Krzysztof PIEŃKOSZ
Politechnika Warszawska

PROBLEM WYZNACZACZANIA MINIMALNEGO ZBIORU ŚCIEŻEK POŁĄCZEŃ

Streszczenie. W pracy rozpatrywany jest problem realizacji połączeń w sieciach telekomunikacyjnych poprzez jak najmniejszą liczbę ścieżek. Zastosowano model sieci przepływowej i zaproponowano dwufazową metodę rozwiązywania. Przedstawiono uogólniony algorytm Dijkstry, który wykorzystano do wyznaczania przepływów łukowych w pierwszej fazie, a następnie do dekompozycji tych przepływów na zbiór ścieżek w fazie drugiej.

ON FINDING A MINIMAL SET OF PATHS FOR A TRAFFIC DEMAND

Summary. In the paper the routing problem in telecommunication networks is considered where the minimal number of used paths is required. This problem is modeled as a network flow problem and a two-stage algorithm is proposed. The generalized Dijkstra algorithm is also presented which was used for finding a feasible network flow in the first stage, and then for the decomposition of this flow into a minimal number of paths in the second stage.

1. Wprowadzenie

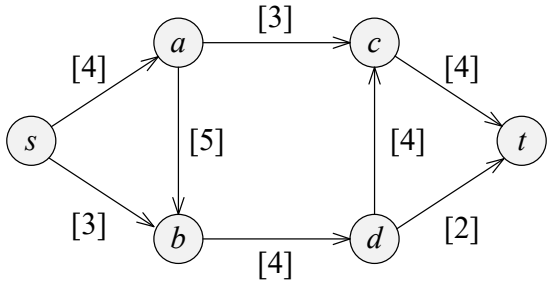
Do wyznaczania ścieżek połączeń w sieciach telekomunikacyjnych wykorzystano model sieci przepływowej. Rozpatrujemy sieci $S = (V, E, c)$ ze zbiorem wierzchołków V, zbiorem łuków E i przepustowościami $c(u,v)$ poszczególnych łuków $(u,v)\in E$. W sieci S wyróżniamy dwa wierzchołki: *źródło* s i *ujście* t i zakładamy, że chcemy zrealizować przepływ o wielkości F ze źródła s do ujścia t, reprezentujący zapotrzebowanie na połączenie między tymi wierzchołkami. Przyjmujemy, że zarówno F jak i $c(u,v)$ dla każdego $(u,v)\in E$ są dodatnimi liczbami całkowitymi.

Dla danej wielkości przepływu F, będziemy poszukiwać sposobu jego przesłania poprzez ścieżki biegnące od wierzchołka s do wierzchołka t. Niech D oznacza zbiór ścieżek użytych do realizacji przepływu F, a $f(p)$ wielkość przepływu w poszczególnych ścieżkach $p\in D$. Niech ponadto $D(u,v)$ oznacza zbiór ścieżek ze zbioru D, które przechodzą przez łuk (u,v).

Celem jest wyznaczenie jak najmniejszego zbioru ścieżek D realizujących przepływ o wielkości F, czyli spełniających warunki:

$$\sum_{p\in D} f(p) = F \quad \text{oraz} \quad \sum_{p\in D(u,v)} f(p) \le c(u,v) \quad \forall (u,v) \in E \qquad (1)$$

W przypadku sieci z rysunku 1, w której należy ścieżkami przesłać F=5 jednostek z wierzchołka s do t, można np. przesłać 3 jednostki ścieżką *s-a-c-t*, 1 jednostkę ścieżką *s-a-b-d-t* oraz 1 jednostkę ścieżką *s-b-d-t*. Byłyby wtedy wykorzystane 3 ścieżki. Można jednak znaleźć lepsze rozwiązanie – z mniejszą liczbą ścieżek, przesyłając 3 jednostki ścieżką *s-a-c-t* oraz 2 jednostki ścieżką *s-b-d-t*.



Rys. 1. Przykład sieci przepływowej
(w nawiasach kwadratowych przepustowości łuków)

2. Metoda wyznaczania ścieżek połączeń

Proponowany schemat wyznaczania jak najmniejszego zbioru ścieżek połączeń składa się z dwóch faz. W pierwszej fazie wyznaczany jest w sieci przepływ o wielkości F między wierzchołkami s i t. W rezultacie otrzymujemy informację o wielkości przepływu w każdym z łuków $(u,v)\in E$. Oczywiście nie może on być większy niż przepustowość $c(u,v)$. W drugiej fazie ten przepływ jest dekomponowany na przepływy ścieżkowe przy wykorzystaniu jak najmniejszej liczby ścieżek.

W pracy [3] wykazano, że problem dekompozycji rozpatrywany w drugiej fazie jest problemem *NP*-trudnym, a zatem *NP*-trudny jest też analizowany problem wyznaczania minimalnego zbiorów ścieżek połączeń, bo w szczególnym przypadku przepustowości łuków c mogłyby być zadane takie jak wielkości przepływów łukowych wyznaczone w fazie pierwszej. Zastosowano więc algorytmy heurystyczne.

2.1. Faza 2 – dekompozycja przepływu

Oznaczmy przez $E(p)$ zbiór łuków ścieżki p, a przez $w(p)$ – *szerokość ścieżki p*, tzn. $w(p) = \min\{c(u,v) : (u,v)\in E(p)\}$, gdzie $c(u,v)$ oznacza aktualną przepustowość łuku (u,v).

Z badań przeprowadzonych w literaturze [2,3] wynika, że jednym z najskuteczniejszych algorytmów drugiej fazy jest algorytm eliminacji najszerszej ścieżki (WIDE) w zmodyfikowanej sieci S_{F2}. Sieć S_{F2} różni się od pierwotnej sieci S tylko przepustowościami łuków. Są one równe przepływom łukowym wyznaczonym w fazie pierwszej. W kolejnych iteracjach algorytmu WIDE wyznaczana jest najszersza ścieżka, tzn. taka którą da się zrealizować największą część pozostałego przepływu.

Algorytm WIDE

1. $D := \varnothing$;
2. Wyznacz w sieci S_{F2} najszerszą ścieżkę p łączącą źródło s z ujściem t
   $f(p) := w(p)$;
3. Zredukuj wszystkie przepustowości na ścieżce p o wartość $f(p)$;
4. $D := D \cup \{p\}$; $F := F - f(p)$;
5. Jeżeli $F > 0$, to idź do 2, w przeciwnym przypadku KONIEC;

W przypadku, gdy w kroku 2 algorytmu WIDE jest kilka ścieżek o tej samej największej szerokości wybierana jest dowolna z nich. W pracy [2] stwierdzono, że zwykle lepsze rezultaty osiąga się, gdy spośród najszerszych ścieżek wybierana jest ta, która składa się z najmniejszej liczby łuków. Wówczas mniej łuków ma redukowane przepustowości w kroku 3 algorytmu i może to spowodować, że ścieżki wybierane w kolejnych iteracjach będą szersze, a w rezultacie będzie ich mniej. Taki wariant algorytmu został nazwany S-WIDE.

Stosując algorytm WIDE lub S-WIDE do zmodyfikowanej sieci S_{F2} mamy zawsze pewność, że cały przepływ o wielkości F będzie zdekomponowany na zbiór ścieżek. Gdybyśmy algorytm WIDE lub S-WIDE zastosowali bezpośrednio do pierwotnej sieci S, to mogłoby się zdarzyć, że suma przepływów przez uzyskany zbiór ścieżek jest mniejsza niż F, mimo że istnieje inny zbiór ścieżek, który przepływ o tej wielkości realizuje. Na przykład, stosując algorytm WIDE do sieci z rys. 1 wyznaczylibyśmy tylko jedną – najszerszą ścieżkę *s-a-b-d-c-t* z przepływem 4, mimo że tą siecią można przesłać aż 6 jednostek. Dlatego w fazie 1 najpierw poszukuje się przepływu od s do t o wielkości F i wyznacza się stosowne przepływy łukowe. Są one w fazie 2 traktowane jako przepustowości łuków w zmodyfikowanej sieci S_{F2}. Następnie dekomponuje się te przepływy na jak najmniejszą liczbę ścieżek.

2.2. Faza 1 – wyznaczanie przepływów łukowych

Przepływ w fazie 1 jest wyznaczany według klasycznego schematu zwiększania przepływu wzdłuż ścieżek powiększających w sieci rezydualnej S_f [1,4].

Dla sieci $S=(V, E, c)$ i ustalonego przepływu f, sieć rezydualna $S_f = (V, E_f, c_f)$, składa się z tego samego zbioru wierzchołków oraz dwóch rodzajów łuków:

- jeżeli $(u, v) \in E$ i $f(u, v) < c(u, v)$, to $(u, v) \in E_f$ i $c_f(u, v) = c(u, v) - f(u, v)$ (2)
- jeżeli $(u, v) \in E$ i $f(u, v) > 0$, to $(v, u) \in E_f$ i $c_f(v, u) = f(u, v)$. (3)

Łuki spełniające warunek (2) nazywamy *łukami zgodnymi* z przepływem, natomiast łuki, dla których zachodzi (3) – *łukami przeciwnymi*. *Ścieżką powiększającą* od s do t dla przepływu f nazywamy ścieżkę łączącą źródło s i ujście t w sieci rezydualnej $S_f = (V, E_f, c_f)$.

Algorytm wyznaczania przepływów łukowych

1. $F_f := 0$, $f(u,v) := 0$ dla każdego $(u,v) \in E_f$;
2. Znajdź w sieci rezydualnej S_f ścieżkę powiększającą p z s do t;
3. $\delta := \min\{w(p), F - F_f\}$;

4. $F_f := F_f + \delta$;
5. Zmodyfikuj przepływ f na ścieżce p, tzn.
   $f(u,v) := f(u,v) + \delta$, gdy (u,v) jest łukiem zgodnym,
   $f(u,v) := f(u,v) - \delta$, gdy (v,u) jest łukiem przeciwnym;
6. Zaktualizuj sieć rezydualną;
7. Jeżeli $F_f = F$ to KONIEC, w przeciwnym przypadku idź do 2;

W powyższym algorytmie zmienna F_f określa ile jednostek przepływu już zrealizowano (na początku 0), natomiast zmienne $f(u,v)$ – przepływy łukowe. Gdyby w kroku 2 nie istniała żadna ścieżka powiększająca, to oznaczałoby to, że nie da się zrealizować przepływu z wierzchołka s do t o zadanej wielkości F. W kroku 3, wyliczając wielkość δ określającą o ile można zwiększyć przepływ wzdłuż wybranej ścieżki, bierze się pod uwagę nie tylko szerokość ścieżki, ale też brakującą do pełnej realizacji przepływu wielkość $F - F_f$.

Od sposobu wyboru ścieżki powiększającej zależy postać wynikowego przepływu sieciowego, przy czym trudno jest stwierdzić jaki przepływ byłby najlepszy z punktu widzenia fazy 2. W pracy [2] sugerowano, żeby przy wyborze ścieżki powiększającej zastosować regułę WIDE lub S-WIDE również dla sieci rezydualnej S_f. W tej pracy proponuje się, żeby brać też pod uwagę liczbę łuków przeciwnych i wybierać w fazie 1 takie ścieżki powiększające, które mają tych łuków najmniej. Gdy jest kilka takich ścieżek (z jednakową liczbą łuków przeciwnych), to spośród nich jest wybierana ścieżka najszersza. Ten wariant fazy 1 będziemy nazywać WIDE-R. Ponadto będziemy też rozpatrywać wariant S-WIDE-R dla przypadków, gdy reguła WIDE-R nie daje jednoznacznego rozstrzygnięcia. W wariancie S-WIDE-R wybierana jest ścieżka powiększającą z najmniejszą liczbą wszystkich łuków spośród najszerszych ścieżek, które mają taką samą najmniejszą liczbą łuków przeciwnych.

3. Implementacja algorytmów

W pracy [2] proponowano, aby do wyznaczania ścieżki według reguły WIDE zastosować zmodyfikowany algorytm Dijkstry, natomiast do reguły S-WIDE – algorytm programowania dynamicznego. W tym drugim przypadku, jak również w przypadku reguł WIDE-R oraz S-WIDE-R, można jednak zaproponować algorytm o niższej złożoności, wykorzystujący również schemat algorytmu Dijkstry, ale w pewien uogólniony sposób.

3.1 Uogólniony algorytm Dijkstry

Niech $P(v)$ oznacza zbiór ścieżek rozpoczynających się w wierzchołku s i kończących się w wierzchołu v, $v \in V \setminus \{s\}$. Jeżeli $p \in P(u)$, to wyrażenie $p + (u,v)$ będzie oznaczać ścieżkę p wydłużoną o dodatkowy łuk (u,v), czyli $p + (u,v) \in P(v)$.

Zakładamy, że istnieje jakiś sposób oceny tych ścieżek i wzajemnego porównywania tworzący porządek liniowy, tzn.

$$p_1 \preceq p_2 \text{ lub } p_1 \succeq p_2 \qquad \text{dla każdej pary ścieżek } p_1, p_2 \tag{4}$$

Zapis $p_1 \leq p_2$ oznacza, że ścieżka p_1 jest „nie gorsza" od ścieżki p_2. Ocena ścieżek może być dokonywana na podstawie dowolnego jednego kryterium np. długości czy szerokości ścieżki, ale może być też brane pod uwagę kilka kryteriów. Oprócz (4), muszą być jedynie spełnione poniższe dwa warunki.

$$p \leqq p + (u,v), \qquad p \in P(u), \tag{5}$$

$$\text{jeżeli } p_1, p_2 \in P(u) \text{ i } p_1 \leqq p_2, \text{ to } p_1 + (u,v) \leqq p_2 + (u,v). \tag{6}$$

Celem jest znalezienie najlepszej ścieżki $p^*(t) \in P(t)$ od wierzchołka s to wierzchołka t, tzn. takiej, że $p^*(t) \leqq p$ dla wszystkich $p \in P(t)$. Okazuje się, że można ją wyznaczyć stosując poniższy uogólniony algorytm Dijkstry.

Uogólniony Algorytm Dijkstry (UAD)

1. $T := \{s\}$; $p^*(s) := (s, s)$;
2. Znajdź łuk (a,b), taki że $a \in T$, $b \in V \setminus T$ oraz
   $p^*(a) + (a,b) \leqq p^*(u) + (u,v)$ dla każdego innego łuku (u,v) z $u \in T$ oraz $v \in V \setminus T$;
3. $T := T \cup \{b\}$; $p^*(b) := p^*(a) + (a,b)$;
4. Jeżeli $b = t$ to KONIEC, w przeciwnym przypadku idź do 2;

W algorytmie UAD symbol $p^*(v)$ oznacza najlepszą ścieżkę od wierzchołka s do wierzchołka v, w szczególności ścieżka $p^*(t)$, to najlepsza szukana ścieżka od s do t. Dla uproszczenia zapisu przyjęto, że istnieje ścieżka od s do t oraz, że $p^*(s)$ to sztuczna „ścieżka" (s, s) składająca się z jednego wierzchołka s.

Twierdzenie 1. *Jeżeli sposób oceny ścieżek spełnia warunki (4), (5) i (6), to ścieżka $p^*(t)$ wyznaczona w algorytmie UAD jest najlepszą ścieżką, tzn. $p^*(t) \leqq p$ dla wszystkich $p \in P(t)$.*

Dowód. Pokażemy, że w każdej iteracji ścieżka $p^*(b)$ jest najlepszą ścieżką ze zbioru $P(b)$. W szczególności $p^*(t)$ jest najlepszą ścieżką z s do t.

Z warunków (4) i (6) oraz sposobu wyboru wierzchołków a i b w kroku 2 algorytmu UAD wynika, że $p^*(b) = p^*(a) + (a,b)$ jest najlepszą ścieżką spośród ścieżek ze zbioru $P(b)$, których przedostatni wierzchołek należy do T. Załóżmy, że istnieje lepsza od $p^*(b)$ ścieżka q z przedostatnim wierzchołkiem, który nie należy do zbioru T. Początkiem ścieżki q jest wierzchołek $s \in T$, zatem niech u będzie ostatnim jej wierzchołkiem, który należy do T, a v wierzchołkiem następnym. Ze sposobu wyboru ścieżki $p^*(b)$ oraz warunku (5) wynika, że $p^*(b) \leqq p^*(u) + (u,v) \leqq q(u) + (u,v) \leqq q$, gdzie $q(u)$ oznacza początkową część ścieżki q od wierzchołka s to u. Przeczy to założeniu, że istnieje lepsza od $p^*(b)$ ścieżka $q \in P(b)$. □

W praktyce, efektywnym sposobem realizacji kroku 2 algorytmu UAD jest zapamiętywanie w każdym wierzchołku ocen najlepszych ścieżek, których przedostatni wierzchołek należy do zbioru T, Po dołączeniu w kroku 3 nowego wierzchołka b do zbioru T, oceny te należy zaktualizować, ale wystarczy to zrobić tylko dla wierzchołków v takich, że $(b,v) \in E$, $v \in V \setminus T$. Przy zastosowaniu takiej implementacji złożoność obliczeniowa uogólnionego algorytmu Dijkstry jest taka jak klasycznego algorytmu Dijkstry [1] i wynosi $O(n^2)$, gdzie n oznacza liczbę wierzchołków sieci.

3.2 Wykorzystanie uogólnionego algorytmu Dijkstry

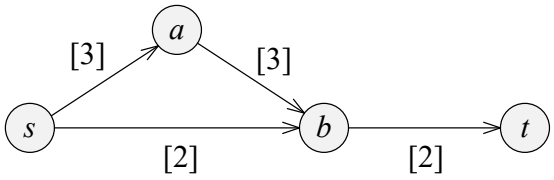
Uogólniony algorytm Dijkstry można wykorzystać do realizacji wariantów WIDE, S-WIDE, WIDE-R oraz S-WIDE-R przedstawionych w rozdziale 2.

W algorytmie WIDE poszukuje się najszerszej ścieżki w każdej iteracji. Relacja między ścieżkami jest więc następująca:

$$p_1 \leq p_2 \;\Leftrightarrow\; w(p_1) \geq w(p_2). \tag{7}$$

Taki sposób oceny ścieżek spełnia warunki (4), (5) i (6) i w związku z tym można zastosować algorytm UAD do wyznaczania najszerszej ścieżki w grafie.

W wariancie S-WIDE stosuje się hierarchicznie dwa kryteria oceny ścieżek. Wybierana jest w pierwszej kolejności ścieżka najszersza, a jeżeli jest kilka takich ścieżek o jednakowej szerokości, to spośród nich wybierana jest ścieżka z najmniejszą liczbą łuków. Niestety taki sposób oceny ścieżek nie spełnia warunku (6). W przykładzie przedstawionym na rys. 2 mamy zgodnie z regułą S-WIDE relację $p_1 \leqq p_2$ dla ścieżek $p_1 = s\text{-}a\text{-}b$ oraz $p_2 = s\text{-}b$, ale ścieżka $p_1 + (b,t)$ jest gorsza niż $p_2 + (b,t)$. Z tego powodu uogólniony algorytm Dijkstry by błędnie wyznaczył ścieżkę $s\text{-}a\text{-}b\text{-}t$ jako najlepszą. Zauważmy jednak, że według kryterium S-WIDE ścieżka $s\text{-}b\text{-}t$ jest lepsza, bo ma tę samą szerokość równą 2, ale mniej łuków.



Rys. 2. Przykład sieci, gdy warunek (6) nie jest spełniony dla reguły S-WIDE (w nawiasach kwadratowych przepustowości łuków)

Regułę S-WIDE można natomiast zrealizować nieco inaczej – w sposób dwuetapowy. Najpierw wyznaczamy szerokość $w_{\max}$ najszerszej ścieżki z s do t stosując algorytm UAD. W drugim etapie wyznaczamy ścieżkę od s do t z najmniejszą liczbą łuków w grafie zredukowanym – zawierającym tylko łuki (u,v) o przepustowości $c(u,v) \geq w_{\max}$. Tu też można wykorzystać algorytm UAD, ale bardziej efektywnym rozwiązaniem będzie zastosowanie techniki przeszukiwania grafu wszerz [4] startując z wierzchołka s.

Niech $l_R(p)$ oznacza liczbę łuków przeciwnych ścieżki p, a $l(p)$ – liczbę jej wszystkich łuków. W regule wyboru ścieżki powiększającej WIDE-R mamy relację:

$$p_1 \leq p_2 \Leftrightarrow\; l_R(p_1) < l_R(p_2) \;\vee\; (l_R(p_1) = l_R(p_2) \wedge w(p_1) \geq w(p_2)). \tag{8}$$

Relacja ta spełnia warunki (4), (5) i (6) i w związku z tym można w tym przypadku bezpośrednio zastosować algorytm UAD.

W wariancie S-WIDE-R stosuje się hierarchicznie trzy kryteria przy wyborze ścieżki powiększającej: jak najmniejsza liczba łuków przeciwnych, jak największa szerokość ścieżki oraz jak najmniejsza liczba wszystkich łuków. Podobnie jak w przypadku reguły S-WIDE warunek (6) może wtedy nie być spełniony. Można jednak wyznaczyć najlepszą ścieżkę według reguły S-WIDE-R metodą dwuetapową stosując

algorytm UAD w obu etapach. W pierwszym etapie określamy szerokość $w_{\max}$ ścieżki wyznaczonej według reguły WIDE-R i relacji (8). W drugim etapie rozpatrujemy zredukowany graf zawierający tylko łuki o przepustowościach $c(u,v) \geq w_{\max}$ i stosujemy algorytm UAD z następującym sposobem oceny ścieżek:

$$p_1 \leqq p_2 \Leftrightarrow l_R(p_1) < l_R(p_2) \vee (l_R(p_1) = l_R(p_2) \wedge l(p_1) \leq l(p_2)). \quad (9)$$

4. Eksperymentalne porównanie algorytmów

W tabelach 1 i 2 przedstawiono wyniki eksperymentów obliczeniowych dla wariantów algorytmów, w których w fazie 1 zastosowano reguły WIDE, S-WIDE, WIDE-R i S-WIDE-R, natomiast w fazie 2 regułę S-WIDE we wszystkich przypadkach. Sieci przepływowe użyte do tych eksperymentów zostały wygenerowane w sposób losowy i zawierały od 100 do 300 wierzchołków. Dla danej liczby wierzchołków rozpatrywane były 3 warianty sieci o różnej liczbie łuków m.

W tabeli 1 przedstawiono wyniki działania powyższych algorytmów. Zadana wielkość przepływu F była równa 90% wielkości maksymalnego przepływu $F_{\max}$ w poszczególnych sieciach.

Tabela 1

Porównanie wyników algorytmów

Parametry sieci		Liczba ścieżek			
n	m	WIDE	S-WIDE	WIDE-R	S-WIDE-R
100	1484	337	329	306	295
100	2475	188	176	167	155
100	3464	486	467	422	425
150	3352	600	573	498	483
150	5587	687	660	632	610
150	7822	1928	1915	1879	1854
200	5969	678	621	586	572
200	9950	1973	1940	1871	1883
200	13929	3355	3330	3260	3266
250	9337	1935	1910	1815	1816
250	15562	3197	3173	3051	3043
250	21787	4679	4641	4600	4604
300	13454	2447	2428	2310	2289
300	22425	3838	3822	3727	3742
300	31394	3624	3565	3534	3522

W tabeli 2 przedstawiono wyniki działania tych samych wariantów algorytmów dla tych samych sieci jak w tabeli 1, ale ze zmodyfikowanym sposobem realizacji fazy 1. Mimo, że w każdym przypadku wymagano zrealizowania przepływu o wartości $0{,}9F_{\max}$, to w zmodyfikowanej fazie 1, wyznaczano przepływ większy, równy $F_{\max}$. W fazie 2 natomiast dekomponowano na ścieżki tylko wymagane $0{,}9F_{\max}$ jednostek

przepływu. Dzięki takiemu podejściu, w fazie 2 była większa elastyczność w wyborze zbioru ścieżek realizujących zadany przepływ. Wyniki przedstawione w tabelach 1 i 2 pokazują, że uzyskuje się wtedy zwykle dużo mniejszą liczbę ścieżek.

Porównując między sobą warianty WIDE, S-WIDE, WIDE-R oraz S-WIDE-R można stwierdzić, że z jednej strony w większości przypadków S-WIDE daje lepsze rezultaty niż WIDE oraz S-WIDE-R jest lepsze niż WIDE-R. Z drugiej strony uwzględnienie dodatkowo liczby łuków przeciwnych przy wyborze ścieżek powiększających też zwykle poprawia końcowe wyniki.

Tabela 2

Porównanie wyników algorytmów przy zmodyfikowanej fazie 1

Parametry sieci		Liczba ścieżek			
n	*m*	WIDE	S-WIDE	WIDE-R	S-WIDE-R
100	1484	254	249	246	245
100	2475	145	145	139	139
100	3464	389	388	373	372
150	3352	440	434	420	418
150	5587	558	556	549	544
150	7822	1736	1736	1717	1705
200	5969	518	505	485	484
200	9950	1749	1742	1716	1709
200	13929	3135	3149	3109	3101
250	9337	1667	1675	1642	1638
250	15562	2912	2923	2858	2848
250	21787	4496	4469	4422	4415
300	13454	2157	2164	2093	2092
300	22425	3569	3577	3533	3521
300	31394	3403	3389	3356	3358

LITERATURA

1. Ahuja R.K., Magnanti T.L., Orlin J.B.: Network Flows: theory, algorithms and applications. Prentice Hall, New Jersey, 1993.
2. Nakibly G., Cohen R., Katzir L.: Optimizing Data Plane Resources for Multipath Flows. IEEE/ACM Transactions on Networking 23, 2013, s. 138-147.
3. Vatinlen B., Chauvet F., Chretienne P., Mahey P.: Simple bounds and greedy algorithms for decomposing a flow into minimal set of paths. European Journal of Operational Research 185, 2008, s. 1390–1401.
4. Wojciechowski J., Pieńkosz K.: Grafy i sieci. PWN, Warszawa 2013.