ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1981

Seria: GÓRNICTWO z. 112 Nr kol. 697

Leon STARZYCZNY, Kazimierz DUSZA
KWK Dębieńsko

Franciszek MARECKI
Instytut Automatyki Politechnika Śląska

HARMONOGRAMOWANIE NIEZALEŻNYCH ZADAŃ BRYGAD UTRZYMANIA RUCHU NA KWK

Streszczenie. W referacie sformułowano model matematyczny pracy brygad utrzymania ruchu na KWK. Rozwiązano problem harmonogramowania niezależnych zadań. Do rozwiązania zaproponowano algorytm harmonogramowania dynamicznego. Podano także przykład obliczeniowy.

1. Wprowadzenie

Na kopalni węgla kamiennego można wyróżnić brygady wydobywcze oraz brygady utrzymania ruchu. Brygady utrzymania ruchu pracują w tzw. czasie dyspozycyjnym, gdy nie odbywa się wydobycie węgla. W tym czasie winny być wykonane wszystkie zadania konserwacyjno-remontowe. Opóźnienie realizacji tych zadań powoduje przestój brygad wydobywczych, a zatem stratę wydobycia węgla.

W referacie przedstawiony jest pewien model pracy brygad utrzymania ruchu. Zakłada się, że zadania są zlokalizowane na różnych oddziałach wydobywczych. Zadania te mogą być wykonane przez operatorów o odpowiednich kwalifikacjach (lub uprawnieniach). Operatorzy przed rozpoczęciem zmiany roboczej znajdują się w ustalonym punkcie KWK. W trakcie pracy każdy operator może wykonać kilka zadań na różnych oddziałach. A zatem trzeba wyróżnić czasy transportu pomiędzy oddziałami oraz czasy wykonywania zadań na oddziałach. W modelu założono, że zadania są niezależne, tzn. można je wykonywać w dowolnej kolejności. Ponadto założono, że każde zadanie może być wykonane przez jednego operatora.

Celem rozwiązania problemu należy określić harmonogram pracy dla każdego operatora. Harmonogram taki winien podawać kolejność realizacji i przedziały czasu, w których należy wykonać każde zadanie. Kryterium optymalizacji harmonogramowania polega na minimalizacji czasu potrzebnego do wykonania wszystkich zadań.

Przykłady analogicznych problemów były przedmiotem prac [4], [7], [9] i [8]. Istota matematyczna problemu sprowadza się do tzw. problemu M - ko-

miwojażerów. Do rozwiązania tego problemu były stosowana metoda programowania całkowitoliczbowego [10], [3]. W niniejszym referacie zostanie przedstawiony algorytm oparty na programowaniu dynamicznym [1], [2], [5], [6]. Algorytm ten zostanie zilustrowany na przykładzie obliczeniowym.

2. Sformułowanie problemu

Załóżmy, że dany jest zbiór elementów systemu:

$$S = \left\{ s_k \right\} \quad (k = \overline{1,K}) \tag{1}$$

gdzie:

s_k - k-ty element systemu,
K - liczba elementów w systemie.

Rozproszenie elementów systemu scharakteryzujemy czasami transportu pomiędzy tymi elementami. Niechaj będzie dana macierz czasów transportu pomiędzy elementami systemu:

$$T = \left[t_{\varkappa,k} \right] \quad (\varkappa = \overline{1,K}) \tag{2}$$

gdzie:

$t_{\varkappa,k}$ - czas transportu od elementu $s_\varkappa$ do elementu s_k.

Załóżmy, że dany jest zbiór zadań, które należy wykonać w systemie:

$$\Omega = \left\{ \omega_n \right\} \quad (n = \overline{1,N}) \tag{3}$$

gdzie:

ω_n - n-te zadanie,
N - liczba zadań.

Niechaj będzie dany wektor czasów wykonywania zadań:

$$\tau = \left[\tau_n \right] \tag{4}$$

gdzie:

τ_n - czas wykonywania zadania ω_n.

Załóżmy, że lokalizację zadań opisuje macierz:

$$A = \left[a_{k,n} \right] \tag{5}$$

Elementy tej macierzy definiujemy następująco:

$$a_{k,n} = \begin{cases} 1 : \text{jeśli zadanie } \omega_n \text{ jest zlokalizowane w } s_k \\ 0 : \text{w przeciwnym przypadku.} \end{cases}$$

Załóżmy, że dany jest zbiór operatorów:

$$W = \{w_m\} \quad (m = \overline{1,M})$$

gdzie:

w_m - m-ty operator,

M - liczba operatorów.

Rozważmy przypadek, gdy operatorzy posiadają kwalifikacje do wykonywania określonych zadań, co opisuje macierz:

$$B = [b_{m,n}] \tag{7}$$

Elementy tej macierzy definiujemy następująco:

$$b_{m,n} = \begin{cases} 1 : \text{jeśli operator } w_m \text{ posiada kwalifikacje do wykonywania zadania } \omega_n \\ 0 : \text{w przeciwnym przypadku.} \end{cases}$$

Zatem każdy operator może wykonywać tylko określone zadania.

Załóżmy, że celem optymalnego harmonogramowania realizacji zadań jest minimalizacja czasu potrzebnego do wykonania wszystkich zadań. Zadania są wykonywane przez operatorów, którzy wychodzą w chwili $t = 0$ z pewnego punktu (załóżmy s_1) i po zrealizowaniu wszystkich zadań do tego samego punktu mają powrócić. Zatem funkcja celu ma postać:

$$Q = \max_{1 \leq m \leq M} q_m \longrightarrow \min \tag{8}$$

gdzie:

q_m - moment powrotu operatora w_m do punktu s_1.

Tak sformułowany problem stanowi pewne rozwinięcie znanego w literaturze problemu M - komiwojażerów.

3. Algorytm

Do rozwiązania sformułowanego problemu zastosujemy oryginalny algorytm oparty na programowaniu dynamicznym. Zauważmy, że przydzielanie zadań operatorom można traktować jako N-etapowy proces decyzyjny. W procesie tym wyróżniamy stany, które zmieniają się po podjęciu decyzji o przydzieleniu zadania. Stan początkowy odpowiada sytuacji, gdy nie przydzielono żadnego zadania. Stany końcowe (których może być więcej niż jeden) odpowiadają różnym wariantom przydziału wszystkich zadań operatorom. Stany można pogrupować w warstwy. W każdej η-tej warstwie ($0 \leq \eta \leq N$) znajdują się stany zawierające η zadań. Z każdym stanem można związać jego wartość, definiowaną jako czas zakończenia realizacji zadań należących do tego stanu. Stan w ostatniej warstwie, który ma najmniejszą wartość, daje optymalne rozwiązanie problemu.

3.1. Generowanie stanów

Wprowadźmy następującą definicję stanu:

Definicja 1: Stanem jest macierz:

$$P(l,\eta) = \left[p_{n,i}(l,\eta)\right]_{\substack{(l = \overline{1,L_\eta}) \\ (i = \overline{1,2})}} \tag{9}$$

gdzie:

η - numer warstwy,

l - numer stanu w ramach warstwy.

Elementy macierzy (9) określamy następująco:

$$\bigvee_{1 \leq n \leq N} p_{n,1}(l,\eta) = \begin{cases} m - \text{jeśli zadanie } \omega_n \text{ zostało przydzielone} \\ \quad \text{operatorowi } w_m; \\ 0 - \text{w przeciwnym przypadku.} \end{cases}$$

$$\bigvee_n \left[p_{n,1}(l,\eta) \quad 0\right] \Rightarrow \left[p_{n,2}(l,\eta) = t_n\right] \tag{10}$$

gdzie:

t_n - moment zakończenia realizacji zadania ω_n.

Tak więc stan $P(1,0)$ jest macierzą zerową, natomiast stan $P(l,N)$ (dla $1 \leq l \leq L_N$) jest macierzą o niezerowych elementach.

Generowanie stanów jest procedurą pozwalającą wyznaczyć pewien stan $P(\lambda, \eta + 1)$ (gdzie: $1 \leq \lambda \leq L_{\eta+1}$) na podstawie stanu $P(l,\eta)$. Przejście to wymaga uzupełnienia stanu $P(l,\eta)$ o pewne zadanie ω_n. Zadanie to mo-

że być przydzielone tylko temu operatorowi, który ma odpowiednie kwalifikacje, tzn. $b_{m,n} = 1$.

Ogólna procedura generowania stanów ma następującą postać:

$$\bigvee_n \bigvee_m \left[p_{n,1}(1,\eta) = 0\right] \wedge (b_{m,n} = 1) \wedge (Q_{k,n} = 1) \Rightarrow \left[P(\lambda,\eta+1) = \right.$$

$$\left. = P(1,\eta) + \Delta P(1,\eta : \lambda : \quad + 1)\right] \tag{11}$$

Macierz $\Delta P(1,\eta;\lambda,\eta+1)$ ma postać:

$$\Delta P(1,\eta:\lambda,\eta+1) = \begin{bmatrix} 0 & 0 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ 0 & 0 \\ m & t_n \\ 0 & 0 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ 0 & 0 \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}0\\\cdot\\\cdot\\\cdot\\0\end{matrix}}\right\} n-1 \text{ wierszy} \\ \\ \left.\vphantom{\begin{matrix}0\\\cdot\\\cdot\\\cdot\\0\end{matrix}}\right\} N-n \text{ wierszy} \end{matrix} \tag{12}$$

Moment t_n wyznaczamy następująco:

$$\bigvee_\vartheta \left[p_{\gamma,1}(1,\eta) = m\right] \wedge \left[\max_\vartheta p_{\vartheta,2}(1,\eta) = p_{\mu,2}(1,\eta)\right] \wedge \left[a_{\varkappa\mu} = 1\right] \Rightarrow \left[t_n = \right.$$

$$\left. = p_{\mu,2}(1,\eta) + t_{\varkappa,k} + \tau_n\right] \tag{13}$$

A zatem z warunku (11) można na podstawie stanów warstwy η-tej wygenerować stany warstwy $\eta + 1$-szej:

Istotną rolę w algorytmie odgrywają tzw. stany alternatywne zdefiniowane następująco:

Definicja 2: Stany $P(l_1,\eta)$ i $P(l_2,\eta)$ (gdzie $l_1 \neq l_2$) nazywamy alternatywnymi, jeżeli spełniają warunek:

$$\bigvee_{1 \leqslant m \leqslant M} \bigvee_n \left\{\left[p_{n,1}(l_1,\eta) = m\right] \Leftarrow \left[p_{n,1}(l_2,\eta) = m\right]\right\} \wedge \left[\mu_m(l_1,\eta) = \right.$$

$$\left. = \mu_m(l_2,\eta)\right] \Rightarrow \left[P(l_1,\eta) \triangleq P(l_2,\eta)\right] \tag{14}$$

gdzie:

$\triangleq$ - symbol alternatywności stanów.

Wartość $\mu_m(1,\eta)$ oznaczają numer ostatniego zadania przydzielonego operatorowi w_m. Wartości te wyznaczamy ze stanu $P(1,\eta)$ następująco:

$$\bigvee_{\vartheta}\left[p_{\vartheta,1}(1,\eta) = m\right] \wedge \left[\max_{\vartheta} p_{\vartheta,2}(1,\eta) = p_{\mu,2}(1,\eta)\right] \Rightarrow \left[\mu_m = \mu_m(1,\eta)\right] \tag{15}$$

W odniesieniu do stanów alternatywnych będzie można zastosować pewną procedurę, zwaną regułą dominacji. Celem tej reguły jest wyeliminowanie z dalszych obliczeń stanów nieperspektywicznych, które nie prowadzą do rozwiązania optymalnego. Reguła dominacji stanów zostanie przytoczona poniżej.

3.2. Wartość stanów

Z każdym stanem $P(1,\eta)$ związżemy jego wartość, którą będziemy oznaczać przez $V(1,\eta)$.

Definicja 3: Wartością stanu jest moment zakończenia realizacji zadań, określamy następująco:

$$\bigvee_{\vartheta}\left[p_{\vartheta,1}(1,\eta) = m\right] \wedge \left[\max_{\vartheta} p_{\vartheta,2}(1,\eta) = p_{\mu,2}(1,\eta)\right] \wedge \left[a_{k_m,\mu_m} = 1\right] \Rightarrow$$

$$\Rightarrow \left\{V(1,\eta) = \max_{1\le m\le M}\left[p_{\mu_m,2}(1,\eta) + t_{k_m,1}\right]\right\} \tag{16}$$

A zatem wartość stanu określa, po jakim czasie operatorzy powrócą do s_1 po wykonaniu wszystkich zadań należących do tego stanu. Jako $V(1,0)$ można przyjąć zerową wartość.

Zauważmy przy tym, że:

$$V_m(1,\eta) = p_{\mu_m,2}(1,2) + t_{k_m,1} \tag{17}$$

gdzie:

$V_m(1,\eta)$ - moment powrotu operatora w_m do punktu s_1 po wykonaniu zadań.

Przy przejściu do stanu $P(1,\eta)$ ulega zmianie tylko moment (17) dla jednego z operatorów. Zatem, gdyby wartości (17) były zapamiętywane (dla każdego m w każdym stanie), to obliczenie wartości stanu opierałoby się na formule:

$$V(1,\eta) = \max\left[V_m(1,\eta);\ \max_{\substack{1\le r\le M\\ r\ne m}} V(\lambda,\eta-1)\right] \tag{18}$$

gdzie:

m - numer operatora, któremu przydzielono żądanie przy przejściu od stanu $P(\lambda,\eta-1)$ do stanu $P(l,\eta)$.

Korzystając z formuły (18) obciąża pamięć komputerową, dlatego wartość stanu lepiej obliczać wprost z (16).

Optymalny stan ostatniej warstwy wyznaczamy z warunku:

$$\exists_{l_o \; 1\leq l\leq L_N} \left[V(1,N) = V(l_o,N)\right] \Rightarrow \left[P(l_o,N) = P_{opt}(N)\right] \qquad (19)$$

gdzie:

$P_{opt}(N)$ - optymalny stan ostatniej warstwy.

Zauważmy, że (19) wynika z funkcji celu (8). Ze stanu (19) wyznaczamy optymalny harmonogram realizacji zadań. A więc:

$$\forall_{1\leq m\leq M} \forall_{n} \left[p_{n,1}(l_o,N) = m\right] \Rightarrow \left[t_n = p_{n,2}(l_o,N)\right] \wedge \left[\xi_n = t_n - \tau_n\right] \qquad (20)$$

gdzie:

t_n - moment zakończenia wykonywania zadania ω_n,

ξ_n - moment rozpoczęcia wykonywania zadania ω_n.

Wartość stanu optymalnego jest minimalnym czasem, po którym operatorzy wykonają wszystkie zadania i powrócą do punktu s_1.

3.3. Dominacja stanów

Ciąg stanów od warstwy zerowej do warstwy N-tej nazwiemy trajektorią. Każda trajektoria interpretuje dopuszczalne rozwiązanie problemu. Zauważmy, że każda trajektoria wychodzi ze stanu $P(1,0)$. Końcowym stanem trajektorii jest $P(l,N)$ (gdzie: $1\leq l\leq L_N$). Trajektoria optymalna ma końcowy stan optymalny $P(l_o,N)$.

W opisanym algorytmie liczba stanów, które należy zapamiętywać w trakcie obliczeń, może być zbyt duża, nawet w przypadku zapamiętywania stanów dwóch sąsiednich warstw. Dlatego też istotną rolę odgrywają reguły, które pozwalają rozstrzygnąć, czy jeden ze stanów $P(l_1,\eta)$ lub $P(l_2,\eta)$ jest nie perspektywiczny. Oznacza to, że posuwając się po trajektorii wychodzącej z tego stanu (np. $P(l_2,\eta)$ otrzymamy gorsze rozwiązanie aniżeli wychodząc ze stanu alternatywnego (np. $P(l_1,\eta)$).

Rozważmy dwie trajektorie wychodzące z $P(1,0)$. Niechaj pierwsza trajektoria przechodzi przez stan $P(l_1,\eta)$ i kończy się w stanie $P(l_1',N)$. Natomiast druga trajektoria przechodzi przez stan $P(l_2,\eta)$ i kończy się w stanie $P(l_2',N)$.

Oznaczamy wartości tych stanów odpowiednio: $V(l_1,\eta)$ i $V(l_1',N)$ oraz $V(l_2,\eta)$ i $V(l_2',N)$.

Definicja 4: Stan $P(l_1,\eta)$ dominuje nad stanem $P(l_2,\eta)$, jeżeli jest spełniony warunek:

$$\left[P(l_1,\eta) \hat{\in} P(l_2,\eta)\right] \wedge \left[V(l'_1,N) < V(l'_2,N)\right] \Rightarrow \left[P(l_2,\eta) \mapsto P(l_1,\eta)\right] \quad (21)$$

gdzie:

$\mapsto$ - symbol dominacji stanów.

Jednakże w obliczeniach dominację stanu $P(l_1,\eta)$ nad stanem $P(l_2,\eta)$ należy rozstrzygnąć w warstwie η-tej a nie po zakończeniu obliczeń w warstwie N-tej. Problem ten rozwiązuje następujące twierdzenie.

TWIERDZENIE: Jeżeli dla: $1 < \eta < N$ zachodzi:

$$\exists_{1\le m\le M} \exists_{1\le r\le M} \left[P(l_1,\eta) \hat{\in} P(l_2,\eta)\right] \wedge \left[V_m(l_1,\eta) \le V_m(l_2,\eta)\right]$$

$$\wedge \left[V_r(l_1,\eta) < V_r(l_2,\eta)\right] \Rightarrow \left[P(l_2,\eta) \Rightarrow P(l_1,\eta)\right] \quad (22)$$

D o w ó d (nie wprost):

Załóżmy, że stan $P(l_2,\eta)$ dominuje nad stanem $P(l_1,\eta)$. Dla stanów alternatywnych $P(l_1,\eta)$ i $P(l_2,\eta)$ zachodzi:

$$\mu_m(l_1,\eta) = \mu_m(l_2,\eta) \quad \forall_{1\le m\le M} \quad (23)$$

A więc momenty zakończenia realizacji zadań (bez powrotu do s_1) dla poszczególnych operatorów wyznaczamy jako:

$$T_m(l,\eta) = V_m(l,\eta) - t_{\mu_m,1} \quad \forall_{1\le m\le M} \quad (24)$$

gdzie:

$T_m(l,\eta)$ - moment zakończenia realizacji zadań przez w_m.

Ponadto

$$\mu_m = \mu_m(l_1,\eta) = \mu_m(l_2,\eta) \quad (25)$$

Z (22), (24) i (25) wynika, że:

$$T_m(l_1,\eta) \le T_m(l_2,\eta) \quad \forall_{1\le m\le M} \quad (26)$$

oraz:

$$T_r(l_1,\eta) < T_r(l_2,\eta) \quad (27)$$

Oznaczamy optymalny harmonogram realizacji pozostałych $N-\eta$ zadań (nie należących do $P(l_2,\eta)$) od stanu $P(l_2,\eta)$, przez $H(l_2,\eta)$. Niechaj składowe $h_m(l_2,\eta)$ tego harmonogramu odpowiadają operatorom w_m. Harmonogramy $h_m(l_2,\eta)$ są realizowane od momentów $T_m(l_2,\eta)$.

Załóżmy, że czasy realizacji tych harmonogramów oznaczymy przez $V\left[h_m(l_2,\eta)\right]$. A zatem:

$$V(l_2',N) = \max_{1\leq m\leq M}\left\{T_m(l_2,\eta) + V\left[h_m(l_2,\eta)\right]\right\} \tag{28}$$

Harmonogramy $h_m(l_2,\eta)$ można realizować od $P(l_1,\eta)$ do momentów $T_m(l_1,\eta)$, przy czym:

$$V(l_1',N) = \max_{1\leq m\leq M}\left\{T_m(l_1,\eta) + V\left[h_m(l_2,\eta)\right]\right\} \tag{29}$$

Uwzględniając (26) i (27) w (28) i (29) otrzymamy:

$$V(l_1',N) \leq V(l_2',N) \tag{30}$$

skąd wynika, że stan $P(l_2,\eta)$ nie dominuje nad stanem $P(l_1,\eta)$, co przeczy założeniu. Tym samym twierdzenie (22) zostało udowodnione.

Wniosek: Na podstawie (24) i (25) otrzymujemy warunek:

$$\bigvee_{1\leq m\leq M}\ \bigwedge_{1\leq r\leq M}\left[P(l_1,\eta) \,\hat{=}\, P(l_2,\eta)\right] \wedge \left[T_m(l_1,\eta) \leq T_m(l_2,\eta)\right] \wedge \left[T_r(l_1,\eta) < T_r(l_2,\eta)\right] \Rightarrow \left[P(l_2,\eta) \mapsto P(l_1,\eta)\right] \tag{31}$$

Ponieważ:

$$T_m(1,\eta) = p_{\mu_m,2}(1,\eta) \tag{32}$$

Zatem w obliczeniach łatwiej jest posługiwać się momentem $T_m(l_2,\eta)$.

4. Przykład

Załóżmy, że dany jest system złożony z czterech elementów:

$$S = \left\{s_1, s_2, s_3, s_4\right\}$$

Element s_1 jest punktem startu i powrotu operatorów. Macierz czasów transportu ma postać:

$$T = \begin{bmatrix} 0 & 2 & 3 & 4 \\ 2 & 0 & 1 & 5 \\ 2 & 2 & 0 & 4 \\ 3 & 4 & 5 & 0 \end{bmatrix}$$

Dany jest zbiór czterech zadań:

$$\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4\}$$

Czasy wykonywania zadań zapisane są w wektorze:

$$\tau = \begin{bmatrix} 3 \\ 2 \\ 4 \\ 3 \end{bmatrix}$$

Lokalizację zadań opisuje macierz:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Załóżmy, że zadania realizuje dwóch operatorów:

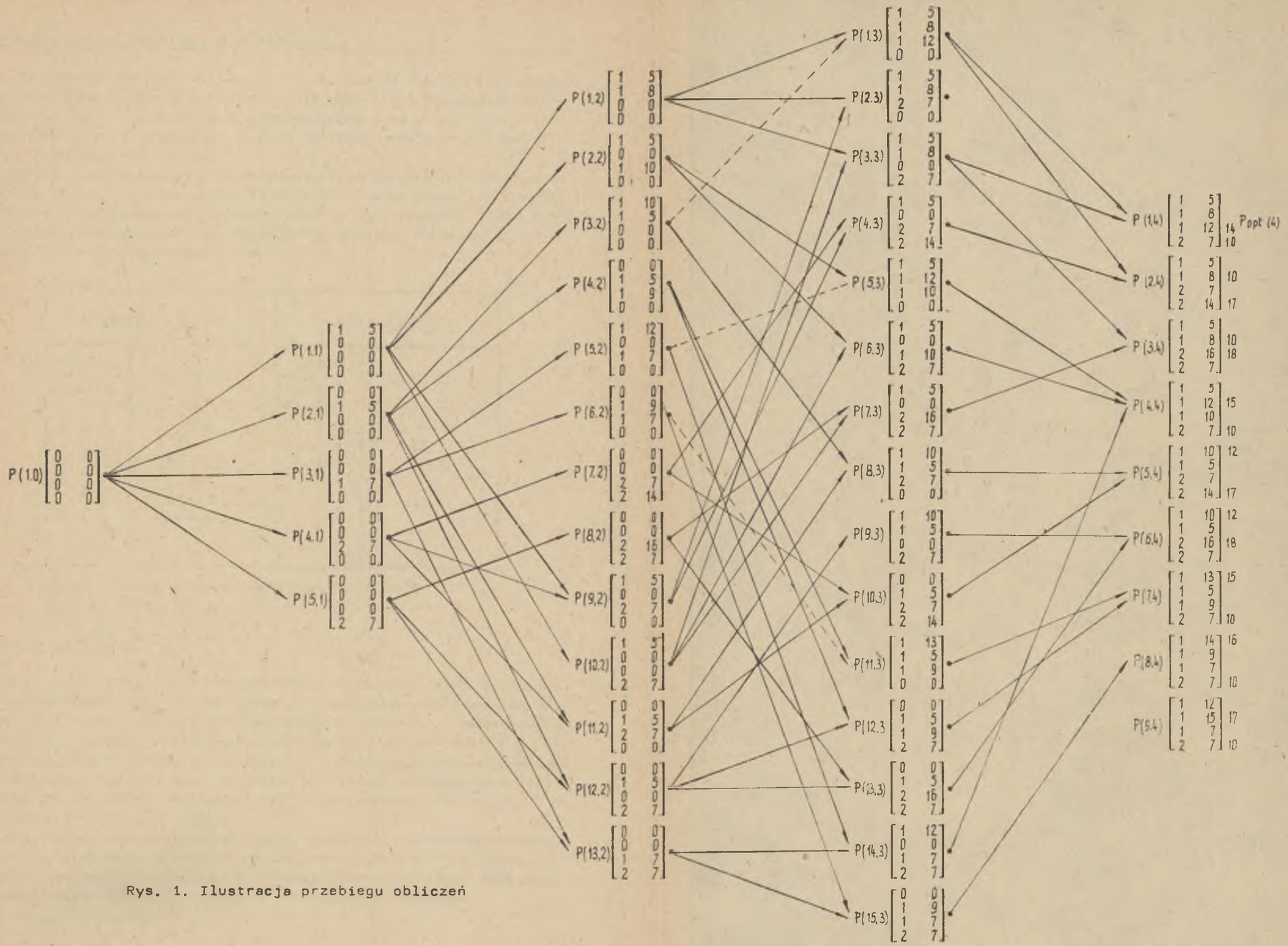
$$W = \{w_1, w_2\}$$

Macierz kwalifikacji operatorów ma postać:

$$B = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Dla powyższych danych należy wyznaczyć optymalny harmonogram realizacji zadań.

Rozwiązanie problemu wg podanego wyżej algorytmu pokazano na rys. 1. Jak widać, stany pogrupowano w pięciu warstwach. Najwięcej stanów występuje w trzeciej warstwie (piętnaście). Łuki skierowane wskazują przejścia pomiędzy stanami. Łuki narysowane linią przerywaną wskazują przejścia, które prowadzą do stanów zdominowanych (np. stan P(1,3) dominuje nad alternatywnym stanem, który można wygenerować wychodząc ze stanu P(3,2)). Stanów zdominowanych nie pokazano na rys. 1. W ostatniej (czwartej) warstwie nie stosowano reguły dominacji. Stosowanie reguły dominacji w tej warstwie jest możliwe (np. P(9,4)→P(4,4) oraz P(8,4)→P(7,4)), ale nie konieczne. Mając stany ostatniej warstwy można wprost wyznaczyć op-



Rys. 1. Ilustracja przebiegu obliczeń

tymalny harmonogram realizacji zadań, co pokazano na rys. 1. Obok stanów ostatniej warstwy zaznaczono (w kółkach) momenty $V_m(1,4)$ powrótów operatorów do punktu s_1. Na tej podstawie wyznaczamy stan $P(1,4)$ jako stan optymalny. Warto zwrócić uwagę na fakt, że tylko jeden stan jest optymalny, dla którego $V(1,4) = 14$.

Natomiast istnieją stany dopuszczalne (dające realizowalne harmonogramy), których wartości są większe o cztery jednostki czasowe. Na rysunku 2 przedstawiono harmonogramy realizacji zadań przez operatorów. Symbol $s_1 - s_k$ oznacza transport pomiędzy odpowiednimi punktami, natomiast ω_n oznacza wykonywanie odpowiedniego zadania.

OPERATOR 1			OPERATOR 2		
czas		czynności	czas		czynności
od	do		od	do	
0	2	$S_1 \rightarrow S_2$	0	4	$S_1 \rightarrow S_4$
2	5	ω_1	4	7	ω_4
5	6	$S_2 \rightarrow S_3$	7	10	$S_4 \rightarrow S_1$
6	8	ω_2			
8	12	ω_3			
12	14	$S_3 \rightarrow S_1$			

Rys. 2. Harmonogram realizacji zadań

5. Uwagi końcowe

Przedstawiony algorytm harmonogramowania realizacji zadań wymaga zastosowania do obliczeń maszyny cyfrowej. W praktyce wielkość dostępnej pamięci operacyjnej komputera może mieć istotne znaczenie. Rozpatrywanie problemów o większej liczbie ograniczeń jest bardziej adekwatne dla tego algorytmu. Ponadto lepsze reguły dominacji stanów zmniejszają zajętość pamięci komputerowej. Wprowadzenie leksykograficznego porządku generowania stanów pozwala uniknąć generowania stanów identycznych (alternatywnych) o jednakowych wartościach $V_m(1,\eta)$.

Jak wynika z przeprowadzanych analiz, wyznaczenie optymalnego harmonogramu bez wykorzystania maszyny cyfrowej jest w praktyce niemożliwe. Natomiast stosowanie heurystycznych harmonogramów realizowalnych może powodować straty w produkcji, z uwagi na nieoptymalność rozwiązań.

LITERATURA

[1] Bellman R.: Dynamic Programming Treatment of the Travelling Salesman Problem, Jour. Assoc. Compt. Machinery, Vol. 9, No. 1, 1962, pp 61--63.

[2] Bellman R.: Adaptacyjne procesy sterowania. PWN, Warszawa 1965, ss. 80-92.

[3] Bazalel Gravish: A Note on "The Formulation of the M - Salesman Travelling Salesman Problem, Management Science, Vol. 22, No 6, 1976, pp. 704-705.

[4] Dusza K., Kowalowski H., Marecki F.: Sterowanie dyspozytorskie obsługą robót spawalniczych na kopalni węgla kamiennego. Materiały VIII Sympozjum nt. "Systemy zarządzania i sterowania kopalniami", Komitet Górnictwa PAN, Szklarska Poręba 1979, ss. 164-174.

[5] Held M., Karp R.M.: A dynamic Programing Approach to Sequencing Problems, Jour. Soc. Indust. Appl. Math., Vol. 10, No. 1 1962, pp. 196-210.

[6] Held M., Karp R.M.: The Construction of Discrete... Dyn. Progr. Alg. IBM System Journal, Vol. 4, No 2, 1965, pp. 136-147.

[7] Kowalowski H., Matecki F., Dusza K.: Sterowanie dyspozytorskie remontami obudów hydraulicznych, ICAMC - 80, Katowice 1980 (komunikat).

[8] Marecki F.: Harmonogramowanie dostaw detali na linie montażowe. Zeszyty Naukowe Politechniki Śląskiej. Automatyka, Gliwice 1980.

[9] Starzyczny L., Dusza K., Marecki F.: Niezawodność systemu utrzymania ruchu w kopalni węgla kamiennego. Materiały Konferencji nt. "VIII dni jakości i niezawodności; Gliwice 1980.

[10] Svestka J.A., Huckfeldt V.E.: Computational Experience with an M - Salesman Traveling Salesman Algorithm, Management Science, Vol. 19, No 7, 1973, pp. 790-799.

Recenzent: Prof. dr hab. inż. Marian Kozdrój

Wpłynęło do Redakcji 25.02.1981 r.

Составление план-графиков независимых задач бригад беспребойного движения в каменноугольной шахте

Р е з ю м е

В докладе сформулирована математическая модель работы бригад беспребойного движения в каменноугольной шахте. Решена проблема составления план-графиков независимых задач. Для решения предложен алгоритм динамического программирования. Представлен тоже расчетный пример.

Graphic scheduling of independent tasks for the gangs of maintaining traffic at hard coal-mines

S u m m a r y

The paper formulates mathematical archetype of work concerning the gangs of maintaining traffic at hard coal-mines. The problem of graphic scheduling of independent tasks has been solved. This has been carried out by means of the algorithm of dynamical programming. An analytical example has also been given.