ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1982

Seria: AUTOMATYKA z. 63 Nr kol. 735

Eugeniusz TOCZYŁOWSKI, Krzysztof RUDOWSKI

Instytut Automatyki, Politechnika Warszawska

ALGORYTM ROZWIĄZYWANIA ZADAŃ PRZYDZIAŁU Z MINIMAKSOWYM KRYTERIUM

Streszczenie. W pracy przedstawiono kilka wersji algorytmu rozwiązywania problemu przydziału. Różnią się one efektywnością w zależności od wymiaru i liczby danych pomocniczego zadania znajdowania pełnego skojarzenia w grafie dwudzielnym. Przedstawiono kilka metod umożliwiających zwiększenie efektywności obliczeń przez wykorzystanie rozrzedzenia grafów dwudzielnych. Zaproponowano efektywne wykorzystanie pamięci i efektywną organizację obliczeń w procedurach realizujących algorytmy.

1. ZADANIE PRZYDZIAŁU

Dany jest zbiór n zadań oraz zbiór m pracowników. Bez straty ogólności możemy założyć, że $m \leq n$. Przydziałem zadań do pracowników nazwiemy dowolne przyporządkowanie zadań do pracowników. Jeżeli każdemu pracownikowi przyporządkowane jest dokładnie jedno, odrębne zadanie, to przydział nazywany jest dopuszczalnym. Niech a_{ij} będzie czasem wykonania i-tego zadania przez j-tego pracownika (jeżeli i-te zadanie nie może być wykonane przez j-tego pracownika, to przyjmujemy $a_{ij} = \infty$). Wprowadźmy zmienne decyzyjne zero-jedynkowe x_{ij}, przy czym $x_{ij} = 1$ wtedy i tylko wtedy, gdy j-te zadanie przydzielono i-temu pracownikowi. Jeżeli założymy, że wszystkie zadania są wykonywane jednocześnie, to przydział minimalizujący czas wykonania zadań można wyznaczyć przez rozwiązanie minimaksowego zadania

$$\text{zminimalizować} \quad z = \max_{i,j} a_{ij} x_{ij}$$

$$\sum_{i=1}^{m} x_{ij} \leq 1 \qquad j = 1,\ldots,n \tag{1}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,\ldots,m$$

$$x_{ij} = 0,1$$

Funkcja celu zadania (1) jest typu wąskie gardło. Z algorytmicznego punktu widzenia będą interesujące inne, równoważne reprezentacje zadania (1) w ujęciu macierzowym i w ujęciu teorii grafów. W reprezentacji macierzowej rozpatrywana jest macierz $A = [a_{ij}]_{m \times n}$, w której poszukiwany jest przydział kolumn do wierszy (tzn. każdemu wierszowi przydzielana jest oddzielna kolumna) i taki, że największy z wyróżnionych w ten sposób elementów a_{ij} jest jak najmniejszy. W notacji teorii grafów konstruowany jest graf dwudzielny $G(M,N,E)$, gdzie $M = \{r_1,\ldots,r_m\}$ jest zbiorem wierzchołków odpowiadających wierszom macierzy A (pracownikom), $N = \{c_1,\ldots,c_n\}$ jest zbiorem wierzchołków odpowiadających kolumnom macierzy A (zadaniom), natomiast E jest zbiorem (nieskierowanych) krawędzi łączących wierzchołki z M i N, tzn. $\{r_i,c_j\} \in E$ wtedy i tylko wtedy, gdy $a_{ij} < \infty$. Wartość a_{ij} przyporządkowana będzie krawędzi $\{r_i,c_j\}$. Przydział na grafie dwudzielnym określany jest pojęciem skojarzenia (skojarzenie wierzchołków należących do zbioru M z wierzchołkami należącymi do N).

Zadanie przydziału ma wiele interpretacji. Przykładem może być zadanie przydziału monterów do stanowisk montażowych na linii montażowej. Innym przykładem jest zadanie wzajemnie jednoznacznego przyporządkowania wierszy i kolumn zadanej macierzy. Zadanie to jest często rozwiązywane podczas analizy strukturalnej wielkich problemów obliczeniowych, gdzie n oraz m może być nawet rzędu kilku tysięcy.

Szczególnie efektywną klasą algorytmów rozwiązujących zadanie przydziału jest klasa algorytmów progowych [1] [4]. W algorytmach tych zasadniczą rolę odgrywa pomocniczy problem decyzyjny parametryzowany przez zmienną progową V. Problem ten polega na sprawdzeniu, czy dla ustalonej wartości zmiennej progowej V, istnieje dopuszczalny przydział o koszcie V. Problem progowy jest równoważny problemowi istnienia pełnego skojarzenia w grafie dwudzielnym.

Złożoność obliczeniowa najlepszych algorytmów minimaksowego zadania przydziału jest równa $O(n^3 \log_2 n)$ [4] dla $n = m$. Intencją tej pracy jest opracowanie algorytmu o jak najmniejszej złożoności obliczeniowej.

Zaproponowano kilka metod zwiększających efektywność algorytmu, wykorzystujących szczególne właściwości rozwiązywanych zadań.

2. PROBLEM MAKSYMALNEGO SKOJARZENIA

Dla ustalonej wartości zmiennej progowej V rozpatrywany jest progowy graf dwudzielny $G(M,N,E_V)$, gdzie $E_V = \{\{r_i,c_j\} : a_{ij} \leq V\}$. Niech $S \subset E_V$ będzie pewnym podzbiorem jego krawędzi. Zbiór S nazywany jest skojarzeniem, jeżeli każdy wierzchołek grafu jest incydentny z co najwyżej jedną krawędzią z S. Jeżeli $|S| = m$, to skojarzenie nazywane jest pełnym. Dla danego grafu $G(M,N,E_V)$ oraz danego skojarzenia S mówimy, że wierz-

chołek $v \in M \cup N$ jest swobodny, jeżeli nie jest incydentny z krawędziami skojarzenia. Droga prosta P w grafie $G(M,N,E_V)$

$$P = (v_1,v_2)(v_2,v_3)\ldots(v_{2k-1},v_{2k}) \tag{2}$$

nazywana jest drogą powiększalną, jeżeli krawędzie drogi należą przemiennie do $E \setminus S, S, \ldots, E \setminus S$, a ponadto wierzchołki krańcowe v_1 i v_{2k} są swobodne. Podstawowym faktem, wykorzystywanym w algorytmach znajdowania maksymalnego skojarzenia jest twierdzenie, że dane skojarzenie S nie jest skojarzeniem o największej liczności wtedy i tylko wtedy, gdy w grafie $G(M,N,E_V)$ istnieje droga powiększalna [1]. Jeżeli nie jest skojarzeniem o największej liczności, czyli gdy istnieje pewna droga powiększalna (2), to skojarzenie S można zastąpić skojarzeniem o liczności większej o jeden, stosując podstawienie $S' = S \oplus P$, gdzie symbol $\oplus$ oznacza różnicę symetryczną dwóch zbiorów. Podstawienie to odpowiada zastępieniu krawędzi skojarzenia odpowiadających parzystym łukom drogi (2) przez krawędzie nieparzyste, których liczba jest dokładnie większa o jeden. Wszystkie algorytmy znajdowania maksymalnego skojarzenia oparte są na powyższym podstawieniu. Rozróżnia się dwie zasadnicze grupy metod:

(i) algorytmy szeregowe (Hall [2]), w których jedna iteracja polega na znalezieniu pojedynczej drogi powiększalnej P i dokonaniu podstawienia $S' = S \oplus P$,

(ii) algorytmy równoległe (Hopcroft-Karp [3]), w których jedna iteracja polega na wyznaczeniu możliwie maksymalnego zbioru rozłącznych dróg powiększalnych $P_1,\ldots,P_k$ a następnie dokonaniu podstawienia $S' = S \oplus P_1 \oplus \ldots \oplus P_k$.

Jeśli oznaczymy $NM = |E_V|$, to asymptotyczna złożoność algorytmów pierwszej i drugiej grupy jest równa odpowiednio $O(m\,NM)$ oraz $O(\sqrt{m} \cdot NM)$. Praktyczna przewaga algorytmów równoległych nad szeregowymi uwidacznia się dopiero dla większych zadań, co wynika z większego współczynnika proporcjonalności w oszacowaniu złożoności obliczeniowej. Jak wykazują doświadczenia numeryczne, algorytmy szeregowe są bardziej efektywne od algorytmów równoległych dla zadań, w których m jest rzędu kilkadziesiąt oraz dla niektórych zadań nawet przy m rzędu kilkaset. Zbiory zadań, dla których algorytmy szeregowe i równoległe są szczególnie efektywne wydają się uzupełniać: przewaga algorytmów równoległych nad szeregowymi uwidacznia się dla grafów z większą liczbą wierzchołków i stosunkowo rozrzedzonych (w których $NM \leq k\,m$, gdzie k jest niewielką stałą), podczas gdy dla grafów z mniejszą liczbą wierzchołków lub dla grafów prawie pełnych (gdy $NM \approx m^2$) bardziej efektywna są algorytmy szeregowe.

Z powyższego powodu podczas rozwiązywania zadania przydziału opłacalne jest wykorzystywanie kilku różnych algorytmów poszukiwania maksymalnego skojarzenia, gdyż zachodzi wtedy potrzeba badania istnienia pełnego skojarzenia dla różnych grafów progowych o różnym stopniu rozrzedzenia.

Dla danego grafu progowego o liczbie krawędzi NM przez A_{NM} oznaczać będziemy macierz progową $[\tilde{a}_{ij}]$, gdzie $\tilde{a}_{ij} = a_{ij}$ jeżeli $\{i,j\} \in E_V$ oraz $\tilde{a}_{ij} = \infty$ w przeciwnym przypadku.

3. WYKORZYSTANIE SZCZEGÓLNYCH WŁAŚCIWOŚCI ZADANIA PROGOWEGO

Jedna duża iteracja algorytmu minimaksowego zadania przydziału polega na rozwiązaniu zadania progowego, czyli zadania maksymalnego skojarzenia, co w najgorszym przypadku może wymagać O(m·NM) obliczeń dla algorytmu Hella. W rozdziale tym zajmiemy się metodami umożliwiającymi uproszczenie rozwiązania zadania progowego. Zauważmy, że w istocie zadanie to polega na odpowiedzi na pytanie, czy dla ustalonej wartości progu w grafie progowym istnieje pełne skojarzenie. Problem ten nie jest problemem optymalizacyjnym, lecz jedynie problemem decyzyjnym i może być rozwiązany łatwiej aniżeli zadanie znalezienia maksymalnego skojarzenia. Podamy teraz metodę upraszczającą rozwiązanie tego problemu decyzyjnego, w przypadku gdy m = n.

Wykorzystanie silnie spójnych składowych

Dla dalszych celów zmodyfikujemy nieco pojęcie skojarzenia. Niech $S \subset M \times N$ będzie pewnym zbiorem krawędzi, niekoniecznie należących do E. Zbiór S nazywamy skojarzeniem, jeżeli każdy wierzchołek $v \in M \cup N$ jest incydentny z dokładnie jedną krawędzią z S. Dla danego grafu dwudzielnego $G(M,N,E_V)$ liczbę $|S \cap E_V|$, czyli liczbę wspólnych krawędzi z E_V nazywać będziemy licznością skojarzenia S. Skojarzenie o liczności m nazywane jest skojarzeniem pełnym. Powyższe definicja skojarzenia pełnego pokrywa się z definicją z rozdz. 2. Droga prosta P w grafie $G(M,N,E_V \cup S)$

$$P = (v_1,v_2),(v_2,v_3),\ldots,(v_{2k-2},v_{2k-1})$$

z parzystą liczbą łuków nazywana jest drogą przesuwalną, jeżeli nieparzyste łuki należą do S, parzyste do E_V-S, a ponadto pierwszy łuk nie należy do E_V (tzn. wierzchołki v_1 i v_2 są swobodne). Jeżeli $v_1 = v_{2k-1}$, to P nazywany jest cyklem przesuwalnym. Jeżeli $v_1 \neq v_{2k-1}$, to z P tworzymy cykl cl(P) jak następuje. Niech $v_{2k} \in M \cup N$ będzie wierzchołkiem określonym jednoznacznie przez warunek $(v_{2k-1},v_{2k}) \in S$. Cykl cl(P) tworzymy dodając do P łuk $(v_{2k-1},v_{2k}) \in S$ oraz łuk domykający (v_{2k},v_1), niekoniecznie należący do E_V, tzn.

$$cl(P) = (v_1,v_2),\ldots,(v_{2k-2},v_{2k-1})(v_{2k-1},v_{2k})(v_{2k},v_1).$$

W cyklu $cl(P)$ znajdują się krawędzie nie należące do E_V. Niech d_1 będzie liczbą tych nieparzystych krawędzi, które nie należą do E_V oraz d_2 liczbą krawędzi parzystych nie należących do E_V. Różnicę d_1-d_2 nazywamy defektem drogi P i oznaczać będziemy przez $df(P)$. Jeżeli $df(P)\geq 1$, to drogę przesuwalną nazywać będziemy drogą powiększalną. Dla tak uogólnionego pojęcia drogi powiększalnej można udowodnić uogólnienia podstawowych twierdzeń z teorii skojarzeń na grafach dwudzielnych.

Twierdzenie 1 [6]. Jeżeli S jest skojarzeniem i P jest drogą przesuwalną, to $S' = S \oplus cl(P)$ jest skojarzeniem i jego liczność jest powiększona o $df(P)$.

Twierdzenie 2 [6]. Jeżeli S i R są skojarzeniami o liczności odpowiednio s i r, $s>r$, to $S \oplus R$ zawiera zbiór wierzchołkowo rozłącznych dróg przesuwalnych $P_1,\ldots,P_k$ względem skojarzenia S, takich że $df(P_1)+\ldots df(P_k) = s-r$.

Dla zdefiniowanego w powyższy sposób skojarzenia S utwórzmy z grafu dwudzielnego $G(M,N,E_V \cup S)$ graf skierowany $\vec{G}_S = \vec{G}(M,N,\vec{E_V \cup S})$ przez skierowanie krawędzi z S od wierzchołków z M (wierszy) do wierzchołków z N (kolumn), natomiast krawędzi z E_V S od wierzchołków z N do wierzchołków z M. Przyjmijmy, że skojarzenie S nie jest skojarzeniem pełnym. Załóżmy, że w grafie $\vec{G}_S$ zostały znalezione silnie spójne składowe. Najbardziej efektywnym algorytmem wyodrębniania silnie spójnych składowych grafu jest algorytm Tarjana [5] o złożoności obliczeniowej $O(NM)$, gdzie NM jest liczbą łuków grafu $\vec{G}_S$. Łuki w grafie $\vec{G}_S$ łączące wierzchołki z różnych składowych silnej spójności nazywamy łukami skrośnymi.

Twierdzenie 3 [6]. Jeżeli S i R są skojarzeniami, R jest skojarzeniem pełnym, to każda droga powiększalna należąca do $S \oplus R$ jest zawarta w jednej z silnie spójnych składowych grafu $\vec{G}_S$.

Dzięki powyższej właściwości usunięcie łuków skrośnych z grafu $\vec{G}_S$ i oddzielne wyznaczanie dróg powiększalnych dla każdej silnie spójnej składowej umożliwia zmniejszenie maksymalnej liczby kroków algorytmów Halla i Hopcrofta-Karpa odpowiednio z m i $2\sqrt{m}$ do p i $2\sqrt{p}$, gdzie $p \leq m$ jest liczbą wierzchołków największej silnie spójnej składowej grafu.

Wykorzystanie algorytmu Tarjana w trakcie wykonywania kolejnych iteracji jednego z algorytmów znajdowania maksymalnego skojarzenia jest korzystne dla grafów rozrzedzonych o dużej liczbie wierzchołków.

Z punktu widzenia rozwiązywania zadania progowego szczególnie interesujący jest fakt, iż wyodrębnianie silnie spójnych składowych grafu $\vec{G}_S$ umożliwia uzyskanie w pewnych sytuacjach odpowiedzi negatywnej "w grafie progowym pełne skojarzenie nie istnieje" bez konieczności znajdowania maksymalnego skojarzenia. Załóżmy, że dla danego skojarzenia S, nie będącego skojarzeniem pełnym w grafie $\vec{G}_S$ istnieje wierzchołek $r \in M$ stanowiący silnie spójną składową grafu. Jeżeli krawędź skojarzenia $\{r,c\} \in S$ incydentna z r nie należy do $\vec{E_V}$, to łatwo wykazać, że skojarzenie peł-

ne w grafie progowym nie istnieje. Wynika to z faktu, że w macierzy A_{NM} można wyodrębnić $k+1$ wierszy zawierających jedynie k niezerowych kolumn, $k \geq 0$.

Wyodrębnianie silnie spójnych składowych pozwala również na wyłączenie z rozważań "nieobiecujących" elementów a_{ij} o których wiadomo, że nie wejdą one w skład pełnych skojarzeń niezależnie od zmian progu V. Niech V będzie progiem górnym, tzn. takim, że w grafie $G(M,N,E_V)$ istnieje pełne skojarzenie S. Przypuśćmy teraz, że dla ustalonego progu górnego V wyodrębniono w grafie G_S jego silnie spójne składowe. Obniżenie progu V prowadzi co najwyżej do rozbicia wyodrębnionych składowych grafu $\vec{G_S}$. Z powyższego faktu oraz z twierdzenia 3 wynika, że łuki skrośne grafu $\vec{G_S}$ nie będą wchodzić w skład pełnych skojarzeń dla dowolnych wartości progowych V mniejszych od progu górnego a zatem mogą one być całkowicie usunięte.

Jest oczywiste, że za powyższe, dodatkowe informacje płacimy kosztem wyodrębniania silnie spójnych składowych (za pomocą algorytmu Tarjana). Koszt ten jest liniową funkcją liczby krawędzi grafu. Można oczekiwać, że usuwanie łuków skrośnych będzie opłacalne dla większych zadań, szczególnie dla mniejszych wartości progu V, kiedy to grafy progowe są stosunkowo rozrzedzone.

4. OPIS REALIZACJI ALGORYTMU

Przyjmiemy wstępnie, że znane są: dolna wartość progowa V_d, dla której w grafie progowym nie istnieje pełne skojarzenie oraz górna wartość progowa V_g, dla której w grafie progowym pełne skojarzenie istnieje. Podstawowy schemat algorytmów progowych można zapisać następująco.

Algorytm

1. (Dobór progu). Dobierz nową wartość progową $V \in (V_d,V_g)$ idź do 2.
2. (Problem pełnego skojarzenia). Zbadaj, czy istnieje w grafie progowym $G(M,N,E_V)$ pełne skojarzenie. Jeżeli istnieje, to idź do 3, jeżeli nie istnieje, to idź do 4.
3. $V_g := V$, idź do 5.
4. $V_d := V$, idź do 5.
5. Jeżeli $V_d < V_g$ idź do 1. W przeciwnym przypadku wyznacz maksymalne skojarzenie w grafie $G(M,N,E_V)$. Stop.

Schemat algorytmu łatwo można zmodyfikować również na przypadek, w którym wartość progu górnego V_g nie jest znana i kiedy wykorzystywany jest wyłącznie próg dolny V_d jak w algorytmie Garfinkela [1]. Omówimy teraz warianty algorytmu doboru progu i algorytmu pełnego skojarzenia.

4.1. Dobór progu

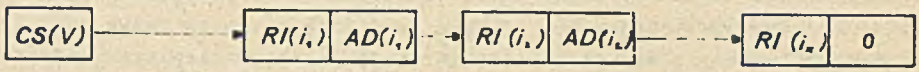
Wyróżnimy dwa zasadnicze sposoby doboru progu. Pierwszy pochodzi od Garfinkela [1] i polega na stopniowym podwyższaniu progu w taki sposób, że w kolejnych krokach uzyskiwane są progi dolne (dla których nie istnieje przydział pełny), natomiast pierwszy przydział pełny jest rozwiązaniem optymalnym. Wartość nowego progu wynika z analizy rozwiązania zadania progowego dla starej wartości progu. Zaletą tej reguły jest to, że grafy progowe rozpatrywane w kolejnych iteracjach algorytmu są stosunkowo rozrzedzone, co generalnie zwiększa efektywność rozwiązywania zadań progowych. Do wad metody należy zaliczyć stosunkowo dużą liczbę kroków metody, niezbędnych do uzyskania dopuszczalnego rozwiązania.

Drugi sposób doboru procu, zaproponowany przez Słomińskiego [4], wykorzystuje podział dychotomiczny i wymaga jedynie $O(\log_2 n)$ kroków. Przyjmijmy, że elementy macierzy $A = [a_{ij}]$ są wstępnie uszeregowane w kolejności niemalejącej. Najlepsze algorytmy sortowania umożliwiają uporządkowanie zbioru o liczności τ za pomocą $O(\tau \log_2 \tau)$ operacji (porównań, przestawień itp.) [7]. Dla uporządkowanego w kolejności niemalejącej ciągu elementów macierzy $[a_{ij}]$ można stosować dwa rodzaje progów. Progiem liczbowym nazwiemy dowolną całkowitą liczbę NM, $1 \leq NM \leq \tau$ określającą zbiór NM pierwszych elementów tego ciągu. Próg V określający zbiór elementów $C = \{a_{ij} : a_{ij} \leq V\}$ nazywamy progiem wartościowym. Dychotomiczny podział można stosować zarówno dla progu liczbowego, jak i dla progu wartościowego. W pierwszym przypadku wartość progu NM jest wyznaczana według reguły $NM = (NM_d + NM_g)/2$ (z zaokrągleniem do liczby całkowitej), gdzie NM_d i NM_g są odpowiednio progami dolnym i górnym. W przypadku ustalania progu wartościowego dobierana jest wartość progu V w taki sposób, że rozpatrywany podciąg elementów o wartościach $(a_{ij} : V_d < a_{ij} \leq V_g)$ jest dzielony na dwa podciągi $(a_{ij} : V_d < a_{ij} \leq V)$ oraz $(a_{ij} : V < a_{ij} \leq V_g)$ zrównoważone "wartościowo", tzn. zawierające tę samą liczbę różnych wartości.

Reguła pierwsza wymaga $O(\log_2 n)$ kroków, podczas gdy reguła druga wymaga jedynie $O(\log_2 z)$ kroków, gdzie z jest maksymalną liczbą elementów macierzy $[a_{ij}]$ o różnych wartościach. Reguła druga może więc wymagać znacznie mniej kroków w przypadku macierzy posiadających dużo elementów o tych samych wartościach. Jej wadą jest jednak konieczność pamiętania dodatkowej tablicy o wymiarze z. W opisywanym algorytmie implementowane są obie reguły, przy czym wybór reguły będzie uzależniony od parametrów zadania. Ponadto zastosowano podział "niesymetryczny" umożliwiający dobór progu stosunkowo blisko progu dolnego, co jak wiadomo zwiększa efektywność rozwiązywania zadania progowego.

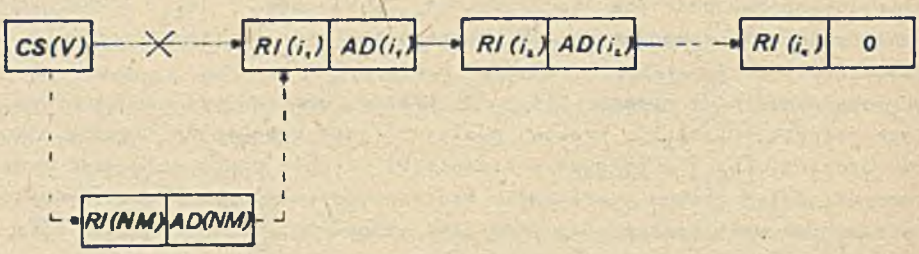
4.2. Realizacja algorytmu pełnego skojarzenia

Dla zadanego progu V poszukiwanie maksymalnego skojarzenia w progowym grafie dwudzielnym $G(M,N,E_V)$ jest równoważne poszukiwaniu przyporządkowania wierszy i kolumn w macierzy A_{NM} takiego, że liczba ograniczonych progiem współczynników a_{ij} na skrzyżowaniu tych wierszy i kolumn jest jak największa. Najbardziej efektywna realizacja algorytmów pełnego skojarzenia opiera się na powyższej równoważności. Niech P będzie macierzą permutacyjną stopnia n. W zapisie macierzowym zadaniu pełnego skojarzenia odpowiada zadanie znalezienia takiej macierzy permutacyjnej P, że macierz $A_{NM}P$ posiada skończoną przekątną. Ogólnie, dowolnemu skojarzeniu S w grafie $G(M,N,E_V)$ odpowiada wzajemnie jednoznacznie pewna macierz permutacyjna P.



Rys. 1. Listowy kolumnowy zapis macierzy A. V - numer kolumny. GS(V) - adres pierwszego elementu listy z kolumną V. Zapis i-tego elementu listy: RI(i) - numer wiersza tego elementu; AD(i) - adres następnego elementu w liście w kolumnie V (jeżeli element i-ty jest ostatnim w kolumnie V, to AD(i) = 0)

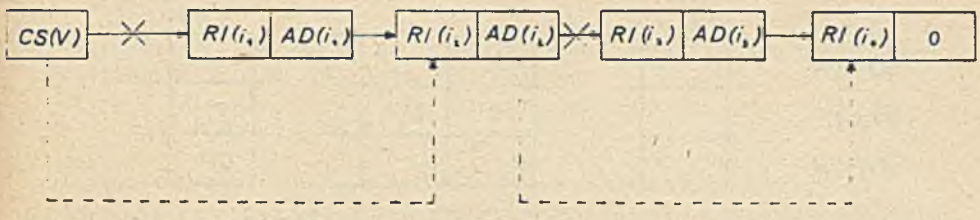
Macierz A_{NM} zapisywana jest w postaci listowej (rys. 1). Dla ustalonej wartości progowej macierz A_{NM} zawiera NM elementów mniejszych od wartości progowej i tylko te elementy są elementami listy. Przyjęto kolumnową postać listową, która pozwala na szybki dostęp do kolejnych elementów kolumn macierzy A_{NM}. Pozwala również na proste modyfikacje listy polegające na dopisaniu (rys. 2) lub usunięciu (rys. 3) pewnej liczby ele-



Rys. 2. Dopisanie nowego elementu a_{wv} przy zmianie progu z NM na wyższy:

NM:=NM+1, AD(NM):=CS(V), CS(V):=NM, RI(NM):=W

mentów listy. Modyfikacje listy są niezbędne przy zmianie progu. Wykorzystanie algorytmu Tarjana do eliminacji łuków skrośnych w grafie (reprezentowanych przez elementy macierzy A_{NM}) również wymaga usuwania elementów listy.



Rys. 3. Usuwanie elementu i_1 oraz i_3

$CS(V):=i_2$, $AD(i_2):=i_4$

Elementy macierzy A_{NM} uszeregowane są według następującego porządku: jeżeli $a_{ij} > a_{kl}$, to element listy odpowiadający elementowi a_{kl} jest umieszczony na pozycji wcześniejszej niż element a_{ij}. Pozwala to w prosty sposób modyfikować listę przy zmianie wartości progu NM.

Algorytmy poszukiwania pełnego skojarzenia w grafie progowym polegają na znalezieniu macierzy permutacyjnej P takiej, że macierz $A_{NM}P$ ma na przekątnej głównej współczynniki o ograniczonych wartościach.

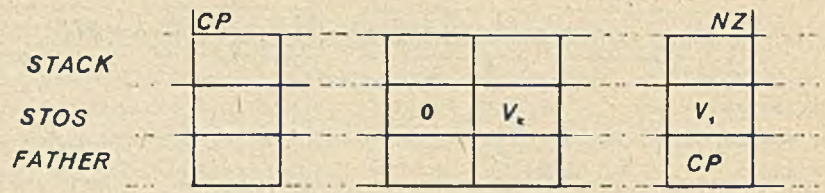
Jeżeli n = m, to dodatkowo wyznaczana jest macierz permutacyjna S, umożliwiająca transformację

$$A_{NM}P \rightarrow S.(A_{NM}P).S^T.$$

gdzie macierz permutacyjna S służy do jednoczesnej permutacji wierszy i kolumn macierzy $A_{NM}P$ do takiej postaci, że prowadzi to do wyodrębnienia silnie spójnych składowych skierowanego grafu progowego (odpowiadających kwadratowym podmacierzom wzdłuż przekątnej głównej). Oznacza to, że jeżeli pominiemy nieskończone współczynniki w macierzy A_{NM}, to wynikowa permutowana macierz posiada strukturę blokowo-trójkątną.

Wszystkie macierze permutacyjne pamiętane są w postaci wektorów permutacyjnych. Przykładowo macierzy P_{nxn} odpowiada n wymiarowy wektor p = (p(1),....,p(n)), taki, że P(i,j) = 1 wtedy i tylko wtedy, gdy j = p(i). W realizowanych algorytmach wektory permutacyjne p oraz s pamiętane są w postaci tablic o nazwie R1 i STACK.

Do pamiętania struktury blokowo-trójkątnej macierzy A służy obok macierzy permutacyjnej S wektor FATHER o wymiarze n (rys. 4). Każdemu wyodrębnionemu blokowi macierzy A odpowiada obszar tablicy FATHER od początkowej pozycji CP do końcowej pozycji NZ, na których znajdują się w tablicy STACK numery kolumn wyodrębnionego bloku. W tablicy FATHER na końcowej pozycji NZ wyodrębnionego bloku wpisany jest adres początku bloku CP. W związku z przyjętym zapisem bloki są analizowane sekwencyjnie od ostatniego do pierwszego. Do zapisu swobodnych kolumn zastosowano dodatkowy wektor STOS o wymiarze n. Kolumny te zapisuje się dla każdego bloku oddzielnie (rys. 4). Znacznikiem końca zbioru swobodnych kolumn jest zero



Rys. 4. Opis bloku odpowiadającego silnie spójnej składowej grafu progowego ($V_1,\ldots,V_k$ - numery swobodnych kolumn bloku)

(jeżeli w bloku wszystkie kolumny są swobodne, to wypełniają cały obszar bloku). Kolumny swobodne rozpatrywane są od kolumny zapisanej w STOS(NZ).

Łączna zajętość pamięci tablic używanych we wszystkich implementowanych algorytmach poszukiwania pełnej przekątnej jest równa 11n + 2NM.

W celu rozwiązania zadania pełnego przydziału wybierany jest jeden z następujących algorytmów

(i) Halla
(ii) Hopcrofta-Karpa
(iii) Halla plus Tarjana
(iv) Hopcrofta-Karpa plus Tarjana

w zależności od wartości parametrów sterujących uzależnionych od rozmiarów i rozrzedzenia macierzy struktury grafu. W przypadkach (iii) oraz (iv) brak pełnego przydziału może być stwierdzony w algorytmie Tarjana. W chwili obecnej autorzy implementują algorytmy (i) oraz (iii). Podstawowym parametrem sterującym jest parametr o nazwie NU określający warunek wywołania algorytmu Tarjana w algorytmach maksymalnego skojarzenia. Reguły dotyczące wartości tego parametru są ustalane na podstawie eksperymentów numerycznych.

Dla zwiększenia efektywności rozwiązywania zadania progowego przy zmianie progu na wyższy zachowywane są dotychczas znalezione przydziały z poprzedniej iteracji. Podobnie przy zmianie progu na niższy znikają jedynie przydziały odpowiadające elementom położonym na przekątnej macierzy A_{NM}^P większym od nowej wartości progu.

PODSUMOWANIE

Omówione w pracy rozwiązania algorytmiczne i programowe stanowią podstawę organizacji rozbudowanego pakietu zunifikowanych procedur służących do efektywnego rozwiązywania zadań przydziału. Realizacja całego pakietu jest zamierzeniem długofalowym i będzie realizowana etapami. W pierwszym etapie zaimplementowano algorytmy Halla i Tarjana w wersji umożliwiającej bezpośrednią i efektywną współpracę tych algorytmów.

LITERATURA

[1] Garfinkel R.S.: An Improved Algorithm for the Bottleneck Assignment Problem, Oons. Res. 19 (1971), 1747-1751.

[2] Hall M.: An Algorithm for Distinct Reprezentations, Amer. Math. Monthly, Vol 63 (1956), 716-717.

[3] Hopcroft J.E., Karp R.M.: An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs, SIAM J. Comput, Vol 2 (1973), No 4. 225-231.

[4] Słominski L.: Bottleneck Assignment Problem: an Efficient Algorithm, Arch. Aut. tom 24, zeszyt 4, 469-482, 1979.

[5] Tarjan R.E.: Depth-First Search and Linear Graph Algorithms SIAM J. Comput. Vol 1, pp. 146-160, 1972.

[6] Toczyłowski E.: A Maximum Matching algorithm for block-triangularization of Large-Scale Matrices. przedłożono do App. Discr. Math.

[7] Trybuś E.: Metody porządkowania, Mat. Stos. IV, 1975, 95-117.

Recenzent: Prof. dr inż. Henryk KOWALCZYKI

Wpłynęło do Redakcji 15.05.1982 r.

АЛГОРИТМ РЕШЕНИЯ МИНИМАКСНОЙ ЗАДАЧИ О НАЗНАЧЕНИИ

Р е з ю м е

В работе представлено несколько новых алгоритмов для их задачу о назначении. Даны методы улучшения эффективности работы ... двудольных графов. Предложено эффективное использование памяти машины также эффективную организацию работы ... их алгоритма

AN ALGORITHM FOR THE BOTTLENECK ASSIGNMENT PROBLEM

S u m m a r y

A new threshold algorithm for the bottleneck Assignment problem is presented. We apply a method which anables to improve the efficiency of the algorithms for finding full matchings in the bipartite graphs. We also show how to reduce the number of steps of the threshold algorithm by making efficient use of the information from the previous steps of the algorithm.