

Badania Operacyjne 2

Implementacja Algorytmu Taboo Search dla problemu podróży po Europie

Autorzy:

Kaniowski Michał

Kapera Mikołaj

Spis treści

[1. Model zagadnienia 3](#_Toc124885263)

[1.1 Opis problemu 3](#_Toc124885264)

[1.2 Matematyczny opis zagadnień 3](#_Toc124885265)

[2. Algorytm 4](#_Toc124885266)

[2.1 Schemat działania algorytmu: 4](#_Toc124885267)

[2.2 Opis opracowanych elementów 4](#_Toc124885268)

[2.3 Parametry algorytmu 4](#_Toc124885269)

[3. Aplikacja 6](#_Toc124885270)

[3.1 Wymagania 6](#_Toc124885271)

[3.2 Format danych oraz wyników 6](#_Toc124885272)

[3.3 Działanie funkcji 6](#_Toc124885273)

[4. Testowanie aplikacji 8](#_Toc124885274)

[4.1 Przypadek podstawowy 8](#_Toc124885275)

[4.2 Zmiana 8](#_Toc124885276)

[5. Podział pracy 8](#_Toc124885277)

[6. Podsumowanie 9](#_Toc124885278)

[6.1 Wnioski 9](#_Toc124885279)

[6.2 Stwierdzone problemy 9](#_Toc124885280)

[6.3 Dalszy rozwój aplikacji 9](#_Toc124885281)

# 1. Model zagadnienia

## 1.1 Opis problemu

Optymalizacji będzie podległa podróż, różnym środkami transportu publicznego (lądowe, wodne, powietrzne) z punktu A do B, pomiędzy stolicami państw Europy. W funkcji kosztów uwzględniona będzie cena przemieszczania się konkretnym środkiem transportu, czas przemieszczania się oraz całkowity czas podróży (uwzględnia czas oczekiwania na transport w przypadku przesiadek). W modelu nie uwzględniliśmy możliwych opóźnień. Czas ten będzie możliwy do zmiany i może być inny dla każdego środku transportu.

Całkowity koszt oraz sposób transportu będzie zależny od wybranego dnia tygodnia oraz godziny (zakładając, że harmonogram wszystkich środków transportu powtarza się co tydzień).

Ograniczeniami modelu jest harmonogram, budżet oraz maksymalny czas podróży.

Danymi potrzebnymi do rozwiązania problemu są: harmonogram (zawiera również czas podróży) oraz cena biletów.

## 1.2 Matematyczny opis zagadnień

Struktury danych:

* – koszt podróży pojazdu danego typu m
* – opóźnienie, wynikające z rozkładu
* – położenie miasta
* – miasto początkowe
* – prędkość pojazdu danego typu m

Postać rozwiązania

* J – kolejność odwiedzonych miast
* – k-te miasto w kolejności

Funkcja celu

* Minimalizacja czasu podroży z uwzględnieniem kosztów transportu
* – czas podróży między dwoma miastami
* – koszt transportu danego typu między miastami
* – chęć odwiedzenia miasta
* – czas spędzony w mieście (minimalny czas spędzony)
* – opóźnienie wynikające z rozkładu, dopełnienie do pełnej godziny

Warunki ograniczające

* Maksymalny koszt podroży
* Maksymalny czas podróży

Zmienne pomocnicze

* Rozkład środków transportu zależny od dnia i godziny określający połączenia między miastami
* Średnia prędkość środków transportu

# 2. Algorytm

## 2.1 Schemat działania algorytmu:

1. Inicjalizacja listy tabu oraz wartości początkowej.
2. Wybór sąsiada aktualnego rozwiązania.
3. Aktualizacja listy ruchów tabu o nowy ruch.
4. Sprawdzenie jakości rozwiązania, nadpisanie aktualnej wartości w przypadku poprawy.
5. Usunięcie z listy tabu ruchów, którym skończył się czas przebywania na liście.
6. Sprawdzenie warunków stopu, w przypadku nie spełnienia powrót do kroku 2.

## 2.2 Opis opracowanych elementów

W naszym zagadnieniu wyróżniliśmy kilka typów struktur, na których operuje aplikacja: miasto, środek transportu, tabela odjazdów.

Opisując kolejno elementy typ miasto opisuje nam kierunek docelowy. Zawiera on informację tj. nazwa stolicy, jej położenie na mapie, tabela czasów odjazdów z miasta oraz czas spędzony na zwiedzaniu.

Środek transportu opisuje czym poruszamy się między kolejnymi stolicami Europy. Jego głównymi atrybutami są typ (np. samolot, bus), prędkość oraz koszt.

Tabela odjazdów, jest to struktura przechowująca informacje o możliwości przejazdów pomiędzy konkretnymi miastami dla określonego transportu, w zadanej godzinie.

Głównym zagadnieniem zaimplementowanego algorytmu jest tytułowa tabela taboo.

## 2.3 Parametry algorytmu

Jako dane wejściowe wyróżniamy (nazwa zmiennej - typ):

* Start point – punkt początkowy, miasto z którego rozpoczynamy podróż;
* End point – punkt końcowy, miasto w którym kończymy drogę;
* Start time –
* Max cost – maksymalny koszt, na jaki możemy sobie pozwolić kubując bilety przejazdu, przez całą drogę;
* Max taboo len – maksymalna długość naszej tablicy taboo;
* Aspiration – kryterium aspiracji;
* Max iter – maksymalna liczba iteracji;
* Bus vel – szybkość jazdy busem;
* Bus cost – koszt przejazdy busem;
* Train vel – szybkość jazdy pociągiem;
* Train cost – koszt przejazdy pociągiem;
* Plane vel – szybkość jazdy samolotem;
* Plane cost – koszt przejazdy samolotem.

Oprócz danych „parametrowych” można również w aplikacji zmienić tabelę odjazdów danych środków transportu. Struktura jest generowana w sposób pseudo losowy, dzięki czemu nie ograniczamy rozważanego problemu tylko dla określonych godzin. Dodatkowo każde miasto posiada atrybuty takie jak chęć odwiedzenia miasta oraz liczbę minimalnych godzin jaką trzeba spędzić w mieście.

# 3. Aplikacja

## 3.1 Wymagania

Wymagania algorytmu:

* Każde miasto musi mieć choć jedno połącznie z innym miastem w przeciągu 24h
* Prędkość pojazdów musi być większa od 0
* Chęć odwiedzenia miasta musi się mieścić w przedziale <1, 10>
* Kryterium aspiracji musi być większa od 0

## 3.2 Format danych oraz wyników

Dane wejściowe aplikacji, musza być podawane według następujących typów (nazwa\_zmiennej - typ):

* Start point – int
* End point – int
* Start time – int
* Max cost – float
* Max taboo len – int
* Aspiration – int
* Max iter – int
* Bus vel – float
* Bus cost – float
* Train vel – float
* Train cost – float
* Plane vel – float
* Plane cost – float

Z kolei wynik otrzymujemy w postaci listy krotek, gdzie indeksy krotek to odpowiednio:

1. typ transportu
2. numer miasta
3. ilość godzin, którą trzeba spędzić w mieście

## 3.3 Działanie funkcji

Generacja rozkładu transportów:

* 1. Wczytanie miast
  2. Pętla po godzinach i miastach
     + Przypisanie każdemu miastu lisy 3 elementowej, w której każdy element jest dobierany z rozkładu jednostajnego dyskretnego 0 lub 1
  3. Zapisanie rozkładu

Tworzenie rozwiązania początkowego:

Algorytm szuka pierwszego możliwego przejścia z miasta startowego do miasta o najniższym indeksie i na końcu pierwszego możliwego przejścia do miasta końcowego.

Generowanie nowego rozwiązania z najbliższego sąsiedztwa:

Algorytm wybiera 2 różne indeksy (poza miastem początkowym i końcowym) z rozkładu jednostajnego dyskretnego i wymienia miejscami kolejność miast z tymi indeksami.

Poprawianie nowego rozwiązania:

Algorytm nie uznaje rozwiązań niemożliwych więc:

* + 1. Usuwa czas oczekiwania z każdego miasta
    2. Szuka pierwszego możliwego przejścia z miasta do miasta
    3. Sprawdza koszt wszystkich typów transportu i wybiera najlepszy

Algorytm przeszukiwania taboo:

1. Inicjalizacja zmiennych oraz rozwiązania początkowych
2. Wybór nowego rozwiązania, dodania do listy taboo i usunięcie ostatniego rozwiązania z listy taboo
3. Sprawdzenie kosztów rozwiązania
4. Sprawdzenie kryterium aspiracji
5. Jeśli żaden z warunków stopu nie jest prawdą to przejścia do kroku 2

# 4. Testowanie aplikacji

W testowanych przypadkach wykorzystaliśmy problem podstawowy, tj. zdefiniowaliśmy losowo wybrane parametry, które będą modyfikowane dla poszczególnych zmian. Proces przeprowadzania każdej zmiany odbywał się poprzez ponowne ustawienie parametrów podstawowych – wybranych na początku, oraz modyfikację wartości parametru podanego w podtytule. W każdym przypadku zaprezentowane został rozwiązanie, wykres funkcji celu oraz opis modyfikacji parametru.

## 4.1 Przypadek podstawowy

## 4.2 Zmiana

# 5. Podział pracy

|  |  |  |
| --- | --- | --- |
| Etap | Osoba | |
| Kaniowski Michał | Kapera Mikołaj |
| Model zagadnienia |  |  |
| Algorytm – opracowanie |  |  |
| Implementacja Aplikacji |  |  |
| Testy |  |  |
| Dokumentacja |  |  |

# 6. Podsumowanie

## 6.1 Wnioski

## 6.2 Stwierdzone problemy

## 6.3 Dalszy rozwój aplikacji

Do zagadnienia modelu problemowego można dołączyć opis problematyki warunków pogodowych (w przypadku samolotów), dynamicznie zmieniających się tras kolejowych (np. w wyniku niebezpiecznych warunków pogodowych, katastrof – odcięcie w danym czasie konkretnego odcinka z tabeli odjazdów), a także ruch publiczny, tj. uzależnić czynnik przejazdu busem od liczby pseudolosowej (np. korki, opcjonalnie wydarzenia hamujące ruch). Zmianie mogą ulec nie tylko środki transportu, lecz także struktura typu miast. Można wycieczkę po Europie wzbogacić o współczynnik charakteryzujący atrakcyjność określonych stolic wedle danych statystycznych, tudzież na podstawie chęci użytkownika. Dodatkowo można dodać funkcjonalność, która determinuje czy dane stolice mogłyby być odwiedzone w określonym czasie – problem podróży mógłby być rozciągnięty w czasie, np. na okres 2 miesięcy, przez co w efekcie działań politycznych, militarnych, epidemiologicznych, niezależnych od człowieka, konkretne stolice zostałyby wykluczone w określonych tygodniach podróży.