ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2006
Seria: AUTOMATYKA z. 143 Nr kol. 1726

Paweł CZADERNA, Konrad WALA
Akademia Górniczo-Hutnicza w Krakowie

SYMULOWANE WYŻARZANIE W MAKSYMALIZACJI WAGI LIŚCI DRZEWA ROZPINAJĄCEGO

Streszczenie. W pracy przedstawiono model kombinatoryczny problemu maksymalizacji wagi liści drzewa rozpinającego grafu zwykłego. Szczegółowo omówiono trzy procedury generowania drzew sąsiednich do bazowego, które, w procesie poprawy drzewa, są eksploatowane przez klasyczny algorytm symulowanego wyżarzania.

SIMULATED ANNEALING IN MAXIMIZATION OF THE LEAFS WEIGHT OF SPANNING TREE

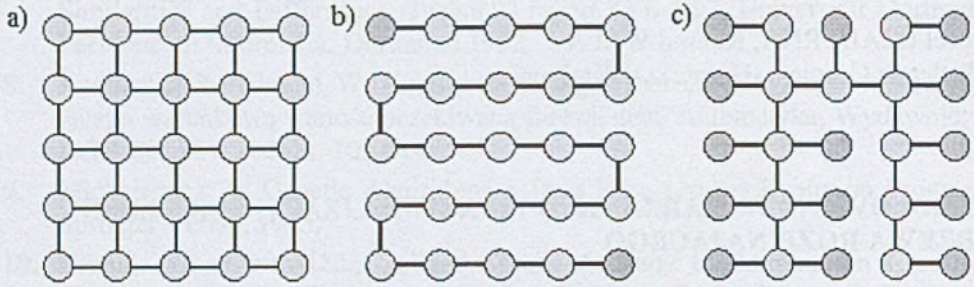
Summary. In the paper the combinatorial model of the weighted maximum leaf tree problem of ordinary graph is presented. We give a detailed description of three procedures of the neighboring trees generation to the basis one which are exploited during the course of tree improving process realized by the classical simulated annealing algorithm.

1. Wprowadzenie

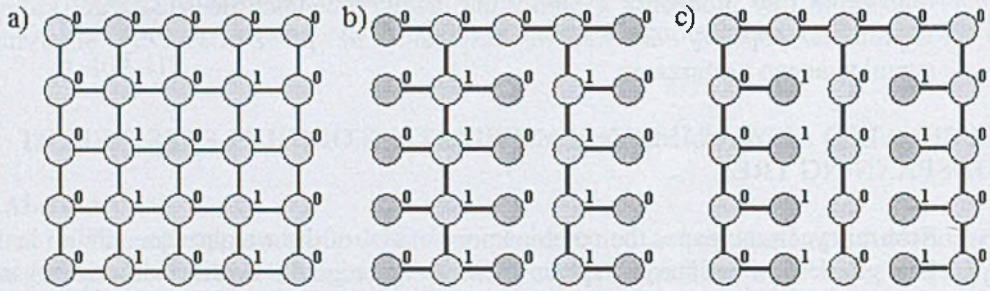
Pierwszym, który analizował problem minimalizacji urządzeń pośredniczących w transmisji pakietów w sieci komputerowej i sformułował go jako problem poszukiwania w grafie zwykłym drzewa rozpinającego z maksymalną liczbą liści, (problem MLSTP (ang. maximum leaf spanning tree problem)), był Dijkstra (por. [4]). Problem MLSTP leży w kręgu zainteresowań badań teoretycznych, dotyczących określonych własności grafów jak też praktycznych, przy projektowaniu połączeń w układach scalonych, sieciach telekomunikacyjnych i komputerowych (por., [1], [3], [4], [6]). Naturalnym uogólnieniem problemu MLSTP jest problem WMLSTP (ang. weighted maximum leaf spanning tree problem), gdzie każdy z wierzchołków posiada przypisaną wagę, która informuje o przydatności wierzchołka do pełnienia roli liścia w drzewie. Lu i Ravi [5] wykazali, że są to problemy NP-trudne.

Różnice, występujące pomiędzy problemami MLSTP i WMLSTP ilustrują przykłady zamieszczone na rysunkach 1 i 2. Dla grafu z rysunku 1a) drzewo rozpinające z maksymalną liczbą liści (14) jest przedstawione na rysunku 1c), podczas gdy drzewo z rysunku 1b) posiada tylko 2 liście. Dla grafu ważonego z rysunku 2a)

maksymalne drzewo rozpinające o wadze 8 jest przedstawione na rysunku 2c), natomiast drzewo z rysunku 2b) ma, pomimo większej liczby liści, wagę równą 0.



Rys. 1. Przykład rozwiązania problemu MLSTP



Rys. 2. Przykład rozwiązania problemu WMLSTP

W rozdziale 2. opisano modele kombinatoryczne problemów WMLSTP i MLSTP. Jako algorytm popraw drzewa rozpinającego zastosowano klasyczny (stąd pominięcie jego opisu w pracy) algorytm symulowanego wyżarzania SA (ang. simulated annealing), gdzie najważniejszą składową algorytmu jest przepis generacji rozwiązań sąsiednich do rozwiązania bazowego, tj. rozwiązania poprawianego w bieżącej iteracji. Po licznych próbach obliczeniowych do generacji drzew sąsiednich drzewa bazowego zastosowano trzy, opisane w rozdziale 3., procedury losowej generacji drzew. W rozdziale 4. przedstawiono wybrane z pracy [2] wyniki badań komputerowych algorytmu SA.

2. Model kombinatoryczny problemu

Dany jest zwykły graf spójny $G = (V, E)$ i funkcja $w: V \to R^+ \cup \{0\}$, określająca wagę w_j każdego wierzchołka $j \in V$, gdzie $V = \{1, ..., j, ..., n\}$ – zbiór wierzchołków grafu, E – zbiór krawędzi, $|E| = m > n\text{-}1$, $|E|$ – liczba elementów zbioru E. Przyjęto następujące oznaczenia:

$T = (V, E_T)$ – drzewo rozpinające w grafie G,

gdzie: $(E_T \subset E) \wedge (|E_T| = n - 1) \wedge (T$ – graf acykliczny),

$N(i) = \{j: \{i,j\} \in E\}$ – zbiór wierzchołków sąsiednich do wierzchołka $i \in V$ w G,

$\delta_G(i) = \{\{i,j\}: \{i,j\} \in E\}$ – zbiór krawędzi incydentnych do wierzchołka $i \in V$ w G,

$\delta_T(i) = \{\{i,j\}: \{i,j\} \in E_T\}$ – zbiór krawędzi incydentnych do wierzchołka $i \in V$ w T.

W problemie WMLSTP poszukiwane jest drzewo rozpinające T^* w grafie G, w którym suma wag liści jest maksymalna, tj.:

$$\sum_{j \in L_T} w_j \rightarrow \max$$

gdzie: $L_T = \{ i \in V: |\delta_T(i)| = 1 \}$ – zbiór liści drzewa T;
natomiast w problemie MLSTP poszukiwane jest drzewo rozpinające z maksymalną liczbą liści: $|L_T| \rightarrow \max$.

Górne ograniczenie wartości funkcji celu UB (ang. upper bound) w problemie WMLSTP, poprzez optymalną wartość funkcji celu dla drzewa rozpinającego w grafie pełnym, jest następujące:

$$\sum_{j \in L_T} w_j \leq UB = \sum_{j=1}^{n} w_j - \min_j w_j$$

podczas gdy w problemie MLSTP: $UB = n$ -1 (liczba liści w drzewie typu gwiazda).

3. Procedury generacji drzew sąsiednich

W procesie poprawy rozwiązania bazowego poszukiwane jest rozwiązanie lepsze w jego sąsiedztwie i podstawowym problemem jest sposób oraz koszt generacji rozwiązań sąsiednich do bazowego. Można tu zastosować „proste i tanie" reguły generacji na zasadzie „swobodnych" modyfikacji rozwiązania bazowego i ponosić znaczne koszty sprawdzania dopuszczalności generowanych rozwiązań. Sprawdzanie dopuszczalności rozwiązania problemu WMLSTP jest zadaniem czasochłonnym, zwłaszcza gdy okazuje się, że większość z generowanych w ten sposób rozwiązań nie jest drzewem i trzeba je po prostu naprawiać lub odrzucać. Po negatywnych wynikach prób, podejście to odrzucono, decydując się na metody generacji drzew, polegające na usuwaniu krawędzi drzewa bazowego i wstawianiu nowych w sposób kosztowniejszy, ale kontrolowany. W opisanych poniżej procedurach proces generowania drzew sąsiednich wygląda następująco: (i) usuwana jest wybrana krawędź lub krawędzie z drzewa bazowego, (ii) drzewo bazowe traci spójność i w wyniku tego powstają składowe spójne $T_1, T_2, ..., T_k$ drzewa, które są identyfikowane za pomocą standardowego algorytmu przeszukiwania grafu DFS (ang. depth-first search), (iii) następuje łączenie składowych za pomocą wybranych krawędzi w nowe drzewo rozpinające. Złożoność obliczeniowa tych kroków w istotny sposób wpływa na efektywność algorytmu popraw.

Na podstawie przeglądu rozwiązań, znajdujących się w literaturze (np.[1], [5]) i badań własnych opracowano i przetestowano trzy procedury generowania losowego drzew sąsiednich do poprawianego drzewa bazowego.

Procedura 1-change. Niech $V' = \{ i \in V: |\delta_T(i)| > 1 \}$ jest zbiorem wierzchołków drzewa T, które nie są liśćmi. Wyznaczamy z rozkładem losowo równomiernym wierzchołek i ze zbioru V', oznaczenie stosowane w pracy: $i = random(V')$ oraz, także losowo, krawędź $\{i, j\}$, gdzie $j = random(\{k \in N(i): \{i, k\} \in \delta_T(i)\})$. Po usunięciu krawędzi $\{i, j\}$ z drzewa bazowego otrzymujemy dwie rozłączne składowe T_1 i T_2 drzewa bazowego T, gdzie identyfikacja wierzchołków przynależnych do składowych $V(T_1)$, $V(T_2)$ jest wykonana za pomocą algorytmu DFS, w czasie $O(n+m_T)$,

ponieważ dla drzew liczba krawędzi $m_T = n$ -1, stąd w tym przypadku złożoność DFS jest O(n). Wierzchołek $i \in V(T_1)$ łączymy z losowo wybranym wierzchołkiem v=*random*($V(T_2) \cap N(i)$), krawędzią $\{i, v\}$, gdzie $v \neq j$. Biorąc pod uwagę złożoność obliczeniową O(n) procedur losowania wierzchołków i algorytmu DFS, złożoność obliczeniowa wygenerowania jednego z $2m$-1 drzew sąsiednich do bazowego jest O(n).

Procedura 1-exchange. Identycznie jak w procedurze *1-change* usuwamy w czasie O(n) losowo wyznaczoną krawędź $\{i, j\}$ i otrzymujemy dwa rozłączne podzbiory wierzchołków $V(T_1)$ i $V(T_2)$. Następnie wyznaczamy zbiór krawędzi $U = \{\{v, \mu\}: (v \in V(T_1)) \wedge (\mu \in V(T_2)) \wedge (\{v, \mu\} \in E)\}$, które mogą połączyć składowe T_1 i T_2 w drzewo. Następnie ze zbioru U wybieramy losowo krawędź $\{k, l\}$ = *random*(U) zastępując nią dopiero co usuniętą. Czynność wyznaczenia krawędzi $\{k, l\}$ ma złożoność O(m), stąd losowa generacja jednego z $m(n$ -1) drzew sąsiednich ma złożoność obliczeniową O(n+m).

Procedura k-change. Niech $V' = \{i \in V: |\delta_T(i)| > 1\}$ jest zbiorem wierzchołków drzewa bazowego T, które nie są liśćmi. Wyznaczamy losowo wierzchołek i = *random*(V') i usuwając wszystkie krawędzie incydentne do i w drzewie T, tj. krawędzie zbioru $\delta_T(i)$, otrzymujemy składowe $T_1, T_2, ..., T_k$; $2 \le k \le n$, drzewa bazowego, gdzie także identyfikacja wierzchołków przynależnych do składowych $V(T_1), V(T_2), ..., V(T_k)$ jest wykonana w czasie O(n) przez algorytm DFS. Następnie w czasie O(n), dla $\mu = 1, 2, ..., k$ wyznaczane są losowo krawędzie $\{i, j_\mu\}$, łączące izolowany wierzchołek i ze składowymi $T_1, ..., T_\mu, ..., T_k$ w drzewo sąsiednie, gdzie j_μ = *random* ($V(T_\mu)$).

Warto odnotować, że w przypadku równości $\delta_T(i) = \delta_G(i)$ dla wylosowanego $i \in V'$ procedury *1-change* i *k-change* wyznaczają ponownie drzewo bazowe, co może się często zdarzać w przypadku grafu rzadkiego. W takim przypadku warto sprawdzać wymieniony warunek i przerywać obliczenia.

W badanym algorytmie symulowanego wyżarzania próba poprawy drzewa bazowego polega na zastosowaniu, z prawdopodobieństwem 1/3, jednej z opisanych procedur i sprawdzeniu jakości wylosowanego drzewa sąsiedniego.

4. Wybrane wyniki badań i uwagi końcowe

Dla klasycznego algorytmu SA i zaproponowanych procedur generacji drzew sąsiednich przeprowadzono badania numeryczne, język C++, procesor: 1,5 GHz i 1 GB pamięci operacyjnej, w celu określenia relacji czasowych w zależności od rozmiaru problemu i jakościowych w stosunku do wyników uzyskanych przez heurystyki konstrukcyjne. Losowo wygenerowano instancje problemów dla liczby wierzchołków grafu n = 25, 50, 75, 100, 125, 150, 175, 200, 225 i 250, przy czym wagi wierzchołków wybrano losowo ze zbioru { 1, 2, ..., 5}. Dla każdego n utworzono 50 instancji testowych (20 – grafy typu siatka, 10 – grafy kubiczne, 20 – grafy z losowymi połączeniami), a dla każdej instancji testowej wykonano 10 procesów optymalizacji z użyciem tych samych parametrów.

Badany algorytm SA okazał się metodą optymalizacji, która w stosunkowo szybkim czasie znajduje rozwiązania dla dużych instancji. W tabeli 1 przedstawiono przykładowe zależności czasu obliczeń, średni czas w sekundach, od rozmiaru

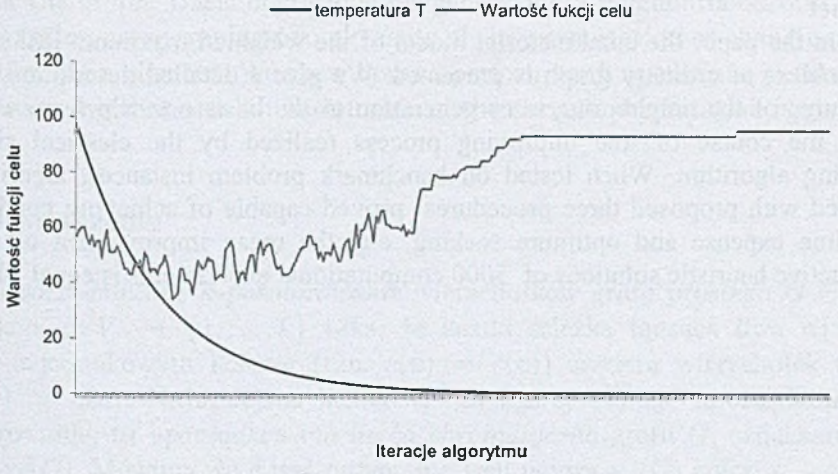
problemu n i liczby iteracji I algorytmu dla stałej temperatury początkowej = 100 i stałej liczby prób w jednej iteracji = 20.

Tabela 1

Średni czas obliczeń algorytmu SA

n/I	100	500	1000	1500	2000
25	0,09	0,41	0,80	1,29	1,61
50	0,17	0,78	1,50	2,25	2,98
75	0,23	1,14	2,22	3,21	4,39
100	0,31	1,48	2,90	4,32	5,73
125	0,39	1,84	3,61	5,39	7,12
150	0,45	2,19	4,28	6,42	8,50
175	0,54	2,58	5,02	7,51	10,00
200	0,61	2,92	5,73	8,56	11,39
225	0,72	3,31	6,52	9,61	14,19
250	0,77	3,98	7,14	11,80	15,62

Średnia poprawa rozwiązań przez algorytm SA w stosunku do najlepszych uzyskanych przez 90 badanych heurystyk konstrukcyjnych (por. [2]) wynosi 14,8 %. Jest to wartość, która pozwala na określenie algorytmu SA z zaproponowanymi trzema procedurami generacji rozwiązań sąsiednich, metodą skuteczną w znajdowaniu rozwiązań dobrych jakościowo. Ponadto algorytm cechuje się bardzo dobrym wskaźnikiem poprawy losowych rozwiązań początkowych, który dla instancji testowych wynosi średnio 47%. Przykładowy przebieg procesu optymalizacji przedstawiono na rysunku 3, gdzie także zaznaczono spadek temperatury podczas procesu. Zauważmy, że proces poszukiwania z podzbioru rozwiązań dobrych w sąsiedztwie rozwiązania początkowego przemieścił się przez „obszar" rozwiązań gorszych do podzbioru rozwiązań lepszych.



Rys. 3. Przykład przebiegu procesu optymalizacji za pomocą algorytmu SA

W zakończeniu podkreślamy, że z jednej strony zaproponowane procedury posiadają małą złożoność obliczeniową jednego kroku $O(n)$ lub $O(n+m)$ umożliwiając realizację procesu poprawy drzew dużego rozmiaru, a z drugiej, są skutecznym narzędziem do wykonania procesu popraw, ponieważ zapewniają wystarczające różnicowanie procesu poszukiwania nowych drzew.

LITERATURA

1. Ahuja R.K., Orlin J.B., Sharma D: Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. Working Paper, University of Florida, 2001. *Submitted for publication.*
2. Czaderna P.: Algorytmy przybliżone optymalizacji sieci komputerowej modelowanej jako problem MLSTP, dyplomowa praca magisterska (promotor: K. Wala). Wydział EAIiE Akademii Górniczo-Hutniczej, Kraków 2006.
3. Fernandes L.M., Gouveia L.: Minimal spanning trees with a constraint on the number of leaves. European J. of Operational Research, Vol.104, 1998, p. 250-261,.
4. Loryś K., Zwoźniak G.: Approximation algorithms for Maximum-leaf Spanning Tree for cubic graphs. Mohring R. and Raman R. (Eds), ESA 2002, LNCS2461, Springer-Verlag 2002, p. 686-698.
5. Lu H., Ravi R.: The power of local optimization: Approximation algorithms for Maximum-leaf Spanning Tree. Proc. of the Thirtieth Annual Allerton Conf. on Communication, Control and Computing, 1992, p. 533-542.
6. Lu H., Ravi R.: A near-linear-time approximation algorithms for Maximum-leaf Spanning Tree. Journal of Algorithms, Vol. 29, No 1, 1995, p. 132-141.

Recenzent: Prof. dr hab. inż. Marek Kubale

Abstract

In the paper the combinatorial model of the weighted maximum leaf spanning tree problem of ordinary graph is presented. We give a detailed description of three procedures of the neighboring trees generation to the basis one which are exploited during the course of tree improving process realized by the classical simulated annealing algorithm. When tested on benchmark problem instances, algorithm SA equipped with proposed three procedures proved capable of achieving good results, both time expense and optimum seeking, e.g. the mean improvement of the best constructive heuristic solutions of 5000 computational experiments is equal 14,8%.