ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1984
Seria: AUTOMATYKA z. 74 Nr kol. 810

Eugeniusz Toczyłowski
Witold Mikoda

Instytut Automatyki
Politechnika Warszawska

PEWNA REALIZACJA RÓWNOLEGŁYCH ALGORYTMÓW DLA ZADAŃ MAKSYMALNEGO PRZYDZIAŁU

Streszczenie. W pracy przedstawiono oryginalną realizację algorytmów równoległych dla zadań maksymalnego przydziału łączącą praktyczne zalety szeregowego algorytmu Halla z teoretycznymi właściwościami algorytmów równoległych.

1. Wprowadzenie

Przypuśćmy, że dany jest dwudzielny graf $G=(X,Y,E)$, gdzie X oraz Y są zbiorami wierzchołków oraz zbiór E, $|E|=\tau$ jest zbiorem krawędzi $\{i,j\}$ takich, że $i \in X$ oraz $j \in Y$. Dla ustalenia uwagi można przyjąć interpretację, że zbiór X, $|X| = m$ odpowiada zbiorowi zadań do wykonania, zbiór Y, $|Y| = n$ odpowiada stanowiskom obsługi /lub pracownikom/ mogącym wykonywać te zadania, przy czym krawędź $\{i,j\} \in E$ oznacza, że i-te zadanie może być przydzielone do wykonania na j-tym stanowisku. Zbiór krawędzi $M \subseteq E$ jest skojarzeniem w grafie G, jeżeli każdy wierzchołek ze zbioru $X \cup Y$ jest incydentny co najwyżej z jedną krawędzią z M. Zadanie wyznaczenia maksymalnego przydziału rozłącznych zadań do stanowisk obsługi polega na znalezieniu w grafie G skojarzenia o największej liczności. Zadanie to ma szerokie zastosowania [1],[4],[6]. Algorytmy dla wielkich zadań przydziału znajdują szczególne zastosowanie m.in. w algorytmach sterowania w systemach transportowych oraz w technikach rozrzedzonych macierzy [3],[9].

Prosta droga w grafie G z nieparzystą liczbą krawędzi nazywana jest ścieżką naprzemienną względem skojarzenia M, jeżeli jej parzyste krawędzie należą do M natomiast krawędzie nieparzyste należą do $E-M$. Wierzchołek $v \in X \cup Y$ nazywany jest swobodnym, jeżeli nie jest on incydentny z krawędziami skojarzenia M. Ścieżka naprzemienna nazywana jest ścieżką powiększającą, jeżeli jej wierzchołki początkowy i końcowy są swobodne. Dla danego skojarzenia M można, jeżeli M nie jest skojarzeniem o liczności maksymalnej, znaleźć skojarzenie N o liczności większej wykorzystując do tego ścieżki naprzemienne [5],[7]. Jeżeli dla danego skojarzenia M istnieje ścieżka powiększająca P rozumiana jako zbiór krawędzi , to zbiór $N = M \oplus P$ jest skojarzeniem o liczności większej o 1.

Wszystkie dotychczas znane algorytmy maksymalnego skojarzenia startują z pewnego skojarzenia M, które nie musi być maksymalnym i konstruują, jeżeli istnieje, skojarzenie o większej liczności dzięki znalezieniu

ścieżek powiększających względem skojarzenia M. Rozróżniamy zasadniczo dwie grupy algorytmów : szeregowe i równoległe. Algorytmy szeregowe, których podstawowym reprezentantem jest algorytm Halla [4], polegają na wyznaczeniu w każdej iteracji pojedynczej ścieżki powiększającej /w czasie $O(\tau)$/a tym samym liczność skojarzenia zwiększa się o jeden. Łączna złożoność obliczeniowa algorytmów szeregowych jest równa $O(n \cdot \tau)$. Algorytmy równoległe Hopcrofta-Karpa (HK) [5] i Toczyłowskiego (T) [8] polegają na wyznaczeniu w każdej iteracji zbiorów ścieżek powiększających /również w czasie $O(\tau)$/ . Maksymalna liczba iteracji w algorytmie Hopcrofta-Karpa jest równa $O(\sqrt{n})$, dla algorytmu T oszacowanie liczby iteracji nie jest gorsze od oszacowania dla algorytmu HK i jak pokazują wyniki obliczeń, w praktyce liczba ta jest znacznie mniejsza.

Realizacje numeryczne algorytmów równoległych są znacznie bardziej skomplikowane niż realizacje algorytmów szeregowych i z tego powodu nie pozwalają na uwidocznienie się teoretycznej przewagi algorytmów równoległych dla zadań o mniejszym wymiarze /a często nawet dla zadań z liczbą n rzędu kilkuset/. W pracy przedstawiono nową realizację algorytmów równoległych wymagającą $2\tau + 4(n+m)$ jednostek pamięci, przypominającą numeryczne realizacje algorytmów szeregowych /przez dokonywanie w każdej iteracji pojedynczego przeszukiwania pierwotnego grafu wszerz i wgłąb/ bez konieczności kosztownej i pamięciochłonnej konstrukcji dodatkowego grafu ścieżek powiększających, np. realizowanej w pracach [5], [2]. Istota metody polega na rozbiciu każdej iteracji algorytmów równoległych na dwie fazy. Faza pierwsza nazywana jest fazą etykietowania wierzchołków grafu i polega na rozszczepieniu zbioru wierzchołków grafu G na warstwy /przez jednokrotne i ograniczone przeszukanie wszerz grafu G / a jednocześnie pozwala na określenie porządku w zbiorze wierzchołków swobodnych ze zbioru Y, według którego będą wyszukiwane ścieżki powiększające. Faza druga polega na znajdowaniu zbioru wierzchołkowo rozłącznych ścieżek powiększających. W algorytmie HK wyszukiwane są jedynie ścieżki powiększające o najkrótszej długości, natomiast w algorytmie T dodatkowo wyszukiwany jest maksymalny zbiór ścieżek powiększających o większej długości. Opis realizacji algorytmów równoległych zostanie przedstawiony tylko dla algorytmu T, ponieważ algorytm HK można utworzyć z algorytmu T przez pominięcie niektórych kroków w algorytmie T. Podstawy teoretyczne konstrukcji algorytmów maksymalnego skojarzenia można znaleźć w książce [7].

2. Opis algorytmu

2.1. Reprezentacja grafu dwudzielnego

Najbardziej wygodnym, z punktu widzenia obliczeń, sposobem pamiętania grafu dwudzielnego $G = (X,Y,E)$ jest pamiętanie dla każdego wierzchołka $v \in X \cup Y$ zbioru indeksów wierzchołków przyległych do v. Ten zapis wymaga

łącznie $m + n + 2\tau$ jednostek pamięci, w tym $m + n$ jednostek pamięci na adresy pierwszych elementów zbiorów wierzchołków przyległych oraz 2τ jednostek na indeksy wszystkich wierzchołków przyległych.

2.2. Zapis skojarzenia M

Aktualne skojarzenie $M \subseteq E$ będące podzbiorem krawędzi grafu G pamiętane jest w dwojaki sposób:

(i) dla każdego wierzchołka $j \in Y$ pamiętany jest indeks wierzchołka $i = COTN(j)$ takiego, że $\{x_i, y_j\} \in M$ /jeżeli wierzchołek $j \in Y$ jest swobodny, to przyjmujemy wartość $i > m$/;

(ii) dla każdego wierzchołka $x_i \in X$ pamiętany jest indeks wierzchołka $j = ROTN(i)$ takiego, że $\{x_i, y_j\} \in M$ /jeżeli wierzchołek $x_i \in X$ jest swobodny, to $j = 0$/.

Łączna zajętość pamięci do zapisu skojarzenia M wynosi $m + n$.

2.3. Opis jednej iteracji algorytmu

Dla danego skojarzenia M, jeżeli nie jest ono skojarzeniem maksymalnym, jedna iteracja algorytmu polega na znalezieniu maksymalnego zbioru wierzchołkowo rozłącznych ścieżek powiększających $P_1, P_2, \ldots, P_l$, a następnie na wykonaniu operacji $N := M \oplus P_1 \oplus P_2 \oplus \ldots \oplus P_l$, w rezultacie czego uzyskiwane jest skojarzenie N o liczności powiększonej o 1. Znajdowanie zbioru ścieżek powiększających realizowane jest w dwóch fazach.

2.3.1. Faza etykietowania wierzchołków

Niech $X_n \subseteq X$ będzie zbiorem wierzchołków swobodnych względem skojarzenia M. Wierzchołkom ze zbioru X_n będziemy przypisywać etykietę n i będziemy umieszczać je w warstwie n-tej wierzchołków. W kolejnych krokach $k = n, n-1, \ldots$ będziemy konstruować pewne rozłączne podzbiory X_k, Y_k wierzchołków z X, Y umieszczonych w warstwie k-tej wierzchołków, którym będziemy przypisywać etykietę k. Zasada konstrukcji tych podzbiorów jest następująca. Kolejno dla $k = n, n-1, \ldots$ zakładając, że znane są $X_n, Y_n, X_{n-1}, Y_{n-1}, \ldots, X_k$, tworzymy podzbiory

$$E_k = \{\{x,y\} : \{x,y\} \in E,\ x \in X_k,\ y \notin Y_n \cup \ldots \cup Y_{k+1}\}$$

$$Y_k = \{y : \{x,y\} \in E_k\}$$

$$Y_k^o = Y_k \cap \{\text{zbiór wierzchołków swobodnych}\}$$

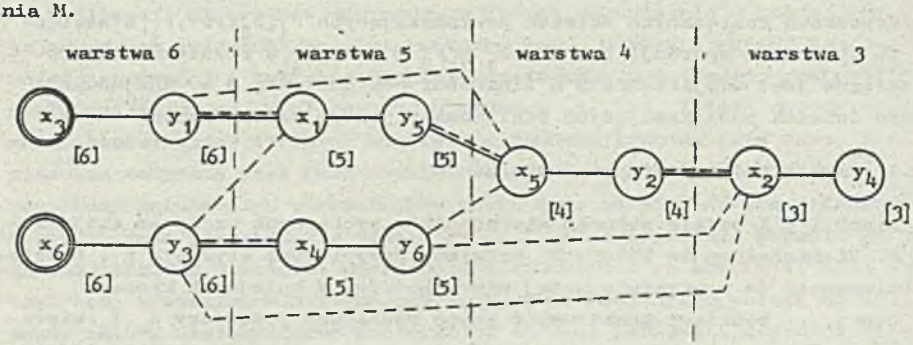
$$X_{k-1} = \{x : \{y,x\} \in M,\ y \in Y_k \setminus Y_k^o\}$$

Niech $k_m = \max\{k : Y_k^o \neq 0\}$, $k_M = \min\{k : Y_k^o \neq 0\}$. Indeks k_m jest etykietą pierwszej warstwy wierzchołków, w której pojawiły się wierzchołki swobodne ze zbioru Y, natomiast k_M jest etykietą ostatniej warstwy zawierającej takie wierzchołki.

Rozważmy przykład grafu dwudzielnego o macierzy przyległości wierzchołków

$$
\begin{array}{c|cccccc|}
 & y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \\
\hline
x_1 & 1 & 0 & 1 & 0 & 1 & 0 \\
x_2 & 0 & 1 & 1 & 1 & 0 & 1 \\
x_3 & 1 & 0 & 0 & 0 & 0 & 0 \\
x_4 & 0 & 0 & 1 & 0 & 1 & 1 \\
x_5 & 1 & 1 & 0 & 0 & 1 & 1 \\
x_6 & 0 & 0 & 1 & 0 & 0 & 0 \\
\hline
\end{array}
\qquad /1/
$$

Załóżmy, że $M = \{\{x_1,y_1\}, \{x_2,y_2\}, \{x_4,y_3\}, \{x_5,y_5\}\}$. Proces etykietowania wierzchołków prowadzi do konstrukcji następującego warstwowego podgrafu grafu G /rys.1/. Jeżeli krawędzie warstwowego podgrafu łączą wierzchołki $\{x,y\}$ tej samej warstwy, to należą do zbioru E-M, natomiast krawędzie łączące wierzchołki warstw sąsiednich są krawędziami skojarzenia M.



Objaśnienia

- wierzchołki swobodne
[ ] - wartość etykiety wierzchołka
krawędzie należące do skojarzenia M
krawędzie należące do E_k
krawędzie pozostałe /nie należące do grafu warstwowego/

Rys.1. Etykietowanie wierzchołków grafu G o macierzy przyległości /1/

Łatwo zaobserwować, że proces etykietowania polega na częściowym, jednokrotnym przeszukaniu grafu G wszerz /tylko z wierzchołków $x \in X$/. W trakcie przeszukiwania grafu można uzyskać również listę wierzchołków swobodnych ze zbioru Y, uporządkowaną w kolejności pojawiania się ich w kolejnych warstwach grafu. Przeszukiwanie wszerz grafu wymaga dostępu do zbioru

krawędzi grafu G tylko wtedy, gdy aktualnie rozpatrywany wierzchołek należy do zbioru X, przy czym dla aktualnej krawędzi $\{x,y\} \in E-M$ wierzchołek y można umieścić w danej warstwie tylko wtedy, gdy nie pojawił się on wcześniej. Fakt pojawienia się wierzchołka $y_j \in Y$ jest odnotowywany w tablicy COTN na pozycji j-tej, przez wstawienie liczby zero dla wierzchołka już skojarzonego lub numeru warstwy /ze znakiem minus / dla wierzchołka swobodnego.

Do realizacji drugiej fazy algorytmu zachodzi potrzeba pamiętania etykiet jedynie dla wierzchołków zbioru X oraz etykiet wierzchołków swobodnych ze zbioru Y. Ponadto konieczne jest zapamiętanie listy wierzchołków swobodnych ze zbioru Y. Łącznie potrzeba na to m + n jednostek pamięci przy następującej organizacji obliczeń. Niech dane będą tablice NUMBER(m) oraz LFC(n). W trakcie pojawiania się i etykietowania nowych wierzchołków k-tej warstwy, dla wierzchołka $x_i \in X_k$ odpowiadający im numer warstwy wstawiamy na pozycji NUMBER(i) natomiast po pojawieniu się nowego wierzchołka $y_j \in Y_k$ postępujemy dwojako. Jeżeli y_j jest wierzchołkiem już skojarzonym, tzn. i = COTN(j) spełnia warunek $0 < i \leq m$, to numer wierzchołka x_i skojarzonego z y_j wstawiamy na kolejną wolną pozycję tablicy LFC /wierzchołek x_i zostaje przydzielony do warstwy niższej o numer/. Jeżeli y_j jest wierzchołkiem swobodnym, tzn. i = COTN(j) > m, to numer wierzchołka y_j wstawiamy ze znakiem minus na kolejną wolną pozycję tablicy LFC. W obu przypadkach modyfikujemy wartość COTN(j) wstawiając odpowiednio Ø lub -k. Dalsze przeszukiwanie grafu wszerz polega na wybieraniu z tablicy LFC kolejnych wierzchołków x_i i przeglądzie listy wierzchołków do nich przyległych, aż do wyczerpania wszystkich wierzchołków w tablicy LFC.

W rezultacie wynikiem obliczeń fazy etykietowania wierzchołków jest tablica LFC zawierająca uporządkowane w kolejności pojawiania się indeksy kolumn swobodnych /mających znak minus/. Etykiety wierzchołków ze zbioru X są umieszczone w tablicy NUMBER a etykiety wierzchołków swobodnych ze zbioru Y w tablicy COTN.

2.3.2. Faza wyszukiwania ścieżek powiększających

Faza ta polega na znalezieniu zbioru wierzchołkowo rozłącznych ścieżek powiększających. Poszukiwanie ścieżek można przeprowadzić startując z kolejnych wierzchołków swobodnych $y \in Y$ umieszczonych w liście wierzchołków swobodnych /zapisanej w tablicy LFC / a następnie przeprowadzając poszukiwanie w głąb grafu G w kierunku niemalejących etykiet warstw wierzchołków grafu G w taki sposób, że krawędzie $\{y,x\}$ ze zbioru E-M łączą wierzchołki tej samej warstwy, natomiast krawędzie $\{x,y\} \in M$ łączą wierzchołki warstw sąsiednich. Przeszukiwanie w głąb grafu wymaga dostępu do zbioru krawędzi grafu G tylko wtedy, gdy aktualny wierzchołek należy do zbioru Y, przy czym dla aktualnej krawędzi $\{x,y\} \in E-M$ dla spełnienia warunku niemalejących etykiet warstw należy sprawdzić, czy etykieta wierzchołka x jest

większa lub równa etykiecie wierzchołka y i w przypadku niepomyślnym krawędź $\{y,x\}$ należy odrzucić. Ze sposobu etykietowania wierzchołków w fazie pierwszej wynika, że test pomyślny jest spełniany wyłącznie wtedy, gdy krawędzie $\{y,x\} \in E-M$ łączą wierzchołki tej samej warstwy. Po napotkaniu w trakcie poszukiwań w głąb grafu na wierzchołek $x \in X$ nie należący do warstwy n-tej przechodzimy po krawędzi skojarzenia $\{x,y\} \in M$ do warstwy wierzchołków o numer większej i kontynuujemy poszukiwania aż do natrafienia na wierzchołek swobodny $x \in X_n$ albo też do stwierdzenia, że taki wierzchołek nie istnieje. W obydwu przypadkach wszystkie wierzchołki pośrednie $x \in X_k$ są zakazane /np. przez przypisanie im w tablicy NUMBER etykiety [0]/, aby nie powtórzyć zbędnych przeszukiwań tej części grafu w trakcie wyznaczania ścieżek powiększających z pozostałych wierzchołków swobodnych z Y.

Przeszukiwanie w głąb grafu w celu wyznaczenia aktualnej ścieżki powiększającej realizowane jest za pomocą stosu FATHER(m) o głębokości nie przekraczającej liczby warstw, czyli $n-k_M+1 \leq m$. Stos ten służy do zapisu kolejnych wierzchołków $y_0, y_1, \ldots, y_i$ /począwszy od pewnego wierzchołka swobodnego y_0/ pojawiających się w trakcie poszukiwań w głąb grafu G, przy czym y_i = FATHER(i+1) jest numerem wierzchołka y_i będącego bezpośrednim poprzednikiem wierzchołka y_{i+1} na stosie. Po znalezieniu ścieżki powiększającej P możliwa jest natychmiastowa zamiana krawędzi skojarzenia należących do P, gdyż dzięki zakazaniu wszystkich wierzchołków pośrednich $x \in X_k$ należących do ścieżki nie nastąpią ponowne odwiedziny tych wierzchołków.

3. Wyniki obliczeń

Program realizujący obie wersje algorytmu /H-K i T/ został napisany w języku Fortran i uruchomiony na m.c. ODRA 1325. Program testowano na serii kilkudziesięciu rozrzedzonych grafów generowanych losowo. Przy poziomie śledzenia Trace 1 czasy wykonywania obu wersji algorytmu były tak niewielkie /nie przekraczały 5 sek dla największych testowanych zadań łącznie z wydrukami pośrednimi/, że dla porównania obu wersji konieczne było przyjęcie poziomu śledzenia Trace 2 /czasy wzrastają ok. 30 razy/. Grafy losowe charakteryzowane były przez 2 parametry: łączną liczbę wierzchołków m+n oraz "zapełnienie" grafu określane jako średnia liczba krawędzi przypadająca na każdy wierzchołek grafu. Testowano grafy rozrzedzone o liczbie wierzchołków m+n = 200, 400, 800, 1200 oraz zapełnieniu od 1.25 do ok. 4. Badano najtrudniejszy do rozwiązywania przypadek, gdy m=n. Wyniki przedstawiono w tabelach.

Wymiar 200				
zapełnienie	Czas wykonania		Liczba iteracji	
	H-K	T	H-K	T
1.25	5	3	2	1
1.25	3	3	1	1
1.75	7	5	4	2
1.75	7	5	4	2
2.25	9	7	4	2
2.25	11	7	5	2
2.75	13	11	6	3
2.75	15	13	6	4
3.25	11	10	4	3
3.25	14	14	5	4
Wymiar 400				
1.25	8	6	2	1
1.25	8	7	2	1
1.75	13	9	4	2
1.75	12	9	4	2
2.25	26	33	8	4
2.25	28	17	8	3
2.75	23	19	5	3
2.75	25	21	6	3
3.25	29	26	6	4
3.25	26	19	6	3

Wymiar 800				
zapełnienie	Czas wykonania		Liczba iteracji	
	H-K	T	H-K	T
1.25	16	13	3	2
1.25	16	12	3	1
1.75	34	24	6	3
1.75	39	26	7	3
2.25	53	35	9	3
2.25	62	42	9	4
2.75	81	60	10	5
2.75	73	51	9	4
3.25	76	56	9	5
3.25	103	60	8	5
3.50	80	69	7	5
3.50	83	54	7	5
3.75	69	53	6	4
3.75	90	73	8	4
4.00	69	61	6	5
4.00	69	61	6	4
4.25	49	47	5	4
4.25	85	65	7	5
4.50	62	63	6	5
4.50	71	56	6	4
Wymiar 1200				
1.33	22	18	2	1
1.33	28	21	3	2
1.66	40	28	5	2
1.66	80	29	6	2
2.00	69	41	9	3
2.00	68	56	8	3
2.33	108	76	11	5
2.33	114	100	11	5
2.66	87	71	9	4
2.66	108	78	9	5
2.83	129	88	10	5
2.83	131	75	10	4
3.00	109	82	8	5
3.00	131	102	10	6
3.17	118	89	9	6
3.17	144	87	10	5
3.33	128	96	9	5
3.33	118	102	9	6

Z porównania obu wersji widać, że liczba iteracji w algorytmie T jest wyraźnie niższa niż w algorytmie H-K /choć zysk na łącznym czasie obliczeń jest łagodzony przez dodatkowy koszt związany z etykietowaniem dodatkowych warstw grafu/. Z przeprowadzonych obliczeń wynika, że czas wykonywania algorytmu T jest dla analizowanych przykładów w przybliżeniu prawie liniową funkcją liczby wierzchołków i krawędzi grafu.

LITERATURA

[1]. Ford L.R.Fulkerson D.R.: Przepływy w sieciach, WNT, Warszawa 1968

[2]. Darby-Dowman K., Mitra G.: An investigation of algorithms used in the restructuring of linear programming basis matrices prior to inversion, Techn. Rep STR/33, August 1979, Brunel University, Middlesex, England.

[3]. Duff I.S.: A survey of sparse matrix research, Proc.IEEE, vol. 65, no.4, pp.500-535, 1977

[4]. Hall M.: Distinct representatives of subsets, Biul.Amer.Math.Soc.54 1948 , pp.922-926.

[5]. Hopcroft J.E., Karp R.M.: An $n^{5/2}$ algorithm for maximum matching in bipartite graphs, SIAM J. Comp. vol. 2, no.4, pp.225-231, 1973

[6]. Kuhn H.W.: The hungarian method for the assignment problem, Noval Res. Logist. Quart 2 1955 , pp.83-97

[7]. Lipski W.: Kombinatoryka dla programistów, WNT, Warszawa 1982

[8]. Toczyłowski E.: Jednoetapowy algorytm blokowej trójkątyzacji macierzy bazowych dla wielkich zadań programowania liniowego, Arch.Autom. i Tel. 1983, zeszyt 3-4.

[9]. Toczyłowski E.: A hierarchical representation of the inverse for sparse matrices, SIAM J. Alg.Disc.Meth. no 5, vol. 1, 1984

Recenzent: Doc.dr hab.inż.Jerzy Klamka

Wpłynęło do Redakcji do 30.03.1984r.

НЕКОТОРАЯ РЕАЛИЗАЦИЯ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ ДЛЯ ЗАДАЧ МАКСИМАЛЬНОГО НАДЕЛА

Р е з ю м е

В работе представлено оригинальную реализацию параллельных алгоритмов для задач максимального надела. Реализации практически совмещают преимущества последовательного алгоритма Халла с теоретическими особенностями параллельных алгоритмов.

A REALIZATION OF PARALLEL ALGORITHMS FOR A MAXIMAL ASSIGNMENT PROBLEM

S u m m a r y

An original realization of parallel algorithms to solve a maximal assignment problem is present. It joins practical properties of sequential Hall's algorithm with theoretical ones of parallel algorithms.