STUDIA INFORMATICA 2000
Volume 21 Number 3 (41)

Marcin SZOŁTYSEK
Politechnika Śląska, Instytut Informatyki

PROBLEM WYZNACZANIA POŁĄCZEŃ AUTOBUSOWYCH

Streszczenie. Problem wyznaczania połączeń autobusowych jest przykładem złożonej optymalizacji kombinatorycznej. W celu zastosowania badanych algorytmów dla rzeczywistej sieci komunikacyjnej wraz z jej ograniczeniami stworzono komputerowy model realnej sieci komunikacyjnej. W dalszych pracach nad problemem połączeń autobusowych planuje się zbadanie algorytmów heurystycznych, takich jak przeszukiwanie tabu. Wymaga to istnienia dowolnego rozwiązania startowego oraz, w celu bezwzględnej oceny algorytmu, znajomości rozwiązania dokładnego, o ile rozmiar danych wejściowych pozwala na jego wyznaczenie. Z tych powodów prace nad problemem wyznaczania połączeń autobusowych rozpoczęto od stworzenia algorytmów realizujących te potrzeby. W niniejszej pracy przedstawiono wyniki działania algorytmu wyszukiwania wyczerpującego, algorytmu przeszukiwania wszerz oraz modyfikację algorytmu Dijkstry do wyznaczania połączeń autobusowych dla danych wejściowych opisujących rzeczywistą sieć komunikacyjną miast Gliwice i Zabrze.

SOLVING THE BUS ROUTING PROBLEM

Summary. The bus routing problem is a difficult combinatorial optimization problem. We have created a real bus network computer model in order to apply our algorithms to a real bus network. In the future research we plan to investigate the heuristic algorithms, including Tabu Search. The heuristics such as Tabu Search require a start solution, and in order to assess the heuristic algorithm efficiency we need to know the optimal solution, provided that the size of input data allows to determine it. For these reasons we have started our work with creating the algorithms for mentioned needs. In this paper the results of the exhaustive search algorithm, Breadth First Search (BFS) and modification of Dijkstra's shortest path algorithm for the bus routing problem, tested on the input data describing a real communication network of Gliwice and Zabrze cities, are presented.

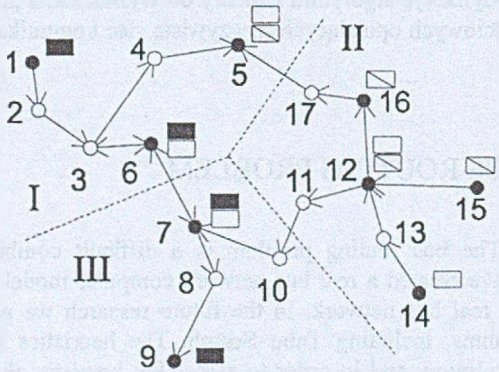
1. Wprowadzenie

Problem wyznaczania połączeń autobusowych jest przykładem złożonej optymalizacji kombinatorycznej. Rozważany problem został zdefiniowany przy uwzględnieniu własności i ograniczeń rzeczywistych sieci komunikacyjnych. Zbiory danych wejściowych zaczerpnięto z rzeczywistych sieci autobusowych. Dla zdefiniowanej sieci, rozkładu jazdy oraz stref cenowych dany jest przystanek początkowy i końcowy, pomiędzy którymi ma odbyć się podróż. Dana jest godzina rozpoczęcia podróży. Należy wyznaczyć zbiór linii autobusowych oraz miejsca przesiadek minimalizujących czas oraz koszt podróży.

Opracowanie algorytmu znajdującego wszystkie możliwe rozwiązania ma na celu stworzenie punktu odniesienia dla porównania wyników algorytmów heurystycznych rozwiązujących ten problem. Często optymalizacja czasu podróży pogarsza jej koszt i odwrotnie. Dlatego celem badanych algorytmów musi być rozwiązanie problemu dwukryterialnego.

2. Sformułowanie problemu wyznaczania połączeń autobusowych

Dana jest sieć autobusowa opisana za pomocą grafu (rys. 1).



Rys. 1. Przykładowa sieć połączeń autobusowych
Fig. 1. A sample network of the bus-system

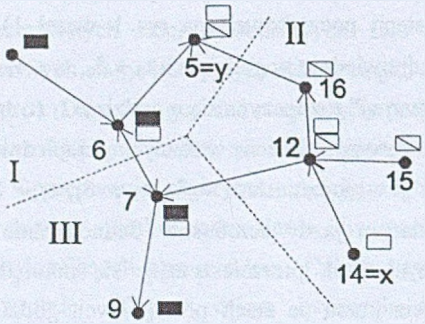
Wierzchołki grafu oznaczone czarnym kolorem reprezentują przystanki lub stacje końcowe autobusów. Wierzchołki grafu oznaczone białym kolorem to węzły pośrednie dodane do opisu sieci w celu umożliwienia wizualizacji rzeczywistej sieci. Informują one, którędy kursuje autobus pomiędzy przystankami. Krawędzie grafu reprezentują drogi

przejazdu pomiędzy przystankami. Droga może prowadzić tylko w jednym kierunku, stąd też opisywany graf jest grafem skierowanym. Autobusy poszczególnych linii (na rys. 1 pokazano trzy linie: „czarną", „białą" i „białą z kreską") kursują pomiędzy stacją początkową i końcową z określoną częstotliwością. Autobus zatrzymuje się na przystankach, przy których umieszczono symbol jego linii – prostokąt wypełniony kolorem linii. Przykładowo, autobus linii oznaczonej kolorem czarnym wyrusza ze stacji początkowej (na rys. 1 węzeł 1) o godzinie t_0 i przejeżdża przez przystanki 6, 7 i 9 odpowiednio w godzinach $t_0 + \delta_0$, $t_0 + \delta_1$, $t_0 + \delta_2$ ($\delta_0 < \delta_1 < \delta_2$). Następny kurs autobusu linii „czarnej" rozpoczyna się o godzinie t_1 ($t_1 = t_0 + \beta$, β jest określone częstotliwością kursowania i może zmieniać się w różnych porach dnia), tak więc autobus dociera do przystanków 6, 7 i 9 w godzinach $t_1 + \delta_0$, $t_1 + \delta_1$, $t_1 + \delta_2$. Wielkości δ_0, δ_1, δ_2 są dane i określone rozkładem jazdy autobusów. Jednocześnie z autobusem linii „czarnej" kursują autobusy innych linii, przemieszczając się pomiędzy swoimi stacjami końcowymi. Godziny rozpoczęcia kursu na stacji początkowej, godziny przybycia na poszczególne przystanki oraz trasa autobusu, tj. sekwencja przystanków i węzłów pośrednich, przez które autobus przejeżdża, są dane. W tym samym czasie w sieci może poruszać się większa liczba autobusów tej samej linii.

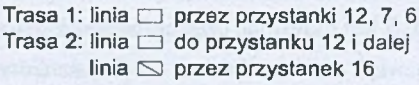
Dla zdefiniowanej sieci i rozkładu jazdy autobusów dany jest przystanek początkowy x oraz końcowy y ($x \neq y$), pomiędzy którymi chcemy odbyć podróż. Dana jest godzina t_x rozpoczęcia podróży na przystanku x (przybycie na przystanek x). Problem wyznaczania połączeń autobusowych polega na wyznaczeniu zbioru linii autobusowych, którymi można odbyć podróż, oraz zbioru przystanków (punkty przesiadkowe). Zbiory mają zostać wyznaczone w taki sposób, by całkowity koszt oraz czas podróży był minimalny.

Czas podróży wynika z przyjętej trasy, ewentualnych przesiadek oraz czasów oczekiwania na nich na połączenie, a także czasów przejazdu pomiędzy przystankami objętymi trasą. Natomiast koszt podróży wyznaczany jest następująco. Bilet na pojedynczy przejazd w ramach jednej strefy kosztuje 1,2 jednostki. Sieć autobusowa podzielona jest na strefy. Przykładowa sieć z rys. 1 podzielona jest na trzy strefy oznaczone I, II i III. W momencie przekraczania strefy koszt biletu wzrasta o 0,3 jednostki. Tak więc koszt przejazdu z przystanku x do y jest sumą kosztów podróży na poszczególnych odcinkach (jeśli występują przesiadki) z uwzględnieniem możliwych zmian stref na tych odcinkach. Dodatkowo niektóre linie autobusowe mogą być oznaczone jako pośpieszne. Przejazd tymi liniami odbywa się szybciej, co uwzględnione jest w rozkładzie jazdy (wielkości δ_0, δ_1, ..., δ_i są odpowiednio mniejsze). Aby wyznaczyć koszt podróży autobusem pośpiesznym, należy pomnożyć przez 2 koszt podróży autobusem zwykłym (dotyczy to także wzrostu kosztu przy przekraczaniu strefy).

Przykładowe dane wejściowe i wszystkie możliwe trasy dla sieci z rys. 1 pokazano na rys. 2 i 3. Przystanek początkowy, x, jest w węźle 14, a końcowy w węźle 5. Dla zwiększenia czytelności na rys. 2 pominięto węzły pośrednie. Dla tras z rys. 3 mając dane godzinę wyjazdu i rozkład jazdy należy obliczyć koszt i czas podróży.



Rys. 2. Przykładowe dane wejściowe
Fig. 2. Sample input data

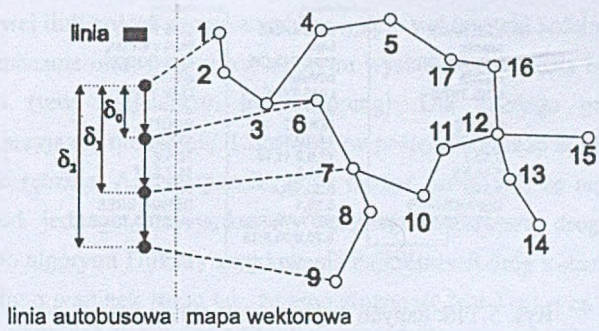


Rys. 3. Możliwe trasy
Fig. 3. Possible routes

3. Format danych wejściowych

3.1. Mapa wektorowa

W celu umożliwienia wizualizacji połączeń autobusowych zdecydowano się powiązać dane linii autobusowych z geograficzną mapą rejonu obsługiwanego przez sieć autobusową. Dlatego skorzystano ze stworzonej podczas badania innych metod optymalizacji, mapy wektorowej wraz z jej edytorem. Stąd też przystanek, poza informacją, że jest n-tym przystankiem m-tej linii, powiązany jest z mapą wektorową poprzez numer węzła, przy którym geograficznie występuje. Powstał więc system GIS (ang. Geographical Information System) składający się z dwóch warstw. Warstwa pierwsza to wektorowa mapa opisująca geograficzne położenie dróg oraz typ dróg i węzłów. Druga warstwa zawiera opis linii autobusowych.

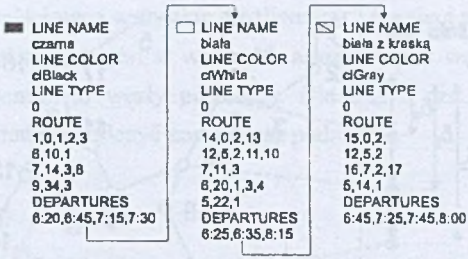


Rys. 4. Powiązanie linii autobusowej z mapą wektorową
Fig. 4. Bus line connected with a vector map

3.2. Format pliku sieci autobusowej

Do tworzenia sieci autobusowej zaimplementowano graficzny edytor linii autobusowych, jednak plik opisujący linie jest plikiem tekstowym, co pozwala na proste modyfikacje tego pliku. Plik zawiera opis dowolnej liczby linii autobusowych. Każda linia autobusowa jest określona przez parametry umieszczone w kolejnych sekcjach:

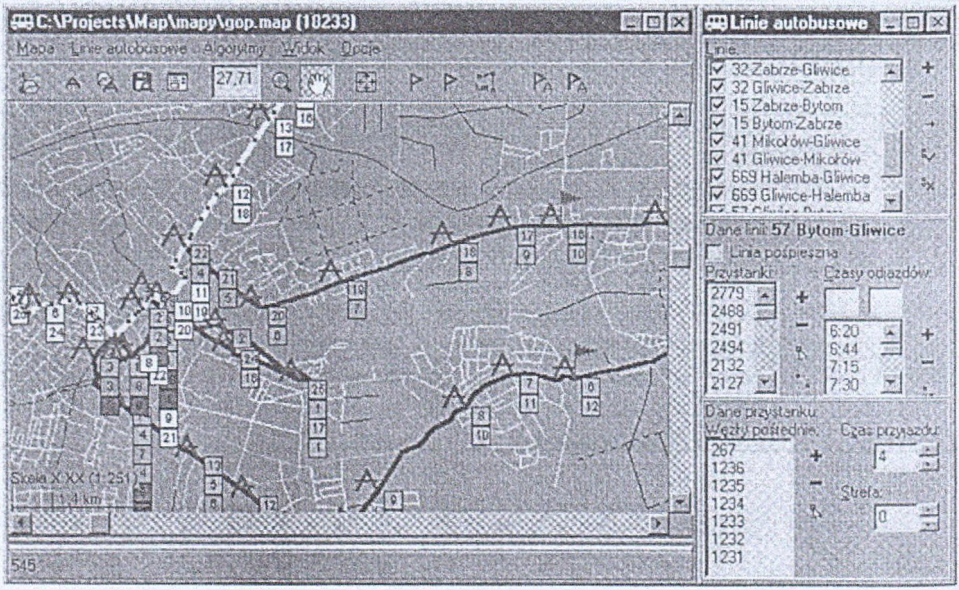
Nazwa sekcji	Opis
LINE NAME	nazwa linii
LINE COLOR	kolor linii do wizualizacji
LINE TYPE	typ linii, np. pośpieszna
ROUTE	dowolna liczba wierszy, każdy wiersz to opis kolejnego przystanku linii i jest zbudowany następująco: <nr węzła>,<czas dojazdu>,<strefa>,<węzły pośrednie>, gdzie <nr węzła> to numer węzła z mapy wektorowej, numer ten określa geograficzne położenie przystanku, <czas dojazdu> - wartość δ_i w minutach, <strefa> to numer strefy, w której przystanek się znajduje, <węzły pośrednie> to lista węzłów, które należy odwiedzić, by dostać się do kolejnego przystanku, lista jest używana wyłącznie w celu wizualizacji
DEPARTURES	czasy odjazdów ze stacji początkowych, tj. t_0, t_1, ..., oddzielonych średnikami, czasy są w formacie <g>:<m>, gdzie <g> i <m>, to godzina i minuty czasu wyjazdu



Rys. 5. Plik danych wejściowych dla sieci z rys. 1
Fig. 5. Input file for the network from fig. 1

4. Graficzny edytor linii autobusowych

Do tworzenia reprezentacji graficznej sieci autobusowej wykonano graficzny edytor linii autobusowych. Edytor umożliwia tworzenie nowych oraz edycję istniejących linii autobusowych. Edytor składa się z dwóch okien: okna wizualizacji i okna właściwości linii (rys. 6).



Rys. 6. Graficzny edytor linii autobusowych
Fig. 6. Bus-lines graphic editor

Tworzenie nowej linii polega na wskazaniu na mapie wektorowej kolejnych przystanków. Operacja ta jest znacznie ułatwiona przez algorytm wyszukiwania węzła będącego najbliżej wybranej pozycji (tzw. magnetyzm lub snapping). Dla każdego przystanku należy wprowadzić czas przyjazdu oraz strefę. Lista węzłów pośrednich może zostać wygenerowana automatycznie lub ręcznie. Automatycznie, gdyż można zauważyć, że najczęściej autobus pokonuje trasę od jednego przystanku do kolejnego najkrótszą drogą. W tym celu zaimplementowano algorytm Dijkstry znajdowania najkrótszych dróg z danego węzła [1]. W algorytmie zmieniono warunek stopu tak, że jego złożoność jest $T(n) = cn^2 = O(n^2)$, gdzie $c \ll 1$. Poniżej przedstawiono pseudokod algorytmu Dijkstry po modyfikacji. Algorytm wyszukuje najkrótszą drogę z węzła x do y. Po wykonaniu algorytmu minimalną długość trasy z x do y można odczytać z $D[y]$. Węzły wchodzące w skład najkrótszej trasy są w tablicy poprzedników *pred*.

1. inicjalizacja tablicy odległości, $C[1..n,1..n]$ pomiędzy wszystkimi wierzchołkami,
2. inicjalizacja tablicy odległości $D[1..n]$, wpisz odległości od x do wszystkich węzłów, jeśli węzły nie są połączone krawędzią, przyjmij wartość *MAXLONG*,
3. dodaj x do pustego zbioru węzłów sprawdzonych S; V oznacza zbiór wszystkich węzłów,
4. **for** $i := 1$ **to** n-1 **do** {n – liczba węzłów}
5. znajdź w zbiorze $V - S$ taki wierzchołek w, że $D[w]$ jest minimalne,
6. dołącz w do zbioru S,
7. **if** $w = y$ **then** zakończ algorytm,
8. **for** każdego wierzchołka v w zbiorze $V - S$ **do** {aktualizacja tablicy D},
9. **if** $D[v] > D[w]+C[w,v]$ **then**,
10. $D[v] := D[w]+C[w,v]$,
11. $pred[v] := w$ {uaktualnij tablicę poprzedników na trasie do y},
12. **end if**,
13. **endfor**.

Edytor został wyposażony w kilka narzędzi ułatwiających projektowanie sieci komunikacyjnej. Na przykład, narzędzie do obliczania odległości i czasów pokonywania zadanych tras, czy narzędzie do automatycznego generowania linii autobusowej „odwrotnej" do zadanej – mając daną linię w jednym kierunku, generowana jest linia w kierunku przeciwnym przeliczając odpowiednio wartości $\delta_0, \delta_1, ..., \delta_i$, oraz kolejność przystanków i węzłów pośrednich.

Edytor ma funkcje środowiska optymalizacyjnego, został bowiem zaprojektowany w sposób umożliwiający włączanie nowych algorytmów jako bibliotek DLL o określonym interfejsie. Badanie algorytmów polega na określeniu zestawu danych wejściowych (plik

mapy rastrowej, plik z opisem linii autobusowych, przystanek początkowy, końcowy oraz godzina wyjazdu) i dalej wybraniu algorytmu, za pomocą którego problem ma zostać rozwiązany. Jeżeli algorytm wymaga podania dodatkowych parametrów, to należy je także wprowadzić.

5. Wyszukiwanie wyczerpujące

W dalszych pracach nad problemem wyznaczania połączeń autobusowych mają zostać zbadane algorytmy heurystyczne (m.in. przeszukiwanie tabu). Wymaga to istnienia dowolnego rozwiązania startowego, oraz w celu bezwzględnej oceny algorytmu heurystycznego, znajomości rozwiązania dokładnego. Z tych powodów prace algorytmiczne nad problemem wyznaczania połączeń autobusowych rozpoczęto od stworzenia algorytmu realizującego te potrzeby.

Zaimplementowany algorytm wyszukiwania wyczerpującego reprezentuje podejście statyczne do problemu połączeń autobusowych. W pierwszym etapie wyznaczane są wszystkie możliwe trasy z przystanku x do y, pomijając rozkład jazdy (tzn. czasy wyjazdów i wartości δ_i). W drugim etapie, dla wszystkich znalezionych tras, obliczany jest koszt oraz czas podróży. Na tym etapie wyszukiwanie kończy się. Wynikiem wyszukiwania jest zbiór par czasów i kosztów. Na koniec pozostaje rozwiązanie problemu dwukryterialnego.

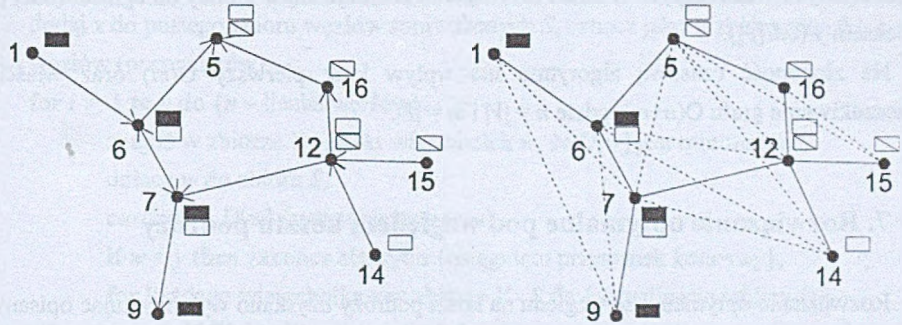
Pseudokod algorytmu wyszukiwania wyczerpującego:

1. *bieżący przystanek* := x,
2. dodaj *bieżący przystanek* do listy,
3. **if** *bieżący przystanek* = y **then** zakończ,
4. pobierz listę linii odwiedzających *bieżący przystanek*,
5. **for** i := 1 **to** *LiczbaPobranychLinii* **do**
6. *tymczasowy przystanek* := następny przystanek na linii i-tej,
7. **if** *tymczasowy przystanek* = brak przystanku **then**, idź do 10,
8. **if** *tymczasowy przystanek* jest już objęty trasą **then**, idź do 10,
9. wywołaj rekurencyjnie niniejszy algorytm od pkt. 2, za *bieżący przystanek* przyjmij *tymczasowy przystanek*,
10. **endfor**.

Złożoność czasowa algorytmu jest wykładnicza, tj. $O(k^n)$, gdzie k jest liczbą linii autobusowych, a n - liczbą przystanków.

6. Proste rozwiązanie

Posługując się algorytmem przeszukiwania grafu wszerz (ang. Breadth First Search), można uzyskać rozwiązanie problemu wyznaczania połączeń, które jest optymalne pod względem liczby przesiadek. Na podstawie grafu sieci połączeń budujemy pomocniczy multigraf skierowany G = (V, E). Graf ten powstaje w następujący sposób: dla każdej pary przystanków (węzłów) dodajemy tyle krawędzi, ile linii autobusowych łączy je bezpośrednio. Przykładowy graf i zbudowany dla niego graf pomocniczy pokazano na rys. 7.



Rys. 7. Przykładowy graf połączeń i zbudowany graf pomocniczy
Fig. 7. Sample connection graph and created additional graph

Dalsze kroki algorytmu polegają na przeszukaniu pomocniczego grafu wszerz, a ściślej znalezieniu w tym grafie ścieżki pomiędzy x i y takiej, że liczba węzłów rozwiązania jest minimalna. W ten sposób powstaje rozwiązanie problemu wyznaczania połączeń autobusowych charakteryzujące się minimalną liczbą przesiadek. Warto zauważyć, że w wielu przypadkach jest to rozwiązanie optymalne zarówno pod względem kosztu, jak i czasu podróży (zwykle wtedy, gdy można odbyć podróż z x do y, korzystając z jednej lub dwóch linii).

Pseudokod algorytmu:

1. zbuduj graf pomocniczy G = (V, E);
2. oznacz wszystkie elementy V jako nie odwiedzone;
3. dodaj *x* do kolejki;
4. oznacz *x* jako odwiedzony;
5. uaktualnij tablicę poprzedników *pred*[*x*] := -1;
6. uaktualnij tablicę odległości *odl*[*x*] := 0;
7. **while** *y* nie jest odwiedzony **do**
8. pobierz przystanek z kolejki i zapisz do *p*;

```
9.        for wszystkich (u) przystanków połączonych z p a nie odwiedzonych do
10.           dodaj przystanek do kolejki;
11.           oznacz przystanek jako odwiedzony;
12.           uaktualnij tablicę poprzedników pred[u] := p;
13.           uaktualnij tablicę odległości odl[u] := odl[p] + 1;
14.       endfor;
15.   endwhile.
```

Po wykonaniu algorytmu tablica *pred*, a dokładnie *y*, *pred*[*y*], *pred*[*pred*[*y*]], ..., *pred*[*pred*[...*pred*[*y*]...]], ..., *x*, zawiera węzły wchodzące w skład rozwiązania. Liczba linii autobusowych wchodzących w skład rozwiązania znajduje się w tablicy odległości (*odl*) pod indeksem *y* (*odl*[*y*]).

Na złożoność czasową algorytmu ma wpływ krok pierwszy O(*m*) oraz właściwe przeszukiwanie grafu O(*n*+*m*), gdzie $n = |V|$ i $m = |E|$.

7. Rozwiązanie optymalne pod względem kosztu podróży

Rozwiązanie optymalne ze względu na koszt podróży uzyskano wykorzystując opisany w poprzednim punkcie graf pomocniczy. Każdej krawędzi grafu pomocniczego, zbudowanego w oparciu o graf połączeń, zostaje nadana waga równa najmniejszemu kosztowi pokonania trasy reprezentowanej przez daną krawędź. Kolejnym krokiem jest na tak zdefiniowanym grafie wykonanie algorytmu znajdowania najkrótszych ścieżek (algorytm Dijkstry). Wynikiem wyszukiwania jest rozwiązanie optymalne pod względem kosztu podróży. O złożoności czasowej decyduje złożoność algorytmu Dijkstry.

8. Dynamiczne wyszukiwanie zachłanne - Algorytm Dijkstry

Rozwiązanie optymalne ze względu na czas podróży uzyskano zmodyfikowanym algorytmem Dijkstry. Zastosowano klasyczny algorytm wyszukiwania najkrótszych ścieżek. Modyfikacja polega na dynamicznym, tj. w trakcie pracy algorytmu, wyznaczaniu wag na krawędziach grafu. Waga krawędzi łączącej wierzchołki *a* i *b* grafu ma wartość funkcji $F(a, b, t)$, która równa się najwcześniejszemu czasowi, o jakim można się znaleźć na przystanku *b* wyruszając z przystanku *a* po lub o czasie *t*. W przypadku gdy niemożliwe jest bezpośrednie połączenie z *a* do *b*, funkcja zwraca wartość ∞. Na przykład, mając dany rozkład jazdy:

wyjazd z przystanku *a*	przyjazd na przystanek *b*
8:10	9:20
9:15	10:23

wartość funkcji $F(a, b, 7{:}15) = 9{:}20$, $F(a, b, 8{:}30) = 10{:}23$.

Pseudokod algorytmu:

1. przyjmij bieżący czas (*curtime*) jako czas przybycia na przystanek początkowy *x*,
2. inicjalizacja tablicy odległości $D[i]$, $i = 1..n$, wartościami funkcji $F(x,i,curtime)$,
3. dodaj *x* do pustego zbioru węzłów sprawdzonych *S*, oznacz jako *V* zbiór wszystkich węzłów (przystanków),
4. **for** $i := 1$ **to** *n* **do** {*n* – liczba węzłów},
5. znajdź w zbiorze $V - S$ taki wierzchołek *w*, że $D[w]$ jest minimalne,
6. dołącz *w* do zbioru *S*,
   curtime := $D[w]$ {ustaw bieżący czas},
7. **if** $w = y$ **then** zakończ algorytm {osiągnięto przystanek końcowy},
8. **for** każdego wierzchołka *v* w zbiorze $V - S$ **do** {aktualizacja tablicy *D*},
9. **if** $D[v] > F(w, v, curtime)$ **then**,
10. $D[v] := F(w, v, curtime)$,
11. $pred[v] := w$ {uaktualnij tablicę poprzedników na trasie do *y*},
12. **end if**,
13. **endfor**.

Po zakończeniu pracy algorytmu tablica *pred* zawiera listę węzłów rozwiązania optymalnego (dokładnie y, $pred[y]$, $pred[pred[y]]$, ..., $pred[pred[...pred[y]...]]$, ..., x). Złożoność czasowa funkcji *F* jest $O(k)$, *k* - liczba linii autobusowych. Złożoność czasowa całego algorytmu jest więc $O(n^2)$.

9. Wyniki badań eksperymentalnych

Badania przeprowadzono dla dużej sieci komunikacyjnej obejmującej dwa duże miasta i okoliczne miejscowości. Sieć składała się z 60 linii autobusowych, w tym 8 linii pospiesznych, obsługujących około 200 przystanków, rozmieszczonych w 7 strefach.

Algorytm wyszukiwania wyczerpującego przebadano dla zestawu danych wejściowych obejmującego wszystkie możliwe pary przystanków (łącznie n(n-1) par) początkowych i końcowych. W tabeli 1 podano: liczbę tras wygenerowanych tą metodą, koszt rozwiązania

najlepszego pod względem kosztu i czas rozwiązania najlepszego pod względem czasu dla przykładowych 5 par. Dodatkowo algorytmem tym sprawdzono liczbę rozwiązań spośród wszystkich możliwych par, które charakteryzują się tym, że ta sama trasa jest rozwiązaniem optymalnym zarówno ze względu na czas, jak i na koszt podróży. Tę część eksperymentu przeprowadzono dla 10 wybranych godzin rozpoczęcia podróży. Wyniki pięciu z nich przedstawiono w tabeli 2.

Tabela 1

Wyniki działania wyszukiwania wyczerpującego dla 5 przykładowych par przystanków

Przystanek początkowy	przystanek końcowy	liczba wszystkich możliwych rozwiązań	koszt najlepszego rozwiązania (pod względem kosztu) [jednostka]	czas najlepszego rozwiązania (pod względem czasu) [min]
Gliwice pl. Piastów	Zabrze ul. Goethego	2 670 301	1,5	29
Zabrze ul. Goethego	Gliwice pl. Piastów	2 403 882	1,5	44
Sośnica Centrum	Gliwice ul. Wrocławska	1 847 430	1,2	32
Zabrze Zaborze	Stare Gliwice	3 598 992	3,0	67
Gliwice ul. Pszczyńska	Zabrze Kończyce	1 545 095	3,0	48

Tabela 2

Wyniki dodatkowego eksperymentu

Godzina wyjazdu	Liczba rozwiązań optymalnych jednocześnie ze względu na koszt, jak i czas podróży [%]
8:00	12
10:00	9
13:00	10
15:00	11
17:00	10

Jako miarę jakości algorytmów: prostego i zachłannego przyjęto sumę czasów podróży pomiędzy wszystkimi parami przystanków danej sieci komunikacyjnej. W tabeli 3 podano uzyskane wartości miary oraz całkowity czas pracy algorytmu podczas eksperymentu

w minutach. Dla porównania efektywności czasowej tych dwóch algorytmów z efektywnością algorytmu wyszukiwania wyczerpującego, dodano do tabeli 3 czas wykonania tego algorytmu oraz dla formalności miarę jakości.

Tabela 3

Porównanie algorytmów: prostego, zachłannego i wyczerpującego

Algorytm	Miara jakości [s]	Czas wykonania [min]
Prosty	669 532	20
Zachłanny	638 562	4
Wyczerpujący	638 562	80 522 (56 dni)

10. Wnioski

Problem wyznaczania połączeń został przebadany dla sieci komunikacyjnej obejmującej miasta Gliwice i Zabrze (60 linii autobusowych i około 200 przystanków). Zbadano algorytm wyszukiwania wyczerpującego dla różnych danych wejściowych – modyfikowano: przystanek początkowy i końcowy oraz godzinę wyjazdu. Główny wniosek, jaki wyciągnięto po przeprowadzeniu badań, stwierdza, że znalezienie rozwiązania dokładnego dla małych systemów komunikacyjnych (50 linii autobusowych) jest możliwe za pomocą wyszukiwania wyczerpującego. Ze względu na dużą złożoność czasową algorytmu nie było możliwe wykonanie algorytmu dla większych sieci oraz większej liczby danych wejściowych, tj. przystanków początkowych i końcowych.

Algorytm wykorzystujący przeszukiwanie grafu wszerz generuje rozwiązanie z minimalną liczbą przesiadek. Rozwiązanie z minimalną liczbą przesiadek jest bliskie rozwiązaniu optymalnemu (ze względu na czas podróży) tylko dla podróży pomiędzy przystankami, między którymi odległość geograficzna jest niewielka. W większości przypadków rozwiązanie uzyskane tą metodą jest optymalne pod względem kosztu podróży. Algorytm ten ze względu na niską złożoność czasową może być wykorzystywany do generowania rozwiązania początkowego dla innych algorytmów, np. złożonych heurystyk.

Zdecydowanie najlepszym algorytmem do rozwiązywania problemu wyznaczania połączeń jest algorytm oparty na klasycznym algorytmie Dijkstry wyszukiwania najkrótszych ścieżek w grafie. Algorytm ten znajduje rozwiązanie optymalne ze względu na czas podróży. Dodatkowo jego złożoność jest niska (kwadratowa).

LITERATURA

1. Czech Z.: Analiza algorytmów. Materiały dydaktyczne, Gliwice 1997.
2. Aho A. V., Hopcroft J. E., Ullman J. D.: Projektowanie i analiza algorytmów komputerowych. WNT, Warszawa 1983.
3. Jungnickel D.: Graphs, Networks and Algorithms. Springer-Verlag, Berlin 1999.
4. Goczyła K., Cielątkowski J.: Journey planning in a public transportation network. Proc. 17th Int. Conf. on Information Technology Interfaces, Pula, s. 425-430, 1995.

Recenzent: Dr Urszula Boryczka

Wpłynęło do Redakcji 26 czerwca 2000 r.

Abstract

The bus routing problem (BRP) is a difficult combinatorial optimization problem. In this paper we present the BRP description, the bus network graphic editor, and the exhaustive search algorithm for the BPR.

The exhaustive search algorithm, Breadth First Search algorithm and modification of Dijkstra's shortest path algorithm were tested on the real data coming from a part of the Silesia bus network. These data include 60 bus-lines and 200 bus stops.

The tests show that it is possible to find the best solution using the exhaustive search algorithm for the data containing 50 bus-lines. The time complexity of the exhaustive search algorithm is exponential. Therefore for a larger number of bus-lines it is impossible to find the optimal solution using that algorithm.