS-2	ANS-3	ANK-1	ANK-2	ANK-3	A*-1	A*-2	A*-3
0	39,19	2,26	41,38	17,08	0,66	0,98	14,68	2,98	22,87
5	49,02	1,63	39,12	24,32	0,38	26,84	30,63	2,91	22,38
10	51,45	38,71	39,76	25,9	0,46	40,75	43,69	2,98	32,58
20	65,1	38,18	41,37	27,3	0,59	42,28	0,57	8,31	32,35
30	85,67	39,24	58,65	26,81	0,71	95,72	0,71	8,69	32,62
40	87,95	99,26	83,85	26,98	0,77	137,77	0,9	0,71	32,72
50	88,71	19,47	125,16	34,24	0,8	184,5	1,23	83,84	34,2
75	91,05	20,8	127,13	33,87	4,53	186,57	6,52	119,22	18,12
100	92,9	21,06	178,76	34,52	4,89	100,17	7,66	11,8	26,5

6.3.4 Porównanie algorytmów - wspólnie

Ostatnia z tras składała się z największej liczby punktów. Wszystkie algorytmy wykonywały się dłużej niż na poprzednich trasach. Każdy z algorytmów poprawił pierwotną trasę. Algorytm genetyczny wygenerował trasę o długości 32374. W porównaniu do oryginału jest to poprawa o 40%. Program potrzebował średnio 8.25 sekundy na optymalizację trasy. Najlepszy rezultat został otrzymany przy pomocy algorytmu mrówkowego. Trasa o długości 28172 jest krótsza o 47% od początkowej ścieżki. Średni czas z jakim pracował algorytm

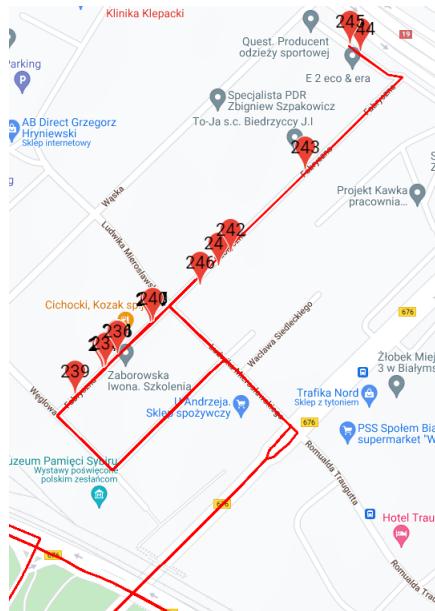


Rysunek 6.39: Trasa III - algorytm zachłanny

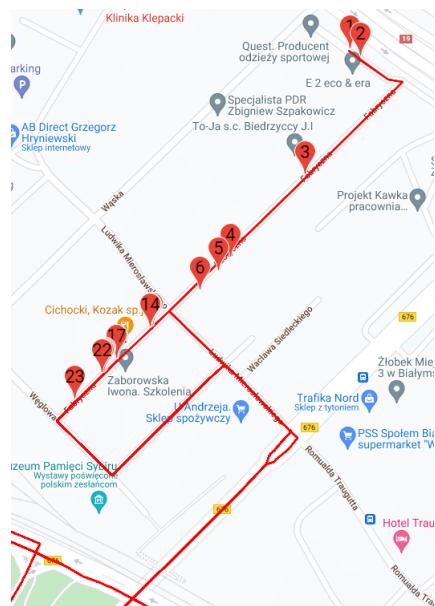
nad wygenerowaniem takiej trasy to 131.146 sekundy. Najkrótsza ścieżka dla algorytmów zachłannych została znaleziona przez algorytm A* o długości 32 855 metrów z czasem działania algorytmu 7.66 sekund. Po zoptymalizowaniu otrzymanych tras metodą 2-opt otrzymano algorymem najbliższego sąsiada lepszy wynik o wartości 31 781 w czasie 65.1 sekundy. Jest to wartość o 41% lepsza od oryginalnej.

Najdłuższa trasa została wygenerowana przy użyciu algorytmu genetycznego. Jednak czas jaki był potrzebny na tą optymalizację był o wiele krótszy od pozostałych dwóch algorytmów. Algorytm najbliższego sąsiada jest niewiele lepszy od algorytmu genetycznego. Czas potrzebny na wszystkie operacje był ponad siedmiokrotnie dłuższy. Najkrótsza trasa została wygenerowana przez algorytm mrówkowy. Niestety czas potrzebny do wygenerowania trasy był najdłuższy.

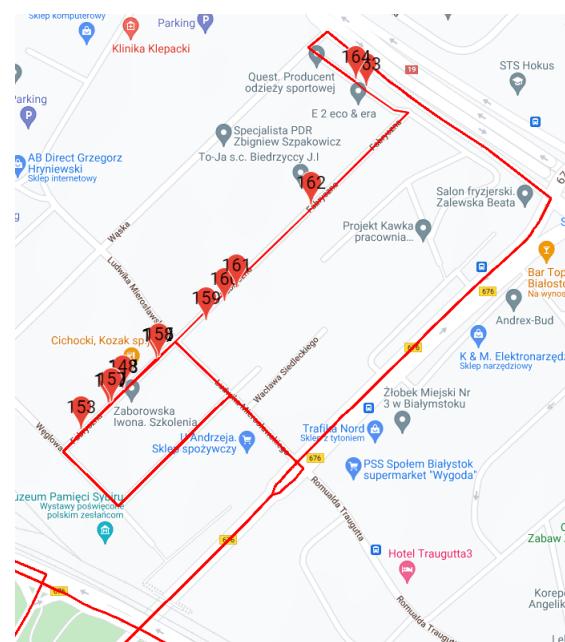
Na zdjęciach 6.26, 6.27, 6.28 zostały pokazane przykładowe różnice w znalezionych trasach na tym samym odcinku przez algorytmy. Każdy z algorytmów w innej kolejności odwiedził te punkty. Algorytm mrówkowy już na samym początku wyselekcyjował widoczne punkty. Natomiast algorytm genetyczny na samym końcu przeanalizował przedstawiony obszar. Jeszcze inaczej zachował się algorytm najbliższego sąsiada, który w środku trasy odwiedził punkty.



Rysunek 6.40: Fragment trasy algorytmu genetycznego



Rysunek 6.41: Fragment trasy algorytmu mrówkowego



Rysunek 6.42: Fragment trasy algorytmu najbliższego sąsiada

Podsumowanie

Celem pracy jaka została wykonana była optymalizacja tras przejazdu ciężarówek z odpadami komunalnymi. Wykorzystane rozwiązania są dedykowane do prac nad optymalizacją problemu komiwojażera. Dla każdego z algorytmów należało znaleźć takie parametry wejściowe, aby optymalizacje długości tras były jak najlepsze.

Dzięki wykorzystaniu realnych danych do optymalizacji ścieżek przejazdów ciężarówek z odpadami komunalnymi można zbadać oraz zaproponować rozwiązanie problemu. Wybrane ścieżki stanowią faktyczne trasy jakie były przebywane. Wybrane trzy losowo trasy składały się odpowiednio z 111, 150 oraz 248 punktów. Dzięki udostępnionym rozwiązaniom przez firmę Google, zostały wyliczone rzeczywiste odległości pomiędzy wszystkimi punktami.

Do optymalizacji zostały wybrane trzy algorytmy. Algorytm genetyczny, który został opisany i zbadany przez Pawła Stypułkowskiego. Kolejnym algorytmem wybranym był algorytm mrówkowy autorstwa Kamila Łętowskiego. Jako ostatnie zostało zaproponowanych kilka algorytmów zachłannych przez Przemysława Noskowicza.

Algorytm genetyczny oraz mrówkowy dla każdej możliwej konfiguracji parametrów wejściowych został uruchomiony dziesięć razy. Algorytm genetyczny został zbadany dla trzech populacji N : 100, 1000 oraz 2000; czterech różnych wartości iteracji k : 100, 1000, 2000 oraz 4000; trzech różnych krzyżowań: PMX , OX oraz CX ; trzech różnych mutacji: *Reverse*, *Swap* oraz *Insert*. Wykonanie wszystkich badań dla jednej trasy trwało kilka godzin. W algorytmie mrówkowym parametrami wejściowymi była liczba iteracji oraz współczynnik parowania feromonów. Wykonanych było 50, 75 lub 100 iteracji a współczynnik zakresu był rozpoczynał się od 0.1 i był zwiększany do 0.9 co 0.1. W ten sposób dla jednej trasy było wykonanych 270 badań. Przedstawicielami algorytmów zachłannych użytych w badaniach były algorytm najbliższego sąsiada, algorytm najmniejszej krawędzi oraz algorytm A*. W celu rozważenia większej ilości tras zastosowano podczas dokonywania wyborów współczynnik błędu. Współczynnik błędu był uruchamiany na trzech różnych etapach budowania docelowej trasy. Utworzone trasy zostały zoptymalizowane metodą 2-opt.

Po przeprowadzeniu badań z pewnością można powiedzieć, że pierwotne trasy nie były optymalne. Omówione rozwiązania były w stanie poprawić oryginalną trasę nawet o 40%. Każdy z algorytmów był w stanie wygenerować wiele krótszych tras. Najlepsze wyniki dla trasy I o długości 48897 metrów:

- 27131 – algorytm mrówkowy, średni czas 21.73 sekund, parametry wejściowe:
 $wspczynnikparowania = 0.4$, $liczbainteracji = 75$,
- 29965 – algorytm zachłanny - A*, czas: 7.91 sekund, współczynnik błędu: 10 metrów,
etap: 1,
- 30939 – algorytm genetyczny, czas średni: 8.33 sekund, parametry wejściowe: $N = 2000$,
 $k = 1000$, krzyżowanie PMX oraz mutacja *Insert*.

Najlepsze wyniki dla trasy II o długości 40402 metrów:

- 24347 – algorytm mrówkowy, średni czas 28.32 sekund, parametry wejściowe:
 $wspczynnikparowania = 0.4$, $liczbainteracji = 75$,
- 27381 – algorytm genetyczny, czas średni: 9.03 sekund, parametry wejściowe: $N = 2000$,
 $k = 2000$, krzyżowanie CX oraz mutacja *Swap*,
- 28757 – algorytm zachłanny - A*, czas: 8.05 sekund, współczynnik błędu: 10 metrów,
etap: 3.

Najlepsze wyniki dla trasy III o długości 53478 metrów:

- 28172 – algorytm mrówkowy, średni czas 131.14 sekund, parametry wejściowe:
 $wspczynnikparowania = 0.1$, $liczbainteracji = 75$,
- 31781 – algorytm zachłanny - ANS, czas: 65.1 sekund, współczynnik błędu: 20 metrów,
etap: 1,
- 32374 – algorytm genetyczny, czas średni: 8.25 sekund, parametry wejściowe: $N = 2000$,
 $k = 1000$, krzyżowanie CX oraz mutacja *Insert*.

Przeprowadzone badania pokazują jak można rozwiązać problem z którym nawet nieświadomie borykają się firmy. W dalszej pracy rozwiązania mogą być rozwijane w wielu kierunkach na przykład można zastosować podejście hybrydowe. Początkowe

trasy do algorytmu genetycznego i mrówkowego, mogłyby być generowane za pomocą algorytmów zachłannych. Na wszystkich znalezionych trasach można zastosować operatory optymalizujące na przykład 2-opt lub inne.

Bibliografia

- [1] Encyklopedia Algorytmów. Encyklopedia algorytmów. http://algorytmy.ency.pl/artykul/algorytm_helda_karpa, 2016.
- [2] Encyklopedia Algorytmów. Encyklopedia algorytmów. http://algorytmy.ency.pl/artykul/algorytm_najblizszego_sasiada, 2016.
- [3] Lenart Andrzej. *Maszynoznawstwo przemysłu spożywczego*. SGGW, Warszawa, 2003.
- [4] Daniel Błaszkiewicz. Algorytmy mrówkowe. *Instytut Informatyki Uniwersytetu Wrocławskiego*, 1(1):5–6.
- [5] Ronald S. Calinger. *An acclaimed biography of the Enlightenment's greatest mathematician*. Princeton University Press, 2015.
- [6] Ewa Figielska. Algorytmy ewolucyjne i ich zastosowania. *Zeszyty Naukowe Warszawskiej Wyższej Szkoły Informatyki*, 1(1):81–92, 2006.
- [7] D. Goldberg and R. Lingle. Alleles, loci and the traveling salesman problem. *Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications*, pages 154–159, 1985.
- [8] David E. Goldberg. *Algorytmy genetyczne i ich zastosowania*. Wydawnictwo Naukowo-Techniczne, Warszawa, 1998.
- [9] Anna Gryko-Nikitin. Niektóre osobliwości algorytmów genetycznych na przykładzie zagadnień logistycznych. *Ekonomia i Zarządzanie*, 1(2):129–138, 2010.
- [10] D. J. d. Smith I. M. Oliver and R. C. J. Holland. A study of permutation crossover operators on the traveling salesman problem. *In Proceedings of the second international conference. on genetic algorithms*, pages 224–230, 1987.
- [11] Hadush Mebrahtu Kusum Deep. New variations of order crossover for travelling salesman problem. *International Journal of Combinatorial Optimization Problems and Informatics*, 2:2–13, 2011.
- [12] Thomas Stützle Marco Dorigo. *Ant Colony Optimization*. The MIT Press, 2004.

- [13] Rutkowski L Rutkowska D., Piliński M. *Sieci neuronowe, algorytmy genetyczne i systemy rozmyte*. Wydawnictwo Naukowe PWN, Warszawa, 1999.
- [14] Krzysztof Schiff. Ant colony optimization algorithm for the 0-1 knapsack problem. *Technical transactions automatic control*, 1(1):40–51, 2013.
- [15] Alexander Schrijver. On the history of combinatorial optimization (till 1960). 1(1):1–3.
- [16] Radosław Winiczenko. Algorytmy genetyczne i ich zastosowania. *Postępy Techniki Przetwórstwa Spożywczego*, 1(2):107–110, 2008.

Spis tabel

Tablica 3.1	Wartości kosztów	35
Tablica 6.1	Trasa I - średnie wyniki dla algorytmu genetycznego	53
Tablica 6.2	Trasa I - skuteczność algorytmu genetycznego	54
Tablica 6.3	Trasa I - najlepsze wyniki dla algorytmu genetycznego	55
Tablica 6.4	Trasa I - średnie czasy wykonywania algorytmu genetycznego	55
Tablica 6.5	Trasa I - średnie wyniki dla algorytmu mrówkowego	59
Tablica 6.6	Trasa I - najlepsze wyniki dla algorytmu mrówkowego	59
Tablica 6.7	Trasa I - skuteczność algorytmu mrówkowego	60
Tablica 6.8	Trasa I - średnie wyniki czasu wykonania dla algorytmu mrówko- wego dla parametru ilości iteracji	60
Tablica 6.9	Trasa I - wyniki algorytmów zachłannych	63
Tablica 6.10	Trasa I - optymalizacja wyników względem oryginalnej trasy	63
Tablica 6.11	Trasa I - zastosowanie metody 2-opt na powstałych trasach	64
Tablica 6.12	Trasa I - optymalizacja powstałych tras przez metodę 2-opt	64
Tablica 6.13	Trasa I - czasy wykonywania algorytmów	64
Tablica 6.14	Trasa II - średnie wyniki dla algorytmu genetycznego	69
Tablica 6.15	Trasa II - skuteczność algorytmu genetycznego	70
Tablica 6.16	Trasa II - najlepsze wyniki dla algorytmu genetycznego	70
Tablica 6.17	Trasa II - średnie czasy wykonywania algorytmu genetycznego	71
Tablica 6.18	Trasa II - średnie wyniki dla algorytmu mrówkowego	74
Tablica 6.19	Trasa II - najlepsze wyniki dla algorytmu mrówkowego	74
Tablica 6.20	Trasa II - skuteczność algorytmu mrówkowego	75
Tablica 6.21	Trasa II - średnie wyniki czasu wykonania dla algorytmu mrówko- wego dla parametru ilości iteracji	75
Tablica 6.22	Trasa II - wyniki algorytmów zachłannych	77
Tablica 6.23	Trasa II - optymalizacja wyników względem oryginalnej trasy	78
Tablica 6.24	Trasa II - zastosowanie metody 2-opt na powstałych trasach	78
Tablica 6.25	Trasa II - optymalizacja powstałych tras przez metodę 2-opt	79
Tablica 6.26	Trasa II - czasy wykonywania algorytmów	79

Tablica 6.27 Trasa III - średnie wyniki dla algorytmu genetycznego	83
Tablica 6.28 Trasa III - skuteczność algorytmu genetycznego	84
Tablica 6.29 Trasa III - najlepsze wyniki dla algorytmu genetycznego	85
Tablica 6.30 Trasa III - średnie czasy wykonywania algorytmu genetycznego . . .	85
Tablica 6.31 Trasa III - średnie wyniki dla algorytmu mrówkowego	88
Tablica 6.32 Trasa III - najlepsze wyniki dla algorytmu mrówkowego	88
Tablica 6.33 Trasa III - skuteczność algorytmu mrówkowego	89
Tablica 6.34 Trasa III - średnie wyniki czasu wykonania dla algorytmu mrówko- wego dla parametru ilości iteracji	89
Tablica 6.35 Trasa III - wyniki algorytmów zachłannych	92
Tablica 6.36 Trasa III - optymalizacja wyników względem oryginalnej trasy	92
Tablica 6.37 Trasa III - zastosowanie metody 2-opt na powstałych trasach	92
Tablica 6.38 Trasa III - optymalizacja powstałych tras przez metodę 2-opt	93
Tablica 6.39 Trasa III - czasy wykonywania algorytmów	93

Spis rysunków

Rysunek 1.1	Graf bez cyklu Hamiltona.	14
Rysunek 1.2	Graf z cyklem Hamiltona.	14
Rysunek 2.1	Schemat algorytmu genetycznego	22
Rysunek 2.2	Metoda ruletki	23
Rysunek 2.3	Klasyczne krzyżowanie	24
Rysunek 2.4	Mutacja genotypu	24
Rysunek 2.5	Kodowanie chromosomów	25
Rysunek 2.6	Krzyżowanie PMX - część 1	26
Rysunek 2.7	Krzyżowanie PMX - część 2	27
Rysunek 2.8	Krzyżowanie OX	28
Rysunek 2.9	Krzyżowanie CX	28
Rysunek 2.10	Mutacja odwracająca	29
Rysunek 2.11	Mutacja wstawiająca	30
Rysunek 2.12	Mutacja zamieniająca	30
Rysunek 3.1	Położenie ścieżek na przykładowym grafie	33
Rysunek 3.2	Rozkład agentów na grafie po N początkowych iteracjach	33
Rysunek 3.3	Rozkład agentów w grafie po optymalizacji	34
Rysunek 3.4	Graf z początkowymi wartościami feromonów	35
Rysunek 3.5	Graf z wagami	36
Rysunek 3.6	Trasa przebyta przez agenta L1	37
Rysunek 3.7	Trasa przebyta przez agenta L2	38
Rysunek 3.8	Graf z wartościami feromonów po modyfikacji	38
Rysunek 4.1	Schemat algorytmu najbliższego sąsiada	40
Rysunek 4.2	Algorytm najbliższego sąsiada	41
Rysunek 4.3	Schemat algorytmu najmniejszej krawędzi	42
Rysunek 4.4	Początkowe rozmieszczenie wierzchołków	43
Rysunek 4.5	Algorytm najmniejszej krawędzi	43
Rysunek 4.6	Schemat blokowy algorytmu A*	45

Rysunek 4.7	Graf użyty do przedstawienia działania algorytmu A*	46
Rysunek 4.8	Poszczególne iteracje algorytmu A*	47
Rysunek 4.9	Metoda 2-opt	47
Rysunek 6.1	Trasa I	53
Rysunek 6.2	Trasa I - porównanie średnich wyników z długością oryginalnej trasy dla N=100	56
Rysunek 6.3	Trasa I - porównanie średnich wyników z długością oryginalnej trasy dla N=1000	57
Rysunek 6.4	Trasa I - porównanie średnich wyników z długością oryginalnej trasy dla N=2000	57
Rysunek 6.5	Trasa I - algorytm genetyczny.	58
Rysunek 6.6	Procentowy rozkład średnich wartości dla 50 iteracji	60
Rysunek 6.7	Procentowy rozkład średnich wartości dla 75 iteracji	61
Rysunek 6.8	Procentowy rozkład średnich wartości dla 100 iteracji	61
Rysunek 6.9	Trasa I - algorytm mrówkowy	62
Rysunek 6.10	Trasa I - porównanie optymalizacji algorytmów z zastosowanie metody 2-opt	65
Rysunek 6.11	Trasa I - algorytm zachłanny	65
Rysunek 6.12	Fragment trasy algorytmu mrówkowego	66
Rysunek 6.13	Fragment trasy algorytmu genetycznego	67
Rysunek 6.14	Fragment trasy algorytmu A*	67
Rysunek 6.15	Trasa II	68
Rysunek 6.16	Trasa II - porównanie średnich wyników z długością oryginalnej trasy dla N=100	71
Rysunek 6.17	Trasa II - porównanie średnich wyników z długością oryginalnej trasy dla N=1000	72
Rysunek 6.18	Trasa II - porównanie średnich wyników z długością oryginalnej trasy dla N=2000	73
Rysunek 6.19	Trasa II - algorytm genetyczny	73
Rysunek 6.20	Procentowy rozkład średnich wartości dla 50 iteracji	75
Rysunek 6.21	Procentowy rozkład średnich wartości dla 75 iteracji	76
Rysunek 6.22	Procentowy rozkład średnich wartości dla 100 iteracji	76

Rysunek 6.23 Trasa II - algorytm mrówkowy	77
Rysunek 6.24 Trasa II - porównanie optymalizacji algorytmów z zastosowaniem metody 2-opt	79
Rysunek 6.25 Trasa II - algorytm zachłanny	80
Rysunek 6.26 Fragment trasy algorytmu genetycznego	81
Rysunek 6.27 Fragment trasy algorytmu mrówkowego	81
Rysunek 6.28 Fragment trasy algorytmu A*	82
Rysunek 6.29 Trasa III	82
Rysunek 6.30 Trasa III - porównanie średnich wyników z długością oryginalnej trasy dla N=100	86
Rysunek 6.31 Trasa III - porównanie średnich wyników z długością oryginalnej trasy dla N=1000	86
Rysunek 6.32 Trasa III - porównanie średnich wyników z długością oryginalnej trasy dla N=2000	87
Rysunek 6.33 Trasa III - algorytm genetyczny	87
Rysunek 6.34 Procentowy rozkład średnich wartości dla 50 iteracji	90
Rysunek 6.35 Procentowy rozkład średnich wartości dla 75 iteracji	90
Rysunek 6.36 Procentowy rozkład średnich wartości dla 100 iteracji	91
Rysunek 6.37 Trasa III - algorytm mrówkowy	91
Rysunek 6.38 Trasa III - porównanie optymalizacji algorytmów z zastosowaniem metody 2-opt	93
Rysunek 6.39 Trasa III - algorytm zachłanny	94
Rysunek 6.40 Fragment trasy algorytmu genetycznego	95
Rysunek 6.41 Fragment trasy algorytmu mrówkowego	95
Rysunek 6.42 Fragment trasy algorytmu najbliższego sąsiada	96