STUDIA INFORMATICA 2012
Volume 33 Number 4 (109)

Jacek WIDUCH, Daniel WICHER
Politechnika Śląska, Instytut Informatyki

ALGORYTM WYZNACZANIA POŁĄCZENIA KOMUNIKACYJNEGO O MINIMALNYM CZASIE PRZEJAZDU Z DODATKOWĄ MINIMALIZACJĄ KOSZTU PRZEJAZDU

Streszczenie. Wyznaczanie tras przejazdu jest jednym z badanych problemów transportowych. W niniejszej pracy przedstawiono problem wyznaczania połączeń w sieciach komunikacyjnych, którego celem jest wyznaczenie połączenia o minimalnym czasie przejazdu dla zadanej pary przystanków początkowego i końcowego oraz godziny rozpoczęcia podróży. Jeżeli istnieje więcej połączeń o minimalnym czasie przejazdu, to spośród nich należy wyznaczyć połączenie o minimalnym koszcie przejazdu. W pracy zaprezentowano algorytm, umożliwiający rozwiązanie badanego problemu. Zaprezentowano także przykładowe wyniki przeprowadzonych badań eksperymentalnych z użyciem rzeczywistej sieci komunikacyjnej.

Słowa kluczowe: problem transportowy, skierowany graf ważony, zmienne wagi, najkrótsza ścieżka, relacyjna baza danych, procedura składowana

A ALGORITHM FOR FINDING THE ROUTE MINIMIZING THE TIME OF TRAVEL AND MINIMIZING THE COST OF TRAVEL ADDITIONALLY

Summary. The routing problem is one of the studied transportation problems. In the paper the communication networks routing problem is presented, in which the goal is to find a route from the start stop to the final stop minimizing the time of travel, where the time of starting travel at the start stop is given. If there are more routes with minimal the time of travel, among them the route with minimal the cost of travel should be determined. In the paper the algorithm for solving the communication networks routing problem is presented. Apart from that a sample results of experimental tests using the real communication network are presented.

Keywords: transportation problem, directed weighted graph, variable weights, shortest path, relational database, stored procedure

1. Wstęp

Problem wyznaczania trasy przejazdu jest jednym z często badanych problemów optymalizacyjnych. Na dobór trasy przejazdu może mieć wpływ wiele czynników, np. czas i koszt przejazdu, komfort podróży, atrakcyjność terenu, przez który następuje przejazd. Przy wyborze trasy przejazdu może zostać uwzględniony tylko jeden z wymienionych czynników lub większa ich liczba. Są też czynniki, na które osoba dokonująca wyboru nie ma wpływu, a do nich należy zaliczyć awaryjność środków transportu, roboty drogowe czy też korki uliczne, które w wielkich aglomeracjach miejskich nie są rzadkością.

W pracach [35, 36, 37, 38, 39, 40] został omówiony problem wyznaczania połączeń w sieciach komunikacyjnych dla zadanej pary przystanków początkowego i końcowego oraz godziny rozpoczęcia podróży. Podczas wyznaczania trasy przejazdu z przystanku początkowego do przystanku końcowego dąży się do jednoczesnej minimalizacji czasu i kosztu przejazdu. Przedstawiony problem jest przykładem zadania optymalizacji wielokryterialnej [5, 26, 30, 32, 33]. W większości przypadków nie istnieje jedna trasa przejazdu, mająca jednocześnie minimalny czas i koszt przejazdu, stąd rozwiązaniem problemu jest zbiór rozwiązań, zwany zbiorem rozwiązań niezdominowanych (rozwiązań Pareto optymalnych [25]). Rozwiązanie problemu wyznaczania tras przejazdu pojazdów komunikacyjnych sprowadza się do rozwiązania dwukryterialnego problemu wyznaczania najkrótszej ścieżki w grafie o zmiennych wagach.

Dwukryterialny problem wyznaczania najkrótszej ścieżki w grafie jest poddawany badaniom i dostępnych jest wiele prac opisujących algorytmy jego rozwiązywania. Algorytmy te można podzielić na następujące grupy: algorytmy korygowania etykiet [3, 6, 8, 29], algorytmy nadawania etykiet [14, 22, 31], algorytmy dwufazowe [24], algorytmy działające na bazie wyznaczenia K najkrótszych ścieżek [4] oraz pozostałe [10, 19, 20, 21, 23, 27]. W każdym z wymienionych algorytmów zakłada się stałe wagi łuków grafu. Jak wykazano w pracy [37], różnice występujące między grafem ze stałymi wagami i grafem ze zmiennymi wagami uniemożliwiają zastosowanie wymienionych algorytmów do wyznaczenia połączeń komunikacyjnych.

W niniejszej pracy został omówiony wariant problemu wyznaczania połączeń w sieciach komunikacyjnych, którego celem jest wyznaczenie połączenia komunikacyjnego między zadaną parą przystanków. Podczas wyznaczania połączenia należy dążyć do minimalizacji czasu przejazdu, a w przypadku istnienia wielu połączeń o minimalnym czasie przejazdu należy zminimalizować koszt przejazdu. Tak więc wyznaczanie połączeń komunikacyjnych odbywa się na podstawie dwóch kryteriów, jednak nie są one uwzględniane jednocześnie, gdyż koszt przejazdu jest uwzględniany wyłącznie w przypadku istnienia wielu połączeń

o identycznym czasie przejazdu. Sieć komunikacyjna oraz rozkład jazdy są przechowywane w relacyjnej bazie danych, a w zaproponowanym algorytmie korzysta się z procedur składowanych, umożliwiających wykonywanie pewnych operacji bezpośrednio na bazie danych, co zmniejsza liczbę wykonywanych operacji w samym algorytmie.

2. Problem wyznaczania połączenia o minimalnym czasie przejazdu z ograniczeniem kosztu przejazdu

2.1. Sformułowanie problemu

Dana jest sieć komunikacyjna złożona z n przystanków s_1, …, s_n, w której kursują pojazdy M linii komunikacyjnych. Dla każdej linii komunikacyjnej jest zdefiniowana trasa przejazdu. Pojazd kursuje w określonym kierunku, tj. jeśli kursuje z przystanku s_i do przystanku s_j, nie oznacza to, że kursuje w przeciwnym kierunku. Pojazd danej linii może kursować w obydwu kierunkach, jednak trasy przejazdu pomiędzy przystankami mogą się różnić. Trasa przejazdu pojazdu jest złożona z sekwencji różnych przystanków, z wyjątkiem przystanków początkowego i końcowego linii, które mogą być takie same.

Pojazdy kursują z określoną częstotliwością. Niech trasa przejazdu pojazdu linii i–tej jest złożona z sekwencji przystanków: $\langle s_0, s_1, \ldots, s_{k-1}, s_k \rangle$, gdzie s_0 i s_k są odpowiednio przystankami początkowym i końcowym linii. Pojazd linii i–tej wyjeżdża z przystanku s_0 o godzinie T_0 i przejeżdża przez kolejne przystanki s_1, …, s_{k-1}, należące do trasy, wyjeżdżając z s_j (j = 1, …, k–1) o godzinie $T_0 + \delta_{j-1}$, i dojeżdża do przystanku s_k o godzinie $T_0 + \delta_{k-1}$. Następny kurs pojazd linii i–tej rozpoczyna na przystanku s_0 o godzinie T_1 ($T_1 = T_0 + \beta_0$), z przystanków s_j (j = 1, …, k–1) wyjeżdża o godzinie $T_0 + \delta_{j-1}$, a do przystanku s_k dojeżdża o godzinie $T_1 + \delta_{k-1}$. Pomija się czas wsiadania i wysiadania pasażerów, zakładając godzinę wyjazdu z przystanku równą godzinie przyjazdu. Ponadto, przyjmuje się idealny model sieci, w którym pojazdy kursują zgodnie z rozkładem jazdy. Pojazd linii i–tej wykonuje p kursów, wyjeżdżając z przystanku s_0 w godzinach T_0, …, T_{p-1} ($T_0 < \ldots < T_{p-1}$), gdzie $T_x = T_{x-1} + \beta_{x-1}$ (x = 1, …, p–1). Wartości T_0, β_0, …, β_{p-2}, δ_0, … , δ_{k-1} są zdefiniowane rozkładem jazdy, przy czym: $0 < \beta_0 < \ldots < \beta_{p-2}$; $0 < \delta_0 < \ldots < \delta_{k-1}$.

Dla danej sieci komunikacyjnej, opisu tras przejazdu pojazdów linii komunikacyjnych i rozkładu jazdy dane są przystanki początkowy s_s i końcowy s_e podróży. Dodatkowo, dane są: godzina rozpoczęcia podróży T_s na przystanku s_s oraz maksymalny czas oczekiwania T_w na przesiadkę. Celem jest wyznaczenie trasy połączenia komunikacyjnego z przystanku s_s do przystanku s_e o minimalnym czasie przejazdu, zakładając, że wyjazd z przystanku s_s nie może nastąpić wcześniej niż w godzinie rozpoczęcia podróży T_s, a czas oczekiwania na przesiadkę

nie jest większy niż T_w. Jeżeli istnieje wiele tras o minimalnym czasie przejazdu, to ograniczeniem jest, aby spośród nich wyznaczyć trasę o minimalnym koszcie przejazdu.

Czas przejazdu wynika z przyjętej trasy przejazdu i jest on równy sumie czasów przejazdu pomiędzy przystankami należącymi do trasy, ewentualnego czasu oczekiwania na przystanku początkowym s_s oraz ewentualnych czasów oczekiwania na połączenie w przypadku przesiadki.

Sieć jest podzielona na strefy, co wpływa na sposób wyznaczania kosztu przejazdu. Koszt przejazdu bez przesiadki w granicach jednej strefy wynosi c_1 ($c_1 > 0$) jednostek, w granicach dwóch stref c_2 ($c_2 > c_1$) jednostek, a w granicach trzech lub więcej stref c_3 ($c_3 > c_2$) jednostek. W przypadku wystąpienia przesiadek łączny koszt przejazdu z przystanku s_s do przystanku s_e jest równy sumie kosztów przejazdu pomiędzy przystankami, na których następuje przesiadka. Niektóre linie komunikacyjne są oznaczone jako pospieszne. Przejazd pojazdem takiej linii odbywa się szybciej, natomiast koszt przejazdu jest dwa razy większy od kosztu przejazdu pojazdem linii zwykłej.

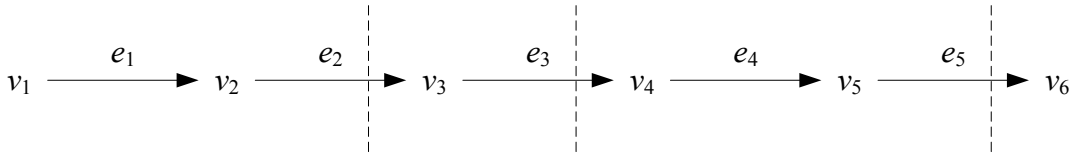
2.2. Analiza problemu

Do opisu sieci komunikacyjnej jest stosowany skierowany graf $G = (V, E)$ z wagami, który składa się z n wierzchołków[1] $v_1, \ldots, v_n$ [7, 17, 18, 28] reprezentujących przystanki pojazdów komunikacyjnych, gdzie v_i reprezentuje przystanek s_i ($i = 1, \ldots, n$). Łuki grafu reprezentują trasy przejazdu pojazdów, gdzie łuk $e_i = (v_j, v_k)$ ($e_i \in E$; $v_j, v_k \in V$) odpowiada jednej linii komunikacyjnej, której pojazdy kursują z v_j do v_k.

Każdy łuk $e_i = (v_j, v_k)$ ma 3 wagi: $l(e_i)$, $t(e_i)$ i $c(e_i)$. Waga $l(e_i)$ opisuje numer linii komunikacyjnej, której pojazd kursuje z v_j do v_k. Waga $t(e_i)$ jest równa różnicy między godziną przyjazdu T_k do v_k a godziną przyjazdu T_j do v_j, tj. $t(e_i) = T_k - T_j$. Waga ta jest wagą o zmiennych wartościach i przyjmuje ona wyłącznie wartości dodatnie. Jest więc ona równa: $t(e_i) = t_{jk} + \Delta t_j$, gdzie t_{jk} jest czasem przejazdu z v_j do v_k zdefiniowanym rozkładem jazdy, a Δt_j jest czasem ewentualnego oczekiwania na wyjazd w v_j. Czas t_{jk} jest stały, tak więc o wartości wagi $t(e_i)$ decyduje czas oczekiwania Δt_j, który jest czasem zmiennym.

Waga $c(e_i)$ przyjmuje wartości nieujemne, jest także wagą o zmiennych wartościach i jest równa $c(e_i) = c_k - c_j$, gdzie c_k jest kosztem przejazdu z v_s do v_k, a c_j jest kosztem przejazdu z v_s do v_j. Jej wartość zależy od tego, do jakiej strefy należą wierzchołki v_j i v_k oraz od tego, czy w v_j jest dokonywana przesiadka. Jeżeli w v_j jest dokonywana przesiadka oraz v_j i v_k znajdują się w tej samej strefie, to $c(e_i) = c_1$, a jeśli wierzchołki należą do różnych stref, to $c(e_i) = c_2$. W przypadku kiedy w v_j nie jest dokonywana przesiadka, a wierzchołki v_j i v_k

należą do tej samej strefy, to podczas przejazdu z v_j do v_k nie jest przekraczana strefa, tak więc koszt przejazdu nie wzrasta i $c(e_i) = 0$. Jeżeli v_j i v_k należą do różnych stref, to o wartości wagi $c(e_i)$ decyduje liczba stref, jakie przekroczono od ostatniej przesiadki przy dojeździe do v_j. Jeżeli nie przekroczono żadnej strefy, to $c(e_i) = c_2 - c_1$, w przypadku przekroczenia jednej strefy $c(e_i) = c_3 - c_2$, a przy przekroczeniu dwóch i więcej stref $c(e_i) = 0$.



Rys. 1. Fragment grafu reprezentującego trasę linii komunikacyjnej
Fig. 1. A part of graph representing a route of communication line

Zmienność wartości wagi $c(e_i)$ zostanie zilustrowana na przykładzie dojazdu do wierzchołka v_6 pojazdem linii komunikacyjnej, której trasa została przedstawiona na rys. 1 (liniami przerywanymi zaznaczono granice stref). Wartości wag $c(e_i)$ łuków e_1, …, e_5 zależą od wierzchołka początkowego v_s, z którego rozpoczynana jest podróż i zostały one przedstawione w tabeli 1.

Tabela 1

Wartości wag $c(e_i)$ łuków grafu z rys. 1 przy wyznaczaniu kosztu przejazdu do wierzchołka v_5 z różnych wierzchołków początkowych v_s

Wierzchołek v_s	$c(e_1)$	$c(e_2)$	$c(e_3)$	$c(e_4)$	$c(e_5)$
v_1	c_1	$c_2 - c_1$	$c_3 - c_2$	0	0
v_2	–	c_2	$c_3 - c_2$	0	0
v_3	–	–	c_2	0	$c_3 - c_2$
v_4	–	–	–	c_1	$c_2 - c_1$
v_5	–	–	–	–	c_2

Trasa połączenia komunikacyjnego z przystanku początkowego s_s do przystanku końcowego s_e jest reprezentowana przez ścieżkę $p_{se} = \langle v_0, (v_0, v_1), v_1, \ldots, v_{m-1}, (v_{m-1}, v_m), v_m \rangle$, z wierzchołka $v_0 = v_s$ do wierzchołka $v_m = v_e$ w grafie G reprezentującym sieć komunikacyjną, oraz przez godziny odjazdów T_i z wierzchołków v_i ($i = 0, \ldots, m-1$; $v_i \in p_{se}$). Długość ścieżki p_{se}, oznaczana przez $len(p_{se})$, jest równa liczbie łuków należących do ścieżki. Ścieżka p_{se} ma dwie wagi $T(p_{se})$ i $C(p_{se})$ równe sumie wag t i c łuków należących do ścieżki (1).

$$T(p_{se}) = \sum_{i=1}^{m} t(v_{i-1}, v_i), \; C(p_{se}) = \sum_{i=1}^{m} c(v_{i-1}, v_i) \tag{1}$$

Waga $T(p_{se})$ reprezentuje czas przejazdu z v_s do v_e, a waga $C(p_{se})$ koszt przejazdu z v_s do v_e. Zgodnie z założeniami przedstawionymi w punkcie 2.1, należy wyznaczyć w grafie G ścieżkę p_{se} minimalizując wartość wagi $T(p_{se})$. W przypadku istnienia wielu ścieżek

[1] W dalszej części pracy przez użycie zwrotu wierzchołek, w odniesieniu do grafu reprezentującego sieć komunikacyjną, będzie rozumiane odwołanie do przystanku reprezentowanego przez ten wierzchołek.

o minimalnej wartości wagi $T(p_{se})$, należy spośród nich wyznaczyć tę, która ma minimalną wartość $C(p_{se})$.

Problem wyznaczania ścieżki o minimalnej wadze jest jednym z często badanych problemów teorii grafów. W literaturze przedstawionych zostało wiele algorytmów umożliwiających jego rozwiązanie, a do najbardziej znanych należą algorytmy Dijkstry [11], Floyda–Warshalla [13, 34], Bellmana–Forda [2, 12] czy też Johnsona [16]. We wszystkich wymienionych algorytmach zakłada się stałe wagi łuków grafu. Ponadto, z powodu braku możliwości wyznaczenia macierzy D_0 nie jest możliwe wyznaczenie ścieżki o minimalnym czasie przejazdu za pomocą algorytmu Floyda–Warshalla [37]. W algorytmie tym w celu wyznaczenia i–tego wiersza macierzy D_0 konieczna jest znajomość godziny przyjazdu do wierzchołka i, a ta jest znana dopiero w trakcie wyznaczania ścieżki.

Jak wykazano w pracy [37], istnieje możliwość zastosowania pozostałych algorytmów do wyznaczenia ścieżki o minimalnym czasie przejazdu, jednak nie ma gwarancji wyznaczenia spośród wszystkich ścieżek o minimalnym czasie tej, która ma minimalny koszt przejazdu. Fakt ten wynika z właściwości ścieżki, która jest wyznaczana w grafie o stałych wagach. Niech, przykładowo, będą dane: ścieżka p_{se} z v_s do v_e o minimalnej wadze oraz wierzchołek v_i należący do tej ścieżki. W grafie o stałych wagach ścieżka p_{se} składa się z dwóch ścieżek o minimalnych wagach: ścieżki z v_s do v_i oraz ścieżki z v_i do v_e. Właściwość ta nie obowiązuje w grafie o zmiennych wagach. W takim przypadku ścieżka p_{se} o minimalnym czasie przejazdu nie musi składać się ze ścieżki o minimalnym czasie przejazdu z v_s do v_i oraz ścieżki o minimalnym czasie przejazdu z v_i do v_e.

Tabela 2

Ścieżka o minimalnym czasie przejazdu z wierzchołka $v_s = 1$ do wierzchołka $v_e = 15$ wyznaczona algorytmami Dijkstry i Bellmana–Forda

Wierzchołek / strefa	Godzina przyjazdu	Godzina wyjazdu	Czas dojazdu [min]	Koszt dojazdu	Numer linii
1 / 1	12:00	12:30	0	0.0	7
13 / 1	12:38	13:16	38	2.0	8
14 / 1	13:21	13:26	81	4.0	6
15 / 1	13:34		94	6.0	

Jako przykład zostanie rozpatrzone wyznaczanie ścieżki o minimalnym czasie przejazdu z $v_s = 1$ do $v_e = 15$ i godziny rozpoczęcia podróży $T_s = 12{:}00$. W tabeli 2 została przedstawiona ścieżka wyznaczona algorytmami Dijkstry i Bellmana–Forda, a w tabeli 3 ścieżka, która spośród wszystkich ścieżek o minimalnym czasie przejazdu ma także minimalny koszt przejazdu. Obydwie ścieżki składają się z tej samej sekwencji wierzchołków, przy czym różnią się liniami komunikacyjnymi, pojazdami, którymi odbywa się przejazd i godzinami wyjazdu z wierzchołków. Czas przejazdu w obydwu ścieżkach jest taki sam i wynosi 94 minuty, natomiast koszt przejazdu w pierwszej ścieżce jest równy 6.0 jednostek, a w drugiej

4.0 jednostki. W ścieżce przedstawionej w tabeli 3 czasy dojazdu do wierzchołków 13 i 14 są równe odpowiednio 74 i 82 minuty i nie są to czasy minimalne. Minimalne czasy dojazdu do tych wierzchołków są równe odpowiednio 38 i 81 minut i są to czasy dojazdu w ścieżce, którą przedstawiono w tabeli 2.

Tabela 3

Ścieżka o minimalnym czasie przejazdu z wierzchołka $v_s = 1$ do wierzchołka $v_e = 15$ mająca jednocześnie minimalny koszt przejazdu

Wierzchołek / strefa	Godzina przyjazdu	Godzina wyjazdu	Czas dojazdu	Koszt dojazdu	Numer linii
1 / 1	12:00	13:08	0	0.0	5
13 / 1	13:14	13:14	74	2.0	5
14 / 1	13:22	13:26	82	2.0	6
15 / 1	13:34		94	4.0	

2.3. Struktura bazy danych i procedury składowane

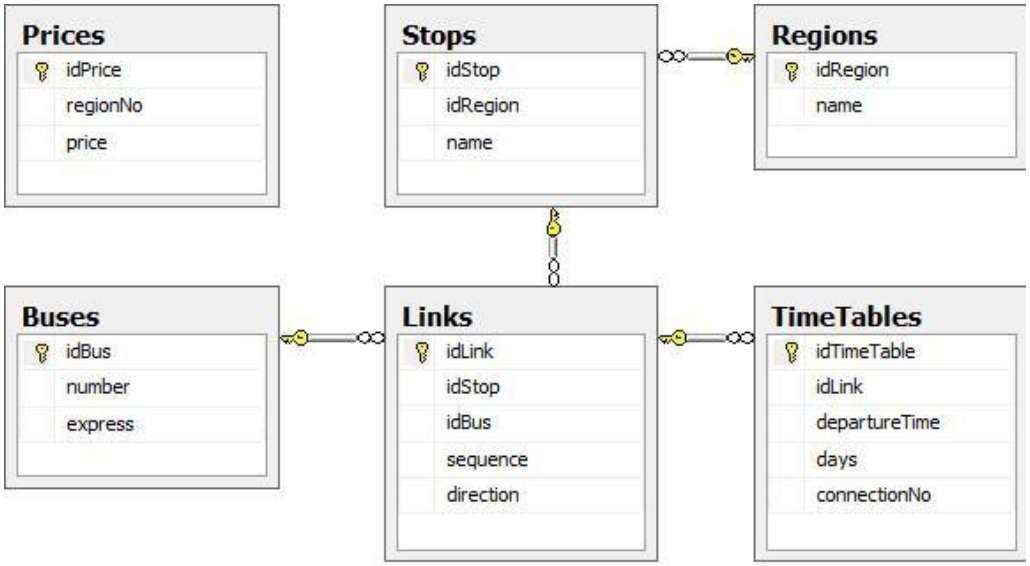
Graf *G* opisujący sieć komunikacyjną oraz rozkład jazdy są przechowywane w relacyjnej bazie danych, której diagram ERD został przedstawiony na rys. 2 [1, 9, 15, 41][2]. Jako serwer bazy danych został użyty Microsoft SQL Server 2008. Baza danych składa się z sześciu tabel: *Prices*, *Stops*, *Regions*, *Buses*, *Links* i *TimeTables*, powiązanych między sobą relacjami, wyjątkiem jest tabela *Prices*, która nie jest związana relacją z żadną inną tabelą.

Tabela *Regions* zawiera nazwy stref, do których należą przystanki. Informacja o przystankach tworzących sieć komunikacyjną znajduje się w tabeli *Stops*, gdzie dla każdego przystanku jest zapisana informacja o jego nazwie (*name*) i strefie, do której on należy (*idRegion*). Informacja o koszcie przejazdu znajduje się w tabeli *Prices*, gdzie liczba stref jest zapisana w kolumnie *regionNo*, a cena w kolumnie *price*. Przyjęte zostały następujące koszty przejazdu: $c_1 = 2{,}8$, $c_2 = 3{,}4$ i $c_3 = 4{,}2$ jednostki.

Tabela *Buses* zawiera informację o liniach komunikacyjnych, których pojazdy kursują w sieci. Dla każdej linii jest przechowywany jej numer (*number*) oraz informacja, czy jest ona linią zwykłą czy pospieszną (*express*). Trasy przejazdu pojazdów linii komunikacyjnych znajdują się w tabeli *Links*. Opis trasy linii *i*–tej składa się z *k*+1 wierszy tabeli, gdzie jednemu przystankowi należącemu do trasy odpowiada jeden wiersz. Pojedynczy wiersz zawiera: identyfikator przystanku w sieci komunikacyjnej (*idStop*), numer linii, której pojazd odjeżdża z danego przystanku (*idBus*) oraz pozycję danego przystanku w trasie linii komunikacyjnej (*sequence*), gdzie 0 oznacza przystanek początkowy linii, a *k* przystanek

[2] Diagram ERD został wygenerowany za pomocą narzędzia Microsoft SQL Server 2008 Management Studio.

końcowy linii. Ponieważ pojazd danej linii komunikacyjnej może kursować w obydwu kierunkach, więc kolumna *direction* zawiera informację o kierunku kursowania pojazdu.



Rys. 2. Diagram ERD bazy danych zawierającej sieć komunikacyjną i rozkład jazdy
Fig. 2. An ERD diagram of database containing the communication network and the timetable

Ostatnią tabelą jest *TimeTables*, która zawiera rozkład jazdy. Pojedynczy wiersz zawiera informację o godzinie odjazdu (*departureTime*) w ramach danego kursu (*connectionNo*) pojazdu określonej linii komunikacyjnej (*idLink*). W zależności od dnia tygodnia może obowiązywać inny rozkład jazdy, stąd w kolumnie *days* jest zapisana informacja o dniu tygodnia, w którym kursuje pojazd danej linii.

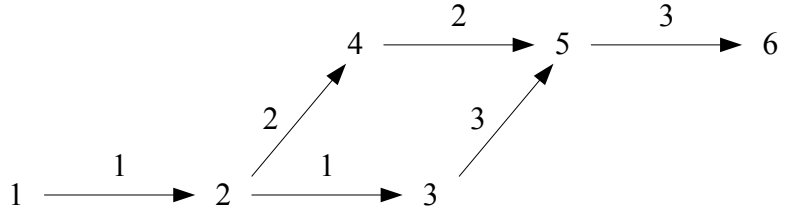
W bazie danych zdefiniowano procedury składowane, które zostały użyte w algorytmie wyznaczania ścieżki z v_s do v_e o minimalnym czasie przejazdu, z uwzględnieniem ograniczenia na koszt przejazdu. Zostały zdefiniowane następujące procedury składowane:

- *GetLines* – procedura zwraca listę wszystkich linii komunikacyjnych, których pojazdy odjeżdżają z zadanego wierzchołka w podanym przedziale czasowym, przy czym nie jest zwracana linia, pojazdem której dojechano do tego wierzchołka,
- *GetZone* – procedura zwraca numer strefy do której należy wierzchołek,
- *GetNextVertices* – procedura zwraca listę wszystkich wierzchołków będących następnikami danego wierzchołka w trasie danej linii komunikacyjnej,
- *GetDepartureTime* – procedura zwraca godzinę odjazdu z zadanego wierzchołka pojazdu zadanej linii komunikacyjnej w zadanym kierunku,
- *GetCost* – procedura zwraca koszt przejazdu w zależności od liczby przekroczonych stref.

3. Algorytm wyznaczania ścieżki o minimalnym czasie przejazdu z uwzględnieniem ograniczenia kosztu przejazdu

Zaproponowany algorytm umożliwia wyznaczenie w grafie G, reprezentującym sieć komunikacyjną, ścieżki z wierzchołka początkowego v_s do wierzchołka końcowego v_e. Wyznaczana jest ścieżka o minimalnym czasie przejazdu, przy czym jeżeli istnieje wiele takich ścieżek, to spośród nich wyznaczana jest ta, która ma minimalny koszt przejazdu. Ścieżki wyznaczane w algorytmie są opisane rekordami, gdzie każdy rekord opisuje pojedynczy wierzchołek, należący do ścieżki, i zawiera następujące pola:

- *current* – wierzchołek należący do ścieżki,
- *parent* – numer rekordu dla poprzedzającego w ścieżce wierzchołek *current* wierzchołka, w którym jest dokonywana ostatnia przesiadka,
- *departureTime* – godzina wyjazdu z wierzchołka *current*,
- *lineNumber* – linia komunikacyjna, pojazdem której dojeżdża się do wierzchołka *current*,
- *direction* – kierunek kursowania pojazdu linii *lineNumber*,
- *travelTime* – czas dojazdu do wierzchołka *current,*
- *travelCost* – koszt dojazdu do wierzchołka *current*.



Rys. 3. Ścieżki z wierzchołka początkowego $v_s = 1$ do wierzchołka końcowego $v_e = 3$
Fig. 3. Paths from the start vertex $v_s = 1$ to the final vertex $v_e = 3$

Niech, przykładowo, będą dane ścieżki z wierzchołka $v_s = 1$ do wierzchołka $v_e = 6$ (rys. 3). Na rysunku przy każdym łuku zaznaczono numer linii komunikacyjnej, której pojazd kursuje między danymi wierzchołkami. Istnieją dwie ścieżki z $v_s = 1$ do $v_e = 6$. Pierwszą jest ścieżka $p_1 = \langle 1, 2, 4, 5, 6\rangle$, a drugą $p_2 = \langle 1, 2, 3, 5, 6\rangle$. Rekordy opisujące ścieżki p_1 i p_2 zostały przedstawione na rys. 4, gdzie przedstawiono wyłącznie pola, pozwalające odtworzyć ścieżkę. Ponieważ rekordy nie opisują pełnych ścieżek tylko wierzchołki, w których są dokonywane przesiadki, więc ścieżka p_1 jest opisana rekordami o indeksach 0, 1, 4, 7, natomiast ścieżkę p_2 opisują rekordy o indeksach 0, 2, 6.

indeks rekordu:	0	1	2	3
current:	1	2	3	4
parent:	-1	0	0	1
lineNumber:	-1	1	1	2

indeks rekordu:	4	5	6	7
current:	5	5	6	6
parent:	1	2	2	4
lineNumber:	2	3	3	3

Rys. 4. Rekordy opisujące ścieżki przedstawione na rys. 3
Fig. 4. Records describing path presented in fig. 3

Algorytm wyznaczania ścieżki został przedstawiony w postaci procedury *FindPath* o następującej treści:

```
Wejście:     G – skierowany graf ważony opisujący sieć komunikacyjną
             v_s, v_e – wierzchołek początkowy i końcowy
             T_s - godzina rozpoczęcia podróży
          T_w – maksymalny czas oczekiwania na przesiadkę
Wyjście: P - wyznaczona ścieżka
 1:  procedure FindPath(G, v_s, v_e, T_s, T_w, P)
 2:  begin
 3:    Q := (v_s, -1, T_s, -1, -1, 0, 0);  {utworzenie rekordu startowego}
 4:    i := 0;                             {indeks aktualnie analizowanego rekordu}
 5:    P := ∅;                             {lista wyznaczonych ścieżek}
 6:    while i < Q.size() do               {dla wszystkich rekordów w liście Q}
 7:      GetLines(Q[i].current, T_w, L);
 8:      S := ∅;
 9:      for each l_i ∈ L do
10:        if CheckLine(l_i, Q, i) then
11:          AddVertices(l_i, Q, i, S);
12:        end if
13:      end for
14:      for each u ∈ S do
15:        if CheckVertex(u, Q, i, P) then
16:          if u.current = v_e then       {znaleziono ścieżkę do wierzchołka v_e}
17:            CheckFinalSolution(u, Q, P);
18:          else
19:            Q := Q ∪ u;
20:          end if
21:        end if
22:      end for
23:      i := i + 1;       {przejście do następnego rekordu w liście Q}
24:    end while
25:  end
```

Rekordy opisujące ścieżki są umieszczone w liście Q, a wyznaczone ścieżki w liście P, która może zawierać wiele ścieżek, jednak wszystkie mają taki sam czas przejazdu. Ścieżki są wyznaczane w iteracji **while**, w której są analizowane wszystkie rekordy, znajdujące się w liście Q oraz następuje odwiedzanie wierzchołków grafu G (wiersze 6–24). W tym celu są analizowane wszystkie linie komunikacyjne, pojazdami których można wyjechać z aktualnie analizowanego wierzchołka $Q[i].current$ (wiersze 9–13). Dla każdego odwiedzonego

wierzchołka jest tworzony i umieszczany w liście *S* (wiersz 11) rekord, opisujący sposób dojazdu do tego wierzchołka. Następnie każdy rekord *u* znajdujący się w liście *S* jest poddawany analizie w celu utworzenia nowej ścieżki z wierzchołka v_s do wierzchołka opisanego rekordem *u* (wiersze 14–22).

Po przeanalizowaniu wszystkich rekordów w liście *Q* (wiersze 6–24 procedury *FindPath*) lista *P* zawiera wszystkie ścieżki z wierzchołka v_s do wierzchołka v_e o minimalnym czasie przejazdu. Jeżeli jest wiele takich ścieżek, to wszystkie mają minimalny koszt przejazdu.

Wierzchołek, do którego można dojechać bez przesiadki z wierzchołka *Q*[*i*].*current*, jest odwiedzany wyłącznie w przypadku, kiedy w aktualnie analizowanym wierzchołku *Q*[*i*].*current* zostanie dokonana przesiadka. Sprawdzenie, czy w wierzchołku tym zostanie dokonana przesiadka, odbywa się w funkcji *CheckLine* (wiersz 10):

```
Wejście: l_i – linia komunikacyjna
         Q – lista rekordów opisujących wierzchołki należące do ścieżek
         i – indeks aktualnie analizowanego rekordu w liście Q
Wyjście: true – jeżeli w ścieżce do wierzchołka opisanego rekordem Q[i] przejazd
                nie odbywa się pojazdem l_i.number w kierunku l_i.direction
         false – w przeciwnym przypadku
 1:  function CheckLine(l_i, Q, i): boolean;
 2:  begin
 3:    while i ≠ -1 do
 4:      if Q[i].direction = l_i.direction and Q[i].number = l_i.number then
 5:        return false;
 6:      end if
 7:      i := Q[i].parent;
 8:    end while
 9:    return true;
10:  end
```

W przypadku dokonania przesiadki w wierzchołku *Q*[*i*].*current*, w procedurze *FindPath* (wiersz 11) następuje odwiedzenie wszystkich wierzchołków, do których można dojechać pojazdem linii komunikacyjnej l_i bez przesiadki. Odwiedzenie wierzchołków odbywa się w procedurze *AddVertices*:

```
Wejście: l_i – linia komunikacyjna
         Q – lista rekordów opisujących wierzchołki należące do ścieżek
         i – indeks aktualnie analizowanego rekordu w liście Q
Wyjście: S – lista rekordów opisujących wierzchołki, do których można dojechać
             pojazdem linii l_i bez przesiadki
 1:  procedure AddVertices(b, Q, i, S);
 2:  begin
 3:    GetNextVertices(Q[i].current, l_i, V_i);
 4:    S := ∅;
 5:    zones := 0;                              {liczba przekroczonych stref}
 6:    currentZone := GetZone(Q[0].current);    {aktualna strefa}
 7:    for each v ∈ V_i do
 8:      if v ≠ Q[0].current then     {wierzchołek v nie jest wierzchołkiem v_s}
 9:        s.direction := l_i.direction;
10:        s.lineNumber := l_i.number;
11:        s.current := v;
12:        s.parent := i;
13:        s.departureTime := GetDepartureTime(s);
14:        s.travelTime := s.departureTime - Q[0].departureTime;
```

```
15:         if currentZone ≠ GetZone(v) then   {przekroczenie strefy}
16:           currentZone := GetZone(v);
17:           zones := zones + 1;
18:         end if
19:         s.travelCost := Q[i].travelCost + GetCost(zones);
20:         S := S ∪ s;
21:       end if
22:     end for
23:   end
```

Wynikiem wykonania procedury *AddVertices* jest lista rekordów S opisujących wierzchołki, do których można dojechać bez przesiadki pojazdem linii l_i. Następnie każdy rekord u znajdujący się w liście S jest poddawany analizie w celu utworzenia nowej ścieżki z wierzchołka v_s do wierzchołka opisanego rekordem u (wiersze 14–22 procedury *FindPath*). Próba utworzenia nowej ścieżki jest podejmowana, jeżeli wierzchołek opisany rekordem u nie należy do ścieżki z wierzchołka v_s do wierzchołka $Q[i].current$ i czas dojazdu do tego wierzchołka nie jest większy od czasu przejazdu dotychczas znalezionego rozwiązania znajdującego się w liście P. Warunki te są sprawdzane za pomocą funkcji *CheckVertex* (wiersz 15 procedury *FindPath*):

```
Wejście: u – rekord opisujący wierzchołek
         Q – lista rekordów opisujących wierzchołki należące do ścieżek
         i – indeks aktualnie analizowanego rekordu w liście Q
         P – lista dotychczas wyznaczonych ścieżek z wierzchołka vs do ve
Wyjście: false – jeżeli czas dojazdu do wierzchołka opisanego rekordem u jest
                 większy od czasu przejazdu dotychczas znalezionej ścieżki
                 w liście P lub wierzchołek opisany rekordem u należy do ścieżki
                 z wierzchołka vs do wierzchołka Q[i].current
         true – w przeciwnym przypadku
 1:  function CheckVertex(u, Q, i, P): boolean;
 2:  begin
 3:    if P.size() > 0 cand u.travelTime > P[0].travelTime then
 4:      return false;
 5:    end if
 6:    while Q[i] ≠ Q[0] do
 7:      if u.current = Q[i].current then
 8:        return false;
 9:      end if
10:      i := Q[i].parent;
11:    end while
12:    return true;
13:  end
```

Rekord u jest poddawany dalszej analizie, jeżeli funkcja *CheckVertex* zwraca wartość **true**. W przypadku kiedy rekord ten opisuje wierzchołek końcowy v_e, oznacza to, że wyznaczono nową ścieżkę z wierzchołka początkowego v_s do wierzchołka v_e i jest dokonywana jej ocena oraz jest podejmowana próba dodania jej do listy wyznaczonych ścieżek P. Operacja ta jest wykonywana za pomocą procedury *CheckFinalSolution* (wiersz 17).

```
Wejście: u – rekord opisujący wierzchołek
         Q – lista rekordów opisujących wierzchołki należące do ścieżek
         P – lista dotychczas wyznaczonych ścieżek z wierzchołka vs do ve
Wyjście: P – lista wyznaczonych ścieżek z wierzchołka vs do ve
 1:  procedure CheckFinalSolution(u, Q, P);
 2:  begin
```

```
 3:    if P.size() > 0 then      {wyznaczono już ścieżkę do wierzchołka v_e}
 4:      if u.travelTime < P[0].travelTime or u.travelTime = P[0].travelTime and
            u.travelCost < P[0].travelCost then
 5:        clear(P);             {usunięcie ścieżek z listy P}
 6:        CreatePath(u, Q, p);      {odtworzenie wyznaczonej ścieżki}
 7:        P := p;
 8:      else if u.travelTime = P[0].travelTime and
                 u.travelCost = P[0].travelCost then
 9:        CreatePath(u, Q, p);      {odtworzenie wyznaczonej ścieżki}
10:        P := P ∪ p;
11:      end if
12:    else
13:      CreatePath(u, Q, p);        {odtworzenie wyznaczonej ścieżki}
14:      P := p;
15:    end if
16:  end
```

Na podstawie rekordów znajdujących się w liście Q oraz rekordu u jest odtwarzana nowa ścieżka z v_s do v_e (wiersze 6, 9 i 13), co odbywa się z użyciem procedury *CreatePath*:

```
Wejście: u – rekord opisujący wierzchołek
         Q – lista rekordów opisujących wierzchołki należące do ścieżek
Wyjście: p – ścieżka z wierzchołka v_s do wierzchołka v_e odtworzona na podstawie
             rekordu u oraz rekordów należących do listy Q
 1:  procedure CreatePath (u, Q, p);
 2:  begin
 3:    p.travelTime := u.travelTime;
 4:    p.travelCost := u.travelCost;
 5:    p.path := u;
 6:    while u.parent ≠ -1 do
 7:      u := Q[u.parent];
 8:      p.path := {u} ∪p.path;
 9:    end while
10:  end
```

4. Wyniki badań eksperymentalnych

Badania eksperymentalne z użyciem algorytmu *FindPath* przedstawionego w rozdziale 3 zostały przeprowadzone z użyciem sieci tramwajowej przewoźnika KZK GOP [42], która składa się z 236 przystanków i 25 linii tramwajowych, a rozkład jazdy zawiera 54749 pozycji. Wykonane zostały badania dla czasów oczekiwania T_w równych odpowiednio 5, 10 i 15 minut. Pojedynczy eksperyment polegał na wyznaczeniu połączenia dla zadanej pary przystanków początkowego i końcowego oraz godziny rozpoczęcia podróży T_s = 8:00.

Badania eksperymentalne zostały wykonane z użyciem komputera PC z procesorem AMD Athlon™ II P320 Dual-Core 2.1GHz oraz 4GB RAM, pod kontrolą systemu operacyjnego Windows 7. Jako serwer bazy danych został użyty Microsoft SQL Server 2008. Wybrane wyniki przeprowadzonych badań przedstawiono w tabelach 4 i 5. W pozostałych eksperymentach czas wyznaczania rozwiązania nie przekroczył 2 [s]. Maksymalny czas wyznaczania połączenia we wszystkich przeprowadzonych eksperymentach nie przekroczył

43 [s]. W niektórych przypadkach wyznaczenie połączenia zakończyło się niepowodzeniem, co było spowodowane przyjętym czasem oczekiwania T_w na przesiadkę, który okazał się za krótki. Czas wyznaczania połączenia w tych przypadkach nie przekroczył jednak 6 [ms].

Tabela 4

Wyniki przeprowadzonych badań eksperymentalnych z użyciem algorytmu *FindPath*

Numer eksp.	Przystanek początkowy – przystanek końcowy	T_w [min]	Liczba przesiadek	Długość trasy	Czas wyznaczania połączenia [s.ms]
1	Bytom Plac Sikorskiego – Gliwice Zajezdnia	5	–	–	0.05
		10	1	36	11.93
		15	1	27	25.14
2	Bytom Plac Sikorskiego – Katowice Dworzec PKP	5	1	39	1.66
		10	1	37	11.07
		15	1	26	25.79
3	Bytom Plac Sikorskiego – Mysłowice Dworzec PKP	5	3	63	29.89
		10	1	48	11.76
		15	1	48	28.26
4	Chorzów Ratusz – Katowice Dworzec PKP	5	1	14	2.86
		10	0	19	0.67
		15	0	19	0.80
5	Gliwice Zajezdnia – Bytom Plac Sikorskiego	5	–	–	0.03
		10	–	–	0.03
		15	1	28	2.58
6	Gliwice Zajezdnia – Chorzów Ratusz	5	–	–	0.03
		10	–	–	0.03
		15	1	41	2.58
7	Gliwice Zajezdnia – Katowice Dworzec PKP	5	–	–	0.03
		10	–	–	0.03
		15	1	49	2.54
8	Gliwice Zajezdnia – Mysłowice Dworzec PKP	5	–	–	0.03
		10	–	–	0.03
		15	2	71	42.97
9	Gliwice Zajezdnia – Ruda Śląska Chebzie Pętla	5	0	23	0.03
		10	0	23	0.03
		15	0	23	0.18
10	Gliwice Zajezdnia – Siemianowice Plac Skargi	5	–	–	0.03
		10	–	–	0.03
		15	2	55	42.48
11	Gliwice Zajezdnia – Sosnowiec Zagórze Pętla	5	–	–	0.03
		10	–	–	0.03
		15	3	76	8.08
12	Katowice Dworzec PKP – Bytom Plac Sikorskiego	5	2	31	2.72
		10	1	40	15.18
		15	1	28	25.81

Tabela 5

Wyniki przeprowadzonych badań eksperymentalnych z użyciem algorytmu *FindPath* – cd.

Numer eksp.	Przystanek początkowy – przystanek końcowy	T_w [min]	Liczba przesiadek	Długość trasy	Czas wyznaczania połączenia [s.ms]
13	Mysłowice Dworzec PKP – Bytom Plac Sikorskiego	5	–	–	0.05
		10	–	–	0.06
		15	2	63	17.57
14	Mysłowice Dworzec PKP – Chorzów Ratusz	5	–	–	0.03
		10	–	–	0.03
		15	1	42	11.13
15	Mysłowice Dworzec PKP – Katowice Dworzec PKP	5	–	–	0.03
		10	–	–	0.03
		15	1	24	10.99
16	Mysłowice Dworzec PKP – Siemianowice Plac Skargi	5	–	–	0.03
		10	–	–	0.03
		15	2	32	14.56
17	Mysłowice Dworzec PKP – Sosnowiec Zagórze Pętla	5	–	–	0.03
		10	–	–	0.03
		15	1	21	11.11
18	Siemianowice Plac Skargi – Bytom Plac Sikorskiego	5	1	32	2.82
		10	1	32	5.32
		15	1	32	8.25
19	Siemianowice Plac Skargi – Mysłowice Dworzec PKP	5	2	31	3.14
		10	1	31	5.40
		15	1	31	8.26
20	Siemianowice Plac Skargi – Sosnowiec Zagórze Pętla	5	2	38	14.83
		10	1	36	5.33
		15	1	36	8.23
21	Sosnowiec Zagórze Pętla – Bytom Plac Sikorskiego	5	–	–	0.03
		10	2	67	8.79
		15	2	67	11.63
22	Sosnowiec Zagórze Pętla – Chorzów Ratusz	5	–	–	0.03
		10	2	51	9.29
		15	2	45	11.48
23	Sosnowiec Zagórze Pętla – Mysłowice Dworzec PKP	5	–	–	0.03
		10	1	21	3.31
		15	1	21	4.13
24	Sosnowiec Zagórze Pętla – Ruda Śląska Chebzie Pętla	5	–	–	0.03
		10	2	55	8.66
		15	2	55	11.93
25	Sosnowiec Zagórze Pętla – Siemianowice Plac Skargi	5	–	–	0.03
		10	2	36	6.90
		15	2	36	14.67

Wyznaczenie połączenia nie jest możliwe w jednym z dwóch przypadków. W pierwszym z nich nie ma możliwości wyjazdu z przystanku początkowego tak, aby czas oczekiwania nie przekroczył czasu T_w. Przykładem są eksperymenty 13–17, gdzie czas oczekiwania na przystanku początkowym był dłuższy niż 10 minut, dlatego wyznaczono połączenie wyłącznie dla czasu T_w = 15 minut. W drugim przypadku w celu dojazdu do przystanku końcowego zachodzi konieczność dokonania przesiadki i czas oczekiwania na przesiadkę przekracza czas T_w. Warunek ten zachodzi np. w eksperymentach 5–8, 10–11, w których nie zostały wyznaczone połączenia, ponieważ nie istnieje możliwość dojazdu do przystanku końcowego bez przesiadki, a przy dokonaniu przesiadki czas oczekiwania jest większy niż T_w = 5 i 10 minut.

Dla każdego z trzech zadanych czasów oczekiwania T_w w większości przypadków zostało wyznaczone połączenie z identyczną liczbą przesiadek. Wyjątkiem są eksperymenty 3, 4, 12, 19, 20, w których dla czasu oczekiwania T_w = 5 minut zostało wyznaczone połączenie z większą liczbą przesiadek niż dla czasów T_w = 10 i 15 minut. Wyznaczenie połączenia z mniejszą liczbą przesiadek dla czasu T_w = 5 minut okazało się niemożliwe, gdyż czas oczekiwania na przesiadkę w każdym z tych połączeń był większy niż T_w.

Dla zadanej pary przystanków początkowego i końcowego, wraz ze wzrostem czasu oczekiwania T_w, można zaobserwować wzrost czasu wyznaczania połączenia. Czas oczekiwania T_w wpływa na liczbę sposobów wyjazdu z danego przystanku, które należy przeanalizować podczas wyznaczania połączenia. Liczba ta rośnie wraz ze wzrostem czasu T_w, a co za tym idzie rośnie rozmiar analizowanej przestrzeni rozwiązań, co wpływa na wzrost czasu wyznaczania rozwiązania.

Tabela 6

Fragment trasy połączenia komunikacyjnego wyznaczonego w eksperymencie nr 2 dla czasu oczekiwania T_w = 5 minut

Przystanek	Godzina przyjazdu	Godzina wyjazdu	Linia tramwajowa
Bytom Pl. Sikorskiego	8:00	8:00	9
…	…	…	…
Godula Inkubator	8:15	8:15	9
Godula Pl. Niepodległości	8:17	8:17	9
Chebzie Pawła	8:22	8:22	9
Chebzie Pętla	8:25	8:30	11
Chebzie Pawła	8:32	8:32	11
Lipiny Zakłady Silesia	8:33	8:33	11
…	…	…	…
Katowice Dworzec PKP	9:18		

Tabela 7

Fragment trasy połączenia komunikacyjnego wyznaczonego w eksperymencie nr 2 dla czasu oczekiwania T_w = 10 minut

Przystanek	Godzina przyjazdu	Godzina wyjazdu	Linia tramwajowa
Bytom Pl. Sikorskiego	8:00	8:00	9
…	…	…	…
Godula Inkubator	8:15	8:15	9
Godula Pl. Niepodległości	8:17	8:17	9
Chebzie Pawła	8:22	8:32	11
Lipiny Zakłady Silesia	8:33	8:33	11
…	…	…	…
Katowice Dworzec PKP	9:18		

Godne uwagi są trasy połączeń komunikacyjnych wyznaczone w eksperymencie 2 dla czasu oczekiwania T_w = 5 i 10 minut, których fragmenty zostały przedstawione w tabelach 6 i 7. Tabele zawierają godzinę przyjazdu i wyjazdu z danego przystanku oraz numer linii tramwajowej, którą następuje przejazd do następnego przystanku. Dla czasu T_w = 10 minut (tabela 7) przesiadka jest dokonywana na przystanku „Chebzie Pawła", gdzie czas oczekiwania na przesiadkę wynosi 10 minut. Z tego powodu dla czasu T_w = 5 minut została wyznaczona inna trasa połączenia, która także zawiera przystanek „Chebzie Pawła", przy czym przesiadka jest dokonywana na przystanku „Chebzie Pętla", a czas oczekiwania wynosi 5 minut (tabela 6). Trasa ta zawiera jednak cykl przebiegający przez przystanki: „Chebzie Pawła", „Chebzie Pętla", „Chebzie Pawła", natomiast pozostała część trasy jest taka sama jak w połączeniu dla czasu T_w = 10 minut. Czas przejazdu w cyklu jest równy 10 minut i jest on równy czasowi oczekiwania na przystanku „Chebzie Pawła" w połączeniu przedstawionym w tabeli 7, stąd czas przejazdu w obydwu połączeniach jest identyczny. Cykl nie wpływa także na koszt przejazdu, gdyż nie jest w nim przekraczana żadna strefa, dlatego obydwa połączenia mają także ten sam koszt przejazdu.

5. Podsumowanie

W niniejszej pracy został omówiony problem wyznaczania połączeń w sieciach komunikacyjnych o minimalnym czasie przejazdu pomiędzy parą przystanków początkowym i końcowym dla zadanej godziny rozpoczęcia podróży. Dodatkowo dany jest maksymalny czas oczekiwania na przesiadkę. W przypadku istnienia wielu połączeń o minimalnym czasie przejazdu konieczne jest wyznaczenie tego połączenia, które ma minimalny koszt przejazdu.

W pracy został zaproponowany algorytm umożliwiający wyznaczenie połączenia komunikacyjnego, które spośród wszystkich połączeń o minimalnym czasie przejazdu ma minimalny koszt przejazdu. Struktura sieci komunikacyjnej oraz rozkład jazdy pojazdów komunika-

cyjnych są przechowywane w relacyjnej bazie danych. W algorytmie podczas wyznaczania połączenia komunikacyjnego korzysta się z procedur składowanych, które są zdefiniowane w bazie danych.

Dla opracowanego algorytmu przeprowadzono badania eksperymentalne z użyciem sieci tramwajowej przewoźnika KZK GOP. Dla zadanej pary przystanków początkowego i końcowego wyznaczono połączenia dla trzech różnych czasów oczekiwania na przesiadkę: 5, 10 i 15 minut. W niektórych przypadkach ograniczenie odnośnie do czasu oczekiwania było powodem braku istnienia połączenia. Najmniejszy czas wyznaczania połączenia uzyskano dla połączeń bez przesiadki i nie przekroczył on 1 [s], a maksymalny czas wyznaczania połączenia komunikacyjnego nie przekroczył 43 [s].

BIBLIOGRAFIA

1. Barker R.: CASE*Method – modelowanie związków encji. WNT, Warszawa 1996.
2. Bellman R.: On a routing problem. Quarterly of Applied Mathematics, Vol. 16(1), 1958, s. 87÷90.
3. Brumbaugh–Smith J., Shier S.: An empirical investigation of some bicriterion shortest path algorithms. European Journal of Operational Research, Vol. 43, 1989, s. 216÷224.
4. Climaco J. C., Martins E. Q. V.: A bicriterion shortest path algorithm. European Journal of Operational Research, Vol. 11, No. 4, 1982, s. 399÷404.
5. Coello Coello C. A., van Veldhuizen D. A., Lamont G. B.: Evolutionary algorithms for solving multi–objective problems. Kluwer Academic Publishers, New York 2002.
6. Corley H. W., Moon I. D.: Shortest paths in networks with vector weights. Journal of Optimization Theory and Application, Vol. 46, No. 1, s. 79÷86, 1985.
7. Cormen T. H., Leiserson Ch. E., Rivest R. L.: Wprowadzenie do algorytmów. WNT, Warszawa 2000.
8. Daellenbach H. G., De Kluyver C. A.: Note on multiple objective dynamic programming. Journal of the Operational Research Society, Vol. 31, No. 7, 1980, s. 591÷594.
9. Date C.J.: Wprowadzenie do baz danych. WNT, Warszawa 1981.
10. Dell'Olmo P., Gentili M., Scozzari A.: On finding dissimilar Pareto–optimal paths. European Journal of Operational Research, Vol. 162, No. 1, 2005, s. 70÷82.
11. Dijkstra E. W.: A note on two problems in connexion with graphs. Numerical Mathematics, Vol. 1, 1959, s. 269÷271.
12. Ford L. R.: Network Flow Theory. Report P–923, The Rand Corporation, Santa Monica, CA, USA, 1956.

13. Floyd R.: Algorithm 97: Shortest path. Communications of the ACM, Vol. 5(6), 1962, s. 345.
14. Hansen P.: Bicriterion path problems. Multiple Criteria Decision Making: Theory and Application, Springer–Verlag , Berlin, Germany, 1980, s. 109÷127.
15. Jaszkiewicz A.: Inżynieria oprogramowania. Wydawnictwo HELION, Gliwice 1997.
16. Johnson D. B.: Efficient algorithms for shortest paths in sparse networks. Journal of the ACM, Vol. 24(1), 1977, s. 1÷13.
17. Jungnickel D.: Graphs, networks and algorithms. Springer, Berlin 1999.
18. Lipski W.: Kombinatoryka dla programistów. WNT, Warszawa 1989.
19. Machuca E., Mandow L., Pérez de la Cruz J. L.: An Evaluation of Heuristic Functions for Bicriterion Shortest Path Problems. Proceedings of the XIV Portuguese Conference on Artificial Inteligence (EPIA 2009), Universidade de Aveiro, 2009, s. 205÷216.
20. Mandow L., Pérez de la Cruz J. L.: Frontier search for Bicriterion Shortest Path Problems. Proceeding of the 2008 conference on ECAI 2008 18th European Conference on Artificial Intelligence, IOS Press, 2008, s. 480÷484.
21. Mandow L., Pérez de la Cruz J. L.: Path recovery in frontier search for multiobjective shortest path problems. Journal of Intelligent Manufacturing, Vol. 21, No. 1, 2008, s. 89÷99.
22. Martins E. Q. V.: On a multicriteria shortest path problem. European Journal of Operational Research, Vol. 16, No. 2, 1982, s. 236÷245.
23. Martí R., González Velarde J. L., Duarte A.: Heuristics for the bi–objective path dissimilarity problem. Computers & Operations Research, Vol. 36, No. 11, 2009, s. 2905÷2912.
24. Mote J., Murthy I., Olson D. L.: A parametric approach to solving bicriterion shortest path problems. European Journal of Operational Research, Vol. 53, No. 1, 1991, s. 81÷92.
25. Pareto V.: Course d'Economie Politique, F. Rouge, Lausanne 1896.
26. Peschel M., Riedel C.: Polioptymalizacja. Metody podejmowania decyzji kompromisowych w zagadnieniach inżynieryjno-technicznych. WNT, Warszawa 1979.
27. Raith A., Ehrgott M.: A comparison of solution strategies for biobjective shortest path problems. Journal Computers and Operations Research, Vol. 36, No. 4, 2009, s. 1299÷1331.
28. Reingold E. M., Nievergelt J., Deo N.: Algorytmy kombinatoryczne. PWN, Warszawa 1985.
29. Skriver A. J. V., Andersen K.A.: A label correcting approach for solving bicriterion shortest-path problems. Computers & Operations Research, Vol. 27, 2000, s. 507÷524.
30. Stadler W.: A survey of Multicriteria Otimization or the Vector Maximum Problem. Part I: 1776–1960, Journal of Optimization Theory & Aplication, Vol. 29, No. 1, 1979, s. 1÷52.

31. Tung C. T., Chew K. L.: A bicriterion Pareto–optimal path algorithm. Asia–Pacific Journal of Operational Research, Vol. 5, 1988, s. 166÷172.
32. Ulungu E. L., Teghem J.: Multi–objective combinatorial optimization problems: a survey. Journal of Multi-Criteria Decision Analysis, Vol. 3, No. 2, 1994, s. 83÷104.
33. Voorneveld M.: Characterization of Pareto dominance. Operations Research Letters, Vol. 31, No. 1, 2003, s. 7–11.
34. Warshall S.: A theorem on Boolean matrices. Journal of the ACM, Vol. 9(1), 1962, s. 11÷12.
35. Widuch J.: Problem wyznaczania połączeń w sieciach komunikacyjnych. Studia Informatica, Vol. 22, nr 4 (46), Gliwice 2001, s. 117÷134.
36. Widuch J.: Wyznaczanie połączeń w sieciach komunikacyjnych o zmiennych wagach. Studia Informatica, Vol. 23, nr 4(51), Gliwice 2002, s. 85÷104.
37. Widuch J.: Algorytmy optymalizacji wielokryterialnej w problemach komunikacyjnych. Rozprawa doktorska, Politechnika Śląska, Gliwice 2008.
38. Widuch J.: Rozwiązanie problemu wyznaczania połączeń w sieciach komunikacyjnych za pomocą zmodyfikowanego algorytmu wyznaczania K najkrótszych ścieżek. Studia Informatica, Vol. 31, nr 1(88), Gliwice 2010, s. 55÷70.
39. Widuch J.: Rozwiązanie problemu wyznaczania połączeń w sieciach komunikacyjnych z zastosowaniem metody skalaryzacji. Studia Informatica, Vol. 32, nr 4A(100), s. 57÷81, Gliwice 2011.
40. Widuch J.: Algorytmy optymalizacji tras przejazdu pojazdów. Studia Informatica, Vol. 32, nr 4A(100), s. 83÷111, 2011.
41. Yourdon E.: Współczesna analiza strukturalna. WNT, Warszawa 1996.
42. Strona domowa przewoźnika KZK GOP: http://www.kzkgop.com.pl/.

Wpłynęło do Redakcji 27 lipca 2012 r.

Abstract

In this paper we considered the communication networks routing problem (CNRP), in particular, we analysed the problem and properties of a route. The goal of the problem is to find a route from the given start stop to the given final stop, where the time of starting travel at the start stop and the maximal time of waiting for a change are given additionally. The goal is to determine a route minimizing the time of travel and if exist more routes with minimal

the time of travel, among them the route with minimal the cost of travel should be determined.

In the paper we proposed the algorithm for solving the CNRP. The communication network and the timetable of the lines are stored in a relational database. Additionally, stored procedures defined in the database are used in the algorithm. The algorithm was implemented and tested for a tramway network of KZK GOP corporation. For a given pair of the start and the final stops the route was computed for three different times of waiting for a change: 5, 10 and 15 minutes. The results of tests are presented in Tab. 4–5. In some cases, restrictions on the maximal time of waiting for a change was the reason for the absence of determination of the route. The smallest computation time was obtained for the route without change and the maximal computation time did not exceed 43 seconds. On the basis of the test results we found that the algorithm exhibits a reasonable execution time for the network of the size about 250 stops.

Adres

Jacek WIDUCH: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16,
44–100 Gliwice, Polska, jacek.widuch@polsl.pl
Daniel WICHER: dwicher@future-processing.com