STUDIA INFORMATICA 2013
Volume 34 Number 3 (113)

Krzysztof STACHOWIAK, Piotr ZWIERZYKOWSKI
Politechnika Poznańska, Wydział Elektroniki i Telekomunikacji

ARCHITEKTURA I IMPLEMENTACJA WIELOKRYTERIALNEGO ALGORYTMU ROUTINGU DLA RUCHU ROZGAŁĘŹNEGO

Streszczenie. Ze względu na złożoność problemu, dla wielokryterialnej optymalizacji trasowania rozgałęźnego zaproponowano szeroką gamę rozwiązań heurystycznych, które cechują się zróżnicowaną złożonością i dokładnością. W artykule przedstawiono propozycję nowego algorytmu, jak również rozważania na temat jego implementacji. Zaproponowany algorytm, zdaniem autorów, pozwala na osiągnięcie dobrego kompromisu pomiędzy wydajnością i jakością wyników.

Słowa kluczowe: Qos, routing, multicast, implementacja

THE ARCHITECTURE AND THE IMPLEMENTATION OF A MULTICRITERIAL MULTICAST ROUTING ALGORITHM

Summary. The multicriterial multicast routing has been given a lot of attention. Algorithms of different precision and complexity have been proposed to solve the problem. In the article a novel algorithm for this class of the problems have been presented, which allows for reaching the compromise between the efficiency and the quality of the results.

Keywords: QoS, routing, multicast, implementation

1. Wprowadzenie

Rozwiązania problemu znalezienia w grafie minimalnego drzewa Steinera z wieloma ograniczeniami (*Multi-Constrained Minimal Steiner Tree – MCMST*) [3] odzwierciedla zestawianie połączeń punkt-wielopunkt w konwergentnych sieciach pakietowych i łączy w sobie, z matematycznej perspektywy, dwa problemy.

W sieciach o charakterze wielousługowym można podzielić ruch ze względu na jego wymagania względem ścieżki połączeniowej. Do oceny ścieżek – z punktu widzenia wymagań różnych klas ruchu (usług) – wprowadzono tzw. parametry jakości usług (ang. *Quality of Service*). W przypadku jednej klasy ruchu, np. usługi czasu rzeczywistego, krytyczna może być minimalizacja takich parametrów jak opóźnienie czy zróżnicowanie opóźnienia (ang. *Jitter*). Chociaż przesyłane pakiety mogą być stosunkowo małych rozmiarów, nie wymagają dużej przepływności ścieżki połączeniowej. Jednak, gdy przesyłane są np. pliki danych, wymagania są wręcz odwrotne – wielkość, a także parametry statystyczne opóźnień poszczególnych ścieżek nie są aż tak istotne, natomiast wielkość przepływności cechuje się dużą wagą. Ze względu na te czynniki, we współczesnych sieciach możemy spotkać różną definicję optimum dla różnych klas ruchu. Dodatkowo, samo określenie tego kryterium jest bardziej złożone niż np., znajdowanie optymalnej ścieżki wyłącznie ze względu na liczbę przeskoków. W sieci konwergentnej opis rozwiązania optymalnego zawiera więcej niż tylko jedną składową i może mieć charakter nie tylko ekstremum, ale i ograniczenia (np. opóźnienie transmisji może mieć dowolną wielkość pod warunkiem, że nie przekroczy zadanego progu).

Drugi wymiar rozwiązywanego w artykule problemu to liczba węzłów, które należy połączyć. W transmisji rozgałęźnej (tzw. *multicast*) liczba dopuszczalnych rozwiązań jest znacznie większa niż w przypadku trasowania punkt-punkt (tzw. *unicast*), co przekłada się na znaczne zwiększenie złożoności obliczeniowej algorytmów optymalizujących ten problem [1,2]. Dodatkowo, nieco bardziej skomplikowany jest opis ilościowy odnalezionych rozwiązań. Jeśli chcemy wyznaczyć parametr taki jak np. koszt zestawienia połączenia rozgałęźnego, to możemy w tym celu wyznaczyć sumę kosztów wszystkich gałęzi drzewa wynikowego. Jeżeli jednak analizujemy opóźnienie transmisji, to musimy wziąć pod uwagę każde z połączeń między wszystkimi parami w transmisji i wyznaczać opóźnienie jedynie na ścieżkach pomiędzy nimi [3].

Ze względu na to, że rozpatrywany problem cechują nietrywialny model opisu oraz duża złożoność obliczeniowa, infrastruktura informatyczna, służąca do jego badań, wymaga szczególnej uwagi podczas projektowania i realizacji eksperymentów badawczych. Przedstawione zostały idea oraz realizacja badań nad nowym algorytmem RDP (ang. *RenDezvouz Point*), w świetle wskazanych powyżej problemów. W rozdziale 2 zaprezentowane zostały dwa spojrzenia na model: matematyczne oraz implementacyjne. W rozdziale 3 wprowadzono definicję zaproponowanego algorytmu w dwóch wariantach wraz z omówieniem dodatkowych zagadnień, potrzebnych do jego opisu oraz struktur danych, służących do jego implementacji. Zaproponowana procedura symulacyjna oraz wyniki badań wraz z opisem i omówieniem znajdują się w rozdziale 4, natomiast rozdział 5 zawiera podsumowanie.

2. Struktury danych

2.1. Model matematyczny

W dalszych rozważaniach wykorzystany zostanie, proponowany we wcześniejszych pracach, model matematyczny [4,5]. Sieć telekomunikacyjna jest modelowana za pomocą nieskierowanego grafu $G(E,V)$, składającego się ze skończonego zbioru wierzchołków V oraz zbioru krawędzi $E \subset \{(u,v) : u,v \in V\}$. Każdej krawędzi przypisane jest M metryk zadanych funkcjami postaci: $(m_i : E \to \Re^+, i = 0,1,\ldots,M-1)$, odzwierciedlającymi addytywne koszty krawędzi.

Ścieżkę $p(s,d)$ między węzłami s a d $(s,d \in V)$ definiujemy jako ciąg niepowtarzających się węzłów $v_1, v_2, \ldots, v_k \in V$, takich że dla każdego $1 \le i \le k$ istnieje w grafie krawędź (v_i, v_{i+1}) oraz $v_1 \equiv s, v_k \equiv d$. Addytywny koszt ścieżki p względem i-tej metryki określamy jako $m_i(p) = \sum_{e \in p} m_i(e)$.

Zakorzenione drzewo multicast $t(s,d_1,d_2,\ldots)$, łączące węzeł źródłowy $s \in V$ z wieloma węzłami docelowymi $d_1, d_2, \ldots \in D \subset V$, definiujemy jako drzewo w grafie G, którego jedyne liście pochodzą ze zbioru $\{s\} \cup D$, a węzeł s wyróżniamy jako korzeń. Addytywny koszt drzewa t względem i-tej metryki określamy jako $m_i(t) = \sum_{e \in t} m_i(e)$. W ramach drzewa $t(s,d_1,d_2,\ldots)$ wyróżniamy ścieżki z korzenia do poszczególnych węzłów docelowych i oznaczamy jako $p_t(s,d_i)$.

Dla danego problemu routingu definiujemy zbiór ograniczeń C jako $(c_i \in \Re^+, i = 1,2,\ldots,M-1)$, w taki sposób, że indeksy ograniczeń odpowiadają indeksom metryk, których dotyczą.

Problem MCMST definiujemy jako znalezienie drzewa t^*, rozpinającego węzeł źródłowy s oraz węzły docelowe ze zbioru D, które spełnia następujące warunki:

$\forall t \in T(s,d_1,d_{2,}\ldots) : m_0(t^*) \le m_0(t)$,

$\forall d_i \in D, c_j \in C : m_j(p_t(s,d_i)) \le c_j$.

2.2. Implementacja

2.2.1. Węzły

Implementacja modelu przedstawionego w podrozdziale 2.1 nie wymaga definiowania specjalnej struktury danych dla reprezentowania węzła. Do tego celu wystarczy unikalny

identyfikator. Chociaż nie jest to konieczne, najlepiej jest stosować identyfikatory numeryczne, co ułatwia ich przechowywanie, porządkowanie oraz przeszukiwanie ich kolekcji. Jeśli wymagane jest przechowywanie dodatkowych informacji o węzłach, to warto zadbać o to, aby identyfikatory węzłów odpowiadały wprost indeksom, pod którymi przechowywane są struktury z tymi informacjami. Dzięki takiemu zabiegowi uzyskujemy łatwy i wydajny sposób dostępu do tych danych. Dodatkowe informacje, jakie mogą być przypisane węzłom to np.:

- współrzędne topograficzne,
- etykiety.

2.2.2. Krawędzie

Struktura definiująca krawędź składa się z pary identyfikatorów węzłów, pomiędzy którymi ona istnieje oraz ciągu jej metryk. Możliwe jest pominięcie informacji o węzłach, które łączy krawędź, jeśli można przenieść je do zewnętrznego kontekstu. Gdyby np. przechowywać krawędzie w dwuwymiarowej kolekcji, przy założeniu, że indeksy tej struktury odpowiadają indeksom węzłów, możliwe jest pominięcie identyfikatorów węzłów, ponieważ wynikają one z położenia struktury w zewnętrznej kolekcji.

2.2.3. Grafy

Informacje o węzłach i krawędziach przechowywane są w strukturze reprezentującej graf, czyli przechowującej informacje topologiczne. Dwa podstawowe sposoby przedstawiania topologii to macierz sąsiedztwa oraz lista sąsiedztwa.

W przypadku macierzy sąsiedztwa mamy do czynienia z dwuwymiarową strukturą, której indeksy odpowiadają identyfikatorom węzłów. Zatem każdy element tej struktury przyporządkowany jest, dzięki swojej lokalizacji, parze węzłów. Taki element może przyjąć wartość pustą, aby odwzorować brak krawędzi między daną parą węzłów, lub pewną niepustą wartość, mogącą oznaczać samo istnienie krawędzi lub np. precyzyjnie określać jej metryki. W ten sposób strukturę otrzymujemy umożliwiającą dostęp do konkretnych krawędzi w czasie stałym, jednak o liniowym czasie dostępu do listy sąsiadów danego wierzchołka. Dodatkowo struktura ta wprowadza znaczną nadmiarowość dla grafów rzadkich, ze względu na konieczność jawnej reprezentacji braku krawędzi pomiędzy większością spośród możliwych par krawędzi.

Lista sąsiedztwa to przyporządkowanie każdemu węzłowi grafu listy jego sąsiadów, z ewentualnym określeniem parametrów krawędzi, która do tego sąsiada prowadzi. Takie rozwiązanie cechuje się liniowym czasem dostępu do krawędzi i stałym czasem dostępu do listy sąsiadów danego węzła; w przeciwieństwie do macierzy sąsiedztwa, nie wprowadza ono redundancji w przypadku grafów rzadkich.

2.2.4. Uogólniony interfejs

W przypadku tworzenia uogólnionej platformy badań algorytmów grafowych, warto dysponować obiema implementacjami, aby móc wydajnie realizować badania dla topologii o różnym charakterze. Dlatego zdefiniowany został wspólny interfejs dla typu grafowego, z możliwością dostarczania wymiennych implementacji. Operacje, jakie powinien umożliwiać taki typ są następujące:

- pobranie kolekcji węzłów,
- pobranie kolekcji krawędzi,
- pobranie węzła o danym identyfikatorze,
- pobranie krawędzi spomiędzy węzłów o danych identyfikatorach,
- pobranie kolekcji węzłów sąsiadujących z danym węzłem.

3. Algorytm

3.1. Schemat działania

3.1.1. Terminologia symulacyjna

Algorytm RDP przedstawiony w artykule opiera się na współbieżnym wykonywaniu wielu instancji algorytmu Dijkstry. Pojęcie współbieżności wprowadza kontekst upływu czasu, przez co najlepiej jest sformułować algorytm Dijkstry z wykorzystaniem terminologii symulacyjnej. Analizę kolejnych wierzchołków grafu traktować będziemy jako symulację propagacji sygnału w grafie. Niektóre operacje algorytmu zostaną przedefiniowane następująco:

- wybór najbliższego węzła stanie się wyborem najwcześniejszego zdarzenia do obsłużenia,
- analiza danego węzła będzie interpretowana jako obsługa dotarcia do tego węzła sygnału,
- etykietowanie sąsiadów analizowanego węzła przeformułujemy jako planowanie zdarzeń dotarcia sygnału do danych sąsiadów.

3.1.2. Wieloźródłowy algorytm Dijkstry

Choć w ramach pracy algorytmu symulujemy wiele instancji algorytmu Dijkstry (nazywanych „procesami zbieżnymi”), to w rdzeniu algorytmu są one synchronizowane. Kolejne węzły są analizowane przez algorytm w podobnej kolejności, w jakiej działoby się to w przypadku algorytmu Dijkstry, w którym podczas inicjalizacji przypisano zerową etykietę więcej niż jednemu węzłowi. Synchronizacja następuje dzięki zastosowaniu szczególnego mechanizmu wyboru kolejnego odwiedzanego węzła. Mechanizm ten polega na analizie możliwości kolejnych odwiedzin wszystkich procesów, a następnie wybiera zdarzenie najwcześniejsze w skali globalnej. Dzięki niezależnemu etykietowaniu węzłów przez każdy spośród proce-

sów, poszczególne symulacje będą się przenikać. Jednak zjawisko to jest śledzone w rdzeniu algorytmu, tak że zachowywane są informacje o liczbie procesów, które odwiedziły dany węzeł.

3.1.3. Węzły zborne

Informacje o liczbie odwiedzin w danym węźle są kluczowe do wyznaczania tzw. węzłów zbornych – węzłów RDP – a więc węzłów, w których spotkały się sygnały wszystkich procesów. Ponieważ każdy spośród procesów odwiedza docelowo wszystkie węzły grafu, a kolejne węzły odwiedzane są po kolei, każdy z węzłów grafu zostanie w pewnym momencie wskazany jako kolejny węzeł RDP. Kolejne oznaczanie węzłów jako RDP to odnajdowanie kolejnych węzłów, w których spotkały się procesy zbieżne. Ponieważ algorytm Dijkstry optymalizuje długość ścieżki, kolejne węzły RDP będą wskazywane w kolejności rosnącego sumarycznego kosztu. Obserwacja ta jest esencją działania algorytmu, choć jest wykorzystywana w różny sposób, w różnych jego wariantach. W ramach dotychczasowych badań zaproponowano dwa takie warianty: „quasi-dokładny” oraz „heurystyczny”.

3.2. Warianty algorytmu

Zbiór wszystkich możliwych węzłów RDP dla wszystkich drzew multicast, jakie można zestawić w grafie tworzy pełną przestrzeń rozwiązań problemu. Gdyby uszeregować wszystkie RDP według rosnącej sumarycznej metryki m_0 i wybrać pierwszy węzeł RDP, spełniający zadane ograniczenia, to otrzymalibyśmy rozwiązanie dokładne dla rozpatrywanego problemu. Procedurę odwiedzania kolejnych węzłów RDP przez proponowany algorytm można określić jako odwiedzanie wybranego podzbioru wszystkich możliwych węzłów RDP, jednak z zachowaniem kolejności rosnącej sumarycznej metryki. Ponieważ przypomina ona poszukiwanie rozwiązania dokładnego, podejście takie zostało nazwane quasi-dokładnym. Została ona przedstawiona w postaci następującego algorytmu:

```
1: procedure RDP_QE(g, s, D, C)
2:     for d ∈ {s} ∪ D do
3:         incjalizuj_proces(d, g)
4:     end for
5:     while true do
6:         p := wybierz_najbliższy_proces()
7:         n := obsłuż_najbliższe_zdarzenie(p)
8:         licznik_wizyt(n) := licznik_wizyt(n) + 1
9:         if licznik_wizyt(n) = |D| + 1 then
10:            t := buduj_drzewo(n, g)
11:            if spełnia_ograniczenia(t, C) then
12:                Zwróć t
13:            end if
14:        end if
15:        if wszystkie_węzły_odwiedzone() then
16:            Zwróć porażkę
17:        end if
18:     end while
19: end procedure
```

Proponowane rozwiązanie jest atrakcyjne koncepcyjnie, jednak może doprowadzić do przejrzenia całego grafu przez wszystkie procesy zbieżne, co przekłada się na dużą złożoność obliczeniową. Złożoność tę można zmniejszyć zabiegiem zaproponowany w podejściu heurystycznym. Ten wariant, w poszczególnych procesach zbieżnych, stosuje heurystyczną optymalizację, która bierze pod uwagę wszystkie metryki, z wykorzystaniem agregacji zaproponowanej wcześniej w [6]. Agregacja ta przekształca zbiór metryk drzewa za pomocą wzoru: $m_{aggr}(t) = \max\left\{\frac{m_1(t)}{c_1}, \frac{m_2(t)}{c_2}, \ldots\right\}$. Przy takim podejściu rośnie średnia jakość rozwiązania budowanego z pierwszego napotkanego węzła RDP, dlatego pierwszy napotkany rezultat traktowany jest jako rozwiązanie ostateczne. Wariant ten przedstawiony został w postaci następującego algorytmu:

```
1: procedure RDP_H(g, s, D, C)
2:     for d ∈ D do
3:         inicjalizuj_proces(d, g)
4:     end for
5:     while true do
6:         p := wybierz_najbliższy_proces()
7:         n := obsłuż_najbliższe_zdarzenie(p)
8:         licznik_wizyt(n) := licznik_wizyt(n) + 1
9:         if licznik_wizyt(n) = |D| + 1 then
10:            t := buduj_drzewo(n, g)
11:            if spełnia_ograniczenia(t, C) then
12:                Zwróć t
13:            else
14:                Zwróć porażkę
15:            end if
16:        end if
17:    end while
18: end procedure
```

3.3. Atomowe operacje

Operacje używane w listingach są w większości wykorzystywane przez oba warianty algorytmu. Ich interpretacja została przedstawiona poniżej:

- *inicjalizuj_proces(d, g)* – inicjalizuje proces zbieżny w grafie *g* dla węzła *d*,
- *wybierz_najbliższy_poces()* – wybiera proces zbieżny, którego najbliższe zdarzenie jest jednocześnie najbliższym spośród wszystkich zdarzeń wszystkich procesów,
- *obsłuż_najbliższe_zdarzenie(p)* – przeprowadza analizę najbliższego węzła w procesie *p* według zasad algorytmu Dijkstry i zwraca identyfikator odwiedzonego węzła,
- *licznik_wizyt* – jest strukturą przyporządkowującą każdemu węzłowi grafu liczbę procesów, które już go odwiedziły,
- *wszystkie_węzły_odwiedzone* – sprawdza, czy wszystkie węzły grafu zostały już odwiedzone przez wszystkie procesy zbieżne.

3.4. Struktury danych

Z poszczególnymi procesami zbieżnymi związane są specyficzne struktury danych, dlatego warto zwrócić uwagę na ich charakterystyczne cechy.

3.4.1. Proces zbieżny optymalizujący koszt

Ten proces odpowiada algorytmowi Dijkstry i dlatego wykorzystuje takie same struktury danych, jak prekursor:

- zbiór „otwartych" węzłów,
- mapa etykiet węzłów: $węzeł \rightarrow \Re^+$,
- mapa poprzedników: $węzeł \rightarrow węzeł$.

3.4.2. Proces zbieżny heurystyczny

Ze względu na charakter obliczeń w procesie heurystycznym, zbiór struktur danych do jego obsługi jest rozszerzony względem wariantu optymalizującego jedynie metrykę m_0. Musi on dodatkowo przechowywać etykiety węzłów, zawierające koszty dojścia do danego węzła względem wszystkich metryk $\{m_0, m_1, ...\}$:

- zbiór „otwartych" węzłów,
- zbiór ograniczeń danego problemu,
- mapa zagregowanych etykiet węzłów: $węzeł \rightarrow \Re^+$,
- mapa konkretnych etykiet węzłów: $węzeł \rightarrow \{m_0, m_1, \ldots\}$,
- mapa poprzedników: $węzeł \rightarrow węzeł$.

3.4.3. Wskazówki implementacyjne

Zasada działania algorytmu w obu przypadkach jest bardzo podobna, jednak nieliczne różnice sprawiają, że stworzenie wspólnego kodu dla części głównej algorytmu byłoby niepraktyczne. Problemem jest np. określenie reakcji na zaobserwowanie warunku końca, który jest częścią rdzenia algorytmu. Ponieważ należy przygotować osobne implementacje rdzenia algorytmu dla poszczególnych jego wariantów, nie istnieje znacząca korzyść, która wynikałaby z ustalania części wspólnej implementacji procesów zbieżnych. Z powyższych powodów zarówno część główna algorytmu, jak i implementacje poszczególnych procesów zbieżnych zostały przygotowane niezależnie i nie współdzielą kodu źródłowego.

4. Eksperyment symulacyjny

4.1. Procedura symulacyjna

Przedstawione rozwiązanie konfigurowalne jest w dwóch głównych płaszczyznach: zakapsułkowanej niskopoziomowej implementacji reprezentacji struktur danych oraz realizacji głównej koncepcji algorytmu w dwóch przedstawionych wariantach. Dla zobrazowania obu tych aspektów, przygotowano eksperyment symulacyjny, mierzący wielkości, które są z nimi związane.

Elastyczność implementacji przedstawiono poprzez symulację zestawienia wielu losowo wybranych drzew multicast, z wykorzystaniem różnych wariantów algorytmu i porównując jakość wyników. Wagę doboru realizacji podstawowych modułów systemu uwypuklono mierząc rzeczywiste czasy wykonania tych samych zadań przy wykorzystaniu różnych implementacji topologii.

Eksperyment opiera się na rozwiązaniu wielu problemów oraz statystycznej analizie ich rozwiązań. Problemy dane są przez zestaw parametrów stałych - dobranych eksperymentalnie na potrzeby danego eksperymentu - oraz zmiennych definiujących wielowymiarową przestrzeń rozwiązań. Analiza statystyczna sprowadza się do przedstawienia wyników na wykresie, w taki sposób, że dwie spośród zmiennych wybrane są jako osie wykresów, na których przedstawione są rozwiązania, zagregowane względem pozostałych zmiennych.

4.1.1. Stałe parametry

Na potrzeby eksperymentu wybrano generator topologii, oparty na algorytmie Waxmana, z parametrami $\alpha = 0{,}15$ oraz $\beta = 0{,}2$. W ramach każdego wygenerowanego grafu zestawiono *300* połączeń multicast, co gwarantuje dobrą jakość statystyczną rozwiązań – uzyskane przedziały ufności są mniejsze o 2 rzędy wielkości od zmierzonych średnich. Wagi krawędzi zostały wylosowane z przedziału $\langle 1{,}1000 \rangle$ z rozkładem równomiernym. Problem MCMST wymaga określenia w każdym przypadku ograniczeń dla metryk $\{m_1, m_2, \ldots\}$, co zostało wykonane według metody przedstawionej w [3] ze współczynnikiem *0,9*.

4.1.2. Zmienne parametry

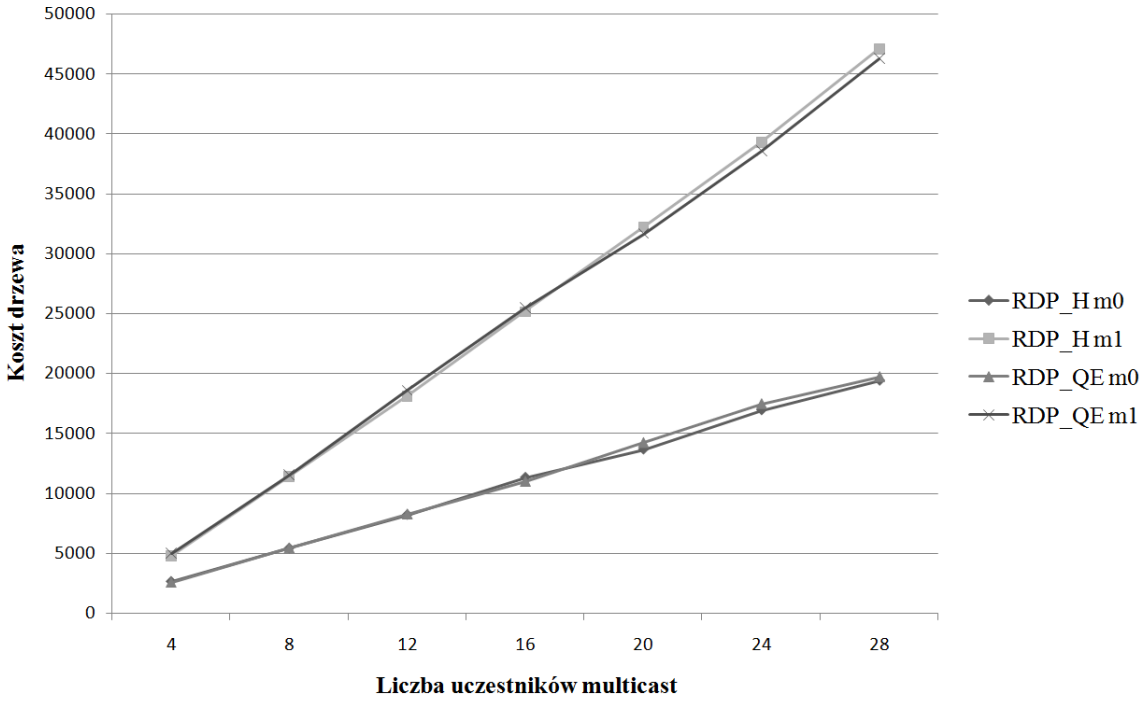
Wyniki zostały wygenerowane dla sieci o różnych rozmiarach: 50, 100 oraz 150 węzłów. Eksperyment był przeprowadzony dla 2, 3 oraz 4 metryk (odpowiednio 1, 2 oraz 3 ograniczenia) oraz dla zbiorów uczestników multicast o liczebnościach 4, 8, …, 28.

4.2. Prezentacja i omówienie wyników

Wyniki zgromadzone w opisanym powyżej eksperymencie zostały odpowiednio pogrupowane i zagregowane, w celu zbadania wpływu zmiennych symulacyjnych na wyniki działania algorytmu. Wyniki zbadano pod kątem jakości uzyskanych drzew multicast, jednak ze względu na tematykę artykułu skupiono się na porównaniu czasów wykonania odpowiednich procedur.

Żaden z wykresów nie przedstawia wszystkich omówionych wcześniej parametrów symulacyjnych. W sytuacji gdy dana zmienna nie została przedstawiona, należy przyjąć odpowiednio liczę węzłów równą 100, liczbę metryk równą 2 oraz implementację topologii za pomocą listy sąsiedztwa. Wielkości te zostały przyjęte na potrzeby prezentacji, ponieważ w zależności od wykresu albo nie miały wpływu na dane wyniki, albo (spośród większej liczby wyników) pozwalały na najlepsze przedstawienie rezultatów.

Na rys. 1 przedstawiono zależność pomiędzy wartościami poszczególnych metryk wyników a liczebnością grupy multicast oraz wybranym wariantem algorytmu. Można zauważyć, że wybór wariantu algorytmu nie miał wpływu na wyniki. Dla poszczególnych metryk otrzymano natomiast różne wartości, co jest spodziewanym efektem, ponieważ algorytm ma za zadanie minimalizować metrykę m_0 oraz maksymalnie zbliżyć metrykę m_1 do zadanego ograniczenia, co owocuje jej stosunkowo dużymi wartościami.

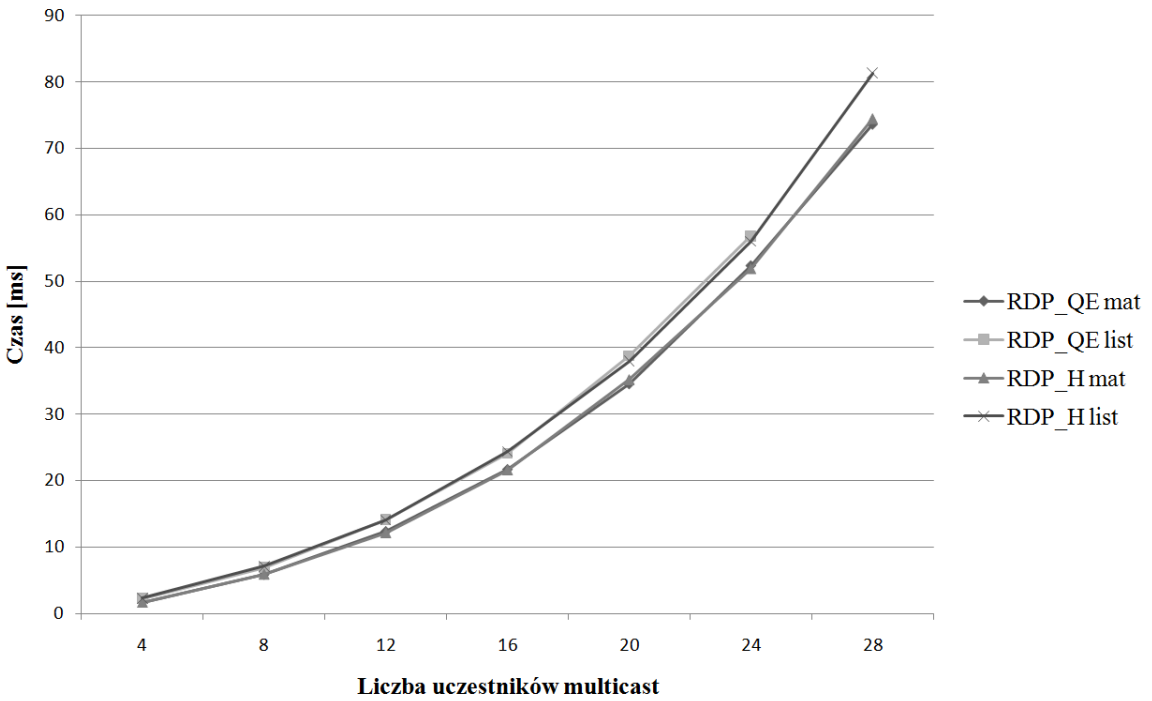


Rys. 1. Zależność kosztów m_0, m_1 drzew od liczebności grupy multicast dla dwóch wariantów algorytmu

Fig. 1. The result trees' costs for the metrics m_0, m_1 and for the two algorithm variants and different multicast group sizes

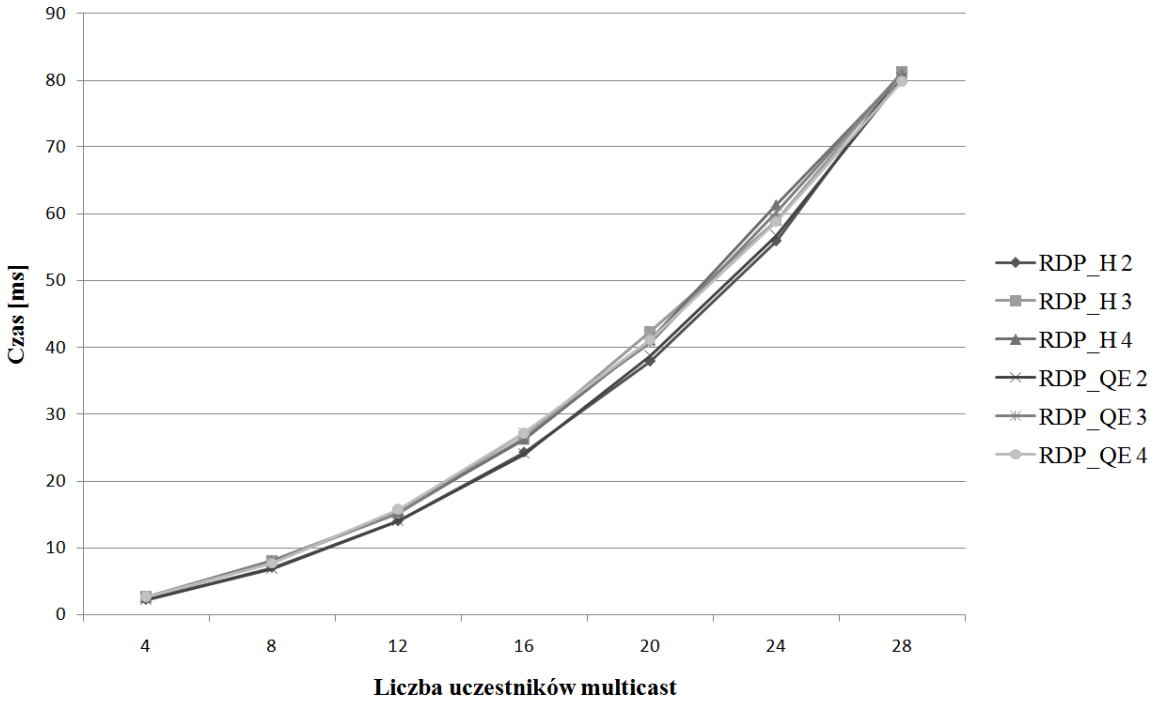
Na rys. 2 przedstawiono średnie wielkości czasu wyznaczania poszczególnych drzew multicast dla różnych implementacji topologii oraz takich samych pozostałych parametrów, jak w przypadku wyników z rys. 1. Dzięki niemu można zaobserwować, że obliczenia wykonywane były nieznacznie szybciej w przypadku wykorzystania macierzy sąsiedztwa, co jest spodziewanym wynikiem, gdyż ta implementacja oferuje większą wydajność dostępu do danych kosztem większego zużycia pamięci.

Rysunek 3 pozwala na zaobserwowanie interesującego zjawiska braku znaczącego wpływu liczby rozpatrywanych kryteriów na czas obliczeń. Oznacza to, że dla danych parametrów symulacyjnych na czas trwania obliczeń nie miał znaczącego wpływu ani wybór wariantu algorytmu, ani liczba rozpatrywanych kryteriów optymalizacji. Dzięki rys. 4 można zaobserwować przestrzenną zależność czasu obliczeń od liczebności grupy multicast oraz rozmiaru sieci. Dla sieci o 50 węzłach wpływ liczby uczestników grupy multicast jest niewielki i w przybliżeniu liniowy. Podobna jest zależność czasu obliczeń od rozmiaru sieci dla małych grup multicast. Natomiast wraz ze wzrostem liczebności grupy łączonych węzłów oraz rozmiaru sieci, można zaobserwować znaczący wzrost kosztu obliczeniowego oraz tendencję mierzonej zależności do przybierania charakteru wykładniczego.



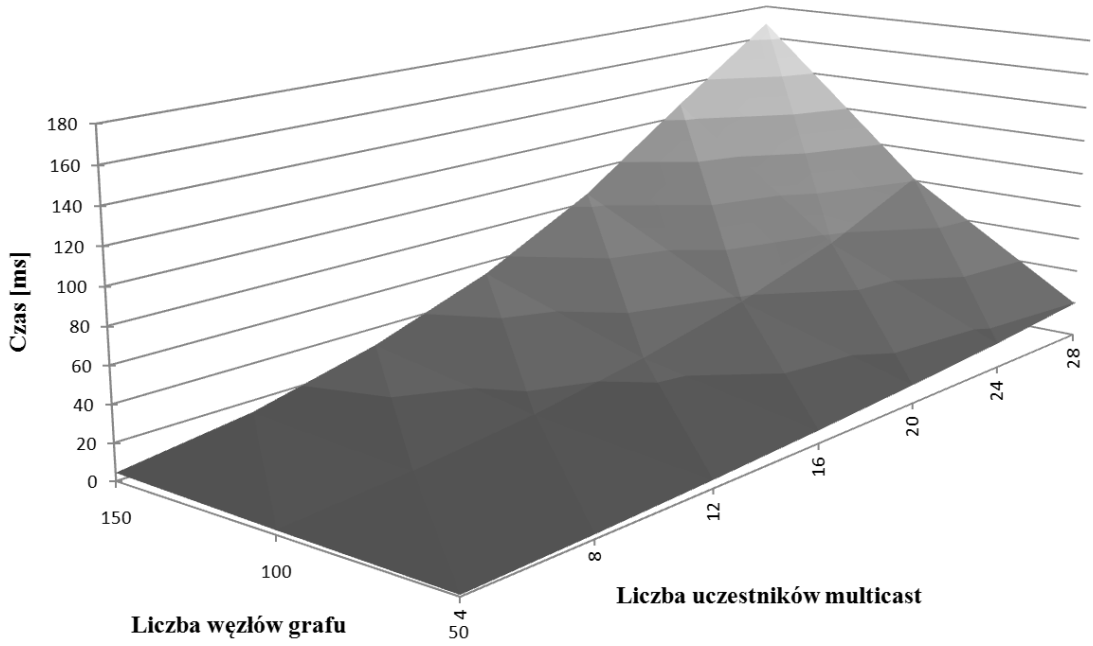
Rys. 2. Zależność czasów obliczeń od liczebności grupy multicast dla dwóch wariantów algorytmu i dwóch implementacji topologii

Fig. 2. The running times for the two algorithm variants and different multicast group sizes



Rys. 3. Zależność czasu obliczeń od liczebności grupy multicast dla dwóch wariantów algorytmu i różnych liczb kryteriów optymalizacji

Fig. 3. The running times for the different criteria counts, algorithm variants and multicast group sizes



Rys. 4. Zależność czasów obliczeń od liczebności grupy multicast i liczby węzłów w sieci

Fig. 4. The running times for the different graph sizes and different multicast group sizes

5. Wnioski

W artykule przedstawiono nowy algorytm wielokryterialnej optymalizacji trasowania transmisji rozgałęźnej. Szczególny nacisk położono na zagadnienia techniczne i algorytmiczne, związane z implementacją oraz analizą jakości algorytmu.

Zaproponowano dwa przykładowe rozwiązania niskopoziomowej implementacji struktur danych dla symulacji realizacji algorytmu oraz dwa warianty realizacji obliczeń. Istotność poruszonych zagadnień podparto wieloma wykresami przedstawiającymi wyniki badań symulacyjnych. Wykresy te ukazują zbiór wyników z różnych punktów widzenia, w celu zobrazowania różnych zjawisk, jakie można było zaobserwować w ramach przeprowadzonego eksperymentu.

Wyniki badań prowadzą do następujących wniosków. Ocena jakościowa uzyskanych wyników pozwala stwierdzić, że algorytm optymalizuje rozpatrywany problem zgodnie z oczekiwaniami. Nie zaobserwowano wpływu liczby rozpatrywanych kryteriów na czas symulacji, natomiast można było stwierdzić zauważalny, choć nieznaczny wpływ wybranej implementacji topologii na ten parametr. Można zaobserwować, że zarówno liczebność łączonej grupy multicast, jak i rozmiar sieci miały znaczący wpływ na czas symulacji, co jest zgodne z oczekiwaniami, ponieważ są to główne parametry wpływające na teoretyczną złożoność obliczeniową problemu.

BIBLIOGRAFIA

1. Ahuja R. K.: Network Flows. Prentice-Hall, Inc., Upper Saddle River, NJ, USA 1993.
2. Chen S, Nahrstedt K.: An overview of quality of service routing for next-generation high-speed networks: problems and solutions. IEEE Network 1998, nr 12, p. 64-79.
3. Gang F.: A multi-constrained multicast QoS routing algorithm. Computer Communications 2006, vol. 29, no. 10, p. 1811-1822.
4. Stachowiak K., Weissenberg J., Zwierzykowski P.: Lagrangian relaxation in the multicriterial routing. IEEE AFRICON 2011, Livingstone, Zambia, p. 1-6.
5. Stachowiak K., Zwierzykowski P.: Lagrangian Relaxation and Linear Intersection Based QoS Routing Algorithm. International Journal of Electronics and Telecommunications 2013, vol. 58, no. 4, p. 307-314.
6. Neve H.D., Mieghem P.V.: Tamcra: a tunable accuracy multiple constraints routing algorithm. Computer Communications 2000, vol. 23 no. 7, p. 667-679.

Wpłynęło do Redakcji 20 marca 2013 r.

Abstract

The multicriterial multicast routing algorithms consist in solving two major problems. First is the connection of the multiple network nodes with a multicast tree. The second one is the optimization of the multiple criteria. This kind of a mathematical problem reflects the real world problem of realizing the group transmission in the packet networks taking the Quality of Service requirements into account.

In the article a novel algorithm for solving this class of problems has been presented, with the emphasis put on the algorithmic aspects of its design, including the implementation details. The algorithm is based on the multiple instances of a Dijkstra's algorithm and proposes a management strategy, that allows merging their results into a multicast communication tree result. A general approach of the algorithm has been presented as well as two particular variants that represent it. Due to the nuances that distinguish the algorithm's realizations, an in-depth discussion of their implementation has been provided.

In order to show the impact of some of the implementation decisions, an extensive simulation based experiment has been conducted. The parameters of the experiment have been divided into two categories: the variable and the fixed parameters, and presented separately. Arbitrarily selected subset of the experiment's results has been presented in Figures 1-4. The results' presentation is followed by a general discussion of the observed phenomena and their explanation.

Adresy

Krzysztof STACHOWIAK: Politechnika Poznańska, Wydział Elektroniki i Telekomunikacji, ul. Polanka 3, 60-965 Poznań, Polska, krzysiek.stachowiak@gmail.com
Piotr ZWIERZYKOWSKI: Politechnika Poznańska, Wydział Elektroniki i Telekomunikacji, ul. Polanka 3, 60-965 Poznań, Polska.