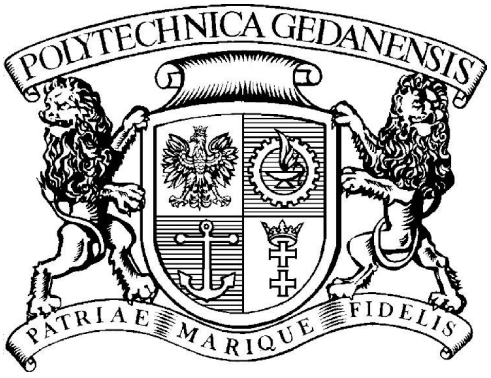


Politechnika Gdańska
Wydział Fizyki Technicznej i Matematyki Stosowanej

PROJEKT INŻYNIERSKI

Problem wielu komiwojażerów

Łukasz Biedak
Biomatematyka sem. VII
06.XII.2011r.

Spis treści

Wstęp

Niniejsza praca poświęcona jest problemowi, który łączy ze sobą trzy dzedziny wiedzy, tj. Matematykę, Informatykę i Transport. Problem wielu komiwojażerów jest silnie związany z Teorią Grafów, Kombinatoryką, Optymalizacją, Algorytmiką i Logistyką. Mimo takiej interdyscyplinarności nie jest on trudny w zrozumieniu. Jak w wielu zadaniach optymalizacyjnych cel jest jasno określony - należy wykonać zadanie tak, aby ponieść najniższy koszt.

W pracy bardzo często stosowane są skróty pochodzące od angielskich nazw problemów powiązanych z problemem wielu komiwojażerów. Używa się **TSP** (*Travelling Salesman Problem*) – problem komiwojażera; **mTSP** (*multiple Travelling Salesmen Problem*) – problem wielu komiwojażerów; **VRP** (*Vehicle Routing Problem*) – problem marszrutyzacji. Dwa ostatnie należą do klasy problemów wielokryterialnych.

Do projektu został dołączony kod programu w języku Matlab napisany przez autora projektu. Został on utajniony. Jego istnienie i oryginalność mają możliwość sprawdzić jedynie promotor tej pracy, jak również recenzent. Zdecydowałem się tak postąpić ze względu na możliwość komercyjnego wykorzystania kodu. Nie uważam, że kod stanowi przełom w pracach dotyczących szukania rozwiązań VRP. Nie wymyśliłem nowej metody rozwiązywania problemu, to znaczy prawdopodobnie ktoś wcześniej napisał coś podobnego. Jednakże sam będę go stosował w komercyjnym przedsięwzięciu (oczywiście na innej platformie niż Matlab), dlatego też chcę zachować pełnię praw autorskich do tego algorytmu. Mam nadzieję, że czytelnik i komisja dyplomowa uszanują mój wybór, a utajnienie kodu nie będzie miało wpływu na ocenę projektu.

Rozdział 1

Wprowadzenie

1.1 Podstawowe definicje

Zanim przejdziemy do omawiania problemu komiwojażera, a później problemu wielu komiwojażerów, konieczne jest wprowadzenie definicji pojęć[1], którymi będziemy posługiwać się w tej pracy. Niektóre z nich opatrzone są komentarzem, który uzasadnia konieczność wprowadzenia danej definicji.

Definicja 1.1 *Grafem nieskierowanym $G = (V, E)$ nazywamy parę złożoną ze skończonego zbioru wierzchołków V oraz zbioru wszystkich krawędzi E łączących wierzchołki. Zbiór E definiujemy jako dwuelementowe podzbiory V, czyli*

$$E \subseteq \{\{u, v\} : u, v \in V, u \neq v\}.$$

W celu uproszczenia zapisu będziemy stosować równoważność $E \ni \{u, v\} \equiv uv$. *Stopniem wierzchołka* $d(v)$ nazywać będziemy liczbę krawędzi wychodzących z v.

Definicja 1.2 *Grafem skierowanym $G = (V, E)$ nazywamy parę złożoną ze skończonego zbioru wierzchołków V i ze zbioru krawędzi $E \subseteq V \times V$.*

Zauważmy, że w grafie skierowanym krawędź $e = (u, v)$ jest skierowana z wierzchołka u do wierzchołka v. W niektórych modelach konieczne jest stosowanie grafów skierowanych, w których krawędzie mogą reprezentować drogi jednokierunkowe lub w przypadku górzystego terenu, gdzie koszt przejazdu pomiędzy punktami A i B różni się od kosztu przejazdu między B i A.

Definicja 1.3 *Graf G nazywamy pełnym, jeżeli dla dowolnych $u, v \in V$ krawędź $uv \in E$.*

Każdy wierzchołek v w grafie pełnym ma stopień $d(v) = |V| - 1$, gdzie $|V|$ oznacza liczbę wierzchołków danego grafu. W większości spotykanych modeli dotyczących VRP mapy miast są reprezentowane przez grafy pełne. Nawet w przypadku, gdy jedyne połączenie drogowe pomiędzy miastami A i C przebiega przez miasto B to punkt odbioru towaru może być „nie po drodze" lub miasto B może mieć obwodnicę.

[1]Większość definicji zaczerpniętych jest z [1] - *Matematyka dyskretna* A. Szepietowski oraz z [2] - *Matematyka dyskretna* K.A. Ross, C.R.B. Wright.

Definicja 1.4 *Graf G nazywamy ważonym, jeśli każdej krawędzi $uv \in E$ przyporządkowana jest liczba rzeczywista*

$$w : E \to \mathbb{R}$$

nazywana wagą (kosztem) krawędzi uv.

W naszych rozważaniach dopuszczać będziemy jedynie dodatnie wagi krawędzi $w : E \to \mathbb{R}_+$.

Definicja 1.5 *Drogą długości n w grafie G nazywamy taki ciąg wierzchołków $(v_0, v_1, \ldots, v_{n-1})$, dla którego*

$$v_0, v_1, \ldots, v_{n-1} \in V$$

$$\forall_{i \in \{0,1,\ldots,n-2\}} \; v_i v_{i+1} \in E.$$

Definicja 1.6 *Ścieżką długości n w grafie G nazywamy drogę $(v_0, v_1, \ldots, v_{n-1})$, w której*

$$\forall_{i,j \in \{0,1,\ldots,n-1\}} \; i \neq j \Rightarrow v_i \neq v_j.$$

Definicja 1.7 *Cyklem prostym długości n nazywamy ścieżkę powiększoną o v_0, co zapisujemy jako ciąg $(v_0, v_1, \ldots, v_{n-1}, v_0)$.*

Definicja 1.8 *Cyklem Hamiltona grafu G nazywamy cykl prosty h, do którego należy każdy wierzchołek z V, czyli $h = (v_0, v_1, \ldots, v_{|V-1|}, v_0)$.*

Warto dodać, że graf, który posiada cykl Hamiltona jest nazywany *grafem hamiltonowskim*. W pracy skupiono się na analizie problemu VRP dla grafów pełnych, w których dowolna permutacja wszystkich wierzchołków jest cyklem Hamiltona. Zasadne jest zatem wprowadzenie kombinatorycznej definicji permutacji zbioru.

Definicja 1.9 *Permutacją bez powtórzeń zbioru złożonego z n różnych elementów nazywamy każdy n-wyrazowy ciąg utworzony ze wszystkich wyrazów zbioru. Wszystkich możliwych permutacji zbioru n - elementowego jest*

$$P_n = n!.$$

Permutacja jest szczególnym przypadkiem wariacji bez powtórzeń.

Definicja 1.10 *Wariacją bez powtórzeń zbioru złożonego z n różnych elementów nazywamy każdy k-wyrazowy ($1 \leqslant k \leqslant n$) ciąg z k różnych wyrazów zbioru. Liczbę wszystkich k-elementowych wariacji zbioru n-elementowego wyraża się wzorem*

$$V_n^k = \frac{n!}{(n-k)!}.$$

Zauważmy, że $V_n^{n-1} = V_n^n = \frac{n!}{(n-n)!} = \frac{n!}{0!} = n! = P_n$.

Definicja 1.11 *Kombinacją k–elemntową zbioru n–elementowego A nazywamy każdy podzbiór k–elementowy zbioru A. Liczbę takich podzbiorów możemy policzyć ze wzoru*

$$C_n^k = \frac{n!}{k! \cdot (n-k)!}.$$

Definicja 1.12 *Łączną wagą cyklu Hamiltona h nazywamy funkcję μ wyrażoną wzorem*

$$\mu(h) = \sum_{i=0}^{|V|-2} w(h_i h_{i+1}) + w(h_{|V|-1} h_0). \qquad (*)$$

W tej pracy często będziemy używać sformułowań bliższych logistyce, transportowi. Oczywiście analizowany będzie pewien model matematyczny, lecz słownictwo z życia codziennego niematematyka ułatwi zrozumienie pracy i podkreśla szeroką gamę zastosowań omawianego modelu. Z tego powodu często cykl będziemy nazywać trasą, wagę krawędzi – kosztem podróży z miasta A do miasta B, a łączną wagę cyklu – kosztem przejazdu trasy, czasami również długością trasy.

Wszystkie powyższe definicje wprowadzamy aby sformułować problem komiwojażera.

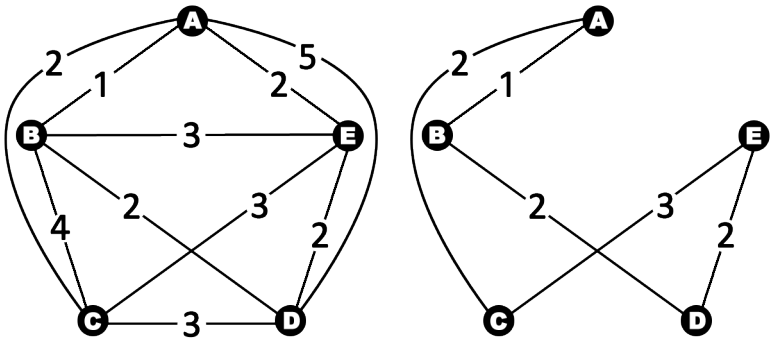
1.2 Problem komiwojażera

1.2.1 Definicja problemu komiwojażera

Definicja 1.13 (klasyczna) *Komiwojażer musi odwiedzić N miast tak, aby w każdym znaleźć się tylko raz i aby długość przebytej przez niego drogi była jak najkrótsza.*

Definicja 1.14 (formalna) *Dla ważonego, hamiltonowskiego grafu $G = (V, E)$ należy znaleźć cykl Hamiltona o najmniejszej łącznej wadze $\mu(h)$ (*).*

Powyższe definicje są sobie równoznaczne jeśli przyjmiemy, że ilość miast i połączenia między nimi w pierwszej definicją są tożsame z licznością zbiorów V i E. W definicji formalnej można zakładać więcej „wygodnych" własności grafu G. W zasadzie najczęściej TSP dotyczy grafów pełnych bądź gęstych[2]. Problem komiwojażera możemy analizować zarówno w grafach skierowanych jak i nieskierowanych. Mówimy wtedy o asymetrycznym (ATSP) i symetrycznym (STSP) problemie komiwojażera.



Rysunek 1.1: *Przedstawienie nieskierowanego grafu ważonego wraz ze znalezieniem najkrótszego cyklu Hamiltona.*

[2] *Graf gęsty* to taki, w którym liczba krawędzi jest duża w stosunku do liczby jego wierzchołków. Jak łatwo policzyć, w grafie pełnym liczba krawędzi wynosi $|E| = \frac{|V|(|V|-1)}{2}$, w przypadku grafu nieskierowanego oraz $|V|(|V|-1)$, w przypadku grafu skierowanego. Dlatego najczęściej za grafy gęste uważa się takie, w których $|E|$ jest zbliżona do $|V|^2$.

1.3 Analiza algorytmów

Z punktu widzenia każdego programisty podczas wyboru algorytmu[3] do rozwiązywania określonego zagadnienia, bądź też projektowania nowego najistotniejsze są cztery jego cechy - **prostota**, **poprawność**, oraz jego **złożoność czasowa** i **pamięciowa**.

1.3.1 Prostota algorytmu

Najistotniejsza cecha z punktu widzenia każdego początkującego programisty. Zastosowanie algorytmu w swoim projekcie wymaga zrozumienia go, co pomaga uniknąć błędów. Niekiedy jednak stosowanie najprostszych sposobów powoduje znaczne wydłużenie czasu szukania rozwiązania problemu, co zmusza nas do użycia metod bardziej rozbudowanych.

1.3.2 Poprawność algorytmu

Naturalne jest oczekiwanie, żeby algorytm dla dowolnych danych spełniających warunki wejściowe będzie rozwiązywał zadanie poprawnie. Błędy mogą być natury formalnej – na przykład źle posortowana tablica liczb, czy w przypadku TSP zwrócenie rozwiązania niebędącego cyklem Hamiltona grafu G.

Czasami w algorytmach błędy wynikają z nieprecyzyjności w sformułowaniu zagadnienia – na przykład *zadanie polega na znalezieniu* ***najlepszego*** *sposobu dystrybucji towaru*; bądź też trudności w znalezieniu odpowiedniej *funkcji celu*, szczególnie w przypadku problemów wielokryterialnych.

1.3.3 Złożoność czasowa i pamięciowa

Przy obecnych możliwościach komputerów *złożoność pamięciowa* reprezentacji danych, czy algorytmu ma coraz mniejsze znaczenie, jednak w niektórych problemach, gdzie dane są reprezentowane za pomocą dużych i gęstych macierzy, konieczny jest dostęp do odpowiedniej ilości pamięci.

Ze względu na różnorodność dostępnych języków programowania, pakietów matematycznych i różnic w działaniu między nimi, nie można mówić o *złożoności czasowej* algorytmu używając jednostek tradycyjnych. Dlatego przyjętą miarą złożoności czasowej jest liczba operacji podstawowych w zależności od rozmiaru danych wejściowych wyrażana funkcją $T(n)$. W przypadku TSP najczęściej n odpowiada liczbie $|V|$, niekiedy również $|E|$. Za operacje podstawowe uważa się między innymi podstawienie, porównanie, czy podstawowe operacje arytmetyczne.

Aby móc porównywać algorytmy, trzeba obiektywnie ocenić ich złożoność. W tym celu wykorzystuje się abstrakcyjne modele obliczeń, do których należą maszyna RAM, czy maszyna Turinga[4], dzięki którym unika się niejednoznaczności w ocenie. Do porównania złożoności czasowej algorytmów używa się notacji asymptotycznej.

[3]Bardzo przystępny opis analizy algorytmów przedstawia Thomas Cormen w [3].

[4]Więcej o maszynie RAM i maszynie Turinga można przeczytać w można przeczytać w [4].

Definicja 1.15 *Dla funkcji* $g : \mathbb{N} \to \mathbb{R}_+$ *oznaczamy przez* $\Theta(g)$ *zbiór funkcji*

$$\Theta(g) = \{f : \exists_{0<c_1<c_2, n_0 \in \mathbb{N}} \ \forall_{n \geqslant n_0} \ c_1 g(n) \leqslant f(n) \leqslant c_2 g(n)\}$$

Zapis $f = \Theta(g)$ czytamy jako: *f jest dokładnie rzędu g.*
Jest to jeden z trzech[5] rodzajów notacji asymptotycznej. Pozostałe z nich – $O(g)$ oraz $\Omega(g)$ oznaczają zbiory funkcji odpowiednio *co najwyżej rzędu g* oraz *co najmniej rzędu g.*

1.3.4 Szacowanie złożoności TSP

Istnieje wiele problemów optymalizacyjnych, w których korzystanie z algorytmów dokładnych[6] powoduje niemożliwość znalezienia najlepszego rozwiązania w zadawalającym czasie. Do tych problemów należy problem komiwojażera jak i problem wielu komiwojażerów. Łatwo policzymy, ile możliwości do sprawdzenia ma algorytm szukający optymalnej trasy komiwojażera dla **20 miast**.

> Niech komiwojażer startuje z miasta A, ma więc do odwiedzenia 19 miast co daje 19! cykli Hamiltona w grafie pełnym. Zakładając, że nieistotne jest z której strony cyklu komiwojażer zaczyna podróż nadal mamy $\frac{19!}{2} = 60\ 822\ 550\ 204\ 416\ 000$ możliwości do rozpatrzenia.

Należy pamiętać, że kolejne miasto do odwiedzenia oznacza 20-krotny przyrost ilości cykli, dwa kolejne – 410-krotny. Algorytm sprawdzający po kolei wszystkie możliwe ułożenia miast ma złożoność rzędu $\Theta(n!)$, gdzie $n = |V|$. Taki wzrost jest określany superwykładniczym.

1.4 Klasy złożoności

Problemy obliczeniowe, jakkolwiek byłyby skomplikowane i różne od siebie, grupuje się w klasy. Wprowadzenie wszystkich z nich i dokładne wyjaśnienie wymagałoby odwołania się do *teorii języków formalnych*, *teorii obliczeń* oraz dobrego wyjaśnienia zasady działania *maszyny Turinga.* Nie będziemy tego robić. W pracy uznano za konieczne wymienienie dwóch ważnych klas problemów decyzyjnych. Konieczne jest wprowadzenie definicji.

Definicja 1.16 *Problemem decyzyjnym nazywamy pytanie sformułowane w języku formalnym, na które można udzielić jedynie odpowiedzi* ***tak*** *lub* ***nie****.*

1.4.1 Klasy P i NP

Definicja 1.17 (P) *Klasą złożoności typu P nazywamy zbiór problemów decyzyjnych, które można rozwiązać w czasie wielomianowym* $O(n^p)$*, gdzie n jest rozmiarem danych wejściowych,* $p \in \mathbb{R}_+$

Definicja 1.18 (NP) *Klasą złożoności typu NP nazywamy zbiór problemów decyzyjnych, dla których nie znaleziono algorytmów o złożoności wielomianowej, jednak można w wielomianowym czasie zweryfikować zaproponowane rozwiązanie.*

[5] Jeśli rozróżniać jeszcze notacje pochodne to jeden z pięciu.
[6] Algorytmy dokładne przeszukują całą przestrzeń możliwych rozwiązań i wskazują najlepsze.

Z powyższych definicji możemy wywnioskować, że $P \subset NP$, gdyż skoro można znaleźć rozwiązanie danego problemu decyzyjnego w czasie wielomianowym, to tym bardziej można je w czasie wielomianowym sprawdzić. Pytanie, czy $P = NP$ należy do *problemów milenijnych* za których rozwiązanie otrzymać można nagrodę w wysokości 1 000 000 USD.

Wśród klasy NP wyróżniamy ważną klasę problemów NP–zupełnych.

Definicja 1.19 (NP–zupełny) *Do klasy problemów NP–zupełnych należy każdy problem P_0, który spełnia poniższe warunki:*
(i) $P_0 \in NP$,
(ii) Każdy problem klasy NP da się sprowadzić w czasie wielomianowym do problemu P_0.

Z powyższej definicji wynika fakt, że gdyby udało się rozwiązać dowolny problem NP–zupełny w czasie wielomianowym, to wszystkie problemy NP możnaby rozwiązać w tym czasie, co implikowałoby równość $P = NP$.

1.4.2 NP–zupełność problemu komiwojażera

Jak to zostało powiedziane we wstępie, problem komiwojażera należy do zagadnień optymalizacyjnych. Zanim zakwalifikujemy ów problem do odpowiedniej klasy złożoności sformułujmy go raz jeszcze w postaci problemu decyzyjnego.

Definicja 1.20 (postać decyzyjna TSP) *Dla ważonego i hamiltonowskiego grafu $G(V, E)$ należy sprwadzić, czy istnieje w nim cykl Hamiltona h, którego $\mu(h)$ jest mniejsza od danego $c \in \mathbb{R}_+$.*

Wprowadzimy tutaj twierdzenie, które jest szerzej omówione w [6] wraz ze szkicem dowodu. My pokażemy tylko pierwszy warunek, który musi być spełniony, aby poniższe twierdzenie można było uznać za prawdziwe.

Twierdzenie 1.21 *Decyzyjny problem komiwojażera jest problemem NP–zupełnym.*

Szkic dowodu.
Na podstawie obecnego stanu wiedzy wiemy, że nie znaleziono algorytmu, który rozwiązywałby problem komiwojażera w czasie wielomianowym. Załóżmy, że zaproponowano rozwiązanie problemu w postaci cyklu C_G. Zgodnie z definicją 1.20 została ustalona stała c, musimy teraz zweryfikować prawdziwość nierówności $\mu(C_G) \leqslant c$. Zanim to zrobimy, w czasie wielomianowym sprawdzamy 4 istotne cechy C_G: czy każda para wierzchołków $u, v \in C_G$ jest połączona krawędzią, czy w cyklu C_G żaden wierzchołek się nie powtarza, zaczyna się i kończy w tym samym wierzchołku, oraz czy zbiór wszystkich wierzchołków jest jednocześnie zbiorem wszystkich wierzchołków grafu G. Dzięki temu możemy stwierdzić, że C_G jest cyklem Hamiltona grafu G. Następnie w czasie liniowym sprawdzamy wartość $\mu(C_G)$ i porównujemy z c. Wszystkie operacje weryfikacji poprawności zaproponowanego rozwiązania wykonano w czasie wielomianowym. □

1.5 Metody rozwiązywania problemu komiwojażera

Wiele z metod zostało dokładnie opisanych w [6], wraz z pseudokodem i rysunkami obrazującymi poszczególne kroki metod. My wymienimy najczęściej używane metody w rozwiązywaniu problemu komiwojażera:

- Algorytmy przybliżania trasy, w tym:
  - metody wykorzystujące minimalne drzewo rozpinające,
  - algorytm najbliższego sąsiedztwa,
  - metoda najbliższych krawędzi,
  - metoda wstawiania do cyklu - kilka odmian insercji,
  - algorytm *Clark & Wright*.
- Metody ulepszania otrzymanych tras:
  - najprostsze modyfikacje - przesuwanie i wstawianie wierzchołków, wstawianie krawędzi,
  - metoda *k-opt*,
  - algorytm *Lin-Kernighan*.
- Algorytmy genetyczne i programy ewolucyjne.
- Algorytmy mrówkowe.
- Metoda *branch & cut*.

1.6 Zastosowania TSP

Do dziedzin w których rzeczywiste problemy modeluje się za pomocą problemu komiwojażera należą przede wszystkim:

- logistyka i transport – najczęściej jako część składowa VRP,
- przemysł, automatyka, robotyka – mechaniczne ramię dokręcające śruby w konstrukcji statku/samolotu,
- elektronika – tworzenie układów scalonych,
- energetyka – planowanie sieci elektrycznej,
- genetyka – wykrywanie profili ekspresji genów - tzw. analiza mikromacierzy DNA[7].

[7]Bardzo ciekawy artykuł na ten temat jak również na temat stosowania algorytmów genetycznych w problemie komiwojażera znajduje się w artykule [8] dostępnym pod adresem http://www.ipipan.waw.pl/∼ stw/mh/komiw.pdf.

Rozdział 2

Problem wielu komiwojażerów

2.1 Definicja mTSP

W dostępnej literaturze dużo miejsca zostało poświęcone problemowi komiwojażera oraz różnym odmianom problemu marszrutyzacji. Autor artykułu [7] dostrzega, że problemowi wielu komiwojażerów, sformułowanemu jako uogólnienie problemu komiwojażera, nie poświęcono dotychczas tyle uwagi. Wprowadźmy podstawową definicję problemu mTSP.

Definicja 1.22 *Dokładnie k komiwojażerów ma za zadanie odwiedzić N miast ($N \geqslant k$) tak aby:*
(i) wszyscy komiwojażerowie startowali z tego samego miasta i tam wracali,
(ii) każde miasto zostało odwiedzone przez dokładnie jednego komiwojażera,
(iii) łączny koszt podróży wszystkich komiwojażerów był jak najkrótszy.

Należy pamiętać, że dopuszczalne jest rozwiązanie, w którym jeden z komiwojażerów odwiedza tylko jedno miasto. W takim przypadku pokonuje on dwukrotnie tę samą krawędź.

2.2 Odmiany mTSP

Definicja, którą przed chwilą wprowadziliśmy, nie jest jedyną możliwą dla problemu wielu komiwojażerów. Myślę, że czytelnik po jej wprowadzeniu oraz krótkim zastanowieniu sam zauważy, że mTSP jest zagadnieniem znajdującym szersze zastosowanie niż TSP, szczególnie w transporcie. Dlatego oprócz klasycznej definicji możemy mówić o problemie wielu komiwojażerów z:

1. *Kilkoma magazynami* – w każdym z nich jest umieszczona pewna liczba komiwojażerów, część z nich wraca do swojej macierzystej bazy, część zatrzymuje się w innej.

2. *(Nie)narzuconą liczbą komiwojażerów* – ich liczba może być podana wprost, lub zawierać się w pewnych granicach. Lub odwrotnie, nie ma wymagań odnośnie liczby komiwojażerów.

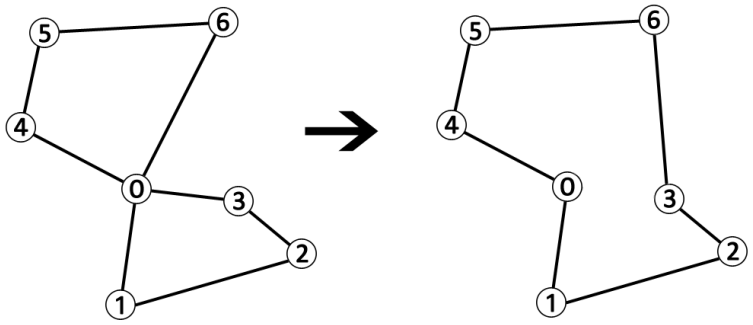
3. *Ograniczeniami dotyczącymi trasy przejazdu* – występują ograniczenia dotyczące na przykład minimalnej i maksymalnej liczby miast, które może odwiedzić komiwojażer lub maksymalnej długości trasy.

4. *Czasowymi ograniczeniami* – dla każdego miasta są zdefiniowane przedziały czasowe, w których może ono zostać odwiedzone.

5. *Kosztami użycia komiwojażerów* – liczba komiwojażerów nie jest ustalona, a każdy dodatkowy komiwojażer zwiększa całkowity koszt, który należy w tym zagadnieniu zminimalizować.

2.2.1 Przewaga TSP nad mTSP

Podczas rozmyślań nad projektem zaobserwowano pewną prawidłowość. Intuicyjnie wydaje się ona być prawidłowa, jednak matematycy wymagają czegoś więcej: dowodu. Zacznijmy jednak od intuicji, później postaramy się przeprowadzić dowód, który wymaga od czytelnika trochę skupienia. Obserwacja dotyczy punktu drugiego – odmiany mTSP z nienarzuconą liczbą komiwojażerów. Poniższy rysunek ma zasugerować hipotezę.



Rysunek 2.1: *Przedstawienie najkrótszej ścieżki dla dwóch komiwojażerów oraz cyklu Hamiltona (nie wiemy, czy jest on najkrótszy). Wagi krawędzi nie zostały wypisane, przyjmujemy, że została użyta metryka euklidesowa.*

Powyższy rysunek sugeruje, że nawet gdy znajdziemy cykle o najkrótszej łącznej wadze dla problemu wielu komiwojażerów, zredukujemy ich liczbę i odpowiednio połączymy trasy, to całkowity koszt podróży się zmniejszy. Zapiszmy to jako twierdzenie.

Twierdzenie 1.23 *W nieskierowanym grafie pełnym i ważonym, w którym krawędzie spełniają nierówność trójkąta*

$$\forall_{t,u,v \in V}\ w(tv) \leqslant w(tu) + w(uv),$$

najkrótsza długość trasy jednego komiwojażera jest zawsze mniejsza bądź równa najmniejszej wartości sumy długości tras wielu komiwojażerów.

Dowód.

Niech graf $G(V, E)$ będzie pełny, nieskierowany i ważony (wagi krawędzi spełniają warunek trójkąta) oraz zawiera k cykli, reprezentujących trasy k komiwojażerów. Oczywiście łączna waga wszystkich k cykli jest najmniejsza. Bez straty ogólności przyjmijmy, że wierzchołki grafu zostały ponumerowane kolejnymi liczbami naturalnymi po znalezieniu tras komiwojażerów, wierzchołek początkowy został oznaczony zerem.

$$n_0 = 0 < n_1 < n_2 < \ldots < n_{k-1} < n_k = |V| - 1.$$

Liczby $n_1, n_2, \ldots, n_k$ określają numery wierzchołków kończących i-ty cykl[1],
$(i \in \{1, 2, \ldots, k\})$.
Łączna suma wag wszystkich k cykli (C_k) jest wyrażona wzorem

$$\omega(C_k) = \sum_{i=0}^{k-1}[w(v_0 v_{n_i+1}) + \sum_{m=n_i+1}^{n_{i+1}-1} w(v_m v_{m+1}) + w(v_{n_{i+1}} v_0)]. \qquad (**)$$

Spełniony jest warunek trójkąta, a zatem

$$\forall_{i\in\{1,2,\ldots,k-1\}}\ w(v_{n_i} v_0) + w(v_0 v_{n_i+1}) \geqslant w(v_{n_i} v_{n_i+1})$$

Wzór (**) po drobnych przekształceniach ma postać

$$\begin{aligned}
\omega(C_k) &= \sum_{i=0}^{k-1}\sum_{m=n_i+1}^{n_{i+1}-1} w(v_m v_{m+1}) + \sum_{i=1}^{k-1}[w(v_0 v_{n_i+1}) + w(v_{n_i} v_0)] + w(v_0 v_1) + w(v_{n_k} v_0) \\
&\geqslant \sum_{i=0}^{k-1}\sum_{m=n_i+1}^{n_{i+1}-1} w(v_m v_{m+1}) + \sum_{i=1}^{k-1} w(v_{n_i} v_{n_i+1}) + w(v_0 v_1) + w(v_{n_k} v_0) \\
&= \sum_{i=0}^{k-1}\sum_{m=n_i+1}^{n_{i+1}-1} w(v_m v_{m+1}) + \sum_{i=0}^{k-1} w(v_{n_i} v_{n_i+1}) + w(v_{n_k} v_0) \\
&= \sum_{i=0}^{k-1}\sum_{m=n_i}^{n_{i+1}-1} w(v_m v_{m+1}) + w(v_{n_k} v_0) \\
&= \sum_{j=0}^{|V|-2} w(v_j v_{j+1}) + w(v_{|V|-1} v_0) \\
&= \mu(h_G).
\end{aligned}$$

□

2.3 Szacowanie złożoności mTSP

Postaramy się teraz przeprowadzić kombinatoryczną analizę złożoności problemu. Najpierw omówimy szczególny przypadek zagadnienia, który ukazuje pewien paradoks złożoności problemu jednego i wielu komiwojażerów. Później postaramy się oszacować złożoność problemu ogólniejszego, jednak korzystać będziemy z klasycznej definicji mTSP.

[1]Przykładowo pierwszy komiwojażer ma odwiedzić 4 miasta, więc $n_1 = 4$, drugi 5 miast, więc $n_2 = 9$.

2.3.1 Przypadek szczególny mTSP. Paradoks złożoności

Niech każdy czytelnik zastanowi się, który z problemów: problem jednego komiwojażera, czy może problem wielu komiwojażerów, wydaje się być bardziej złożony? Co podpowiada intuicja? Prawdopodobnie część z odbiorców bez wahania odpowie, że problem wielu komiwojażerów jest znacznie bardziej złożony obliczeniowo: przecież trzeba wybrać odpowiednie miasta, które dany komiwojażer ma odwiedzić. Następnie należy znaleźć najkrótszą trasę dla każdego komiwojażera. Ci, którzy opowiedzieli się za większą złożonością mTSP wobec TSP mają w ogólności rację. Teraz kolejne pytanie: co w przypadku, gdy narzucona jest liczba komiwojażerów i miast, które mają odwiedzić? Co podpowiada intuicja? Przeanalizujmy ten przypadek.

Niech graf G - pełny i ważony, ma $|V| = N + 1$ [2] wierzchołków. Należy znaleźć najkrótszą łączną trasę dla k komiwojażerów (oczywiście $1 < k < N$). Każdy i-ty komiwojażer ma za zadanie odwiedzić dokładnie $n_i \geqslant 2$ miast startując z magazynu w wierzchołku 0. Oczywiście $\sum_{i=1}^{k} n_i = N$. Policzmy, zatem ile możliwości trzeba przeanalizować, żeby znaleźć najkrótszą trasę. Pierwszy komiwojażer wybiera ze zbioru wszystkich miast ciąg długości n_1 (kolejność ma znaczenie), i-ty wybiera spośród pozostałych miast ciąg długości n_i. Dodatkowo bez znaczenia jest, z której strony zaczniemy podróż, tzn ciągi (12345) i (54321) uznajemy za tożsame. Postarajmy się zapisać wzorem całkowitą liczbę możliwości L_{c_k}:

$$L_{c_k} = \frac{V_N^{n_1}}{2} \cdot \frac{V_{N-n_1}^{n_2}}{2} \cdot \ldots \cdot \frac{V_{N-\sum_{i=1}^{j} n_i}^{n_{j+1}}}{2} \cdot \ldots \cdot \frac{V_{n_{k-1}+n_k}^{n_{k-1}}}{2} \cdot \frac{V_{n_k}^{n_k}}{2} =$$

$$= \left(\frac{1}{2}\right)^k \cdot N \cdot (N-1) \cdot \ldots \cdot (N - \sum_{i=1}^{j} n_i) \cdot (N - \sum_{i=1}^{j} n_i - 1) \cdot \ldots \cdot 3 \cdot 2 \cdot 1.$$

Oczywiście $j < k$, ostatecznie otrzymujemy

$$L_{c_k} = \left(\frac{1}{2}\right)^k \cdot P_N$$

Jak szybko policzymy, w grafie G jest dokładnie $L_h = \frac{P_N}{2}$ różnych[3] cykli Hamiltona rozpoczynających się w wierzchołku 0. Każdy dostrzega, że

$$L_{c_k} = \left(\frac{1}{2}\right)^k \cdot P_N < L_h = \frac{P_N}{2}.$$

Ten wynik zaskakuje każdego, kto pośpieszył się z odpowiedzią na poprzednie pytania. Zaskoczył również mnie.

[2] każdy graf ma $|V|$ wierzchołków, jednak ze względu na symbol wariacji bez powtórzeń, V użyjemy innego oznaczenia.

[3] Ponownie zakładamy tożsamość ciągów typu (1234) z (4321).

2.3.2 Przypadek ogólny mTSP

Jako przypadek ogólny uznajemy w tutaj taki, w którym liczba komiwojażerów nie zostaje narzucona z góry.

Twierdzenie 1.24 *W uogólnionym przypadku problemu wielu komiwojażerów, liczba możliwych rozwiązań problemu jest znacznie większa niż dla problemu komiwojażera w tym samym, pełnym grafie G. To znaczy, że liczba rozwiązań* $L_{mTSP} \geqslant \frac{(|V|-1)!}{2}$.

Dowód.
Należy zauważyć, że uougólniony przypadek problemu mTSP dopuszcza, aby trasę przejechał jeden komiwojażer. Może to zrobić na $\frac{(|V|-1)!}{2}$ sposobów. Oprócz tego, gdy $|V| > 2$ każde miasto może być odwiedzone przez jednego komiwojażera, co potwierdza prawdziwość nierówności z powyższego twierdzenia.

Wniosek 1.25 *Algorytm sprawdzający wszystkie możliwości rozwiązania uogólnionego problemu komiwojażera ma złożoność* $\Omega(n!)$.

Trudne jest znalezienie funkcji $f(|V|)$ opisującą liczbę możliwości, które trzeba sprawdzić w ogólnym przypadku zagadnienia. Z tego powodu wykonamy rachunki kombinatoryczne dla problemu, w którym liczba miast, które należy odwiedzić, jest znana i wynosi $|V|-1 = 6$. Wyniki przedstawiono w tabeli wraz z uzasadnieniem, gdzie k – liczba komiwojażerów:

k	Liczba możliwości	Uzasadnienie
1	$\frac{P_6}{2} = 360$	Dokładnie tak samo jak w TSP.
2	$\frac{V_6^5}{2} = 360$	Pierwszy komiwojażer odwiedza 5 miast, drugi 1.
	$\frac{V_6^4}{2} \cdot \frac{V_2^2}{2} = 180$	Pierwszy komiwojażer odwiedza 4 miasta, drugi 2.
	$\frac{1}{P_2} \cdot \frac{V_6^3}{2} \cdot \frac{P_3}{2} = 90$	Pierwszy komiwojażer odwiedza 3 miasta, drugi 3.
3	$\frac{V_6^4}{2} \cdot \frac{V_2^1}{2} = 180$	Pierwszy odwiedza 4 miasta, pozostali po jednym.
	$\frac{V_6^3}{2} \cdot \frac{V_3^2}{2} = 180$	Pierwszy odwiedza 3, drugi 2, trzeci 1.
	$\frac{1}{P_3} \cdot \frac{V_6^2}{2} \cdot \frac{V_4^2}{2} \cdot \frac{V_2^2}{2} = 30$	Każdy komiwojażer odwiedza po 2 miasta.
4	$\frac{V_6^3}{2} = 60$	Pierwszy odwiedza 3 miasta, pozostałe po jednym.
	$\frac{1}{P_2} \cdot \frac{V_6^2}{2} \cdot \frac{V_4^2}{2} = 45$	Dwóch komiwojażerów odwiedza po 2 miasta, dwóch po jednym.
5	$\frac{V_6^2}{2} = 15$	Jeden komiwojażer odwiedza dwa miasta, pozostali po jednym.
6	1	Każdy z komiwojażerów odwiedza jedno miasto.
	Suma = 1531	

Tablica 2.1: *Przedstawienie sposobu na znalezienie wszystkich możliwych rozwiązań uogólnionego problemu mTSP dla grafu pełnego, z 7 wierzchołkami.*

Rozdział 3

Problem marszrutyzacji

Problem marszrutyzacji jest typowym zagadnieniem transportowym. Przedsiębiorcy już dawno zdali sobie sprawę z faktu, że algorytmy i modele matematyczne są narzędziami, które warto wykorzystywać zarówno w dużych firmach, np. kurierskich, jak i u średnich i małych przedsiębiorców. Posługiwanie się komputerowym wspomaganiem decyzji dla tego problemu jest szczególnie wskazane dla wszystkich firm posiadających odbiorców, którzy nie zmieniają swojego położenia. W takich przypadkach łatwiejsze jest zbudowanie modelu dostaw, gdyż można korzystać z historii tras transportowych, używać ich w przyszłości, jak również efektywniej wykorzystać działanie sieci neuronowych.

3.1 Definicja VRP

Problem marszrutyzacji został po raz pierwszy sformułowany przez G.B. Dantziga i R.H. Ramsera w 1959 roku, w artykule *The Truck Dispatching Problem* opublikowanym w *Management Science*. Jest „spokrewniony" z mTSP, jednakże istnieje między tymi zagadnieniami różnica, wymieniona w punkcie *(3)* poniższej definicji.

Definicja 1.26 *Problemem marszrutyzacji jest modelowany za pomocą pełnego, ważonego grafu $G(V, E)$, dodatkowo zakłada się, że:*

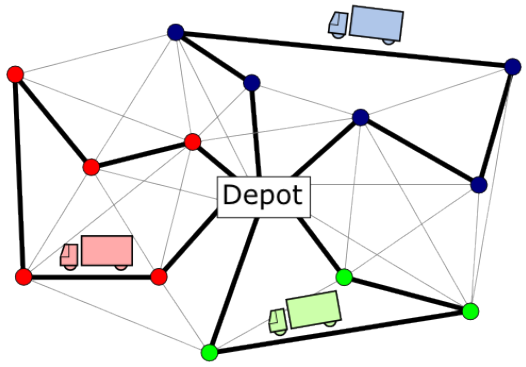
1. *$V = \{v_0, v_1, \ldots, v_n\}$, gdzie v_0 oznacza wierzchołek startu każdego pojazdu – magazyn,*
2. *C jest macierzą kosztów transportu c_{ij} między wierzchołkami v_i oraz v_j,*
3. *$D = (d_1, d_2, \ldots, d_n)$ jest wektorem zapotrzebowania na towar,*
4. *R_i jest trasą i-tego pojazdu.*
5. *m jest maksymalną liczbą pojazdów jaką dysponujemy, więc $i \in \{1, 2, \ldots, m\}$,*
6. *Każdy pojazd ma ograniczoną ładowność/pojemność q_i[1].*

Ponadto zakłada się, że każdy odbiorca ma być odwiedzony przez dokładnie jednego dostawcę towaru. Zadanie polega na tym, aby zminimalizować wartość funkcji:

$$S = \sum_{i=1}^{m} \sum_{v_p \in V} \sum_{v_q \in V} c_{pq} \cdot \mathbb{1}_{\{R_i\}}(v_p v_q)$$

[1]Niekiedy ten punkt jest pomijany w podstawowej definicji problemu.

Często wierzchołki grafu G, symbolizujące punkty odbioru towaru, są obdarzone oprócz wielkości zapotrzebowania d_i, również czasem obsługi t_i. Na ogół czas obsługi jest skorelowany z zapotrzebowaniem – im większe zapotrzebowanie danego odbiorcy tym dłuższy czas jego obsługi. Niekiedy pełna obsługa wszystkich sklepów jest niemożliwa, np. nie pozwala na to łączna pojemność/ładowność pojazdów, więc część odwiedzanych punktów odbioru otrzymuje tylko $\delta_i < d_i$ towaru. Poniżej przedstawiono rysunek obrazujący przykładowe rozwiązanie problemu[2]



Rysunek 3.1: *Przykładowe rozwiązanie problemu VRP.*

3.2 Odmiany VRP

W literaturze występują również rozwinięcia klasycznego problemu marszrutyzacji[3]. Należą do nich m.in.:

- Problemy uwzględniające niesymetryczność kosztów przewozu pomiędzy wierzchołkami.
  Takie rozwinięcie może być związane z ukształtowaniem terenu, bądź też charakterystyką dróg, np. w miastach mogą występować drogi jednokierunkowe.

- Problemy uwzględniające niehomogeniczność taboru.
  Bardzo często przedsiębiorstwo posiada pojazdy o różnej ładowności/pojemności.

- Problemy uwzględniające ograniczenie maksymalnej długości trasy.
  Kodeks pracy reguluje, jak długo podczas dnia, kierowca może prowadzić pojazd. Czasami skutkuje to oszacowaniem maksymalnego dystansu, jaki może pokonać, co ułatwia programowi wykluczenie tras nierealnych do zrealizowania przez jednego kierowcę.

- Problemy umożliwiające ustalenie baz (jednej lub kilku), w których pojazdy zaczynają i kończą podróż (Multiple Depot VRP).
  Rozwinięcie dotyczące w szczególności dużych przedsiębiorstw.

- Problemy umożliwiające dodanie baz pomocniczych (VRP with Satellite Facilities).

[2]Rysunek pochodzi ze strony http://ls11-www.cs.uni-dortmund.de/people/chimani/VehicleRouting/vr.png.
[3]Lista częściowo zaczerpnięta z http://pl.wikipedia.org/wiki/Problem_marszrutyzacji

- Problemy umożliwiające ustalenie częstotliwości odbioru/dostawy ładunku. Należy pamiętać, że w przedsiębiorstwach, pomimo stałej „mapy" punktów odbioru, bardzo często dzieje się tak, że nie wszystkie z nich są odwiedzane. Dlatego prawdziwym wyzwaniem dla każdego programu jest pomóc dyspozytorowi zadecydować, które punkty odbioru należy w danym dniu odwiedzić. Oczywiste jest, że nie warto jechać z bardzo małą ilością towaru do sklepu oddalonego o kilkaset kilometrów.

- Problemy umożliwiające uwzględnienie okien czasowych (VRP with Time Windows) odbioru/wysłania towaru. Oczywiste jest, że punkty odbioru towaru, np. sklepy są otwarte tylko w danych godzinach, więc po zamknięciu sklepu przyjęcie towaru nie jest możliwe.

- Problemy uwzględniające możliwość obsługi jednego klienta przez kilka pojazdów (Split Delivery VRP). Rzadko spotykane, najczęściej w czasie gdy powstaje nowy punkt odbioru, np salon meblowy i trzeba wypełnić go towarem.

- Problemy w których kosztowa funkcja celu zastąpiona została innymi parametrami (np. czas wykonania zleceń, długość tras, ilość przewiezionego ładunku).

- Problemy uwzględniające możliwości zwrotów i wysyłki towarów przez klientów (VRP with Backhauls oraz VRP with Pick-Up and Delivering – problem rozwózkowo-zwózkowy).

Przed przystąpieniem do projektowania oprogramowania dla danej firmy konieczne jest przeprowadzenie analizy przedwdrożeniowej. Należy zebrać maksymalną liczbę istotnych informacji dotyczących danego zagadnienia i przekształcić je na język formalny. Zazwyczaj stosuje się modele uproszczone, które jednak nie powinny odbiegać mocno od rzeczywistości. Zrozumiałe jest, że nie da się uwzględnić wszystkich czynników - np. różnych średnich prędkości przejazdu wśród wszystkich kierowców, wypadków drogowych, objazdów etc. Mimo to, komputery, dzięki swojej szybkości, mają możliwość przeszukania wielokrotnie większej przestrzeni rozwiązań danego zagadnienia niż człowiek.

3.3 Metody rozwiązywania VRP

W tym miejscu wymienimy algorytmy pozwalające znaleźć rozwiązania bliskie optymalnemu. Dwie techniki postępowania, zdaniem autora najciekwasze, krótko omówimy. Gdybyśmy chcieli omówić wszystkie, to rozdział byłby pewnie kilkakrotnie dłuższy, gdyż wymagałby wprowadzenia wielu definicji. Po za tym metody wymienione poniżej nie należą do prostych w wyjaśnieniu.

1. Algorytmy dokładne:

   - Metoda *Branch & Bound*.
   - Metoda *Branch & Cut*.

2. Heurystyki konstruujące trasy:

   - Algorytm *savings*, wymyślony przez Clark'a i Wright'a w 1964 roku.
   - Algorytm Bazowego dopasowania (*Matching based*).

- Heurystyki ulepszające (*Multi-route Improvement Heuristics*) zaproponowane przez:
  - Thompson'a i Psaraftis'a,
  - Van Breedam'a,
  - Kinderwater'a i Savelsbergh'a.

3. Meta-heurystyki:
   - Algorytmy mrówkowe.
   - Algorytmy genetyczne[4].
   - Symulowane wyżarzanie.
   - Przeszukiwanie Tabu.

Algorytmy genetyczne

Powstało wiele książek o algorytmach genetycznych[5], są one już dobrze poznane, a spektrum ich wykorzystania jest bardzo szerokie i nie ogranicza się tylko do problemów związanych z grafami. Postaram się opisać ich działanie jak najprościej, a czytelnik, jeśli taka będzie jego wola, aby uzupełnić swoją wiedzę, zajrzy do odpowiedniej literatury.

1. W dowolnym modelu pierwszym krokiem ($T = 0$) jest wylosowanie populacji złożonej z N osobników, będącej potencjalnymi rozwiązaniami danego problemu. Bardzo często rozwiązania te nie należą nawet do dość dobrych.

2. Następnie należy *zweryfikować przystosowanie* się osobników (jakość proponowanych rozwiązań) oraz wybrać te, które powinny przetrwać oraz krzyżować się.

3. Dzięki *krzyżowaniu* uzyskujemy rozwiązania podobne do najlepszych (uprzednio wybranych do krzyżowania), lecz nie takie same.

4. W trakcie krzyżowania może zajść *mutacja* danego osobnika. Mutacja to lokalna zmiana w genotypie danego osobnika.

5. Tym sposobem uzyskujemy nową populację ($T = 1$), wracamy do kroku drugiego. Cały algorytm powtarzamy tak długo, aż rozwiązania będą nas satysfakcjonowały.

Powyższy, jakże uproszczony, opis postępowania miał na celu pomóc czytelnikowi wyobrazić sobie ideę działania algorytmów genetycznych. Jak widać, działają one na zasadzie doboru naturalnego – przeżywają osobniki najlepiej przystosowane. Czytelnik musi wiedzieć, że aby algorytm dawał dobre rezultaty, musi być zaimplementowany z pewną starannością i dla wielu problemów, w tym również VRP implementacja jest niełatwa. Należy też wspomnieć, że algorytmy genetyczne, jak i inne heurystyki nie gwarantują znalezienia rozwiązania optymalnego. Zachęcam w tym miejscu do odwiedzenia stron internetowych [10] i [11], z których mozna pobrać dwa przykłady implementacji algorytmu genetycznego w programie Matlab. Celowo wspominamy o tym po raz drugi.

[4]Bardzo interesujące programy w języku Matlab znajdują się na stronach [10] i [11], dotyczą one rozwiązywania omawianych problemów za pomocą algorytmów genetycznych. Autor pracy zachwyca się nimi do teraz.

[5]Szczególnie polecam obszerną pozycję autorstwa Zbigniewa Michalewicza – *Algorytmy genetyczne + Struktury danych = Programy ewolucyjne.*

Algorytmy mrówkowe

Algorytmy mrówkowe jak również algorytmy genetyczne reprezentują klasę algorytmów probabilistycznych – nie gwarantujących otrzymania rozwiązania optymalnego. AM znakomicie radzą sobie w problemach grafowych, szczególnie w przypadku szukania najkrótszej ścieżki w grafie pomiędzy dwoma wierzchołkami. Na tym przykładzie postaramy się wyjaśnić ich działanie, gdyż jest to przypadek najprostszy i dokładnie przed takim samym problemem stoją mrówki zdobywając pożywienie, materiały do budowy mrowiska, etc.

Wyobraźmy sobie niezwykle duży graf $G(V, E)$, gęsty i ważony, ale nie pełny. W grafie G nie musi być spełniona nierówność trójkąta, co pozwala stosować algorytm dla znacznie większej ilości grafów. Zadanie polega na tym, żeby znaleźć najkrótszą ścieżkę łączącą dwa wierzchołki grafu G. Im bardziej „oddalone" od siebie, tym lepiej[6]. Algorytm wygląda następująco:

1. Dla $T = 0$ wszystkie N mrówek wyrusza z wierzchołka A wgłąb grafu, w poszukiwaniu „pożywienia", to znaczy punktu B.

2. Mrówki komunikują się za pomocą wydzielanych przez siebie feromonów. Z tego powodu na „skrzyżowaniach", czyli w wierzchołkach, w których mogą wybrać krawędź, którą chcą dalej podążać, jest większe prawdopodobieństwo, że wybiorą tą, po której szło do tej pory więcej mrówek, innymi słowy wybiorą tą krawędź, gdzie woń feromonów jest intensywniejsza. Oczywiście na początku przemieszczanie mrówek jest zupełnie chaotyczne, gdyż na ścieżkach nie czuć feromonów.

3. Mrówki podróżują tak długo, aż dojdą do wierzchołka B. Dojście do pożywienia skutkuje tym, że mrówki wydzielają intynesywniej feromony, więc wracając do wierzchołka A, ścieżki po których szły mrówki są silniej obdarzone zapachem.

Algorytm ten powtarzamy tak długo, aż większość mrówek będzie przebywała drogę z A do B tą samą ścieżką. Najistotniejszą cechą, która powoduje, że z czasem większość mrówek wybiera najkrótszą ścieżkę jest fakt, że feromony ulatniają się z czasem. To oznacza, że gdy mrówka pokonała drogę z A do B lub odwrotnie dłuższą drogą to zapach przez Nią pozostawiony na krawędziach tej ściezki jest mniej intensywny od tego, który pozostawiła mrówka idąca drogą krótszą. To powoduje, że w kolejnych przejściach mrówki z większym prawdopodobieństwem wybiorą ścieżki, na których czują intensywniejszy zapach feromonu, czyli defacto ścieżki krótsze (gdzie feromony nie zdążyły się ulotnić).

Wystarczy teraz, że ten opis zaimplementujemy i mamy sprawnie działający algorytm mrówkowy. Gdy zmodyfikujemy go, tak aby mrówka roznosiła w różne miejsca (wierzchołki grafu) np. piasek, który niesie na plecach (może unieść określoną ilość piasku) i „cieszyła się" wydzielając feromony, gdy już cały rozniesie, to otrzymamy opis pozwalający zaimplementować algorytm mrówkowy dla VRP.

[6]Dla małych grafów stosuje się algorytmy Dijkstry (Złożoność $O(|V|^2)$), Bellmana–Forda (Złożoność $O(|V| \cdot |E|)$)

Rozdział 4

Algorytm i analiza wyników

4.1 Wybór analizowanego wariantu

Zdecydowano przeanalizować jeden z wariantów zagadnienia. Dodatkowo do projektu został dołączony program napisany w języku Matlab dla tego wariantu. Program jest dostępny tylko dla promotora pracy i recenzenta z powodów wyjaśnionych we wstępie do pracy. Natomiast w tym rozdziale znajduje się schemat blokowy pokazujący w uproszczeniu kolejne kroki algorytmu.

Własności modelu.

Liczba miast, które należy odwiedzić:	$N \in [10, 12]$
Zapotrzebowanie i-tego miasta:	$d_i = 5 + 9 * rnd([0, 1]),\ d_i \in [5, 14]$
Łączne zapotrzebowanie:	$\|D\| \in [50, 168]$
Położenie miast:	losowe – $(x_i, y_i) \in ([0, 20], [0, 20])$
Położenie magazynu:	losowe $(x_{N+1}, y_{N+1}) \in ([0, 20], [0, 20])$
Reprezentacja grafu:	Macierz sąsiedztwa
Wagi krawędzi, metryka euklidesowa:	$w(v_i v_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$

Tablica 4.1: *Przedstawienie własności grafu modelującego „mapę" dystrybucji analizowanego problemu.*

Grafy można reprezentować za pomocą macierzy sąsiedztwa, listy sąsiedztwa, oraz macierzy incydencji. Dla naszego przypadku - grafu pełnego, najlepszą reprezentacją jest macierz sąsiedztwa, która w wierszu i, kolumnie j przechowuje $w(v_i v_j)$.

Ograniczenia.

Każdy komiwojażer odwiedzał $n_j \in \{2, 3, 4\}$ miast. Ładowność pojazdu wynosi $l \in [20, 29]$ – oznacza to, że pojazd przyjmował minimalnie 20 jednostek towaru, a maksymalnie 28.

4.2 Schemat algorytmu

Poniżej przedstawiamy schemat, który obrazuje działanie napisanego programu.

Rysunek 4.1: *Schemat przedstawiający działanie zaprojektowanego algorytmu.*

Pierwsze dwa kroki algorytmu nie nastręczają większych kłopotów. Deklarujemy kryteria, które musi spełniać każdy z możliwych cykli (C_k). Cykle modyfikujemy tak, aby miały najmniejszą łączną wagę. Jak nietrudno policzyć maksymalna liczba cykli, dla naszego przypadku, wynosi $C_N^2 + C_N^3 + C_N^4 = \{375, 550, 781\}$.

W kolejnym, trzecim kroku cykle należy połączyć. To znaczy trzeba wybrać te cykle, których elementy wypełnią nasz zbiór wierzchołków $\{1, 2, \ldots, N\}$. Oczywiście wszystkie cykle (jako zbiory) muszą być rozłączne. W tym momencie trafiamy na zagadnienie podobne do kolejnego problemu klasy NP – problemu minimalnego pokrycia zbioru.

4.2.1 Problem minimalnego pokrycia zbioru

Definicja 1.27 *Dany jest zbiór $U = \{1, 2, \ldots, m\}$ oraz S składający się z n jego podzbiorów. Celem jest znalezienie jak najmniejszej liczby elementów zbioru S, których suma daje zbiór U.*

Oczywiście powyższa definicja może być sformułowana w postaci problemu decyzyjnego.

Jak zostało udowodnione w 1972 problem minimalnego pokrycia zbioru należy do klasy problemów NP–zupełnych. Możemy powiedzieć, że problem VRP jest podwójnie trudny, gdyż, aby znaleźć optymalne rozwiązanie dla tego zagadnienia należy rozwiązywać dwa problemy klasy NP. W naszym problemie wymagamy, aby podzbiory (cykle) były rozłączne. Dodatkowo, my staramy się znaleźć wszystkie możliwe połączonia cykli, aby wybrać takie, które daje najmniejszy łączny koszt przejazdu.

4.3 Prezentacja i omówienie wyników.

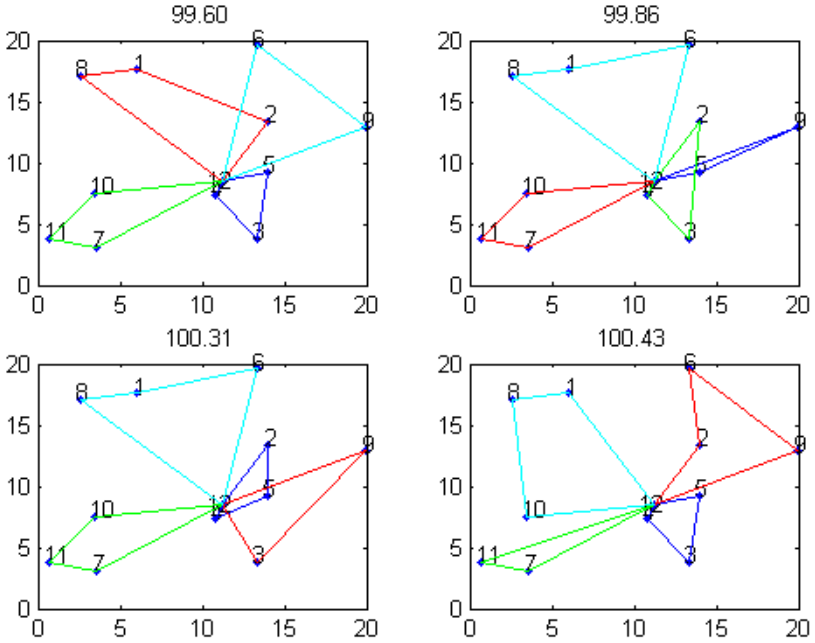
Omawiając wyniki skupimy się na pokazaniu jak złożonym problemem jest VRP nawet dla niewielkiej liczby miast. Zaprezentujemy również średnie długości czasu wykonywania się programu w zależności od liczby miast. Pokażemy, że nie zawsze możliwe jest znalezienie rozwiązania zagadnienia, szczególnie, gdy będziemy nieelastyczni. Na początku zaprezentujmy rozwiązanie przykładowego problemu:

Zapotrzebowanie:
1 – 9.3382
2 – 6.0855
3 – 10.3056
4 – 7.0357
5 – 8.4616
6 – 10.2469
7 – 7.2663
8 – 7.6140
9 – 10.5538
10 – 7.3875
11 – 12.4194.

Czas wykonywania:
25.3060 [s].

19⩽**Ładowność**⩽28

Liczba cykli/rozwiązań:
133/2265



Rysunek 4.2: *Przykładowe rozwiązanie problemu.*

Przyjrzyjmy się powyższym wizualizacjom czterech najlepszych tras. Widzimy, że pomimo znaczących różnic, jeśli chodzi o dobór miast, nie widać dużych różnic w całkowitym dystansie (tytuł każdego wykresu). Zauważmy też, że 3 najlepsze trasy mają jeden cykl dokładnie taki sam – $(v_{12}v_{10}v_{11}v_7v_{12})$, który trudno zastąpić lepszym, gdyż obejmuje „skupisko miast"[1]. Jeśli policzymy wykorzystanie pojemności/ładowności pojazdu, korzystając z listy zapotrzebowania miast, to zauważymy, że jest ono blisko 97%. Trudno zatem zastąpić ten fragment całej trasy lepszym.

[1]Miasta są relatywnie blisko siebie.

Czas wykonywania i liczba rozwiązań[2].

Do pracy została dołączona płyta CD z plikiem *dane.xls*, który przechowuje wyniki wielokrotnego wykonania się programu. Tą symulację wykonano, abyśmy mogli zaobserwować, jak ilość miast wpływa na czas szukania rozwiązań oraz ich ilość. Program wykonywał sie dla 10 miast 540 razy, dla 11 miast 135 razy, natomiast dla 12 miast 60 razy. Średnie wyniki najważniejszych wielkości podsumowano w tabeli poniżej:

N	Czas[s]	Liczba cykli	Liczba rozwiązań	Zapotrzebowanie	Liczba komiwojażerów	Brak rozwiązań
10	0,86	80,52	341,84	95,12	3,88	71
11	23,36	106,46	1021,73	105,06	4,24	9
12	586,59	147,37	4472,78	113,17	4,68	3

Tablica 4.2: *Uśrednione, najważniejsze własności dotyczące symulacji.*

Wszyscy widzimy, że czas szukania rozwiązania bardzo szybko rośnie wraz ze zwiększającą się ilością miast. Po części jest to na pewno wina samego algorytmu, jak i środowiska – Matlab, w którym został zaimplementowany. Programy pisane w Matlabie należą są interpretowane i choć Matlab doskonale radzi sobie z całkowaniem numerycznym, mnożeniem macierzy itp. to w przypadku czasu wykonywania operacji elementarnych – podstawienia, inkrementacji, porównywania, które w tym algorytmie mogą być wykonywane dziesiątki tysięcy razy, jest relatywnie (w porównaniu do C/C++, Javy) długi.

Warto zwrócić uwagę również na to, jak często algorytm nie znalazł rozwiązania. Wyjaśnimy na przykładzie dlaczego tak się dzieje. Załóżmy, że dla 10 miast, w skład wszystkich cykli spełniających kryteria algorytmu, wchodziłyby zawsze dokładnie 3 miasta. W tym wypadku, nie da się znaleźć takiego połączenia cykli, które pokryłoby zbiór wszystkich miast. Jak widać po wynikach naszej symulacji taka sytuacja dla 10 miast zdarzała się dość często.

Podczas badania czasu wykonywania zauważono również, że czas generowania cykli spełniających proponowane kryteria, jest relatywnie krótki w porównaniu do czasu potrzebnego na wygenerowanie wszystkich możliwych połaczeń cykli.

Krótkie podsumowanie

Powyższy projekt inżynierski miał na celu przybliżyć czytelnikowi tematykę związaną z problemem wielu komiwojażerów. Autor jest świadom, że temat został zaledwie pobieżnie omówiony, jednak nie sposób w tak krótkiej pracy inżynierskiej, omówić chociażby połowę problemów, metod rozwiązywania i zagadnień, które są z nim związane. Mam nadzieję, że jeśli ta praca zainteresuje czytelnika, to zasięgnie on do literatury oraz sam spróbuje zmierzyć się z tym zagadnieniem.

[2]Symulacja wykonywana była na komputerze z procesorem dwurdzeniowym o taktowaniu 1.86 Mhz, wyposażonym w system operacyjny Windows 7, dostępem do 2 GB pamięci RAM, DDR 2. Korzystano z programu Matlab R2009.b.

Bibliografia

[1] Andrzej Szepietowski, *Matematyka Dyskretna*, Wydawnictwo UG, Gdańsk, 2006.

[2] Kenneth A. Ross; Charles R. B. Wright, *Matematyka Dyskretna*, tł. z ang., PWN, Warszawa, 2006.

[3] Thomas Cormen, *Wprowadzenie do algorytmów*, tł. z ang., WNT, Warszawa, 2005.

[4] Christos H. Papadimitriou *Złożoność obliczeniowa*, WNT, Warszawa 2002.

[5] Witold Lipski, *Kombinatoryka dla programistów*, WNT, Warszawa, 1989.

[6] Jan Dolata, *Problem Komiwojażera*, praca magisterska PG, Gdańsk, 2009.

[7] Tolga Bektas, *The multiple traveling salesman problem: an overview of formulations and solution procedures*, Omega (34) str. 209-219, 2006, artykuł dostępny on-line.

[8] Sławomir T. Wierzchoń, *Problem komiwojazera* IPI PAN, UG, Gdańsk, 2008, *artykuł dostępny pod adresem* http://www.ipipan.waw.pl/ stw/mh/komiw.pdf

[9] Strona internetowa poświęcona VRP: http://neo.lcc.uma.es/radi-aeb/WebVRP/.

[10] Joseph Kirk, program w języku Matlab oparty na algorytmach genetycznych dotyczący TSP: http://www.mathworks.com/matlabcentral/fileexchange/13680

[11] Joseph Kirk, program w języku Matlab oparty na algorytmach genetycznych dotyczący mTSP: http://www.mathworks.com/matlabcentral/fileexchange/21299