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. .....

Słowa 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

Temat po angielsku.

Summary

Streszczenie pracy po angielsku.

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

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 jakie 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 zauważyli wiele problemów z jakimi boryka się miasto. Jednak największą uwagę organizatorów przykuły inne anomalie.

Jedna z grup zauważyła, że trasy jakie pokonują ciężarówki są nieoptymalne. Jako przykład została przedstawiona droga losowej ciężarówki. Można było zauważyć, 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 przykładali 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.

W naszej pracy magisterskiej wykorzystamy algorytmy dla których znajdziemy takie parametry wejściowe aby wynik ich prac był jak najbardziej optymalny. Jako pierwsze rozwiązanie zostanie przedstawiony algorytm genetyczny który został opracowany przez Pawła Stypułkowskiego. Kamil Łętowski przygotował i opisał zagadnienia związane z algorytmem mrówkowym. Na samym końcu kilka rozwiązań heurystycznych zostało omówionych przez Przemysława Noskowicza.

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.

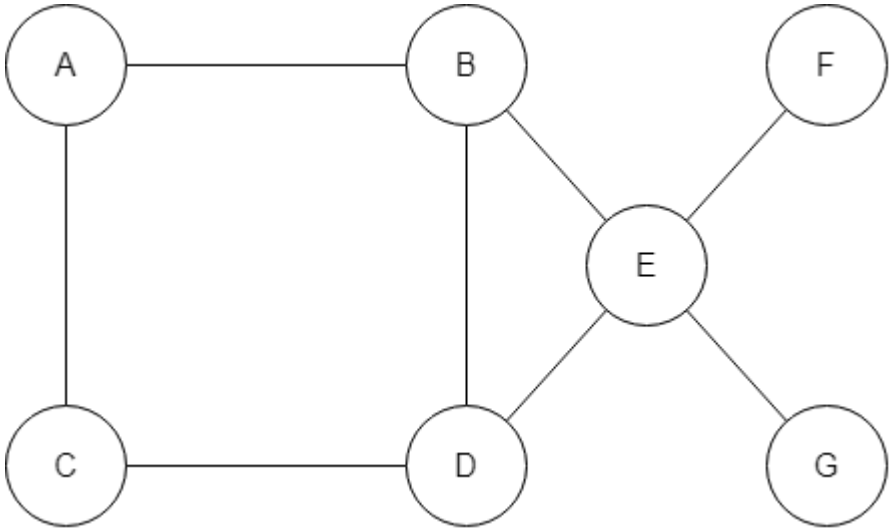
Podobnym problemem do TSP jest problem konika szachowego. Problem ten również należy do rodziny problemów NP-zupełnym. Już w XVIII wieku badania nad tym problemem rozpoczął Euler. Rozwiązanie tego problemu polega na znalezieniu ścieżki jaką ma przebyć konik szachowy, tak aby odwiedzić każde pole na szachownicy 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 podróż 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

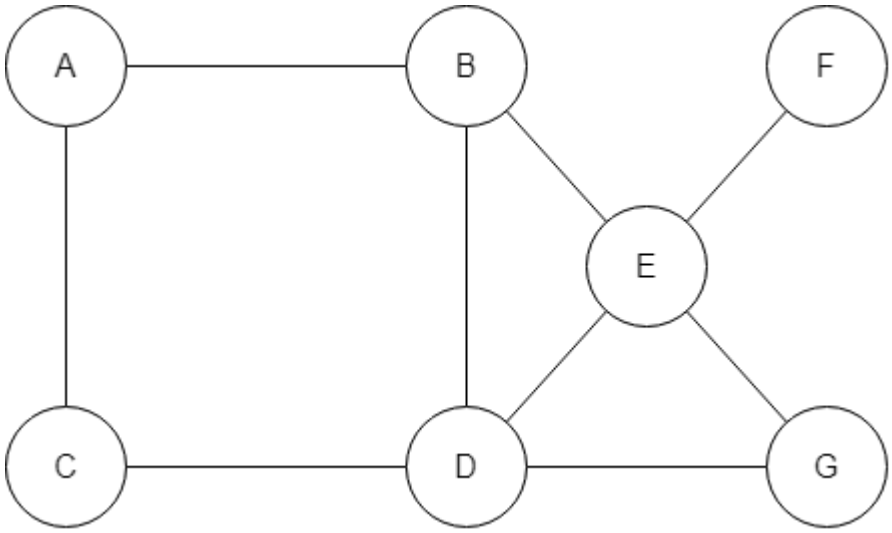
Hamiltona. Jeśli w grafie można wyróżnić cykl z opisanymi powyżej warunkami, to graf jest grafem Hamiltonowskim.

Na **??** 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 **??** 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 algorytmy 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 rozwiązania 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 mogę 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 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 informację, lekarz musi podjąć decyzję, czy np. nowotwór jest złośliwy, czy ma łagodny przebieg. Algorytm genetyczny pozwala wspomóc lekarza przy podejmowania takich decyzji, przetwarzając i analizując te ogromne zbiory. [10]. 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, ciśnieniu 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 [9].

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. 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 mogłoby być odtworzenie przebytej ścieżki. Na początku należałoby zadać sobie pytanie w jaki sposób te niewielkich rozmiarów owady są w stanie znacząco 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 pozostawionemu 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 jakichś 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 bardzo podobnie w tym przypadku. 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 bardziej zadowalające przedmioty.

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 konwencją przez autora artykułu 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 odmianą 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 [6].

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 najniższy kolor który nie został dotychczas użyty do kolorowania któregoś z sąsiadów.

Tak jak w przypadku poprzednich algorytmów wykorzystywane są zapachy pozostawiane przez mrówki. Ilość 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[6].

W tej metodzie, podobnie jak w metodzie do rozwiązania 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 rozwiązania 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 w 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.

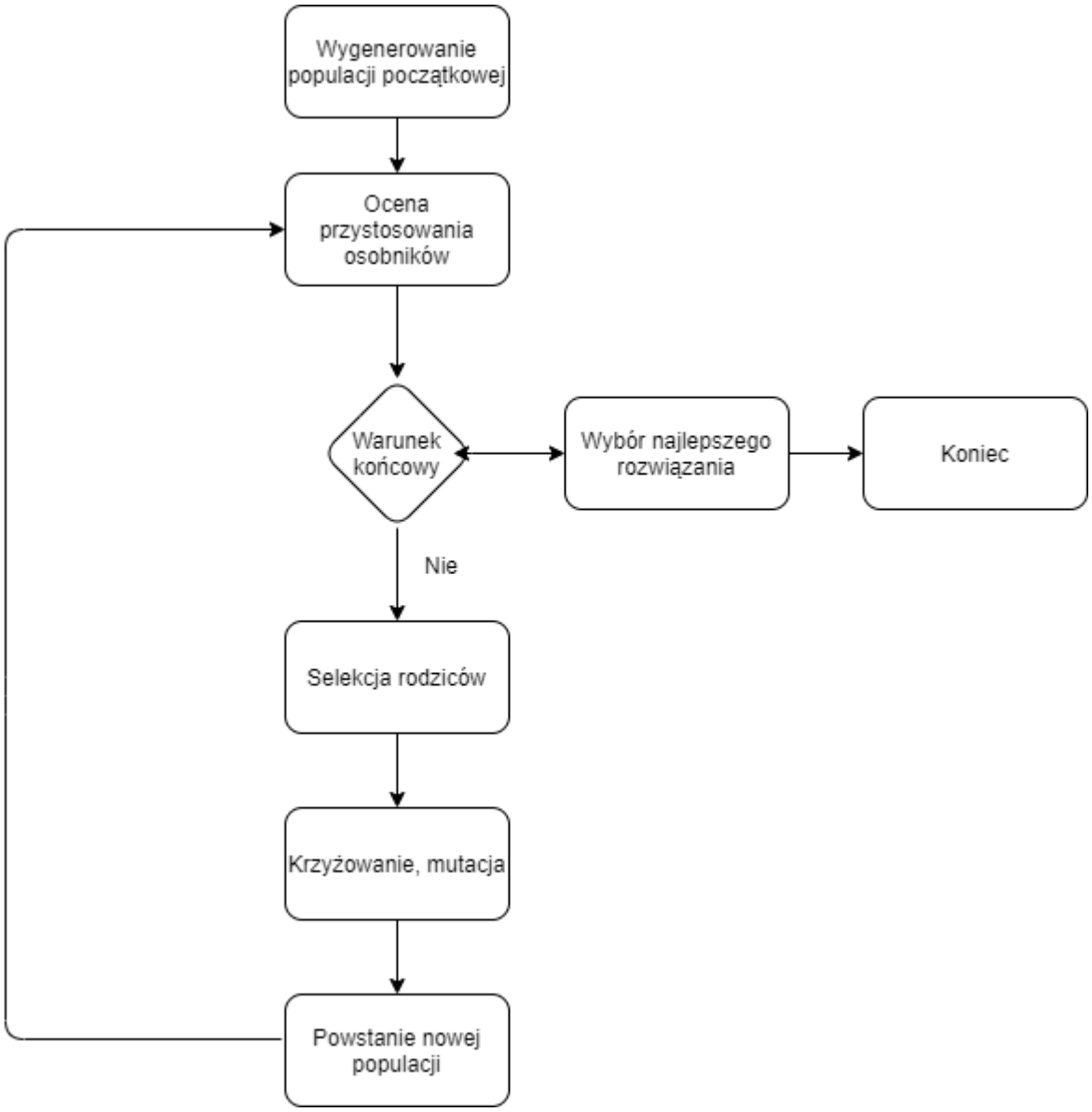
2.1 Wprowadzenie do algorytmu

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 cesze 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 algorytm 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



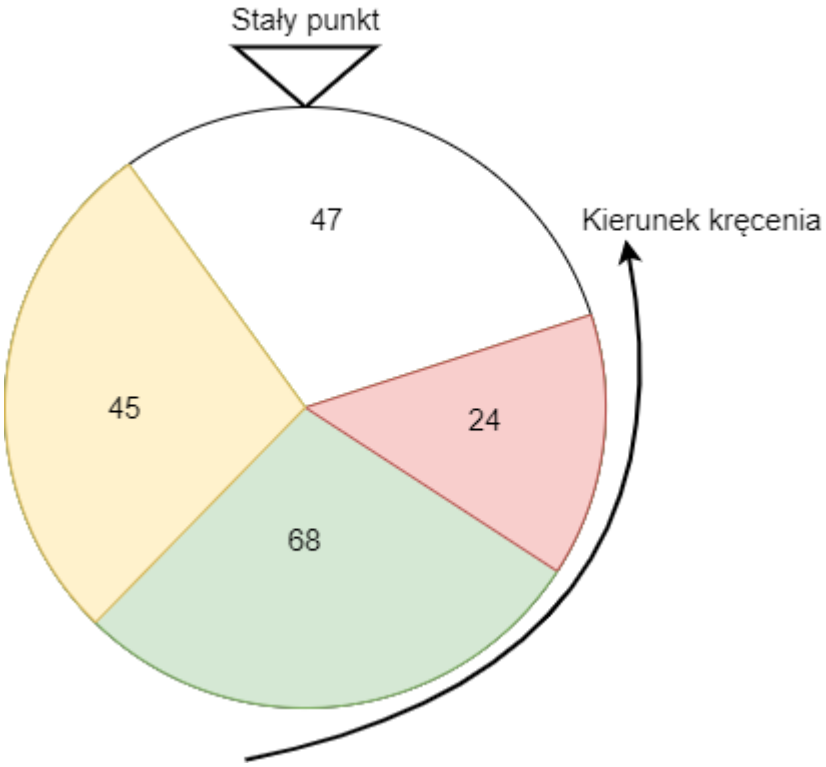
Rysunek 2.1: Schemat algorytmu genetycznego

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 szansę na pozostanie rodzicem i przekazanie swoich cech. [7]. Najpopularniejszymi metodami wyboru rodziców jest metoda ruletki oraz turniejowa.



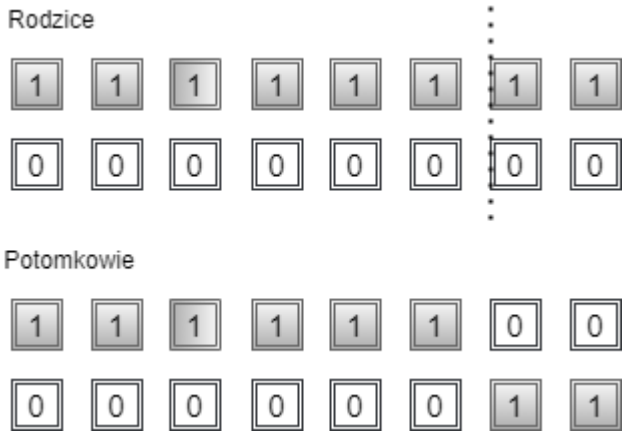
Rysunek 2.2: Metoda ruletki

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.

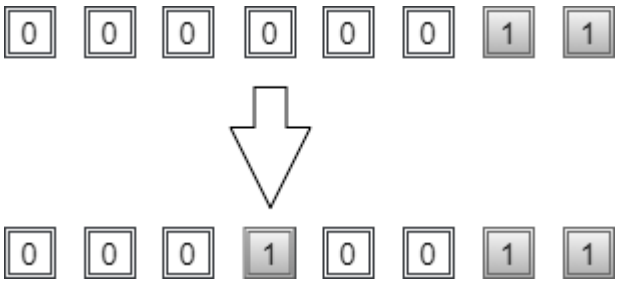
Wybrani rodzice zostają poddani operatorom genetycznym: krzyżowanie(ang. *crossover*) oraz mutacji(ang. *mutation*). Krzyżowanie polega na połączeniu części chromosomu jednego rodzica z częścią drugiego. Wynikiem takiego połączenia jest nowy osobnik.

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 rozładunkowy 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

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

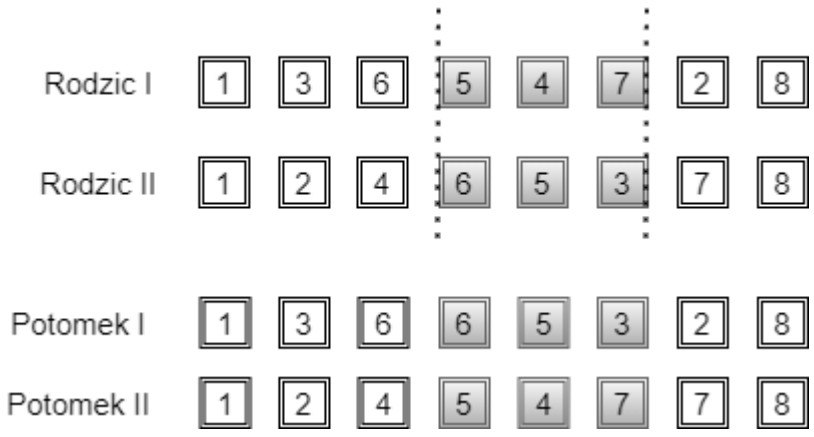
gdzie s - długość trasy, czyli suma odległości pomiędzy genami w chromosomie(1 - 2 - 3 - 4 - 5 - 6 - 7).

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 punk nie mogą się przemieścić.

2.3.1 Krzyżowanie z częściowym odwzorowaniem - PMX

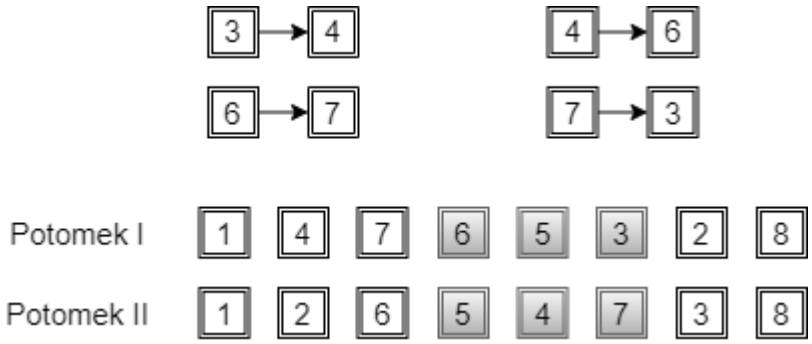
PMX[8] (ang. *Partially Mapped Crossover*) jest odmianą krzyżowania dwupunktowego, w którym powstaje dwójka potomstwa. Zakładając, że wyselekcjonowano dwóch rodziców: 1 - 3 - 6 - 5 - 4 - 7 - 2 - 8 oraz 1 - 2 - 4 - 6 - 5 - 3 - 7 - 8 (Rys. 2.6). Losowane są



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 zamienia 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 powtórzone geny, które znajdują się poza segmentami, zamienić na te geny, których brakuje. W tym celu następuje określenie mapowania pomiędzy genami, które następnie zamienia się ze sobą w chromosomach (Rys. 2.7). W pierwszym dziecku należy zamienić odpowiednie geny: 3 -> 4 oraz 6 -> 7. W drugim

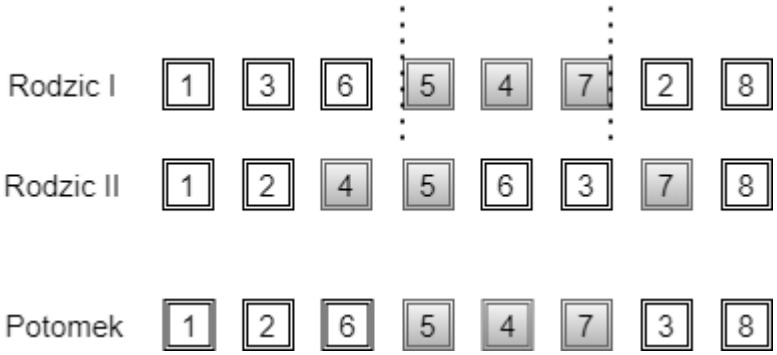


Rysunek 2.7: Krzyżowanie PMX - część 2

natomiast 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

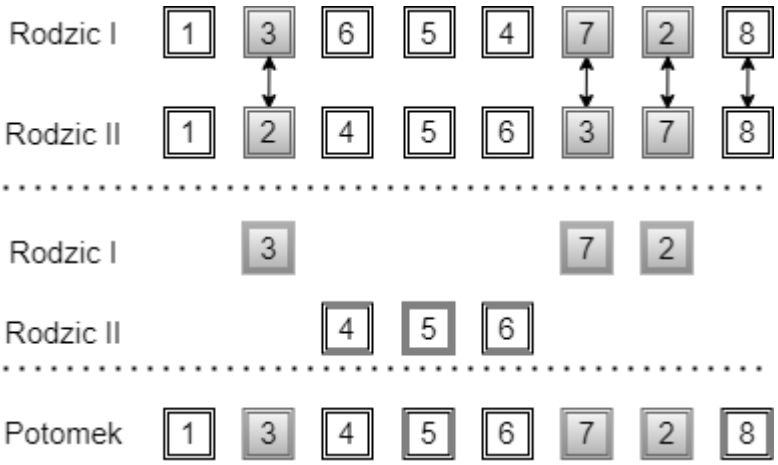
OX[12] (ang. *Order Crossover*) jest również odmianą krzyżowania dwupunktowego. W przeciwieństwie do PMX wynikiem będzie tylko jedno dziecko. Krzyżując ze sobą dwa chromosomy: 1 - 3 - 6 - 5 - 4 - 7 - 2 - 8 oraz 1 - 2 - 4 - 5 - 6 - 3 - 7 - 8. Pierwszym krokiem jest wylosowanie dwóch dowolnych punktów 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[11] (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. Należt 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,

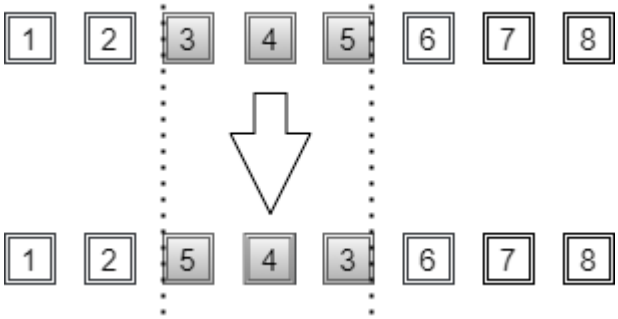
w X należy wpisać geny z drugiego rodzica, które nie wystąpiły w cyklu. W tak powstałym dziecku 1 - 3 - 4 - 5 - 6 - 7 - 2 - 8, wszystkie geny zajmują taką samą pozycję jak w którymś z rodziców, inaczej niż to było przy krzyżowaniu OX.

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. Załóżmy, 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.

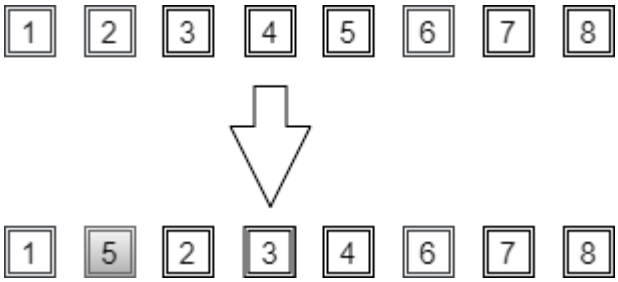


Rysunek 2.10: Mutacja odwracająca

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 rozwiązaniach

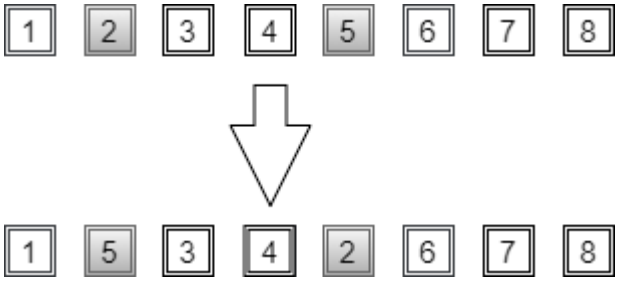
w całkowicie nowych przestrzeniach przeszukiwań. Punkt znajdujący się na końcu trasy, może znaleźć się na początku. Na rysunku 2.11 został wylosowany gen 5, który znajdował się na piątym miejscu, po mutacji znalazł się na drugim miejscu, w rezultacie chromosom po mutacji ma postać 1 - 5 - 2 - 3 - 4 -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

Słowo rozwój może być zestawiane z wieloma rzeczownikami. Wiele dziedzin ciągle się rozwija, powstają nowe udogodnienia które wpływają na wiele dziedzin życia. Dzięki rozwojowi techniki ludzie są w stanie osiągać cele które jakiś czas temu mogły być tylko marzeniami. Rozwój techniki również potrzebuje inspiracji do tworzenia nowych, lepszych rozwiązań

Szybkość oraz dokładność rozwoju zależy od wielu czynników. Dzięki pracy zespołowej pewne problemy mogą być rozwiązywane szybciej i dokładniej. Wysiłek włożony przez grupę procentuje szybko, a same efekty mogą być również satysfakcjonujące. Istotnym czynnikiem jest wysiłek wkładany przez każdego członka grupy.

Obserwując przyrodę możemy zauważyć 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 wiele gatunków zwierząt żyjących na ziemi przystosowało się do panujących tu warunków. Jedną z takich gatunków są mrówki. Te niewielkich rozmiarów owady posiadają zdolności pomagające im przetrwać wśród najcięższych warunków. Mrówki żyją w stadach w związku z czym wykorzystują pracę zespołową do rozwiązywania problemów jakie codziennie napotykają na swojej drodze.

Aby zapewnić przetrwanie stadu 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 przez nas rozwiązań do wyznaczania zoptymalizowanej trasy jest algorytm mrówkowy, inaczej nazywany ACO - Ant Colony Optimization. Pomysł ten został zaczerpnięty z natury. Jak sama nazwa wskazuje działanie algorytmy jest związane z

mrówkami, a dokładnie z kolonią mrówek. Pomysł na algorytm został zaproponowany na początku lat 90 XX wieku przez włoskiego badacza - Marco Dorigo.

Tak jak zostało wspomniane wcześniej, algorytm opiera się na pracy mrówek. Chodzi tutaj dokładnie o trasę jaką mrówki pokonują od swojego siedliska do miejsca w którym znajduje się pożywienie. Ważne jest znaczenie tutaj pracy zespołowej. Trasę kształtuje cała kolonia mrówek, a nie pojedyncze przypadki. Mrówki z każdą kolejną podróżą wykształcają coraz to bardziej optymalną trasę.

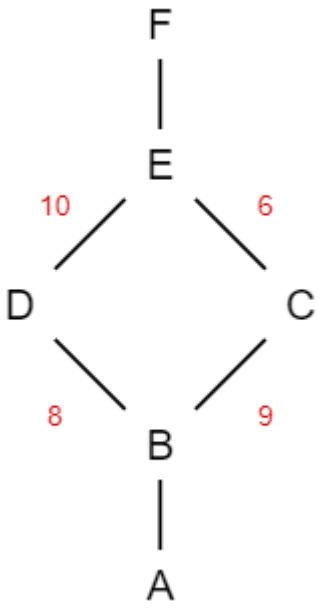
3.2 Opis działania algorytmu

Mrówka w celu znalezienia pokarmu w sposób losowy wyrusza z mrowiska. 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 mrówki wyruszają do miejsca spoczynku pożywienia. Chcąc uniknąć sytuacji w której zdobycz może zostać zabrana przez inne owady, mrówki muszą jak najbardziej zoptymalizować trasę jaką mają do pokonania.

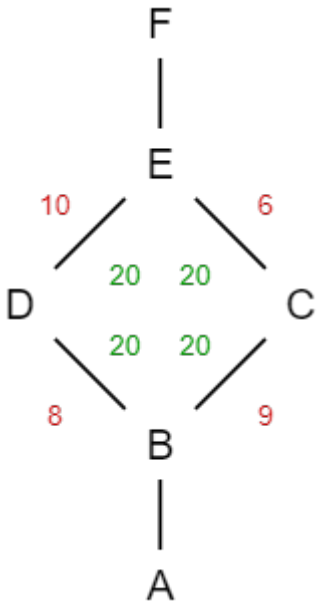
Mimo posiadania informacje o znalezionym pokarmie, każda mrówka sama musi zlokalizować źródło. Czy mrówki poruszają się tą samą trasą przy każdym wyjściu z mrowiska? Aby trafić do miejsca w którym znajduje się pokarm, wspomniane owady wykorzystują ślady pozostawione przez osobników które już natrafiły na pożywienie. W ten sposób mrówka która wyrusza w sposób losowy, 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 wagi. Na początku załóżmy, że mamy do dyspozycji 80 agentów. Przez N pierwszych iteracji mrówki poruszały się losowo i powstał następujący podział:

Tak jak można to zauważyć na grafice 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 najoptymalniejszej ścieżki.

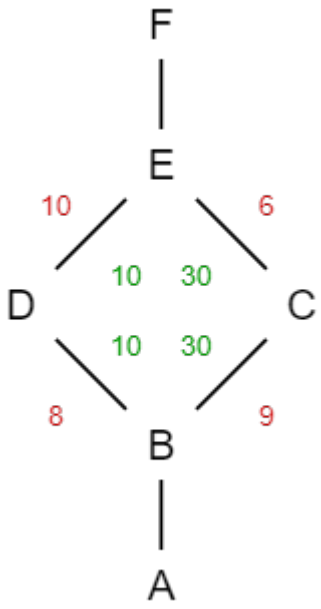


Rysunek 3.1: Położenie ścieżek na przykładowym grafie



Rysunek 3.2: Rozkład agentów na grafie po N początkowych iteracjach

Liczba agentów odwiedzających ścieżki zmienia się. Bardziej optymalna trasa zyskuje widoczną przewagę. W kolejnych iteracjach mrówki wykorzystują siłę feromonów. Bardziej optymalne ścieżki są częściej odwiedzane w związku z czym zapach na tych krawędziach jest silniejszy oraz podtrzymywany. Na mniej optymalnych trasach zapach zanika i przestają one być atrakcyjne dla agentów. Opisana sytuacja prowadzi do wyznaczenia trasy która jest najatrakcyjniejsza do przebycia dla mrówek.



Rysunek 3.3: Rozkład agentów w grafie po optymalizacji

3.3 Feromony

Feromony posiadają cechę która może się wydawać, że negatywnie wpływa na wyznaczanie ścieżki. Chodzi tutaj o ulatnianie się zostawionego zapachu. Na pierwszy rzut oka może się to zjawisko wydawać niepożądanym, ale w rzeczywistości ma duży wpływa na optymalizację. Jeśli feromony nie straciłyby na swojej sile, to bardzo prawdopodobne, że pierwotna ścieżka mogłaby zostać uznana za najbardziej optymalną.

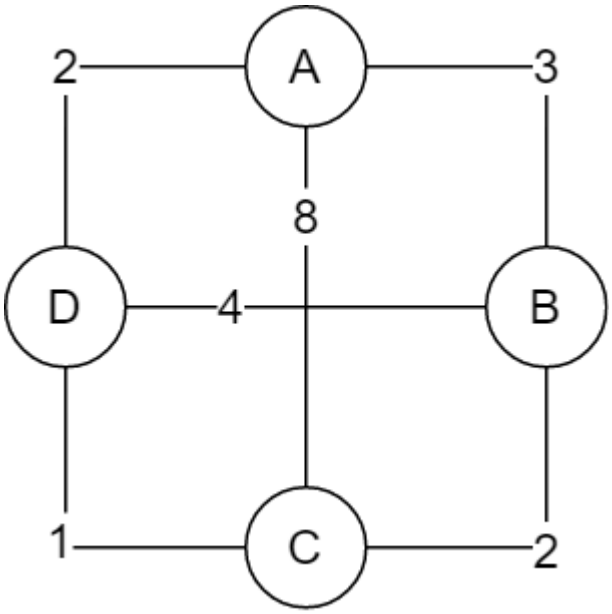
W jaki sposób wyznaczona zostaje najbardziej optymalna ścieżka? Zapach feromonów jest podtrzymywany przez wędrujące mrówki. Z czasem owady te same zbaczają z drogi w celu poszukiwania alternatywnej trasy. Jeśli wybrana trasa jest optymalniejsza od pozostałych to ślad jest podtrzymywany, a na innych zanika. Dzięki temu w sposób iteracyjny można wyróżnić trasę najoptymalniejszą, 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 zaletę, a nie wadę. 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 3.1 przedstawione są przykładowe odległości.

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



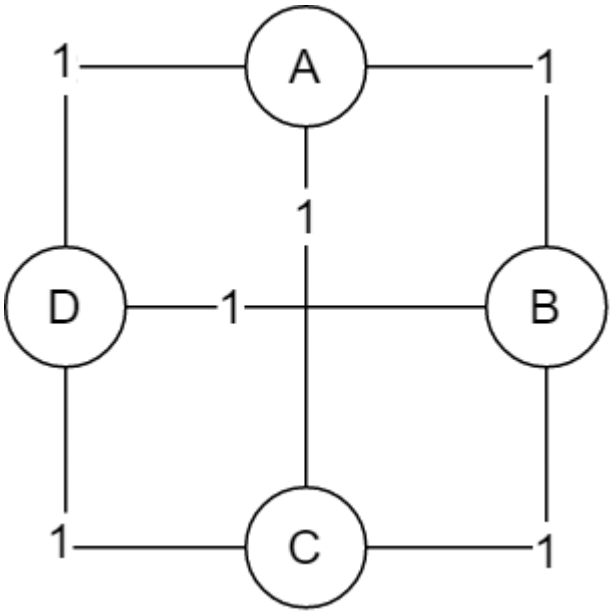
Rysunek 3.4: Graf z wagami

Równie ważne jest wyznaczenie początkowych współczynników feromonów. Na początku nadajmy wszystkim krawędziom w grafie wartości równe 1, a współczynnik parowania niech wynosi 0.5.

Tabela 3.2: Wartości feromonów

	A	B	C	D
A	0	1	1	1
B	1	0	1	1
C	1	1	0	1
D	1	1	1	0

Posiadając początkowe dane wyznaczmy w sposób losowy trasy dla dwóch agentów: L1 i L2. Agent L1 poruszał się trasą w której odwiedził wierzchołki w następującej kolejności: A, B, C, D, A.



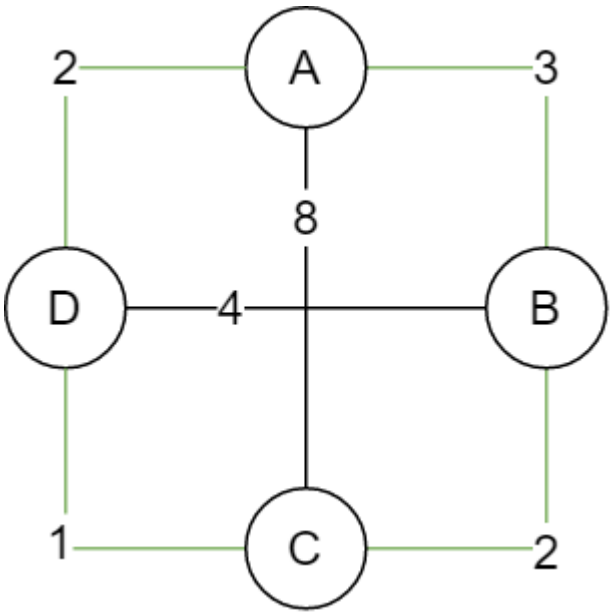
Rysunek 3.5: Graf z początkowymi wartościami feromonów

Agent L2 wyznaczył następującą trasę: A, C, B, D, A.

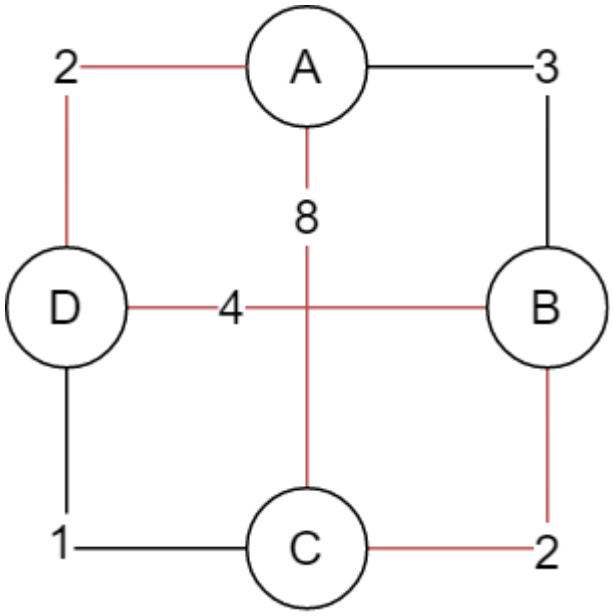
Mrówki odpowiednio pokonały dystans 8 i 16 punktów. Dzięki tej informacji można zaktualizować wartości feromonów na poszczególnych krawędziach. Do obliczeń wykorzystany jest wzór: $\tau_{\mathrm{i,j}} = (1-p)\tau_{\mathrm{i,j}} + \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}$. Krawędzie A-B i C-D zostały odwiedzona 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.

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. Według wzoru $p_{\mathrm{i,j}} = \frac{(\tau_{\mathrm{i,j}})^{\alpha}(\eta_{\mathrm{i,j}})^{\beta}}{\sum(\tau_{\mathrm{i,j}})^{\alpha}(\eta_{\mathrm{i,j}})^{\beta})}$ wyliczane jest prawdopodobieństwo dla wszystkich możliwości. Przejście z krawędzi B do krawędzi A wynosi około 20%, do krawędzi D 30%, a do krawędzi C około 50%.

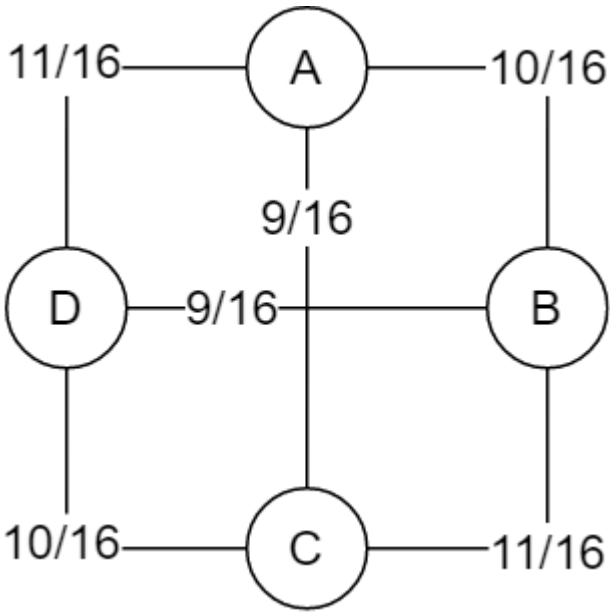
Optymalna trasa zostanie wykształcona po wykonaniu wielu iteracji. Ilość iteracji nie jest zdefiniowana i dla każdego przypadku może być różna. Sytuacja wygląda identycznie w przypadku wyboru wartości p. Ważne jest natomiast to, aby w trakcie działania algorytmu nie modyfikować tej wartości. Powinno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



Rysunek 3.6: Trasa przebyta przez agenta L1



Rysunek 3.7: Trasa przebyta przez agenta L2



Rysunek 3.8: Graf z wartościami feromonów po modyfikacji

4. Algorytmy zachłanne

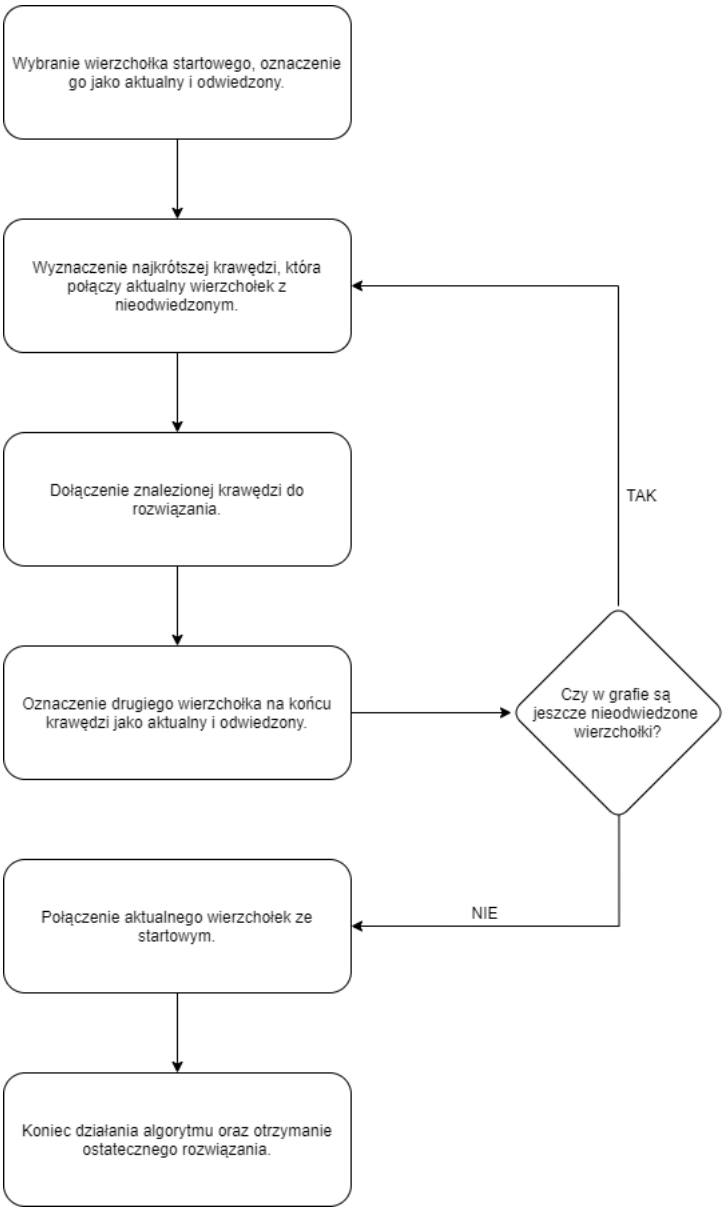
Istnieje wiele algorytmów zachłannych pozwalających otrzymać optymalne rozwiązanie dla problemu znalezienia najkrótszej trasy. W poniższym rozdziale opisane dokładniej zostało działanie algorytmu najbliższego sąsiada, algorytmu najmniejszej krawędzi oraz algorytmu a*. W celu zoptymalizowania otrzymanych rozwiązań zastosowane zostaną operatory 2-opt oraz 3-opt.

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 znajdującego się najbliżej poprzednio odwiedzonego.

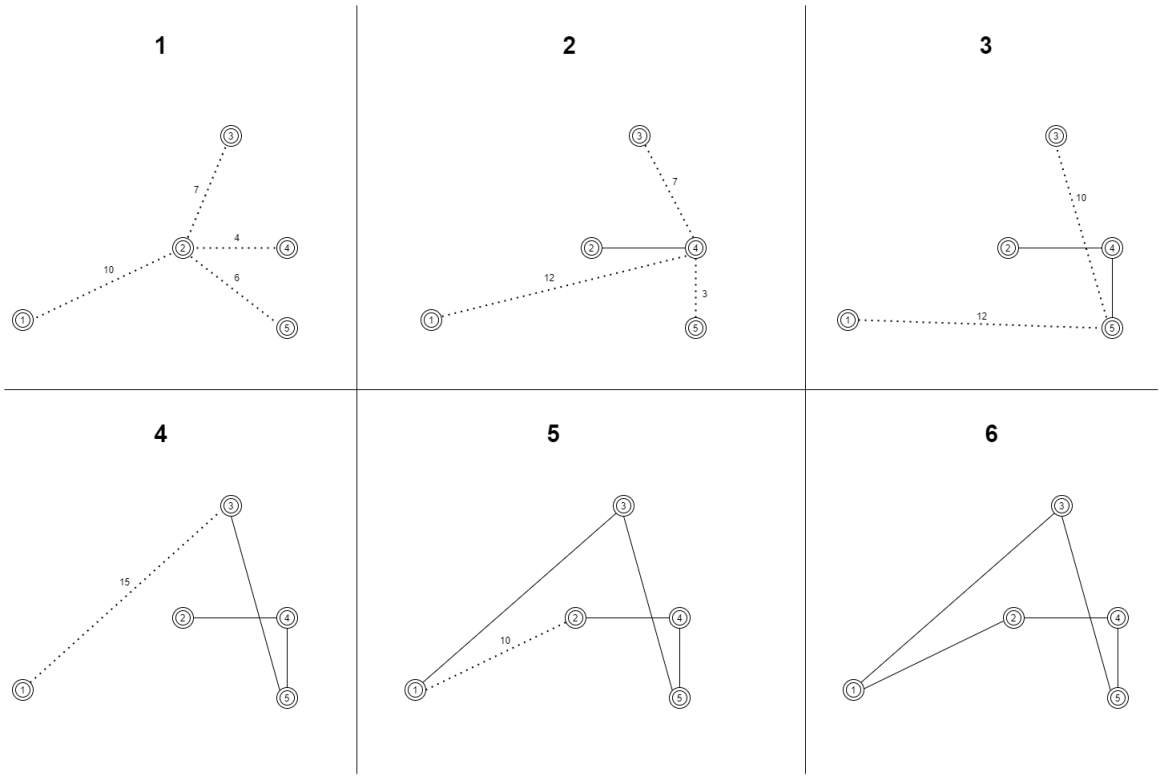
Na rysunku 4.1 został pokazany schemat blokowy algorytmu najbliższego sąsiada. Pierwszym krokiem jest wyznaczenie wyznaczenie wierzchołka startowego. Następnie dla aktualnego wierzchołka należy obliczyć najkrótszą krawędź łączącą aktualny wierzchołek spośród kolekcji wierzchołków nieodwiedzonych. Wybór 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 5.1 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 wybrana 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.

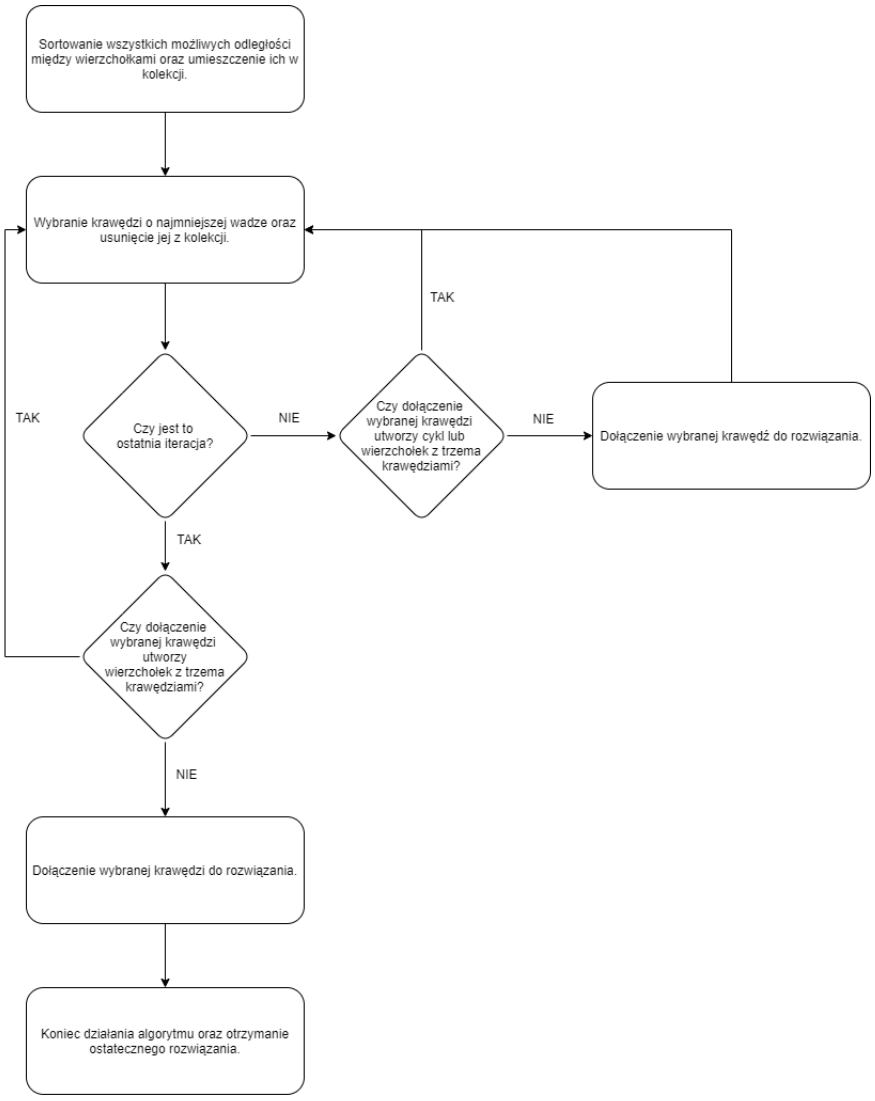


Rysunek 4.2: Algorytm najbliższego sąsiada

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

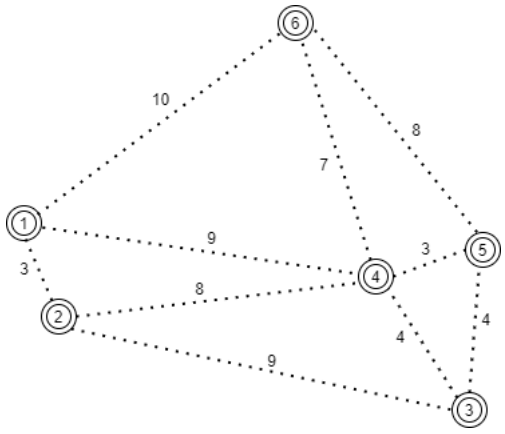


Rysunek 4.3: Schemat algorytmu najmniejszej krawędzi

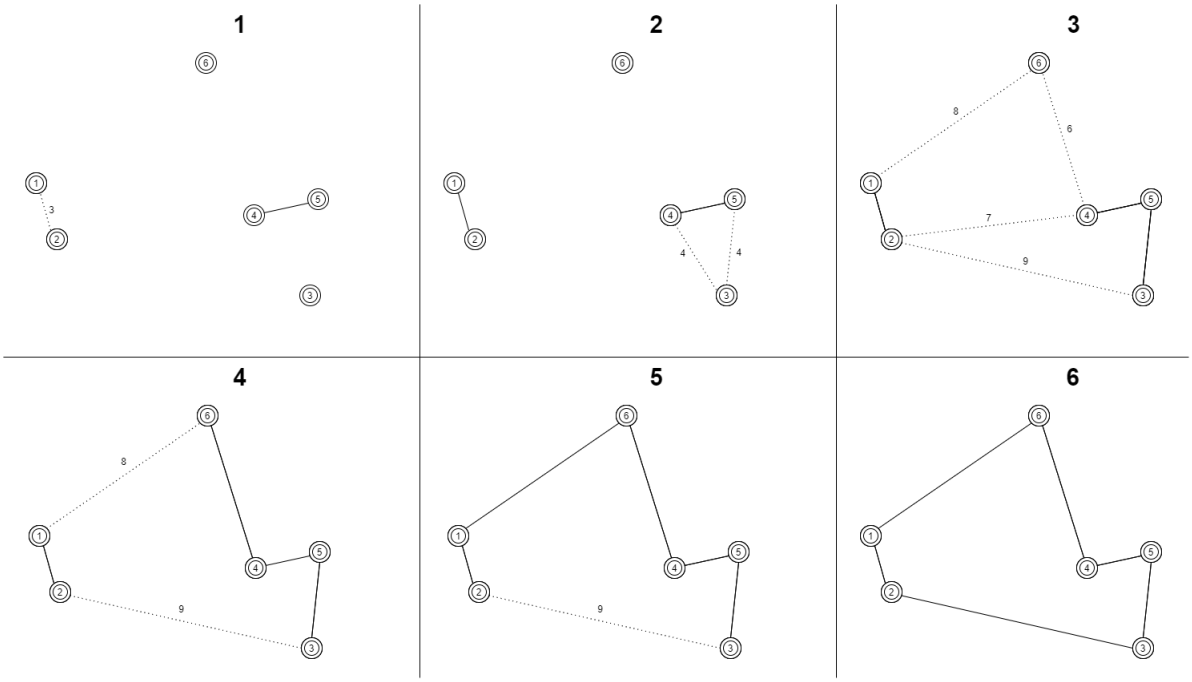
warunku z utworzeniem cyklu. Po zakończeniu działania algorytmu otrzymano optymalne rozwiązanie dla algorytmu najmniejszej krawędzi.

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 rozpocząć 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 cyklu. W przypadku drugiej iteracji algorytm wybiera krawędź o najmniejszej wadze, 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 sytuacje, 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 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. Dla struktur drzewiastych algorytm tworzy najlepsze rozwiązania. Główną zasadą algorytmu a* jest minimalizacja funkcji kosztu oraz funkcji heurystycznej, gdzie ta ostatnia 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 iloczyn liczby krawędzi do wybrania i najmniejsza wartość krawędzi. Zapewnia to w ten sposób, że taka funkcja jest nadmiernie optymistyczna.

<po badaniach powstanie tutaj schemat blokowy oraz przykład, tak jak w poprzednich algorytmach, ponieważ algorytm jest jeszcze dopracowywany>

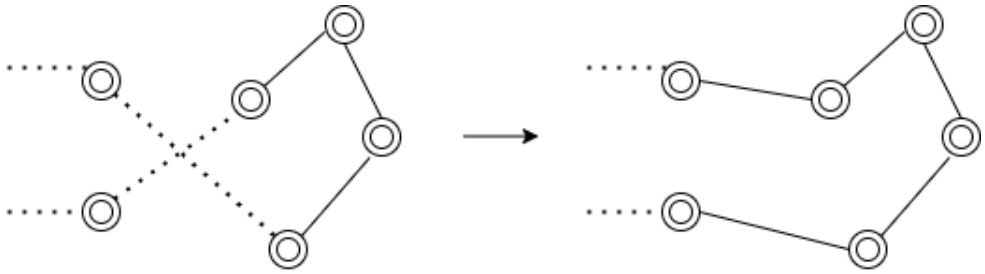
4.4 Optymalizacja otrzymanych rozwiązań

W ostatnim podrozdziale skupimy się na sposobach optymalizacji otrzymanych przez algorytmy zachłanne rozwiązań. Dokładniej omówiona zostanie metoda 2-opt oraz metoda 3-opt.

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

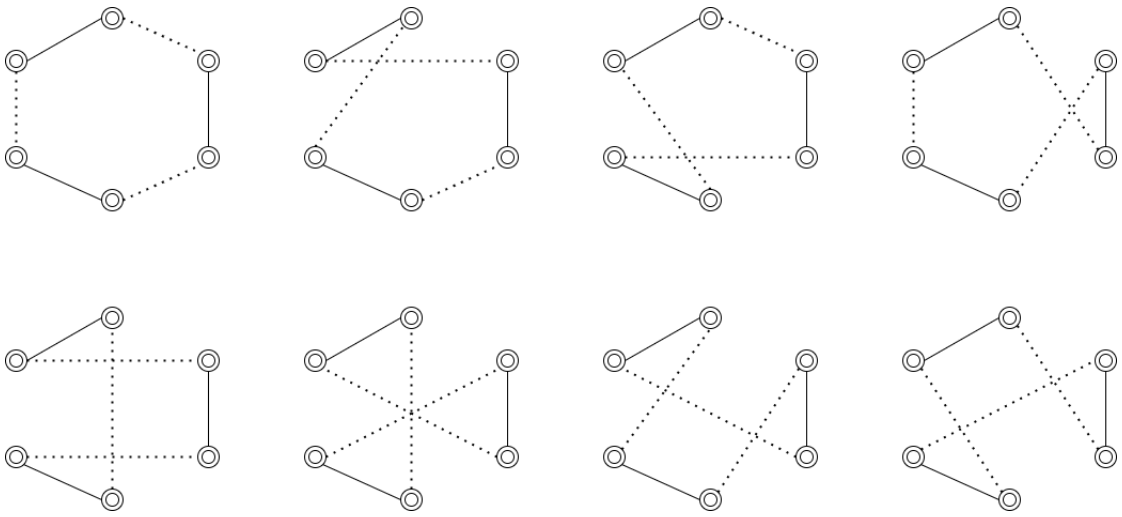
należy modyfikować rozwiązania. Algorytm można wykonywać wielokrotnie, w ten sposób zostanie zrealizowane minimum lokalne. Dla lepszego przedstawienia działania metody 2-opt, należy spojrzeć na rysunek 5.4.



Rysunek 4.6: Metoda 2-opt

4.4.2 Metoda 3-opt

W odróżnieniu do algorytmu 2-opt, w metodzie 3-opt rozważane są wszystkie cykle, które można uzyskać wymieniając trzy krawędzie z cyklu aktualnego, przy czym jest możliwe wykonanie tego na wiele sposobów tak jak zostało to przedstawione na rysunku 5.5.



Rysunek 4.7: Metoda 3-opt

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.

Analizując zbiory danych można zauważyć, 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.

Do obliczeń odległości pomiędzy kontenerami 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.

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. Kolorem jasno-żółtym są oznaczone wartości które są mniejsze od oryginalnej trasy. Kolorem żółtym zostały oznaczone pole które są najlepsze dla przedstawionych parametrów. Najlepszy wynik został oznaczony kolorem zielonym.

Algorytm genetyczny został zbadany dla trzech różnych wartości populacji N: 100,1000 oraz 2000; czterech różnych wartości iteracji (pokoleń): 100, 1000, 2000 oraz 4000. Dla każdej wariacji tych parametrów zostało zbadane krzyżowania PMX, OX oraz CX, z różnymi kombinacjami mutacji: wstawiająca - insert, zamieniająca - swap oraz odwracająca - revert. Współczynnik mutacji dla wszystkich uruchomień wynosi 0.1. Dla każdej wariacji parametrów wejściowych algorytm został uruchomiony dziesięć razy. W tabelach dotyczących wyników algorytmu genetycznego, zostały przedstawione agregowane dane.

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 wybrane 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 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.

6.1 Trasa I

Pierwsza trasa składa się ze 111 punktów, odwiedzonych przez pojazd. Długość oryginalna tej trasy wynosi 48897 metrów. W kolejnych trzech podrozdziałach zostaną przedstawione wyniki optymalizacji każdego z algorytmów, a w czwartym podrozdziale te algorytmy zostaną porównane ze sobą.

6.1.1 Wyniki algorytmu genetycznego - Paweł

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 został zoptymalizowany średni wynik. Dla N = 1000 zoptymalizowanych wyników jest już więcej. Gdzie zawsze najlepszy wynik w populacji został osiągnięty dla krzyżowania CX i mutacji $Insert$. Dla tej populacji udało się zoptymalizować trasę do 33843 metrów przy 2000 iteracjach. Dla N = 2000, k = 2000 oraz krzyżowaniu CX i mutacji $Insert$ został osiągnięty najlepszy wynik globalny 32326 metrów. Widać również, 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 $Revert$.

Tabela 6.1: Trasa I - średnie 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	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

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. Udało się to osiągnąć od populacji 1000 osobników przy 1000 iteracjach. Krzyżowanie PMX oraz CX osiągnęło 100% skuteczności w połączniu z każdą mutacji dla N = 2000 oraz k = 2000 i k = 4000. Krzyżowanie OX z mutacją $Revert$ ani razu nie osiągnęła 100%.

Tabela 6.2: Trasa I - skuteczność 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	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 danych parametrów wejściowych. Dla N = 100 w przypadku dwóch krzyżowań PMX, CX oraz mutacji $Insert$ zostały osiągnięte wyniki lepsze od oryginału, nie udało się tego osiągnąć jedynie dla k = 100. Dla tej populacji najlepszy wynik został znaleziony dla krzyżowania PMX oraz mutacji $Insert$ i wyniósł 40964 metrów. Dla N = 1000 już najlepiej 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 = 200, k = 100, krzyżowaniu CX oraz mutacji $Insert$ i wyniósł 30939 metrów. Od N = 100 i k = 4000 zawsze najlepszy wynik wystąpił dla krzyżowania CX i mutacji $Insert$.

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

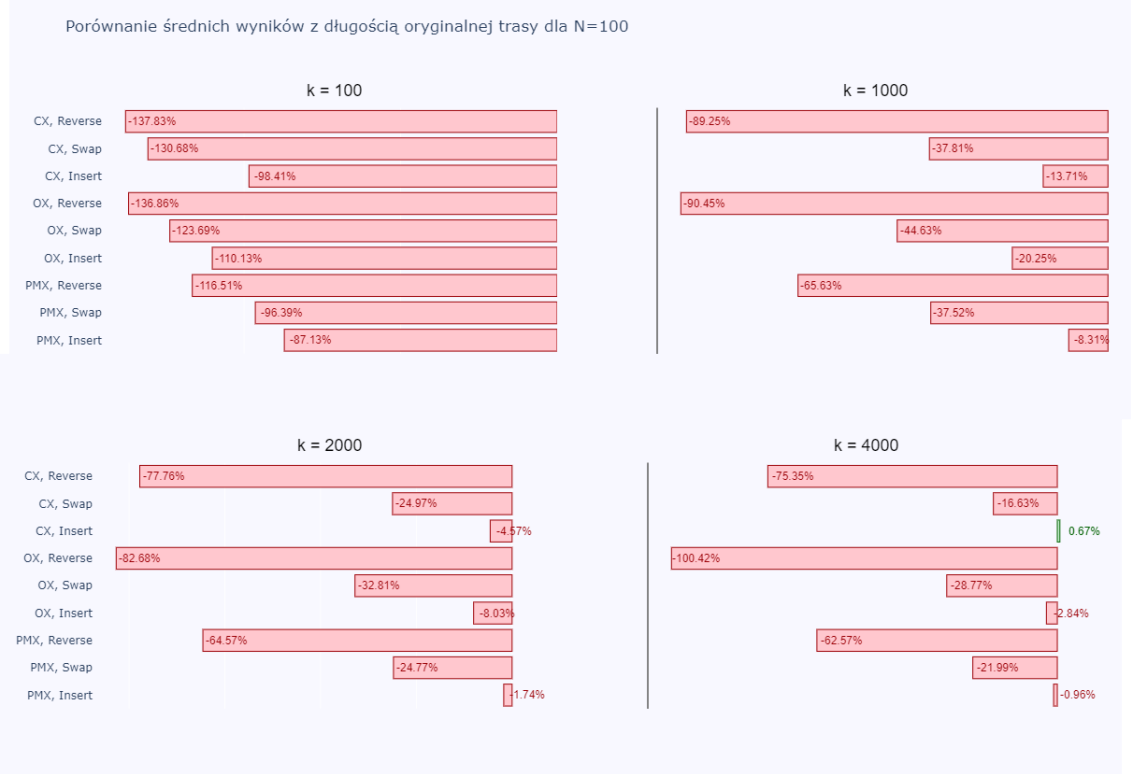
N	k	PMX Insert	PMX Swap	PMX Revert	OX Insert	OX Swap	OX Revert	CX Insert	CX Swap	CX Revert
100	100	80385	87226	81154	90764	99185	101137	72268	103408	92859
100	1000	45076	62481	67443	53948	60699	78382	45134	56918	79955
100	2000	40964	53316	70100	45500	55396	80719	43142	54430	79636
100	4000	44853	52397	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

W tabeli 6.4 zostały przedstawione średnie czasy wywołań. Tylko w dwóch przypadkach najszybsze nie było krzyżowanie CX oraz mutacja $Revert$. Najlepszy średni wynik optymalizacji pomiędzy populacjami 1000 osobników, a 2000 osobników poprawił się o około 1500 metrów (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.

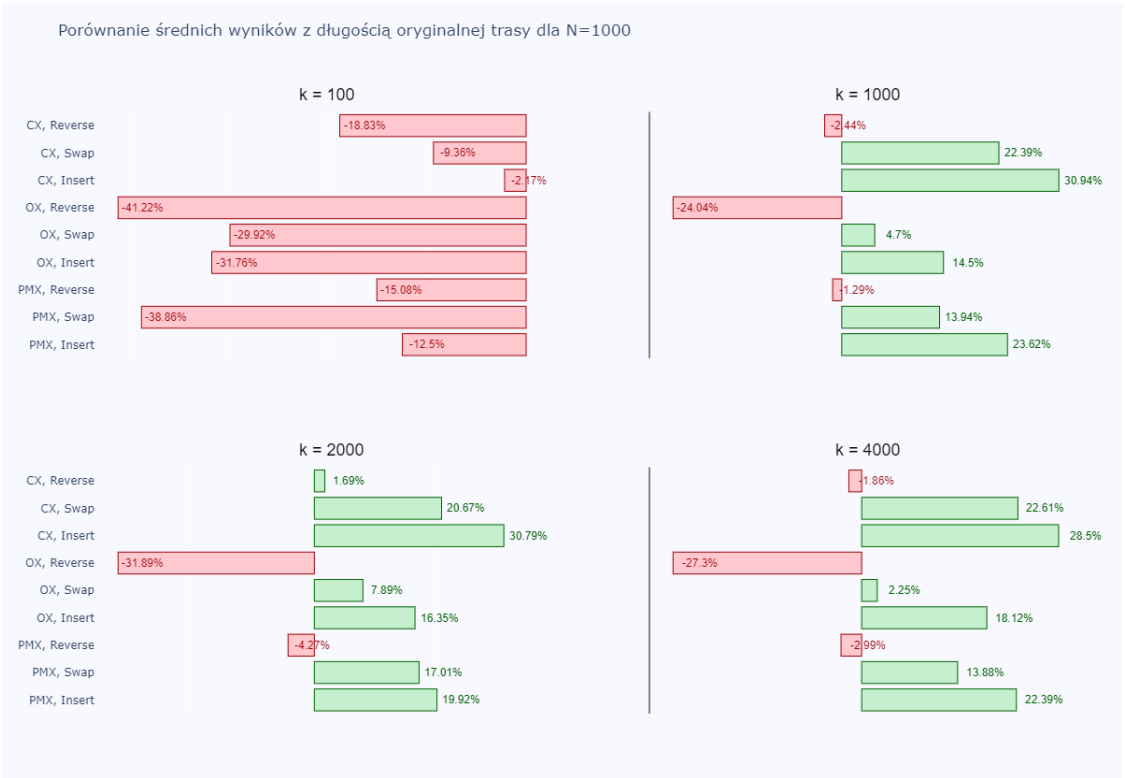
Tabela 6.4: Trasa I - średnie czasy wykonywania algorytmu dla danych parametrów wejściowych

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	15,26

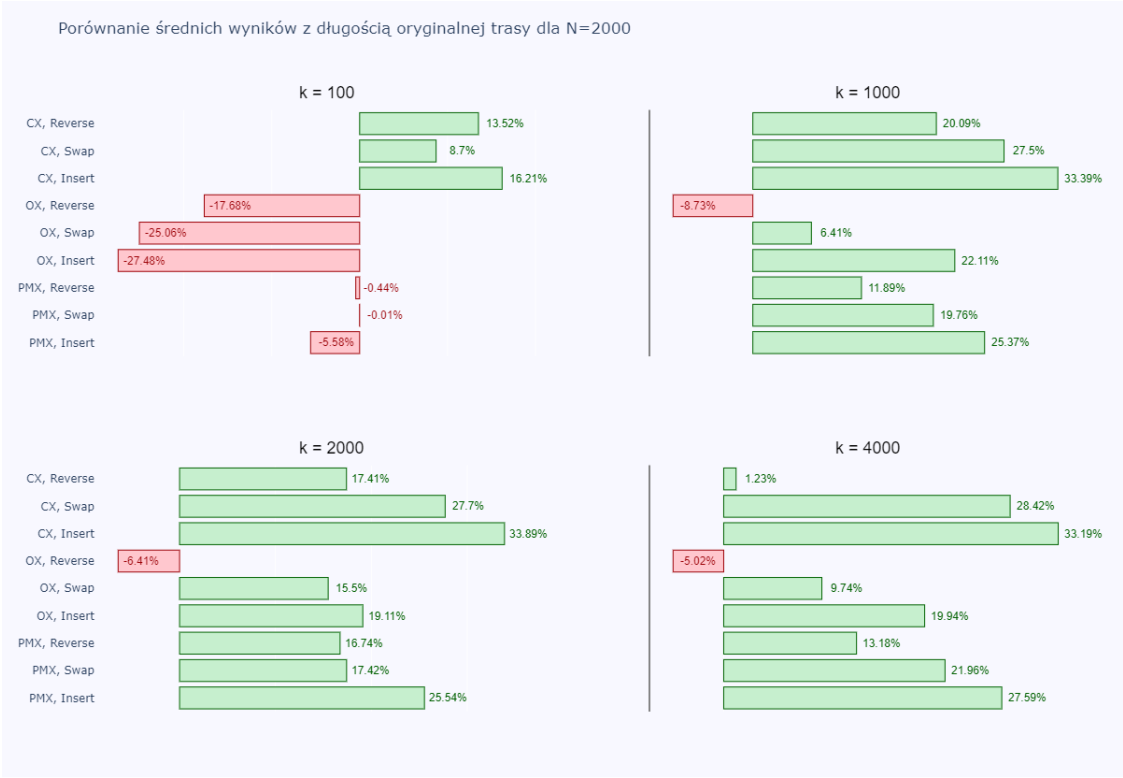
Rysunki 6.1, 6.2 oraz 6.3 przedstawiają wykresy, które porównują o ile różni się rozwiązanie od długości trasy oryginalnej, kolejno dla N = 100, N = 1000 oraz N = 2000. Dla populacji 1000 osobników (Rys. 6.1) w jednym przypadku udało się zoptymalizować trasę o 0.67%, co jest znikomą wartością. W pozostałych przypadkach wyniki są gorsze o kilkanaście, a nawet kilkadziesiąt procent. Dla N = 1000 (Rys. 6.2) oraz k = 100, wszystkie kombinacje krzyżowań i mutacji generowały wyniki gorsze o kilkadziesiąt procent. Udało się zoptymalizować trasę powyżej 25% dla krzyżowania CX dla kolejnych wartości k. Krzyżowanie PMX optymalizowało trasę w zależności od parametru k oraz mutacji od 13 do 23 procent. Natomiast od N = 2000 (Rys. 6.3) oraz k = 100, każda wariacja parametrów optymalizowała rozwiązanie o kilkanaście procent, z wyjątkiem krzyżowania OX i mutacji $Revert$. Krzyżowanie CX z mutacją $Insert$ dla k = 1000, 2000 oraz 4000 optymalizowało rozwiązanie średnio o 33%. Czas pomiędzy tymi iteracjami wzrastał już dwukrotnie (Tab. 6.4).



Rysunek 6.1: Trasa I - porównanie średnich wyników z długością oryginalnej trasy dla N=100



Rysunek 6.2: Trasa I - porównanie średnich wyników z długością oryginalnej trasy dla N=1000



Rysunek 6.3: Trasa I - porównanie średnich wyników z długością oryginalnej trasy dla N=2000

6.1.2 Wyniki algorytmu mrówkowego

Algorytm został uruchomiony dziesięciokrotne dla każdej możliwej wariacji 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.

Najlepsze wyniki algorytmu mrówkowego dla trasy numer 1 zostały przedstawione w tabeli **??**. Spośród wszystkich wariacji jedynie dwa wyniki zostały sklasyfikowane jako gorsze od oryginalnej odległości. Te najgorsze wyniki wystąpiły dla najniższej

wartości iteracji. Najlepsza wartość jaką udało się osiągnąć to 38111. Jest to wartość jaka została osiągnięta dla 75 ilości iteracji oraz dla współczynnika parowania wynoszącego 0.2. Wartości lepsze od oryginalnych wystąpiły dla około 93% możliwych kombinacji parametrów wejściowych.

Tabela 6.5: Trasa I - najlepsze wyniki algorytmu mrówkowego dla danych parametrów wejściowych

k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
50	57946	40150	39964	38773	47140	45598	57402	45451	39443
75	44971	38111	38650	39966	41654	41462	43910	44911	43740
100	39659	40036	40583	39383	41258	43156	43066	41932	41156

Średnia wartość najlepszych wyników dla współczynnika k równego 50 wynosi 45 763. Średnia dla ilości iteracji równej 75 jest wyższa i jest równa 41 931. Ostatnia próba daje średnią na poziomie 41 167. Różnica pomiędzy 50 iteracjami a 75 wynosi 3 832, a różnica między 75 iteracjami a 100 jest już znacznie mniejsza i wynosi 764. Wyliczając średnią wartość dla wszystkich współczynników parowania można zauważyć, że najlepsze wyniki przypadają dla wartości od 0.2 do 0.4. Średnia wyników jest wtedy mniejsza od 40 000.

W tabeli numer **??** 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. Wartości współczynnika równe i wyższe od 0.5 dla każdej ilości iteracji są gorsze od wartości oryginalnej. Najlepsza średnia pomiarów wynosi 41 440. Wystąpiła ona dla k równego 75 oraz współczynnika parowania równego 0.2.

Tabela 6.6: Trasa I - średnie wyniki algorytmu mrówkowego dla danych parametrów wejściowych

k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
50	63984	43885	43058	42144	56972	58786	60393	55217	53567
75	48456	41440	46490	47186	52814	59712	55389	57640	56455
100	43634	42900	47014	41756	54957	55704	59148	53122	54116

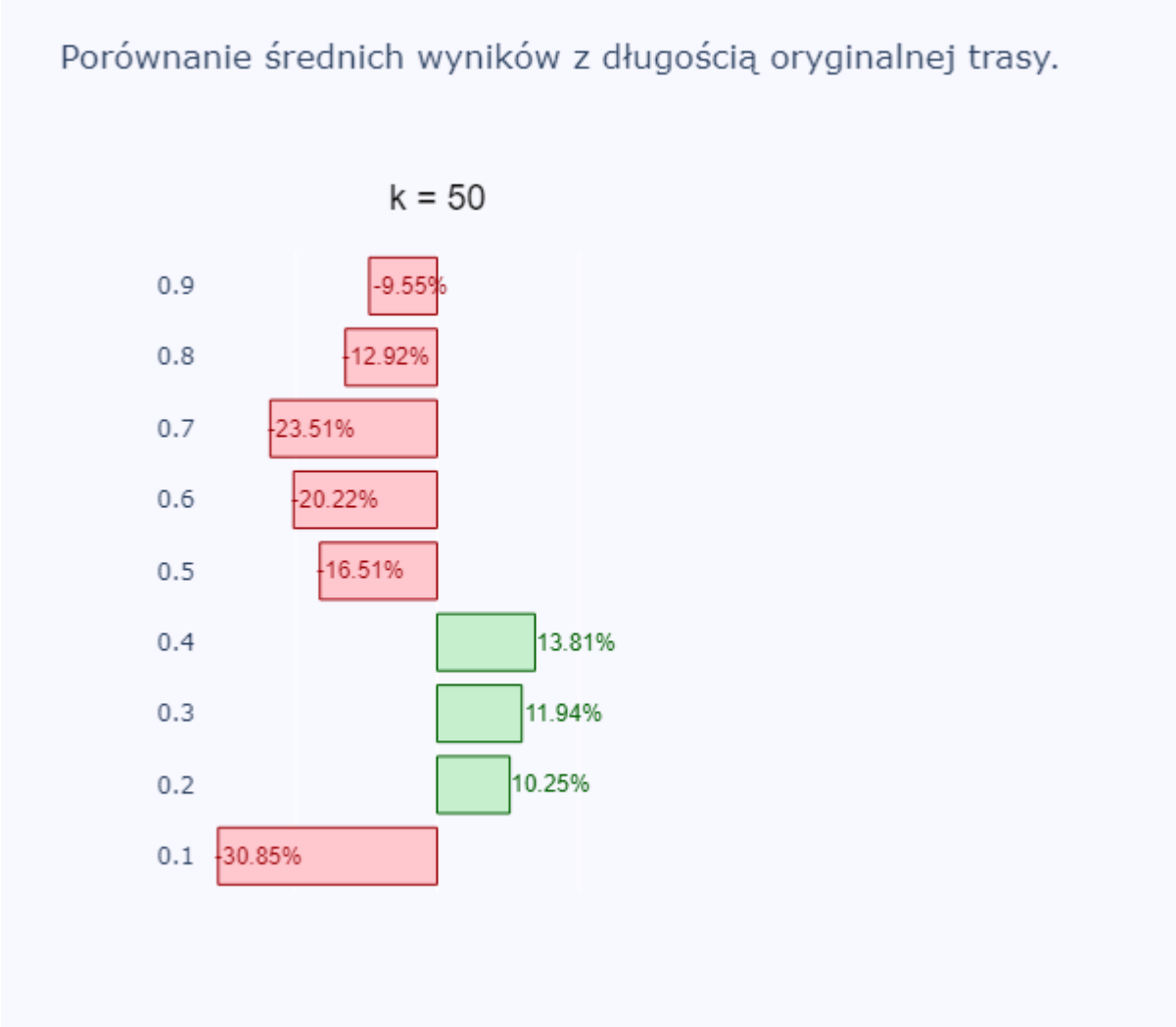
Istotnym parametrem jaki należy wziąć pod uwagę jest to, jak często algorytm jest w stanie poprawić oryginalną trasę. W tym celu została stworzona tabela numer **??**. Możemy zauważyć z jaką skutecznością program jest w stanie poprawić trasę. Wyniki potwierdzają wcześniejsze wnioski. Najlepsze wyniki można zestawić ze współczynnikiem parowania

feromonów z zakresu od 0.1 do 0.4. Co więcej w zależności od ilości iteracji dla każdego ze współczynników trasa została polepszona.

Tabela 6.7: Trasa I - suma lepszych tras w porównaniu do oryginalnej

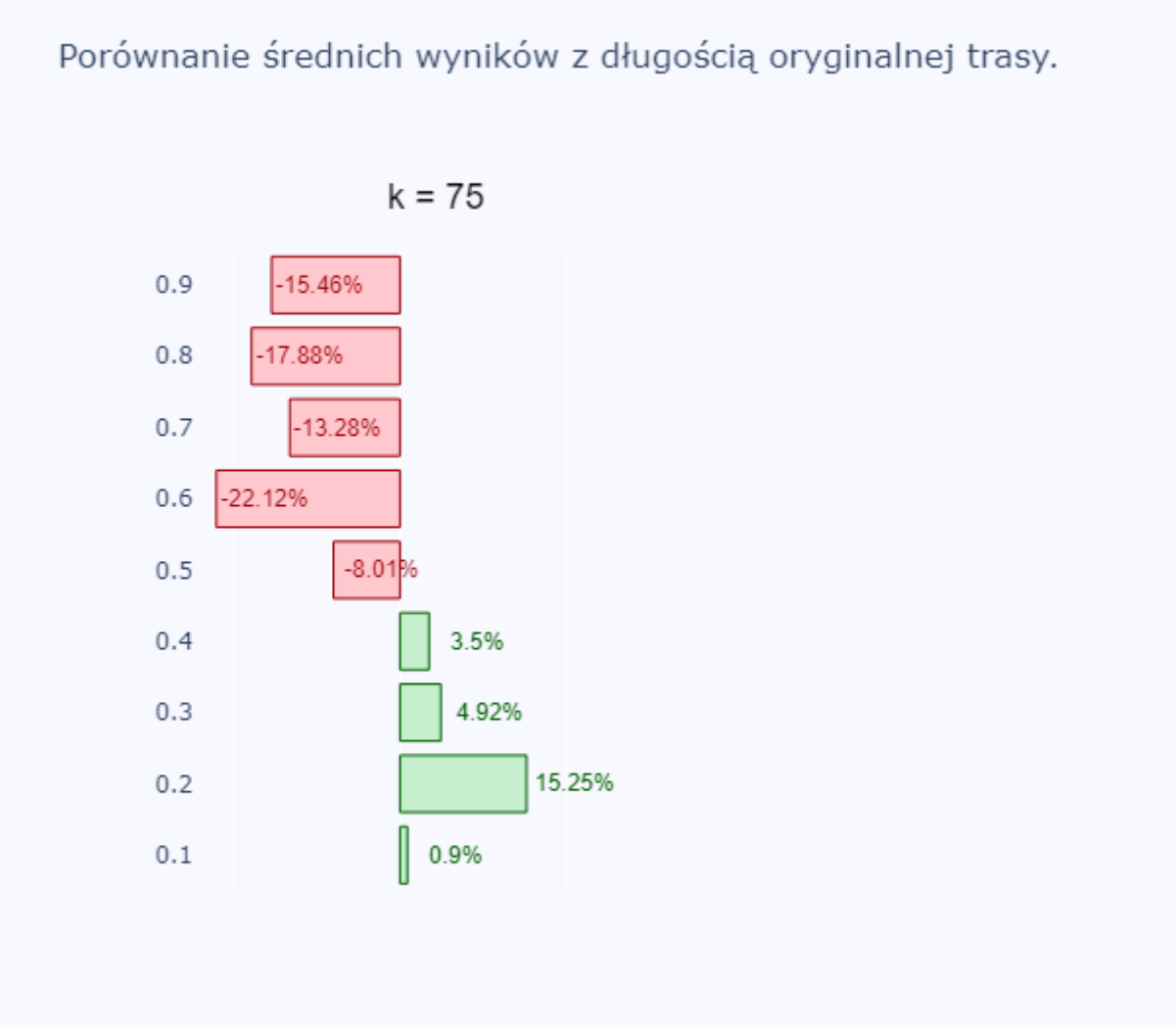
k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
50	0%	100%	100%	100%	20%	10%	0%	40%	30%
75	60%	100%	80%	80%	50%	40%	20%	10%	30%
100	100%	100%	70%	100%	30%	30%	10%	50%	40%

Na wykresie 6.6 przedstawione zostało procentowe porównanie średniej wartości dla każdego współczynnika parowania i wartości k równej 50. Z wykresu można odczytać, że algorytm mrówkowy w najgorszym przypadku zadziałał ze średnią o 30% większą od wartości oryginalnej. Optymalizacje natomiast sięgają wartości od 10.25% do 13.81%.



Rysunek 6.4: Wykres 1

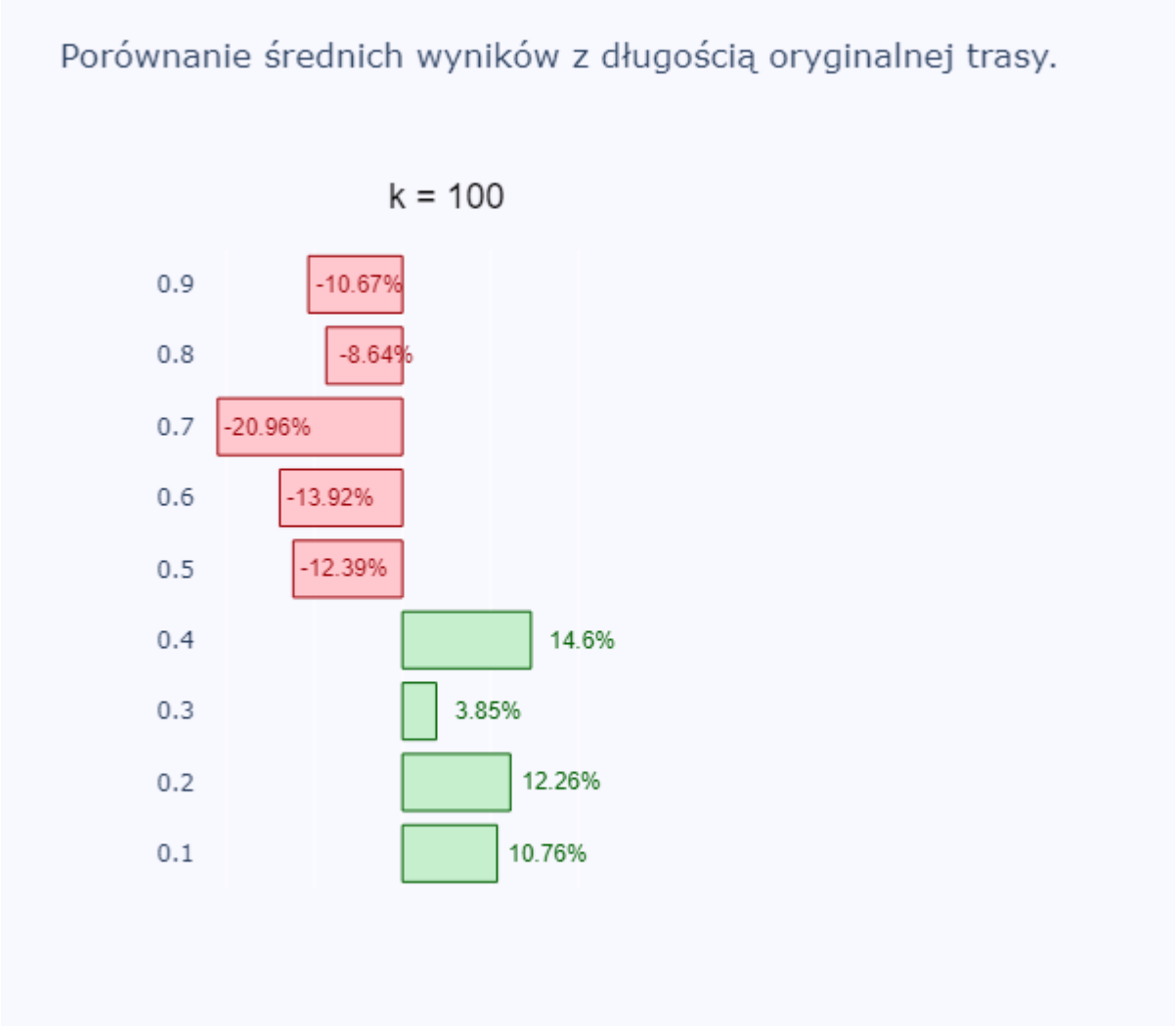
Na wykresie **??** zwiększona została ilość iteracji do 75 wyniki uległy poprawie. Najgorsza wartość została zredukowana o ok 9%, a najlepszy wynik doszedł do 15.25% co jest też najwyższą średnią pośród wszystkich możliwości.



Rysunek 6.5: Wykres 1

Dla ostatniej możliwej wartości iteracji wartość najgorsza ponownie uległa poprawie i wynosiła 20.96%. Niestety najwyższa wartość nie zwiększyła się względem mniejszej liczbie iteracji. Sytuacja ta została przedstawiona na wykresie **??**

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



Rysunek 6.6: Wykres 1

wyprodukowane przez program, można zauważyć, 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

6.1.3 Wyniki algorytmów zachłannych

6.1.4 Porównanie algorytmów

6.2 Trasa II

Druga trasa składa się ze 150 punktów, odwiedzonych przez pojazd. Długość oryginalna tej trasy wynosi 40402 metrów. W kolejnych trzech podrozdziałach zostaną przedstawione wyniki optymalizacji tej trasy przez badane algorytmy, a w czwartym zostaną one porównane.

6.2.1 Wyniki algorytmu genetycznego - Paweł

W tabeli 6.9 zostały przedstawione średnie wyniki algorytmu genetycznego dla każdej kombinacji parametrów. W przeciwieństwie do poprzedniej trasy, nie zawsze krzyżowanie CX w połączeniu z mutacją $Insert$ optymalizowała trasę. Dla N = 2000 oraz k = 100 najlepszy średni wynik osiągnęło krzyżowanie CX z mutacją $Revert$, a dla N = 2000 i k = 2000 lub k = 4000 najlepszy wyniki były dla krzyżowania CX oraz mutacji $Swap$. Dla 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 średni 30385 metrów, poprawa około 10 kilometrów. Podobnie jak na poprzedniej trasie wynik ten osiągnęło krzyżowanie CX z mutacją $Insert$ dla k = 1000. Natomiast najlepszy wynik globalny osiągnięto dla N = 2000 oraz k = 1000, krzyżowanie CX z mutacją $Insert$ i jest to 28196 metrów. 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.

W tabeli 6.10 została przedstawiona procentowa ilość wyników lepszych od długości trasy oryginalnej. Krzyżowanie PMX w czterech kombinacjach osiągnęło 100% skuteczności, dla N = 1000 lub 2000 z k = 2000 lub 4000 z mutacją $Insert$. Krzyżowanie OX z mutacją $Insert$ było bezbłędne dla N = 2000 z k >= 1000. Krzyżowanie CX okazało się jedynym, które miało 100% skuteczności w połączeniu z innymi mutacjami. Dla N = 1000 były to mutacje $Insert$ oraz $Swap$, a dla N = 2000 również z mutacją $Revert$. Mutacja $Revert$ nie poradziła sobie z krzyżowaniem OX, nie osiągnięto ani jednego wyniku lepszego niż długość trasy oryginalnej.

Tabela 6.9: Trasa II - średnie 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	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

Tabela 6.10: Trasa II - skuteczność 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	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%

W tabeli 6.11 pokazane zostały najlepsze wyniki jakie osiągnął algorytm genetyczny podczas 10 wykonań dla danych parametrów wejściowych. Dla populacji N = 100 najlepszy wynik 37172 metrów, osiągnęło krzyżowanie CX z mutacją $Insert$. Krzyżowanie CX z każdą mutacją, osiągnęło najlepszy wynik lepszy od trasy oryginalnej dla N >= 1000 i k >=100. Dla populacji N = 1000 najmniejszą odległość 27858 metrów, osiągnął algorytm przy parametrach: k = 2000, krzyżowaniu CX oraz mutacji $Swap$. Dla tego krzyżowania mutacja $Revert$ znalazła najlepsze rozwiązanie dla k = 100, zarówno dla populacji N = 1000 oraz N = 2000. Najlepsze globalne rozwiązanie zostało osiągnięte dla N = 2000, k = 2000, krzyżowania CX oraz mutacji $Swap$ i wynosi 27381 metrów. Porównując do populacji N = 1000 poprawiło się o około 500 metrów. Krzyżowanie PMX znalazło najlepsze globalne rozwiązanie wynoszące 29758 metrów, a krzyżowanie OX wynoszące 29602 metrów, oba z mutacją $Insert$.

Tabela 6.11: Trasa II - 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	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

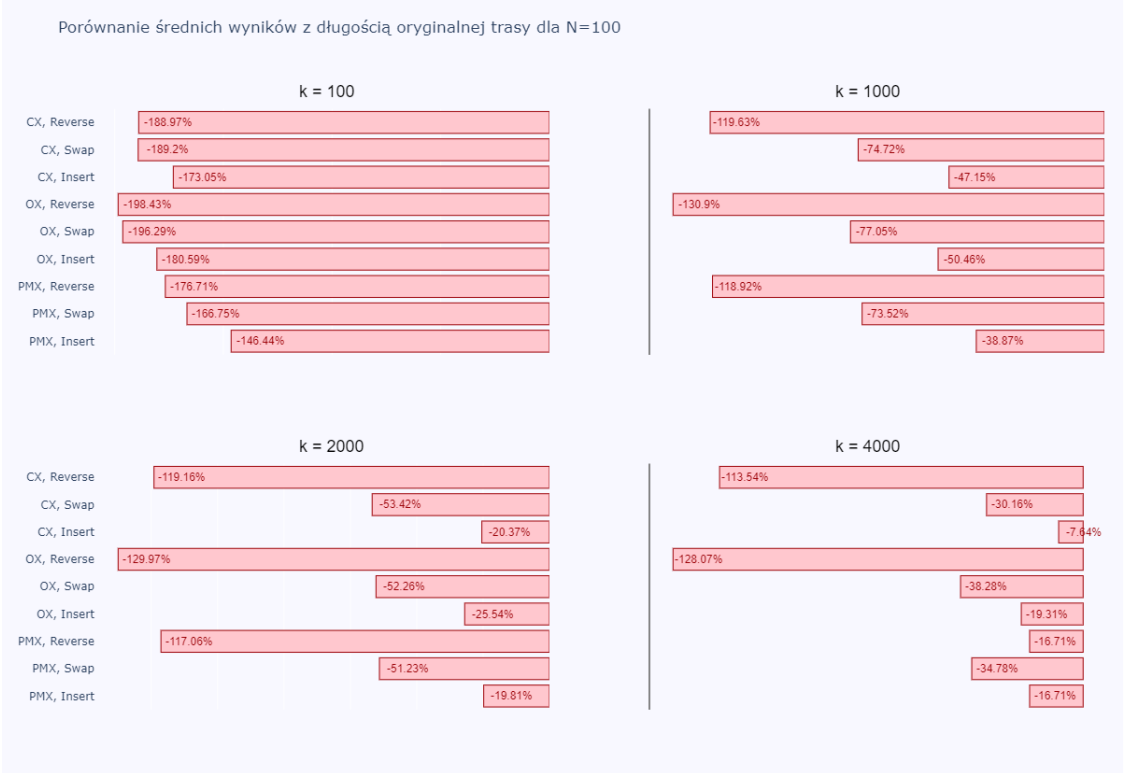
W tabeli 6.12 zostały pokazane średnie czasy trwania algorytmu genetycznego. Rozkład najlepszych średnich czasów jest bardzo podobny do trasy pierwszej. Najszybsze dla każdego N i k było krzyżowanie CX oraz mutacja Revert. Pomiędzy N = 1000, a N= 2000 średnia najlepsza optymalizacja trasy poprawiła się około 500 metrów (Tab. 6.11), 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 6.12: Trasa II - średnie czasy wykonywania algorytmu dla danych parametrów wejściowych

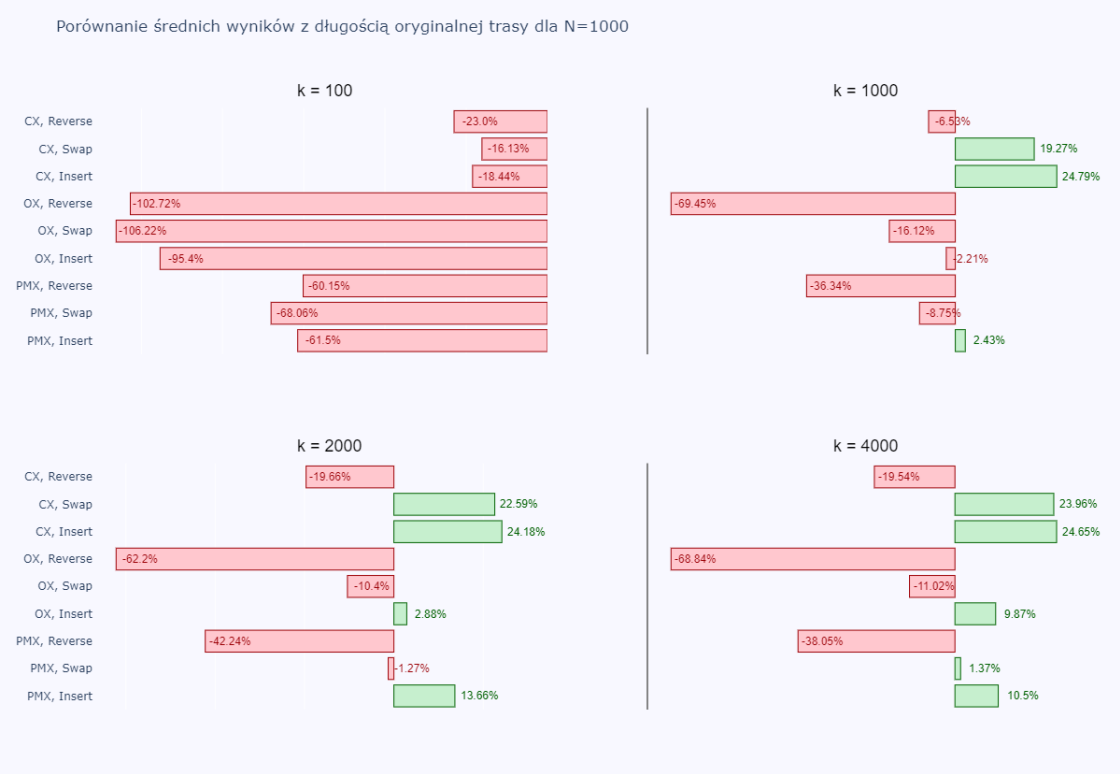
N	k	PMX Insert	PMX Replace	PMX Revert	OX Insert	OX Replace	OX Revert	CX Insert	CX Replace	CX Revert
100	100	0,04	0,04	0,03	0,02	0,02	0,02	0,03	0,02	0,02
100	1000	0,27	0,26	0,24	0,20	0,19	0,18	0,24	0,16	0,15
100	2000	0,50	0,49	0,46	0,38	0,37	0,35	0,41	0,31	0,29
100	4000	0,95	0,95	0,95	0,75	0,72	0,69	0,74	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 6.7, 6.8 oraz 6.9 przedstawiają wykresy, które porównują o ile różni się rozwiązanie od długości trasy oryginalnej, kolejno dla N = 100, N = 1000 oraz N = 2000. Dla N = 100 (Rys. 6.7) wszystkie kombinacje parametrów wejściowych nie potrafiły

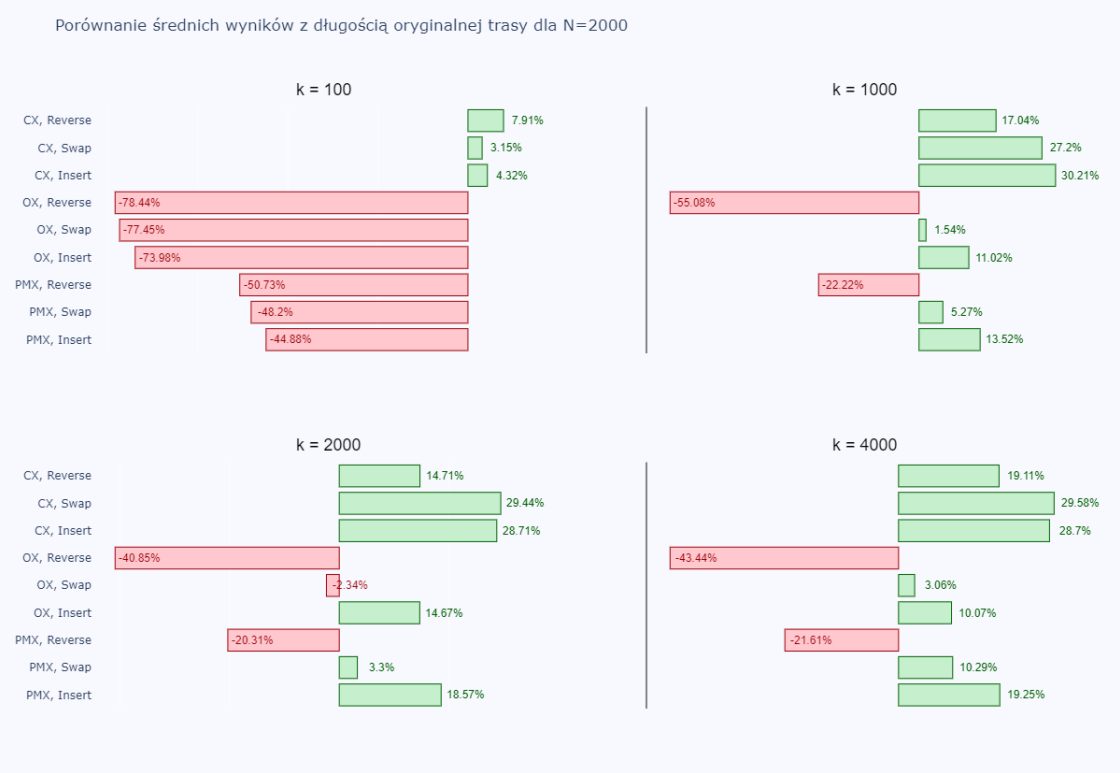
zoptymalizować rozwiązania. Dla k = 100 wygenerowane wyniki mają gorszą wartość od -195% do -150%. Przy k = 4000 wartości dla niektórych parametrów są średnio o gorsze o kilkanaście procent od trasy oryginalnej. Dla N = 1000 (Rys. 6.8), nie udało się jedynie zoptymalizować trasy dla k = 100, wszystkie 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. 6.9) 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 6.7: Trasa II - porównanie średnich wyników z długością oryginalnej trasy dla N=100



Rysunek 6.8: Trasa II - porównanie średnich wyników z długością oryginalnej trasy dla N=1000



Rysunek 6.9: Trasa II - porównanie średnich wyników z długością oryginalnej trasy dla N=2000

6.2.2 Wyniki algorytmu mrówkowego

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.

Algorytm mrówkowy nie poradził sobie z optymalizacją trasy numer 2. W tabeli numer **??** przedstawione zostały najlepsze wyniki algorytmu mrówkowego dla tej trasy. Nestety żaden z wyników nie jest krótszy od oryginalnego. Zwiększanie ilości iteracji nie przynosiło ulepszeń wyniku. Najlepszy odnotowany wynik został sklasyfikowany dla 50 iteracji i wynosi 75 069. Jest to wynik gorszy od oryginalnego o 34 667. Najgorszy z wyników przekracza ponad 100 000 metrów co jest ponad dwukrotnie gorszym wynikiem.

Tabela 6.13: Trasa II - najlepsze wyniki algorytmu mrówkowego dla danych parametrów wejściowych

k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
50	101010	86933	75226	78697	75069	77189	77539	76637	76199
75	91453	81128	75485	78774	76617	78815	76810	77423	76446
100	85587	81400	76782	76559	75351	77897	75233	76713	77081

Tabela 6.14: Trasa II - średnie wyniki algorytmu mrówkowego dla danych parametrów wejściowych

k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
50	105951	90688	77772	82462	80346	81396	81451	80728	80210
75	94734	83801	78304	82241	78774	81767	80815	80590	80721
100	90821	83212	78372	80373	79902	81995	79855	80240	80458

czas

Tabela 6.15: Trasa II - średnie wyniki czasu wykonania algorytmu mrówkowego dla parametru ilości iteracji

iteracje	50	75	100
czas	28.325	42.689	60.281

6.2.3 Wyniki algorytmów zachłannych

6.2.4 Porównanie algorytmów

6.3 Trasa III

Trasa trzecia składa się z 248 punktów i ma długość 53478 metrów.

6.3.1 Wyniki algorytmu genetycznego - Paweł

W tabeli 6.16 zostały pokazane średnie wyniki algorytmu genetycznego dla trasy III. Krzyżowanie OX ani razu nie osiągnęło średniego wyniku lepszego od długości trasy oryginalnej. Dla N = 100, 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 423728 metrów, jest to poprawa długości około 10 kilometrów. 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. Dla populacji N = 2000 wyniki rozłożyły się podobnie do poprzedniej. Najlepszy globalny wynik 34168 metrów został osiągnięty dla krzyżowania CX, mutacji Insert o k = 4000.

Tabela 6.16: Trasa III - Średnie 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	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 6.17 została przedstawiona procentowa ilość wyników lepszych od długości trasy oryginalnej. Dla N= 100, N = 1000 i k = 100 oraz N = 2000 i k = 100 nie wystąpił

Tabela 6.17: Trasa III - tbd algorytmu genetycznego dla danych 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%	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%

ani jeden wynik, który optymalizowałby trasę. 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. Dopiero dla krzyżowania CX i N = 2000 algorytm genetyczny miał skuteczność na poziomie 40 - 60 procent. Krzyżowanie CX z mutacją Insert uzyskało dla pięciu różnych kombinacji wartości N i k 100 procent skuteczności.

W tabeli 6.18 przedstawiono najlepsze wyniki algorytmu genetycznego 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 prawie 20 kilometrów. Algorytm genetyczny z krzyżowanie PMX dla tej populacji osiągnęło najlepszy wynik 43708 metrów. Najlepszy globalny wynik został osiągniety dla N = 2000, k = 1000, krzyżowania CX oraz mutacji Insert wyniósł 32372 metrów. Krzyżowanie PMX poprawiło się nieznacznie.

6.3.2 Wyniki algorytmu mrówkowego

AVG

BEST

czas

6.3.3 Wyniki algorytmów zachłannych

6.3.4 Porównanie algorytmów

Tabela 6.18: Trasa III - 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	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 6.19: Trasa III - średnie czasy wykonywania algorytmu dla danych parametrów wejściowych

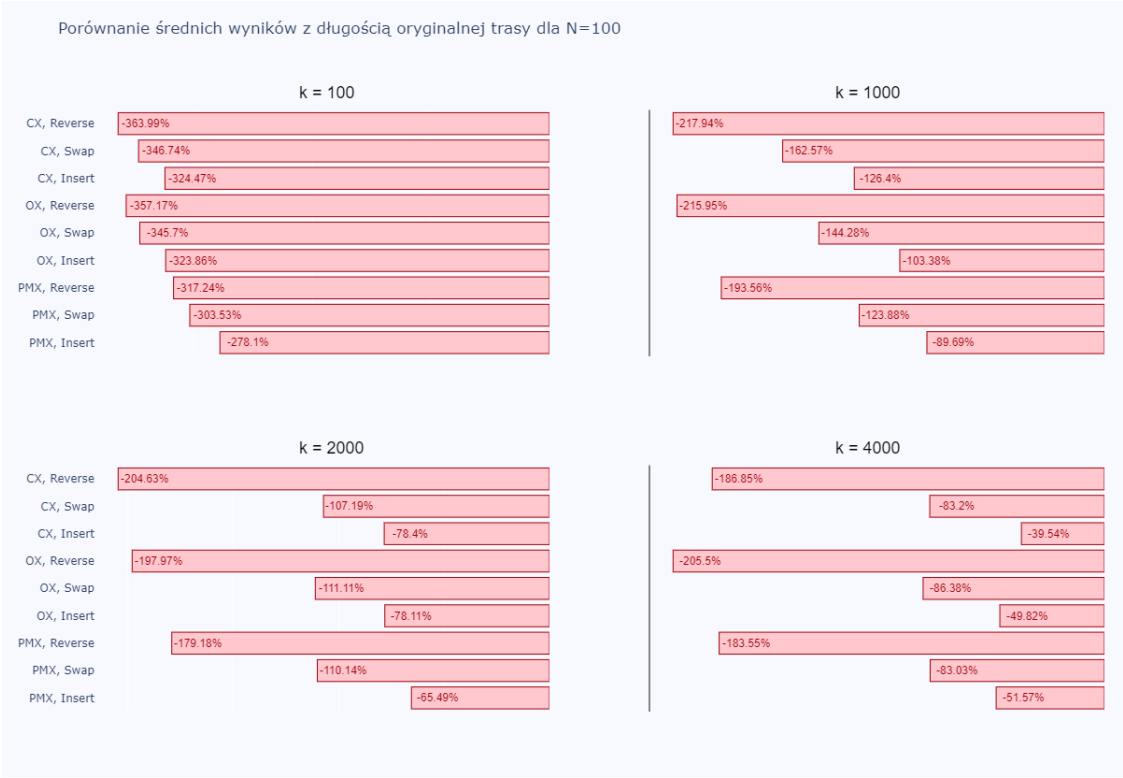
N	k	PMX Insert	PMX Swap	PMX Revert	OX Insert	OX Swap	OX Revert	CX Insert	CX Swap	CX Revert
100	100	0,09	0,07	0,06	0,04	0,04	0,03	0,07	0,03	0,03
100	1000	0,44	0,43	0,35	0,32	0,31	0,26	0,48	0,25	0,22
100	2000	0,83	0,79	0,69	0,60	0,58	0,52	0,78	0,46	0,44
100	4000	1,51	1,48	1,35	1,12	1,11	1,03	1,28	0,89	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

Tabela 6.20: Trasa III - średnie wyniki algorytmu mrówkowego dla danych parametrów wejściowych

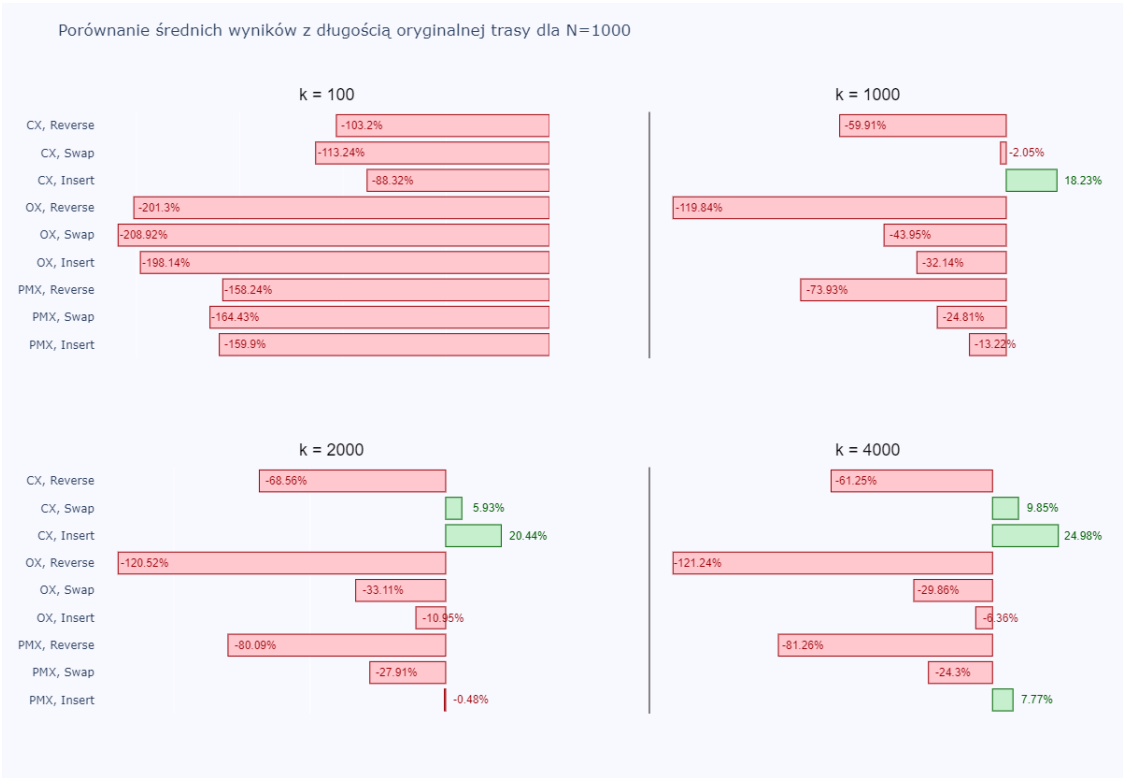
k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
50	61628	45302	43699	42668	48201	47483	50840	47157	48293
75	49664	43667	44621	44292	46181	48248	48021	48112	48609
100	45504	45802	46286	45465	45721	49379	47921	46410	47433

Tabela 6.21: Trasa III - najlepsze wyniki algorytmu mrówkowego dla danych parametrów wejściowych

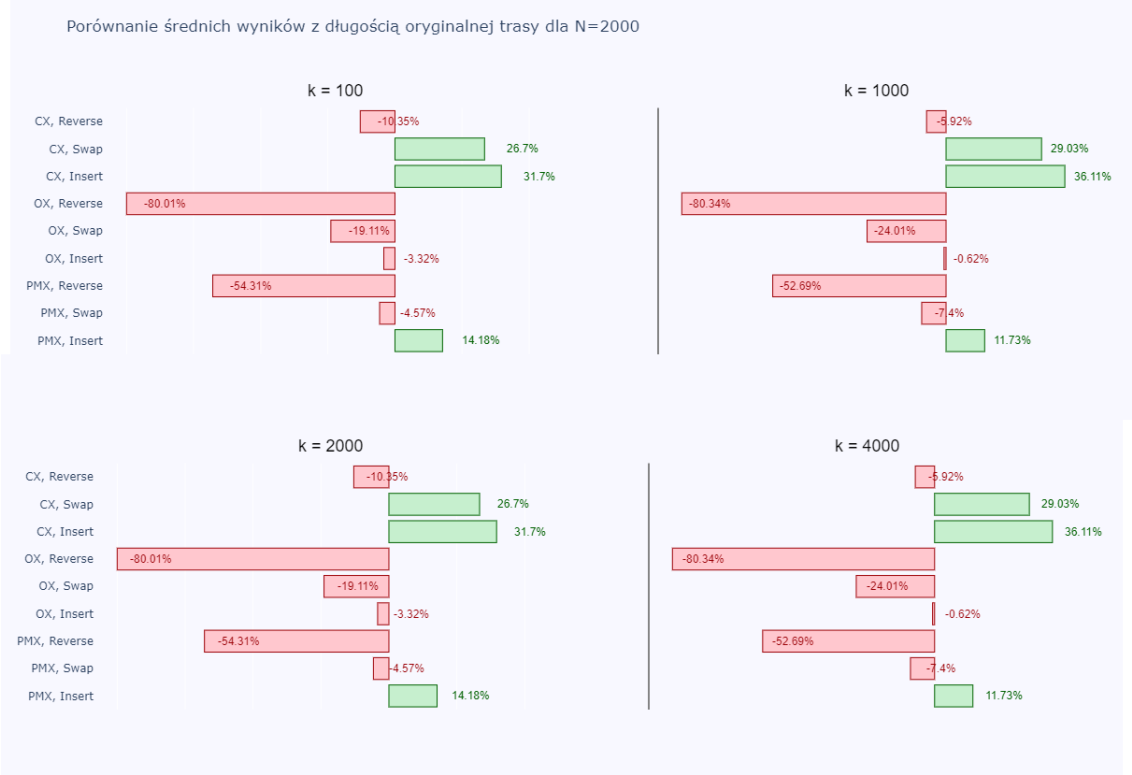
k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
50	59369	43287	40214	41158	44252	43387	48045	41331	41858
75	46953	42126	41568	41313	43466	43728	45216	45947	46255
100	40410	41250	40509	41796	40815	42413	44712	42284	43308



Rysunek 6.10: Trasa III - porównanie średnich wyników z długością oryginalnej trasy dla N=100



Rysunek 6.11: Trasa III - porównanie średnich wyników z długością oryginalnej trasy dla N=1000



Rysunek 6.12: Trasa III - porównanie średnich wyników z długością oryginalnej trasy dla N=2000

Tabela 6.22: Trasa III - średnie wyniki czasu wykonania algorytmu mrówkowego dla parametru ilości iteracji

iteracje	50	75	100
czas	84.451	131.146	176.874

Podsumowanie

Tutaj będzie podsumowanie.

Bibliografia

[1] Encyklopedia AlgorytmĂłw. Encyklopedia algorytmĂłw. http://algorytmy.ency.pl/artykul/algorytm_najblizszego_sasiada, 2016.

[2] Encyklopedia AlgorytmĂłww. Encyklopedia algorytmĂłww. http://algorytmy.ency.pl/artykul/algorytm_helda_karpa, 2016.

[3] Lenart Andrzej. *Maszynoznawstwo przemysĹ‚u spoĹ‚ywczego*. SGGW, Warszawa, 2003.

[4] J. Autor. Nazwa strony internetowej. http://www.dlugi.adres.url.zlamie.sie.gdzies.w.srodku.com, stan z 01.01.2010 r.

[5] U. Autor and W. Kolejny. TytuĹ‚, publikacji. *Nazwa czasopisma*, 12(2):132–145, May 2012.

[6] Daniel BĹ‚aszkiewicz. Algorytmy mrĂłwkowe. *Instytut Informatyki Uniwersytetu Wrocd'ź˝awskiego*, 1(1):5–6.

[7] Ewa Figielska. Algorytmy ewolucyjne i ich zastosowania. *Zeszyty Naukowe Warszawskiej WyĹ‚szej SzkoĹ‚y Informatyki*, 1(1):81–92, 2006.

[8] 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.

[9] David E. Goldberg. *Algorytmy genetyczne i ich zastosowania*. Wydawnictwo Naukowo-Techniczne, Warszawa, 1998.

[10] Anna Gryko-Nikitin. NiektĂłre osobliwoĹ›ci algorytmĂłw genetycznych na przykĹ‚adzie zagadnieĹ„ logistycznych. *Ekonomia i ZarzÄ…dzanie*, 1(2):129–138, 2010.

[11] 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*, page 224â€“230, 1987.

[12] 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.

[13] Rutkowski L Rutkowska D., PiliĹ„ski 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] RadosĹ‚aw Winiczenko. Algorytmy genetyczne i ich zastosowania. *PostÄ™py Techniki PrzetwÃłrstwa SpoĹ»ywczego*, 1(2):107–110, 2008.

Spis tabel

Spis rysunków

Spis listingów

Spis algorytmów