**AKADEMIA NAUK STOSOWANYCH**

**W NOWYM SĄCZU**

**WYDZIAŁ NAUK INŻYNIERYJNYCH**

**PRACA DYPLOMOWA**

**ANALIZA I OCENA EFEKTYWNOŚCI ALGORYTMÓW ROZWIĄZANIA PROBLEMU KOMIWOJAŻERA**

**Autor: Filip Rzepiela**

**Kierunek: Informatyka**

**Nr albumu: 29757**

**Promotor: dr inż. Józef Zieliński**

**Akceptacja promotora: ……………………………………………………**  data i podpis

**NOWY SĄCZ 2024**

Spis treści

[Wstęp 5](#_Toc167301968)

[1. Problem komiwojażera 6](#_Toc167301969)

[1.1. Geneza problemu komiwojażera 6](#_Toc167301970)

[1.2. Opis problemu 21](#_Toc167301971)

[1.3. Rodzaje problemów 24](#_Toc167301972)

[2. Podstawowe pojęcia 27](#_Toc167301973)

[2.1. Graf 27](#_Toc167301974)

[2.2. Cykl Eurela 27](#_Toc167301975)

[2.3. Twierdzenie Eurela 28](#_Toc167301976)

[2.4. Graf eurelowski i półeurelowski 29](#_Toc167301977)

[2.5. Cykl Hamiltona 30](#_Toc167301978)

[2.6. Graf Hamiltonowski 30](#_Toc167301979)

[2.7. Problem cyklu Hamiltona 33](#_Toc167301980)

[3. Przegląd algorytmów 35](#_Toc167301981)

[3.1. Algorytmy k-optymalne. 35](#_Toc167301982)

[3.1.1. Algorytm 2-optymalny 36](#_Toc167301983)

[3.1.2. Algorytm 3-optymalny 37](#_Toc167301984)

[3.2. Algorytm zachłanny 38](#_Toc167301985)

[3.2.1. Algorytm Kruskala 40](#_Toc167301988)

[3.2.2. Algorytm Prima 42](#_Toc167301989)

[3.3. Algorytm selekcji krawędzi 44](#_Toc167301990)

[3.4. Algorytm genetyczny 46](#_Toc167301991)

[3.5. Algorytm mrówkowy 51](#_Toc167301992)

[3.6. Algorytm Christofidesa 55](#_Toc167301993)

[3.7. Algorytm Little’a 57](#_Toc167301994)

[4. Cel i zakres pracy 62](#_Toc167301995)

[5. Metodyka badań 63](#_Toc167301996)

[6. Implementacja komputerowa 64](#_Toc167301997)

[7. Badanie porównawcze algorytmów 64](#_Toc167301998)

[8. Podsumowanie i wnioski 64](#_Toc167301999)

[Bibliografia 65](#_Toc167302000)

[Spis rysunków 67](#_Toc167302001)

[Spis tabel 68](#_Toc167302002)

# Wstęp

W dzisiejszych czasach szybkość przetwarzania informacji oraz zdolność do efektywnej optymalizacji procesów mają decydujące znaczenie dla konkurencyjności, a także wydajności przedsiębiorstw oraz organizacji na całym globie. Generowane w obecnych czasach ery cyfryzacji astronomiczne ilości danych, przetwarzanie oraz analiza niemalże w czasie rzeczywistym, algorytmy optymalizacyjne stanowią fundament nie tylko w informatyce, ale również dają możliwość zarządzania zasobami, planowaniem logistyki, produkcji oraz wielu innych aspektów wszelakich działalności gospodarczych. Wszelakie nauki informatyczne zajmujące się badaniem oraz poszukiwaniem rozwiązania m.in. w dziedzinie optymalizacji kombinatorycznej szczególne miejsce poświęcają problemowi komiwojażera   
(TSP – Travelling Salesman Problem). Wspomniany problem słynący ze swojej pozornej prostoty polegającej na odnalezieniu najkrótszej możliwej ścieżki przechodzącej przez określoną liczbę punktów i powracającej ostatecznie do punktu wyjścia, stanowi jedno z najbardziej złożonych i trudnych wyzwań obliczeniowych, a co za tym idzie uznawany jest za NP-trudny. Oznacza to, że przy większej ilości punktów nie istnieje algorytm, który będzie potrafił rozwiązać problem w czasie wielomianowym [[1]](#footnote-1) dla wszystkich możliwych instancji.

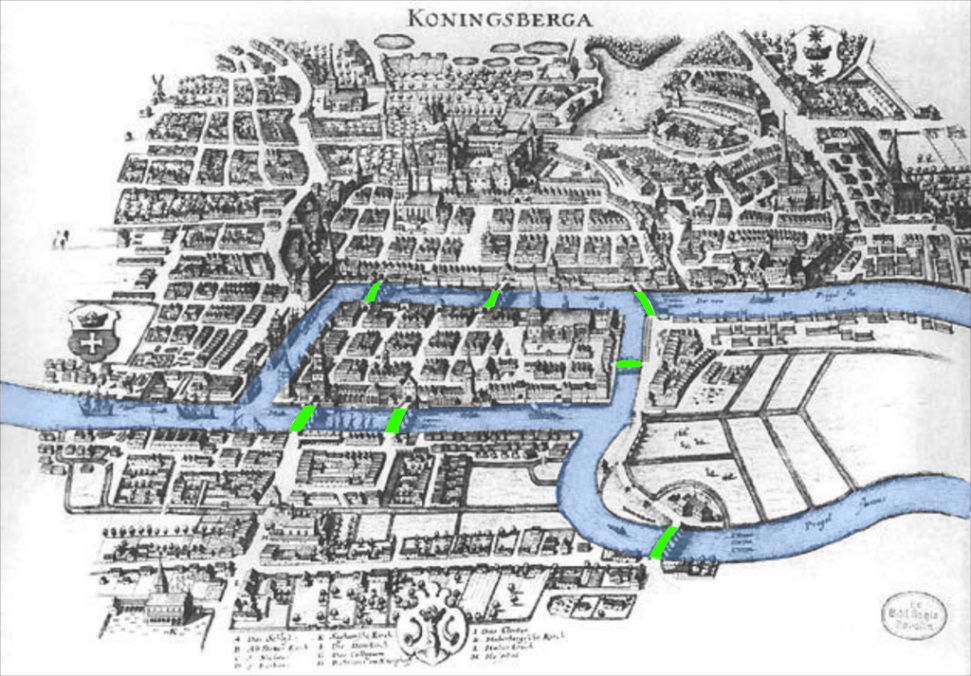
W pierwszym rozdziale przedstawiona zostanie geneza problemu komiwojażera na przestrzeni lat, począwszy od czasów prehistorycznych, aż do czasów ówczesnych. Rozdział drugi przedstawiać będzie podstawowe pojęcia, z pomocą których zrozumienie idei problemu komiwojażera będzie łatwe do przyswojenia. Następnie rozdział trzeci będzie zawierał przegląd istniejących algorytmów, za pomocą których można podjąć próbę rozwiązania NP-trudnego problemu komiwojażera. Rozdział czwarty zawierać będzie cel i zakres pracy. W kolejnym rozdziale przedstawiona zostanie metodyka przeprowadzonych badań. Rozdział szósty zawierać będzie implementację komputerowych wybranych algorytmów. Kolejnym krokiem będzie przeprowadzenie badań porównawczych algorytmów. Pracę zwieńczy podsumowanie wraz z wnioskami z przeprowadzonych badań w ramach realizacji pracy dyplomowej.

# Problem komiwojażera

Rozdział pierwszy skupia swoją uwagę na przedstawieniu historii problemu komiwojażera od czasów prehistorycznych, po lata ówczesne. Druga część rozdziału objaśnia problem komiwojażera. Trzeci podrozdział z kolei przedstawia rodzaje problemów rozpoczynając od prostych rozwiązywalnych oraz kończąc na nierozwiązywalnych.

## Geneza problemu komiwojażera

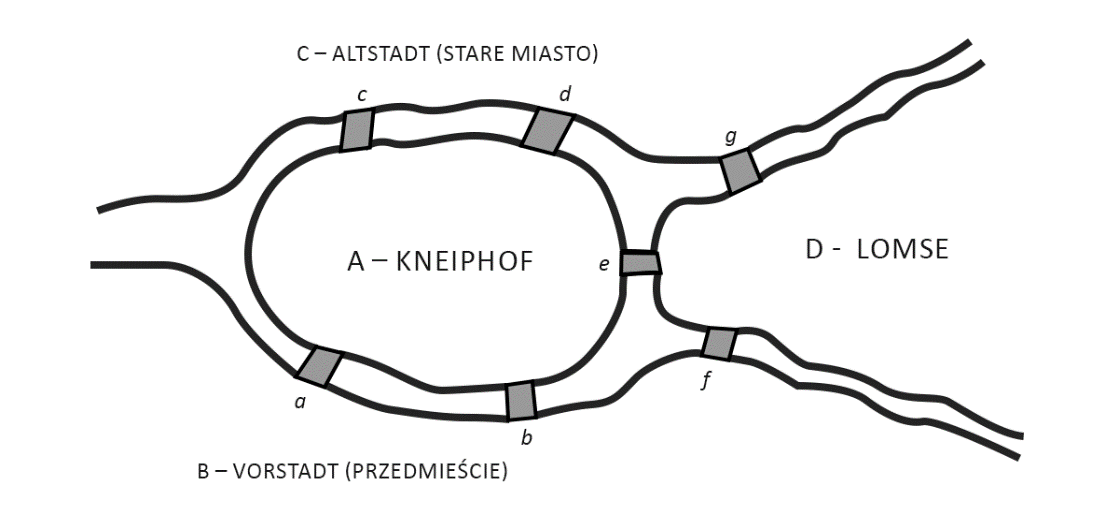
(Cook, 2012) Konieczność optymalizacji tras sięga czasów prehistorycznych, kiedy to jaskiniowcy rozwiązywali małe wersje problemu komiwojażera w czasie polowań oraz zbieractwa, niemniej jednak nie został on sformułowany w sposób matematyczny. Na przestrzeni wieków przedstawiciele różnorakich zawodów starali się zaplanować trasę podróży. (Wikipedia, 2023) W 1741 roku Leonhard Euler opublikował pracę „Solutio problematis ad geometriam situs pertinentis w Commentarii academiae scientiarum Petropolitanae”[[2]](#footnote-2), która opisywała problem objazdowy. (Cook, 2012) Euler we wspomnianym wcześniej dziele opisał problem mostów królewieckich. Rysunek 1 przedstawia plan miasta Królewiec, w którym Euler przeprowadził badanie. Kolorem niebieskim zaznaczona została rzeka Pregel, obecnie nazywana Pregoła, natomiast kolorem zielonym zaznaczone zostały mosty.



Rysunek 1. Plan miasta Królewiec z czasów Eulera

[*źródło:* (Kohlstedt, 2022)]

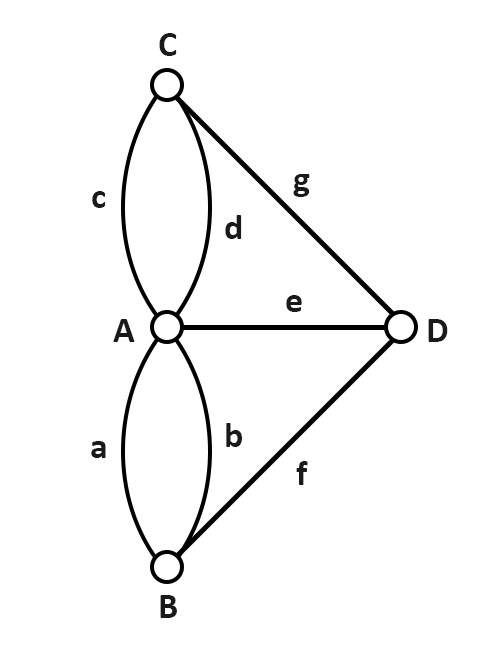
W epoce Eulera rzekę przecinało siedem mostów łączących kolejno wyspę Kneiphof ze Starym Miastem (region na północy), Przedmieściem (region południowy) oraz Lomse (wschodnia wyspa), a także wschodnią wyspę z Przedmieściem oraz Starym Miastem. Wyzwaniem przed jakim stawali obywatele miasta, było przekroczenie podczas spaceru każdego z mostów wyłącznie raz. Analizy i próby rozwiązania problemu podjął się Euler, początkowo szkicując plan miasta, rzekę oraz mosty, a następnie opisał części Królewca jako litery A, B, C oraz D, a także mosty Green, Köttel, Krämer, Schmiede, Honey, Lomse i Wood jako litery a-g, co zostało przedstawione na rysunku 2.



Rysunek 2. Przedstawienie mostów w Królewcu według Eulera

[*źródło:* opracowanie własne na podstawie (Cook, 2012)]

Zaproponowane przez autora etykiety były wystarczające do opisania dowolnej trasy np. z punktu A do C przez most c. Warto zauważyć, że przedstawione przez Leonharda Eulera argumenty były w pełni oparte na manipulacji trasami jak ciągami symboli, natomiast wielkość lądowa nie ogrywała żadnej roli, co pozwoliło przedstawić mosty królewieckie na postawie prostego wykresu, w którym części miasta stanowią punkty A-D, natomiast mosty A-G przestawione zostały jako linie za pomocą których punkty zostały połączone, co zilustrowane zostało na rysunku 3.



Rysunek 3. Wykres przedstawiający reprezentacje Mostów Królewieckich

[*źródło:* opracowanie własne na podstawie (Cook, 2012)]

Według przestawionego powyżej wykresu, można zauważyć, że spacer po Królewcu opiera się na przemieszczaniu z wierzchołka do wierzchołka za pomocą krawędzi przebiegającej między nimi. Przykładowo dla zaplanowanego spaceru z punktu B do D, przez każdy z siedmiu mostów, trasa wygląda w następujący sposób: B a AcCg De Ab B f D. W przykładowej trasie spacerowej w punkcie B spotykają się trzy krawędzie – a, b oraz f, w punkcie A natomiast dochodzi do spotkania czterech krawędzi a,c,e oraz b. Analizując kolejne punkty we wierzchołku C spotyka się krawędź c oraz g, natomiast w D ponownie dochodzi do zetknięcia się ze sobą trzech krawędzi, mianowicie g, e oraz f. Euler zaobserwował, że liczby spotkań w danym punkcie mają wzór nieparzysty-parzysty-parzysty-nieparzysty. Podczas spaceru pomiędzy dwoma różnymi punktami spotykana jest nieparzysta liczba krawędzi, natomiast z innymi wierzchołkami stykają się parzystą ilością krawędzi. Jeżeli pod uwagę wzięty zostanie spacer zaczynający się i kończący w tym samym punkcie, to w tej sytuacji każdy z punktów spotyka parzystą ilość krawędzi. Wspomniana obserwacja pomogła zakończyć debatę mieszkańców Królewca, ze względu na fakt, że wszystkie cztery części miasta spotykają się z nieparzystą ilością krawędzi i nie istnieje możliwość przeprowadzenia spaceru z wykorzystaniem każdej z krawędzi wyłącznie raz. Teoria Eulera stanowiła podwaliny pod teorię grafów, która stała się kluczowa dla rozwoju problemu komiwojażera.

W późniejszym czasie Euler opisał kolejny problem objazdowy – problem wędrówki rycerza w szachach. Celem rzeczonego wyzwania jest znalezienie sekwencji ruchów rycerza na szachownicy, która umożliwi mu odwiedzenie co drugiego pola dokładnie raz, zaczynając i kończąc na polu startowym. Zaproponowane przez autora rozwiązanie, w którym kolejność każdego ruchu została opisana liczbowo w polach szachownicy została przedstawiona na rysunku 4.



Rysunek 4. Rozwiązanie Eurela problemu wędrówki rycerza

[*źródło:* opracowanie własne na podstawie (Cook, 2012)]

W przypadku problemu wędrówki rycerza dla każdego z pól szachownicy ustalony został wierzchołek z dwoma wierzchołkami połączonymi krawędzią, w przypadku gdy rycerz chce przemieścić się między polami za pomocą tylko jednego ruchu. Podobnie jak w przypadku mostów królewieckich stanowi to spacer zamknięty, ponieważ należy zacząć i zakończyć w tym samym punkcie, przechodząc przez każdą krawędź tylko raz, co zostało przedstawione na rysunku 5.

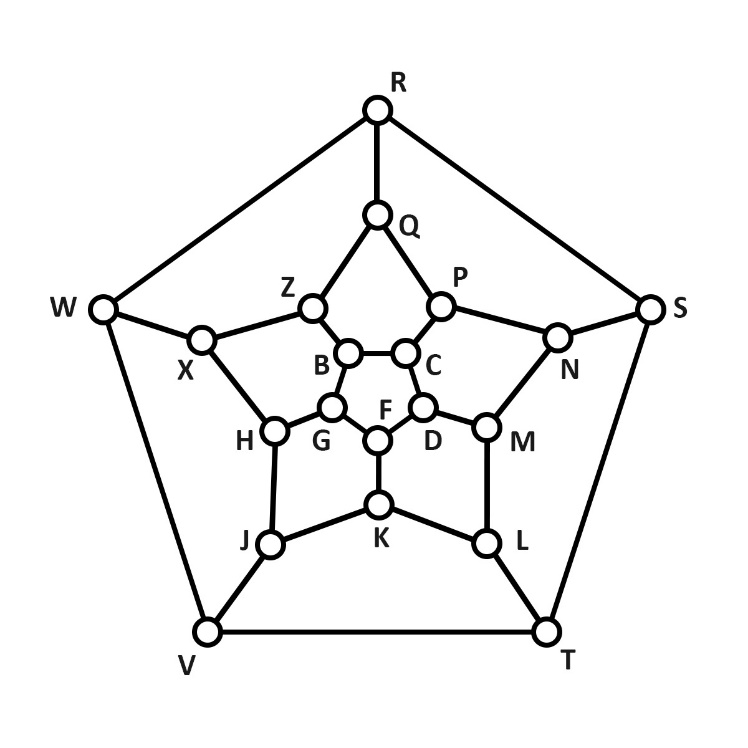
Obraz zawierający wzór, linia, Symetria, zrzut ekranu

Opis wygenerowany automatycznie

Rysunek 5. Trasa rycerska na wykresie szachownicy.

[*źródło:* (Cook, 2012)]

Wiek po Eulerze temat wycieczek po konkretnym wykresie zainteresował kolejnego uczonego, Sir Williama Rowana Hamiltona, który rozważał sposoby odwiedzenia wszystkich dwudziestu punktów narożnych dwunastościanu dwunastościennej bryły platońskiej. Podczas analizy problemu opracował abstrakcyjny rysunek, znany jako Icosian, w którym to linie reprezentują geometryczne krawędzie dwunastościanu, natomiast okręgi przedstawiają jego narożniki. Graf opracowany przez Hamiltona przedstawiony został na rysunku 6.



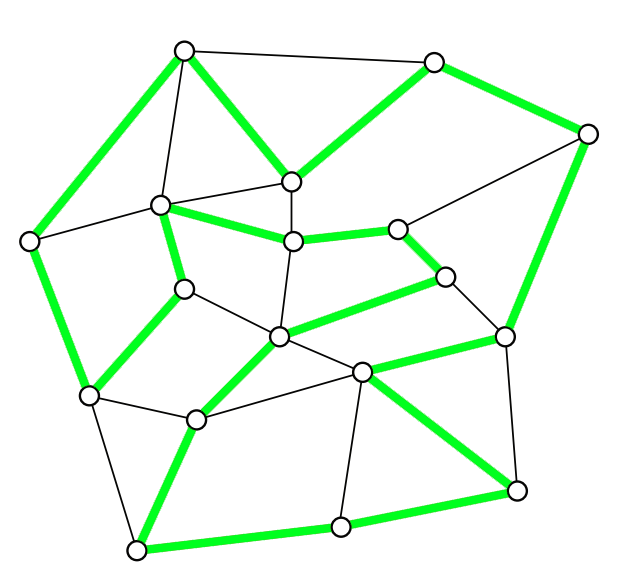
Rysunek 6. Icosian

[*źródło:* opracowanie własne na podstawie (Cook, 2012)]

W grafie Hamiltona wycieczki rozpoczynają się od wierzchołka do wierzchołka, podróżując wzdłuż krawędzi wykresu. W Icosianie Hamilton wprowadził system algebraiczny do reprezentowania możliwych ścieżek w grafie. Przyjął symbole i, κ oraz λ, których zadaniem jest spełnianie czterech następujących równań:

Równanie numer 1 symbolizuje operację, która przeprowadzana dwukrotnie nie zmienia położenia na grafie, czyli jest to jakby powrót do punktu wyjścia. Równanie numer 2 natomiast reprezentuje operację, która po trzykrotnym wykonaniu również prowadzi z powrotem do punktu wyjścia, z kolei równanie numer 3 wskazuje na to samo dla operacji wykonanej pięciokrotnie. Ostatnie równanie numer 4 wskazuje na relację między tymi operacjami. Przeprowadzone przez Hamiltona abstrakcje pozwoliły dokonać manipulacji ścieżkami w Icosianie bez potrzeby wizualizowania każdego etapu, co było przełomowym rozwiązaniem dla jego przyszłych prac nad kwaternionami. Na podstawie Icosianu Hamilton opracował grę ikozjańską, która stanowiła nie tylko rozrywkę matematyczną, ale była również platformą badań nad strukturami grafów. Celem gry było odnalezienie ścieżki przechodzącej przez każdy z wierzchołków dwunastościanu Hamiltona dokładnie raz.

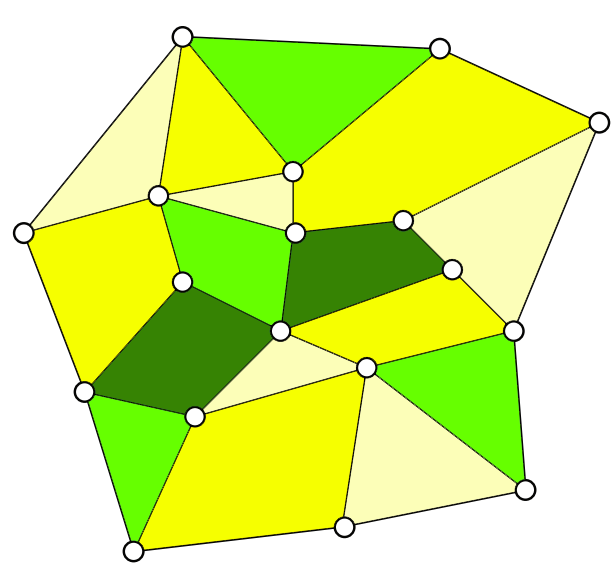
Głównym aspektem wspomnianej gry było zastosowanie idei obwodów, które później znalazły swoje zastosowanie w definicji cykli Hamiltona. (Wilson, 2000) Cykl Hamiltona to zamknięta ścieżka w grafie, w której zbiór wierzchołków, wykluczając punkt startowy oraz końcowy, są odwiedzane wyłącznie raz. Cykl rozpoczyna się i kończy w tym samym wierzchołku, odwiedzając pozostałe wyłącznie jeden raz. W przypadku odnalezienia cyklu Hamiltona o minimalnej sumie wag odwiedzonych krawędzi rozwiązany zostanie problem komiwojażera. (Cook, 2012) Określenie czy graf posiada obwód hamiltonowski, czy też nie jest jednak problemem NP-zupełnym. W 1879 roku Alfred Kempe opublikował dowód twierdzenia o czterech kolorach, który był akceptowany przez matematyków przez prawie dekadę. Strategia Kempa polegała na tym, że każda sytuacja, w której cztery kolory mogłyby być konieczne, mogła być przekształcana za pomocą tych łańcuchów, co pozwalało zachować zgodność kolorów wzdłuż granic regionów. Zainspirowany dowodem Kempa szkocki fizyk i matematyk Peter Guthrie Tait przypuszczał, że pewien typ wykresów ma zawsze cykl hamiltonowski. Aby zauważyć związek pomiędzy podróżowaniem, a kolorowaniem mapy należy wyobrazić sobie granice poszczególnych regionów mapy jako krawędzie grafu, natomiast punkty przecięcia jako wierzchołki. Jeżeli przez dany wykres przechodzi cykl Hamiltona, to wykres graniczny umożliwia pokolorowanie mapy. Rysunek 7 przedstawia przykład grafu, przez który przechodzi cykl Hamiltona.



Rysunek 7. Cykl Hamiltona w grafie.

[*źródło:* opracowanie własne na podstawie (Cook, 2012)]

Kolorem zielonym na wspomnianym rysunku przedstawione zostały krawędzie tworzące obwód Hamiltona. Można zauważyć, że obwód nie przecina się sam ze sobą, zatem posiada granice zewnętrzne oraz wewnętrzne. Krawędzie graniczne od wewnątrz przecinają obszar wewnętrzny. Zatem istnieje możliwość pokolorowania tych obszarów dwoma kolorami, zmieniając kolor za każdym razem, gdy przekroczona zostanie jedna z krawędzi nie obwodowych. Taką samą zasadę można zastosować do pokolorowania dwoma kolorami obszarów zewnętrznych znajdujących się poza obwodem Hamiltona. W rezultacie uzyskana została mapa czterokolorowa. W przedstawionym na rysunku 8 przykładzie obszary znajdujące się w cyklu Hamiltona posiadają kolor ciemnożółty oraz jasnożółty, natomiast pozostałe obszary kolor ciemnozielony i jasnozielony.



Rysunek 8. Pokolorowana mapa cyklu Hamiltona.

[*źródło:* opracowanie własne na podstawie(Cook, 2012)]

Peter Guthrie Tait miał świadomość, że nie wszystkie mapy posiadają w granicach obwody Hamiltona, jednak dostępne sposoby dały możliwość ograniczenia problemu czterech kolorów do map w taki sposób, że każdy wierzchołek wykresu granicznego spotyka dokładnie trzy krawędzie. Należy założyć, że graf brzegowy nie może podzielić wykresu na dwie części poprzez usunięcie wierzchołków. Tait zakładał, że stosując ograniczenie do trzech połączonych map sprawi, że obwody Hamiltona będą zawsze dostępne. Z biegiem czasu okazało się, że zarówno teoria Alfreda Kempe, jak i Petera Guthrie Tait’a okazały się błędne. Problem czterech kolorów jest bardzo złożony, a jego dowód został odnaleziony dopiero w 1976 roku, kiedy to Kenneth Appel i Wolfgang Haken udowodnili twierdzenie o czterech barwach, co było przełomem w długiej historii tego problemu. (Wikipedia, 2024) Dowód Appela i Hakena opierał się na redukcji problemu do mniejszej liczby niezmiennych konfiguracji, które mogą wystąpić na mapie. Autorzy dowodu zidentyfikowali 1936 niezmiennych konfiguracji, które musiały zostać sprawdzone. W celu dokonania serii skomplikowanych sprawdzeń wykorzystali oni komputer, aby wykazać, że każda z zidentyfikowanych konfiguracji mogła zostać pokolorowana czterema kolorami, tak aby nie naruszyć zasad twierdzenia. Ostateczny dowód twierdzenia wymagał także dowiedzenia, że każda możliwa mapa może być zredukowana do jednej z tych konfiguracji. Ostatecznie przyjęto następującą definicję twierdzenia o czterech barwach, zgodnie z którą w grafie planarnym składającym się z wierzchołków i krawędzi można przypisać każdemu wierzchołkowi jedną z czterech liczb , odpowiadającym kolorom, tak aby żadne dwa sąsiadujące wierzchołki, czyli wierzchołki połączone bezpośrednio krawędzią, nie miały przypisanego tego samego koloru. Matematycznie relację tę opisują następujące wzory:

(5)

(6)

Wzór numer 5 opisuję funkcję kolorowania , która przypisuje wierzchołkowi w grafie jedną z czterech wartości . Te wartości reprezentują cztery różne kolory. Funkcja ta jest aplikacją, czyli przyporządkowaniem, które dla każdego punktu (wierzchołka) grafu planarnego przypisuje dokładnie jeden z czterech dostępnych kolorów. Wzór 6 natomiast głosi, że dla każdej pary wierzchołków i , które są połączone krawędzią czyli sąsiadują ze sobą na grafie, kolor przypisany do jest różny od koloru przypisanego do . Symbol oznacza "dla wszystkich", co w tym kontekście oznacza, że każda krawędź w grafie musi łączyć wierzchołki różnych kolorów.

(Cook, 2012) Euler wraz z Hamiltonem w swoich badaniach skupili się na badaniu podróży, jednak ich badania były dalekie od problemu sprzedawcy w podróży, który oczekiwał jak najkrótszej długości. Tematem tym zainteresował się Karl Menger, który w latach dwudziestych XX wieku zajmował się badaniem technik pomiaru długości krzywych w przestrzeni. Badania te stały się inspiracją dla ogłoszenia bliskiego problemowi komiwojażera problemu posłańca. Problem posłańca to zadanie matematyczne i logistyczne, które na co dzień napotyka wielu listonoszy i podróżników. Polega na znalezieniu najkrótszej trasy łączącej określoną liczbę punktów, między którymi z góry wiadomo, jakie są odległości. Celem problemu posłańca jest wytyczyć ścieżkę, która połączy te punkty w taki sposób, aby cała droga była jak najkrótsza, nie wracając przy tym do punktu wyjścia. Problem ten można łatwo uogólnić do problemu komiwojażera poprzez dodanie fikcyjnego miasta, które służy jako punkt łączący koniec i początek trasy. Przyjmuje się, że koszt dojazdu z tego dodatkowego punktu do każdego z rzeczywistych miast jest zerowy, dzięki czemu nie wpływa on na wybór miejsca startowego ani końcowego na trasie. W ten sposób, mimo że zadanie wydaje się być zmodyfikowaną wersją TSP, jest ono z nim ściśle powiązane i wymaga podobnych technik rozwiązania.

Zagadnienie problemu posłańca, które do tej pory rozważane było głównie z punktu widzenia codziennej praktyki i logistyki, zaczęło przechodzić fascynującą transformację za sprawą naukowców, takich jak Merrill Flood. Zastosowane przez Flooda podejście do kwestii optymalizacji, pomimo zakorzenienia w konkretnych potrzebach transportowych    
- a konkretniej do próby zastosowania matematycznych metod w celu ulepszenia trasowania autobusów szkolnych - szybko zyskało na znaczeniu w teoretycznych badaniach matematycznych. Zarówno Flood, jak i prace Hasslera Whitneya zaowocowały otwarciem nowego rozdziału w matematycznej teorii optymalizacji. Whitney, znany z wprowadzenia teorii grafów w matematykę, mógł nie być bezpośrednio związany z problemem komiwojażera, ale jego prace niewątpliwie wpłynęły na badania nad problemem komiwojażera, rozwijane później przez Flooda oraz innych naukowców m.in. George Dantziga, który opracował słynny algorytm sympleksowy służący rozwiązywaniu złożonych problemów optymalizacyjnych.

Jednym z bardziej znanych przykładów wczesnego podejścia do optymalizacji tras, które z czasem przekształciło się w badania nad problemem komiwojażera, jest doświadczenie Henry'ego Cleveland z Page Seed Company z 1925 roku. Wspomniany sprzedawca zbierający zamówienia na produkty rolnicze, takie jak kukurydza, był niezadowolony z istniejących planów tras i zdecydował się na wprowadzenie własnych ulepszeń. Jego zoptymalizowana trasa obejmowała 350 przystanków w stanie Maine, rozpoczynając się w Kittery i kończąc w pobliżu Springvale, co zostało przedstawione na rysunku 9. Przebieg tej trasy, który odbył się od 9 lipca do 24 sierpnia 1925 roku, jest uznawany za jedno z pierwszych praktycznych zastosowań koncepcji optymalizacji trasy, będącej zalążkiem późniejszego problemu komiwojażera.

Obraz zawierający mapa, tekst, atlas

Opis wygenerowany automatycznie

Rysunek 9. Zoptymalizowana trasa podróży po stanie Maine z 1925 roku

[*źródło:* (Cook, 2012)]

Opisany przykład pokazuje, jak innowacyjne podejście Cleveland'a do planowania tras mogło stanowić prototyp dla metod, które później znalazłyby zastosowanie w bardziej skomplikowanych algorytmach komiwojażera. Użycie przez niego map, pinezek i sznurków do wizualizacji i planowania trasy było swoistym analogowym narzędziem, które przewidziało późniejsze cyfrowe narzędzia optymalizacji.

Prace George'a Dantziga, Raya Fulkersona i Selmera Johnsona na przełomie lat czterdziestych i pięćdziesiątych XX wieku przyniosły rewolucję w sposobie rozwiązywania problemów komiwojażera dzięki zastosowaniu nowoczesnych algorytmów matematycznych. (Kowalik, 2011) W 1949 roku Dantzig wprowadził do programowania liniowego algorytm simplex, co zmieniło sposób rozwiązywania złożonych problemów optymalizacyjnych. Algorytm ten pozwalał na efektywne znalezienie najlepszego rozwiązania wśród wielu możliwych opcji, co było kluczowe w kontekście zastosowań w planowaniu i zarządzaniu, a także w problemie komiwojażera, gdzie trzeba wybrać najkrótszą możliwą trasę spośród wielu kombinacji. Wraz z Fulkersonem i Johnsonem w 1954 roku, opracował algorytm, który znacząco przyspieszył rozwiązywanie dużych instancji problemów sieciowych, w tym problemu komiwojażera. Ich prace były oparte na metodach dekompozycji, które umożliwiały rozkładanie dużych problemów na mniejsze, łatwiejsze do zarządzania fragmenty. Dało to możliwość do rozwiązywania skomplikowanych problemów logistycznych związanych z optymalizacją tras oraz było fundamentem dla kolejnych pokoleń systemów informatycznych zajmujących się optymalizacją. Mapy, pineski oraz sznurki zamieniono na skomplikowane algorytmy zdolne do zarządzania i optymalizacji tras w dynamicznie zmieniającym się świecie globalnego handlu i logistyki.

Przełomowe badania Danzinga, Fulkersona i Johnsona spowodowały, że świat nauki zaczął poszukiwać kolejnych możliwości usprawnienia podejść do problemu komiwojażera. Pomimo, że algorytm simplex, jak i metody dekompozycji diametralnie przyśpieszyły możliwość rozwiązywania złożonych problemów logistycznych, to wciąż poszukiwane były różnego rodzaju metody, które byłyby zdolne do radzenia sobie z jeszcze większymi instancjami problemu, których złożoność wykraczała poza możliwości nawet najbardziej zaawansowanych technik eklektycznych. Takie zapotrzebowanie sprawiło, że znaczenia zaczęły nabierać algorytmy aproksymacyjne i heurystyczne, które umożliwiały znalezienie rozwiązań dobrych, choć nie zawsze optymalnych, w sposób znacznie szybszy niż metody tradycyjne. (Wikipedia, 2023) Algorytm aproksymacyjny to algorytm znajdujący rozwiązanie problemu optymalizacyjnego, który jest bliski rozwiązaniu optymalnemu, lecz wygenerowane w sposób znacznie szybszy niż przy użyciu dokładnych metod. Algorytmy te stosowane są do problemów, dla których nieznane są algorytmy dokładne m.in. do problemu komiwojażera oraz innych problemów NP-zupełnych. Z kolei algorytmy heurystyczne (Encyklopedia Algorytmów, 2020) to metody obliczeniowe stosowane w celu rozwiązania problemów optymalizacyjnych, gdy problem jest NP-trudny, a znalezienie dokładnego rozwiązania w rozsądnym czasie jest niepraktyczne lub niemożliwe, jednak nie zawsze prowadzą do optymalnego rozwiązania, ale są na ogół wystarczająco dobre w kontekście praktycznych zastosowań. Przykładami algorytmów heurystycznych związanych bezpośrednio z problemem komiwojażera jest algorytm najbliższego sąsiada, algorytmy genetyczne, algorytm mrówkowy. Do kluczowych cech algorytmów heurystycznych zaliczamy szybkość działania, przybliżenie oraz elastyczność. Heurystyki są zaprojektowane tak, aby szybko generować rozwiązania, nawet dla bardzo dużych i złożonych problemów. Algorytmy heurystyczne zwracają rozwiązania, które są bliskie optymalnym, choć nie zawsze są one dokładne, a dzięki zastosowaniu elastyczności mogą być łatwo dostosowywane do konkretnych problemów i zmieniających się warunków.

W erze cyfryzacji zaawansowane obliczenia zaczęły być wykonywane z pomocą wszechobecnego sprzętu komputerowego. Zaczęto opracowywać różnego rodzaju oprogramowania oraz narzędzia, które miały za zadanie sprawdzić czy teoretyczne koncepcje znajdują zastosowanie w praktyce. Jednym z bardziej znanych przykładów oprogramowania specjalistycznego wykorzystującego algorytmy heurystyczne do rozwiązywania problemu komiwojażera jest Condore TSP Solver autorstwa Davida Applegate, Roberta Bixby, Václava Chvátala oraz Williama Johna Cooka. (Wikipedia, 2023) Program Concorde z pomocą różnych technik heurystycznych i dokładnych daje możliwość efektywnego rozwiązania problemu komiwojażera. Oprogramowanie umożliwia odnalezienie najkrótszej trasy, która pozwala odwiedzić listę miast i wrócić do punktu wyjścia, odwiedzając każde miasto tylko raz. Dodatkowo jest w stanie znaleźć trasę, która jest nie tylko teoretycznie optymalna, ale również praktycznie użyteczna w realnych scenariuszach. Na rysunku 10 przedstawiony został interfejs użytkownika programu Concorde. Program ten przedstawia graficznie trasę komiwojażera jako sieć połączonych wierzchołków i krawędzi.

Obraz zawierający tekst, zrzut ekranu, diagram, linia

Opis wygenerowany automatycznie

Rysunek 10. Zrzut ekranu z programu Concorde

[*źródło:* opracowanie własne]

Niebieskie punkty reprezentują miasta lub punkty, które komiwojażer musi odwiedzić, z kolei czerwone linie przedstawiają trasę łączącą te miasta w kolejności, w jakiej powinny być odwiedzane, aby suma przebytych odległości była jak najmniejsza. Na powyższym przykładzie obliczona została trasa dla trasy najbliższego sąsiada, której łączna długość wynosi 1011 jednostek. Następnie program przechodzi do zaawansowanej optymalizacji za pomocą technik programowania liniowego i algorytmów cięcia, które systematycznie redukują długość trasy przez iteracyjne dodawanie ograniczeń matematycznych. Każda kolejna iteracja przynosi poprawę rozwiązania, gdzie długość trasy jest sukcesywnie zmniejszana, aż do osiągnięcia znacznie krótszej trasy wynoszącej 773 jednostki. Przedstawione oprogramowanie ukazuje ewolucję w dziedzinie algorytmów, przekształcając teoretyczne koncepcje w nowoczesne potężne aplikacje, z pomocą których użytkownicy mogą w efektywny sposób spróbować swoich sił z jednymi z najbardziej złożonych wyzwań optymalizacyjnych spotykanych w praktyce. (Wikipedia, 2024) Program Concorde został wykorzystany w maju 2004 roku do wyznaczenia najkrótszej trasy dla wszystkich 24 978 miast w Szwecji, czego rezultatem było otrzymanie trasy mierzącej 72 500 kilometrów. Udowodniono również, że krótsza droga nie istnieje. W 2005 ustanowiono kolejny rekord, tym razem polegał on na odwiedzeniu 33 810 punktów na płytce drukowanej za pomocą programu Concorde i oszacowano, że trasa wyniosła 66 048 945 jednostek oraz ponownie udowodniono, że krótsza droga nie istnieje. (Traveling Salesman Problem, 2015) Z kolei w 2006 rozwiązano problem komiwojażera dla 85 900 punktów co stanowiło ówcześnie rekord świata. Rysunek 11

Obraz zawierający tekst, czarne i białe

Opis wygenerowany automatycznie

Rysunek 11. Rozwiązanie TSP dla 85 900 miast w zastosowaniu chipa komputerowego.

[*źródło:* (Cook, 2012)]

Z kolei rysunek 12 przedstawia zbliżenie małego obszaru części trasy obejmującej 85 900 miast w sposób prostszy do obserwacji.

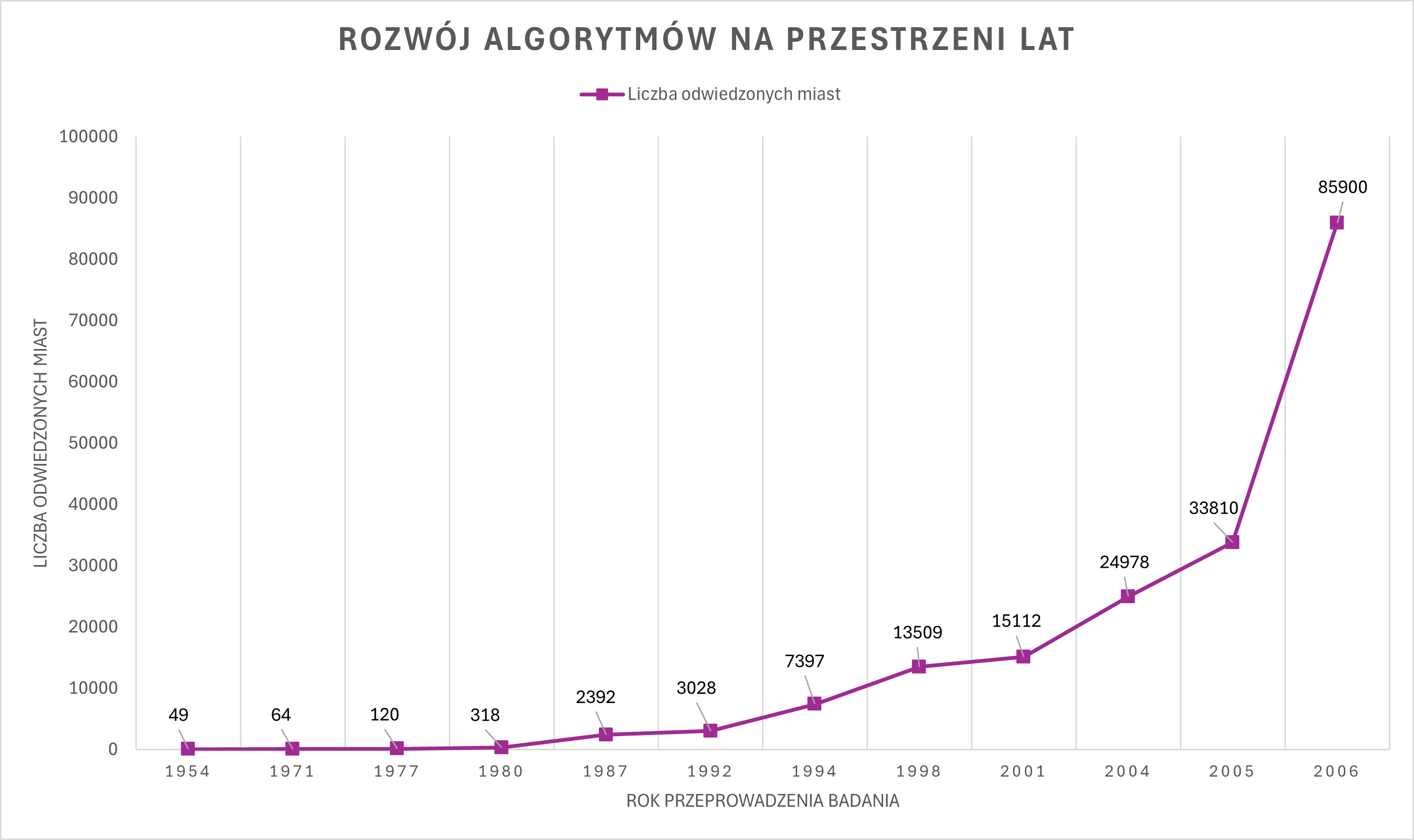
Obraz zawierający tekst, linia, Czcionka

Opis wygenerowany automatycznie

Rysunek 12. Przedstawienie małego obszaru trasy obejmującej 85 900 iast

[*źródło:* (Cook, 2012)]

Rysunek 12 prezentuje rozwój algorytmów na przestrzeni lat. (Cook, 2012) Wykres rozpoczyna się od ręcznie opracowanego przez Danzinga w 1954 roku przykładu dla 49 miast, aż do ostatniego rozwiązania z 2006 roku.



Rysunek 13. Rozwój algorytmów na przestrzeni lat

[*źródło:* opracowanie własne na podstawie(Cook, 2012)]

(World Traveling Salesman Problem, 2021) W 2007 roku duński informatyk Keld Helsgaun z wykorzystaniem programu Concorde w połączeniu z solverem[[3]](#footnote-3) programowanie CPLEX[[4]](#footnote-4) ustanowił nowy wynik najlepszego dolnego ograniczenia długości trasy, która wyniosła 7 512 218 268 metrów, co uświadamia, że trasa Helsguna jest zaledwie o 0,0471% dłuższa niż optymalna trasa. W 2021 roku Keld Helsgaun wykorzystując własny algorytm heurystyczny LKH osiągnął nowy rekord długości trasy, który wyniósł 7 515 755 956 metrów. (Strąk, 2017) Algorytm heurystyczny Lin-Kernighan-Heuristic jest jednym z najbardziej efektowych algorytmów heurystycznych do rozwiązywania problemu komiwojażera. Jest to ulepszona wersja algorytmu Lin-Kernighan[[5]](#footnote-5) zawierajaca techniki opracowane przez Kelda Helsgauna, które poprawiają wydajność oraz jakość znajdowanych rozwiązań. LKH wykorzystuje metody lokalnego przeszukiwania, aby iteracyjnie poprawiać istniejące rozwiązanie, zamieniając segmenty trasy w taki sposób, aby uzyskać jak najkrótszą możliwą ścieżkę łączącą wszystkie punkty.

Problem komiwojażera mimo swojej pozornej prostoty znajduje szerokie zastosowanie w różnych dziedzinach, co potęguje jego znaczenie zarówno w teorii, jak i w praktyce. Przede wszystkim problem ten wykorzystywany jest w logistyce oraz planowaniu tras, gdzie pomaga w optymalizacji tras dostaw, minimalizując czas podróży i koszty paliwa. Wykorzystywany jest również między innymi w bioinformatyce do sekwencjonowania DNA, gdzie kolejność fragmentów DNA musi być ułożona w możliwie najbardziej efektywny sposób. Problem komiwojażera obecny jest również w dziedzinie robotyki, na przykład w przypadku automatycznych magazynów oraz systemach produkcji. W wspomnianych sektorach roboty wykorzystują algorytmy bazujące na problemie TSP do optymalizacji ścieżek ruchu, w celu wykonywania zadań efektywnie i z minimalnym zużyciem energii. Wspomniane przykłady stanowią tylko część przykładowych praktycznych zastosowań. W związku z ciągle rozwijającą się technologią, szczególnie w dziedzinie uczenia maszynowego, a także sztucznej inteligencji istnieje szansa na powstanie nowych metod rozwiązywania tego NP-trudnego problemu. Algorytmy uczenia maszynowego mogą być wykorzystywane do tworzenia bardziej efektywnych heurystyk, które mogą nauczyć się wzorców i optymalizować decyzje w bardziej dynamicznych i nieprzewidywalnych środowiskach. Jednym z głównych wyzwań na przyszłość jest poprawa skalowalności algorytmów do bardzo dużych instancji problemów. Istnieje również możliwość, że współpraca pomiędzy różnymi dziedzinami nauki może dać możliwość wprowadzenia nowych zastosowań problemu komiwojażera.

## Opis problemu

(Sysło, Deo i Kowalik, 1995) Problem komiwojażera stanowi klasyczny problem optymalizacji kombinatorycznej. Polega na odnalezieniu najkrótszej możliwej drogi, w której istnieje możliwość odwiedzenia wszystkich dostępnych miast wyłącznie jeden raz, powracając na samym końcu do punktu startowego. (Cormen, Leiserson, Rivest i Stein, Wprowadzenie do algorytmów, 2018) Problem TSP ma wiele zastosowań praktycznych np. w logistyce do planowania tras flot pojazdów, czy też planowania wycieczek. Stosowany jest również przy optymalizacji ścieżek narządzi na liniach produkcyjnych w dziedzinie produkcji, czy też po pewnych modyfikacjach do sekwencjonowania DNA, gdzie opisane miasto zastępuje się fragmentami DNA, natomiast odległość zastąpiona jest funkcją podobieństwa między poszczególnymi fragmentami materiału genetycznego DNA.

(Sysło, Deo i Kowalik, 1995) W wariancie decyzyjnym problemu komiwojażera dany jest nieskierowany graf pełny posiadający nieujemne i całkowite wagi k krawędzi. Graf ten można przedstawić jako macierz wag . W tym przypadku symbolizują wagę krawędzi pomiędzy wierzchołkami oraz . (Cormen, Leiserson, Rivest i Stein, Wprowadzenie do algorytmów, 2007) Problem komiwojażera stanowi klasyczny przykład problemu NP-trudnego. Certyfikat w problemie decyzyjnym TSP składa się z wierzchołków cyklu, wymienionych po kolei. Za pomocą czasu wielomianowego można w prosty sposób zweryfikować czy krawędziami danego cyklu istnieje możliwość dotarcia do wszystkich wierzchołków oraz czy ich waga łącznie wynosi k bądź mniej. Fakt, że problem komiwojażera jest NP-trudny można zaprezentować za pomocą redukcji problemu cyklu Hamiltona, gdzie posiadając na wejściu graf G należy skonstruować graf pełny G’, który posiada te same wierzchołki co poprzednik. Kolejny krok stanowi nadanie krawędzią (u,v) w nowo utworzonym grafie wagi 0 w przypadku, gdy (u,v) są krawędzią w grafie G lub wagi 1 jeżeli w poprzedniku taka krawędź nie występuje.

Dla przykładu można założyć, że dane jest pięć miast A, B, C, D oraz E. Odległości pomiędzy nimi przedstawione zostały w tabeli 1. Kolorem błękitnym zaznaczone zostały odległości pomiędzy poszczególnymi miastami.

Tabela 1.

Odległości pomiędzy miastami A, B, C, D, E.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E |
| A | - | 2 | 9 | 10 | 7 |
| B | 2 | - | 6 | 4 | 3 |
| C | 9 | 6 | - | 8 | 3 |
| D | 10 | 4 | 8 | - | 1 |
| E | 7 | 3 | 3 | 1 | - |

[*źródło:* opracowanie własne]

W tym przykładzie należy odnaleźć najkrótszą trasę rozpoczynającą się w jednym z miast, odwiedzając przy tym wszystkie możliwe miasta wyłącznie jeden raz, wracając na końcu do miasta, w którym podróż została rozpoczęta. Jednym z możliwych rozwiązań jest trasa , której całkowita długość wynosi .

Problem komiwojażera rozróżniany jest jako dwie klasy problemu:

* Symetryczny problem komiwojażera
* Asymetryczny problem komiwojażera

(Wikipedia, 2024) Symetryczny problem komiwojażera (STSP – Symmetric Travelling salesman problem) to wariant klasycznego problemu TSP. Klasa ta cechuje się tym, że odległości pomiędzy dowolnymi miastami A i B są identyczne w obydwie strony. Macierz odległości jest symetryczna.

(Wikipedia, 2024) Asymetryczny problem komiwojażera (ATSP – Asymmetric Travelling salesman problem) jest to druga klasa problemu TSP, która charakteryzuje się tym, że odległość pomiędzy miastem A oraz B jest inna niżeli z miasta B do miasta A.

Sporym wyzwaniem dla problemu komiwojażera stanowi bardzo duża ilość danych, które należy poddać analizie. (Wikipedia, 2024) Odległość w TSP pełni zdecydowanie kluczową rolę przez co można go podzielić na problem metryczny oraz problem niemetryczny. Problem metryczny stanowi wariant, w którym odległości pomiędzy miastami muszą spełniać nierówność trójkąta. Nierówność trójkąta głosi, że dla trzech dowolnych miast A, B oraz C, odległość pomiędzy miastem A i miastem C jest mniejsza lub równa sumie odległości pomiędzy miastem A oraz B oraz B i C. Formalnie za pomocą wzoru można zapisać to w następujący sposób:

W powyżej przedstawionym wzorze X można zaobserwować, że nierówność trójkąta została spełniona oraz odległości są symetryczne.

Z kolei w przypadku problemu niemetrycznego odległości pomiędzy miastami nie mają narzuconego warunku spełniania nierówności trójkąta tj. może istnieć przypadek, gdzie odległość bezpośrednia między danymi dwoma miastami jest większa niż suma odległości przez inne możliwe miasto.

Rozważając możliwe koszty przejazdu możliwymi środkami transportu np. koleją czy też z pomocą linii lotniczych, może wystąpić sytuacja, w której koszt przejazdu koleją odwiedzając przy tym dodatkowe miasta może być niższy niżeli bezpośredni lot samolotem. Z tego powodu można wyróżnić następujące rodzaje metrycznych problemów komiwojażera:

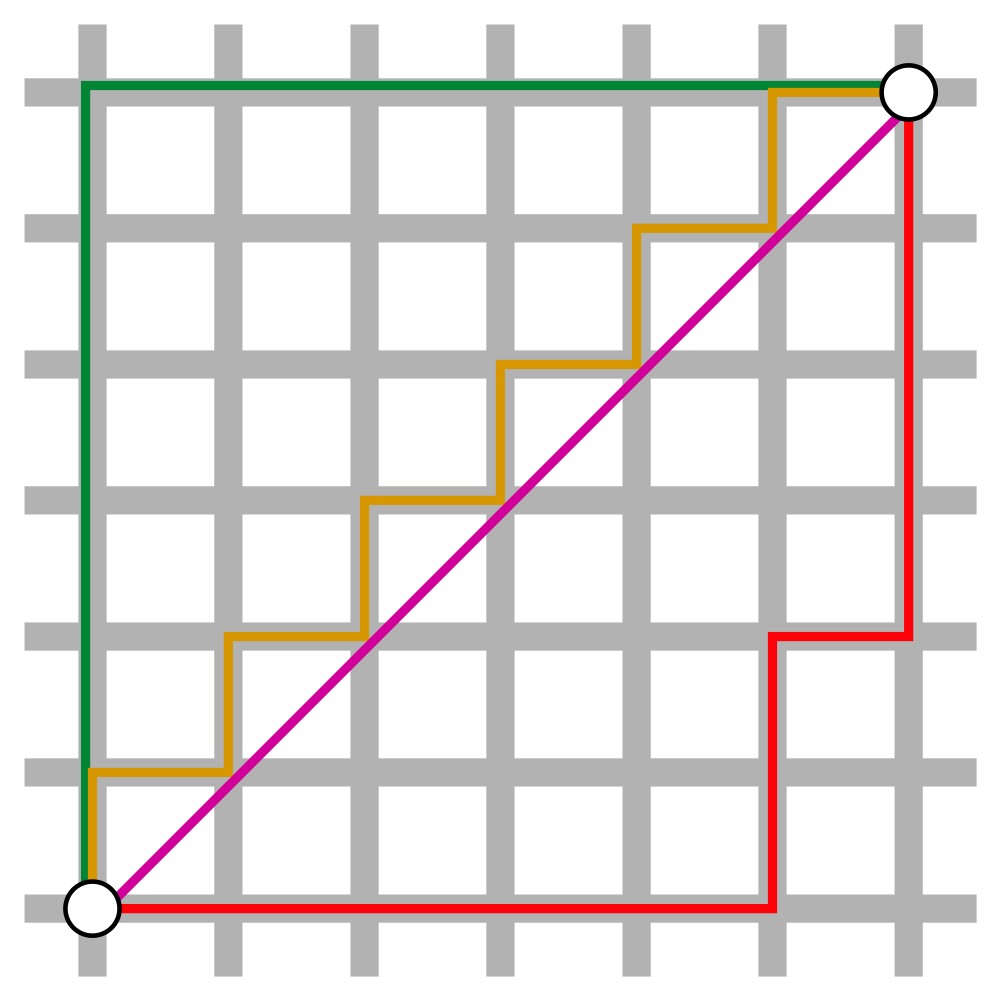
* Metryka euklidesowa;
* Metryka Manhattan,

(Wikipedia, 2024) Metryka Euklidesowa stanowi klasyczną miarę odległości pomiędzy dwoma punktami w przestrzeni euklidesowej. W odniesieniu do problemu komiwojażera metryka ta stosowana jest w celu obliczenia rzeczywistej odległości między miastami na płaszczyźnie. Dla dwóch punktów oraz , odległość euklidesowa d określana jest wzorem:

(Wikipedia, 2024) Metryka Manhattan, nazywana również odległością miejską bądź siatką taksówkarską odpowiada za pomiar odległości między dwoma punktami rozmieszczonymi w przestrzeni jako suma bezwzględnych różnic ich współrzędnych. W przypadku problemu komiwojażera wykorzystuje się ją do obliczeń odległości w przypadkach, gdzie ruch odbywa się wzdłuż prostych osi, tak jak w siatce miejskiej. Dla punktów oraz   
, odległość Manhattan d wyznacza się za pomocą ogólnego wzoru:

Odległość Manhattan można również przedstawić jako dwuwymiarową przestrzeń za pomocą wzoru:

(abcdef.wiki, 2024) W kontekście problemu komiwojażera, w którym odległość pomiędzy punktami wyliczana jest za pomocą metryki Manhattan, trasa minimalizuje odległość wzdłuż osi poziomych oraz pionowych. Cecha ta jest korzystna w środowiskach miejskich, w których przemieszczanie się pomiędzy punktami odbywa się po ulicach tworzących siatkę. Jako ciekawostkę warto dodać, że nazwa metryki miejskiej tzw. metryki Manhattan wywodzi się od nowojorskich taksówek poruszających się po ulicach miasta oraz wykorzystywana była do zminimalizowania czasu przejazdu stosując ograniczenie w ruchu w kierunkach północ-południe oraz wschód-zachód. Z kolei w przypadku obliczeń odległości za pomocą metryki euklidesowej, trasa dokonuje minimalizacji rzeczywistych odległości w przestrzeni dwuwymiarowej. Na rysunku X przedstawione zostało porównanie omówionych metryk. Kolorem fioletowym oznaczona została metryka euklidesowa, natomiast pozostałymi kolorami tj. zielonym, pomarańczowym oraz czerwonym zaznaczona została metryka miejska.



Rysunek 14. Porównianie metryki Manhattan oraz metryki euklidesowej

[*źródło:* opracowanie własne na podstawie (Wikipedia, 2024)]

Metryki Manhattan zgodnie z założeniami mają takie same odległości, z kolei metryka euklidesowa oznaczona na fioletowo posiada najkrótszą długość.

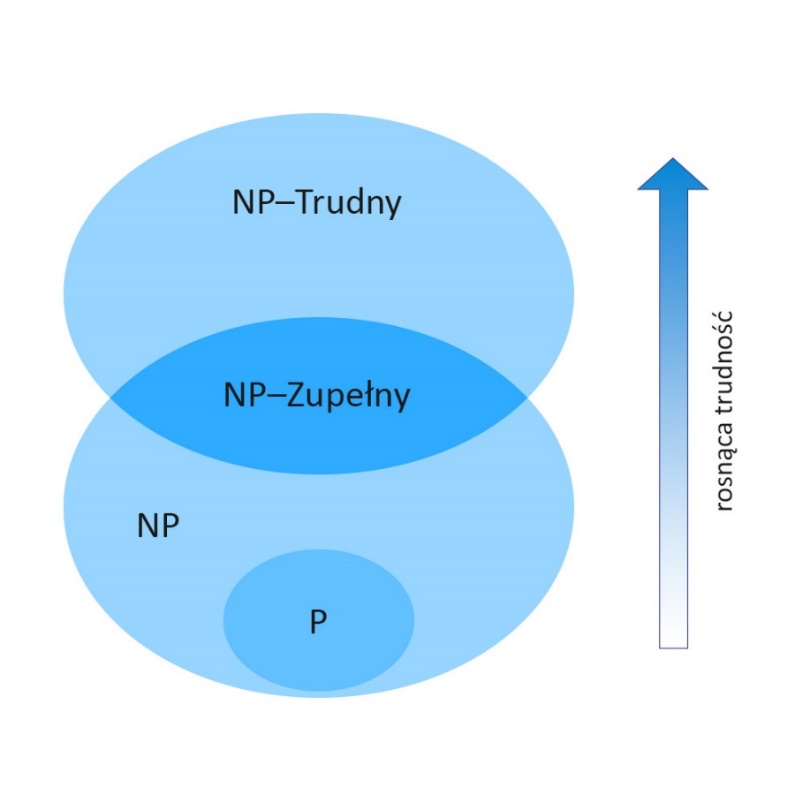
## Rodzaje problemów

(Wikipedia, 2024) Problemy obliczeniowe klasyfikujemy w różnego rodzaju klasy złożoności. Klasy te są charakteryzowane poprzez trudność rozwiązywania problemów pod względem różnego rodzaju zasobów m.in. czasu bądź pamięci niezbędnej do rozwiązania tych problemów. Rozważany w tej pracy problem komiwojażera zalicza się do klasy NP-trudnych. Oznacza to, że jest co najmniej tak trudny, jak najtrudniejsze problemy w tej klasie oraz nie istnieje algorytm działający w czasie wielomianowym, który znalazłby optymalne rozwiązanie problemu komiwojażera. (Brilliant.org, 2024) NP-trudność uświadamia, że dany problem nie należy do klasy NP, charakteryzującej się występowaniem problemów, w których można szybko sprawdzić poprawność rozwiązania, lecz każdy problem klasy NP z pomocą transformacji działającej w czasie wielomianowym można do niego sprowadzić.

(Ladner, 1975) Istnieje wiele klas złożoności problemów obliczeniowych. Zaliczamy do nich następujące klasy:

* Klasa P (Polynomial) – jest to najmniejsza klasa problemów, które można rozwiązać w czasie wielomianowym poprzez deterministyczną maszynę Turinga[[6]](#footnote-6).
* Klasa NP (Nondeterministic Polynomial time) – Klasa ta zawiera problemy decyzyjne, które można rozwiązać w czasie wielomianowym z pomocą niedeterministycznej maszyny Turinga, bądź równoważnie, których rozwiązania mogą być zweryfikowane w czasie wielomianowym przez deterministyczną maszynę.
* Klasa NP-Zupełne (Nondeterministic Polynomial Complete) – stanowi podzbiór klasy NP, w którym umiejscowione zostały najtrudniejsze problemy klasy złożoności obliczeniowej, ze względu na brak możliwości odnalezienia rozwiązania problemu w czasie wielomianowym.
* Klasa NP-Trudne (Nondeterministic Polynomial Hard) – stanowi zbiór problemów co najmniej tak trudnych, jak najtrudniejsze problemy klasy NP. W odróżnieniu od problemów klasy Nondeterministic Polynomial nie są to koniecznie problemy decyzyjne, ponieważ mogą być trudniejsze niż NP, oraz obejmować problemy spoza NP.

Na rysunku 15 zaprezentowana została hierarchia problemów decyzyjnych w teorii złożoności obliczeniowej, ukazując względne położenie i związki pomiędzy klasami problemów. Strzałka po prawej stronie obrazuje wzrastającą trudność problemów w ramach przedstawionych klas od najłatwiejszej P, poprzez NP, aż do najtrudniejszych NP-Trudnych.



Rysunek 15. Hierarchia problemów decyzyjnych

[*źródło:* opracowanie własne na podstawie (Brilliant.org, 2024)]

# Podstawowe pojęcia

W rozdziale drugim przedstawione zostały podstawowe pojęcia, z którymi warto się zapoznać w celu głębszego zrozumienia problemu. (tu nw co napisać)

## Graf

(Wojciechowski i Pieńkocz, 2013) Graf to uporządkowana para zbioru wierzchołków oraz krawędzi , wykorzystywana do badania relacji między obiektami. Graf musi posiadać przynajmniej jeden wierzchołek , a także przecięcie zbiorów wierzchołków oraz zbioru krawędzi jest w grafie zbiorem pustym . Oznacza to, że ma żadnych elementów wspólnych pomiędzy wierzchołkami a krawędziami grafu, co jest zgodne z definicją grafu, w którym wierzchołki i krawędzie są traktowane jako oddzielne, niezależne zbiory. Krawędź grafu stanowi uporządkowana para wierzchołków , , . (Wilson, 2000) W najprostszych słowach, to zbiór punktów, które łączymy liniami tak, że każda linia zaczyna się i kończy na jednym z punktów. (Wojciechowski i Pieńkocz, 2013)W przypadku, gdy w zbiorze krawędzi występują powtarzające się elementy, to w takiej sytuacji krawędzie nazywamy równoległymi lub wielokrotnymi. Liczbę wierzchołków grafu oznaczamy jako , natomiast liczbę krawędzi jako . Zapis ten wykorzystujemy do klasyfikacji grafów skończonych, w których oraz są liczbami skończonymi. W przypadku, gdy oraz krawędź nazywamy krawędzią niezorientowaną, zwaną również nieskierowaną i oznaczamy symbolem . W przypadku, gdy oraz , krawędź nazywamy łukiem, bądź krawędzią zorientowaną   
lub skierowaną. (Wilson, 2000) Można nadać poszczególnym wierzchołka numeracje, a także mogą być reprezentacją obiektów, natomiast krawędzie stanowić relację pomiędzy nimi. W przypadku, gdy krawędzie posiadają kierunki to mowa o grafie skierowanym. Krawędzie mogą posiadać wagi reprezentujące m.in. odległość pomiędzy poszczególnymi wierzchołkami. (Wikipedia, 2024) Według dostępnych informacji pierwszym badaczem grafów przyjęty został Leonard Euler, który wykorzystał grafy w swojej teorii dotyczącej mostów królewieckich, natomiast przyjęcie słowa „graf” przypadło Jamesowi Josephowi Sylvesterowi w 1878 roku.

## Cykl Eurela

(Weisstein, 2024) Cykl Eurela stanowi trasę w grafie rozpoczynającą się i kończącą na tym samym wierzchołku, przekraczając każdą z krawędzi wyłącznie raz. Pojęcie cyklu Eurela wynika z pracy autora nad problemem mostów królewieckich, co zostało przedstawione w genezie problemu. Tym samym udowodnił, że aby graf mógł posiadać cykl Eurela, każdy z jego wierzchołków powinien posiadać parzystą ilość krawędzi. W przypadku stopnia parzystego graf spełnia warunek przejścia przez każdą z krawędzi wyłącznie raz, bez zatrzymania się w jakimkolwiek wierzchołku. Atutem cyklu Eurela jest fakt, że żadna z krawędzi grafu nie jest pomijana, obejmując w ten sposób cały graf powracając do punktu startowego, co stanowi kluczową cechę m.in. w aplikacjach do projektowania efektywnych tras, bez konieczności powtarzania tych samych tras.

## Twierdzenie Eurela

(Wilson, 2000) Twierdzenie Eurela obwieszcza, że dla każdego spójnego grafu planarnego, oznaczonego symbolem , który można zilustrować na płaszczyźnie bez przecinających się krawędzi, spełniona zostaje równość:

(8)

Gdzie:

* oznacza liczbę wierzchołków grafu
* oznacza liczbę krawędzi grafu
* oznacza liczbę ścian grafu , włączając w to ścianę zewnętrzną

Dowód twierdzenia wykorzystuje indukcję matematyczną względem liczby wierzchołków bądź krawędzi w grafie. W podstawowym założeniu dla grafów z jednym wierzchołkiem bez krawędzi ściana zewnętrzna jest jedyną ścianą, a co za tym idzie równość zostaje spełniona. Kolejny krok to rozważenie przypadków w sytuacji, gdy dodane zostały kolejne krawędzie doprowadzając do analizy ilości ścian. W przypadku, gdy graf jest acyklicznym grafem spójnym, czyli innymi słowy drzewem to równość także zachodzi, ze względu na to, że drzewo z wierzchołkami ma zawsze krawędzi, a także jedną ścianę zewnętrzną. Refleksje na temat grafów nie będącymi drzewami obejmują schematy, gdzie grafy zawierają cykle, a co za tym idzie dodatkowe ściany wewnętrzne. W takich sytuacjach usunięcie krawędzi z cyklu doprowadza do zmniejszenia liczby ścian o jedną, dzięki czemu równość zarówno w przypadku dodatnia, jak i usunięcia krawędzi zostaje zachowana.

## Graf eurelowski i półeurelowski

(Wilson, 2000) Graf eurelowski jest to spójny graf, w którym istnieje możliwość odnalezienia cyklu Eurela. Proces ten stanowi zamkniętą trasę przechodzącą przez każdą krawędź grafu wyłącznie jeden raz, rozpoczynając się i kończąc w tym samym punkcie, co stanowi kluczowy wymóg cyklu Eurela. Kluczowym warunkiem grafu eulerowskiego jest stopień wierzchołków w grafie, który musi być parzysty. Oznacza to, że do każdego z wierzchołków musi wchodzić parzysta ilość krawędzi. Przykładowy graf eurelowski przedstawiony został na rysunku 11.

Obraz zawierający linia, krąg, Symetria

Opis wygenerowany automatycznie

Rysunek 16. Graf eurelowski

[*źródło:* opracowanie własne na podstawie (Wilson, 2000)]

W przeciwnym przypadku jest to graf półeurelowski, który zezwala na istnienie dwóch wierzchołków o stopniu nieparzystym, a co za tym idzie istnieje możliwość rozpoczęcia i zakończenia ścieżki w dwóch różnych wierzchołkach, przechodząc przez każdą z krawędzi wyłącznie raz. Przykład takiego grafu został przedstawiony na rysunku 12.

Obraz zawierający linia, krąg

Opis wygenerowany automatycznie

Rysunek 17. Graf półeurelowski

[*źródło:* opracowanie własne na podstawie (Wilson, 2000)]

## Cykl Hamiltona

(Cormen, Leiserson, Rivest i Stein, Wprowadzenie do algorytmów, 2018)Cyklem Hamiltona nazywamy ścieżkę w grafie, która przechodzi przez wszystkie wierzchołki grafu wyłącznie jeden raz, rozpoczynając i kończąc w tym samym punkcie, tworząc w ten sposób zamknięty cykl. Formułując cykl Hamiltona matematycznie dla grafu zawierającym zbiór wierzchołków oraz zbiór krawędzi , cykl opisuje się jako permutację wierzchołków w następujący sposób:

dla wszystkich , (9)

, zamykając cykl (10)

W równaniu 9 wzór informuje, że dla każdego wierzchołka istnieje krawędź, która łączy go bezpośrednio z następnym wierzchołkiem w sekwencji cyklu. Oznacza to, że z każdego wierzchołka cyklu istnieje możliwość bezpośredniego przejścia, aż do przedostatniego.   
Równanie 10 odnosi się do ostatniego w cyklu wierzchołka , a także do pierwszego wierzchołka . Cykl zostanie zamknięty tylko wtedy, kiedy pomiędzy tymi wierzchołkami istnieje krawędź. W sytuacji, gdy wszystkie wierzchołki zostaną odwiedzone, ostatni musi być bezpośrednio połączony z pierwszym wierzchołkiem, tworząc zamkniętą pętlę.

(Wojciechowski i Pieńkocz, 2013)W przypadku grafu niezorientowanego, cykl Hamiltona przechodzi przez wszystkie wierzchołki wyłącznie raz, a na końcu powraca do punktu początkowego. Z kolei w grafie zorientowanym ścieżka przechodzi przez skierowane krawędzie, zachowując kierunkowość grafu.

## Graf Hamiltonowski

(Wilson, 2000) Graf Hamiltonowski to graf spójny, w którym istnieje cykl hamiltonowski przechodzący przez każdy z wierzchołków wyłącznie jeden raz. Cykl rozpoczyna i kończy bieg w tym samym wierzchołku. Na rysunku 13 znajdującym się poniżej przedstawiony został przykładowy graf hamiltonowski.

Obraz zawierający linia

Opis wygenerowany automatycznie

Rysunek 18. Graf hamiltonowski

[*źródło:* opracowanie własne na podstawie (Wilson, 2000)]

(Wikipedia, 2024) W 1952 roku węgiersko-angielski matematyk Gabriel Andrew Dirac opracował twierdzenie mogące stwierdzić, czy graf jest Hamiltonowski. (Wilson, 2000) Dirac podał bezpośredni warunek, który głosił, że w przypadku, gdy w grafie prostym G każdy z wierzchołków, gdzie , graf jest Hamiltonowski w momencie, gdy dla każdego wierzchołka v zachodzi następujące równanie:

(11)

Gdzie:

* wyraża stopień wierzchołka v w grafie,
* – oznacza ilość wierzchołków w grafie
* – oznacza połowę wierzchołków w grafie

Warunek Diraca głosi, że w przypadku, gdy każdy wierzchołek w grafie prostym ma stopień co najmniej równy połowie wierzchołków w grafie, to taki graf posiada cykl Hamiltona. Twierdzenie to pozwala na szybką weryfikację potencjalnej hamiltonowskości grafu, bez konieczności przeszukiwania wszystkich możliwych ścieżek.

Natomiast według twierdzenia norweskiego matematyka Øysteina Orego z 1960 roku, w przypadku, kiedy graf prosty ma wierzchołków, gdzie oraz dla każdej pary niesąsiadujących wierzchołków i suma stopni spełnia warunek opisany następującym wzorem:

(12)

W równaniu numer 12 oznacza stopnie dowolnych niesąsiadujących ze sobą wierzchołków oznaczonych jako oraz . Symbol stanowi ilość wierzchołów znajdujących się w danym grafie. W przypadku, gdy warunek zostanie spełniony i suma stopni każdej pary niepołączonych ze sobą krawędzią wierzchołków jest równa lub większa od ilości wierzchołków w grafie, to według twierdzenia Ore graf jest hamiltonowski. Dowód Twierdzenia Ore oparty jest na dodaniu wszystkich krawędzi i analizie zmian w stopniach wierzchołków, doprowadzając tym samym do konstrukcji cyklu Hamiltona. Matematycznie cykl Hamiltona w grafie G można przedstawić jako ciąg wierzchołków:

(13)

W ciągu przedstawionym w równaniu 13 każdy z wierzchołków ciągu odwiedzany jest wyłącznie jeden raz, natomiast sam cykl kończy się w tym samym punkcie, w którym rozpoczął, co spełnia zamkniętość cyklu.

W sytuacji, gdy graf nie zawiera cyklu Hamiltona, jednak istnieje w nim ścieżka przechodząca przez każdy wierzchołek wyłącznie raz, jednak nie tworzy ona zamkniętego cyklu to taki graf nazywamy grafem półhamiltonowskim. Przykład takiego grafu przedstawiony został na rysunku 14.

Obraz zawierający linia

Opis wygenerowany automatycznie

Rysunek 19. Graf półhamiltonowski

[*źródło:* opracowanie własne na podstawie (Wilson, 2000)]

Z kolei grafem niehamiltonowskim jest graf, w którym nie istnieje cykl obejmujący każdy z wierzchołków wyłącznie jeden raz. Przykład takiego grafu został przedstawiony na poniższym rysunku 15.

Obraz zawierający linia

Opis wygenerowany automatycznie

Rysunek 20. Graf niehamiltonowski

[*źródło:* opracowanie własne na podstawie (Wilson, 2000)]

## Problem cyklu Hamiltona

(Cormen, Leiserson, Rivest i Stein, Wprowadzenie do algorytmów, 2018) Problem cyklu Hamiltona poleca na odnalezieniu ścieżki grafu nieskierowanego, odwiedzającej każdy wierzchołek wyłącznie jeden raz, zamykając tym samym cykl poprzez powrót do wierzchołka początkowego. Problem można zdefiniować jako język formalny HAM-CYCLE= {G: G jest grafem hamiltonowskim}.   
Jest to zbiór wszystkich grafów G, które posiadają cykl Hamiltona. HAM-CYCLE określa nazwę rozwiązywanego problemu. {G} symbolizuje zbiór wszystkich grafów. Zapis G jest grafem hamiltonowskim stanowi kryterium, które muszą spełniać grafy należące do zbioru HAM-CYCLE, innymi słowy każdy graf musi posiadać cykl Hamiltona.

Problem cyklu Hamiltona zaliczany jest do problemów NP-zupełnych. Innymi słowy każdy problem NP można sprowadzić do problemu cyklu Hamiltona w czasie wielomianowym, przez co jest on jednym z najtrudniejszych w teorii obliczeń. Dla instancji problemu   
HAM-CYCLE, algorytm weryfikacji wymaga dokonania sprawdzenia permutacji wierzchołków oraz przeprowadzenia badania, czy któraś z nich stanowi cykl Hamiltona. Czas działania algorytmu weryfikacji jest ekspotencjalny względem liczby wierzchołków grafu, z powodu przeszukiwania permutacji.

Istnieje wiele technik rozwiązania problemu cyklu Hamiltona m.in. poprzez użycie technik redukcji problemu NP-zupełnego. Przykładem redukcji problemu jest VERTEX-COVER. (Wikipedia, 2024) Pokrycie wierzchołkowe grafu stanowi zbiór wierzchołków wybranych tak, aby każda z krawędzi była incydentalna z przynajmniej jednym z nich. Znajdowanie minimalnego pokrycia wierzchołkowego stanowi typowy NP-trudny problem optymalizacyjny, co sugeruje, że nie istnieje algorytm wielomianowy, który byłby w stanie go rozwiązać, w przypadku, gdy P NP. Problem ten jest również trudny do aproksymacji, ale istnieją proste algorytmy aproksymacyjne, które osiągają współczynnik 2.

(Cormen, Leiserson, Rivest i Stein, Wprowadzenie do algorytmów, 2018) Definicja problemu VERTEX-COVER głosi, że dany jest graf , a także liczba k wierzchołków, które należy dobrać w taki sposób, aby każda z krawędzi grafu była incydentalna z przynajmniej jednym wierzchołkiem. Aby dokonać redukcji VERTEX-COVER do HAM-CYCLE należy skonstruować nowy graf  powstały z grafu . Graf posiada cykl Hamiltona wtedy i tylko wtedy, gdy graf posiada pokrycie wierzchołkowe o rozmiarze , bądź mniejszym. Na początku dla każdego z wierzchołków grafu należy skonstruować gadżet w ’. Gadżet będzie składał się z nowych wierzchołków oraz krawędzi, których celem jest wymuszenie przejścia cyklu wyłącznie przez określone wierzchołki, symulując tym samym wybór wierzchołka w pokryciu. Następnie między gadżetami w grafie należy umieścić krawędzie, które będą odpowiadały krawędzią w . W przypadku wybrania wierzchołka w pokryciu w grafie , krawędzie łącząc gadżety w zostaną połączone w taki sposób, aby możliwe było utworzenie cyklu Hamiltona. Można to wyrazić wzorem Oznacza to, że problem pokrycia wierzchołkowego jest redukowalny w sensie wielomianowym do problemu cyklu Hamiltona. Metoda ta jest kluczowa w badaniach nad NP-zupełnością.

# Przegląd algorytmów

Zmiązane z cyklami Hamiltona albo komiwojażera. skupic się na cyklowych a nie drogi

## Algorytmy k-optymalne.

(Anholcer, 2023) Algorytmy k-optymalne stanowią jedną z powszechnie znanych metod heurystycznych, służących do rozwiazywania problemu komiwojażera. Zadaniem wspomnianych heurystyk jest przeszukiwanie lokalne rozwiązania za pomocą iteracyjnych wymian k tras powstałym wcześniej rozwiązaniu, podejmując próbę odnalezienia rozwiązania o niższym koszcie. W przypadku tych algorytmów należy rozważyć minimum miast, ze względu na fakt, iż tylko w takim przypadku istnieje możliwość odnaleźć k rozłącznych tras. Kiedy liczba miast n jest równa k, wtedy istnieje możliwość odnalezienia dokładnego rozwiązania. (Encyklopedia Algorytmów, 2017) Procedura algorytmu składa się z trzech kroków: usunięcia krawędzi, sprawdzenia połączeń oraz zapamiętania najlepszego rozwiązania. Na początek dokonuje się usunięcia krawędzi z cyklu k oraz zastąpienia ich innymi wybranymi krawędziami, w celu utworzenia prawidłowego cyklu. Usuwając krawędzie dokonywany jest podział cyklu na ilość k fragmentów. (Anholcer, 2023) Przykładowo dla cyklu:

, (X)

gdzie oraz symbolizują fragmenty cyklu, natomiast oraz to krawędzie, które zostaną usunięte. Kolejny krok to dokonanie analizy każdego możliwego połączenia powstałych wcześniej fragmentów oraz sprawdzenie czy taki wariant będzie lepszy od aktualnego. Dla przedstawionego wcześniej matematycznie przykładu istnieją dwie możliwości przekształcenia cyklu:

, (X)

(X)

W równaniu X symbolizuje odwrócony fragment , natomiast w równaniu X to odwrotność fragmentu . W przypadku, gdy nowe rozwiązanie jest optymalniejsze od odnalezionego wcześniej, należy dokonać zapisu oraz kontynuować proces. (Encyklopedia Algorytmów, 2017) Złożoność algorytmu określa się za pomocą liczby możliwych wariantów przekształceń cyklu co wyrażone jest wzorem:

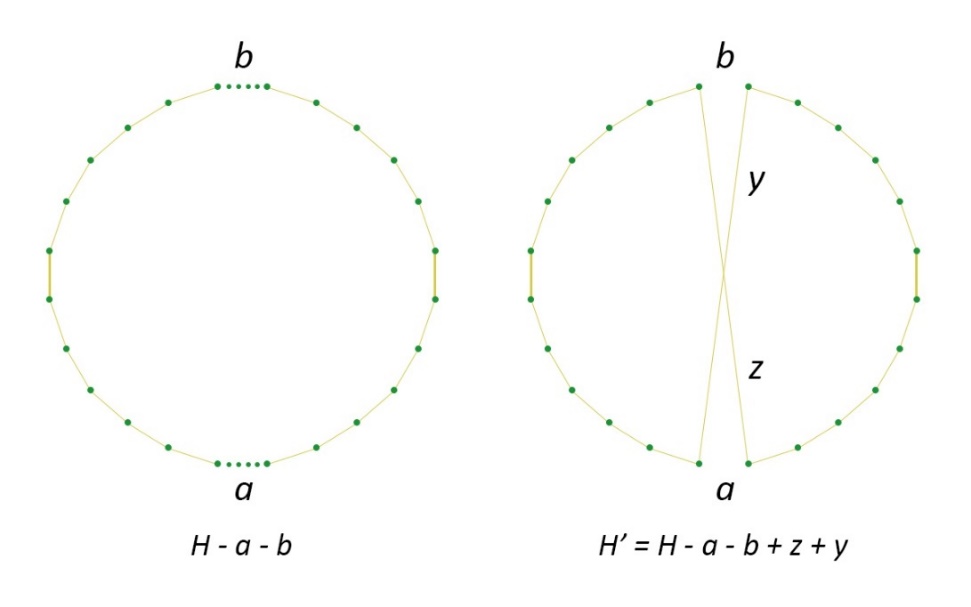
Gdzie:

* – symbolizuje poziom optymalności
* – ilość krawędzi

Złożoność rzędu nawet w przypadku dużych złożoności nie przekroczy . Najpopularniejsze heurystyki k-optymalne to algorytm 2-optymalny oraz 3-optymalny.

### Algorytm 2-optymalny

(Sidford, 2020) Algorytm 2-optymaly jest jedną z popularnych heurystyk, które zostały zastosowane w problemie komiwojażera. Heurystyka ta polega na iteracyjnej poprawie danego rozwiązania przez lokalne zmiany. W algorytmie tym cykl obejmujący każdy wierzchołek grafu, podlega stopniowej optymalizacji za pomocą zmiany dwóch nieprzylegających krawędzi, powodując powstanie nowego cyklu. Proces powtarzany jest do momentu, w którym nie ma możliwości uzyskania krótszej ścieżki poprzez dokonywania kolejnych zmian.  
(Sysło, Deo i Kowalik, 1995) Na początku należy wybrać losowy bądź heurystycznie ustalony początek cyklu oraz dokonać usunięcia dwóch łuków. Kolejnym krokiem jest utworzeniu cyklu zastępując usunięte krawędzie innymi. W przypadku, gdy koszt cyklu będzie mniejszy od cyklu , należy zastąpić cykl pierwotny zastąpić cyklem oraz kontynuować zamianę łuków. Jednak w przypadku, gdy koszt cyklu H jest mniejszy, należy dokonać w nim zamiany innych krawędzi. Proces ten powtarzany jest do momentu uzyskania najlepszej trasy. W trakcie każdorazowego przebiegu algorytmu przeprowadza się badanie par łuków, natomiast złożoność czasowa pojedynczej iteracji wynosi . Proces wymiany dwu-optymalnej przedstawiony został na rysunku X.



Rysunek 21. Wymiana 2-optymalna

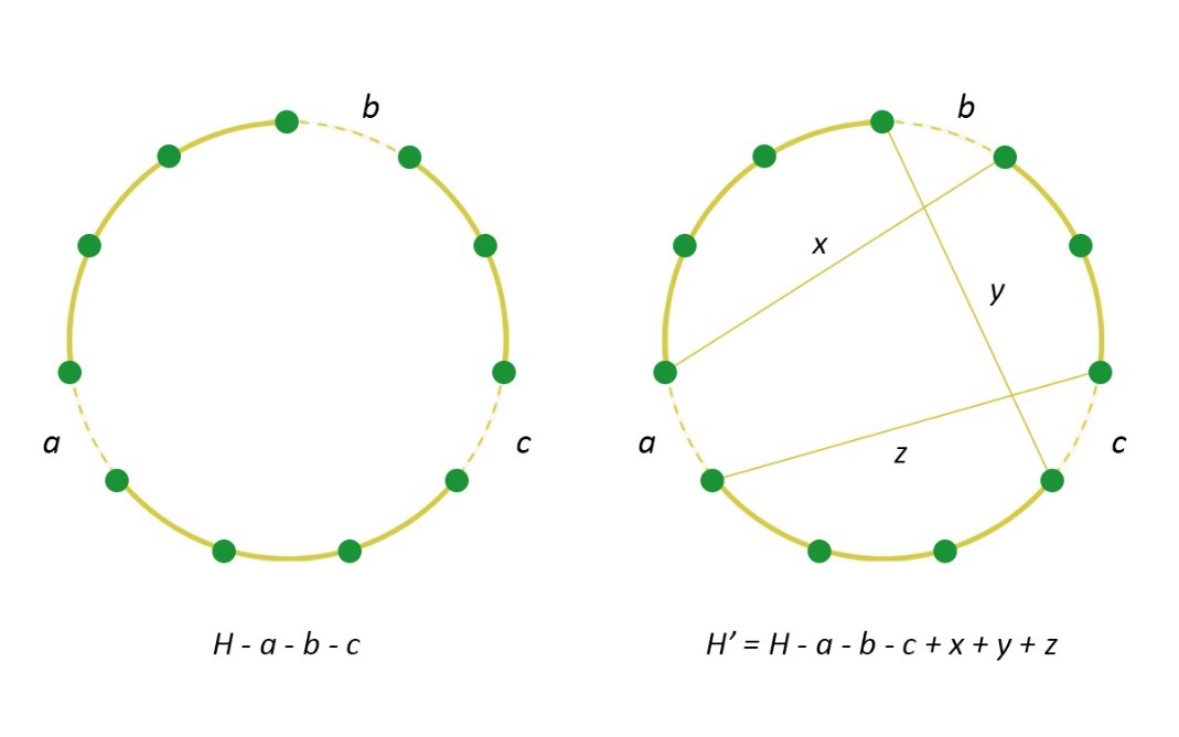
[*źródło:* opracowanie własne na podstawie (Sysło, Deo i Kowalik, 1995)]

Matematyczne uzasadnienie wymiany 2-optymalnej opiera się na porównaniu wag tras przed i po zaproponowanej zmianie. W przypadku, gdy różnica w wagach, która określana jest jako daje wynik dodatni, wtedy nowo utworzona trasa H’ określana jest jako krótsza od H. Przypadek ten jest istotny, aby kryterium zamiany było większe od zera, co sugeruje możliwość skrócenia trasy. Kryterium zamiany opisane jest wzorem:

Kryterium zamiany przedstawione w równaniu X zawiera wagi krawędzi przed oraz po dokonaniu zamiany. Pierwsza część równania przedstawia wagi oryginalnych krawędzi w trasie, które rozważane są do zmiany. Oznaczają one bezpośrednie połączenie pomiędzy wierzchołkami, które będą zamieniane. Droga część równania stanowi wagi krawędzi otrzymane po zamianie. oraz reprezentują nowe połączenia powstałe po zamianie krawędzi oraz .

### Algorytm 3-optymalny

(Sysło, Deo i Kowalik, 1995) Algorytm trój-optymalny to heurystyka wykorzystywana do rozwiazywania problemu komiwojażera, polegająca na iteracyjnej poprawie istniejącej trasy poprzez zamianę trzech krawędzi. Operacja ma na celu skrócenie całkowitej długości trasy, jednocześnie sprawdzając różne opcje ponownego połączenia fragmentów trasy po usunięciu wybranych krawędzi. Na początku każdego kroku algorytm wybiera trzy krawędzie, które zostaną usunięte, a następnie ponownie łączy powstałe trzy segmenty w innej kolejności, analizując w ten sposób czy istnieje możliwość osiągnięcia krótszej trasy, tworząc tym samym nowe cykle. Każda z tras, które zostały utworzone jest oceniana pod kątem jej długości. W przypadku, gdy nowy cykl jest krótszy od poprzedniego, wtedy zmiana jest akceptowana. Operacja powtarzana jest do momentu, kiedy nie ma możliwości kontunuowania popraw bądź osiągnięta zostanie określona liczba iteracji. Algorytm 3-optymalny ma potencjał do odnajdywania zdecydowanie krótszych cykli, niżeli algorytm 2-optymalny. Dodatkowym atutem tej metody jest większa złożoność, przez co może lepiej eksplorować przestrzeń rozwiązań. Mimo, iż algorytm ten jest bardziej skuteczny w wyszukiwaniu krótszych cykli, nie zapewnia on odnalezienia zawsze najkrótszego możliwego cyklu. Dodatkowym ograniczeniem jest jego czasochłonność, ze względu na znacznie większą ilość kombinacji do sprawdzenia przy każdej z iteracji. Rysunek X przedstawia przykładową wymianę trój-optymalną.



Rysunek 22. Wymiana trój-optymalna

[*źródło:* opracowanie własne na podstawie (Sysło, Deo i Kowalik, 1995)]

## Algorytm zachłanny

(CodingDrills, 2024) Algorytm zachłanny to jedna z metod optymalizacji, której celem jest podejmowanie lokalnych optymalnych decyzji, w każdym kroku działania algorytmu, dążąc do globalnie optymalnego rozwiązania. W kontekście problemu komiwojażera, celem algorytmu zachłannego jest wyszukanie najkrótszej możliwej drogi, za pomocą której będzie możliwość odwiedzenia wszystkich możliwych miast wyłącznie jeden raz, powracając na samym końcu do miejsca rozpoczęcia drogi. Na początku procesu należy dokonać wyboru miasta startowego , gdzie określa zbiór wszystkich miast. Następnie należy dodać miasto C do zbioru odwiedzonych miast . Tak długo jak wszystkie miasta nie zostaną odwiedzone   
, należy podjąć próbę odnalezienia najbliższego nieodwiedzonego miasta, oznaczonego symbolem , minimalizując jednocześnie odległość , gdzie symbolizuje funkcję odległości oraz wyrażamy wzorem:

(X)

Gdzie:

* arg min – symbolizuje operator matematyczny, zwracający wartość zmiennej, dla której dana funkcja osiąga minimum.
* – oznacza, że przeszukiwany jest zbiór miast nie odwiedzonych
* - zbiór wszystkich miast
* – zbiór miast odwiedzonych
* – wyraża różnicę zbiorów oraz , wszystkie miasta z , których nie ma   
  w zbiorze
* – wyraża funkcję odległości między miastem bieżącym , a miastem .

Kolejny krok to dodanie miasta j do zbioru odwiedzonych miast , oraz ustawienia miasta , jako miasto . Na zakończenie należy dodać krawędź powrotną do miasta, w którym droga została rozpoczęta, w celu zamknięcia cyklu i zakończenia trasy. Wyrażamy to wzorem . Przedstawione matematyczne sformułowania najprościej przedstawić na prostym przykładzie, w którym rozważona została macierz odległości dla czterech miast A, B, C oraz D. Rysunek X przedstawia macierz odległości między miastami przykładowego zadania.

Obraz zawierający tekst, Czcionka, biały, typografia

Opis wygenerowany automatycznie

Na początku zgodnie z zasadą algorytm zachłanny rozpocznie od miasta A, wybierając najbliższe nieodwiedzone miasto, aż do momentu odwiedzenia każdego z miast. Należy ustawić zbiór miast odwiedzonych S = {A} oraz wskazać bieżące miasto C = A. W kolejnym kroku należy odnaleźć najbliższe nieodwiedzone miasto , minimalizując odległość . Rozważane zostają odległości pomiędzy miastami , oraz względem miasta początkowego A. Według przeprowadzonej analizy wybrane ze względu na najmniejsza odległość do miasta A zostaje miasto B. Następnie należy dokonać aktualizacji oraz dodanie miasta do zbioru odwiedzonych miast  
. Ponownie następuje wyszukanie najbliższego nieodwiedzonego miasta względem miasta, tym razem względem punktu B. Pod uwagę brane są miasta oraz . Wybór ze względu na mniejsza odległość pada na miasto , dokonywana jest aktualizacja oraz dodanie miasta do zbioru miast odwiedzonych . Operacja wyszukania najbliższego nieodwiedzonego miasta jest ponawiana, jednak ze względu, że pozostało tylko miasto C, dlatego względem niego odliczana jest odległość . Dokonywana jest aktualizacja obecnego miasta oraz następuje dodanie ostatniego miasta do zbioru miast odwiedzonych . Następnie dodawana jest krawędź powrotna do miasta startowego . Ostateczna trasa prezentuje się w następujący sposób: , a jej długość całkowita wynosi .

Algorytmy zachłanne nie zawsze gwarantują uzyskanie optymalnego rozwiązania, mimo to dla niektórych są one wystarczające. Do algorytmów dokładnych zaliczamy m.in. Algorytm Prima, Algorytm Kruskala, natomiast do niedokładnych algorytm najmniejszej krawędzi oraz najbliższego sąsiada.



### Algorytm Kruskala

(Cormen, Leiserson, Rivest i Stein, Wprowadzenie do algorytmów, 2007) Algorytm Kruskala reprezentuje algorytmy służące do znajdowania minimalnego drzewa rozpinającego MST[[7]](#footnote-7) w grafie ważonym. (Wojciechowski i Pieńkocz, 2013) Został opracowany i zaprezentowany przez Josepha Bernarda Kruskala w 1956 roku. Jego działanie opiera się na wyborze krawędzi o najmniejszej wadze, które nie tworzą cyklu z wybranymi wcześniej krawędziami, natomiast suma wag krawędzi jest minimalna. (Sysło, Deo i Kowalik, 1995) Jeżeli wystąpi sytuacja, w której poddana badaniu krawędź osiągnie cykl z wybranymi wcześniej krawędziami, wtedy krawędź ta jest pomijana. Proces trwa, dopóki nie zostanie wybrane n-1 krawędzi, bądź wszystkie krawędzie w grafie nie zostaną rozpatrzone. Formalnie dla grafu G = (V, E), gdzie V to zbiór wierzchołków, natomiast E to zbiór krawędzi proces ten można opisać w następujący sposób:

* Inicjalizacja – Należy utworzyć zbiór A, który początkowo będzie zbiorem pustym . Następnie trzeba utworzyć zbiór, w którym każdy wierzchołek jest reprezentowany przez osobne drzewo
* Sortowanie krawędzi – Należy posortować krawędzie w zbiorze E według wag
* Dodawanie krawędzi – W tym kroku należy przejść przez posortowane krawędzie dodając każdą z nich do zbioru A w przypadku, gdy nie tworzą one cyklu z innymi wybranymi krawędziami. W przypadku, gdy wierzchołki oraz ze zbioru  
   w zbiorze krawędzi E, należą do różnych drzew , należy dodać do zbioru oraz połączyć drzewa zawierające wierzchołki oraz za pomocą funkcji do zarządzania drzewami
* Zakończenie – W przypadku osiągnięcia w zbiorze A krawędzi, należy zwrócić zbiór A jako minimalne drzewo rozpinające

Czas działania algorytmu Kruskala w grafie G = (V, E) zależny jest od operacji sortowania, jak i operacji na zbiorach rozłącznych. Zakładając, że zastosowane zostały zastosowane struktury danych z heurystyką łączenia zarówno według rangi, jak i kompresji ścieżek, czas działania algorytmu wynosi dla sortowania krawędzi oraz dla operacji FIND – SET[[8]](#footnote-8) oraz UNION[[9]](#footnote-9) wykonywanych w pętli, gdzie stanowi funkcję odwrotną do funkcji Ackermanna, która charakteryzuje się bardzo powolnym wzrostem oraz możliwością traktowania jej jako stałej w praktycznych zastosowaniach. Złożoność czasowa wynosi , bądź jest równoważne , w przypadku, gdy E jest maksymalnie równe .

Obraz zawierający diagram, linia, origami

Opis wygenerowany automatycznie

Proces działania algorytmu Kruskala warto przedstawić na prostym przykładzie. (Yadav, 2022) W przedstawionym powyżej przykładzie zawartym na rysunku X na początku wybrana została krawędź o najmniejszej wadze, wynoszącej 1. Krok ten jest powtarzany dla kolejnej krawędzi posiadającej najmniejszą wagę, w tym przypadku jest to krawędź numer 2. Można zaobserwować, że wybrane krawędzie nie są ze sobą w żaden sposób połączone, więc w kolejnym etapie wybrana została krawędź o najmniejszym koszcie wynoszącym 3, która dodatkowo połączy poprzednio wybrane krawędzie ze sobą. Ze względu na fakt, że połączenie wybranych dotąd krawędzi z krawędziami o kosztach 4,6 oraz 8 spowodowałoby utworzenie cyklu, odcinki te zostają pominięte, a do zbioru krawędzi dobieramy krawędź o wadze równej 5. Po zsumowaniu wszystkich wag wybranych krawędzi otrzymany zostaje koszt drzewa rozpinającego, który wynosi .

### Algorytm Prima

(Cormen, Leiserson, Rivest i Stein, Wprowadzenie do algorytmów, 2018) Algorytm Prima to przedstawiciel algorytmów, których celem jest odnalezienie minimalnego drzewa rozpinającego w grafie ważonym. Po raz pierwszy algorytm ten został opublikowany w 1930 roku przez czeskiego matematyka Vojtěcha Jarníka, natomiast niezależnie odkryty został w 1957 roku przez amerykańskiego matematyka Roberta C. Prima III. (Wojciechowski i Pieńkocz, 2013) Działanie algorytmu opiera się na rozbudowaniu drzew, za pomocą dodania kolejnych krawędzi posiadających najmniejszy koszt, które łączą wierzchołki będące już w drzewie z wierzchołkami, które nie są obecnie dostępne w drzewie. Sam proces trwa do momentu włączania wszystkich wierzchołków do drzewa. Kroki algorytmu Prima można przedstawić w kilku prostych krokach. Na samym początku wskazujemy dowolny wierzchołek z grafu , który będzie pełnił funkcję wierzchołka startowego. Kolejny krok to stworzenie dwóch zbiorów: V1, który będzie zawierał wierzchołki drzewa, a także E1, pełniący funkcję zbioru zawierającego krawędzie drzewa. W tym momencie dokonywana zostaje inicjalizacja zbiorów jako oraz , a następnie należy dokonać ustawienia początkowego kosztu drzewa jako. Kolejny etap to iteracja, w której należy wybrać krawędź posiadającą minimalny koszt względem innych krawędzi, gdzie oraz . Następnie należy dodać wierzchołek do zbioru , a także do zbioru oraz dokonać aktualizacji kosztu drzewa , gdzie stanowi wagę krawędzi e. Podsumowując iterację T = (V, E1) określane jest jako drzewo minimalnie rozpinające, a jego koszt wynosi wt. Złożoność czasowa algorytmy zależna jest od użytej struktury danych. W przypadku zastosowania macierzy sąsiedztwa wynosi ona . Z kolei w wypadku zastosowania kopca binarnego złożoność czasowa wynosi . Istnieje również możliwość zastosowania kopca Fibonacciego, w którym złożoność wynosi   
.

*Obraz zawierający diagram, linia

Opis wygenerowany automatycznie*

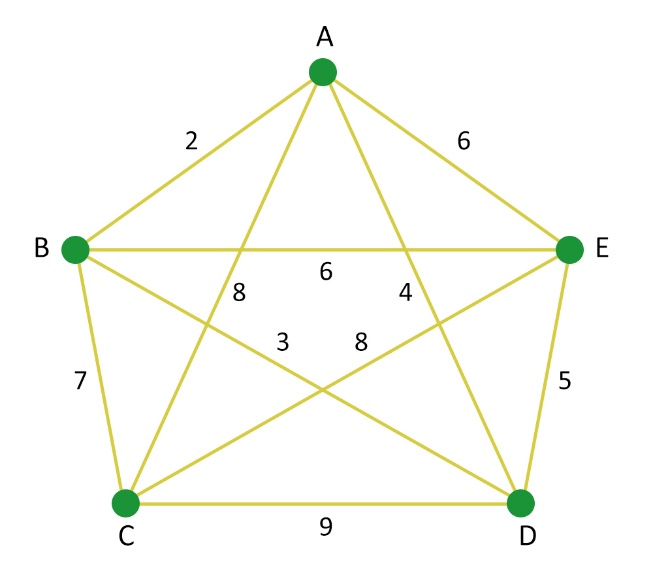
W przypadku algorytmu Prima analizę należy rozpocząć od dowolnie wybranego wierzchołka. Kolejny krok to wskazanie nowego wierzchołka, który sąsiaduje z wcześniej wybranym wierzchołkiem. (Vasava, 2019) W przykładzie przedstawionym powyżej na rysunku X na początku wybrana została krawędź, której waga wynosiła 1. Kolejny krok to wybór odpowiedniej krawędzi z pozostałych o wagach wynoszących 2, 3 oraz 4. Dokonano wyboru odcinka, którego koszt wynosi 2. Kontynuując iteracje do wyboru pozostają krawędzie 3,4 oraz 5. Z uwagi, że wybór krawędzi 3 spowodowałby utworzenie cyklu, wybrana zostaje krawędź z wagą 4. Reasumując koszt powstałego MST wynosi 1+2+4 = 7.

## Algorytm selekcji krawędzi

(Zieliński, 2024) Algorytm selekcji krawędzi w odróżnieniu od innych znanych algorytmów nie zajmuje się budową drzew oraz nie tworzy żadnych cykli pomocniczych. Utworzenie tego algorytmu zależne jest od faktu, czy spełnione zostaną następujące postulaty:

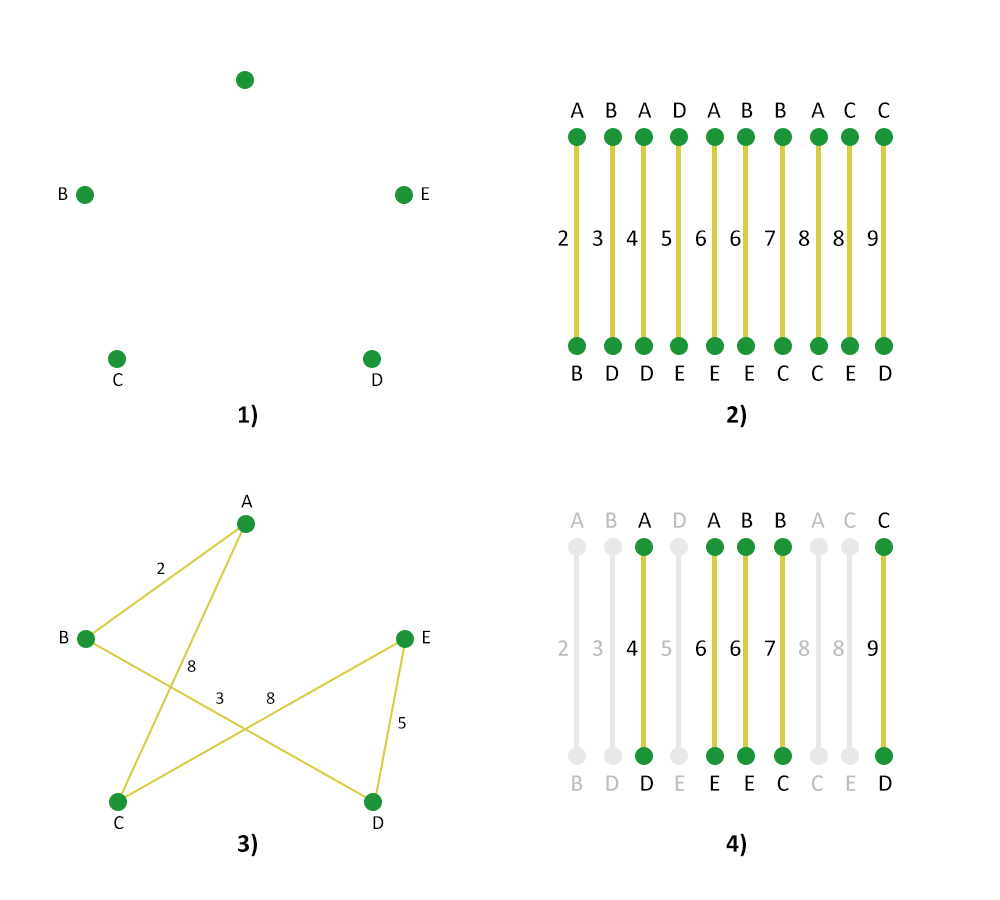
* + 1. W grafie posiadającym cykl Hamiltona, stopnie wszystkich wierzchołków wynoszą 2
    2. W grafie musi istnieć przynajmniej jeden cykl Hamiltona
    3. Krawędzie cyklu Hamiltona posiadają najmniejsze możliwie wagi

Algorytm w nieuporządkowany sposób wybiera krawędzie, które spełniają powyższej wymienione wymagania. Rozpoczyna pracę od stworzenia listy m krawędzi, następnie sortując je według wag. Następnie graf pusty o n wierzchołkach przekształcany jest w graf regularny[[10]](#footnote-10) stopnia drugiego tj. krawędzie ze zbioru m zostają dołączone n krawędzie z zastosowaniem sposobu rosnących wag. Do dołączenia wybierane są wyłącznie krawędzie nie doprowadzające do zwiększenia jakiegokolwiek wierzchołka do większej wartości niż 2. Dodatkowo cykl nie powinien zostać przedwcześnie zamknięty z powodu dołączenia krawędzi, z wyjątkiem krawędzi ostatniej, której zadaniem jest zamknięcie cyklu Hamiltona. Na rysunku X przedstawiono przykładowy graf regularny.



Strukturę danych stanowi zbiór krawędzi, oznaczony symbolem E, a także zbiór wierzchołków oznaczony jako V. Krawędzie e muszą spełniać następujący warunek: pierwszy incydenty wierzchołek v1, drugi incydenty wierzchołek v2 oraz posiadać wagę w. Z kolei wszystkie wierzchołki v spełniają warunek: sąsiedni wierzchołek to v1, sąsiedni wierzchołek to v2 oraz stopień wierzchołka r. Ponadto istnieje licznik krawędzi, którego zadaniem jest zliczenie krawędzi uwzględnionych dotychczas w trasie komiwojażera.

Należy rozpocząć od wprowadzenia do zbioru E danych grafu oraz zresetowania licznika krawędzi oraz dodatkowo dokonania resetu w zbiorze wierzchołków V. Algorytm sortuje od najmniejszego elementy zbioru E oraz w odpowiedni sposób przenosi dane zgromadzone w zbiorze E, do wyzerowanego zbioru V. Jako pierwsza wprowadzana jest krawędź posiadająca najmniejszą wagę, w sposób, w którym w wierzchołkach v1 oraz v2, które są połączone krawędzią ze zbioru V dodaje się stopień, a także numer sąsiadującego wierzchołka. W ten sam sposób wprowadzana jest również druga krawędź. W kolejnych przypadkach można wprowadzić krawędzie wyłącznie, gdy ich incydentalne wierzchołki posiadają stopień 0 bądź 1. W przypadku, gdy oba wierzchołki posiadają stopień 1, należy sprawdzić czy cykl nie powstanie za wcześnie. To czy cykl istnieje sprawdzane jest poprzez przechodzenie od wierzchołka v1 do sąsiednich wierzchołków o stopniu równym 2 do samego końca, czyli wierzchołka, którego stopień jest równy 1. W przypadku, gdy skrajny wierzchołek badanej krawędzi to drugi wierzchołek incydentalny v2, krawędź badana nie może zostać zaliczona do rozwiązania, ze względu na wystąpienie cyklu. Zbiór rozwiązania poszerzany jest o krawędź, której wierzchołek różni się od v2. Algorytm kończy działanie, gdy licznik dołączonych krawędzi będzie taki sam, jak liczba wierzchołków w grafie.



Na rysunku X przedstawiona została zasada działania algorytmu selekcji krawędzi podzieloną na cztery etapy. W etapie pierwszym przedstawiony został graf w formie wierzchołków rozmieszczonych w losowy sposób w przestrzeni bez połączeń. Następnie w etapie drugim przedstawiona została lista wszystkich możliwości połączeń pomiędzy wierzchołkami wraz z podanymi wartościami wagowymi. Kolumny reprezentują wierzchołki, a wiersze krawędzie. Etap trzeci przedstawia graf z połączonymi między sobą wierzchołkami. Krawędzie oznaczone zostały liczbami, które odpowiadają ich wagom. Etap ostatni przedstawia wyniki działania algorytmu. Zaznaczone są wybrane krawędzie, które tworzą optymalny zbiór połączeń zgodnie z kryterium algorytmu. Algorytm dzięki swojej prostocie jest bardzo wydajny i daje bardzo dobre wyniki. Dla przykładu przedstawionego na rysunku X optymalne rozwiązanie wynosi 26.

## Algorytm genetyczny

(Debudaj-Grabysz, Deorowicz i Widuch, 2012) Algorytmy genetyczne są to metaheurystyczne techniki optymalizacyjne, za pomocą których istnieje możliwość rozwiązywania złożonych problemów optymalizacyjnych, a także poszukiwanie globalnych ekstermów funkcji. Za ich pomocą istnieje możliwość symulacji procesów biologicznej ewolucji m.in. selekcji naturalnej, krzyżowania, czy też mutacji w celu przekształcania populacji ewentualnych rozwiązań ku lepszemu przystosowaniu do zadanej funkcji celu. Algorytmy genetyczne zostały przedstawione w 1975 roku przez amerykańskiego naukowca Johna Hollanda z Uniwerstytetu Michigan. Do podstawowych elementów algorytmów genetycznych zalicza się:

* Reprezentacje rozwiązań;
* Generowanie populacji początkowej;
* Ocena jakości chromosomów;
* Selekcja;
  + Selekcja proporcjonalna;
  + Selekcja rankingowa;
  + Selekcja turniejowa;
* Krzyżowanie;
* Mutacja;
* Warunek zatrzymania,

W etapie pierwszym chromosomy pełną funkcję przedstawicieli możliwych rozwiązań problemu. Każdy z nich składa się z genów, które mają możliwość przyjęcia wszelakich wartości tzw. alleli. Najczęściej chromosomy w algorytmach genetycznych kodowane są binarnie. Początkowo dochodzi do wygenerowania populacji w losowy sposób, przez co każdy chromosom xi stanowi ciąg bajtów o długości l. Można to wyrazić za pomocą wzoru: . Następnie każdy z chromosomów oceniany jest za pomocą funkcji przystosowania, której zadaniem ustalenie na ile prawidłowo dany każdy z chromosom rozwiązuje problem. Funkcje przystosowania można zapisać za pomocą wzoru:

Gdzie:

* – całkowita korzyść qi, związana z xi
* – zmienna binarna wskazująca czy element został wybrany (1), czy nie (0)
* – wartość związana z i
* – kara za przekroczenie ograniczeń wagowych
* – współczynnik wagowy
* – całkowita waga przekraczająca dopuszczalny limit W. Jeżeli waga całkowita jest mniejsza lub równa W, wartość wyrażenia jest równa 0.

Współczynnik wagowy wyznaczany jest za pomocą wzoru:

Gdzie:

* – symbolizuje maksymalną wartość korzyści spośród elementów
* – oznacza minimalną wagę spośród wszystkich elementów

W kolejnym etapie dokonuje się selekcji chromosomów do reprodukcji oraz sukcesji. Najczęściej stosuje się selekcje proporcjonalną opartą na metodzie ruletki, w której każdy z osobników x obecnej populacji przyporządkowuje się prawdopodobieństwo reprodukcji px, którą można opisać wzorem:

We wzorze X fx symbolizuje funkcję oceny chromosomu, a zatem szansa na wylosowanie chromosomu jest wprost proporcjonalna do wartości funkcji przystosowania. Aby wskazać pulę rodzicielką chromosomów, koło ruletki, której obwód wynosi 1 należy podzielić je na wycinki o długości łuku px. Kolejny krok to wylosowanie liczby z przedziału [0,1) wskazującą punkt na obwodzie określonego wycinka koła ruletki. W przypadku funkcji przystosowania wyrażonej wzorem X istnieje prawdopodobieństwo wystąpienia ujemnych wartości. Aby tego uniknąć należy zastosować skalowanie przystosowania, które oblicza się według wzoru:

We wzorze X symbolizuję funkcję przystosowania najgorszego osobnika.

Istnieje również drugi wariant dokonania selekcji – selekcja rankingowa, w której chromosomy posortowane są wedle nierosnących wartości funkcji oraz nadaje im się numery rozpoczynające się od 0. Najczęściej stosowana jest funkcja liniowa określona wzorem:

We wzorze X symbole a oraz k są parametrami, przy których wyborze należy uwzględnić następujące warunki:

Warto również określić, że jeżeli, to . Metoda selekcji rankingowa jest odpowiednia w przypadku zarówno problemów minimalizacji, jak i maksymalizacji.

Selekcja turniejowa polega na losowym wyborze kilku osobników z populacji, którzy rywalizują ze sobą w "turnieju". Zwycięzca turnieju, czyli osobnik o najwyższej wartości funkcji przystosowania, jest wybierany do dalszego przetwarzania, na przykład do krzyżowania. Proces selekcji turniejowej rozpoczyna się od losowego wyboru k osobników z bieżącej populacji. Losowanie może zostać przeprowadzone zarówno z powtórzeniami, jak i bez. Wybrani osobnicy tworzą grupę turniejową, z której zostanie wyłoniony zwycięzca. Każdy z chromosomów oceniany jest za pomocą funkcji przystosowania f(x), której zadaniem jest zmierzenie jakości rozwiązania jakie dany osobnik reprezentuje. Zwycięzcą jest osobnik xbest, który ma najwyższą wartość funkcji przystosowania spośród wszystkich uczestników turnieju. Sam wybór zwycięzcy można opisać wzorem:

Następnie rozważana jest opcja przeprowadzenia wymiany pokoleń. Najpopularniejszą metodą sukcesji jest całkowite zastępowanie, w której populacja potomna staje się bieżącą. Innym sposobem jest sukcesja z częściowym zastępowaniem, w której część najlepszych chromosomów rodzicielskich przekazywana jest do nowego pokolenia bez jakichkolwiek zmian, a tym samym nie podlegają operacji krzyżowania oraz mutacji. Na początku dokonuje się usunięcia osobników populacji bazowej, a następnie w ich miejsce wprowadza się osobniki potomne. symbolizuje współczynnik wymiany. Dokonuje się usunięcia chromosomów podobnych do potomnych bądź najgorzej przystosowanych. W przypadku sukcesji elitarnej dokonuje się skopiowania przynajmniej najlepszego osobnika z poprzedniego pokolenia na samym początku nowej generacji.

Kolejnym etapem jest krzyżowanie, które poleca na wymianie fragmentów chromosomów pomiędzy dwoma rodzicami w celu stworzenia potomstwa. Istnieją różne metody krzyżowania, takie jak krzyżowanie jednopunktowe, dwupunktowe i krzyżowanie z częściowym odwzorowaniem (PMX). Krzyżowanie jednopunktowe poleca na przecięciu chromosomów rodzicielskich w wylosowanym punkcie k, w taki sposób, że pierwszy potomek otrzymuje k genów od pierwszego rodzica, natomiast resztę od drugiego, a w przypadku drugiego potomka odwrotnie. Przykład krzyżowania jednopunktowego przedstawiony został w tabeli X

|  |  |  |  |
| --- | --- | --- | --- |
| Rodzic 1 | | Rodzic 2 | |
| X1 | X2 | Y1 | Y2 |
| X3 | X4 | Y3 | Y4 |
| Potomek 1 | | Potomek 2 | |
| X1 | X2 | Y1 | Y2 |
| Y3 | Y4 | X3 | X4 |

W przypadku krzyżowania dwupunktowego, jeden z potomków otrzymuje geny do pierwszego punktu cięcia oraz po drugim punkcie cięcia od pierwszego rodzica, natomiast geny pomiędzy punktami cięcia od drugiego z rodziców. Analogicznie tworzony jest kolejny potomek.

Z kolei w przypadku krzyżowania z częściowym odwzorowaniem Patrially Mapped Crossover należy na początku wybrać dwa punkty cięcią np. 123|4567|89 oraz 862|1753|94. Następnie należy dokonać wymiany odcinkami pomiędzy punktami cięcia, w przypadku przytoczonego powyżej przykładu cyferki 4567 oraz 1753 zamieniają się stronami. Następnie należy utworzyć tabelę odwzorowań . Kolejny krok to ustawienie genów rodziców, które nie będą powodować konfliktów, w analizowanym przykładzie \*2\*|1753|89 oraz 8\*2|4567|9\*, a następnie w miejscach konfliktów ustawić geny zgodnie z tabelą odwzorowań, dzięki czemu uzyskany zostaje efekt 426|1753|89 oraz 832|4567|91.

Kolejny etap funkcjonowania algorytmów genetycznych to mutacja. Mutacja wprowadza losowe zmiany w chromosomach potomstwa, w celu utrzymania różnorodności genetycznej populacji. Najczęściej spotykaną metodą mutacji jest mutacja bitowa, w której każdy bit jest odwracany z prawdopodobieństwem pm, które zwykle jest bardzo małe.

Algorytm kończy działanie po spełnieniu określonego warunku, np. osiągnięciu maksymalnej liczby iteracji T bądź znalezieniu rozwiązania spełniającego kryteria przystosowania f(x\*).

(Anholcer, 2023) W odniesieniu do problemu komiwojażera, algorytm genetyczny ma obowiązek uwzględniać specyficzne wymagania z nim związane. W tym przypadku stosowane jest kodowanie kolejnościowe, w którym kod składa się z tylu znaków, ile miast występuje w zadaniu, a każdy ciąg kodowy jest przedstawicielem permutacji miasta, które należy odwiedzić. Przykładowo dane są cztery miasta, a rozwiązanie, w którym komiwojażer będzie miał możliwość odwiedzenia miast w kolejności 1,3,2,4 można zakodować jako (1,3,2,4), dzięki czemu każde miasto zostanie odwiedzone wyłącznie jeden raz. Problem pojawia się w momencie próby krzyżowania oraz mutacji. W przypadku krzyżowania jednopunktowego na dwóch chromosomach (1,3,2,4) oraz (2,4,1,3) istnieje możliwość otrzymania potomstwa nie będącego poprawną permutacją miast np. (1,3,1,3). Jednym z rozwiązań tego problemu jest krzyżowanie porządkowe OX, które polega na przepisywaniu w każdym z ciągów początkowych elementów, które umieszczone są w odpowiednich pozycjach w ciągu drugim. Przykładowo w przypadku rodziców (1,3,2,4) oraz (2,4,1,3), wybrano miejsce krzyżowania pomiędzy punktem 2 oraz 3. Potomstwo wyglądać będzie w sposób następujący:

* Potomek 1: (1,3,1,3)
* Potomek 2: (2,4,2,4)

Kolejnym krokiem będzie wstawienie w odpowiednie miejsca elementów drugiego rodzica, tak aby zachować permutację. Po zmianie potomstwo wygląda następująco:

* Potomek 1: (1,3,4,2)
* Potomek 2: (2,4,1,3)

W przypadku mutacji odnosząc się do problemu komiwojażera należy również zastosować specjalistyczne podejście. Wyklucza się mutacje jednopunktową, stosując mutację polegającą na zamianie miejscami dwóch losowo wybranych elementów ciągu. Przykładowo dla ciągu (1,3,4,2) po zastosowaniu zamiany miejscami na pozycjach 2 oraz 4 otrzymany zostaje nowy ciąg (1,2,4,3).

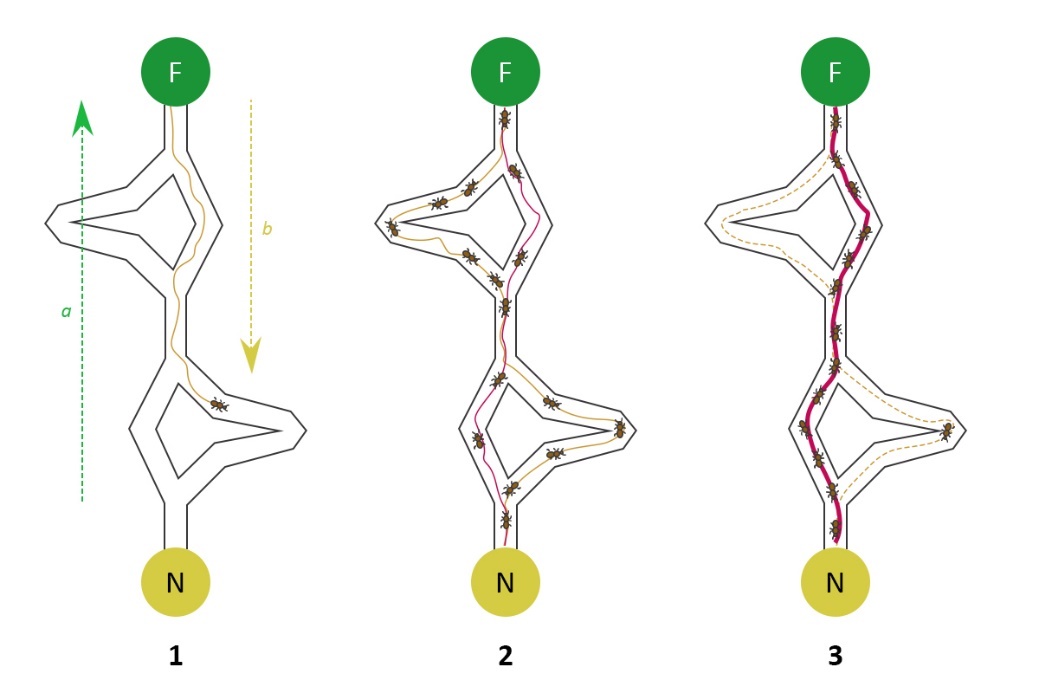
W kontekście problemu TSP funkcja przystosowania stanowi odwrotność funkcji celu, czyli zamiast minimalizować długość trasy, dokonuje się maksymalizacji przystosowania, które jest odwrotnością trasy, co można przedstawić wzorem:

## Algorytm mrówkowy

(Dorigo i Stützle, 2004) Algorytm mrówkowy (Ant Colony Optimization – ACO) to metaheurystyka, która nawiązuje do zachowania mrówek w kontekście poszukiwania najkrótszych ścieżek do źródła pokarmu za pomocą feromonów. (Błaszkiewicz, 2011) Algorytm zaproponowany został w 1991 roku w pracy doktorskiej Marco Dorigo, która była zainspirowana opublikowaną w 1989 roku pracą o zachowaniu mrówek argentyńskich. (Dorigo i Stützle, 2004) Metaheurystyka rozwijana przez Marco Dorigo oraz Thomasa Stützle znalazła zastosowanie do rozwiązywania trudnych problemów optymalizacyjnych. Dokonując analizy algorytmu należy poznać podstawowe pojęcia, z pomocą których prościej będzie zrozumieć ideę algorytmu. Należą do nich:

* **Feromony** – substancje chemiczne wydzielane przez organizm, wywołujące specyficzne reakcje u osobników tego samego ratunku. W przypadku mrówek wykorzystywane są do komunikacji np. do oznaczenia ścieżki do źródła pokarmu. Pozostałe mrówki kierują się ścieżkami z wyczuwalnym silniejszym zapachem dzięki czemu wzmacniają go jeszcze bardziej. W kontekście algorytmu ACO dochodzi do komunikacji sztucznych mrówek, za pomocą sztucznego zapachu, symbolizującego numeryczną informację rozproszoną. Feromony wykorzystywane są do probabilistycznej konstrukcji rozwiązania problemu, natomiast ich wartości podlegają aktualizacji podczas działania ACO odzwierciedlając tym samym doświadczenia mrówek podczas poszukiwań.
* **Reguła prawdopodobieństwa** – mrówki dokonują wyboru drogi stosując prawdopodobieństwo zależne od ilości feromonów oraz innych heurystyk.
* **Aktualizacja feromonów** – feromony aktualizowane są na podstawie jakości odnalezionych rozwiązać, dzięki czemu zwiększana jest atrakcyjność dobrych ścieżek.

Algorytm swoją inspirację zaczerpnął z przeprowadzonego przez Deneubourga oraz jego współpracowników eksperymentu z podwójnym mostem, w którym to mrówki posiadały wybór dwóch ścieżek o różnych długościach, które prowadziły do źródła pokarmu. Na początku osobniki wybierały drogę losowo, jednak po pewnym czasie znaczna część mrówek zaczęła wybierać krótszą drogę. Krótsza ścieżka starała się znacząco oznaczona feromonami, co doprowadziło do zwiększenia jej popularności wyboru. Cały proces stanowi przykład autokatalizy, w której lokalne interakcje doprowadzają do globalnego wzorca. Doświadczenie w formie graficznej przedstawione zostało na rysunku X. Symbol F znajdujący się w zielonym kółku symbolizuje gniazdo mrówek, natomiast N znajdujące się w żółtym kółku symbolizuje źródło pożywienia.



Modele matematyczne algorytmu mrówkowego opisują w jaki sposób mrówki dokonują wyboru odpowiedniej ścieżki oraz w jaki sposób aktualizowane są poziomy feromonu. Pierwszy etap stanowi prawdopodobieństwo wyboru ścieżki, w którym to mrówki wybierają trasę za pomocą ilości feromonów na krawędziach oraz heurystyk danej krawędzi, zazwyczaj odwrotności długości tzn. im krótsza krawędź tym wyższa heurystyka. Prawdopodobieństwo pij(t) wyboru krawędzi oraz przez mrówkę w węźle w czasie wyraża się za pomocą wzoru:

Gdzie:

* – symbolizuje ilość feromonów na krawędzi w czasie ,
* – heurystyka dla krawędzi , najczęściej przyjmowana jako odwrotność procesu długości tej krawędzi: ,
* α – współczynnik wpływu feromonu na prawdopodobieństwo wyboru
* β – współczynnik wpływu heurystyki na prawdopodobieństwo wyboru
* „allowed” – zbiór krawędzi możliwych do wybrania przez mrówkę

W kolejnym etapie następuje aktualizacja feromonów znajdujących się na krawędziach rozpoczynając od zmniejszenia ich poziomu z powodu parowania oraz dodania ich nowej ilości na podstawie tras przebytych przez mrówki. Proces parowania wyrażany jest wzorem:

Gdzie:

* – symbolizuje ilość feromonów na krawędzi po aktualizacji w czasie ,
* – symbolizuje ilość feromonów na krawędzi w czasie równym
* – współczynnik parowania feromonów, który zazwyczaj wynosi

Następnie po procesie parowania dochodzi do aktualizacji feromonów poprzez dodanie nowej ich ilości pozostawionych przez przechodzące daną krawędzią mrówki, co opisywane jest wzorem:

W równaniu X symbolizuje liczbę mrówek, natomiast wyrażenie przedstawia ilość feromonów pozostawionych przez k-tą mrówkę na krawędzi . Ilość feromonów, jaką każda z mrówek pozostawiła na krawędzi, zależy od długości trasy, którą przebyła.

Algorytm mrówkowy został z powodzeniem zastosowany w kontekście rozwiązania problemu komiwojażera. ACO dla TSP wykorzystuje graf, w którym węzły reprezentują miasta, natomiast krawędzie symbolizują połączenia pomiędzy nimi. Każda krawędź posiada przypisaną wagę, która odpowiada odległości między miastami. Istnieje kilka głównych algorytmów mrówkowych dla problemu komiwojażera, do których zalicza się:

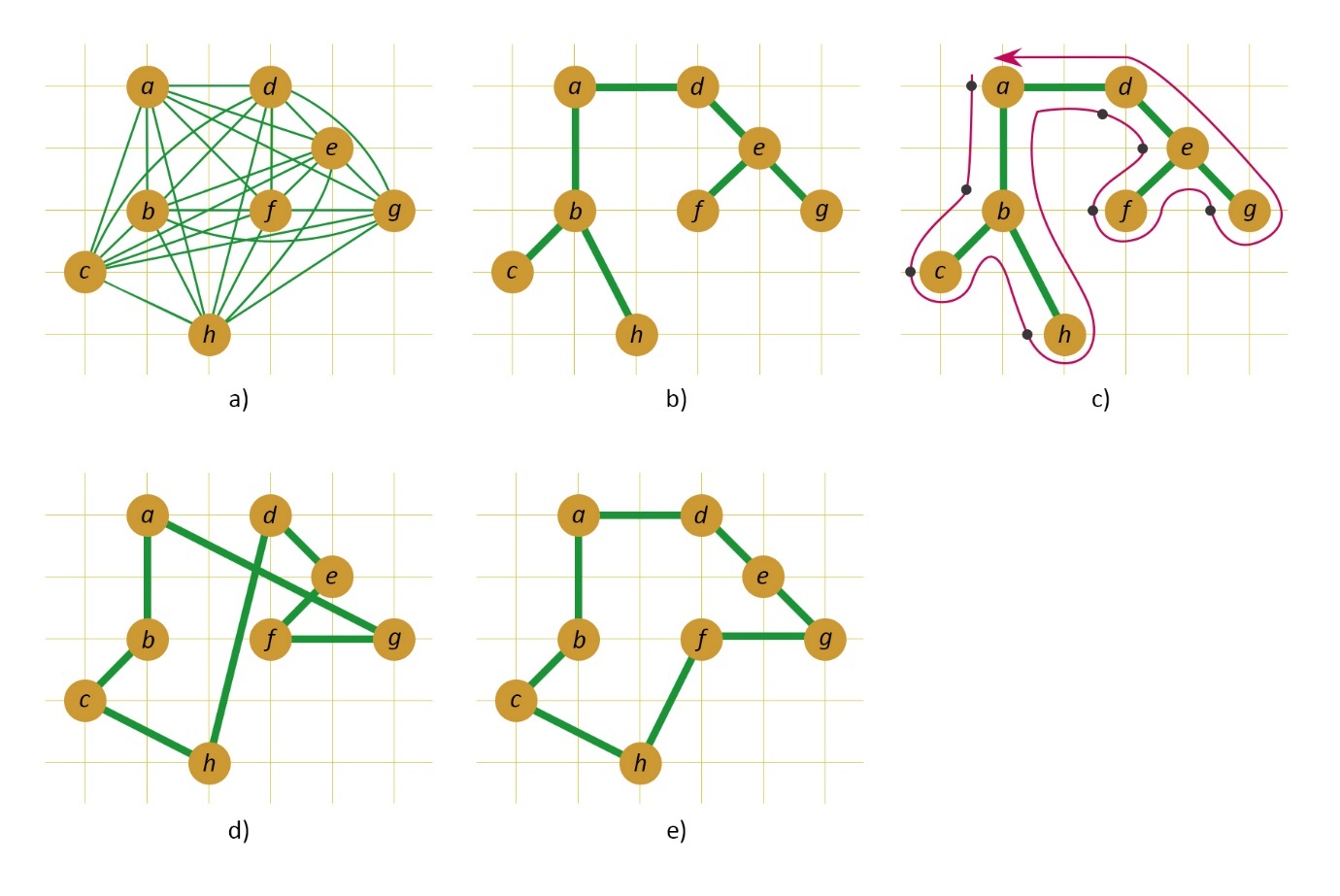
* **Ant System (AS)** – Jest to pierwszy algorytm mrówkowy, którego zadaniem jest inicjalizacja badań nad optymalizacją z pomocą kolonii mrówek. Stosuje równomierne rozmieszczenie feromonów na trasach, które wybrane zostały przez wszystkie mrówki, co z kolei doprowadziło do dobrej wydajności małych inicjalizacji problemu komiwojażera, jednak gorzej skalowało się do większych problemów.
* **Elitist Ant System (EAS)** – Wersja AS, której zadaniem jest wzmocnienie śladu feromonu na najlepszej odnalezionej trasie, co powodowało przyśpieszenie konwergencji algorytmu.
* **Rank-based Ant System (ASrank)** – jego zadaniem jest różnicowanie ilości feromonów na podstawie ich rankingu w bieżącej iteracji.
* Max-Min Ant System (MMAS) – wprowadza ograniczenia na minimalne oraz maksymalne wartości feromonów, dzięki czemu unika się przedwczesnej konwergencji i stagnacji.
* Ant Colony System (ACS) – algorytm wprowadza lokalne aktualizacje feromonów podczas opracowywania trasy oraz globalne aktualizacje dokonywane na najlepszej trasie, co poprawia wydajność na różnych instancjach problemu komiwojażera.

Algorytmy mrówkowe charakteryzują się ogromną skutecznością w rozwiązywaniu problemu komiwojażera, szczególnie w przypadku, gdy są połączone z technikami optymalizacji lokalnej.

## Algorytm Christofidesa

(Cormen, Leiserson, Rivest i Stein, Introduction to Algorithms, 2009) Algorytm Christofidesa jest przedstawicielem algorytmów aproksymacyjnych dla problemu komiwojażera, gdzie odległości spełniają nierówności trójkąta. Celem algorytmu jest odnalezienie cyklu Hamiltona, którego koszt nie będzie gorszy niż 1,5 razy koszt optymalnego cyklu oraz musi spełniać warunek nierówności trójkąta, a także wagi krawędzi nie mogą być ujemne. W odróżnieniu od heurystyk czy algorytmów genetycznych od początku gwarantuje on jakość rozwiązania, która bywa wystarczająca. Proces działania algorytmu składa się z kilku kroków. Na początku należy odnaleźć minimalne drzewo rozpinające dla grafu pełnego   
, który zawiera funkcję kosztu , a samo drzewo można odnaleźć z pomocą algorytmu Prima, bądź Kruskala. Samo drzewo jest podgrafem zawierającym wszystkie wierzchołki V oraz minimalną sumę kosztów krawędzi. W kolejnym kroku następuje identyfikacja wierzchołków o nieparzystym stopniu w zbiorze w zbiorze T, jednak z uwagi na właściwość drzew liczba wierzchołków jest parzysta. Po dokonaniu identyfikacji nieparzystych wierzchołków algorytm wyszukuje minimalne doskonałe dopasowanie[[11]](#footnote-11) M, tworząc tym samym multigraf[[12]](#footnote-12) będący grafem Eurela. Następnie zadaniem algorytmu jest przejście przez cały cykl Eurela w grafie oraz dokonanie przekształcenia go w cykl Hamiltona za pomocą pominięcia powtarzających się wierzchołków. Nierówność trójkąta zapewnia, że taki proces nie zwiększa całkowitego kosztu trasy.

Algorytm Christofidesa oparty jest na nierówności trójkąta, stanowiącej dla dowolnych wierzchołków koszt bezpośredniego przejścia z wierzchołka u do wierzchołka w, który jest mniejszy lub równy sumie kosztów przejścia z wierzchołka u do v oraz z v do w, co można wyrazić za pomocą wzoru . Właściwość ta zapewnia algorytmowi możliwość stworzenia cyklu Hamiltona o koszcie nie gorszym niż 1.5 razy kosztu cyklu optymalnego. Na rysunku X zaprezentowane zostało działanie algorytmu na przykładzie grafu.



Powyżej umieszczony rysunek przedstawia krok po kroku etapy algorytmu Christofidesa na przykładzie grafu, przechodząc przez kolejne etapy tworzenia minimalnego drzewa rozpinającego, doskonałego dopasowania oraz przejścia do cyklu Hamiltona. Podpunkt a przedstawia pełny graf nieskierowany G, w którym wierzchołki są punktami na przecięciu linii siatki całkowitoliczbowej. Kolejny podpunkt przedstawia minimalne drzewo rozpinające T dla grafu pełnego G, wyznaczone za pomocą algorytmu Prima. Wierzchołek a stanowi korzeń. Podpunkt c przedstawia przejście drzewa T metodą preorder, tworząc cykl Hamiltona. Wierzchołki odwiedzane są w kolejności a,b,c,h,d,e,f,g. Kolejna część rysunku przedstawia minimalne doskonałe dopasowanie M dla wierzchołków o nieparzystym stopniu w T. Ostatni element stanowi połączenie krawędzi drzewa T dopasowania M, tworząc multigraf Eurela, z którego cykl Eurelowski jest przekształcany w cykl Hamiltona poprzez omijanie powtarzających się wierzchołków.

## Algorytm Little’a

(Anholcer, 2023) Algorytm Little’a to efektywne podejście do rozwiązania problemu komiwojażera, którego celem jest minimalizacja kosztu całkowitego podróży pomiędzy miastami, powracając na końcu do punktu wyjścia. (EconPapers, 2023) Został opracowany w 1963 roku przez Johna D. C. Little’a, Katta G. Murty’ego, Dura W. Sweeney’a oraz Caroline Karel. (Anholcer, 2023) Polega on na systematycznym redukowaniu macierzy kosztów podróży pomiędzy miastami, gdzie każda z komórek macierzy reprezentuje koszt podróży pomiędzy miastem i oraz j. Redukcja osiągana jest poprzez odjęcie najmniejszego elementu z każdego wiersza oraz kolumny, co niesie za sobą obniżkę kosztów bez zmiany relatywnej efektywności różnych tras.

Proces algorytmu składa się z trzech kroków: redukcji macierzy kosztów, wyboru i zakazania ścieżek oraz rekurencyjnego podziału problemu. Na wstępie algorytm redukuje macierz celem zmniejszenia kosztu ogólnego bez zmiany optymalnego rozwiązania poprzez odjęcie najmniejszego elementu w każdym wierszu. Proces ten opisany jest wzorem:

(X)

Gdzie:

* – symbolizuje koszt podróży pomiędzy miastami i oraz j
* – minimalna wartość w wierszu i

Kolejny krok to redukcja macierzy poprzez odjęcie najmniejszej wartości z każdej kolumny, co zostało opisane wzorem:

(X)

We wzorze X reprezentuje minimalną wartość w kolumnie j po przeprowadzeniu redukcji wierszy. W kolejnym etapie wybierane są ścieżki, które nie tworzą cykli, minimalizując przy tym koszty podróży. W celu uniknięcia cykli mogących zakłócić rozwiązanie, algorytm Little’a przypisuje niektórym ścieżką nieskończenie duże koszty. Po dokonaniu wyboru ścieżki pomiędzy miastem i oraz j, koszt wykluczenia obliczany jest jako suma najmniejszych wartości z pozostałych elementów wiersza oraz kolumny, z której ścieżka została wybrana, co można zapisać wzorem:

(X)

We wzorze x symbolizuje oszacowany koszt zamknięcia ścieżki pomiędzy miastami i oraz j, natomiast k to indeksy pozostałych miast. Kolejnym etapem jest podział problemu na mniejsze zarządzalne segmenty, eliminując tym samym nieoptymalne ścieżki, skupiając uwagę wyłącznie na najbardziej obiecujących trasach. Proces dzieli się na coraz mniejsze fazy, w których każdy kolejny podział jest traktowany jako nowy problem komiwojażera do rozwiązania, do momentu osiągnięcia minimalnego kosztu podróży.

Dla zobrazowania działania algorytmu Little’a przedstawiony zostanie przykład zadania. W tabeli 1 znajduje się macierz kosztów. Dodatkowy wiersz oraz kolumna zawierają koszty redukcji obliczone za pomocą równań numer X oraz X. Przykładowo dla przykładu a1 minimalny koszt wynosi . W przypadku kolumn obliczane jest minimum różnic kosztów dla tras oraz kosztów redukcji w wierszach np. dla b3 koszt wynosi:

Tabela 2

Początkowe koszty dla tras i koszty redukcji dla podzbioru D0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Do | M | W1 | W2 | W3 | ai |
| M | ∞ | 1 | 7 | 2 | 1 |
| W1 | 6 | ∞ | 7 | 4 | 4 |
| W2 | 8 | 3 | ∞ | 5 | 3 |
| W3 | 2 | 7 | 4 |  | 2 |
| bj | 0 | 0 | 2 | 0 |  |

[*źródło:* opracowanie własne na podstawie (Anholcer, 2023)]

W kolejnym kroku należy dokonać redukcji kosztów odejmując od każdego kosztu w tabeli koszty redukcji wyznaczone dla wiersza oraz kolumny zgodnie ze wzorem X:

. (X)

Przykładowo dla tras z pierwszego wierszu wyniki wynoszą: , . Wyniki redukcji kosztów przedstawione zostały w tabeli 2.

Tabela 3

Zredukowana macierz kosztów

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Do | M | W1 | W2 | W3 |
| M |  | 0 | 4 | 1 |
| W1 | 2 |  | 1 | 0 |
| W2 | 5 | 0 |  | 2 |
| W3 | 0 | 5 | 0 |  |

Kolejnym krokiem jest oszacowanie kosztów redukcji, które dotychczas wynosiło 0. Oszacowanie obliczane jest według wzoru:

(X)

Dla przykładu z tabeli 2 oszacowanie wynosi: . Z uwagi na fakt, że zbiór D0 nie jest pusty, a macierz kosztów ma wymiary 4 x 4, oznacza to, że zbiór ten nie jest jednoelementowy i nie można zamknąć go na tym etapie, ponieważ jest jedynym aktywnym zbiorem i ulega podziałowi. Aby dokonać podziału należy wyznaczyć koszt rezygnacji z poszczególnych tras za pomocą wzoru X. Przykładowo dla trasy <M, W1> koszt wynosi: .

Tabela 4

Koszty rezygnacji z podzbioru D0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Do | M | W1 | W2 | W3 |
| M |  | 1 | 4 | 1 |
| W1 | 2 |  | 1 | 2 |
| W2 | 5 | 2 |  | 2 |
| W3 | 2 | 5 | 1 |  |

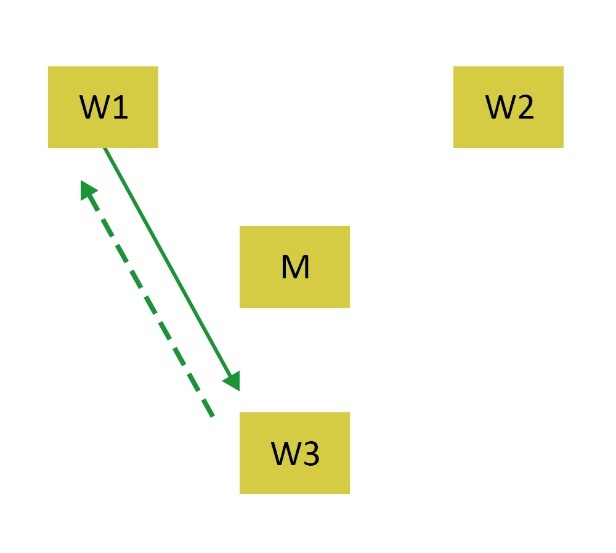
Koszty podziału przedstawione zostały w tabeli 3, ich najwyższy wynik wynosi 2, co odpowiada trasom <W1, W3>, <W2, W1> oraz <W3, M>. Z kolei podzbiór D1 powstał jako zbiór cykli zawierający wyłącznie trasę <W1, W3>. Aby utworzyć macierz dla zbioru D1 należy wykreślić wiersz W1, ponieważ z tego miejsca nie można już udać się nigdzie indziej, a także kolumnę W3, ze względu na fakt, że nie można już do niej przyjechać z żadnej miejscowości. Powstała nowa macierz przedstawiona została w tabeli 4.

Tabela 5

Macierz kosztów po wykreśleniu trasy <W1, W3>

|  |  |  |  |
| --- | --- | --- | --- |
| D1 | M | W1 | W2 |
| M |  | 0 | 4 |
| W2 | 5 | 0 |  |
| W3 | 0 | 5 | 0 |

Kolejnym krokiem, który należy wykonać jest zablokowanie trasy <W3, W1>, w celu niedopuszczenia do powstania cyklu pomiędzy dwoma miejscowościami. Proces ten przedstawiony został na rysunku X, gdzie przerywana linia symbolizuje zablokowaną linię.



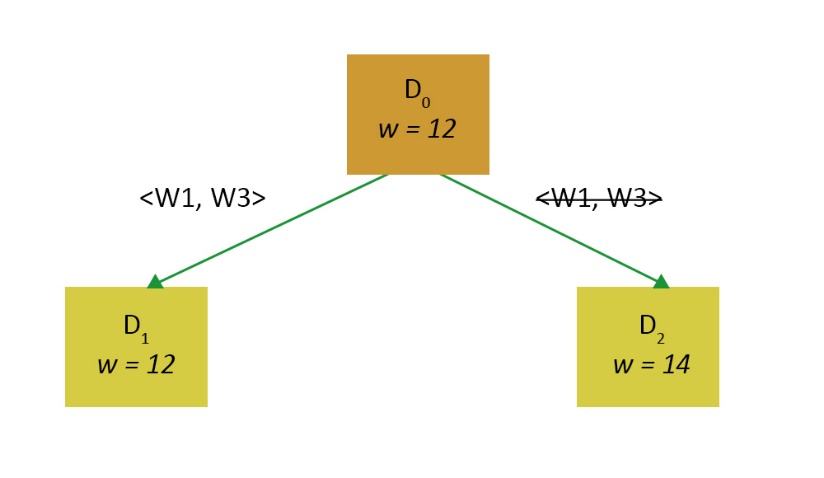
Aby dokonać blokady trasy <W3, W1> należy wstawić ∞ w miejsce ich kosztu w tabeli, co zostało przedstawione w tabeli 5.

Tabela 6

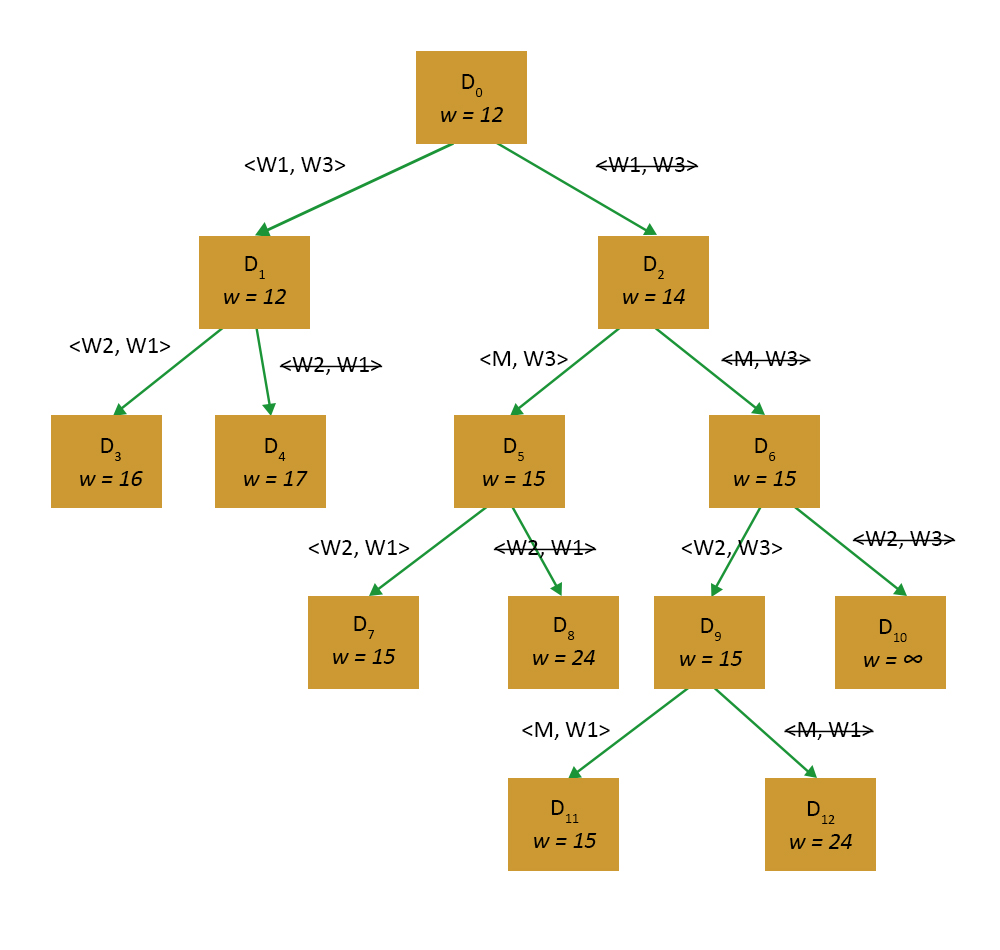
Macierz kosztów dla podzbioru D1 po zablokowaniu trasy <W3, W1>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D1 | M | W1 | W2 | **ai** |
| M |  | 0 | 4 | **0** |
| W2 | 5 | 0 |  | **0** |
| W3 | 0 | 5 | 0 | **0** |
| **bj** | **0** | **0** | **0** |  |

W kolumnie ai oraz wierszu bj zamieszczone zostały koszty po redukcji. Ze względu na fakt, że ich suma wynosi , natomiast podzbiór D1 powstał z podzbioru D0 oszacowany koszt dla niego jest równy . Aby wyznaczyć oszacowanie dla podzbioru D2 należy dodać do wcześniej obliczonego oszacowania koszt rezygnacji z trasy <W1, W3>, który wynosił 2. Zatem otrzymujemy . Wyniki, które zostały dotychczas uzyskane prezentowane są na drzewie rozwiązań, które zostało przedstawione na rysunku X. Kolor pomarańczowy symbolizuje zbiory zamknięte, natomiast żółty otwarte.



Proces podziałów oraz redukcji kontynuowany jest, aż do momentu, gdy uzyskane zostaną drzewa jednocelowych zbiorów. Ze względu na złożoność tego procesu poniżej przedstawione zostało ostateczne drzewo rozwiązań, które zilustrowane zostało na rysunku X. W wyniku analizy powstały trzy zbiory jednoelementowe, do których zaliczamy D3 (cykl (M, W2, W1, W3, M), koszt równy 16), D7 (cykl (M, W3, W2, W1, M), koszt równy 15) oraz D11 (cykl (M, W1, W2, W3, M), koszt równy 15. Ze względu na najniższy koszt zbiory D7 oraz D11 są optymalne.



# Cel i zakres pracy

Celem pracy jest implementacja oraz analiza efektywności wybranych algorytmów rozwiązujących problem komiwojażera (TSP – Travelling Salesman Problem). Praca ma na celu porównanie algorytmów pod kątem ich złożoności obliczeniowej oraz skuteczności w znajdowaniu optymalnych, bądź suboptymalnych tras dla komiwojażera. Podczas realizacji celu pracy przeanalizowane zostaną różnego rodzaju algorytmy, za pomocą których zostanie podjęta próba rozwiązania problemu komiwojażera.

Wybrane podczas analizy algorytmy zostaną zaimplementowane w języku C++ w postaci programów komputerowych.

# Metodyka badań

W celu doboru metodyki badań odpowiedniej dla tej pracy magisterskiej niezbędny był przegląd dostępnej literatury, co otworzyło możliwość na zbudowanie fundamentu wiedzy w temacie analizy oraz efektywności algorytmów rozwiązywania problemu komiwojażera. Dzięki wiedzy zdobytej poprzez szeroki przegląd wszelkiej dostępnej literatury opracowano plan badawczy mający na celu porównanie wybranych algorytmów, za pomocą których istnieje możliwość rozwiązania problemu komiwojażera.

# Implementacja komputerowa

# Badanie porównawcze algorytmów

# Podsumowanie i wnioski

# Bibliografia

abcdef.wiki. (2024, Maj 20). Pobrano z lokalizacji https://pl.abcdef.wiki/wiki/Metric\_%28mathematics%29

Anholcer, M. (2023). *Badania operacyjne.* Poznań: Uniwersystet Ekonomiczny w Poznaniu.

Błaszkiewicz, D. (2011, Maj 11). *Instytut Informatyki Uniwersytetu Wrocławskiego.* Pobrano z lokalizacji https://ii.uni.wroc.pl/~prz/2011lato/ah/opracowania/alg\_mrow.opr.pdf

Brilliant.org. (2024, Maj 08). Pobrano z lokalizacji https://brilliant.org/wiki/complexity-classes/

CodingDrills. (2024, Maj 15). Pobrano z lokalizacji https://www.codingdrills.com/tutorial/introduction-to-greedy-algorithms/traveling-salesman

Cook, W. J. (2012). *In Pursuit of the Traveling Salesman : Mathematics at the Limits of Computation.* Princeton: Princeton University Press.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. i Stein, C. (2007). *Wprowadzenie do algorytmów.* Warszawa: WTN.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. i Stein, C. (2009). *Introduction to Algorithms.* London: Massachusetts Institute of Technology.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. i Stein, C. (2018). *Wprowadzenie do algorytmów.* Warszawa: PWN.

Debudaj-Grabysz, A., Deorowicz, S. i Widuch, J. (2012). *Algorytmy i struktury danych - Wybór zaawansowanych metod.* Gliwice: Wydawnictwo Politechniki Śląskiej.

Dorigo, M. i Stützle, T. (2004). *Ant Colony Optimization.* London: MIT Press.

EconPapers. (2023, Listopad 11). Pobrano z lokalizacji https://econpapers.repec.org/article/inmoropre/v\_3a11\_3ay\_3a1963\_3ai\_3a6\_3ap\_3a972-989.htm

Encyklopedia Algorytmów. (2017, Czerwiec 17). Pobrano z lokalizacji http://algorytmy.ency.pl/artykul/k\_opt

Encyklopedia Algorytmów. (2020, Maj 1). Pobrano z lokalizacji http://algorytmy.ency.pl/artykul/algorytmy\_heurystyczne

Kohlstedt, K. (2022, Październik 4). *99percentinvisible*. Pobrano z lokalizacji https://99percentinvisible.org/article/the-seven-bridge-problem-how-an-urban-puzzle-inspired-a-new-field-of-mathematics/

Kowalik. (2011, Czerwiec 29). *Informatyka MIMUW*. Pobrano z lokalizacji http://smurf.mimuw.edu.pl/node/1122

Ladner, R. E. (1975). *On the structure of polynomial time reducibility.* Journal of the ACM.

Sidford, A. (2020, Wrzesień 15). *Stanford, University.* Pobrano z lokalizacji https://web.stanford.edu/~sidford/courses/20fa\_opt\_theory/sidford\_2020fa\_mse213\_cs269o\_lec1.pdf#:~:text=URL%3A%20https%3A%2F%2Fweb.stanford.edu%2F~sidford%2Fcourses%2F20fa\_opt\_theory%2Fsidford\_2020fa\_mse213\_cs269o\_lec1.pdf%0AVisible%3A%200%25%20

Strąk, Ł. L. (2017, Luty 06). Adaptacyjny algorytm optymalizacji stadnej cząsteczek dla dynamicznego problemu komiwojażera. Praca doktorska. Katowice : Uniwersytet Śląski. Katowice, Śląskie, Polska.

Sysło, M., Deo, N. i Kowalik, J. (1995). *Algorytmy optymalizacji dyskretnej z programami w języku Pascal.* Warszawa: PWN.

Toksari, M. D. (2015, Grudzień 03). *ScieneDirect.* Pobrano z lokalizacji https://www.sciencedirect.com/science/article/abs/pii/S0142061515005840

Traveling Salesman Problem. (2015, Marzec). Pobrano z lokalizacji https://math.uwaterloo.ca/tsp/concorde/index.html

Vasava, M. (2019, Sierpień 03). *Medium.com*. Pobrano z lokalizacji https://medium.com/@mitalvasava411/what-is-a-minimum-spanning-tree-2be2b3cec66a

Weisstein, E. W. (2024, Maj 03). Pobrano z lokalizacji MathWorld - A Wolfram Web Resource: https://mathworld.wolfram.com/EulerianCycle.html

Wikipedia. (2023, Maj 16). Pobrano z lokalizacji https://pl.wikipedia.org/wiki/Zagadnienie\_mostów\_królewieckich

Wikipedia. (2023, Luty 10). Pobrano z lokalizacji https://pl.wikipedia.org/wiki/Algorytm\_aproksymacyjny

Wikipedia. (2023, Grudzień 23). Pobrano z lokalizacji https://en.wikipedia.org/wiki/Concorde\_TSP\_Solver

Wikipedia. (2024, Marzec 25). Pobrano z lokalizacji https://pl.wikipedia.org/wiki/Twierdzenie\_o\_czterech\_barwach

Wikipedia. (2024, Maj 1). Pobrano z lokalizacji https://pl.wikipedia.org/wiki/Graf\_(matematyka)

Wikipedia. (2024, Marzec 25). Pobrano z lokalizacji https://pl.wikipedia.org/wiki/Twierdzenie\_Diraca

Wikipedia. (2024, Marca 1). Pobrano z lokalizacji https://en.wikipedia.org/wiki/Vertex\_cover

Wikipedia. (2024, Maj 07). Pobrano z lokalizacji https://en.wikipedia.org/wiki/Computational\_complexity\_theory

Wikipedia. (2024, Maj 08). Pobrano z lokalizacji https://en.wikipedia.org/wiki/Travelling\_salesman\_problem

Wikipedia. (2024, Maj 11). Pobrano z lokalizacji https://en.wikipedia.org/wiki/Metric\_space

Wikipedia. (2024, Styczeń 08). Pobrano z lokalizacji https://pl.wikipedia.org/wiki/Przestrzeń\_euklidesowa

Wikipedia. (2024, Luty 28). Pobrano z lokalizacji https://en.wikipedia.org/wiki/Taxicab\_geometry

Wilson, R. J. (2000). *Wprowadzenie do teorii grafów.* Warszawa: PWN.

Wojciechowski, J. i Pieńkocz, K. (2013). *Grafy i sieci.* Warszawa: PWN.

World Traveling Salesman Problem. (2021, Luty 15). Pobrano z lokalizacji https://www.math.uwaterloo.ca/tsp/world/

Yadav, A. (2022, Kwiecień 03). *Medium.com*. Pobrano z lokalizacji https://aditya-yadav.medium.com/kruskals-algorithm-for-minimum-spanning-tree-f3c0c0b7b386

Zieliński, J. (2024). Heurystyczny algorytm komiwojażera oparty na selekcji krawędzi. *(Artykuł niepublikowany)*.

# Spis rysunków

[Rysunek 1. Plan miasta Królewiec z czasów Eulera 6](#_Toc166106237)

[Rysunek 2. Przedstawienie mostów w Królewcu według Eulera 7](#_Toc166106238)

[Rysunek 3. Wykres przedstawiający reprezentacje Mostów Królewieckich 7](#_Toc166106239)

[Rysunek 4. Rozwiązanie Eurela problemu wędrówki rycerza 9](#_Toc166106240)

[Rysunek 5. Trasa rycerska na wykresie szachownicy. 9](#_Toc166106241)

[Rysunek 6. Icosian 10](#_Toc166106242)

[Rysunek 7. Cykl Hamiltona w grafie. 12](#_Toc166106243)

[Rysunek 8. Pokolorowana mapa cyklu Hamiltona. 12](#_Toc166106244)

[Rysunek 9. Zoptymalizowana trasa podróży po stanie Maine z 1925 roku 15](#_Toc166106245)

[Rysunek 10. Zrzut ekranu z programu Concorde 17](#_Toc166106246)

[Rysunek 11. Graf eurelowski 21](#_Toc166106247)

[Rysunek 12. Graf półeurelowski 21](#_Toc166106248)

[Rysunek 13. Graf hamiltonowski 23](#_Toc166106249)

[Rysunek 14. Graf półhamiltonowski 24](#_Toc166106250)

[Rysunek 15. Graf niehamiltonowski 25](#_Toc166106251)

[Rysunek 16. Hierarchia problemów decyzyjnych 26](#_Toc166106252)

# Spis tabel

[Tabela 1 44](#_Toc167128923)

1. Czas wielomianowy – klasa złożoności algorytmów, których czas działania rośnie wielomianowo w stosunku do rozmiaru danych wejściowych. [↑](#footnote-ref-1)
2. „Solutio problematis ad geometriam situs pertinentis w Commentarii academiae scientiarum Petropolitanae.” – z łacińskiego „Rozwiązanie problemu dotyczącego geometrii stanowiska w Komentarzach Petropolitan Academy of Sciences.” [↑](#footnote-ref-2)
3. Solver – narzędzie komputerowe służące do automatycznego znajdowania rozwiązań złożonych problemów matematycznych. [↑](#footnote-ref-3)
4. CPLEX – zaawansowane oprogramowanie do optymalizacji liniowej i całkowitoliczbowej. [↑](#footnote-ref-4)
5. Algorytm Lin-Kernighan to heurystyka do rozwiązywania problemu komiwojażera, opierająca się na metodzie lokalnego przeszukiwania w celu poprawy istniejącej trasy przez zamianę segmentów. [↑](#footnote-ref-5)
6. Maszyna Turinga – teoretyczny model obliczeniowy używany do opisu algorytmów, składający się z nieskończenie długiej taśmy, głowicy odczytującej, zapisującej oraz zestawu instrukcji stresujących działaniem głowicy. [↑](#footnote-ref-6)
7. MST – Minimum Spanning Tree – minimalne drzewo rozpinające grafu ważonego to podzbiór krawędzi, który łączy wszystkie wierzchołki grafu w taki sposób, że suma wag tych krawędzi jest minimalna i nie zawiera cykli. [↑](#footnote-ref-7)
8. FIND-SET - Operacja w strukturze zbiorów rozłącznych, która zwraca reprezentanta zbioru zawierającego dany element. [↑](#footnote-ref-8)
9. UNION - Operacja w strukturze zbiorów rozłącznych, która łączy dwa zbiory w jeden. [↑](#footnote-ref-9)
10. Graf regularny - graf, w którym każdy wierzchołek ma ten sam stopień. [↑](#footnote-ref-10)
11. **Minimalne doskonałe dopasowanie** (ang. perfect matching) to zbiór krawędzi w grafie, który łączy wszystkie wierzchołki w pary, tak aby każdy wierzchołek był dokładnie raz końcem jednej z krawędzi, a suma wag tych krawędzi była najmniejsza możliwa. [↑](#footnote-ref-11)
12. **Multigraf** - graf, w którym między dowolnymi dwoma wierzchołkami może istnieć więcej niż jedna krawędź, a także mogą występować pętle, czyli krawędzie łączące wierzchołek sam ze sobą. [↑](#footnote-ref-12)