STUDIA INFORMATICA 2003
Volume 24 Number 4 (56)

Jacek WIDUCH
Politechnika Śląska, Instytut Informatyki

PROBLEM WYZNACZANIA OPTYMALNEJ TRASY TRANSPORTU PRZESYŁEK[1]

Streszczenie. W pracy przedstawiono algorytm rozwiązywania problemu wyznaczania optymalnej trasy transportu przesyłek. Podstawą do opracowania algorytmu był algorytm rozwiązujący dwukryterialny problem najkrótszych ścieżek w grafie ważonym o stałych wagach. Problem będący tematem pracy jest przykładem optymalizacji dwukryterialnej. Rozwiązaniem problemu optymalizacji dwukryterialnej jest zbiór rozwiązań niezdominowanych. Przedstawiony algorytm umożliwia wyznaczenie wszystkich tras należących do zbioru rozwiązań niezdominowanych. Zaprezentowano przykładowe wyniki działania algorytmu.

Słowa kluczowe: problem transportowy, optymalizacja wielokryterialna, zbiór rozwiązań niezdominowanych, dwukryterialny problem najkrótszej ścieżki

SOLVING THE OPTIMAL ROUTS OF PACKAGES TRANSPORTATION PROBLEM

Summary. The paper describes an algorithm for solving the problem of determining the optimal route of packages. The algorithm is based on the algorithm for solving the bicriterion shortest path problem in a weighted graph with constant weights. The problem considered in this work is an example of bicriteria optimization. The solution to a bicriteria optimization problem is the set of non-dominated solutions. The algorithm which determines all routes which belong to the set of non-dominated solutions is shown. The result of the tests are presented.

Keywords: transportation problem, multicriteria optimization, set of non-dominated solutions, bicriterion shortest path problem

[1] Publikacja powstała w wyniku wykonania prac w ramach badań BK-279/RAu2/2002.

1. Wstęp

W ostatnich latach można zaobserwować w naszym kraju rozwój różnego rodzaju firm spedycyjnych. Firmy te oferują dostarczenie przesyłki do dowolnego miejsca w kraju, a także do miejsc za granicą. Firma spedycyjna składa się z sieci wyodrębnionych jednostek zwanych węzłami sortującymi, których zadaniem jest zbieranie przesyłek z określonego obszaru, sortowanie ich, przeładunek i załadunek przesyłek na środki transportu. Pomiędzy poszczególnymi węzłami utrzymywane są stałe połączenia komunikacyjne umożliwiające transport przesyłek.

Z transportem przesyłek związanych jest kilka problemów. Przesyłka powinna dotrzeć do miejsca przeznaczenia w możliwie jak najkrótszym czasie. Ważne jest także; aby odbyło się to przy minimalnym nakładzie kosztów. W związku z tym konieczne jest zaplanowanie trasy transportu przesyłki, tj. sekwencji węzłów sortujących, pomiędzy którymi powinna być przesłana. Innym problemem jest ustalenie optymalnego rozkładu jazdy oraz minimalizacji środków transportu użytych do przewozu przesyłek.

W pracy [1] autor opisuje jedną z firm spedycyjnych działających na terenie naszego kraju. Autor omawia problem doboru trasy transportu przesyłki oraz ustalania rozkładu jazdy pojazdów transportowych. Problemy są rozpatrywane w sposób teoretyczny bez przedstawienia konkretnych algorytmów. W niniejszej pracy zostanie zaprezentowany algorytm rozwiązujący problem doboru optymalnej trasy transportu przesyłek. Z tego powodu niniejszą pracę można traktować jako rozszerzenie pracy [1].

2. Sformułowanie problemu

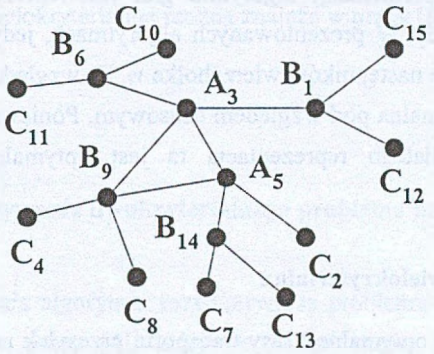
Dana jest sieć transportowa składająca się z N węzłów sortujących ponumerowanych od 1 do N (rys. 1). Pomiędzy poszczególnymi węzłami utrzymywane jest stałe połączenie komunikacyjne umożliwiające transport przesyłek. Węzły sortujące podzielone są na klasy A, B i C. Węzły z poszczególnych klas różnią się między sobą funkcjami. W sieci znajduje się N_A węzłów klasy A, N_B węzłów klasy B oraz N_C węzłów klasy C, gdzie $N_A \neq 0$, $N_A + N_B + N_C = N$[1].

W węzłach klasy A odbywa się sortowanie zasadnicze. Każda przesyłka musi przejść przez co najmniej jeden węzeł klasy A. Każdy węzeł klasy A ma połączenie komunikacyjne z wszystkimi pozostałymi węzłami klasy A oraz z pewną liczbą węzłów klasy B i C.

[1] W rzeczywistych sieciach transportowych często zachodzi $N_A < N_B < N_C$.

W węzłach klasy *B* odbywa się sortowanie wtórne, polegające na przeładunku kontenerów z przesyłkami pomiędzy różnymi środkami transportu. Koszt sortowania w tych węzłach jest znacznie niższy od kosztu sortowania w węzłach klasy *A*. Każdy węzeł klasy *B* ma połączenie komunikacyjne z co najmniej jednym węzłem klasy *A* i co najmniej dwoma węzłami klasy *C*, nie ma natomiast połączeń z innymi węzłami klasy *B*.

Zadaniem węzłów klasy *C* jest zbieranie przesyłek z określonego obszaru i rozwożenie ich do odbiorców. Funkcję tę spełniają także węzły klasy *A* i *B*. W węzłach klasy *C* nie odbywa się sortowanie. Każdy węzeł klasy *C* ma tylko jedno połączenie z węzłem klasy *A* lub *B*.



Rys. 1. Przykładowa sieć transportowa ($N = 15$)
Fig. 1. A sample transportation network ($N = 15$)

Dla tak zdefiniowanej sieci transportowej dany jest węzeł źródłowy v_p i węzeł docelowy v_k, pomiędzy którymi ma być przesłana przesyłka. Należy wyznaczyć trasę transportu przesyłki tak, aby czas i koszt transportu były minimalne. Czas transportu jest sumą czasów przejazdu pomiędzy węzłami sortującymi oraz czasów sortowania, przeładunku i załadunku w węzłach. Natomiast koszt transportu jest sumą kosztów przejazdu pomiędzy węzłami oraz kosztów sortowania, przeładunku i załadunku w węzłach. W rzeczywistej sieci transportowej do czasu i kosztu transportu należy także doliczyć czas i koszt dowozu przesyłki od nadawcy do węzła źródłowego v_p oraz czas i koszt dostarczenia przesyłki z węzła docelowego v_k do odbiorcy. Wartości te nie są stałe i zależą od rodzaju przesyłki, odległości pomiędzy nadawcą i węzłem źródłowym oraz odległości pomiędzy węzłem docelowym i odbiorcą.

3. Podstawy teoretyczne

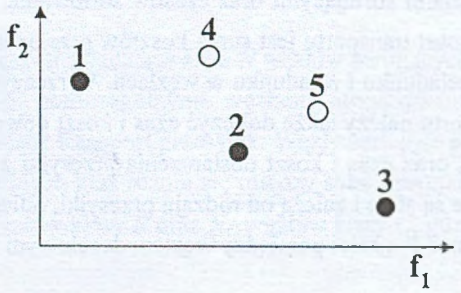
3.1. Reprezentacja sieci transportowej

Sieć transportowa reprezentowana jest za pomocą skierowanego grafu ważonego [2]. Wierzchołki grafu reprezentują węzły sortujące a krawędzie – połączenia komunikacyjne pomiędzy węzłami. Ponieważ przejazdy mogą się odbywać w obu kierunkach, każdą krawędź można uważać za dwa łuki przeciwnie skierowane (rys. 1). Każda krawędź ma dwie wagi: czas i koszt przejazdu pomiędzy węzłami. Graf nie zawiera krawędzi równoległych [2].

W komputerowej implementacji algorytmu graf jest reprezentowany za pomocą list sąsiedztwa (incydencji) [3]. W prezentowanych algorytmach, jedyną operacją wykonywaną na grafie jest wyznaczenie następników wierzchołka v_i. Ze względu na wykonywaną operację jest to reprezentacja optymalna pod względem czasowym. Ponieważ graf reprezentujący sieć jest grafem rzadkim, dlatego reprezentacja ta jest optymalna także pod względem pamięciowym.

3.2. Optymalizacja wielokryterialna

Problem wyznaczania optymalnej trasy transportu przesyłek należy do klasy problemów optymalizacji wielokryterialnej, a dokładniej optymalizacji dwukryterialnej [8]. W problemie będącym tematem niniejszej pracy o wyborze rozwiązania decydują jednocześnie dwa kryteria: czas i koszt przejazdu. Obydwa kryteria są kryteriami minimalizowanymi [8]. Niech będą dane dwa rozwiązania X, Y oraz funkcja kryterialna f. Jeżeli funkcja f jest kryterium minimalizowanym, to rozwiązanie X jest lepsze od rozwiązania Y, jeżeli zachodzi $f(X) < f(Y)$.



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

Rozwiązaniem zadania optymalizacji wielokryterialnej jest zbiór rozwiązań niezdominowanych (zbiór kompromisów, zbiór rozwiązań Pareto-optymalnych).

Rozwiązanie X jest rozwiązaniem niezdominowanym, jeżeli nie istnieje rozwiązanie Y, które je dominuje. Niech będzie danych n minimalizowanych funkcji kryterialnych $f_1, f_2, ..., f_n$. Mówimy, że rozwiązanie Y dominuje rozwiązanie X, jeżeli: $\forall i \in \{1, 2, ..., n\}$ $f_i(Y) \leq f_i(X)$ oraz $f_j(Y) < f_j(X)$ dla przynajmniej jednej wartości $j \in \{1, 2, ..., n\}$.

Niech będą dane dwie funkcje kryterialne f_1 i f_2 (rys. 2). Obydwa kryteria są kryteriami minimalizowanymi. Zbiór wszystkich rozwiązań składa się z pięciu rozwiązań ponumerowanych od 1 do 5. Jednak tylko rozwiązania 1, 2 i 3 należą do zbioru rozwiązań niezdominowanych. Rozwiązanie 4 jest zdominowane przez rozwiązanie 1, a rozwiązanie 2 z kolei dominuje rozwiązanie 5. Więcej przykładów zbiorów rozwiązań niezdominowanych oraz zadań optymalizacji wielokryterialnej można znaleźć w pracy [8].

4. Algorytmy

4.1. Algorytm rozwiązywania dwukryterialnego problemu najkrótszej ścieżki w grafie ważonym

Podstawą do opracowania algorytmu rozwiązywania problemu transportu przesyłek był algorytm przedstawiony w pracy [4], który w dalszej części pracy będzie nazywany algorytmem BRUM, od nazwiska jednego z autorów. Algorytm BRUM rozwiązuje dwukryterialny problem najkrótszej ścieżki (ang. bicriterion shortest-path problem) w grafie ważonym [5] z ustalonego węzła źródłowego v_p do wszystkich pozostałych węzłów grafu. Każdej krawędzi grafu łączącej węzły v_i i v_j przyporządkowane są dwie wagi: czas i koszt przejazdu. Dla każdego węzła v_i pamiętany jest zbiór rozwiązań niezdominowanych $R(v_i)$ [6]. Rozwiązania ze zbioru $R(v_i)$ składają się z pary etykiet (t_k^{vi}, c_k^{vi}) określających, odpowiednio, czas i koszt przejazdu z węzła źródłowego v_p do węzła v_i, natomiast nie zawierają trasy dojazdu z v_p do v_i. Rozwiązania ze zbioru $R(v_i)$ są różne między sobą i są uporządkowane rosnąco względem kosztu przejazdu. Ponieważ są one rozwiązaniami niezdominowanymi, dlatego ich uporządkowanie rosnąco względem kosztu przejazdu powoduje automatyczne uporządkowanie malejąco względem czasu przejazdu. Algorytm BRUM można opisać za pomocą następującego pseudokodu [4]:

```
1: for v_i ∈ V do                    {inicjalizacja}
2:    R(v_i) := ∅;
3: end for
4: R(v_p) := {(0, 0)};
5: kolejka := {v_p};                 {umieść w kolejce węzeł źródłowy}
```

```
 6: while kolejka ≠ ∅ do
 7:   pobierz i usuń z kolejki węzeł v_i;
 8:   for all v_j ∈ adj[v_i] do
 9:     nowy := scal(R(v_j), R(v_i) ⊕ krawędź(v_i, v_j));   {utwórz nowy zbiór rozwiązań}
10:     if nowy ≠ R(v_j) then   {znaleziono nowy zbiór rozwiązań dla węzła v_j}
11:       R(v_j) := nowy;   {zapamiętaj nowy zbiór rozwiązań}
12:       if v_j ∉ kolejka then   {wstaw do kolejki węzeł v_j}
13:         kolejka := kolejka + {v_j};
14:       end if
15:     end if
16:   end for
17: end while
```

Do wyznaczenia rozwiązań została użyta metoda przeszukiwania grafu wszerz [7]. Węzły, dla których został wyznaczony nowy zbiór rozwiązań, przechowywane są w kolejce FIFO [7]. W części inicjalizacyjnej algorytmu w kolejce umieszczany jest węzeł źródłowy v_p (krok 5) oraz każdemu węzłowi v_i przyporządkowany jest pusty zbiór rozwiązań (krok 2) z wyjątkiem węzła źródłowego. W zbiorze rozwiązań $R(v_p)$ węzła źródłowego umieszczane jest rozwiązanie o koszcie i czasie przejazdu równym 0 (krok 4). Rozwiązania dla pozostałych węzłów wyznaczane są w kolejnych iteracjach, które są wykonywane do momentu, aż kolejka będzie pusta (kroki 6-17). W pojedynczej iteracji pobierany jest i usuwany z kolejki pierwszy węzeł v_i (krok 7), a następnie odwiedzane są wszystkie węzły v_j, do których prowadzą krawędzie z v_i (krok 8). Dla każdego odwiedzanego węzła v_j wyznaczany jest nowy zbiór rozwiązań (krok 9). Jeżeli nowy zbiór rozwiązań jest różny od poprzedniego, to jest on zapamiętywany (krok 11) oraz węzeł v_j jest wstawiany do kolejki, jeżeli się w niej nie znajduje (krok 13).

Wyjaśnienia wymagają funkcje *krawędź* i *scal* oraz operacja $\oplus$, które zostały użyte w algorytmie. Funkcja *krawędź*(v_i, v_j) jako wynik zwraca parę wartości (T_{ij}, C_{ij}) będących wagami krawędzi prowadzącej z węzła v_i do v_j i określających odpowiednio czas i koszt przejazdu. Operacja $\oplus$ jest zdefiniowana w następujący sposób:

$$X \oplus (T_{ij}, C_{ij}) = \{(t_1^x + T_{ij}, c_1^x + C_{ij}), (t_2^x + T_{ij}, c_2^x + C_{ij}), ..., (t_k^x + T_{ij}, c_k^x + C_{ij})\},$$

gdzie:

$$X = \{(t_1^x, c_1^x), (t_2^x, c_2^x), ..., (t_k^x, c_k^x)\}$$

jest zbiorem rozwiązań niezdominowanych. Ponieważ wartości (T_{ij}, C_{ij}) są stałe, a zbiór X zawiera rozwiązania niezdominowane, dlatego zbiór uzyskiwany w wyniku wykonania powyższej operacji także zawiera rozwiązania niezdominowane.

Funkcja *scal*(*X*, *Y*), której argumentami są zbiory rozwiązań niezdominowanych, zwraca jako wynik zbiór będący sumą zbiorów *X* i *Y* oraz zawierający wyłącznie rozwiązania niezdominowane. Zbiór zwracany jako wynik nie zawiera takich samych rozwiązań, są one usuwane na etapie scalania. Funkcja *scal* jest opisana następującym pseudokodem [4]:

```
scal(X, Y):
    X = {x1, ..., xp} – zbiór rozwiązań; elementy indeksowane przez i
    Y = {y1, ..., yq} – zbiór rozwiązań; elementy indeksowane przez j
 1: i := 1; j := 1; scalone := false;
 2: S := ∅;                                        {scalony zbiór rozwiązań}
 3: while not scalone do
 4:   if i > p then
        {nie ma już rozwiązań w X, dołącz na koniec zbioru S pozostałe rozwiązania z Y}
 5:     S := S + {yj, ..., yq}; scalone := true;
 6:   else if j > q then
        {nie ma już rozwiązań w Y, dołącz na koniec zbioru S pozostałe rozwiązania z X}
 7:     S := S + {xi, ..., xp}; scalone := true;
 8:   else if ci^x < cj^y then
 9:     while ti^x ≤ tj^y do j := j + 1; end while    {pomiń rozw. zdominowane ze zbioru Y}
10:     S := S + {xi}; i := i + 1;                    {dołącz na koniec zbioru S rozwiązanie xi}
11:   else if cj^y < ci^x then
12:     while tj^y ≤ ti^x do i := i + 1; end while    {pomiń rozw. zdominowane ze zbioru X}
13:     S := S + {yj}; j := j + 1;                    {dołącz na koniec zbioru S rozwiązanie yj}
14:   else if ti^x < tj^y then
15:     while ti^x ≤ tj^y do j := j + 1; end while    {pomiń rozw. zdominowane ze zbioru Y}
16:     S := S + {xi}; i := i + 1;                    {dołącz na koniec zbioru S rozwiązanie xi}
17:   Else
18:     while tj^y ≤ ti^x do i := i + 1; end while    {pomiń rozw. zdominowane ze zbioru x}
19:     S := S + {yj}; j := j + 1;                    {dołącz na koniec zbioru S rozwiązanie yj}
20:   end if
21: end while
22: scal(X, Y) := S;
```

4.2. Algorytm rozwiązywania problemu transportu przesyłek

Algorytm rozwiązywania problemu transportu przesyłek został opracowany przez wprowadzenie odpowiednich modyfikacji w algorytmie BRUM, przestawionym w

podpunkcie 4.1. Zmodyfikowany algorytm, podobnie jak algorytm BRUM, umożliwia wyznaczanie wszystkich rozwiązań niezdominowanych z ustalonego węzła źródłowego v_p do wszystkich pozostałych węzłów grafu. Bezpośrednie zastosowanie algorytmu BRUM do rozwiązania problemu będącego tematem niniejszej pracy nie jest możliwe z następujących powodów:

- Rozwiązania wyznaczane za pomocą algorytmu BRUM zawierają wyłącznie etykiety określające czas i koszt przejazdu z węzła v_p do v_i, natomiast nie zawierają trasy przejazdu pomiędzy tymi węzłami; w przypadku omawianego problemu konieczna jest także znajomość trasy przejazdu.
- Jeżeli istnieje więcej niż jedno rozwiązanie niezdominowane o identycznym czasie i koszcie przejazdu, a różniące się trasą przejazdu, to za pomocą algorytmu BRUM zostanie wyznaczone tylko jedno z nich, natomiast w przypadku problemu transportu przesyłek konieczna jest znajomość wszystkich rozwiązań.
- Algorytm BRUM nie uwzględnia klas węzłów, dlatego trasa rozwiązania może nie zawierać węzła klasy *A*, co nie spełnia założeń problemu transportu przesyłek.
- Rozwiązanie wyznaczone za pomocą algorytmu BRUM nigdy nie zawiera cyklu [2], ponieważ rozwiązanie zawierające cykl jest zawsze rozwiązaniem zdominowanym, natomiast w przypadku transportu przesyłek wymagane jest, aby trasa zawierała co najmniej jeden węzeł klasy *A*, dlatego rozwiązanie może zawierać cykl.
- Jeżeli węzeł docelowy jest taki sam jak węzeł źródłowy, to czas i koszt transportu nie są równe 0, ponieważ:
  - w przypadku węzła klasy *A* należy uwzględnić czas i koszt sortowania w węźle,
  - w przypadku węzła klasy *B* lub *C* jest konieczność wystąpienia w trasie co najmniej jednego węzła klasy *A*, czyli przesyłka musi opuścić węzeł źródłowy, a co za tym idzie czas i koszt transportu są większe od 0.

Zanim zostanie przedstawiona zmodyfikowana wersja algorytmu BRUM, zdefiniujemy funkcje: *węzełC*(*v*), *koszt*(*v*) i *czas*(*v*). Funkcja *węzełC*(*v*) zwraca wartość *true* w przypadku, kiedy węzeł *v* jest węzłem klasy *C*. Natomiast funkcje *koszt*(*v*) i *czas*(*v*) zwracają odpowiednio koszt i czas sortowania w węźle *v*.

Uwzględniając wymienione różnice, zmodyfikowaną wersję algorytmu BRUM rozwiązującą problem transportu przesyłek można opisać za pomocą pseudokodu:

```
1: for v_i ∈ V do                    {inicjalizacja}
2:    R(v_i) := ∅;
3: end for
4: R(v_p) := {(czas(v_p), koszt(v_p), <v_p>)};
```

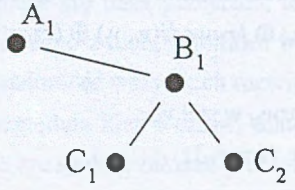
5: *kolejka* := $\{v_p\}$; {umieść w kolejce węzeł źródłowy}
6: **while** *kolejka* ≠ ∅ **do**
7: pobierz i usuń z *kolejki* węzeł v_i;
8: **for all** $v_j \in adj[v_i]$ **do**
9: $temp_{vj} := \emptyset$; {zbiór rozwiązań dla v_j utworzony ze zbioru rozwiązań dla v_i}
10: **for all** $rozw_{vi} \in R(v_i)$ **do** {utwórz z rozwiązań dla węzła v_i rozwiązania dla v_j}
11: **if not** *węzełC*(v_j) **or** trasa $rozw_{vi}$ zawiera węzeł klasy A **then**
12: **if** $v_i = v_p$ **then** {węzeł v_i jest węzłem źródłowym trasy}
13: $rozw_{vj} := rozw_{vi} \oplus krawędź(v_i, v_j)$;
14: **else** {węzeł v_i nie jest węzłem źródłowym}
{uwzględnij w rozwiązaniu czas i koszt sortowania w węźle v_i}
15: $rozw_{vj} := rozw_{vi} \oplus krawędź(v_i, v_j) \oplus (czas(v_i), koszt(v_i))$;
16: **end if**
17: włącz do trasy $rozw_{vj}$ węzeł v_j;
18: $temp_{vj} := temp_{vj} + \{rozw_{vj}\}$;
19: **end if**
20: **end for**
21: **if** $temp_{vj} \neq \emptyset$ **then** {ze zbioru rozwiązań dla v_i utworzono zbiór rozwiązań dla v_j}
22: $nowy_{vj} := scal(temp_{vj}, R(v_j))$; {utwórz nowy zbiór rozwiązań dla v_j}
23: **if** $nowy_{vj} \neq R(v_j)$ **then** {znaleziono nowy zbiór rozwiązań dla węzła v_j}
24: $R(v_j) := nowy_{vj}$; {zapamiętaj nowy zbiór rozwiązań}
25: **if** $v_j \notin$ *kolejka* **and not** *węzełC*(v_j) **then** {wstaw do kolejki węzeł v_j}
26: *kolejka* := *kolejka* + $\{v_j\}$;
27: **end if**
28: **end if**
29: **end if**
30: **end for**
31: **end while**

Pierwszą konieczną modyfikacją, jaką należy wprowadzić w algorytmie BRUM, jest rozszerzenie rozwiązania dla węzła v_i o trasę prowadzącą z węzła v_p do v_i. Zatem pojedyncze rozwiązanie ze zbioru $R(v_i)$ ma postać $(t_k^{vi}, c_k^{vi}, trasa_k^{vi})$; składa się ono z trzech elementów: pary etykiet określających odpowiednio czas i koszt dojazdu z węzła źródłowego do węzła v_i oraz trasy przejazdu. Trasa składa się z sekwencji węzłów $<v_1, v_2, ..., v_n>$, gdzie $v_1 = v_p$ i $v_n = v_i$. Rozwiązania będą uporządkowane niemalejąco względem kosztu przejazdu. Ponieważ są one rozwiązaniami niezdominowanymi, powoduje to automatyczne uporządkowanie

nierosnąco względem czasu przejazdu. Rozwiązania o identycznym czasie i koszcie przejazdu są dodatkowo uporządkowane rosnąco w porządku leksykograficznym względem trasy przejazdu [7]. Niech będą dane trasy $p = <p_1, ..., p_m>$ i $q = <q_1, ..., q_n>$. Trasa p poprzedza leksykograficznie trasę q, co zapisujemy $p < q$, jeżeli spełniony jest jeden z warunków:

1. Istnieje taki indeks j, $1 \leq j \leq min(m, n)$, że zachodzi zależność $p_i = q_i$ dla wszystkich wartości $i = 1, ..., j - 1$ oraz $p_j < q_j$.
2. Zachodzi $m < n$ oraz $p_i = q_i$ dla każdego $i = 1, ..., m$.

Elementami tras p i q są odpowiednio węzły p_i, q_i. Porównanie węzłów odbywa się na zasadzie porównania ich numerów.



Rys. 3. Przykładowa sieć połączeń
Fig. 3. A sample of the routing network

Kolejna modyfikacja polega na sposobie wyznaczania rozwiązania. W przypadku algorytmu BRUM trasa wyznaczonego rozwiązania nie zawiera cyklu [7], ponieważ każde rozwiązanie zawierające cykl jest rozwiązaniem zdominowanym. W przypadku problemu transportu przesyłek trasa rozwiązania może zawierać cykl. Jest to spowodowane wymogiem, aby trasa zawierała co najmniej jeden węzeł klasy A. Rozpatrzmy to na przykładzie. Wyznaczymy trasę przesyłki z węzła C_1 do węzła C_2 (rys. 3). Gdyby wyznaczyć trasę za pomocą algorytmu BRUM, wówczas uzyskalibyśmy następujące rozwiązanie: $C_1 \rightarrow B_1 \rightarrow C_2$. Jednak rozwiązanie to nie spełnia założeń przedstawionych w sformułowaniu problemu, ponieważ trasa nie zawiera węzła klasy A. Prawidłowym rozwiązaniem jest rozwiązanie postaci: $C_1 \rightarrow B_1 \rightarrow A_1 \rightarrow B_1 \rightarrow C_2$. Trasa rozwiązania zawiera cykl, jednak spełnia wymagania przedstawione w sformułowaniu problemu. Rozpatrzmy inny przypadek, kiedy węzeł docelowy jest taki sam, jak węzeł źródłowy oraz nie jest to węzeł klasy A. Taka sytuacja ma miejsce, kiedy wysyłamy przesyłkę np. z węzła C_1 do C_1 lub z węzła B_1 do B_1. Ponieważ każda przesyłka musi przejść przez co najmniej jeden węzeł klasy A, dlatego w obydwu przypadkach przesyłka musi opuścić węzeł źródłowy, aby przejść sortowanie zasadnicze. W pierwszym przypadku trasa przesyłki ma postać: $C_1 \rightarrow B_1 \rightarrow A_1 \rightarrow B_1 \rightarrow C_1$, natomiast w drugim: $B_1 \rightarrow A_1 \rightarrow B_1$. W obydwu przypadkach cała trasa rozwiązania stanowi cykl. W ogólnym przypadku, jeżeli węzeł docelowy jest taki sam jak węzeł źródłowy oraz nie

jest to węzeł klasy A, to cała trasa rozwiązania stanowi cykl, który zawiera węzeł klasy A. Rozwiązanie to ma koszt i czas transportu większy od zera. Natomiast, jeżeli węzeł docelowy jest taki sam jak źródłowy i jest on węzłem klasy A, to przesyłka nie musi go opuszczać, ponieważ sortowanie zasadnicze może się odbyć w tym węźle. Czas i koszt takiego rozwiązania jest równy czasowi i kosztowi sortowania w węźle klasy A.

Oznaczmy trasę transportu przesyłki jako $<v_1, ..., v_n>$, gdzie węzeł v_1 jest węzłem źródłowym ($v_1 = v_p$), natomiast węzeł v_n jest węzłem docelowym ($v_n = v_k$). Węzeł klasy C ma tylko jedno połączenie – z węzłem klasy A lub B. Z tego wynika, że węzeł klasy C może być tylko węzłem źródłowym lub docelowym, nigdy nie jest węzłem pośrednim, przez który przechodzi przesyłka. Węzły pośrednie, tj. węzły znajdujące się w trasie na pozycjach od 2 do n-1 (węzły $v_2,..., v_{n-1}$), są klasy A lub B. Dlatego w momencie wyznaczania nowego rozwiązania dla węzła v_j (kroki 12-17 algorytmu) na podstawie rozwiązań dla węzła v_i sprawdzane jest, czy węzeł v_j jest węzłem klasy C (krok 11). Jeżeli v_j jest węzłem klasy C, to oznacza, że musi być węzłem docelowym w trasie a nie pośrednim. Z tego wynika, że trasa w rozwiązaniu dla węzła v_i musi zawierać co najmniej jeden węzeł klasy A (krok 11). Tylko w takiej sytuacji jest tworzone nowe rozwiązanie dla węzła v_j. Dzięki sprawdzaniu tych zależności nigdy nie zostanie utworzone rozwiązanie dla węzła klasy C, które nie zawiera w trasie co najmniej jednego węzła klasy A. Zatem rozwiązanie $C_1 \rightarrow B_1 \rightarrow C_2$ nie zostanie utworzone. Ponieważ węzeł klasy C nie może być węzłem pośrednim, po wyznaczeniu nowego zbioru rozwiązań dla węzła klasy C nie jest on umieszczany w kolejce (krok 26). W przypadku gdy węzeł v_j jest węzłem klasy B, rozwiązanie jest tworzone niezależnie od liczby węzłów klasy A w trasie rozwiązania dla węzła v_i. Rozwiązania nie zawierające w trasie węzłów klasy A są usuwane na etapie scalania zbioru rozwiązań w funkcji *scal*.

Modyfikacji uległa także część inicjalizacyjna algorytmu. Dla węzła źródłowego nie jest tworzone rozwiązanie o zerowym czasie i koszcie, tylko rozwiązanie o czasie i koszcie równym czasowi i kosztowi sortowania w węźle źródłowym, a w trasie umieszczany jest węzeł źródłowy v_p. Czas i koszt sortowania w węźle są także uwzględniane w momencie wyznaczania nowego rozwiązania dla węzła v_j na podstawie rozwiązania dla węzła v_i (kroki 12-17). Wówczas uwzględniane są czas i koszt sortowania w węźle v_i. Jednak należy tutaj zwrócić uwagę na sytuację, kiedy węzeł v_i jest węzłem źródłowym v_p. W takim przypadku w rozwiązaniu dla węzła v_j nie są uwzględniane czas i koszt sortowania w węźle v_i (krok 13 algorytmu), ponieważ zostały one już uwzględnione w rozwiązaniu, które utworzono w części inicjalizacyjnej algorytmu dla węzła v_p.

W związku z modyfikacją postaci rozwiązania jak i samego algorytmu należy także zmodyfikować funkcję *scal*. Pseudokod zmodyfikowanej funkcji *scal* przedstawia się następująco:

```
scal(X, Y):
    X = {x_1, ..., x_p} – nowo utworzony zbiór rozwiązań; elementy indeksowane przez i
    Y = {y_1, ..., y_q} – stary zbiór rozwiązań; elementy indeksowane przez j
 1: i := 1; j := 1; scalone := false;
 2: S := ∅;                                          {scalony zbiór rozwiązań}
 3: if Y ≠ ∅ cand trasa rozwiązania j-tego w zbiorze Y nie zawiera węzła klasy A then
 4:   j := j + 1;                                    {pomiń rozwiązanie}
 5: end if
 6: while not scalone do
 7:   if i > p then
        {nie ma już rozwiązań w X, dołącz na koniec zbioru S pozostałe rozwiązania z Y}
 8:     S := S + {y_j, ..., y_q}; scalone := true;
 9:   else if j > q then
        {nie ma już rozwiązań w Y, dołącz na koniec zbioru S pozostałe rozwiązania z X}
10:     S := S + {x_i, ..., x_p}; scalone := true;
11:   else if c_i^x < c_j^y then
12:     while t_i^x ≤ t_j^y do j := j + 1; end while   {pomiń rozw. zdominowane ze zbioru Y}
13:     S := S + {x_i}; i := i + 1;                    {dołącz na koniec zbioru S rozwiązanie x_i}
14:   else if c_j^y < c_i^x then
15:     while t_j^y ≤ t_i^x do i := i + 1; end while   {pomiń rozw. zdominowane ze zbioru X}
16:     S := S + {y_j}; j := j + 1;                    {dołącz na koniec zbioru S rozwiązanie y_j}
17:   else if t_i^x < t_j^y then
18:     while t_i^x ≤ t_j^y do j := j + 1; end while   {pomiń rozw. zdominowane ze zbioru Y}
19:     S := S + {x_i}; i := i + 1;                    {dołącz na koniec zbioru S rozwiązanie x_i}
20:   else if t_j^y < t_i^x then
21:     while t_j^y ≤ t_i^x do i := i + 1; end while   {pomiń rozw. zdominowane ze zbioru X}
22:     S := S + {y_j}; j := j + 1;                    {dołącz na koniec zbioru S rozwiązanie y_j}
23:   else                                  {rozwiązania o takim samym czasie i koszcie}
24:     if trasa_i^x < trasa_j^y then
25:       S := S + {x_i}; i := i + 1;                  {dołącz na koniec zbioru S rozwiązanie x_i}
26:     else if trasa_j^y < trasa_i^x then
27:       S := S + {y_j}; j := j + 1;                  {dołącz na koniec zbioru S rozwiązanie y_j}
28:     else                                           {identyczne rozwiązania}
29:       i := i + 1;                                  {pomiń rozwiązanie ze zbioru X}
30:     end if
31:   end if
```

32: **end while**

33: *scal*(X, Y) := S;

W przypadku gdy węzeł, dla którego scalane są zbiory rozwiązań, nie jest węzłem klasy A, to stary zbiór rozwiązań może zawierać rozwiązanie, w którym trasa nie zawiera żadnego wierzchołka klasy A. Zbiór ten składa się tylko z jednego rozwiązania. Z sytuacja tą możemy mieć do czynienia wtedy, gdy węzłem źródłowym jest węzeł klasy C i jest on połączony z węzłem klasy B. Przy wyznaczaniu trasy z węzła C_1 do węzła C_2 (rys. 3) dla węzła B_1 zostanie wyznaczone i umieszczone w zbiorze rozwiązań rozwiązanie postaci C_1 –> B_1. Rozwiązanie C_1 –> B_1 –> C_2 nie zostanie utworzone, o czym już wspominano. Następnie zostanie wyznaczone rozwiązanie C_1 –> B_1 –> A_1, z którego w następnej kolejności rozwiązanie dla węzła B_1 o postaci C_1 –> B_1 –> A_1 –> B_1. Następnie do funkcji *scal* zostaną przekazane dwa zbiory rozwiązań:

- stary zbiór Y: C_1 –> B_1,
- nowy zbiór X: C_1 –> B_1 –> A_1 –> B_1.

Rozwiązanie ze zbioru Y nie jest zdominowane przez rozwiązanie ze zbioru X. Jednak zostanie ono pominięte (kroki 3-5) i nie będzie dołączone do scalonego zbioru rozwiązań S, ponieważ w trasie nie zawiera węzła klasy A. Zbiór S będzie zawierał tylko jedno rozwiązanie – rozwiązanie ze zbioru X.

Drugim przypadkiem, kiedy rozwiązanie ze zbioru Y nie zawiera w trasie węzła klasy A, jest sytuacja, w której węzeł docelowy jest taki sam, jak węzeł źródłowy i nie jest to węzeł klasy A. Wówczas zbiór Y zawiera rozwiązanie, które zostało utworzone w części inicjalizacyjnej algorytmu (krok 4). Jako przykład można podać wyznaczanie trasy z węzła C_1 do C_1. Początkowe etapy wyznaczania rozwiązania są takie same, jak w przypadku wyznaczania rozwiązania z węzła C_1 do C_2, tzn. wyznaczane są kolejno następujące rozwiązania:

- C_1 –> B_1,
- C_1 –> B_1 –> A_1,
- C_1 –> B_1 –> A_1 –> B_1.

Z rozwiązania C_1 –> B_1 –> A_1 –> B_1 zostanie utworzone rozwiązanie dla węzła C_1 i zbiory przekazane do funkcji *scal* będą zawierały rozwiązania:

- stary zbiór Y: C_1,
- nowy zbiór X: C_1 –> B_1 –> A_1 –> B_1 –> C_1,

gdzie zbiór Y zawiera rozwiązanie utworzone w części inicjalizacyjnej algorytmu. Rozwiązanie ze zbioru Y zostanie pominięte w funkcji *scal* w krokach 3-5, a w zbiorze S zostanie umieszczone rozwiązanie ze zbioru X.

Zbiór rozwiązań niezdominowanych może zawierać rozwiązania o identycznym czasie i koszcie przejazdu, ale różniące się trasą przejazdu. Takie rozwiązania są sortowane na etapie scalania rosnąco w porządku leksykograficznym względem trasy przejazdu (kroki 24-30).

5. Złożoność czasowa algorytmu

Algorytm rozwiązywania problemu wyznaczania optymalnej trasy transportu przesyłek został opracowany na podstawie algorytmu BRUM. Maksymalna długość trasy jest równa liczbie wierzchołków n w grafie. Wyznaczając kandydata na i-tą pozycję w trasie należy rozpatrzyć m wierzchołków, gdzie m jest równe stopniowi wyjściowemu wierzchołka [7]. Zatem algorytm BRUM ma wykładniczą złożoność czasową, która jest $O(m^n)$ [5].

W przypadku transportu przesyłek trasa nie może zawierać wszystkich węzłów, co wynika z ich podziału na klasy A, B i C. Węzeł klasy C może się znajdować się w trasie tylko na pierwszej lub ostatniej pozycji, czyli jest węzłem źródłowym albo docelowym. Zatem węzeł klasy C nie może być węzłem pośrednim, tzn. węzłem, przez który przechodzi przesyłka, co zostało omówione w podpunkcie 4.2. Jako pośrednie mogą wystąpić tylko węzły klasy A i B. Zatem maksymalna długość trasy jest równa $N_A + N_B + 2 < N$. Ponieważ węzeł klasy C może być tylko węzłem docelowym lub źródłowym, a nie może być węzłem pośrednim, dlatego maksymalna liczba kandydatów na i-ty węzeł w trasie jest równa $k = N_A + N_B$. Zatem złożoność algorytmu jest wykładnicza i jest $O(k^k)$.

Mimo złożoności wykładniczej przedstawiony algorytm można stosować nawet dla sieci o dużym rozmiarze, ponieważ graf reprezentujący sieć transportową jest z reguły grafem rzadkim.

6. Wyniki badań eksperymentalnych

Badania eksperymentalne zostały przeprowadzone dla sieci transportowej opisującej rzeczywistą sieć jednej z firm spedycyjnych działających na terenie naszego kraju. Sieć miała następujące parametry:

- liczba węzłów $N = 50$,
- liczba węzłów klasy A $N_A = 3$,
- liczba węzłów klasy B $N_B = 5$,
- liczba węzłów klasy C $N_C = 42$.

Badania przeprowadzono dla wszystkich par węzłów. Liczba ta jest równa liczbie 2-elementowych wariacji bez powtórzeń N-elementowego zbioru, czyli:

$$\binom{N}{2}2! = \frac{50!}{48!2!}2! = 2450.$$

Następnie został przeprowadzony eksperyment dla sieci o znacznie większym rozmiarze. Ponieważ nie dysponowano rzeczywistą siecią transportową o większym rozmiarze, dlatego

Tabela 1

Wyniki badań eksperymentalnych dla zmodyfikowanego algorytmu BRUM

	$N = 50$	$N = 1250$
czas eksperymentu [s]	1	68
liczba wykonanych iteracji	468	268 490
maksymalna długość trasy	5	7
liczba wyznaczonych rozwiązań niezdominowanych	2 901	1 864 137
liczba zbiorów rozwiązań niezdominowanych z 1 rozwiązaniem	2 135	1 354 105
liczba zbiorów rozwiązań niezdominowanych z 2 rozwiązaniami	333	139 641
liczba zbiorów rozwiązań niezdominowanych z 3 rozwiązaniami	28	48 688
liczba zbiorów rozwiązań niezdominowanych z 4 rozwiązaniami	4	16 274
liczba zbiorów rozwiązań niezdominowanych z 5 rozwiązaniami	0	3 162
liczba zbiorów rozwiązań niezdominowanych z 6 rozwiązaniami	0	630

została ona stworzona na potrzeby testów. Sieć została stworzona z zachowaniem właściwości sieci transportowej opisanej w pkt. 2. Sieć ta miała następujące parametry:

- liczba węzłów $N = 1250$,
- liczba węzłów klasy A $N_A = 75$,
- liczba węzłów klasy B $N_B = 125$,
- liczba węzłów klasy C $N_C = 1050$.

Podobnie jak w poprzednim przypadku badania zostały przeprowadzone dla wszystkich par węzłów. Liczba ta jest równa 1 561 250. Wyniki badań eksperymentalnych zostały przedstawione w tab. 1. Przez pojedynczą iterację rozumie się wykonanie kroków 8-24 algorytmu, natomiast przez długość trasy liczbę węzłów tworzących trasę.

Oprócz badań eksperymentalnych, dla zmodyfikowanego algorytmu BRUM przedstawionego w punkcie 4.2, przeprowadzono także badania dla algorytmu opartego na metodzie powrotów [3]. Dla sieci składającej się z 50 węzłów czas wyznaczania rozwiązań dla wszystkich par węzłów, podobnie jak w przypadku zmodyfikowanego algorytmu BRUM, nie przekroczył 1 sekundy. Wyniki badań dla sieci składającej się z 1250 węzłów zostały zaprezentowane w tab. 2. W tabeli tej przedstawiono czas wyznaczania rozwiązań z ustalonego węzła sieci do wszystkich pozostałych. Ze względu na znaczny czas działania algorytmu opartego na metodzie powrotów nie przeprowadzono badań dla wszystkich par

Tabela 2

Wyniki badań eksperymentalnych dla algorytmu opartego na metodzie powrotów

Węzeł początkowy	Czas eksperymentu	Węzeł początkowy	Czas eksperymentu
7	2:15.04	405	0:03.52
45	2:15.15	472	0:07.42
124	1:31.23	506	0:20.57
213	2:15.17	582	0:02.39
289	0:59.47	593	0:03.31
346	0:28.03	837	0:59.40

węzłów. Z przeprowadzonych badań wynika, że zmodyfikowany algorytm BRUM jest zdecydowanie lepszy od algorytmu opartego na metodzie powrotów. Umożliwia on wyznaczenie rozwiązań w rozsądnym czasie dla wszystkich par węzłów, co w praktyce jest niemożliwe do wykonania za pomocą metody powrotów, o czym świadczą wyniki przedstawione w tab. 2. Czas wyznaczania rozwiązań za pomocą zmodyfikowanego algorytmu BRUM dla wszystkich par węzłów jest równy około 1 minuty (tab. 1). Natomiast czas wyznaczania rozwiązań za pomocą algorytmu opartego na metodzie powrotów tylko z jednego węzła do pozostałych węzłów sieci w kilku przypadkach jest rzędu 2 godzin (tab. 2).

7. Podsumowanie

W pracy przedstawiono algorytm rozwiązywania problemu wyznaczania optymalnej trasy transportu przesyłek ze względu na dwa kryteria jednocześnie: czas i koszt transportu. Omówiony problem należy do klasy problemów optymalizacji wielokryterialnej, a dokładnie do klasy problemów optymalizacji dwukryterialnej. Rozwiązaniem problemu optymalizacji wielokryterialnej nie jest jedno konkretne rozwiązanie, tylko zbiór rozwiązań zwany zbiorem rozwiązań niezdominowanych. Przedstawiony algorytm umożliwia wyznaczenie wszystkich tras stanowiących zbiór rozwiązań niezdominowanych.

Rozwiązanie problemu będącego tematem niniejszej pracy sprowadza się do rozwiązania dwukryterialnego problemu najkrótszej ścieżki – BSP (ang. bicriterion shortest-path problem) w grafie ważonym o stałych wagach. W pracy został omówiony algorytm BRUM rozwiązujący problem BSP, który był podstawą do opracowania algorytmu rozwiązywania problemu wyznaczania optymalnej trasy transportu przesyłek. Bezpośrednie zastosowanie wymienionego algorytmu do rozwiązania problemu będącego tematem niniejszej pracy jest niemożliwe, ponieważ algorytm BRUM nie uwzględnia klas węzłów. Z tego powodu w algorytmie BRUM wprowadzono odpowiednie modyfikacje uwzględniające podział węzłów na trzy klasy oraz konsekwencje z tego wynikające, które zostały przedstawione w

podpunkcie 4.2. Dzięki wprowadzonym modyfikacjom uzyskano algorytm rozwiązywania rozważanego problemu.

Przedstawiony algorytm ma wykładniczą złożoność czasową. Jednak nie przekreśla to możliwości zastosowania go w praktyce. Grafy opisujące rzeczywiste sieci transportowe są z reguły grafami rzadkimi i nie zawierają dużej liczby węzłów. Graf opisujący sieć transportową jednej z firm spedycyjnych działających na terenie Polski składa się z 50 węzłów. Wyznaczenie zbiorów rozwiązań niezdominowanych dla wszystkich par węzłów tego grafu nie przekracza 1 sekundy. Istnieje możliwość zastosowania opracowanego algorytmu także dla większych sieci transportowych, co w praktyce nie jest możliwe do wykonania za pomocą algorytmu opartego na metodzie powrotów. Zostało to potwierdzone przeprowadzonymi badaniami eksperymentalnymi.

LITERATURA

1. Kuczora M.: Optymalizacja transportu przesyłek przy wykorzystaniu stałych linii komunikacyjnych i wyróżnionych węzłów sortujących. Studia Informatica, vol. 23, nr 4(51), ss. 105-124, 2001.
2. Reingold E., Nievergelt J., Deo N.: Algorytmy kombinatoryczne. PWN, Warszawa 1985.
3. Lipski W.: Kombinatoryka dla programistów. WNT, Warszawa 1989.
4. Brumbaugh-Smith J., Shier S.: An empirical investigation of some bicriterion shortest path algorithms. European Journal of Operational Research, vol. 43, pp. 216-224, North-Holland 1989.
5. Skriver A.J.V., Andersen K.A.: A label correcting approach for solving bicriterion shortest-path problems. Computers & Operations Research, vol. 27, pp. 507-524, 2000.
6. Stadler W.: A survey of Multicriteria Otimization or the Vector Maximum Problem. Part I: 1776-1960, Journal of Optimization Theory & Aplication, vol. 29, nr 1, pp. 1-52, USA wrzesień 1979.
7. Cormen T. H., Leiserson Ch. E., Rivest R. L.: Wprowadzenie do algorytmów. WNT, Warszawa 2000.
8. Peschel M., Riedel C.: Polioptymalizacja. Metody podejmowania decyzji kompromisowych w zagadnieniach inżynieryjno-technicznych. WNT, Warszawa 1979.

Recenzent: Dr Urszula Boryczka

Wpłynęło do Redakcji 5 lutego 2003 r.

Abstract

Finding the optimal routs of packages transportation problem (ORPT) is an example of bicriteria optimization problems. There are two optimization criteria: time and cost of transportation. The solution to a bicriteria optimization problem is the set of non-dominated solutions.

The paper presents an algorithm which determines all routes from one node to other nodes of transportation network. The routes belong to the set of non-dominated solutions. The algorithm solving ORPT problem is based on the BRUM algorithm presented in paper [4]. The BRUM algorithm solves the bicriterion shortest path problem in the weighted graph with constant weights.

The algorithm solving the ORPT problem was tested on two transportation networks. The first network described a transportation network for one of transportation companies. The network consisted of 50 nodes. The second network consisted of 1250 nodes and was created only for tests. It did not represent any real transportation network.

The complexity of the algorithm is exponential but it can be used for finding routes in real transportation networks. The graphs representing real transportation networks are usually sparse and do not contain many nodes. The algorithm was tested on a big transportation network (consisted of 1250 nodes) and the time of finding the solutions for this network was equal 68 seconds.

Adres

Jacek WIDUCH: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16,
44-101 Gliwice, Polska, jwiduch@star.iinf.polsl.gliwice.pl .