STUDIA INFORMATICA 2002
Volume 23 Number 4 (51)

Jacek WIDUCH
Politechnika Śląska, Instytut Informatyki

WYZNACZANIE POŁĄCZEŃ W SIECIACH KOMUNIKACYJNYCH O ZMIENNYCH WAGACH

Streszczenie. Problem wyznaczania połączeń w sieciach komunikacyjnych jest przykładem optymalizacji wielokryterialnej. Rozwiązaniem problemu optymalizacji wielokryterialnej jest zbiór rozwiązań niezdominowanych. Wyznaczenie połączeń polega na rozwiązaniu dwukryterialnego problemu najkrótszych ścieżek w grafie ważonym o zmiennych wagach. W pracy przedstawiono algorytm umożliwiający wyznaczenie wszystkich połączeń należących do zbioru rozwiązań niezdominowanych. Zaprezentowano także przykładowe wyniki działania algorytmu. Oprócz tego przedstawiono dwa algorytmy rozwiązujące dwukryterialny problem najkrótszych ścieżek w grafie ważonym o stałych wagach.

SOLVING THE ROUTING PROBLEM IN COMMUNICATION NETWORKS WITH NON-CONSTANT WEIGHTS

Summary. The communication networks routing problem is an example of mulitcriteria optimization. The solution to a multicriteria optimization problem is the set of non-dominated solutions. Establishing routes consists in solving the bicriterion shortest path problem in the weighted graph with non-constant weights. In the paper an algorithm which determines all routes which belong to the set of non-dominated solutions is shown. The result of the tests are also presented. In addition, two algorithms for solving the bicriterion shortest path problem in a weighted graph with constant weights are presented.

1. Wstęp

Wiek XX charakteryzował się gwałtownym rozwojem przemysłu, co doprowadziło do powstania dużych aglomeracji miejskich. Potrzebą chwili stało się stworzenie odpowiedniej

sieci drogowej i komunikacyjnej. Pojawiły się pewne problemy. Jednym z nich jest dobór trasy przejazdu. O doborze trasy przejazdu decyduje wiele czynników, takich jak czas i koszt przejazdu, komfort podróży, atrakcyjność terenu, przez który mamy podróżować. O sposobie dojazdu w określone miejsce decydują też inne czynniki, na które nie mamy wpływu. Do takich czynników należy zaliczyć awaryjność środków transportu, roboty drogowe, czy też korki uliczne, które w wielkich aglomeracjach nie są rzadkością. Wybierając trasę przejazdu można uwzględnić tylko jeden z wymienionych czynników lub uwzględnić większą ich liczbę.

W wielu przypadkach przy projektowaniu i analizie systemów komunikacyjnych korzysta się z modeli sieciowych. Sieci te są zwykle tak duże, że ich analiza bez pomocy komputera jest uciążliwa, a niekiedy praktycznie niemożliwa. Z tego powodu tworzone są komputerowe systemy wspomagania decyzji dotyczących wyboru tras podróży. Jako przykład można podać system HAFAS, umożliwiający wyznaczenie połączeń kolejowych w Europie. System jest dostępny w postaci aplikacji lub w sieci Internet pod adresem http://bahn.hafas.de.

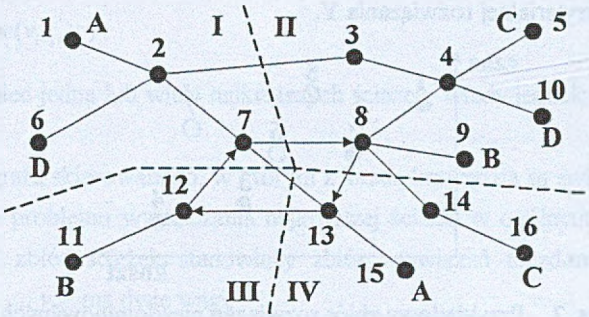
W niniejszej pracy został przedstawiony algorytm umożliwiający wyznaczenie połączeń w sieci komunikacyjnej ze względu na dwa kryteria jednocześnie: czas i koszt podróży. Problem, w którym o wyborze rozwiązania decyduje wiele kryteriów jednocześnie, należy do klasy problemów optymalizacji wielokryterialnej. Rozwiązaniem problemu optymalizacji wielokryterialnej jest w ogólnym przypadku zbiór rozwiązań niezdominowanych. Przedstawiony algorytm umożliwia wyznaczenie wszystkich połączeń komunikacyjnych należących do tego zbioru.

2. Sformułowanie problemu

Dana jest sieć komunikacyjna opisana za pomocą grafu skierowanego (rys. 1). W sieci kursują pojazdy czterech linii komunikacyjnych: A, B, C i D. Wierzchołki grafu o numerach 1-16 reprezentują przystanki lub stacje końcowe pojazdów komunikacyjnych. Stacje końcowe oznaczone są etykietami linii komunikacyjnych. Krawędzie grafu reprezentują trasy przejazdu pojazdów komunikacyjnych pomiędzy przystankami. Ponieważ pojazdy danej linii kursują w obydwu kierunkach, dlatego każdą krawędź nieskierowaną należy traktować jako parę krawędzi przeciwnie skierowanych. Trasy przejazdu pomiędzy stacjami końcowymi mogą się różnić. Pojazd linii B (rys. 1) przejeżdża z przystanku 9 do 11 inną trasą niż z przystanku 11 do 9, co zostało zaznaczone na rysunku za pomocą krawędzi skierowanych.

Pojazdy poszczególnych linii komunikacyjnych kursują z określoną częstotliwością określoną rozkładem jazdy. Częstotliwość kursowania pojazdów może się zmieniać w różnych porach dnia. Godziny rozpoczęcia kursu ze stacji początkowej, trasa przejazdu, tj. sekwencja

przystanków, przez które przejeżdża pojazd danej linii oraz czasy przejazdu pomiędzy poszczególnymi przystankami są dane. Czasy przejazdu pomiędzy przystankami są stałe i nie zmieniają się w różnych porach dnia. Zakładamy idealny model sieci komunikacyjnej, tzn. sieć komunikacyjną, w której pojazdy kursują zgodnie z rozkładem jazdy oraz pomijamy czas oczekiwania pojazdu na przystanku, tzn. godzina wyjazdu z przystanku jest równa godzinie przyjazdu na przystanek. Dodatkowo sieć komunikacyjna podzielona jest na strefy. Przykładowa sieć komunikacyjna przedstawiona na rys. 1 podzielona jest na 4 strefy oznaczone symbolami I, II, III i IV.



Rys. 1. Przykładowa sieć komunikacyjna
Fig. 1. A sample communication network

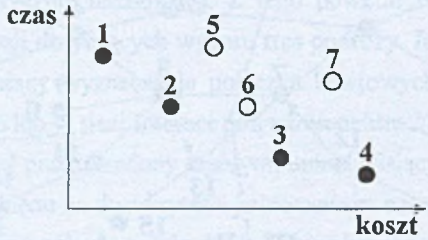
Dla zadanej sieci komunikacyjnej i rozkładu jazdy dany jest przystanek początkowy p_p oraz przystanek końcowy p_k, pomiędzy którymi chcemy odbyć podróż. Dana jest także godzina rozpoczęcia podróży na przystanku p_p. Należy wyznaczyć linie komunikacyjne oraz przystanki ewentualnych przesiadek tak, aby czas i koszt podróży pomiędzy przystankami p_p i p_k były minimalne.

Czas podróży wynika z przyjętej trasy przejazdu. Jest sumą czasów przejazdu pomiędzy przystankami zdefiniowanych rozkładem jazdy, ewentualnego czasu oczekiwania na przystanku początkowym oraz ewentualnych czasów oczekiwania na połączenie w przypadku przesiadki.

Koszt podróży wyznaczany jest w następujący sposób. Koszt przejazdu bez przesiadki w granicach jednej strefy wynosi 2.0 jednostki, w granicach dwóch stref 2.3 jednostki, a w granicach trzech lub więcej stref 2.6 jednostki. W przypadku wystąpienia przesiadek łączny koszt przejazdu z przystanku p_p do przystanku p_k jest sumą kosztów przejazdu na poszczególnych odcinkach. Niektóre linie komunikacyjne oznaczone są jako pośpieszne. Przejazd pojazdem takiej linii odbywa się szybciej, co jest uwzględnione w rozkładzie jazdy, natomiast koszt przejazdu jest dwa razy większy niż koszt przejazdu pojazdem linii zwykłej.

3. Optymalizacja wielokryterialna

Problem wyznaczania połączeń w sieciach komunikacyjnych jest przykładem optymalizacji wielokryterialnej, a dokładniej optymalizacji dwukryterialnej [1]. O wyborze rozwiązania zadania optymalizacji wielokryterialnej decyduje jednocześnie więcej niż jedno kryterium jakości. W omawianym problemie kryteriami jakości są czas i koszt podróży. Obydwa kryteria są kryteriami minimalizowanymi [1]. W przypadku kryteriów minimalizowanych rozwiązanie X jest lepsze od rozwiązania Y, jeżeli wartość funkcji kryterialnej rozwiązania X jest mniejsza od wartości funkcji kryterialnej rozwiązania Y.



Rys. 2. Przykładowy zbiór rozwiązań niezdominowanych
Fig. 2. A sample set of non-dominated solutions

Rozwiązaniem zadania optymalizacji wielokryterialnej nie jest jedno konkretne rozwiązanie, ale zbiór rozwiązań, zwany zbiorem rozwiązań niezdominowanych (zbiorem kompromisów, zbiorem rozwiązań Pareto-optymalnych [2]). Rozwiązanie X należy do zbioru rozwiązań niezdominowanych, jeżeli nie istnieje rozwiązanie Y, które je dominuje. Wówczas rozwiązanie X nazywamy rozwiązaniem niezdominowanym (Pareto-optymalnym), [3]. Załóżmy, że danych jest n minimalizowanych funkcji kryterialnych $f_1, f_2, \ldots, f_n$. Rozwiązanie Y dominuje rozwiązanie X, jeżeli: $\forall i \in \{1, \ldots, n\}$ $f_i(Y) \leq f_i(X)$ oraz $f_j(Y) < f_j(X)$ dla przynajmniej jednej wartości $j \in \{1, \ldots, n\}$. Przykładowy zbiór rozwiązań niezdominowanych został przedstawiony na rys. 2. Funkcjami kryterialnymi są czas i koszt podróży. Obydwa kryteria są kryteriami minimalizowanymi. Zbiór wszystkich możliwych rozwiązań składa się z siedmiu rozwiązań ponumerowanych od 1 do 7. Jednak do zbioru rozwiązań niezdominowanych należą tylko rozwiązania 1, 2, 3 i 4. Pozostałe rozwiązania są rozwiązaniami zdominowanymi. Rozwiązanie 5 jest zdominowane przez rozwiązania 1 i 2, rozwiązanie 6 jest zdominowane przez rozwiązanie 2, natomiast rozwiązania 2, 3 i 6 dominują rozwiązanie 7. Inne przykłady zadań optymalizacji wielokryterialnej oraz zbiorów rozwiązań niezdominowanych można znaleźć w pracy [1].

4. Dwukryterialny problem wyznaczania najkrótszej ścieżki w grafie ważonym o stałych wagach

Przyjmijmy, że dany jest skierowany graf ważony, tj. z każdą krawędzią grafu jest związana jedna, określona waga (np. odległość, czas lub koszt przejazdu). Wyznaczenie najkrótszej ścieżki pomiędzy dwoma wierzchołkami polega na znalezieniu ścieżki, której waga jest minimalna [5]. Wagą ścieżki $p = <v_0, v_1, ..., v_n>$ jest suma wag krawędzi ją tworzących [5] [9]:

$$w(p) = \sum_{i=1}^{n} w(v_{i-1}, v_i).$$

W grafie może istnieć jedna lub wiele najkrótszych ścieżek, wtedy jednak wszystkie one mają taką samą wagę.

W przypadku grafu skierowanego, w którym z każdą krawędzią są związane dwie wagi w_1 i w_2, rozwiązaniem problemu wyznaczania najkrótszej ścieżki w ogólnym przypadku nie jest jedna ścieżka, ale zbiór ścieżek stanowiący zbiór rozwiązań niezdominowanych. Każda ścieżka $p = <v_0, v_1, ..., v_n>$ ma dwie wagi:

$$w_1(p) = \sum_{i=1}^{n} w_1(v_{i-1}, v_i), \; w_2(p) = \sum_{i=1}^{n} w_2(v_{i-1}, v_i).$$

W odróżnieniu od poprzedniego przypadku ścieżki p_i, p_j $(i \neq j)$, będące rozwiązaniem problemu, mogą mieć różne obydwie wagi, tzn. $w_1(p_i) \neq w_1(p_j)$ i $w_2(p_i) \neq w_2(p_j)$ lub obydwie wagi takie same. Algorytmy rozwiązujące dwukryterialny problem najkrótszej ścieżki w grafie ważonym o stałych wagach zostały zaprezentowane w pracach [6] i [7].

W pracy [6] zaprezentowano algorytm rozwiązujący dwukryterialny problem najkrótszej ścieżki w skierowanym grafie ważonym. Każda krawędź grafu ma dwie wagi: czas i koszt przejazdu. Z każdym wierzchołkiem v_i jest związany zbiór rozwiązań niezdominowanych $D(v_i)$. Pojedyncze rozwiązanie jest parą etykiet określających czas i koszt dojazdu z wierzchołka początkowego v_p do wierzchołka v_i. Z tego powodu zbiór $D(v_i)$ będzie nazywany zbiorem par etykiet. Pary etykiet w zbiorze $D(v_i)$ są różne i są uporządkowane rosnąco względem kosztu przejazdu. Ponieważ $D(v_i)$ jest zbiorem rozwiązań niezdominowanych, dlatego uporządkowanie etykiet rosnąco względem kosztu przejazdu spowoduje automatyczne uporządkowane etykiet malejąco względem czasu przejazdu. Zanim zostanie zaprezentowany algorytm, zdefiniujmy dwie funkcje: *scal* i *waga*.

Funkcja *scal*(A, B) zwraca jako wynik zbiór będący sumą zbiorów $A = \{a_1, a_2, \ldots, a_p\}$ i $B = \{b_1, b_2, \ldots, b_q\}$, gdzie $a_i = (t_i^a, c_i^a)$, $b_j = (t_j^b, c_j^b)$ są parami etykiet oznaczającymi odpowiednio czas i koszt przejazdu. Zbiory A, B i wynikowy $S = scal(A, B)$ zawierają rozwiązania niezdominowane, zbiór S wynikowy nie zawiera takich samych rozwiązań.

Operacja scalania zbiorów jest wykonywana w czasie $O(|A| + |B|)$. Funkcję można przedstawić za pomocą pseudokodu [6]:

```
1:  i := 1; j := 1; scalone := false;
2:  S := ∅;                          {scalony zbiór par etykiet}
3:  while not scalone do
4:    if i > p then
      {nie ma już rozwiązań w A, dołącz na koniec zbioru S pozostałe rozwiązania z B}
5:      S := S + {b_j, ..., b_q}; scalone := true;
6:    else if j > q then
      {nie ma już rozwiązań w B, dołącz na koniec zbioru S pozostałe rozwiązania z A}
7:      S := S + {a_i, ..., a_p}; scalone := true;
8:    else if c_i^a < c_j^b then
9:      while t_i^a ≤ t_j^b do j := j + 1; end while   {pomiń rozw. zdominowane ze zbioru B}
10:     S := S + {a_i}; i := i + 1;                    {dołącz na koniec zbioru S rozwiązanie a_i}
11:   else if c_j^b < c_i^a then
12:     while t_j^b ≤ t_i^a do i := i + 1; end while   {pomiń rozw. zdominowane ze zbioru A}
13:     S := S + {b_j}; j := j + 1;                    {dołącz na koniec zbioru S rozwiązanie b_j}
14:   else if t_i^a < t_j^b then
15:     while t_i^a ≤ t_j^b do j := j + 1; end while   {pomiń rozw. zdominowane ze zbioru B}
16:     S := S + {a_i}; i := i + 1;                    {dołącz na koniec zbioru S rozwiązanie a_i}
17:   else
18:     while t_j^b ≤ t_i^a do i := i + 1; end while   {pomiń rozw. zdominowane ze zbioru A}
19:     S := S + {b_j}; j := j + 1;                    {dołącz na koniec zbioru S rozwiązanie b_j}
20:   end if
21: end while
22: scal(A, B) := S;
```

Funkcja $waga(v_i, v_j)$ zwraca jako wynik parę wartości (T_{ij}, C_{ij}) będących wagami krawędzi (czas i koszt przejazdu) prowadzącej z wierzchołka v_i do wierzchołka v_j.

Dodatkowo zdefiniujmy następującą operację:

$$A \oplus (T_{ij}, C_{ij}) = \{(t_1^a + T_{ij}, c_1^a + C_{ij}), (t_2^a + T_{ij}, c_2^a + C_{ij}), \ldots, (t_k^a + T_{ij}, c_k^a + C_{ij})\},$$

gdzie:

$$A = \{(t_1^a, c_1^a), (t_2^a, c_2^a), \ldots, (t_k^a, c_k^a)\}.$$

Zbiór A jest zbiorem rozwiązań niezdominowanych, a wartości (T_{ij}, C_{ij}) są stałe, dlatego zbiór będący wynikiem wykonania powyższej operacji jest także zbiorem rozwiązań

niezdominowanych. Pary etykiet w zbiorze wynikowym są uporządkowane rosnąco względem kosztu (malejąco względem czasu) przejazdu.

Algorytm, w dalszej części pracy nazywany algorytmem 1, można opisać za pomocą pseudokodu [6]:

1: **for** $v_i \in V$ **do**
2: $D(v_i) := \emptyset$;
3: **end for**
4: $D(v_p) := \{(0, 0)\}$;
5: *kolejka* $:= \{v_p\}$;
6: **while** *kolejka* $\neq \emptyset$ **do**
7: pobierz i usuń z *kolejki* wierzchołek v_i;
8: **for all** $v_j \in adj[v_i]$ **do**
9: $D_{nowy}(v_j) := scal(D(v_j), D(v_i) \oplus waga(v_i, v_j))$;
10: **if** $D_{nowy}(v_j) \neq D(v_j)$ **then**
11: $D(v_j) := D_{nowy}(v_j)$;
12: **if** $v_j \notin$ *kolejka* **then**
13: *kolejka* := *kolejka* + $\{v_j\}$;
14: **end if**
15: **end if**
16: **end for**
17: **end while**

W algorytmie zastosowano metodę przeszukiwania grafu wszerz [5]. Ideą algorytmu jest odwiedzanie wszystkich wierzchołków v_j, do których prowadzą krawędzie z danego wierzchołka v_i (krok 8) i wyznaczanie nowego zbioru etykiet dla wierzchołka v_j (krok 9). W przypadku uzyskania nowego, różnego od poprzedniego, zbioru par etykiet (krok 10) zbiór ten jest zapamiętywany (krok 11) a wierzchołek v_j, jeżeli nie znajduje się w kolejce (krok 12), to jest do niej wstawiany (krok 13). W algorytmie została zastosowana kolejka FIFO [5]. Działanie algorytmu kończy się w momencie, kiedy kolejka jest pusta (krok 6). W części inicjalizacyjnej algorytmu każdemu wierzchołkowi grafu przyporządkowany jest pusty zbiór par etykiet (kroki 1-3), natomiast z wierzchołkiem początkowym v_p (krok 4) zbiór jednoelementowy zawierający parę etykiet (0, 0).

W pracy [7] zostały zaprezentowane dwie modyfikacje algorytmu 1. Omówimy w dalszym ciągu tylko jedną z nich, gdyż była ona podstawą do opracowania algorytmu rozwiązującego

problem będący tematem niniejszej pracy. W dalszej części modyfikacja algorytmu 1 będzie nazywana algorytmem 2. Algorytm 2 jest opisany następującym pseudokodem:

1: $D(v_p) := \{(0, 0)\}$;
2: wyznacz rozwiązania o minimalnym koszcie z wierzchołka v_p do wszystkich pozostałych;
3: wyznacz rozwiązania o minimalnym czasie z wierzchołka v_p do wszystkich pozostałych;
4: **for** $v_i \in V$ **do**
5: jeżeli identyczne pary etykiet to usuń jedną z nich;
6: **end for**
7: *kolejka* := $\{v_1, v_2, ..., v_n\}$;
8: **while** *kolejka* $\neq \emptyset$ **do**
9: pobierz i usuń z *kolejka* wierzchołek v_i;
10: **for** $v_j \in adj[v_i]$ **do**
11: **if** zbiór $D(v_i) \oplus waga(v_i, v_j)$ niezdominowany **then**
12: $D_{nowy}(v_j) := scal(D(v_j), D(v_i) \oplus waga(v_i, v_j))$;
13: **if** $D_{nowy}(v_j) \neq D(v_j)$ **then**
14: $D(v_j) := D_{nowy}(v_j)$;
15: **if** $v_j \notin$ *kolejka* **then**
16: *kolejka* := *kolejka* + $\{v_j\}$;
17: **end if**
18: **end if**
19: **end if**
20: **end for**
21: **end while**

Modyfikacja algorytmu 1 polega na częściowym eliminowaniu rozwiązań zdominowanych w momencie tworzenia nowego zbioru etykiet (krok 9 algorytmu 1). Ponieważ pary etykiet w zbiorze D dla każdego wierzchołka są uporządkowane rosnąco względem kosztu (oraz malejąco względem czasu) przejazdu, umożliwia to sprawdzenie czy cały zbiór par etykiet przekazywany funkcji *scal* w drugim parametrze (krok 9 algorytmu 1) jest zdominowany przez wcześniej wyznaczone dla wierzchołka v_j rozwiązanie o minimalnym czasie lub koszcie przejazdu. Jeżeli zbiór jest zdominowany, to nie jest on tworzony, co oznacza pominięcie krawędzi prowadzącej z wierzchołka v_i do v_j.

Przyjmijmy, że dane są zbiory etykiet:

$$D(v_i) = \{(t_1^i, c_1^i), (t_2^i, c_2^i), ..., (t_k^i, c_k^i)\},$$
$$D(v_j) = \{(t_1^j, c_1^j), (t_2^j, c_2^j), ..., (t_q^j, c_q^j)\}.$$

Ponieważ pary etykiet w zbiorach są posortowane, więc spełnione są zależności:

$$t_1^i > t_2^i > \ldots > t_k^i,\ c_1^i < c_2^i < \ldots < c_k^i,$$

$$t_1^j > t_2^j > \ldots > t_q^j,\ c_1^j < c_2^j < \ldots < c_q^j.$$

Zbiór par etykiet $D(v_i) \oplus waga(v_i, v_j)$ jest zdominowany przez istniejące dla wierzchołka v_j rozwiązanie o minimalnym czasie przejazdu, jeżeli zachodzą zależności:

$$c_1^i + C_{ij} \geq c_q^j \text{ oraz } t_k^i + T_{ij} \geq t_q^j$$

lub jest zdominowany przez rozwiązanie o minimalnym czasie przejazdu w przypadku gdy:

$$t_k^i + T_{ij} \geq t_1^j \text{ oraz } c_1^i + C_{ij} \geq c_1^j.$$

Sprawdzenie, czy cały zbiór $D(v_i) \oplus waga(v_i, v_j)$ jest zdominowany (krok 11), nie wymaga tworzenia tego zbioru. Jest to możliwe dzięki uporządkowaniu par etykiet w zbiorach $D(v_i)$ i $D(v_j)$. Operacja ta odbywa się bez konieczności przeglądania całej zawartości wymienionych zbiorów [7].

Druga istotna modyfikacja algorytmu 1 dotyczy części inicjalizacyjnej. Za pomocą algorytmu Dijkstry [5] wyznaczane są rozwiązania o minimalnym koszcie (krok 2) i czasie (krok 3) przejazdu z wierzchołka początkowego v_p do wszystkich pozostałych wierzchołków grafu. Rozwiązania te są umieszczane w zbiorach etykiet dla każdego z wierzchołków. Rozwiązania o minimalnym koszcie i czasie przejazdu mogą być identyczne dla danego wierzchołka. W takim przypadku usuwane jest jedno z rozwiązań (kroki 4-6). Po wykonaniu części inicjalizacyjnej zbiór par etykiet każdego wierzchołka zawiera jedną lub dwie pary etykiet, dlatego do kolejki wstawiane są wszystkie wierzchołki grafu (krok 7).

5. Reprezentacja sieci komunikacyjnej

Sieć komunikacyjna jest reprezentowana przez skierowany graf ważony [4]. Wierzchołki grafu reprezentują przystanki, natomiast krawędzie grafu – trasy przejazdu pojazdów komunikacyjnych pomiędzy przystankami. Graf reprezentujący sieć zawiera krawędzie równoległe. Każda krawędź prowadząca z wierzchołka i do wierzchołka j ($i \neq j$) odpowiada danej linii komunikacyjnej, której pojazdy kursują z przystanku i do przystanku j, i ma dwie wagi: czas i koszt przejazdu z przystanku i do przystanku j.

Wagi krawędzi są wagami zmiennymi, tzn. ich wartości nie są stałe, tylko są ustalane w momencie wyznaczania połączenia. Rozpatrzmy połączenie z przystanku x do przystanku y (rys. 3). Załóżmy, że trasa połączenia przebiega przez przystanki i oraz j. Waga krawędzi określająca czas przejazdu z przystanku i do j (w_{ij}) ma następującą własność:

$$\tau_i + w_{ij} = \tau_j,$$

gdzie τ_i jest czasem przybycia na przystanek i, a τ_j jest czasem przybycia na przystanek j. Wartość wagi w_{ij} można określić dopiero wtedy, kiedy jest znany czas τ_i. Jest ona określona zależnością:

$$w_{ij} = r_{ij} + \Delta\tau_i,$$

gdzie r_{ij} jest czasem przejazdu z przystanku i do przystanku j, który jest zdefiniowany rozkładem jazdy, a $\Delta\tau_i$ jest czasem oczekiwania na przystanku i. W przypadku gdy na przystanku i nie dokonujemy przesiadki, czas oczekiwania $\Delta\tau_i$ jest równy zero i wartość wagi jest równa czasowi przejazdu z przystanku i do j. W przeciwnym przypadku, tzn. gdy na przystanku i konieczna jest przesiadka, czas oczekiwania $\Delta\tau_i$ może mieć wartość większą od zera, ponieważ czas wyjazdu może być różny od czasu przyjazdu na przystanek i.

$$x \longrightarrow \bullet \bullet \bullet \longrightarrow \underset{\tau_i}{i} \xrightarrow[w_{ij}]{r_{ij}} \underset{\tau_j}{j} \longrightarrow \bullet \bullet \bullet \longrightarrow y$$

Rys. 3. Fragment połączenia z przystanku x do przystanku y
Fig. 3. A part of route from stop x to stop y

O wartości wagi określającej czas przejazdu praktycznie decyduje to, czy następuje przesiadka na przystanku i. W przypadku wagi określającej koszt przejazdu, wpływ na jej wartość ma więcej czynników. W sytuacji kiedy na przystanku i nie dokonujemy przesiadki, koszt przejazdu z przystanku i do j może wynosić zero lub 0.3 jednostki. Koszt przejazdu wynosi zero w przypadku, gdy przystanki i oraz j znajdują się w tej samej strefie lub gdy znajdują się w różnych strefach, ale dojeżdżając do przystanku i od ostatniej przesiadki przekroczono co najmniej dwie strefy. Natomiast koszt wynosi 0.3 jednostki, jeżeli przystanki i oraz j znajdują się w różnych strefach i od ostatniej przesiadki przekroczono mniej niż dwie strefy. W przypadku gdy na przystanku i konieczna jest przesiadka, koszt przejazdu z przystanku i do j wynosi 2.0 lub 2.3 jednostki. Koszt przejazdu wynosi 2.0, jeżeli przystanki i oraz j znajdują się w tej samej strefie. W przeciwnym przypadku koszt przejazdu jest równy 2.3 jednostki.

Powyższa analiza kosztów przejazdu dotyczyła sytuacji, w której z przystanku i do przystanku j dojeżdżamy pojazdem linii zwykłej. W przypadku linii pośpiesznej powyższe koszty przejazdu należy pomnożyć przez dwa.

Wyznaczenie połączeń w sieci komunikacyjnej pomiędzy dwoma zadanymi przystankami sprowadza się do rozwiązania dwukryterialnego problemu wyznaczania najkrótszej ścieżki w grafie pomiędzy dwoma wierzchołkami (ang. *bicriterion shortest-path problem*).

6. Algorytm wyznaczania połączeń w sieci komunikacyjnej

Algorytm, w dalszej części pracy nazywany algorytmem 3, został opracowany na podstawie algorytmu 2 omówionego w punkcie 4. Algorytm 2 został opracowany dla grafu ważonego o stałych wagach, natomiast wagi grafu opisującego sieć komunikacyjną nie mają stałych wartości. Z tego powodu nie jest możliwe bezpośrednie zastosowanie algorytmu 2 do wyznaczenia połączeń w sieci komunikacyjnej i zachodzi konieczność wprowadzenia pewnych modyfikacji.

W algorytmie 2 wyznaczane są same etykiety oznaczające czas i koszt dojazdu, natomiast pomijana jest trasa dojazdu do danego wierzchołka. W przypadku problemu będącego tematem niniejszej pracy wyznaczenie samych etykiet nie jest wystarczające, gdyż konieczna jest także znajomość trasy przejazdu pomiędzy przystankiem początkowym i końcowym. Dlatego w algorytmie 3 pojedyncze rozwiązanie dla wierzchołka reprezentującego przystanek v_j zawiera:

- etykiety określające czas i koszt dojazdu,
- numer wierzchołka v_i, który poprzedza w trasie wierzchołek v_j,
- numer rozwiązania dla wierzchołka v_i, z którego utworzono aktualne rozwiązanie,
- numer linii komunikacyjnej, jaką dojeżdża się z wierzchołka v_i do v_j.

Pamiętanie wymienionych danych umożliwia odtworzenie trasy dojazdu z przystanku początkowego do przystanku v_j.

połączenie 1: $38 \xrightarrow{L\,23} 2 \xrightarrow{L\,1} 24$

połączenie 2: $38 \xrightarrow{L\,30} 17 \xrightarrow{L\,7} 24$

połączenie 3: $38 \xrightarrow{L\,23} 2 \xrightarrow{L\,1} 24 \xrightarrow{L\,25} 125 \xrightarrow{L\,23} 214$

połączenie 4: $38 \xrightarrow{L\,30} 17 \xrightarrow{L\,7} 24 \xrightarrow{L\,25} 125 \xrightarrow{L\,23} 214$

Rys. 4. Połączenia z przystanku 38 do 24 i z przystanku 38 do 214
Fig. 4. A routes from stop 38 to 24 and from stop 38 to 214

W algorytmie 2 dla każdego wierzchołka v_i pamiętany jest zbiór rozwiązań niezdominowanych $D(v_i)$. W przypadku wyznaczania połączeń komunikacyjnych pamiętanie dla każdego wierzchołka zbioru rozwiązań niezdominowanych jest niewystarczające i nie zapewnia wyznaczenia wszystkich rozwiązań niezdominowanych dla wierzchołka reprezentującego przystanek końcowy podróży. Niech będzie dany przystanek początkowy reprezentowany przez wierzchołek x, przystanek końcowy reprezentowany przez wierzchołek

y oraz zbiór rozwiązań dla wierzchołka i, będących połączeniami z przystanku x do przystanku i. Przyjmijmy, że trasa jednego z połączeń przebiega przez przystanek i (rys. 3).

Twierdzenie 1

Z rozwiązania zdominowanego dla wierzchołka i można uzyskać rozwiązanie niezdominowane dla wierzchołka y.

Dowód

Dowód twierdzenia zostanie przedstawiony na przykładzie. Niech będzie dany przystanek początkowy 38 i przystanek końcowy 214 (rys. 4). Zbiór rozwiązań niezdominowanych składa się z dwóch połączeń: połączenia 3 i 4. W obydwu połączeniach czas przejazdu jest równy 127 minut, natomiast koszt przejazdu wynosi 8.00 jednostek. Połączenia te nie są identyczne, ponieważ różnią się trasą przejazdu, a dokładniej trasą dojazdu z przystanku 38 do przystanku 24. Z przystanku 38 do 24 można dojechać na dwa sposoby: połączenia 1 i 2 na rys. 4. W obydwu połączeniach koszt przejazdu jest równy 4.00 jednostki. W połączeniu 1 czas przejazdu wynosi 76 minut, natomiast w połączeniu 2 jest on równy 77 minut. Zatem połączenie 2 jest zdominowane przez połączenie 1.

W połączeniach 3 i 4 trasa dojazdu z przystanku 24 do 214 jest taka sama oraz w obydwu przypadkach przejazd odbywa się pojazdami tej samej linii komunikacyjnej, dlatego czas i koszt przejazdu w obydwu przypadkach są identyczne. Ponieważ w obydwu połączeniach godzina wyjazdu z przystanku 24 jest taka sama, dlatego różnica jednej minuty w czasie dojazdu do przystanku 24 w połączeniach 1 i 2 jest kompensowana przez czas oczekiwania na przesiadkę na przystanku 24. Z tego powodu z połączenia 2, będącego rozwiązaniem zdominowanym, uzyskano rozwiązanie niezdominowane, tj. połączenie 4.

Z twierdzenia 1 wynika, że pamiętanie dla każdego wierzchołka zbioru rozwiązań niezdominowanych nie jest wystarczające. Dla każdego wierzchołka należy pamiętać więc wszystkie rozwiązania, które nie są zdominowane przez rozwiązania o minimalnym czasie i koszcie przejazdu dla wierzchołka końcowego. Wyjątkiem jest wierzchołek końcowy, dla którego pamiętane są tylko rozwiązania niezdominowane. Jeżeli dla dowolnego wierzchołka rozwiązanie jest zdominowane przez rozwiązanie o minimalnym czasie lub koszcie przejazdu dla wierzchołka końcowego, to takiego rozwiązania nie trzeba pamiętać i można je pominąć, ponieważ uzyska się z niego zdominowane rozwiązanie dla wierzchołka końcowego.

W kroku 12 algorytmu 2 tworzony jest nowy zbiór par etykiet. Powstaje on z dwóch zbiorów: $D(v_j)$ i $D(v_i) \oplus waga(v_i, v_j)$. Zbiory $D(v_j)$, $D(v_i)$ są zbiorami rozwiązań niezdominowanych. Ponieważ wagi krawędzi mają stałe wartości, dlatego zbiór $D(v_i) \oplus waga(v_i, v_j)$ jest także zbiorem rozwiązań niezdominowanych. W przypadku grafu reprezentującego sieć komunikacyjną wagi krawędzi nie mają stałych wartości, a zatem zbiór

$D(v_i) \oplus waga(v_i, v_j)$ może zawierać rozwiązania zdominowane. Z tego powodu zbiór ten nie jest tworzony. Próba utworzenia nowego zbioru rozwiązań dla wierzchołka v_j odbywa się w k iteracjach (kroki 6-16 algorytmu 3), gdzie k jest równe liczbie rozwiązań w zbiorze $D(v_i)$. W każdej iteracji na podstawie kolejnych rozwiązań ze zbioru $D(v_i)$ jest tworzone nowe rozwiązanie dla wierzchołka v_j (krok 7). Nowe rozwiązanie jest dołączane do zbioru $D(v_j)$, jeżeli nie należy ono do tego zbioru i nie jest ono zdominowane przez rozwiązanie o minimalnym koszcie lub czasie przejazdu dla wierzchołka v_{koniec} (krok 9). W przypadku gdy wierzchołkiem v_j jest wierzchołek końcowy v_{koniec}, przy dołączeniu rozwiązania do zbioru $D(v_j)$ następuje usunięcie rozwiązań zdominowanych (krok 13). W przeciwnym razie następuje ustawienie znacznika *nowe_rozw* (krok 11), który informuje, że do zbioru $D(v_j)$ zostało dołączone nowe rozwiązanie i po jego utworzeniu należy wstawić do kolejki wierzchołek v_j (kroki 17-19).

Algorytm wyznaczania połączeń w sieciach komunikacyjnych można przedstawić za pomocą następującego pseudokodu:

1: *MinCzas*(); *MinKoszt*();
2: **while** *kolejka* $\neq \emptyset$ **do**
3: pobierz i usuń z kolejki wierzchołek v_i;
4: **for** $v_j \in adj[v_i]$ **do**
5: *nowe_rozw* := **false**;
6: **for** $rozw_{vi} \in D(v_i)$ **do**
7: $rozw_{vj} := rozw_{vi} + waga(v_i, v_j)$;
8: **if** $rozw_{vj}$ niezdominowane i $rozw_{vj} \notin D(v_j)$ **then**
9: $D(v_j) := D(v_j) + \{rozw_{vj}\}$;
10: **if** $v_j \neq v_{koniec}$ **then**
11: *nowe_rozw* := **true**;
12: **else**
13: usuń rozwiązania zdominowane;
14: **end if**
15: **end if**
16: **end for**
17: **if** *nowe_rozw* i $v_j \notin$ *kolejka* **then**
18: *kolejka* := *kolejka* + $\{v_j\}$;
19: **end if**
20: **end for**
21: **end while**

W części inicjalizacyjnej algorytmu wyznaczane są za pomocą algorytmu Dijkstry rozwiązania o minimalnym czasie i koszcie przejazdu. Aby wyznaczyć rozwiązanie o minimalnym koszcie przejazdu konieczne jest dodanie dodatkowych krawędzi w grafie reprezentującym sieć komunikacyjną. Dla każdej pary wierzchołków oprócz krawędzi już istniejących dodawanych jest tyle krawędzi, ile linii komunikacyjnych łączy je bezpośrednio, tzn. bez konieczności przesiadki [8]. Algorytm Dijkstry umożliwia wyznaczenie rozwiązań o minimalnym czasie i koszcie przejazdu z wierzchołka początkowego do pozostałych wierzchołków w sieci. Jednak w omawianym przypadku poszukiwania są przerywane w momencie znalezienia rozwiązania dla wierzchołka końcowego. Wierzchołki, dla których zostały wyznaczone rozwiązania o minimalnym czasie lub koszcie przejazdu, są umieszczane w kolejce. Wyjątkiem jest wierzchołek końcowy, który nie jest umieszczany w kolejce. W trakcie wyznaczania rozwiązania o minimalnym koszcie przejazdu sprawdza się czy rozwiązania o minimalnym czasie i koszcie przejazdu nie są identyczne. W takim przypadku w zbiorze rozwiązań pamiętane jest tylko jedno z nich.

W sieci komunikacyjnej może istnieć wiele połączeń o takim samym koszcie i czasie przejazdu, lecz różniących się trasą przejazdu lub liniami komunikacyjnymi, jakimi przejeżdża się pomiędzy poszczególnymi przystankami. Takie połączenia nie są traktowane jako identyczne. Są one wyznaczane i umieszczane w zbiorze rozwiązań.

7. Złożoność czasowa algorytmu

Algorytm wyznaczania połączeń komunikacyjnych składa się z dwóch etapów. W części inicjalizacyjnej wyznaczane są połączenia o minimalnym czasie i koszcie przejazdu. W drugim etapie wyznaczane są pozostałe połączenia należące do zbioru rozwiązań niezdominowanych.

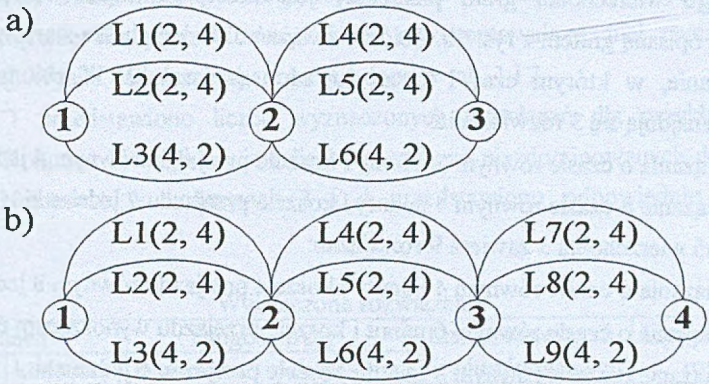
Do wyznaczenia połączeń o minimalnym czasie i koszcie przejazdu został zastosowany algorytm Dijkstry, którego złożoność czasowa jest $O(n^2)$, gdzie n jest równe liczbie wierzchołków w grafie reprezentującym sieć komunikacyjną [5].

Pozostałe połączenia komunikacyjne należące do zbioru rozwiązań niezdominowanych wyznaczane są za pomocą metody przeszukiwania grafu wszerz [5]. Etap ten ma złożoność wykładniczą względem liczby wierzchołków w grafie. Rozpatrzmy sieć komunikacyjną reprezentowaną przez graf z rys. 5a. Na krawędziach grafu zaznaczono, jaka linia (oznaczana przez Li) kursuje pomiędzy przystankami oraz czas (pierwsza wartość) i koszt przejazdu (druga wartość). Przyjmijmy, że pojazdy komunikacyjne kursują co 1 minutę. Niech przystankiem początkowym będzie przystanek 1, a przystankiem końcowym przystanek 3. Wszystkie możliwe połączenia pomiędzy tymi przystankami są rozwiązaniami

niezdominowanymi. Trasa każdego z połączeń jest identyczna. Połączenia różnią się liniami komunikacyjnymi, jakimi przejeżdża się pomiędzy przystankami. Istnieją następujące połączenia:

- 4 połączenia o czasie równym 4 minuty i koszcie przejazdu równym 8 jednostek,
- 4 połączenia o czasie równym 6 minut i koszcie przejazdu wynoszącym 6 jednostek,
- 1 połączenie o czasie równym 8 minut i koszcie przejazdu 4 jednostki.

Łącznie istnieje 9 połączeń będących rozwiązaniami niezdominowanymi. Liczbę wszystkich możliwych połączeń można opisać zależnością m^{n-1}, gdzie m jest równe stopniowi wyjściowemu wierzchołka.



Rys. 5. Przykładowe sieci komunikacyjne
Fig. 5. A sample communication networks

Rozpatrzmy teraz sieć składającą się z $n = 4$ przystanków, w której kursują pojazdy $k = 9$ linii komunikacyjnych (rys. 5b). Podobnie jak w poprzednim przypadku założymy, że pojazdy komunikacyjne kursują co 1 minutę. Przystankiem początkowym niech będzie przystanek nr 1, a przystankiem końcowym przystanek nr 4. Zbiór rozwiązań niezdominowanych składa się z:

- 8 połączeń o czasie równym 6 minut i koszcie przejazdu równym 12 jednostek,
- 12 połączeń o czasie równym 8 minut i koszcie przejazdu wynoszącym 10 jednostek,
- 6 połączeń o czasie równym 10 minut i koszcie przejazdu 8 jednostek,
- 1 połączenia o czasie równym 12 minut i koszcie przejazdu wynoszącym 6 jednostek.

Łącznie zbiór rozwiązań niezdominowanych zawiera $m^{n-1} = 27$ połączeń. Podobnie jak w poprzednim przypadku są to wszystkie możliwe połączenia pomiędzy przystankiem początkowym i końcowym.

Liczba wszystkich możliwych połączeń pomiędzy przystankiem początkowym i końcowym jest równa m^{n-1}. W omawianych dwóch przykładach $m < k$. Jednak maksymalna wartość stopnia wyjściowego wierzchołka (m) jest równa liczbie linii komunikacyjnych (k) w

sieci. Wobec tego złożoność czasowa wyznaczania pozostałych połączeń należących do zbioru rozwiązań niezdominowanych jest $O(k^n)$.

O złożoności czasowej algorytmu decyduje drugi etap wyznaczania połączeń, który ma złożoność wykładniczą. Zatem złożoność czasowa algorytmu wyznaczania połączeń w sieci komunikacyjnej jest $O(k^n)$.

8. Złożoność pamięciowa algorytmu

Dla każdego wierzchołka grafu pamiętany jest zbiór rozwiązań. Rozpatrzmy sieć komunikacyjną opisaną grafem z rys. 5b. Zbiór rozwiązań dla wierzchołka startowego zawiera jedno rozwiązanie, w którym czas i koszt dojazdu są równe 0. W zbiorze rozwiązań wierzchołka 2 znajdują się 3 rozwiązania:

- 2 rozwiązania o czasie równym 2 minuty i koszcie przejazdu równym 4 jednostki,
- 1 rozwiązanie o czasie równym 4 minuty i koszcie przejazdu 2 jednostki.

Zbiór rozwiązań wierzchołka 3 zawiera 9 rozwiązań:

- 4 rozwiązania o czasie równym 4 minuty i koszcie przejazdu równym 8 jednostek,
- 4 rozwiązania o czasie równym 6 minut i koszcie przejazdu wynoszącym 6 jednostek,
- 1 rozwiązanie o czasie równym 8 minut i koszcie przejazdu 4 jednostki.

Natomiast zbiór rozwiązań wierzchołka końcowego składa się z 27 rozwiązań niezdominowanych. Łącznie pamiętanych jest 1 + 3 + 9 + 27 = 40 rozwiązań. Wartości 1, 3, 9, 27 tworzą ciąg geometryczny o ilorazie $q = m = 3$ i pierwszym wyrazie $a_1 = 1$. Zatem liczba wszystkich pamiętanych rozwiązań jest równa sumie N początkowych wyrazów ciągu, która jest określona wzorem:

$$S_N = a_1 \frac{1-q^N}{1-q}.$$

W naszym przypadku N jest równe liczbie wierzchołków w grafie reprezentującym sieć komunikacyjną. Zatem złożoność pamięciowa algorytmu, podobnie jak złożoność czasowa, jest wykładnicza względem liczby wierzchołków w grafie.

Odległość pomiędzy przystankiem początkowym i końcowym ma wpływ na liczbę pamiętanych rozwiązań. Im przystanki początkowy i końcowy są bliżej siebie, tym minimalny czas i koszt dojazdu jest mniejszy. Dzięki temu znaczna liczba rozwiązań nie jest pamiętana, ponieważ są one zdominowane przez rozwiązanie o minimalnym czasie i koszcie dla przystanku końcowego. W przypadku znacznej odległości pomiędzy przystankiem początkowym i końcowym czas i koszt przejazdu są odpowiednio większe. Z tego powodu znacznie mniejsza liczba rozwiązań jest zdominowana przez rozwiązania o minimalnym czasie

i koszcie przejazdu, czyli muszą one być pamiętane. W takim przypadku może pojawić się problem z brakiem pamięci mogącej pomieścić tak znaczną liczbę rozwiązań. Obawy te zostały potwierdzone w trakcie badań eksperymentalnych.

9. Wyniki badań eksperymentalnych

Badania eksperymentalne algorytmu wyznaczania połączeń w sieci komunikacyjnej zostały przeprowadzone dla sieci komunikacyjnej składającej się z 1211 przystanków. Sieć jest podzielona na 26 stref, kursują w niej pojazdy 500 linii komunikacyjnych. Została ona stworzona wyłącznie dla potrzeb testów i nie reprezentowała żadnej rzeczywistej sieci komunikacyjnej. Wyniki badań zostały przedstawione w tab. 1 i 2.

W tabeli 1 przedstawiono liczbę wyznaczonych rozwiązań dla przykładowych par przystanków. Kolumna 5 tabeli zawiera liczbę rozwiązań niezdominowanych dla przystanku końcowego. Natomiast w kolumnach 3 i 4 przedstawiono odpowiednio minimalną i

Tabela 1

Wyznaczone rozwiązania

Przystanek		Długość trasy		Liczba rozwiązań	Liczba wszystkich
początkowy	końcowy	minimalna	maksymalna	niezdominowanych	rozwiązań
1211	672	21	25	3	184598
703	1095	23	28	6	799572
337	667	27	32	12	478851
484	1074	32	34	2	201564
1155	512	22	25	9	217485
750	1108	18	20	7	39296
648	675	36	36	1	108963
296	353	18	25	6	104567
602	738	20	24	7	498502
574	741	19	28	6	662223

maksymalną długość trasy połączenia należącego do zbioru rozwiązań niezdominowanych przystanku końcowego. Przez długość trasy należy rozumieć liczbę przystanków w trasie. W kolumnie 6 przedstawiono liczbę wszystkich rozwiązań, jakie były pamiętane podczas wyznaczania połączeń z przystanku początkowego do końcowego. Są to połączenia z przystanku początkowego do innych przystanków w sieci. Mimo że maksymalna długość trasy nie przekracza 40, liczba pamiętanych rozwiązań jest rzędu 10^5. W przypadku sieci komunikacyjnej składającej się z 1211 przystanków odległość nie przekraczającą 40 należy uznać za niewielką. Przeprowadzenie eksperymentu dla przystanków znajdujących się w

większej odległości okazało się niemożliwe ze względu na brak pamięci mogącej pomieścić wszystkie rozwiązania.

W tabeli 2 zaprezentowano porównanie czasu wyszukiwania połączeń za pomocą algorytmu zaprezentowanego w niniejszej pracy, oznaczonego jako algorytm BFS, oraz algorytmu przedstawionego w pracy [8], w którym do wyznaczania połączeń zastosowano metodę powrotów. Czas wyszukiwania za pomocą algorytmu BFS jest gorszy od czasu wyszukiwania za pomocą metody powrotów. Wpływ na to mają następujące czynniki. Rozwiązanie dla wierzchołka v_i zawiera m.in. numer wierzchołka, który go poprzedza w trasie, ale nie zawiera informacji o wszystkich wierzchołkach należących do trasy przejazdu z wierzchołka początkowego do wierzchołka v_i. Ponieważ rozwiązania nie mogą zawierać cykli,

Tabela 2

Czas wyszukiwania połączeń

Przystanek		Czas wyszukiwania	
początkowy	końcowy	Metoda powrotów	Algorytm BFS
1211	672	0:00.04	0:01.00
703	1095	0:00.05	0:13.08
337	667	0:00.03	0:02.59
484	1074	0:00.05	0:01.11
1155	512	0:00.06	0:01.28
750	1108	0:00.02	0:00.24
648	675	0:00.04	0:00.29
296	353	0:00.06	0:00.38
602	738	0:00.04	0:03.20
574	741	0:00.05	0:04.42

dlatego w czasie próby utworzenia nowego rozwiązania dla wierzchołka v_j należy sprawdzić, czy nie znajduje się on już w trasie. W tym celu należy każdorazowo odtworzyć trasę przejazdu z wierzchołka początkowego do wierzchołka v_i. W przypadku metody powrotów zastosowana została tablica o rozmiarze n, w której dla każdego wierzchołka pamiętane jest czy należy on już do trasy. Sprawdzenie, czy wierzchołek v_j należy do trasy, odbywa się w czasie stałym bez konieczności odtwarzania trasy przejazdu.

10. Podsumowanie

W niniejszej pracy został zaprezentowany algorytm wyznaczający połączenia w sieci komunikacyjnej ze względu na dwa kryteria jednocześnie: czas i koszt podróży. Wyznaczone połączenia należą do zbioru rozwiązań niezdominowanych. Połączenia są wyznaczane w dwóch etapach. W pierwszym etapie wyznaczane są połączenia o minimalnym czasie i koszcie

przejazdu. W drugim etapie za pomocą metody przeszukiwania grafu wszerz (ang. *breadth first search*) wyznaczane są pozostałe rozwiązania niezdominowane.

Rozwiązanie problemu wyznaczania połączeń w sieci komunikacyjnej sprowadza się do rozwiązania dwukryterialnego problemu najkrótszej ścieżki w grafie ważonym o wagach dynamicznych. Podstawą do opracowania algorytmu zaprezentowanego w niniejszej pracy był algorytm zaprezentowany w pracy [7]. Wymieniony algorytm rozwiązuje dwukryterialny problem najkrótszej ścieżki w grafie ważonym o stałych wagach. Algorytm ten został zaprezentowany w niniejszej pracy.

Przedstawiony algorytm wyznaczania połączeń w sieciach komunikacyjnych ma wykładniczą złożoność zarówno czasową, jak i pamięciową. Decydujący wpływ na czas wyznaczania połączeń ma drugi etap algorytmu o złożoności wykładniczej. Do wyznaczenia połączeń o minimalnym czasie i koszcie przejazdu zastosowano algorytm Dijkstry o złożoności kwadratowej. Czas wyznaczania połączeń o minimalnym czasie lub koszcie przejazdu nie przekracza 1 sekundy.

Ze względu na wykładniczą złożoność pamięciową algorytmu wyznaczenie połączeń w niektórych przypadkach może okazać się niemożliwe. Sytuacja taka ma miejsce w przypadku dużych sieci lub gdy przystanek początkowy i końcowy znajdują się w dużej odległości od siebie. W takim przypadku konieczne jest zapamiętanie znacznej liczby rozwiązań, które mogą nie zmieścić się w pamięci. Potwierdziły to wyniki badań eksperymentalnych.

Algorytm zaprezentowany w niniejszej pracy można stosować do wyznaczania połączeń w niewielkich sieciach komunikacyjnych lub w przypadku, gdy przystanek początkowy i końcowy znajdują się w bliskiej odległości od siebie.

LITERATURA

1. Peschel M., Riedel C.: Polioptymalizacja. Metody podejmowania decyzji kompromisowych w zagadnieniach inżynieryjno-technicznych. WNT, Warszawa 1979.
2. Stadler W.: A survey of Multicriteria Otimization or the Vector Maximum Problem. Part I: 1776-1960, Journal of Optimization Theory & Aplication, tom 29, nr 1, pp. 1-52, USA wrzesień 1979.
3. Górecki H.: Optymalizacja systemów dynamicznych. Wydawnictwo Naukowe PWN, Warszawa 1993.
4. Wilson R. J.: Wprowadzenie do teorii grafów. Wydawnictwo Naukowe PWN, Warszawa 1998.
5. Cormen T. H., Leiserson Ch. E., Rivest R. L.: Wprowadzenie do algorytmów. WNT, Warszawa 2000.

6. Brumbaugh-Smith J., Shier S.: An empirical investigation of some bicriterion shortest path algorithms. European Journal of Operational Research, tom 43, pp. 216-224, North-Holland 1989.
7. Skriver A.J.V., Andersen K.A.: A label correcting approach for solving bicriterion shirtest-path problems. Computers & Operations Research, tom 27, pp. 507-524, 2000.
8. Widuch J.: Problem wyznaczania połączeń w sieciach komunikacyjnych. Studia Informatica, vol. 22, nr 4(46), ss. 117-134, 2001.
9. Reingold E., Nievergelt J., Deo N.: Algorytmy kombinatoryczne. Państwowe Wydawnictwo Naukowe, Warszawa 1985.

Recenzent: Dr Urszula Boryczka

Wpłynęło do Redakcji 27 grudnia 2001 r.

Abstract

The communication networks routing problem (CNRP) is an example of multictriteria optimization. We have two optimization criteria: time and cost of travel. The paper presents the algorithm to find routes between two stops. The routes belong to the set of non-dominated solutions.

Establishing routes consists in solving the bicriterion shortest path problem in the weighted graph with non-constant weights. The algorithm solving CNRP is based on the algorithm presented in paper [7]. This algorithm solves the bicriterion shortest path problem in the weighted graph with constant weights.

The presented algorithm was tested on the communication network which consisted of 1211 stops and 500 communication lines. The communication network was created only for tests and it did not represent any real communication-system.

The time and memory complexities of the algorithm are exponential. Because the memory complexity is exponential it is not possible sometimes to use the presented algorithm to find the routes. If the start and end stops are far from each other it is necessary to store a considerable numer of solutions what is not always possible. The tests confirmed it. The presented algorithm can be used for finding the routes in small communication networks or for finding the routes between two stops which are not far from each other.