Władysław Matwin

ZASTOSOWANIE PODSTAWOWYCH POJĘĆ TEORII AUTOMATÓW DO MODELOWANIA NIEKTORYCH PROBLEMÓW TEORII GIER

Napisane pod kierownictwem doktora
Zdzisława Pawlaka,
docenta Uniwersytetu Warszawskiego

1966

Matematyczna teoria automatów jest dyscypliną bardzo młodą, w encyklopedii nie znajdziemy o niej dziś jeszcze niczego. Nie ulega jednak wątpliwości, że należy do tych badań, które będą rozwijane. Jej głównym przedmiotem zainteresowań jest matematyczno-logiczny mechanizm maszyn cyfrowych. Ale jest ona zarazem pewnym sposobem patrzenia na wiele innych zjawisk, wykraczających poza świat maszyn cyfrowych. Teoria automatów może być przydatna w analizie każdego procesu, niezależnie od jego materialnej natury, jeśli procesem nazwiemy ciąg oddziaływań, stanów i przemian związanych określoną zależnością.

Teoria gier powstała wcześniej, kiedy teorii automatów jeszcze nie było, ma bogatą literaturę i głośne imię. Jednym z zastosowań teorii gier stały się niektóre procesy techniczno-ekonomiczne i zagadnienia związanych z nimi optymalnych decyzji, rachunek prakseologiczny, prowadzony obecnie coraz częściej przy pomocy automatycznych urządzeń liczących.

A co jeśli do analizy takich procesów techniczno-ekonomicznych i związanych z nimi zagadnień spróbujemy zastosować teorię automatów? Czy pojęciem automatu nie można w pewnych przypadkach i w pewnej mierze zastąpić pojęcie gry? Autorem pomysłu jest pan Zdzisław Pawlak. Następne stronice to próba objaśnienia tej propozycji na kilku konkretnych przykładach.

S P I S   R Z E C Z Y

1. A U T O M A T

1. 1. P o d s t a w o w e   p o j ę c i a

A u t o m a t e m   A będziemy nazywali   p i ą t k ę

$A = \langle Q, X, Y, f_q, f_y \rangle$, przy czym

$Q = \{q_0, q_1, \ldots, q_n\}$ jest zbiorem elementów zwanych s t a n a m i wewnętrznymi automatu,

/1.1.1/ $X = \{x_1, x_2, \ldots, x_p\}$ jest zbiorem elementów zwanych s y m b o l a m i   w e j ś c i o w y m i /a l f a - b e t wejściowy/ automatu,

$Y = \{y_1, y_2, \ldots, y_r\}$ jest zbiorem elementów zwanych s y m b o l a m i   w y j ś c i o w y m i /alfabet wyjściowy/ automatu,

$f_q : Q \times X \longrightarrow Q$ oznacza funkcję, która przyporządkowuje każdemu elementowi $/q_i, x_j/$ pewnego podzbioru iloczynu kartezjańskiego $Q \times X$ jeden element $q_s = f_q(q_i, x_j)$ pewnego podzbioru zbioru Q;

$f_y : Q \longrightarrow Y$ oznacza funkcję, która przyporządkowuje każdemu elementowi q_k pewnego pozbioru zbioru Q jeden element $y_l = f_y(q_k)$ zbioru Y;

zbiory Q , X , Y są skończone.

Niech $x = (x_{j_1}, x_{j_2}, \dots, x_{j_k})$ będzie słowem w alfabecie X zadanym automatowi /1.1.1/ w stanie q_{i_1}. Automat odczytuje symbol wejściowy x_{j_1}, przechodzi w stan $f_q(q_{i_1}, x_{j_1}) = q_{i_2}$, wytwarza symbol wyjściowy $f_y(q_{i_2}) = y_{i_1}$, odczytuje symbol wejściowy x_{j_2}, przechodzi w stan $f_q(q_{i_2}, x_{j_2}) = q_{i_3}$ itd.

W zbiorze Q można wyróżnić element q_α, np. q_0 jako stan p o c z ą t k o w y, podzbiór $Q' = \{q_0, q_1, \dots, q_m\}$ stanów c z y n n y c h i podzbiór $Q^* = \{q_{m+1}, q_{m+2}, \dots, q_n\}$ stanów k o ń c o w y c h. Stan q_α nazywamy początkowym, jeśli automat rozpoczyna działanie w stanie q_α. Stan q_i nazywamy czynnym, jeśli istnieje $x_j \in X$ taki, że $f_q(q_i, x_j) \in Q$. Stan q_i nazywamy końcowym, jeśli $f_q(q_i, x_j) \in \emptyset$ dla każdego $x_j \in X$; symbol $\emptyset$ oznacza zbiór pusty.

Automat A ze stanem początkowym q_0 i podzbiorami Q', Q^* zapiszemy jako s z ó s t k ę,

/1.1.2/
$$A = \langle Q, X, Y, f_q, f_y, q_0 \rangle,$$
$$O = \{q_0, q_1, \dots, q_m, q_{m+1}, \dots, q_n\} \quad \text{itd.}$$

Powiemy, że automat /1.1.1/ p r z y j m u j e w stanie q_i słowo $x = (x_{j_1}, x_{j_2}, \dots, x_{j_k})$, jeśli odczytując x, A osiąga kolejno k stanów:

$q_{i_1}, q_{i_2}, \dots, q_{i_k}$ takich, że $f_q(q_i, x_{j_1}) = q_{i_1}$, $f_q(q_{i_1}, x_{j_2}) = q_{i_2}, \dots, f_q(q_{i_{k-1}}, x_{j_k}) = q_{i_k}$.

Funkcję f_y można również określić jako odwzorowanie pewnego podzbioru iloczynu Q x Q w alfabet Y a funkcje f_q i f_y można zastąpić funkcją $f_{qy}: Q \times X \longrightarrow Q \times Q$. Funkcje f_q, f_y, f_{qy} mogą być zadane przy pomocy t a b l i c lub g r a f u. W niektórych przypadkach możliwe jest przedstawienie analityczne.

1. 2. Przykład. Samoczynna łącznica

Działanie samoczynnej łącznicy na trzy aparaty telefoniczne daje się przedstawić przy pomocy automatu

$A_t = \langle Q, X, Y, f_q, f_y, q_0 \rangle$, przy czym

$Q = \{q_0, q_1, q_2, q_3, q_{12}, q_{13}, q_{21}, q_{23}, q_{31}, q_{32}, p_{12}, p_{13}, p_{23}\}$,

$X = \{s_1, s_2, s_3, x_1, x_2, x_3, x_{12}, x_{13}, x_{21}, x_{23}, x_{31}, x_{32}, x_1 \wedge x_2, x_1 \wedge x_3, x_2 \wedge x_3\}$,

$Y = \{y_1, y_2, y_3\}$,

$f_q : Q \times X \longrightarrow Q$ dana jest grafem

$f_y : Q \longrightarrow Y$ dana jest tabliczką,

q_{12}	q_{13}	q_{21}	q_{23}	q_{31}	q_{32}
y_2	y_3	y_1	y_3	y_1	y_2

Słowo $x = (s_2, x_{21}, s_1, x_1 \wedge x_2)$ oznacza tutaj ciąg działań: podniesienie słuchawki 2, wybranie cyfry 1 w aparacie 2, podniesienie słuchawki 1, położenie na widełki słuchawek 1 i 2. Automat A zaczyna pracę w stanie q_0 /wszystkie słuchawki na widełkach/, kończy również w stanie q_0. Funkcja f_y przyporządkowuje stanowi q_{21} symbol wyjściowy y_1 - dzwonek w aparacie 1 gdy A w stanie q_2 przyjął symbol x_{21} i przeszedł w q_{21}.

Automat A_t jest obrazem działania nie tylko łącznicy telefonicznej. A_t obrazuje zarazem wiele innych zjawisk rozmaitej natury. Może on być całkiem dobrze obrazem np. działań wojennych. Słowo $x = (s_1, x_{12}, x_1)$ może oznaczać wówczas, że atakujący, który przeprowadził uderzenie s_1 otrzymał kontruderzenie x_{12}, po czym wycofał się z pola.

1. 3. P r z y k ł a d. Maszyna Turinga.

Niech będzie

$A_M = \langle Q, X, Y, f_{qy}, q_1 \rangle$,

$Q = \{q_0, q_1, q_2, !\}$,

$X = \{0, 1, 2, \ldots, 9, \emptyset, |\}$,

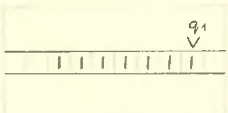
$Y = \{0L, 1N, 2N, \ldots, 9N, \emptyset L, L, P\}$,

$f_{qy} : Q \times X \longrightarrow Q \times Y$ przyporządkowuje parze

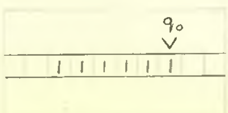
$(q_{i_k}, x_{j_k}) \in Q \times X$ parę $(q_{i_{k+1}}, y_{l_k}) \in Q \times Y$ i dana jest tabliczką

	0	1	2	3	4	5	6	7	8	9	∅	|
q_0	q_2 1N	q_2 2N	q_2 3N	q_2 4N	q_2 5N	q_2 6N	q_2 7N	q_2 8N	q_2 9N	q_0 0L	q_2 1N	q_0 L
q_1	!	!	!	!	!	!	!	!	!	!	!	q_0 ∅L
q_2	q_2 P	q_2 P	q_2 P	q_2 P	q_2 P	q_2 P	q_2 P	q_2 P	q_2 P	q_2 P	q_1 L	q_2 P

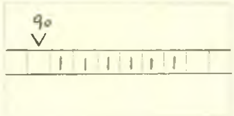
Maszyna M składa się z pamięci /taśma podzielona na kratki/ i z trzech mechanizmów: automatu A_M, czytnika i urządzenia piszącego. Kratki taśmy są puste albo zawierają po jednym symbolu alfabetu X. Niech w 7 kolejnych miejscach pamięci będzie zapisany układ 7 pałeczek |. Niech czytnik znajduje się nad ostatnią pałeczką po prawej i niech A_M będzie w stanie początkowym q_1.



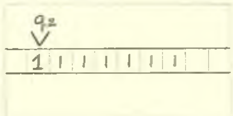
A_M przejdzie więc w stan q_0 i wyprodukuje symbol wyjściowy ∅ L : na miejscu odczytanej pałeczki zapisany będzie symbol ∅ - znak pusty /pałeczka zostanie wymazana/ a czytnik przesunie się w lewo.



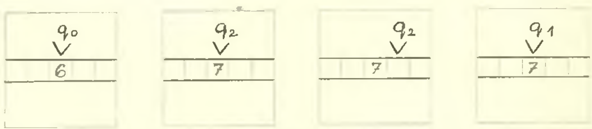
Ponieważ $f_{qy}\left(q_0, |\right) = \left(q_0, L\right)$, A_M pozostanie w stanie q_0 a czytnik znów przesunie się w lewo itd., aż osiągnięty zostanie układ



Wówczas $f_{qy}\left(q_0, \emptyset\right) = \left(q_2, 1\ N\right)$, czyli A_M przejdzie w stan q_2, w pustą kratkę maszyna wpisze symbol wyjściowy 1 i czytnik pozostanie w tym samym miejscu,



Wobec tego, że $f_{qy}\left(q_2, 1\right) = \left(q_2, P\right)$, nastąpi seria przesunięć w prawo aż do pierwszej pustej kratki, wtedy $f_{qy}\left(q_2, \emptyset\right) = \left(q_1, L\right)$, dalej - wymazanie pałeczki, jak na początku itd. Po wymazaniu ostatniej pałeczki nastąpią kolejno układy



Teraz $f_{qy}\left(q_1, 7\right) = !$, co oznacza zatrzymanie maszyny.

Tak więc maszyna M przelicza pałeczki i zapisuje wynik w systemie dziesiętny.

Jeśli pamięć maszyny M będzie miała dowolną liczbę kratek, wówczas M będzie - znaną z teorii maszyn cyfrowych i teorii algorytmów - maszyną Turinga.

1. 4. O b i e k t y o b r a z o w a n e p r z e z a u t o m a t

To, co nazywamy automatem jest strukturą matematyczno-logiczną, abstrakcyjną kategorią pojęciową takiego typu i zasięgu, jak na przykład obraz zależności funkcjonalnej. Może być automat obrazem mechanizmów działających w maszynie cyfrowej. Można przez automat wyrazić to, co uznamy za najważniejsze w zmiennym, zależnym od środowiska i stanów wewnętrznych zachowaniu się zwierzęcia, lub w ciągu zdarzeń następujących po sobie w jego sieci nerwowej. Używamy pojęcia automatu do ścisłego badania dynamiki zjawisk techniczno - ekonomicznych. Jest przydatny do analizy takich fenomenów, jak język i gramatyka. Może być automat obrazem wszystkich ewentualności jakie mogą nastąpić w przebiegu działań wojennych lub w skomplikowanej grze między dużą liczbą współdziałających albo zwalczających się uczestników.

Jako uogólniony obraz p r o c e s u i możliwości w nim zawartych, automat oddaje to co jest wspólne dla całej klasy obrażowanych przez niego obiektów.

2. M O D E L O W A N I E   A U T O M A T U

2. 1. O b r a z   i   m o d e l

Niech będą dane automaty

$$A = \langle Q , X , Y , f_q , f_y \rangle ,$$
$$Q = \{ q_1 , \ldots , q_m , q_{m+1} , \ldots , q_n \} ,$$
$$X = \{ x_1 , \ldots , x_v \} , \quad Y = \{ y_1 , \ldots , y_z \} ,$$
$$f_q : Q \times X \longrightarrow Q , \; f_y : Q \longrightarrow Y , \text{ oraz}$$
$$\bar{A} = \langle \bar{Q} , \bar{X} , \bar{Y} , \bar{f}_q , \bar{f}_y \rangle ,$$
$$\bar{Q} = \{ \bar{q}_1 , \ldots , \bar{q}_m , \bar{q}_{m+1} , \ldots , \bar{q}_n \} ,$$
$$\bar{X} = \{ \bar{x}_1 , \ldots , \bar{x}_v \} , \quad \bar{Y} = \{ \bar{y}_1 , \ldots , \bar{y}_z \} ,$$
$$\bar{f}_q : \bar{Q} \times \bar{X} \longrightarrow \bar{Q} , \; \bar{f}_y : \bar{Q} \longrightarrow \bar{Y}$$

Powiemy, że automaty A i $\bar{A}$ są i z o m o r f i c z n e , jeśli istnieją przyporządkowania wzajemnie jednoznaczne

$$q_i \in Q \longrightarrow \bar{q}_i \in \bar{Q} , \quad x_j \in X \longrightarrow \bar{x}_j \in \bar{X} , \quad y_h \in Y \longrightarrow \bar{y}_h \in \bar{Y} ,$$

przy czym

$$\left[f_q (q_i , x_j) = q_k \wedge f_y (q_k) = y_h \right] \Longleftrightarrow \left[\bar{f}_q (\bar{q}_i , \bar{x}_j) = \bar{q}_k \wedge \bar{f}_y \, \bar{q}_k = \bar{y}_h \right] .$$

Przypuśćmy, że $\bar{A}$ jest fizycznie istniejącym obiektem o wysokim stopniu złożoności. Badanie tego obiektu zjajduje się w takim stadium, że mamy już obraz obiektu $\bar{A}$ w postaci właśnie automatu A , jednakże dalecy jeszcze jesteśmy od wyjaśnienia interesujących nas problemów obiektu $\bar{A}$, bo jego obraz A jest skomplikowany, zbiór Q liczy wiele dziesiątków elementów, a funkcje f_q i f_y są zawiłe. Budujemy więc automat $\bar{\bar{A}}$ w postaci urządzenia technicznego, które nazwiemy m o d e l e m automatu A. $\bar{\bar{A}}$ tak samo jak $\bar{A}$ jest izomorficzny względem A i daje się łatwo dowolną ilość razy uruchomić. Obserwując działanie urządzenia $\bar{\bar{A}}$ studiujemy automat A i kontynuujemy badanie obiektu $\bar{A}$.

Ten sposób rozwiązywania problemów związanych z działaniem automatu A nazwiemy m o d e l o w a n i e m automatu A.

Jeżeli dysponujemy maszyną cyfrową M z pamięcią odpowiadającą rozmiarom automatu A, to mamy niemalże gotowy jego model $\bar{\bar{A}}$. M jest bowiem sama pewnym automatem, którego elementy można zużytkować jako elementy urządzenia $\bar{\bar{A}}$, to znaczy tak zaprogramować M, że będzie ona dokładnie naśladować automat A.

2. 2. P r o g r a m

Dajmy na to, że chcemy wiedzieć, jakie spośród stanów końcowych $q_{m+1}, \dots , q_n$ automatu A mogą być osiągnięte, jeśli na wejście automatu A podamy wszystkie słowa ~~w~~ postaci $(x_{j_1}, x_{j_2}, \dots , x_{j_w})$ złożone z w symboli wejściowych. Rozwiązaniem tego zadania jest następujący program algolowy:

```
begin integer i,j,k,m,r,s,t,v,w; input (m, v, w) ;
  begin
    integer array a [1:m, 1:v] , p [1:w, 1:v↑w] ; input(a);
    for j : = 1 step 1 until v do p [1,j] : = a[1,j]×100+j;
    for i : = 2 step 1 until w do
    for k : = 1 step 1 until v ↑ (i-1) do
    for j : v × (k-1) + 1 step 1 until v × k do
```

```
      begin if j ≤ v  then s : = j else s :=j-j ÷ v ;
        r : = p [ i-1,k]÷100  ↑  (i-1)  ;
        if r  ≠  o ∧ r ≤ m  then
        begin  t : = p [i-1,k] - r × 100  ↑ (i-1)  ;
          p [ i,j]  : = a [ r,s]  × 100  i + t   100 + s ;
          if a[r,s] ≥ m  then
          output   (p  [i,j ] ,  i)
        end
        else  p [ i,j ] : = 0
      end
    end
  end;
```

2. 3. I n n e   r o z w i ą z a n i a

Funkcję f_q zadajemy tutaj tablicą a $[1 : m, 1 : v]$, której elementami są numery stanów będących wartościami funkcji f_q. Dla tych par (q_i, x_j) , dla których f_q nie jest określona, przyjmujemy $f_q = 0$.

Instrukcja output $(p [i,j] , i)$ jest symbolicznym oznaczeniem instrukcji polecającej wydrukowanie wszystkich znalezionych przez maszynę par liczb $(p [i,j] , i)$.

W każdej parze liczba i oznacza ilość symboli wejściowych $x_{k_1}, x_{k_2}, \dots, x_{k_i}$ odczytanych w drodze do stanu końcowego $q_{i_{i+1}} = f_q(q_{i_i}, x_{k_i}) = q_u$, którego numer u to pierwsza część zapisu liczby $p [i,j]$, $u = p [i,j] \div 100 \uparrow i$. Następujące po cyfrach liczby u pary cyfr to numery symboli x_{k_1}, $x_{k_2}, \dots, x_{k_i}$. Jeśli np. maszyna wydrukuje parę (67 05 12 01 10 08, 5) to u = 67 a ciągiem, który doprowadził do q_{67} był ciąg $x_5, x_{12}, x_1, x_{10}, x_8$.

Aby wykonać ten program, maszyna musi przebiec wszystkie możliwe słowa w X długości w, wyjąwszy te, które doprowadzają A do jednego ze stanów końcowych wcześniej niż po odczytaniu w symboli wejściowych.

Może być postawione skromniejsze zadanie: chcemy się dowiedzieć przez jakie stany przejdzie A, gdy odczyta jedno słowo $x = (x_{j_1}, x_{j_2}, \dots, x_{j_w})$, przy czym symbol x_{j_i} zostaje odczytany w stanie $q_r \in Q$. To zadanie rozwiązuje program przedstawiony w załączniku 1.

Może też być potrzebny program zlecający odczytanie nie jednego słowa, lecz dowolnego ciągu słów. Wówczas program przedstawiony w załączniku 1, można zastosować wielokrotnie w formie procedury algolowej. Jeśli zlecimy maszynie wytworzenie wszystkich słów długości w nad X a następnie każemy jej przebiec ten zbiór słów, to otrzymamy inne rozwiązanie zadania postawionego przed programem z 2. 2., bardziej racjonalne z punktu widzenia angażowanej pamięci /rezygnujemy z dużej macierzy p $[1:w, \quad 1:v \uparrow w]$ zawierającej $w \times v^w$ elementów/.

3. GRA I AUTOMAT

3.1. Definicja gry

G r ę G skończoną z pełną informacją można zdefiniować jako s z ó s t k ę,

$G = \langle S, P, W, f_s, f_w, s_\alpha \rangle$, przy czym

$S = \{ s_1, s_2, \ldots, s_m, s^*_{m+1}, \ldots, s^*_n \}$ jest zbiorem elementów zwanych s t a n a m i gry; wśród nich wyróżnia się jeden stan s_α jako stan początkowy i jeden lub więcej stanów końcowych;

/3.1.1/ $P = \{ p_1, p_2, \ldots, p_v \}$ jest zbiorem elementów zwanych k r o k a m i elementarnymi;

$W = \{ w_1, w_2, \ldots, w_z \}$ jest zbiorem elementów zwanych w y p ł a t a m i; są to w e k t o r y o tylu składowych, wielu jest uczestników w grze G;

$f_s : S \times P \longrightarrow S$ jest odwzorowaniem zbioru par (s_i, p_j) w zbiór stanów gry;

/#/ $f_s(s_i, p_j) \neq f_s(s_k, p_l)$ dla $i \neq k$;

$f_w : S \longrightarrow W$ jest funkcją w a r t o ś c i u j ą c ą stany gry.

3. 2. Porównanie

Zwróćmy uwagę na zawarty w definicji /3.1.1/ warunek /#/. Oznacza on, że w grze G istnieje tylko jeden ciąg $\left(p_{j_1}, \ldots, p_{j_r}\right)$ kroków elementarnych prowadzący do stanu s_{i_r} : według teorii gier dwie pozycje w partii /tutaj - dwa stany gry/ są równe, jeśli - poza wszystkim innym - mają również tę samą historię. Natomiast w definicji /1.1.1/ automatu A nie nałożyliśmy żadnych warunków na funkcję f_q będącą odpowiednikiem funkcji f_s. Dlatego może istnieć więcej niż jedno słowo $\left(x_{j_1}, \ldots, x_{j_r}\right)$ prowadzące do stanu q_{i_r}.

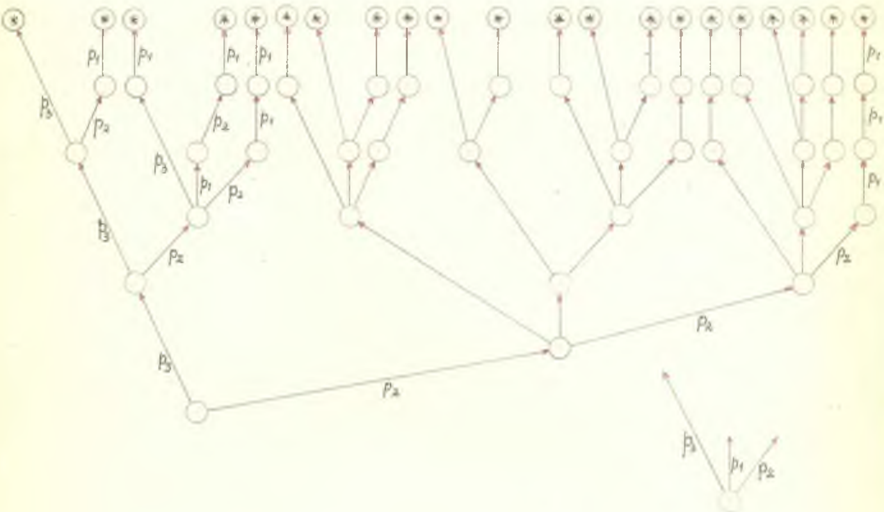
Porównując obie definicje widzimy, że poza ograniczeniem /#/ gra G niczym się nie różni od automatu A, a więc, że pojęcie automatu w sensie /1.1.1/ jest szersze aniżeli pojęcie gry w sensie /3.1.1/.

Tak więc każda gra G jest pewnym automatem A /choć nie na odwrót/; przestrzeń stanów gry to przestrzeń stanów wewnętrznych automatu, zbiór kroków elementarnych to alfabet wejściowy, zbiór wypłat w grze to alfabet wyjściowy, wykres /dendryt/ gry to graf automatu, funkcja wypłat f_w gry to funkcja f_y automatu.

Z rozdziału 2. wynika, że każdy automat A można przy pomocy prostego programu modelować na maszynie cyfrowej. Zatem tak samo daje się modelować każda gra G.

3. 3. Przykład. Nim

W trzech rogach stołu leży po jednej parze zapałek. Pierwszy z dwóch graczy bierze 1 lub 2 zapałki - tylko z jednego rogu, tak samo drugi itd. Wygrywa ten, który weźmie ostatnią zapałkę. Jest to jeden z wariantów starożytnej gry N i m. Gra ~~ta~~ Ta G_1 jest zamknięta, zerowa, wygrana /1/ jednego partnera oznacza przegraną /-1/ drugiego. Można ją śledzić na następującym grafie. Wyraża on wszystkie możliwe przebiegi gry. G_1 ma 58 stanów, w tym jeden początkowy i 22 końcowych. Krok elementarny p_1 oznacza wzięcie jednej samotnej zapałki, p_2 - wzięcie 1 zapałki z kompletnej pary, p_3 wzięcie kompletnej pary.



Widać, że G_1 spełnia warunek /#/, bo w żadne kółko nie wchodzi więcej niż jedna strzałka, do każdego ze stanów prowadzi więc tylko jedna droga.

Rozpoczynający wygrywa biorąc w 1. kroku 2 zapałki, $p_{j_1} = p_3$, bp jeśli $p_{j_2} = p_3$ to w 3. kroku $p_{j_3} = p_3$ daje wygraną rozpoczynającemu. Jeśli zaś $p_{j_2} = p_2$ to $p_{j_3} = p_2$ zmusza do $p_{j_4} = p_1$ i 5. krok $p_{j_5} = p_1$ też daje wygraną rozpoczynającemu.

3. 4. P o s t a ć n o r m a l n a

Wyznaczmy zgodnie z teorią gier s t r a t e g i e obydwu graczy. Będziemy mieli 36 strategii ξ_i dla rozpoczynającego i 28 strategii η_j dla drugiego gracza. Np. $\xi_1 (l, ll)$ oznacza, że w pierwszym chodzie rozpoczynający wykona lewy ruch a w drugim chodzie na lewy ruch partmera odpowie lewym i na prawy również lewym.

Strategia η_{15} (ps, lls) : drugi gracz w pierwszym chodzie na lewy ruch pierwszego gracza odpowie prawym, a na prawy - środkowym, zaś w drugim chodzie na lewy odpowie lewym, na środkowy również lewym, na prawy - środkowym ruchem. Oto część macierzy 36 X 28, która przedstawia grę w postacie normalnej.

	η_1 ll.plp	η_2 ll.ppp	η_3 ls.ll	η_{15} ps.lls	η_{16} ps.llp	η_{17} ps.lpl	η_{18} ps.lps	η_{19} ps.lpp	η_{20} ps.pll	η_{21} ps.pls	η_{28} pp.ppp
ξ_1 l.ll	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
ξ_2 l.ls	1	1	1	-1	-1	1	1	1	-1	-1	1
ξ_3 l.lp	1	1	1	1	1	1	1	1	1	1	1
ξ_8 p.llp	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1
ξ_9 p.sll	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1
ξ_{35} p.ppp.plp	1	1	1	1	-1	1	1	-1	1	1	-1
ξ_{36} p.ppp.ppp	1	1	1	-1	-1	1	-1	-1	1	-1	-1

W ten sposób dowodzi się, że nasza gra ma jedną c z y s t ą strategię optymalną; jest nią $\xi_3(1, 1p)$: jakakolwiek będzie strategia η_i - na przecięciu otrzymamy zawsze 1 czyli wygraną dla rozpoczynającego.

3. 5. U p r o s z c z e n i e

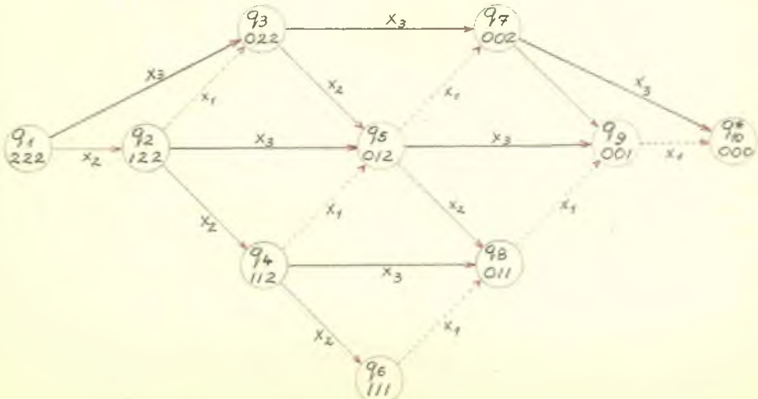
Czy nie można znaleźć jakiegoś bardziej przejrzystego obrazu, wyrażającego wszelkie ewentualności tej gry? / Niech dla automatu

$A_1' = \langle Q, X, Y, f_q, f_y, q_1 \rangle$ będzie

$Q = \{q_1, q_2, \dots, q_9, q_{10}^*\}$, przy czym q_1 jest stanem początkowym: na stole 6 = 2 + 2 + 2 zapałek, q_2 : na stole 5 = 1 + 2 + 2 zapałek, q_3 to stan: 4 = 0 + 2 + 2 zapałki, q_4 to 4 = 1 + 1 + 2 zapałki, ..., q_9 to 1 = 0 + 0 + 1 zapałka, q_{10} to 0 = 0 + 0 + 0 zapałek;

$X = \{x_1, x_2, x_3\}$, przy czym x_j znaczy to ~~samí~~ samo co p_j w 3.3.

Funkcja f : Q × X ⟶ Q ma formę grafu



Automat A_1' nie spełnia warunku /#/ ~~teorii gier~~, mimo to pozwala śledzić naszą grę lepiej niż G_1. Dajmy na to, że mamy do czynienia z bardziej skomplikowanym wariantem gry typu Nim, np. 20 + 20 + 20 = 60 zapałek przy pozostałych zasadach gry takich samych jak poprzednio. Chcemy ją przestudiować przy pomocy modelowania na maszynie cyfrowej. Wydaje się, że powinniśmy wówczas użyć uproszczonego automatu typu A_1'. Można zastosować przy tym interesujące środki oparte na pojęciach funkcji Grundy'ego i jądra grafu, podane w [1].

4. ŁADOWANIE OKRĘTU

4.1. Optymalny rozkład

Niewielki statek, 150 ton wyporności, może załadować materiał a, który jest wart 20 dolarów tona, b po 75 dolarów tona, lub c o wartości 102 dolary tona. Stawka przewozowa jest proporcjonalna do wartości ładunku. Szyper powinien więc przyjąć 150 ton c za 15 ~~606~~ 300 dolarów. Jednakże szkopuł polega na tym, że towar może być ładowany tylko w całych partiach, materiał a po 49 ton, b po 50 ton, c po 51 ton w partii.

Mamy tu do czynienia z grą jednoosobową G_2, w której szyper poszukuje najkorzystniejszego rozkładu swych możliwości. Niech a, b, c oznaczają wzięcie na statek jednej partii odpowiedniego materiału. Potraktujmy te symbole jako wejścia automatu. Zadanie sprowadza się wówczas do znalezienia optymalnego słowa nad alfabetem $\{a, b, c\}$, przy czym szyper ma do wyboru 7 słów:

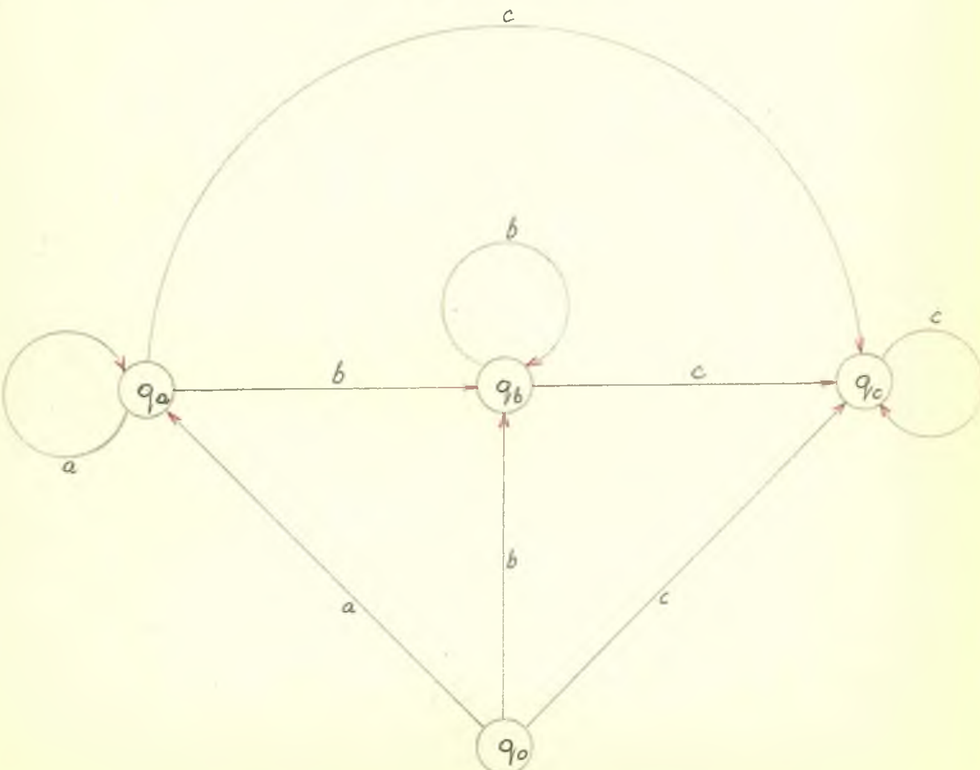
a a a	- 147 ton a	=	2940 dolarów
a a b	- 98 ton a + 50 ton b	=	5710 "
a a c	- 98 ton a + 51 ton c	=	7162 "
a b b	- 49 ton a + 100 ton b	=	8480 "

a b c	– 49 ton a + 50 ton b + 51 ton c	= 9932	dolarów
b b b	– 150 ton b	= 11250	"
c c	– 102 tony c	= 10404	"

Najkorzystniejsza jest więc kombinacja przedostatnia.

4. 2. P r z y p a d e k o g ó l n y

Poszukując rozwiązania, trzeba przejrzeć wszystkie kombinacje, których waga nie przekracza 150 ton. Porządek, w którym symbole a, b, c następują po sobie nie ma znaczenia, (a,b,c) jest warte i waży tyle samo co (c,a,b). Do modelowania gry G_2 można użyć automatu A_2' z takim oto grafem



A'_2 nie spełnia warunku $(\#)$ ale to mu nie przeszkadza dobrze odegrać rolę mechanizmu obrazującego ładowanie okrętu. Przypuśćmy, że materiały a, b, c znajdują się w trzech regionach nabrzeży portowych, q_a, q_b, q_c i że w każdym z regionów odpowiedni materiał zgromadzony jest w partiach w pewnej odległości jedna od drugiej. ~~Powiemy, że~~ Nasz statek wędruje wzdłuż brzegu jak automat A'_2 po swoim grafie: po decyzji (a,b,c) szyper przybije najpierw do q_a, następnie do q_b i q_c, a w przypadku b,b,b – trzy razy do q_b. Powtórzmy to, co o modelowaniu powiedziano w 2. 1 i zbudujmy urządzenie $\overset{=}{A}{}''_2$ złożone z modelu $\overset{=}{A}{}'_2$ automatu A'_2 i trzech mechanizmów, którymi będą: generator słów postaci $(a,a,\ldots, a,b,b, \ldots, b,c,c, \ldots, c)$, waga nie dopuszczająca słów – kombinacji powyżej 150 ton i mechanizm wybierający słowo-kombinację o największej wartości. Jasne, że $\overset{=}{A}{}''_2$ to dla naszego szypra rozwiązanie stojącego przed nim problemu.

W przypadku ogólnym, gdy mamy v różnych materiałów, z których każdy może być ładowany partiami po x_j o wartości y_j a statek ma wyporność z, posłużymy się automatem A'_v,

$$A = \langle Q, X, Y, f_q, f_y, q_0 \rangle,$$
$$Q = \{q_0, q_1, \ldots, q_v\},$$
$$X = \{x_1, x_2, \ldots, x_v\},$$
$$Y = \{y_1, y_2, \ldots, y_v\},$$
$$f_q(q_i, x_j) = q_j, \quad f_y(q_i) = y_i.$$

Aby uzyskać rozwiązanie, zaprogramujemy dla maszyny model $\overset{=}{A}{}''_v$, który z alfabetu X będzie tworzył słowa x postaci

/4.2.1/ $$x = (\overbrace{x_1, x_1, \ldots, x_1}^{a_1 \text{ symboli}}, \overbrace{x_2, x_2, \ldots, x_2}^{a_2 \text{ symboli}}, \ldots, \overbrace{x_v, x_v, \ldots, x_v}^{a_v \text{ symboli}})$$

takiej, że

$$z - \tilde{x}_j < a_1 x_1 + a_2 x_2 + \ldots + a_v x_v \leq z ,$$

a naturalne lub zero,

$\tilde{x}_j$ - masa jednej partii j - tego materiału;

$\stackrel{=}{A}{}^{"}_{v}$ przejrzy wszystkie słowa postaci /4.2.1/ i wybierze ~~te~~ to,

dla którego $\sum_{i=1}^{v} a_i y_i$ osiąga wartość największą.

4. 3. Postać analityczna zadania

Dana jest funkcja n zmiennych

$$R\left(x_1, x_2, \ldots , x_n\right) = g_1\left(x_1\right) + g_2\left(x_2\right) + \ldots + g_n\left(x_n\right)$$

Mamy znaleźć maksimum dla R przy założeniu, że $x_i \geq 0$, $x_1 + x_2 + \ldots + x_n = h$ = dowolnie wybrana stała. Funkcje g_i są osobliwe, z punktami nieciągłości lub nieróżniczkowalności, nieujemne. Klasyczne metody, np. metoda czynników Lagrange'a, nie dają się tu zastosować. Jednakże rozwiązanie przynosi automat analogiczny do A_v'.

Istotnie, niech funkcje g_i będą tablicowane wszystkie z tym samym krokiem Δ. Zamieńmy w A_v' z 4. 2 liczbę v na liczbę n i funkcję $f_y\left(q_i\right) = y_i$ na funkcję $f_y\left(q_i\right) = g_i\left(x_i\right) = g_i\left(a_i \Delta\right)$, a_i naturalne lub zero. Otrzymujemy automat A_n'. Skonstruujmy dla niego model $\stackrel{=}{A}{}^{"}_{n}$ taki samo, jak budowaliśmy modele $\stackrel{=}{A}{}^{"}_{2}$ i $\stackrel{=}{A}{}^{"}_{v}$. Zrealizowany na maszynie, $\stackrel{=}{A}{}^{"}_{n}$ przejrzy wszystkie słowa postaci

$$\Big(\overbrace{x_1, x_1, \ldots, x_1}^{a_1 \text{ symboli}}, \overbrace{x_2, x_2, \ldots, x_2}^{a_2 \text{ symboli}}, \ldots, \overbrace{x_n, x_n, \ldots, x_n}^{a_n \text{ symboli}}\Big)$$

takiej, że $\sum_{i=1}^{n} a_i \triangle = h$ i wybierze to słowo, dla którego

$R = \sum_{i=1}^{n} g_i \left(a_i \triangle \right)$ osiąga maksimum.

4. 4. L i n i o w o ś ć

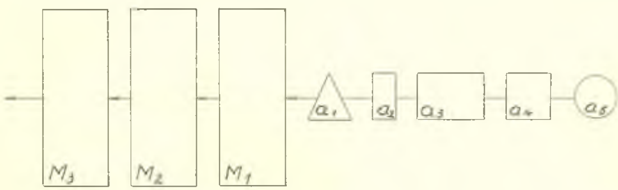
Dana jest funkcja n zmiennych, $L \left(x_1, x_2, \ldots , x_n \right) = \sum_{i=1}^{n} c_i x_i$, chcemy znaleźć jej maksimum przy warunkach $x_i \geq 0$, $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$, $i = 1, 2, \ldots , m; m < n$. Tak formułuje się ogólne zadanie programowania liniowego : L to funkcja celu, warunki $x_i \leqq 0$ to warunki brzegowe, następne m nierówności to warunki bilansowe. Z 4.2 i 4.3 wynika, że rozwiążemy to zadanie budując model $\overset{=}{A}{}''_{L}$ analogiczny do poprzednich, jeśli $\overset{=}{A}{}''_{L}$ wyposażymy nie w jeden lecz w m mechanizmów ważących /nierówności bilandowe/.

Wydaje się, że metoda rozwiązywania zadań programowania liniowego przy pomocy modelu $\overset{=}{A}{}''_{L}$ ma pewne zalety. Jest elementarna, jej idea jest oczywista. Jest odporna na przypadki naruszenia liniowości, widzieliśmy w 4. 3, że nie lęka się najbardziej kapryśnych funkcji typu R. Szkicując mechanizmy, w jakie ma być wyposażony $\overset{=}{A}{}''_{L}$, budujemy zarazem kręgosłup programu dla maszyny. Wreszcie,w niektórych sytuacjach szczególnych /gdy np. poszukujemy rozwiązania całkowitego w niezbyt wielkim przedziale/ metoda modelowania może być racjonalniejsza od metod opartych na algebrze liniowej.

5. LINIA PRODUKCYJNA

5. 1. Optymalna kolejność

Mamy 3 obrabiarki i 5 różnych detali, które kolejno przechodzą najpierw przez pierwszą, potem przez drugą i trzecią obrabiarkę.



Czas obróbki różnych detali na poszczególnych obrabiarkach jest r ó ż n y . Dlatego powstają okresy oczekiwania i czas liczony od chwili wejścia pierwszego detalu na pierwszą obrabiarkę do chwili zejścia piątego detalu z trzeciej obrabiarki jest krótszy lub dłuższy zależnie od kolejności w jakiej ciąg detali przechodzi przez linię obrabiarek. Zadanie polega na tym, by zbadać tę linię pod względem czasu potrzebnego na obróbkę detali w różnych kolejnościach i znaleźć optymalną zarówno dla kompletu jak na rysunku i kompletów z powtórzeniami, /np. a_1 a_2 a_2 a_5 a_4 a_3/, jak również kompletów tworzonych z n różnych detali na linii z dowolnej liczby m obrabiarek.

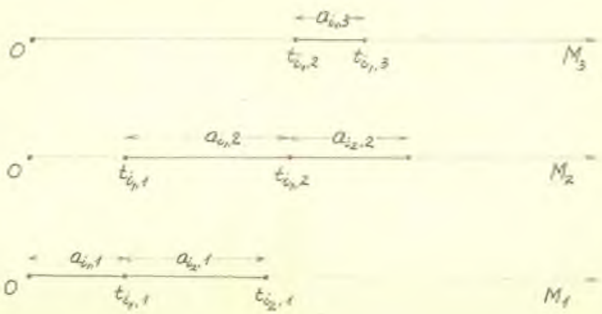
5. 2. R a c h u n e k

Oznaczmy i-ty detal przez a_i, j-tą obrabiarkę przez M_j, czas potrzebny na obrobienie a_i na M_j przez a_{ij}, czas o którym a_{i_k} schodzi z M_j przez $t_{i_k,j}$. W a_i wliczony jest czas potrzebny na przebycie drogi od jednej obrabiarki do drugiej.

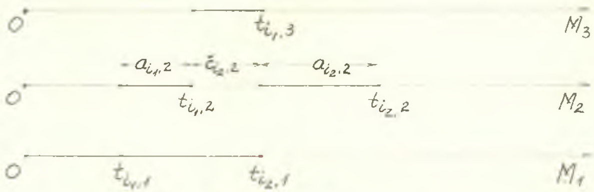
Dla detalu a_{i_1} i poszczególnych obrabiarek mamy

$$t_{i_1,1} = a_{i_1,1} ,$$
$$t_{i_1,2} = t_{i_1,1} + a_{i_1,2} ,$$
$$t_{i_1,3} = t_{i_1,2} + a_{i_1,3} .$$

Detal a_{i_2} schodzi z M_1 o czasie $t_{i_2,1} = t_{i_1,1} + a_{i_2,1}$. M_2 kończy detal a_{i_1} w chwili $t_{i_1,2}$. Jeśli $t_{i_1,2} \geq t_{i_2,1}$, to a_{i_2} odrazu wchodzi na M_2 i schodzi z niej o czasie $t_{i_2,2} = t_{i_1,2} + a_{i_2,2}$.



Natomiast gdy $t_{i_1,2} < t_{i_2,1}$, wówczas obrabiarka M_2 musi czekać na detal a_{i_2} przez czas $\check{o}_{i_2,2} = t_{i_2,1} - t_{i_1,2}$.



W tym przypadku $t_{i_2,2} = t_{i_1,2} + a_{i_2,2} + \check{o}_{i_2,2} = t_{i_1,2} + a_{i_2,2} + t_{i_2,1} - t_{i_1,2} = t_{i_2,1} + a_{i_2,2}$.

Mamy więc

$$t_{i_2,2} = t_{i_1,2} + a_{i_2,2} \quad \text{jeśli } t_{i_1,2} \geqq t_{i_2,1} ,$$

zaś w przeciwnym razie

$$t_{i_2,2} = t_{i_2,1} + a_{i_2,2} .$$

Podobnie otrzymujemy

$$t_{i_2,3} = t_{i_1,3} + a_{i_2,3} \quad \text{jeśli } t_{i_1,3} \geqq t_{i_2,2} ,$$

w przeciwnym razie

$$t_{i_2,3} = t_{i_2,2} + a_{i_2,3} ,$$

. . . .

$$t_{i_k,m} = t_{i_{k-1},m} + a_{i_k,m} \quad \text{jeśli } t_{i_{k-1},m} \geqq t_{i_k,m-1} ,$$

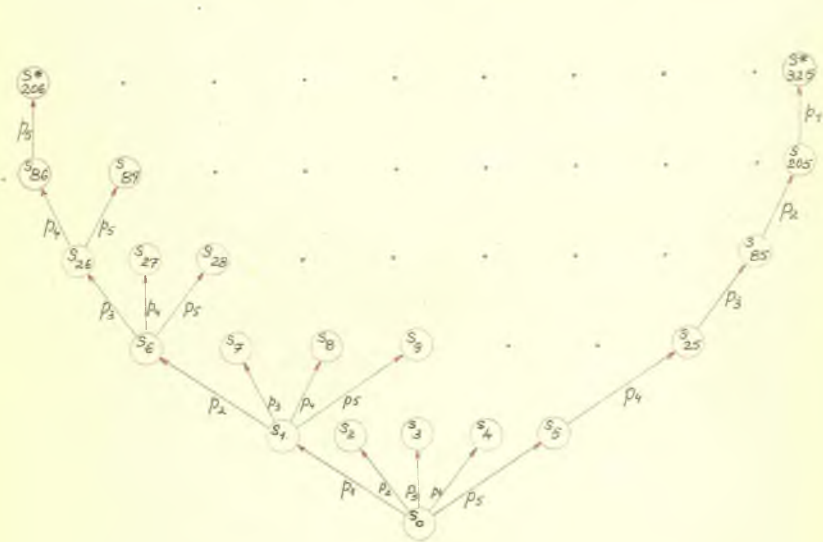
w przeciwnym razie

$$t_{i_k,m} = t_{i_k,m-1} + a_{i_k,m} .$$

5. 3. Gra kooperacyjna

Trzy obrabiarki można uważać za trzech kooperantów. Mamy wówczas do czynienia z 3-osobową wielochodową grą kooperacyjną, w której pierwszy z uczestników ma do wyboru w swym j-tym chodzie 6-j kroków elementarnych, a pozostali uczestnicy dostosowują się do wyboru pierwszego. Stan, który osiąga gra określony jest więc działaniem kooperanta M_i. Każdemu ze stanów przyporządkowujemy trójwymiarowy wektor wypłat - będą to czasy, o których a_i schodzi z M_j.

Otrzymujemy grę G_3, której graf dla 5 detali i dowolnej liczby obrabiarek ~~uczestników~~ można sobie wyobrazić w ten oto sposób:



Mamy tu 120 stanów końcowych: $S^*_{206}, \dots , S^*_{325}$ i 120 możliwych przebiegów gry. Istotnie, na tyle różnych sposobów można ustawić na linii obrabiarek 5 różnych detali, $5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 5! = 120$. Krok elementarny p_i oznacza obrobienie i-tego detalu.

5. 4. D w a   r o d z a j e   g i e r

Dla gry kooperacyjnej G_3 nie istnieje automat uproszczony A_3' , który miałby ilość stanów mniejszą niż G_3 i obrazował istotę procesu kooperacyjnego również dobrze jak G_3. Przyczyna leży w tym, że G_3 jest grą innego rodzaju, aniżeli gra G_1 typu Nim lub gra G_2 obrazująca ładowanie okrętu.

O grze G_1 można powiedzieć, że jest p r z e m i e n n a, stan wytworzony ciągiem kroków elementarnych p_2, p_2, p_1 faktycznie niczym się nie różni od stanu wytworzonego ciągiem p_2, p_1, p_2 . W obu przypadkach w trzech rogach stołu będzie ta sama sytuacja: 2, 1 i 0 zapałek. Dlatego proces obrazowany grą G_1 daje się przedstawić przy pomocy uproszczonego automatu A_1', w którym słowo (x_2, x_2, x_1) generuje ten sam stan co słowo (x_2, x_1, x_2). Przemienna jest również gra G_2, bo waga i wartość ładunku wziętego na statek są jednakowe w przypadku (a, b, c) i w przypadku (b, c, a) .

Natomiast gra kooperacyjna G_3 nie jest przemienna. Ciąg kroków elementarnych (p_1, p_2, p_3) oznaczający obróbkę różnych detali w kolejności najpierw a_1, potem a_2, a końcu a_3 - wymaga z reguły innego czasu produkcyjnego niż ciąg (p_3, p_1, p_2), a więc oba te ciągi wytwarzają dwie różne sytuacje, dwa różne stany procesu kooperacji. Automat A_3 służący do studiowania tego procesu musi spełniać warunek /#/ ,

$$f_q(q_i, x_j) \neq f_q(q_k, x_l) \quad \text{dla } i \neq k,$$

nie może zatem przybrać uproszczonej postaci A_3' analogicznej do A_1' czy A_2' .

Jeśli chcemy, by A_3 obrazował dowolny proces kooperacyjny typu G_3 : zespół m obrabiarek przebiega ciąg p detali, wśród których jest n różnych, przyjmiemy dla A_3 alfabet wejściowy $X = \{x_1, x_2, \ldots, x_n\}$, funkcję f_q określimy formułą

$$f_q\left(q_\alpha, x_{j_k}\right) = q_\beta, \quad \beta = 10\alpha + j_k.$$

Alfabet wyjściowy **Y** i funkcję $f_y : Q \longrightarrow Y$ skonstruujemy przyporządkowując każdemu $q_i \in Q$ czas potrzebny do jego wytworzenia na linii. Wartość f_y obliczamy korzystając z rekurencyjnych wzorów 5. 2.

Dajmy np. na wejście automatu A_3 w stanie q_0 słowo $x = \left(x_{j_1}, x_{j_2}, \ldots, x_{j_k}\right)$ w alfabecie $\{x_1, x_2, \ldots, x_5\}$ np. $x = \left(x_4, x_3, x_2, x_2, x_3, x_5, x_1, x_1\right)$. A_3 osiągnie stan końcowy $q_{43223511}$ i da na wyjściu czas $f_y\left(q_{43223511}\right)$ potrzebny na obróbkę ciągu detali a_4, a_3, a_2, a_2, a_3, a_5, a_1, a_1. Załącznik 2. zawiera odpowiedni program. Można również zlecić maszynie generowanie wszystkich ~~stanów~~ słów pewnego typu, np. wszystkich permutacji ciągu x_1, x_2, x_4, x_5 i wybranie tej z nich, która najszybciej przechodzi przez linię /załącznik 3/. Bardzo proste rozwiązanie tego ostatniego zadania dla dowolnej liczby detali ale tylko 2 obrabiarek znajdujemy w [2].

5. 5. Analiza sieci

Przypuśćmy, że technologia procesu nie dopuszcza żadnych zmian kolejności, w jakiej pięć detali a_i wchodzi na linię trzech obrabiarek M_j. Przy pewnych nakładach jest natomiast możliwe ulepszenie techniki obróbki, zwiększenie tempa pracy i zmniejszenie niektórych czasów a_{ij} obróbki i-tych detali na j-tych obrabiarkach. Na które a_{ij} skierować należy wysiłki? Z rysunku 5. 2. na str. 25 widać np., że nawet najdalej idące skrócenie czasu $a_{i_1, 2}$ przy niezmienionym $a_{i_2, 1}$ nie przyniesie żadnego skrócenia cyklu, bo wszystko jedno następny po a_{i_1} detal a_{i_2} wejdzie na M_2 nie wcześniej niż w chwili $t_{i_2, 1}$, to jest w chwili, gdy zejdzie z M_1.

Niech kolejnością detali będzie a_1, a_2, a_3, a_4, a_5 i niech czasy a_{ij} określa tablica

$a[1 : 5, 1 : 3]$:

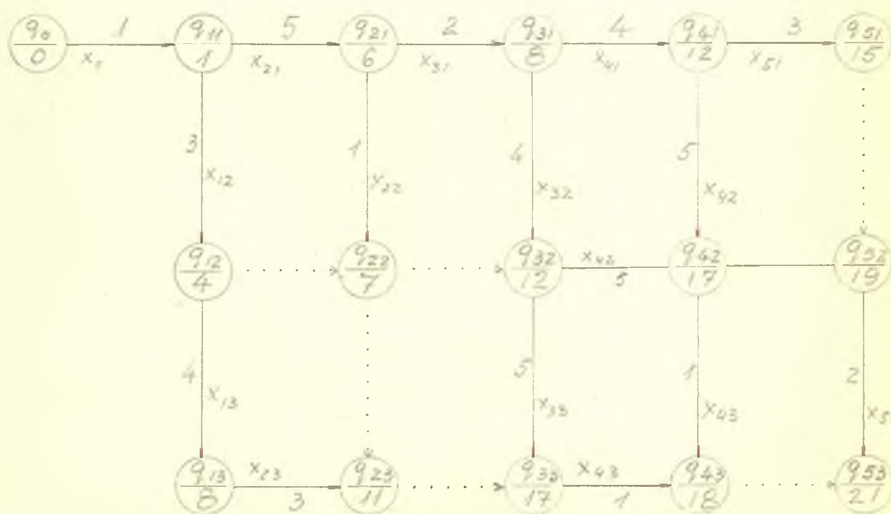
	M_1	M_2	M_3
a_1	1	3	4
a_2	5	1	3
a_3	2	4	5
a_4	4	5	1
a_5	3	2	2

Aby znaleźć wąskie gardła procesu, skonstruujemy automat A_4 inaczej niż poprzednie automaty. Jego stan początkowy q_0 to sytuacja na linii przed rozpoczęciem procesu obróbki. Stan q_{ij} oznacza, że na M_j zakończona została obróbka i-tego detalu; q_{ij} będziemy więc pojmowali jako element istniejący tylko w chwili swego powstania. Wejście x_{ij} oznacza działanie j-tej obrabiarki względem i-tego detalu. Każdemu działaniu x_{ij} przyporządkowujemy czas a_{ij} jego trwania. Działanie x_{ij} może być rozpoczęte nie wcześniej niż w chwili, gdy zostały zakończone działania $x_{i-1,j}$ oraz $x_{i,j-1}$. W rezultacie otrzymamy A_4 jako ó s e m k ę ,

$$A_4 = \langle Q, X, Y, T, f_q, f_y, f_t, q_0 \rangle ,$$
$$Q = \{q_0, q_{ij}\} , \quad X = \{x_{ij}\}, \quad i = 1,2,3;$$
$$Y = \{y_{ij}\} , \quad T = \{a_{ij}\}, \quad j = 1,2,\ldots,5 ;$$

funkcje

$f_q : Q \times X \to Q$, $f_y : Q \to Y$, $f_t : X \to T$ określimy następującym grafem. Przedstawia on te elementy procesu, które dają się połączyć w nurty ciągłe. Widać, że np. stanu q_{52} nie można osiągnąć ze stanu q_{51} w sposób ciągły, bo obróbka x_{52} piątego detalu na drugiej obrabiarce, dająca w rezultacie stan q_{52}, nie może się zacząć w chwili $f_y(q_{51}) = y_{51} = 15$, ponieważ o tym czasie nie została jeszcze zakończona obróbka x_{42} czwartego detalu na M_2, a więc M_2 nie przyjmie piątego detalu. Natomiast w chwili $f_y(q_{42}) = y_{42} = 17$, gdy M_2 się zwalnia, stan q_{51} już nie istnieje. Dlatego q_{52} może być osiągnięty tylko z q_{42} . Rozumując w ten sposób budujemy taką oto siatkę.



Z przeprowadzonej analizy wynika, że stan q_{53} może być generowany dwoma słowami,

$$x = \left(x_{11}, x_{21}, x_{31}, x_{41}, x_{42}, x_{52}, x_{53} \right) ,$$

$$x' = \left(x_{11}, x_{21}, x_{31}, x_{32}, x_{42}, x_{52}, x_{53} \right) ,$$

które różnią się symbolami x_{41}, x_{32}. Ponieważ $a_{41} = a_{32}$, w obu przypadkach otrzymujemy ten sam czas procesu

$$a_{11} + a_{21} + a_{31} + a_{41} + a_{42} + a_{52} + a_{53} = 21 = y_{53}$$

Zadanie zostało rozwiązane: proces obróbki może ulec skróceniu, jeśli zredukujemy czasy a_{11}, a_{21}, a_{31}, a_{41} na M_1, a_{32}, a_{42}, a_{52} na M_2 i a_{53} na M_3. Jeśli natomiast te czasy ulegną zwiększeniu, to proces się przedłuży. Pozostałych 7 czasów, a_{51} na M_1, a_{12}, a_{22} na M_2 i a_{13}, a_{23}, a_{33}, a_{43} na M_3 nie bierze udziału w tworzeniu czasu y_{53} procesu.

Analiza wąskich gardeł, którą przeprowadziliśmy tutaj, nosi nazwę analizy sieciowej /metoda PERT/. Słowa x i x′ to ścieżki /ślady/ krytyczne sieci.

Model $\bar{\bar{A}}_4^p$, przy pomocy którego zmechanizujemy rachunek, zaprogramujemy w ten sposób, że $\bar{\bar{A}}_4^p$ zapisze x_{11} jako pierwszy symbol na ścieżce krytycznej. Jeśli $a_{21} > a_{12}$, to następnym symbolem ścieżki krytycznej $x = \left(x_{11}, x_{j_2}, \ldots, x_{j_s} \right)$ będzie $x_{j_2} = x_{21}$. Jeśli $a_{21} = a_{12}$ to przyjęte zostaną oba symbole, x_{21} i x_{12} i oprócz słowa $x = \left(x_{11}, x_{21}, \ldots, x_{j_s} \right)$, drugą ścieżką krytyczną jest $x' = \left(x_{11}, x_{12}, \ldots , x_{j_s} \right)$. Jeśli $a_{21} < a_{12}$ to drugim elementem słowa x będzie $x_{j_2} = x_{12}$ itd.

Można modelowi $\bar{\bar{A}}_4^p$ powierzyć również odszukanie ścieżki p o d k r y t y c z n e j , to jest tej drogi na grafie automatu A_4, która jest krótsza od ścieżki krytycznej i dłuższa od pozostałych dróg od q_0 do q_{53}, biorąc przy tym pod uwagę również te przejścia, które na linii obrabiarek przy zadanych a_{ij} nie są ciągłe.

6. W N I O S K I

6. 1. Na wstępie było postawione pytanie, czy podstawowych pojęć teorii automatów nie można zastosować do analizy pewnych procesów techniczno-ekonomicznych. Przedstawiony materiał daje odpowiedź pozytywną. Pojęcie automatu jest szczególnie przydatne, gdy problem jest rozwiązywany metodami programowania dynamicznego. [2]

6. 2. Kiedy modelujemy nieantagonistyczne zagadnienia produkcyjne, transportowe lub organizacyjne przy pomocy automatów, nie ma potrzeby korzystać z pojęć tegorii gier. Natomiast celowe może być wówczas zastosowanie teorii grafów. [3]

6. 3. Zaproponowana w 2. 1 definicja gry jako automatu wymaga dyskusji. Należało by w oparciu o tę definicję dać określenie strategii w kontrowersyjnej grze wielochodowej. Wydaje się,że w modelowaniu złożonych procesów konfliktowych należało by połączyć środki, jakimi rozporządza teoria gier, teoria grafów i teoria automatów.

6. 4. Można przypuszczać, że efektywne byłoby zastosowanie automatów s t o c h a s t y c z n y c h do modelowania problemów związanych z teorią m a s o w e j o b s ł u g i /teoria kolejek/.

Z a ł a c z n i k   1.

```
begin
  integer  j, m, n, p, r, t, v, w, z;
  input  (m, n, r, v, w) ;
  begin
    integer array  a [1 ; m, 1 : w]  , s [1 : v] , y [1 : n] ;
    input  (a, s, y)  ;
    p : = r ;
    for  j : = 1  step  1  until  v  do
    begin
      t : = s [j] ;  p : = a [p, t]  ;  if  p < m
      then
      begin
        output (p) ;  if  p = 0  then go to K N
      end
      else
      begin
        z : = y [p - m] ;  output (p, z) ;
      end
    end;
    K N :
  end
end;
```

Program ten służy do modelowania automatu A, dla którego

$Q = \{q_1, q_2, \dots, q_m, q_{m+1}, \dots, q_{m+n}\}$,

$X = \{x_1, \dots, x_w\}$, $Y = \{y_1, \dots, y_u\}$,

$f_q : Q \times X \longrightarrow Q$, $f_y : Q \longrightarrow Y$ dane są tablicami $a[1 ; m, 1 : w]$, $y[1 : n]$, gdzie zamiast elementów q_i, x_j, y_ℓ występują ich numery i, j, ℓ . Funkcja f_y jest określona na podzbiorze Q^* ; dla tych par (q_i, x_j) , dla których f_q nie jest określona, przyjmujemy $a[i, j] = 0$.

Na wejście automatu A dajemy słowo $x = (x_{j_1}, x_{j_2}, \dots, x_{j_v})$ w postaci tablicy $s[1 : v]$, przy czym symbol x_{j_1} zostaje odczytany w stanie q_r.

Wykonując program, maszyna drukuje ciąg $\{p_i\}$ złożony z k liczb, k = v, lub k = v + 1, - jeśli A przyjmuje słowo x ; w przeciwnym razie $1 \leq k < v$. Jeśli, w toku odczytywania słowa x, A nie osiągnie żadnego ze stanów końcowych, to $p_i \leq m$, w przeciwnym razie przedostatnia liczba będzie numerem stanu końcowego q_i, a ostatnia - wartością $f_y(q_i)$. Ciąg $\{p_i\}$ kończy się zerem, jeśli A nie przyjmuje x i w w toku odczytywania nie osiąga żadnego ze stanów końcowych.

Z a ł ą c z n i k 2.

```
begin
  integer i, j, k, m, n, p, r ;
  real TEMP;
  input (m, n, p) ;
  begin
    integer array s [1 : p] ;
    array a [1 : m, 1 : n] , t [1 : m, 1 : p] ;
    input (s, a) ;
    r : = s [1] ; t [1, 1] : = a [1, r] ;
    for i : = 2 step 1 until m do
    t [i, 1] : = t [i - 1, 1] + a [i, r] ;
    for j : 2 step 1 until n do
    begin
      r : = s [j] ; t [1, j] : = t [1, j - 1] + a [1, r] ;
      for k : = 2 step 1 until m do
      t [k, j] : = if t [k, j - 1] ≤ t [k - 1, j]
      then t [k - 1, j] + a [k, r] else
      t [k, j - 1] + a [k, r]
    end;
    TEMP : = t [ m,p ] ; output (TEMP)
  end
end;
```

Liczba TEMP jest czasem, o którym detal a_{i_p} z zadanego ciągu detali $a_{i_1}, a_{i_2}, \ldots, a_{i_p}$ schodzi z ostatniej m-tej obrabiarki; w tym ciągu p detali jest n detali różnych. Czas potrzebny na obróbkę i-tego detalu na j-tej obrabiarce zadany jest tablicą a [1 : m, 1 : n] .

Z a ł ą c z n i k   3 .

```
begin
integer array  S [1:5] , P [1:5, 1:5] ;
array  a [1:3, 1:5] ,  t [1:3, 1:5] ;
integer  b, c, d, e, f, h, i, j ;
real  sub, optymal, z ;
real procedure  TEMP (a, s, t) ;
integer array s ;
array  a, t;
  begin
  integer i, j, k, r;
    r := s [1]; t [1,1] := a [1, r] ;
    for i := 2  step  1 until  3 do
    t [i, 1] := t [i-1, 1]  + a [i, r ]  ;
    for  j := 2 step 1 until 5 do
    begin
      r : = s [j] ;
      t [1,r ] := t [1, j-1 ]  + a [1, r] ;
      for  k := 2 step  1 until  3 do
      t [k,j] := if  t  k,j-1     t k-1,j   then
      t [k-1,j]  +a [k,r]  else
      t [k,j-1] +a [k,r]
    end;
    TEMP := t [3,5]
  end  procedury;
  input (a) ;
  h:= 1;
  z:= 1000;
  for  b:= 1, 2, 3, 4, 5  do
  for  c:= 1, 2, 3, 4, 5  do
  for  d:= 1, 2, 3, 4, 5  do
  for  e:= 1, 2, 3, 4, 5  do
  for  f:= 1, 2, 3, 4, 5  do
  begin
    if  b ≠ c ∧ b ≠ d ∧ b ≠ e ∧ b ≠ f ∧ c ≠ d ∧ c ≠ e ∧ c ≠ f ∧
        d ≠ e ∧ d ≠ f ∧ e ≠ f  then
    begin
      S [1] := b;  S [2] := c;
      S [3] := d;  S [4] := e;
      S [5] := f;
      sub:= TEMP (a,S,t) ;
      if  sub < z  then  h := 1;
      if  sub =  z  then  h:= h+1;
      if  sub < z  then
      begin
        optymal := sub;
        if  h < 5  then
        for  j:=  1 step 1 until 5 do
        P [h,j] := S [j] ;
        z := sub
      end
    end;
  end;
```

```
  for  i:= 1 step  1 until  if h ≤ 5  then h  else 5 do
  begin
    outcr;
    for j:= 1 step  1 until  5 do
    output (≮d≯  , P [i,j])
  end;
  outcr;
  output (≮dd≯  ,  optymal, h)
end  programu;

 2, 3, 5, 7, 9,
10, 4,10, 5, 2,
 3, 9, 2, 4, 7.
```

Program ten pracował na maszynie GIER ZON Uniwersytetu Warszawskiego. Maszyna podała dwie optymalne kolejności: 12453 i 12534 z takim samym czasem 38.

CYTOWANE KSIĄŻKI

[1] Claude Berge. Théorie des graphes et ses appli~~k~~cations. Paris 1958.

[2] R. Bellman, S. Dreyfus. Prikładnyje zadaczi dinamiczeskogo programmirowanija. Moskwa 1965.

[3] G. Avondo-Bodino. Economic applications of the theory of graphs. New York 1962.