STUDIA INFORMATICA 2001
Volume 22 Number 4 (46)

Jacek WIDUCH
Politechnika Śląska, Instytut Informatyki

PROBLEM WYZNACZANIA POŁĄCZEŃ W SIECIACH KOMUNIKACYJNYCH

Streszczenie. W pracy przedstawiono algorytm wyznaczania połączeń w sieciach komunikacyjnych. Algorytm umożliwia wyznaczenie połączeń należących do zbioru rozwiązań niezdominowanych. Omówiono również problem optymalizacji wielokryterialnej, którego przykładem jest problem będący tematem pracy. Oprócz tego przedstawiono wyniki badań eksperymentalnych.

SOLVING THE COMMUNICATION NETWORKS ROUTING PROBLEM

Summary. The paper shows an algorithm for the communication networks routing problem. The algorithm makes possible to find the routes which belong to the set of non-dominated solutions. We also discuss the multicriteria optimization problem, which example is the problem being the subject of this paper. Apart from that the results of experimental tests are presented.

1. Wstęp

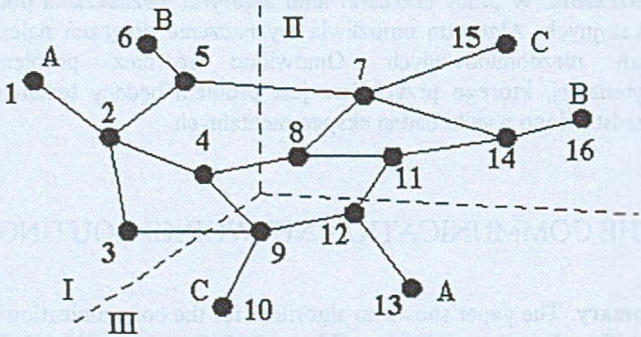
Wiele systemów transportowych, dystrybucyjnych i komunikacyjnych jest projektowanych i badanych za pomocą modeli sieciowych. W większości przypadków grafy wykorzystywane w tego typu zagadnieniach są tak duże, że ich analiza bez pomocy komputera jest uciążliwa, a niekiedy niemożliwa. Coraz powszechniejsze staje się stosowanie komputerów jako systemów wspomagania decyzji. Jednym z nich jest system HAFAS, dostępny w postaci aplikacji lub w sieci Internet pod adresem http://bahn.hafas.de. System umożliwia wyszukanie połączeń kolejowych w Europie.

Algorytm przedstawiony w niniejszej pracy może być podstawą do stworzenia podobnego systemu, który umożliwia znalezienie połączeń w dowolnej sieci komunikacyjnej.

Przedstawiony problem należy do klasy problemów optymalizacji wielokryterialnej. W tego typu zadaniach o wyborze konkretnego rozwiązania decyduje nie jedno, ale wiele kryteriów. W przedstawionym problemie na wybór rozwiązania ma wpływ czas i koszt podróży. Rozwiązaniem zadania tej klasy w ogólnym przypadku nie jest jedno konkretne rozwiązanie, ale zbiór rozwiązań. W skrajnych przypadkach zbiór ten może składać się tylko z jednego rozwiązania. Przedstawiony algorytm umożliwia wyznaczenie wymienionego zbioru rozwiązań.

2. Sformułowanie problemu

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



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

Wierzchołki grafu, o numerach 1-16, reprezentują przystanki lub stacje końcowe pojazdów komunikacyjnych (w dalszej części pracy nazywanych w skrócie pojazdami). Stacje końcowe są dodatkowo oznaczone etykietami linii komunikacyjnych, np. A, B, C itd. Krawędzie grafu reprezentują drogi przejazdu pomiędzy przystankami. Ponieważ przejazdy mogą się odbywać w obu kierunkach, każdą krawędź można uważać za dwa łuki przeciwnie skierowane. Pojazdy poszczególnych linii (A, B, ...) kursują pomiędzy stacjami końcowymi z określoną częstotliwością. Przykładowo pojazd linii A wyrusza ze stacji końcowej (przystanek 1) o godzinie t_0 i przejeżdża przez przystanki 2, 4, 8, ..., 13 odpowiednio o godzinach $t_0 + \delta_0$, $t_0 + \delta_1$, $t_0 + \delta_2$, ..., $t_0 + \delta_i$ ($\delta_0 < \delta_1 < \delta_2 < \ldots < \delta_i$). Następny kurs pojazdu linii A rozpoczyna się o godzinie t_1 ($t_1 = t_0 + \beta$, β jest określone częstotliwością kursowania i może się zmieniać w różnych porach dnia), tak więc pojazd dociera do przystanków 2, 4, ... w godzinach $t_1 + \delta_0$, $t_1 + \delta_1$, ... Wielkości δ_0, δ_1, ..., δ_i są dane i określone rozkładem jazdy. Jednocześnie z pojazdem linii A kursują pojazdy innych linii komunikacyjnych przemieszczając

się pomiędzy swoimi stacjami końcowymi. Godziny rozpoczęcia kursu na stacji początkowej, godziny przybycia na poszczególne przystanki oraz trasa przejazdu, tj. sekwencja przystanków, przez które pojazd przejeżdża, są dane. Trasy przejazdu pojazdu pomiędzy stacjami końcowymi mogą się różnić, np. pojazd linii A może przejeżdżać z przystanku 1 do 13 inną trasą niż z przystanku 13 do 1. W tym samym czasie w sieci może się poruszać większa liczba pojazdów tej samej linii, w tym samym kierunku oraz w kierunku przeciwnym. Dodatkowo sieć komunikacyjna podzielona jest na strefy. Przykładowa sieć z rys.1 podzielona jest na 3 strefy oznaczone I, II i III.

Dla zdefiniowanej sieci i rozkładu jazdy dany jest przystanek początkowy x oraz przystanek końcowy y ($x \neq y$), pomiędzy którymi chcemy odbyć podróż. Dana jest godzina t_x rozpoczęcia podróży na przystanku x. Należy wyznaczyć linie komunikacyjne oraz przystanki ewentualnych przesiadek minimalizując czas oraz koszt podróży pomiędzy przystankami x oraz y.

Czas podróży wynika z przyjętej trasy, ewentualnych przesiadek oraz czasów oczekiwania na nich na połączenie, a także czasów przejazdu pomiędzy przystankami określonych rozkładem jazdy. Koszt podróży wyznaczany jest następująco. Bilet na pojedynczy przejazd w granicach jednej strefy kosztuje 2.0 jednostki, w granicach dwóch stref 2.3 jednostki, a w granicach trzech lub więcej stref 2.6 jednostki. Tak więc koszt przejazdu z przystanku x do y jest sumą kosztów podróży na poszczególnych odcinkach (jeśli występują przesiadki) z uwzględnieniem możliwych zmian stref na tych odcinkach. Niektóre linie komunikacyjne mogą być oznaczone jako pośpieszne. Przejazd tymi liniami odbywa się szybciej, co uwzględnione jest w rozkładzie jazdy (wielkości δ_0, δ_1, ..., δ_i są odpowiednio mniejsze). Aby wyznaczyć koszt podróży pojazdem linii pośpiesznej, należy pomnożyć przez 2 koszt podróży pojazdem linii zwykłej.

3. Optymalizacja wielokryterialna

3.1. Cel optymalizacji wielokryterialnej

Przez optymalizację rozumie się osiągnięcie pewnego maksymalnego efektu przez różnego rodzaju zamierzone działania. Aby osiągnąć jak najlepsze rozwiązanie, konieczna jest możliwość dowolnego wyboru i zmiany pewnej liczby zmiennych sterujących q_1, q_2, ..., q_k. Idealnym przypadkiem byłoby występowanie tylko jednego kryterium jakości. W większości przypadków jedynie silne ograniczenie stawianych wymagań odnośnie do rozwiązania pozwala zastosować tylko jedno kryterium jakości. Dzięki zastosowaniu jednego kryterium jakości przez silne ograniczenie wymagań możemy uzyskać rozwiązanie za pomocą mniej

skomplikowanych metod, jednak rozwiązanie uzyskane w ten sposób nie uwzględnia wielu aspektów występujących w rzeczywistych problemach, dlatego jest ono mniej dokładne. Aby uzyskać rozwiązanie uwzględniające większą liczbę aspektów występujących w rzeczywistych problemach, stosuje się optymalizację wielokryterialną [1]. W takiej sytuacji stosuje się pewną skończoną liczbę kryteriów jakości $Q_1, Q_2, \ldots, Q_n$, które są zależne od zmiennych sterujących:

$$Q_i = Q_i(q_1, q_2, \ldots, q_k) \qquad i = 1, 2, \ldots, n.$$

Jednym z problemów optymalizacji jest wybór kryteriów jakości, które możliwie dokładnie oceniają różne aspekty badanego zjawiska. W zadaniu optymalizacji mogą wystąpić dwa rodzaje kryteriów: maksymalizowane i minimalizowane. W przypadku kryteriów maksymalizowanych rozwiązanie 1 jest lepsze od rozwiązania 2, jeżeli wartość funkcji kryterialnej rozwiązania 1 jest większa od wartości funkcji kryterialnej rozwiązania 2. Dla kryteriów minimalizowanych rozwiązanie 1 jest lepsze, jeżeli ma mniejszą wartość funkcji kryterialnej niż rozwiązanie 2.

W procesie optymalizacji możemy mieć do czynienia z dwoma przypadkami, w których kryteria jakości są funkcjami ciągłymi zmiennych sterujących lub funkcjami dyskretnymi. Przykładowe zadania, w których występują dyskretne i ciągłe funkcje kryterialne, można znaleźć w pracy [1].

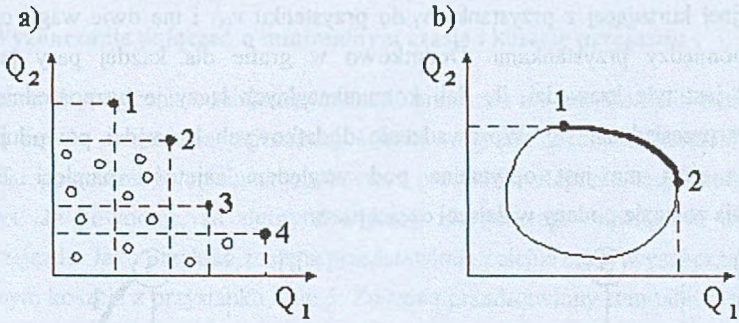
Przez pojęcie optymalizacji wielokryterialnej rozumiemy optymalizację zadanego zbioru kryteriów jakości $Q_1, Q_2, \ldots, Q_n$, które zależą od pewnej skończonej liczby zmiennych sterujących $q_1, q_2, \ldots, q_k$. Z doświadczeń praktycznych wynika, że funkcje jakości tylko w niewielu przypadkach osiągają jednocześnie wartości optymalne. Celem optymalizacji jest określenie, w obszarze wartości zmiennych sterujących, pewnego optymalnego zbioru, zwanego zbiorem rozwiązań niezdominowanych (kompromisów, rozwiązań Pareto-optymalnych [2]). Zbiór rozwiązań niezdominowanych jest podstawą dla wyboru rozwiązania kompromisowego. Po wyznaczeniu zbioru rozwiązań niezdominowanych zadanie optymalizacji wielokryterialnej uważa się za rozwiązane.

3.2. Zbiór rozwiązań niezdominowanych

Wartości kryteriów $Q_1, Q_2, \ldots, Q_n$ można uważać za składowe pewnego n-wymiarowego wektora, zwanego wektorem celu. Obszar, w którym leżą wszystkie wektory celu, nazywamy obszarem celu [1]. Zbiór rozwiązań niezdominowanych jest zbiorem wszystkich wektorów celu, których w ramach dysponowanych środków nie można już dalej równocześnie poprawić. Sformułowanie definicji zbioru rozwiązań niezdominowanych można ująć jako częściowe uporządkowanie wektorów. Jeżeli do porównania wektorów użyć normy, wówczas zagadnienie sprowadza się do porównania wartości skalarnych, tak jak w przypadku

optymalizacji jednokryterialnej. W przypadku optymalizacji wielokryterialnej konieczne jest porównanie poszczególnych składowych wektora. Wyciągnięcie jednoznacznych wniosków możliwe jest jedynie wtedy, gdy wszystkie składowe danego wektora są mniejsze lub większe od odpowiednich składowych wektora porównywanego. Nie można ze sobą porównywać wektorów, dla których pewne składowe są większe, inne natomiast są mniejsze od odpowiednich składowych wektora porównywanego. Z taką sytuacją mamy do czynienia w przypadku zbioru rozwiązań niezdominowanych.

Definicja zbioru rozwiązań niezdominowanych oparta jest na koncepcji rozwiązania niezdominowanego [3], zwanego inaczej rozwiązaniem Pareto-optymalnym [2]. Przyjmijmy, że danych jest n kryteriów. Wszystkie kryteria są kryteriami maksymalizowanymi. Wektor celu $Q' = (Q_1', Q_2', ..., Q_n')$ dominuje wektor $Q = (Q_1, Q_2, ..., Q_n)$, jeżeli $\forall i \in N$, $Q_i' \geq Q_i$ oraz zachodzi nierówność $Q_j' > Q_j$ dla przynajmniej jednej wartości $j \in N$, gdzie $N = \{1, 2, ..., n\}$. Zbiór wszystkich niezdominowanych wektorów celu tworzy zbiór rozwiązań niezdominowanych. Analogiczną definicję można sformułować dla przypadku, w którym wszystkie kryteria są minimalizowane.



Rys. 2. Dyskretny (a) i spójny (b) zbiór rozwiązań niezdominowanych
Fig. 2. Discrete (a) and connected (b) set of non-dominated solutions

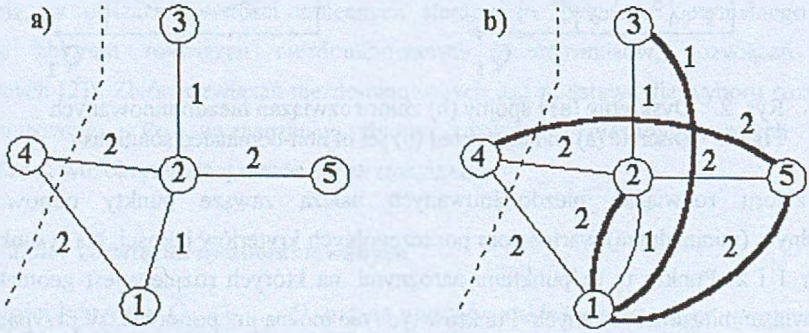
Do zbioru rozwiązań niezdominowanych należą zawsze punkty odpowiadające maksymalnym (minimalnym) wartościom poszczególnych kryteriów jakości. Na rysunku 2b są to punkty 1 i 2. Punkty te są punktami narożnymi, na których rozpięty jest geometrycznie zbiór rozwiązań niezdominowanych. Punktów tych nie można już poprawić. W przypadku gdy funkcje kryterialne są funkcjami ciągłymi, obszar celu jest spójny. W takiej sytuacji zbiór rozwiązań niezdominowanych zawiera punkty brzegowe obszaru celu (rys. 2b). W przypadku gdy funkcje kryterialne są funkcjami dyskretnymi, obszar celu jest obszarem dyskretnym. Zbiór rozwiązań niezdominowanych jest wówczas określany przez porównanie wszystkich wektorów celu zgodnie z warunkiem uporządkowania częściowego (rys 2a).

3.3. Rozwiązanie kompromisowe

Z reguły wyznaczenie zbioru rozwiązań niezdominowanych jest tylko etapem pośrednim na drodze do podjęcia decyzji. Zbiór rozwiązań niezdominowanych ma ułatwić wybór określonego, w podanych warunkach możliwie wszechstronnego, rozwiązania kompromisowego. Wybrane rozwiązanie kompromisowe powinno pochodzić ze zbioru rozwiązań niezdominowanych, który został wyznaczony w procesie optymalizacji wielokryterialnej. Jednak metody matematyczne nie definiują, w jaki sposób należy je wybrać. Przy wyborze decydującą rolę odgrywa wiedza i doświadczenie osoby podejmującej decyzję.

4. Komputerowa reprezentacja sieci komunikacyjnej

Sieć komunikacyjna jest reprezentowana za pomocą skierowanego grafu ważonego [4]. Wierzchołki grafu reprezentują przystanki, a krawędzie grafu – trasy przejazdu pomiędzy przystankami. Graf zawiera krawędzie równoległe. Każda krawędź odpowiada jednej linii komunikacyjnej kursującej z przystanku v_i do przystanku v_{i+1} i ma dwie wagi: czas i koszt przejazdu pomiędzy przystankami. Dodatkowo w grafie dla każdej pary przystanków dodawanych jest tyle krawędzi, ile linii komunikacyjnych łączy je bezpośrednio, tzn. bez konieczności przesiadania się. Wprowadzenie dodatkowych krawędzi powoduje, że taka reprezentacja sieci nie jest optymalna pod względem zajętości pamięci. Powód ich wprowadzenia zostanie podany w dalszej części pracy.



Rys. 3. Sieć komunikacyjna i graf ją reprezentujący
Fig. 3. A communication network and graph its representing

Na rys. 3a przedstawiono przykładową sieć komunikacyjną składającą się z pięciu przystanków, w której kursują pojazdy dwóch linii. Trasa linii 1 prowadzi przez przystanki 1, 2, 3, natomiast linii 2 przez przystanki 1, 4, 2, 5. Każdą krawędź należy traktować jako parę łuków przeciwnie skierowanych. Przy krawędziach zaznaczono numer linii komunikacyjnej,

która łączy parę przystanków. Nie zaznaczono wag krawędzi, ponieważ nie są one istotne dla zrozumienia sposobu reprezentacji sieci za pomocą grafu. Na rys. 3b przedstawiono graf reprezentujący sieć komunikacyjną z rys. 3a. Dodatkowe krawędzie zostały zaznaczone liniami pogrubionymi. Dla linii 1 dodano krawędź łączącą przystanki 1 i 3, natomiast dla linii 2 krawędzie pomiędzy przystankami 1 i 2, 1 i 5, 4 i 5. Wyjaśnienia wymaga dodanie krawędzi pomiędzy przystankami 1 i 2. W grafie istniała już taka krawędź, ale odpowiadała ona linii 1. Dodana krawędź odpowiada linii 2. Są to dwie różne krawędzie.

5. Algorytm wyznaczania połączeń w sieciach komunikacyjnych

Wyznaczenie połączeń w sieciach komunikacyjnych odbywa się w dwóch etapach. W pierwszym etapie wyznaczane są za pomocą algorytmu Dijkstry [5] połączenia o minimalnym czasie i koszcie przejazdu. W drugim etapie wyznaczane są pozostałe połączenia należące do zbioru rozwiązań niezdominowanych.

5.1. Wyznaczanie połączeń o minimalnym czasie i koszcie przejazdu

Wyznaczenie połączenia o minimalnym czasie przejazdu nie jest skomplikowane, ponieważ sprowadza się do znalezienia najkrótszej ścieżki w grafie. Problem komplikuje się przy wyznaczaniu połączenia o minimalnym koszcie przejazdu. Reprezentacja sieci za pomocą grafu z rys. 3a powoduje, że algorytm Dijkstry nie generuje rozwiązania o minimalnym koszcie przejazdu. Jako przykład zostaną przedstawione kolejne etapy wyznaczania połączenia o minimalnym koszcie z przystanku 1 do 5. Zostanie przedstawiony stan tablic: *koszt* (zawiera koszty dojazdu z przystanku początkowego do danego przystanku), *ustalony* (zawiera informację, czy został ustalony koszt dojazdu z przystanku początkowego do danego przystanku), *poprzednik* (zawiera numer przystanku, z którego przyjechano do danego przystanku), *nr_linii* (zawiera numer linii, którą dojechano do danego przystanku).

W czasie inicjacji zostaje ustalony koszt dojazdu do przystanku początkowego, który jest równy 0, natomiast koszt dojazdu do pozostałych przystanków ma wartość tymczasową równą ∞.

v	1	2	3	4	5
koszt (*v*)	0	∞	∞	∞	∞
ustalony (*v*)	true	false	false	false	false
poprzednik (*v*)	-1	-1	-1	-1	-1
nr_linii (*v*)	-1	-1	-1	-1	-1

W pierwszej iteracji zostaje zmieniony koszt dojazdu do przystanków 2 i 4 oraz zostaje ustalony koszt dojazdu do przystanku 2.

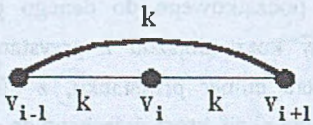
v	1	**2**	3	4	5
koszt (*v*)	0	**2.0**	∞	2.3	∞
ustalony (*v*)	true	**true**	false	false	false
poprzednik (*v*)	-1	**1**	-1	1	-1
nr_linii (*v*)	-1	**1**	-1	2	-1

W drugiej iteracji zmianie ulegają koszty dojazdu do przystanków 3 i 5. Zostaje ustalony koszt dojazdu do przystanku 3.

v	1	2	**3**	4	5
koszt (*v*)	0	2.0	**2.0**	2.3	4.0
ustalony (*v*)	true	true	**true**	false	false
poprzednik (*v*)	-1	1	**2**	1	2
nr_linii (*v*)	-1	1	**1**	2	2

W trzeciej i czwartej iteracji nie ulegają zmianie wartości kosztu dla żadnego z przystanków. Zostaje ustalony koszt dojazdu do przystanku 4 (w iteracji trzeciej) i 5 (w iteracji czwartej).

Z otrzymanego rozwiązania wynika, że minimalny koszt dojazdu z przystanku 1 do 5 wynosi 4.0 jednostki. Z przystanku 1 wyjeżdżamy pojazdem linii 1 i dojeżdżamy do przystanku 2, tam się przesiadamy i dojeżdżamy do przystanku końcowego, tj. przystanku 5 pojazdem linii 2. Znalezione rozwiązanie nie jest rozwiązaniem o minimalnym koszcie. Z rys. 3a można odczytać, że istnieje bezpośrednie połączenie pojazdem linii 2 i jest ono połączeniem o minimalnym koszcie, który jest równy 2.6 jednostki.



Rys. 4. Fragment grafu reprezentującego sieć komunikacyjną
Fig. 4. A part of graph representing the communication network

Aby móc zastosować algorytm Dijkstry do wyznaczenia połączenia o minimalnym koszcie, konieczne jest wprowadzenie dodatkowych krawędzi w grafie i zastosowanie reprezentacji sieci komunikacyjnej z rys. 3b. Drobnej modyfikacji ulega sposób wyboru linii komunikacyjnej, którą pojedzie się z przystanku v_i do v_{i+1}. W poprzednim przypadku rozpatrywało się wszystkie linie łączące przystanki v_i i v_{i+1}. W tym przypadku pomija się linię, którą dojechano do v_i. Zostanie to uzasadnione na przykładzie (rys. 4). Po ustaleniu czasu dojazdu do przystanku v_{i-1} próbuje się poprawić koszty dojazdu dla jego sąsiadów, czyli przystanków v_i i

v_{i+1}. Następnie ustalany jest koszt dojazdu do przystanku v_i i próbuje się poprawić koszt dojazdu do przystanku v_{i+1}. Linii k nie trzeba uwzględniać, ponieważ została ona już uwzględniona w momencie poprawiania kosztów dla sąsiadów przystanku v_{i+1}.

Po wprowadzeniu do grafu dodatkowych krawędzi (rys. 3b) można wyznaczyć za pomocą algorytmu Dijkstry połączenie o minimalnym koszcie przejazdu.

Rozpatrzmy, w jaki sposób zostanie wyznaczone połączenie o minimalnym koszcie przejazdu po wprowadzeniu do grafu dodatkowych krawędzi (rys. 3b). W czasie inicjacji, podobnie jak w poprzednim przypadku, zostaje ustalony koszt dojazdu do przystanku początkowego, który jest równy 0, natomiast koszt dojazdu do pozostałych przystanków ma wartość tymczasową równą ∞. W pierwszej iteracji zmieniane są koszty dojazdu do przystanków 2, 3, 4 i 5. Przystanki 2 i 3 mają najmniejszą tymczasową wartość kosztu równą 2.0 jednostki. W takiej sytuacji zostaje ustalony koszt dojazdu dla przystanku, który ma najmniejszy czas dojazdu. Aby dojechać do przystanku 3, trzeba najpierw dojechać do przystanku 2, więc czas dojazdu do przystanku 3 jest większy niż czas dojazdu do przystanku 2. Zostaje ustalony koszt dojazdu do przystanku 2.

v	1	2	3	4	5
koszt (v)	0	**2.0**	2.0	2.3	2.6
ustalony (v)	true	**true**	false	false	false
poprzednik (v)	-1	**1**	1	1	1
nr_linii (v)	-1	**1**	1	2	2

W kolejnych iteracjach nie ulegają zmianie koszty dojazdu do żadnego z przystanków. W iteracji drugiej zostaje ustalony koszt dojazdu do przystanku 3, w trzeciej iteracji – do przystanku 4, a w czwartej iteracji – koszt dojazdu do przystanku 5. Z otrzymanego rozwiązania wynika, że minimalny koszt dojazdu z przystanku 1 do 5 wynosi 2.6 jednostki, co jest prawidłowym rozwiązaniem. Jedyną wadą tego rozwiązania jest niekompletna trasa przejazdu. Jednak otrzymanie pełnej trasy przejazdu nie jest trudne, ponieważ znane są trasy przejazdu dla każdej linii komunikacyjnej.

5.2. Wyznaczanie pozostałych połączeń należących do zbioru rozwiązań niezdominowanych

Po wyznaczeniu połączeń o minimalnym czasie i koszcie przejazdu (wiersze 1-7 pseudokodu algorytmu) wyznaczane są pozostałe połączenia należące do zbioru rozwiązań niezdominowanych. W przypadku gdy połączenie o minimalnym czasie jest także połączeniem o minimalnym koszcie przejazdu, wyszukiwanie nie jest przerywane, ponieważ może istnieć wiele takich rozwiązań. Będą się one różniły trasą przejazdu lub liniami komunikacyjnymi,

którymi należy przejechać trasę. Wszystkie te połączenia należą do zbioru rozwiązań niezdominowanych.

Pseudokod algorytmu wyznaczania połączeń w sieciach komunikacyjnych:

```
1:  połączenia := ∅;                    // zbiór niezdominowanych połączeń
2:  min_czas := MinCzas();              // wyznacz połączenie o minimalnym czasie przejazdu
3:  połączenia := połączenia ∪ {min_czas};
4:  min_koszt := MinKoszt();            // wyznacz połączenie o minimalnym koszcie przejazdu
5:  if min_czas ≠ min_koszt then
6:     połączenia := połączenia ∪ {min_koszt};
7:  end if
    // wyznacz pozostałe połączenia należące do zbioru rozwiązań niezdominowanych
8:  trasa := v_start;                   // dołącz do trasy przystanek początkowy
9:  koniec := false;
10: while not koniec do
11:    znajdź nowy przystanek v_i do umieszczenia w trasie;
12:    if istnieje v_i then
13:       trasa := trasa ∪ {v_i};
14:       CzasKoszt(trasa);             // wyznacz czas i koszt dojazdu z v_start do v_i
15:       if rozwiązanie niezdominowane then
16:          if v_i = v_koniec then     // dołączono do trasy przystanek końcowy
17:             połączenia := połączenia ∪ {trasa};
18:             UsuńZdominowane();     // usuń rozwiązania (połączenia) zdominowane
19:             powrót;
20:          end if
21:       else                          // rozwiązanie (połączenie) zdominowane, pomiń je
22:          powrót;
23:       end if
24:    else                             // nie istnieje v_i
25:       powrót;
26:       if powrót z v_start then      // powrót z przystanku początkowego, koniec poszukiwań
27:          koniec := true;
28:       end if
29:    end if
30: end while
```

Do wyznaczania połączeń stosowana jest metoda powrotów [6]. Główna idea tej metody polega na stopniowym konstruowaniu rozwiązania. Ogólnie, mając dane rozwiązanie

częściowe (v_1, ..., v_i), próbuje się znaleźć dopuszczalny wierzchołek (przystanek) v_{i+1}, co do którego nie można od razu wykluczyć, że (v_1, ..., v_i, v_{i+1}) daje się rozszerzyć do pewnego rozwiązania (lub już nim jest). Jeżeli taki dopuszczalny, niewykorzystany jeszcze wierzchołek v_{i+1} istnieje, to dodawany jest on do rozwiązania częściowego (wiersz 13). Po dołączeniu nowego wierzchołka (przystanku) do rozwiązania częściowego wylicza się czas i koszt dojazdu z przystanku początkowego do przystanku, który został dołączony (wiersz 14). Jeżeli po wyliczeniu czasu i kosztu dojazdu okaże się, że rozwiązanie częściowe jest zdominowane, to nie jest ono dalej rozpatrywane i następuje powrót (wiersz 21). W przeciwnym przypadku kontynuuje się proces rozszerzania rozwiązania dla ciągu (v_1, ..., v_i, v_{i+1}). Jeżeli dołączony przystanek v_{i+1} jest przystankiem końcowym, to rozwiązanie jest kopiowane do zbioru dotychczas znalezionych rozwiązań (wiersz 17), następuje usunięcie rozwiązań zdominowanych oraz powrót do rozwiązania częściowego (v_1, ..., v_i).

Jeżeli nie istnieje przystanek v_{i+1}, który mógłby być dołączony do rozwiązania częściowego, to następuje powrót (wiersz 25) do rozwiązania (v_1, ..., v_{i-1}) i kontynuuje się poszukiwania nowego, jeszcze nie wykorzystanego wierzchołka v'_i. Jeżeli powrót do rozwiązania (v_1, ..., v_{i-1}) nie daje możliwości znalezienia wierzchołka v'_i, wówczas powraca się jeszcze raz (do rozwiązania częściowego (v_1, ..., v_{i-2})) i próbuje znaleźć nowy wierzchołek v'_{i-1} itd. Poszukiwania są przerywane przy próbie powrotu z wierzchołka startowego v_1 (wiersz 26).

Mówiąc dokładniej, zakłada się, że dla każdego $k > 1$ jest dany pewien ustalony zbiór S_k, z którego wybiera się kandydatów na k-tą pozycję rozwiązania częściowego. W rozpatrywanym przypadku zbiór S_k stanowi lista następników wierzchołka v_{k-1} z wykluczeniem tych, których dodanie na k-tą pozycję spowodowałoby, że rozwiązanie częściowe stanowiłoby cykl [4].

5.3. Złożoność czasowa algorytmu

Istotą algorytmu jest wyznaczenie w pierwszym etapie połączeń o minimalnym czasie i koszcie przejazdu, a następnie wyznaczenie w drugim etapie pozostałych połączeń należących do zbioru rozwiązań niezdominowanych.

Do wyznaczenia połączeń o minimalnym czasie i koszcie przejazdu zastosowano algorytm Dijkstry, który ma złożoność czasową $O(n^2)$, gdzie n jest równe liczbie przystanków w sieci komunikacyjnej.

W drugim etapie są wyznaczane za pomocą metody powrotów pozostałe połączenia należące do zbioru rozwiązań niezdominowanych. Etap ten ma złożoność wykładniczą względem liczby przystanków w sieci komunikacyjnej (n) [4]. Pojedyncze rozwiązanie może zawierać maksymalnie n przystanków. Zatem, aby wyznaczyć pozostałe połączenia należące

do zbioru rozwiązań niezdominowanych, konieczne jest przeszukanie w grafie reprezentującym sieć

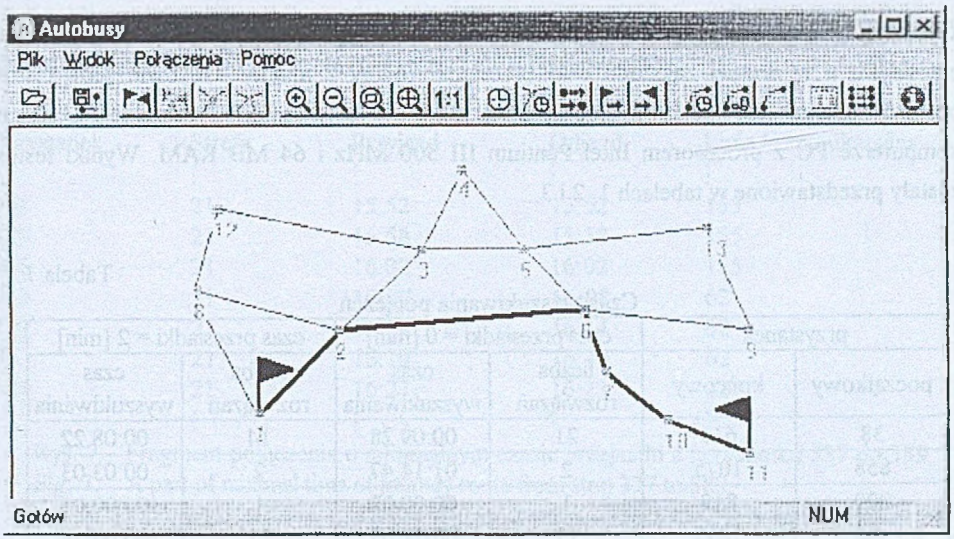
$$\prod_{i=1}^{n} |v_i|$$

wierzchołków, gdzie $|v_i|$ jest liczbą kandydatów na i-ty przystanek w trasie. W kroku 11 algorytmu wybierany jest przystanek v_i, który zostanie wstawiony do rozwiązania na pozycję i-tą. Liczba kandydatów na i-ty przystanek w rozwiązaniu jest równa stopniowi wyjściowemu [7] wierzchołka (reprezentującego przystanek w grafie) znajdującego się na pozycji i-1 rozwiązania. Maksymalna wartość stopnia wyjściowego wierzchołka jest równa liczbie linii w sieci komunikacyjnej (k). Zatem złożoność czasowa jest równa $O(k^n)$.

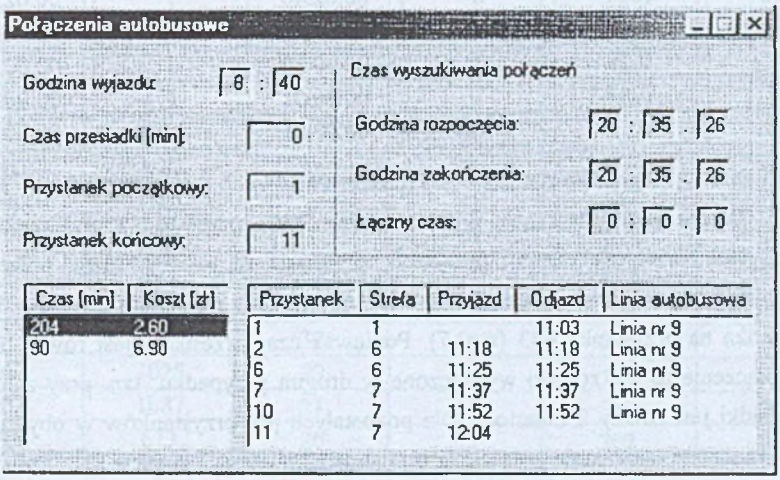
O złożoności czasowej algorytmu decyduje drugi etap wyznaczania połączeń o złożoności wykładniczej. Stosowanie algorytmu o złożoności wykładniczej jest możliwe dla danych o małym rozmiarze. W przypadku danych o dużym rozmiarze należy się spodziewać, że czas wyszukiwania mimo pomijania rozwiązań zdominowanych na etapie konstruowania pojedynczego rozwiązania może być duży. Obawy te zostały potwierdzone podczas przeprowadzania badań eksperymentalnych dla niektórych danych testowych.

6. Implementacja algorytmu

Algorytm opisany w punkcie 5 został zaimplementowany w programie, który został napisany w języku C++ za pomocą środowiska Visual C++ 5.0 oraz biblioteki MFC (Microsoft Foundation Class) będących składnikiem pakietu Visual Studio 97. W programie jako przykładową sieć komunikacyjną zastosowano sieć autobusową. Program umożliwia wczytanie z pliku dowolnej sieci autobusowej, która jest wyświetlana w głównym oknie programu (rys. 5) oraz przeprowadzenie badań eksperymentalnych dla tej sieci, tj. wyznaczenie połączeń stanowiących zbiór rozwiązań niezdominowanych dla zadanego przystanku początkowego i końcowego oraz zadanej godziny rozpoczęcia podróży. Dodatkowo można ustalić czas przesiadki, który domyślnie jest pomijany. Wyniki eksperymentu są prezentowane w oddzielnym oknie (rys. 6) oraz zaznaczana jest w sieci autobusowej znajdującej się w głównym oknie programu trasa przejazdu dla każdego wyznaczonego połączenia. Program umożliwia zapisanie do pliku wyników badań eksperymentalnych. Dodatkowo wyświetlana jest godzina rozpoczęcia i zakończenia eksperymentu oraz łączny czas trwania eksperymentu.



Rys. 5. Główne okno programu
Fig. 5. The application's main window



Rys. 6. Prezentacja wyników eksperymentu
Fig. 6. Presentation results of test

7. Wyniki badań eksperymentalnych

Badania zostały przeprowadzone dla dużej sieci składającej się z 1211 przystanków, w której kursuje 500 linii komunikacyjnych, która jest podzielona na 26 stref. Była to sieć stworzona na potrzeby testów i nie reprezentowała ona żadnej z rzeczywistych sieci

komunikacyjnych. Przeprowadzono dwie serie testów: w pierwszej nie uwzględniono czasu przesiadki, a w drugiej założono czas przesiadki równy 2 minuty. Obydwie serie testów przeprowadzono dla godziny początkowej 12:00. Testy zostały przeprowadzone na komputerze PC z procesorem Intel Pentium III 500 MHz i 64 MB RAM. Wyniki testów zostały przedstawione w tabelach 1, 2 i 3.

Tabela 1

Czas wyszukiwania połączeń

przystanek		czas przesiadki = 0 [min]		czas przesiadki = 2 [min]	
początkowy	końcowy	liczba rozwiązań	czas wyszukiwania	liczba rozwiązań	czas wyszukiwania
38	611	21	00:09.28	14	00:08.22
858	1075	2	01:14.47	2	00:03.03
900	819	1	00:00.02	1	00:00.01
62	1095	11	00:11.03	11	00:08.38
997	484	9	00:02.47	4	00:02.44
38	827	1	00:00.01	1	00:00.00
1202	1004	8	00:02.46	7	00:02.26
337	389	31	00:09.49	17	00:06.29
147	1028	3	00:00.35	3	00:00.24
667	1087	1	00:00.01	1	00:00.00
648	613	19	00:00.01	19	00:00.04

Wyjaśnienia wymagają wyznaczone połączenia dla dwóch par przystanków: 337, 389 i 858, 1075. Wyznaczone połączenia o minimalnym czasie przejazdu z przystanku 337 do przystanku 389 różnią się czasem przejazdu, ponieważ są to dwa różne połączenia. W wyznaczonym połączeniu dla przypadku, w którym pomijany jest czas przesiadki, konieczna jest przesiadka na przystanku 443 (rys. 7). Ponieważ czas przesiadki jest równy 1 minucie, dlatego połączenie to nie zostało wyznaczone w drugim przypadku, tzn. przy założeniu, że czas przesiadki jest równy 2 minutom. Dla pozostałych par przystanków w obydwu seriach zostały wyznaczone takie same połączenia o minimalnym czasie i koszcie przejazdu. Dla pary przystanków 858, 1075 komentarza wymaga czas wyznaczania wszystkich połączeń. W przypadku gdy nie jest uwzględniany czas przesiadki, czas wyszukiwania jest prawie 25 razy większy niż w przypadku, gdy czas przesiadki wynosi 2 minuty. Różnica czasu wynika z liczby rozwiązań częściowych, które są rozpatrywane przy wyznaczaniu za pomocą metody powrotów pozostałych połączeń należących do zbioru rozwiązań niezdominowanych. W przypadku gdy nie jest uwzględniany czas przesiadki, rozpatrywanych jest 175,252,214 rozwiązań częściowych. Natomiast przy założeniu, że czas przesiadki jest równy 2 minuty, rozpatrywanych jest 7,239,920 rozwiązań częściowych. W drugim przypadku jest

rozpatrywana mniejsza liczba rozwiązań częściowych. Pozostałe rozwiązania są pomijane w 21. kroku algorytmu, ponieważ są rozwiązaniami zdominowanymi.

Przystanek	Strefa	Przyjazd	Odjazd	Linia komunikacyjna
⋮	⋮	⋮	⋮	⋮
581	21	15:52	15:52	155
449	21	15:58	15:58	155
445	21	16:02	16:02	155
443	21	**16:07**	**16:08**	62
442	21	16:14	16:14	62
94	21	16:19	16:19	62
424	21	16:23	16:23	62
⋮	⋮	⋮	⋮	⋮

Rys. 7. Fragment połączenia o minimalnym czasie przejazdu z przystanku 337 do 389
Fig. 7. A part of minimal time of journey route from stop 337 to 389

Tabela 2

Wyznaczone połączenia bez uwzględniania czasu przesiadki

przystanek		długość trasy[1]		minimalny	
początkowy	końcowy	minimalna	maksymalna	czas przejazdu [min]	koszt przejazdu [jednostka]
38	611	34	44	334	9.8
858	1075	27	30	279	11.8
900	819	28	28	234	7.5
62	1095	28	32	192	11.2
997	484	32	39	253	9.5
38	827	23	23	219	6.6
1202	1004	27	39	174	10.9
337	389	32	40	296	11.2
147	1028	29	36	192	11.5
667	1087	21	21	156	7.2
648	613	26	35	202	9.2

[1] Przez długość trasy należy rozumieć liczbę przystanków tworzących trasę przejazdu z przystanku początkowego do końcowego.

Tabela 3

Wyznaczone połączenia dla czasu przesiadki 2 minuty

przystanek		długość trasy[1]		minimalny	
początkowy	końcowy	minimalna	maksymalna	czas przejazdu [min]	koszt przejazdu [jednostka]
38	611	35	44	234	9.8
858	1075	27	30	279	11.8
900	819	28	28	234	7.5
62	1095	28	32	192	11.2
997	484	34	40	253	9.5
38	827	23	23	219	6.6
1202	1004	27	34	174	10.9
337	389	32	40	309	11.2
147	1028	29	36	192	11.5
667	1087	21	21	156	7.2
648	613	26	35	202	9.2

8. Podsumowanie

W niniejszej pracy[1] dla problemu wyznaczania połączeń w sieciach komunikacyjnych został przedstawiony algorytm wyznaczania zbioru rozwiązań niezdominowanych. Zbiór rozwiązań niezdominowanych jest wyznaczany w dwóch etapach. W pierwszym etapie wyznaczane są za pomocą algorytmu Dijkstry połączenia o minimalnym czasie i koszcie przejazdu. W drugim etapie za pomocą metody powrotów wyznaczane są pozostałe połączenia należące do zbioru rozwiązań niezdominowanych. Dzięki wyznaczaniu na bieżąco czasu i kosztu dojazdu do danego przystanku możliwe jest odrzucanie rozwiązań zdominowanych w trakcie ich konstruowania. Powoduje to skrócenie czasu wyszukiwania, ponieważ nie są wyznaczane wszystkie możliwe połączenia z przystanku początkowego do końcowego. Mimo to algorytm ma wykładniczą złożoność czasową.

Wyszukiwanie połączeń za pomocą przedstawionego algorytmu jest możliwe dla małych sieci komunikacyjnych. W przypadku większych sieci, jak np. sieć, która została użyta do badań eksperymentalnych, czas wyszukiwania jest duży, co potwierdziły wyniki przeprowadzonych testów.

Na czas wyznaczania zbioru rozwiązań niezdominowanych decydujący wpływ ma drugi etap algorytmu, który ma złożoność wykładniczą. Dzięki zastosowaniu algorytmu Dijkstry,

[1] Objaśnienie jak w tabeli 2.

który ma złożoność kwadratową, wyznaczenie połączeń o minimalnym czasie i koszcie przejazdu odbywa się w krótkim czasie, który nie przekracza 1 sekundy.

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, p. 1-52, USA wrzesień 1979.
3. Górecki H.: Optymalizacja systemów dynamicznych. Wydawnictwo Naukowe PWN, Warszawa 1993.
4. Reingold E., Nievergelt J., Deo N.: Algorytmy kombinatoryczne. PWN, Warszawa 1985.
5. Sysło M., Deo N., Kowalik J.: Algorytmy optymalizacji dyskretnej z programami w języku Pascal. Wydawnictwo Naukowe PWN, Warszawa 1995.
6. Lipski W.: Kombinatoryka dla programistów. WNT, Warszawa 1989
7. Cormen T., Leiserson Ch., Rivest R.: Wprowadzenie do algorytmów. WNT, Warszawa 2000

Recenzent: Dr Urszula Boryczka

Wpłynęło do Redakcji 4 kwietnia 2001 r.

Abstract

The communication networks routing problem (CNRP) is an example of multicriteria optimization problem (MOP). The paper presents the CNRP description and the fundamental informations about MOP. The solution of MOP is the set of non-dominated solutions. We present the algorithm to find the routes belonging to the set of non-dominated solutions.

The presented algorithm was tested on the communication network which included 1211 stops and 500 communiaction lines. This communiation network did not describe any from the real communication-systems. It was create only for tests.

communication-system like that which has been used for testing. To find the minimal time or cost of travel the route we used Dijkstra's shortest-path algorithm. The complexity of this algorithm is $O(n^2)$, where n is the number of stops, and we can use it for large communication-systems, like the communication-systems which we used for tests. The time of finding the solutions wasn't greater than 1 second.