ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1982

Seria: AUTOMATYKA z. 64 Nr kol. 736

Franciszek MARECKI

Instytut Automatyki
Politechniki Śląskiej

HARMONOGRAMOWANIE ZAŁADUNKU KONTENERÓW W PRZEMYSŁOWYM MAGAZYNIE WYSOKIEGO SKŁADOWANIA

Streszczenie. W pracy przedstawiono model matematyczny procesu załadunku kontenerów w przemysłowym magazynie wysokiego składowania. Dla optymalnego harmonogramowania procesu przyjęto kryterium minimalizacji czasu realizacji wszystkich zadań. Sformułowany problem rozwiązano algorytmem programowania wieloetapowego.

WSTĘP

Procesy załadunku i wyładunku kontenerów w przemysłowych magazynach wysokiego składowania (PMWS) stanowią kompleks operacji z ograniczeniami w czasie i przestrzeni. Dla potrzeb sterowania należy przeprowadzić identyfikację PMWS i sprecyzować model matematyczny analizowanego procesu [1]. Jednym z elementów sterowania PMWS jest harmonogramowanie [2] (w pracy [2] przedstawiono przegląd problematyki sterowania PMWS na podstawie obszernego przeglądu literatury).

W niniejszym referacie zostanie rozpatrzona faza załadunku PMWS. Wyróżnimy transport kontenerów do regałów oraz transport międzyregałowy. A zatem dla załadunku każdy kontener "przepływa" przez dwa agregaty (środki transportowe), przy tym windy międzyregałowe pracują równolegle, a wózek dostawczy w szereg z nimi. Z uwagi na marszruty zadań (załadowywanych kontenerów) proces załadunku zachodzi w systemie o strukturze drzewa. Problem harmonogramowania polega na przypisaniu dla każdego zadania agregatu i przedziału czasu realizacji zadania na tym agregacie [3]. Z punktu widzenia złożoności obliczeniowej jest to problem NP - zupełny [4] dla systemu równoległego, a tym samym dla systemu o strukturze drzewa.

W referacie dla optymalnego rozwiązania tego problemu wykorzystano algorytm programowania wieloetapowego.

1. ZAŁOŻENIA - SFORMUŁOWANIE PROBLEMU

W magazynach wysokiego składowania proces załadunku (i rozładunku) kontenerów odbywa się automatycznie. Każdy kontener jest dostarczany przez

wózek dostawczy A do strefy wybranych m-tych regałów. Stąd za pomocą wózka z windą kontener jest lokowany w wybranym miejscu regału. Wózki pracują automatycznie z ustalonymi prędkościami.

Załóżmy, że dany jest zbiór zadań:

$$\Omega = \{\omega_n\}_{(n=\overline{1,N})} \tag{1}$$

gdzie:

ω_n - n-te zadanie,

N - liczba zadań.

Zadanie interpretujemy jako polecenie ulokowania określonego kontenera na wybranym regale.

Załóżmy, że dany jest zbiór agregatów:

$$\mathcal{A} = \{A_m\}_{(m=\overline{0,M})} \tag{2}$$

gdzie:

A_m - m-ty agregat,

M+1 - liczba agregatów.

Agregatami są wózki transportujące kontenery.

Załóżymy, że każde zadanie przechodzi przez agregat A_0 oraz jeden z agregatów A_m, $(m=\overline{1,M})$. Przed rozpoczęciem realizacji zadań wszystkie agregaty znajdują się w swoich punktach startowych. Można założyć, że dane są momenty dostępności agregatów, zapisane w wektorze:

$$R = [r_m] \tag{3}$$

gdzie:

r_m - moment dostępności agregatu A_m.

Założymy, że na punktach startowych agregatów A_m, $(m=\overline{1,M})$ może znajdować się co najwyżej jedno zadanie. A zatem w punkcie tym może znajdować się jeden kontener dostarczony przez A_0, w momencie gdy A_m lokuje w regale inny kontener.

Dane są czasy transportu kontenerów przez wózek A_0 do punktów przeładunkowych przy regałach. Czasy te zapiszemy wektorem:

$$T = [\tau_m]_{(m=\overline{1,M})} \tag{4}$$

gdzie:

τ_m - czas transportu kontenera do m-tego punktu przeładunkowego.

Przyporządkowanie zadań do agregatów (regałów) podaje macierz:

$$U = \left[u_{m,n}\right]_{\substack{(m=\overline{1,M})\\(n=\overline{1,N})}} \tag{5}$$

gdzie:

$$u_{m,n} = \begin{cases} 1 : \text{jeśli zadanie } \omega_n \text{ może być wykonane przez agregat } A_m. \\ 0 : \text{w przeciwnym przypadku} \end{cases} \tag{6}$$

Jeżeli zadanie ω_n jest wykonane przez agregat A_m, tzn., że odpowiedni kontener jest ulokowany w m-tym regale.

Niechaj czasy wykonywania zadań przez agregaty A_m, $(m=\overline{1,M})$ będą dane macierzą:

$$\Theta = \left[\vartheta_{m,n}\right] \tag{7}$$

gdzie:

$\vartheta_{m,n}$ - czas realizacji zadania ω_n przez agregat A_m.

Ponadto założymy, że czasy powrotów wózków A_m do punktów startowych wynoszą $\vartheta_{m,n}$ lub odpowiednio τ_m. Zauważmy, że czas realizacji zadania ω_n na agregacie A_o jest zależny od tego, na którym agregacie A_m zadanie to jest wykonywane w następnej kolejności.

Załóżmy, że dane są terminy dostępności zadań w systemie, zapisane wektorem:

$$\phi = \left[\varphi_n\right] \tag{8}$$

gdzie:

φ_n - termin dostępności zadania ω_n.

Dostępność zadania w systemie oznacza możliwość realizacji tego zadania przez agregat A_o. Założymy, że agregat A_o nie może mieć przestojów celowych. Oznacza to, że jeżeli przed magazynem oczekują zadania, to agregat A_o nie może oczekiwać na zadanie, które pojawi się przed magazynem później. Natomiast dopuszczamy wymuszony przestój agregatu A_o, związany z brakiem zadań przed magazynem lub z zajętością punktów przeładunkowych przed regałami. Zagadnienie dostępności zadań będzie szczegółowo analizowane w dalszej części pracy.

Dla optymalizacji harmonogramu realizacji zadań przyjmiemy kryterium minimalizacji czasu wykonania wszystkich zadań. A zatem funkcja celu ma postać:

$$Q = \max_{0 \leq m \leq M} \max_{1 \leq n \leq N} t_{m,n} \rightarrow \min \qquad (9)$$

gdzie:

$t_{m,n}$ - moment zakończenia wykonywania zadania ω_n na agregacie A_m.

Rezultatem obliczeń winny być optymalne harmonogramy pracy wszystkich agregatów w postaci:

$$H_o = \ll \varrho_{o,1}; t_{o,1} \gg, \ldots, \ll \varrho_{o,n}; t_{o,n} \gg, \ldots, \ll \varrho_{o,N}; t_{o,N} \gg \qquad (10)$$

gdzie:

$\varrho_{o,n}$ - moment rozpoczęcia wykonywania zadania ω_n na agregacie A_o.

Dla agregatów A_m, $(m=\overline{1,M})$ w harmonogramie analogicznym do (10) wystąpią tylko niektóre zadania.

2. ALGORYTM

Przed podaniem formuł matematycznych algorytmu przedyskutujemy pewne założenia o organizacji pracy wózka dostawczego oraz wózków z windą. Wózek dostawczy A_o może oczekiwać tylko w swoim punkcie startowym. Podobne założenie przyjmiemy dla każdego wózka z windą A_m, $(m=\overline{1,M})$. Ponadto przyjmiemy, że na punkcie przeładunkowym z A_o na A_m może znajdować się tylko jeden kontener. A zatem, jeżeli wózek A_o ma przekazać kontener wózkowi A_m, to po przybyciu do punktu przeładunkowego musi mieć możliwość rozładunku. Wózek A_m w tym czasie może znajdować się również na punkcie przeładunkowym lub realizować swoje poprzednie zadanie. Prowadzi to do wniosku, że wózek A_o rozpoczyna realizację zadania w takiej chwili, by było ono wykonane bez przestojów wózka A_o.

Algorytm rozwiązania sformułowanego problemu będzie oparty na programowaniu wieloetapowym. Zasadniczymi elementami tego algorytmu są: stan procesu decyzyjnego, wartość stanu procedury generowania stanów oraz procedury eliminowania stanów (które nie prowadzą do rozwiązania optymalnego).

2.1. Stan procesu decyzyjnego i wartość stanu

Przydzielenie zadań do realizacji będziemy nazywali decyzjami. Ciąg decyzji nazwiemy strategią. Po podjęciu każdej decyzji stan procesu decyzyjnego będzie ulegał zmianie.

Definicja 1: Stanem procesu decyzyjnego jest macierz o postaci:

$$P^{\lambda,\varrho} = \left[p^{\lambda,\varrho}_{n,j}\right]_{\substack{(n=\overline{1,N})\\(j=\overline{1,3})\\(\varrho=\overline{0,N})\\(\lambda=\overline{1,L_\varrho})}} \tag{11}$$

gdzie:

ϱ - numer etapu decyzyjnego,

λ - numer stanu w ramach etapu,

L_ϱ - liczba stanów ϱ -tego etapu decyzyjnego.

Elementy macierzy (11) mają następujące znaczenia:

$$p^{\lambda,\varrho}_{n,1} = \begin{cases} t_{o,n} & \text{- jeśli zadanie } \omega_n \text{ zostało wykonane przez } A_o; \\ 0 & \text{- w przeciwnym przypadku} \end{cases} \tag{12}$$

$$p^{\lambda,\varrho}_{n,2} = \begin{cases} m & \text{- jeśli zadanie } \omega_n \text{ zostało wykonane przez } A_m, \ (m=\overline{1,M}); \\ 0 & \text{- w przeciwnym przypadku} \end{cases} \tag{13}$$

$$p^{\lambda,\varrho}_{n,3} = \begin{cases} t_{m,n} & \text{- jeżeli } p^{\lambda,\varrho}_{n,2} = m; \\ 0 & \text{- w przeciwnym przypadku} \end{cases} \tag{14}$$

A zatem stan początkowy $P^{1,0}$ jest macierzą zerową, natomiast stany końcowe $P^{\lambda,N}$ mają wszystkie współrzędne dodatnie. Liczba etapów decyzyjnych wynosi N, bowiem zadanie ω_n będzie jednocześnie przydzielone na A_o oraz wybrany A_m.

Z każdym stanem $P^{\lambda,\varrho}$ związemy jego wartość, którą oznaczymy przez $v^{\lambda,\varrho}$.

Definicja 2: Wartość stanu jest liczbą, którą wyznaczymy z formuły:

$$v^{\lambda,\varrho} = \max\left(\max_{1\leqslant n\leqslant n} p^{\lambda,\varrho}_{n,1}; \max_{1\leqslant n\leqslant N} p^{\lambda,\varrho}_{n,3}\right) \tag{15}$$

W powyższych formułach przyjmujemy, że wykonanie zadania kończy się w momencie, gdy wózek wraca do swego punktu startowego. W ten sposób (15) wyraża moment zakończenia pracy przez wszystkie wózki po wykonaniu zadań należących do odpowiedniego stanu. A zatem wartości stanów $P^{\lambda,N}$ oznaczają czasy wykonania wszystkich zadań.

Zgodnie z kryterium (9) optymalny stan wyznaczamy z warunku:

$$\left(\min_{1 \leq \lambda \leq L_N} v^{\lambda,N} = v^{\lambda^o,N} \right) \Rightarrow \left(p^{\lambda^o,N} = p^o \right) \tag{16}$$

gdzie:

p^o - optymalny stan końcowy.

Z optymalnego stanu końcowego wyznaczamy wprost optymalny harmonogram realizacji zadań:

- dla A_o

$$\left.\begin{aligned} \varrho_{o,n} &= p^o_{n,1} - 2\tau_m \\ t_{o,n} &= p^o_{n,1} \end{aligned}\right\} \tag{17}$$

przy tym:

$$m = p^o_{n,2} \tag{18}$$

- dla A_m, $(m=\overline{1,M})$:

$$\left.\begin{aligned} \varrho_{m,n} &= p^o_{n,3} - 2\vartheta_{m,n} \\ t_{m,n} &= p^o_{n,3} \end{aligned}\right\} \tag{19}$$

pod warunkiem, że prawdziwa jest równość (18).

Reasumując, można powiedzieć, że celem wyznaczenia optymalnego rozwiązania problemu należy wygenerować wszystkie stany od $P^{1,O}$ do $P^{\lambda_N,N}$.

2.2. Generowanie stanów

W generowaniu stanów będziemy wykorzystywali założenie, że wózek A_m, $(m=\overline{0,M})$ nie może mieć przestojów celowych, a jedynie przestoje wymuszone. A zatem wózek A_o nie będzie oczekiwał na nadejście zadań późniejszych, jeżeli nie zrealizował innych dostępnych zadań. Eliminacja przestojów celowych dla wózków A_m, $(m=\overline{1,M})$ wynika z możliwości przechowania na punkcie przeładunkowym tylko jednego zadania. Tak więc wózek ten nie może realizować najpierw zadania, które dotarło na punkt przeładunkowy później niż inne zadanie. Wózek A_m ma przestój wymuszony, jeżeli przybędzie do swojego punktu startowego wcześniej niż kolejne zadanie, które ma zrealizować.

Procedura generowania stanów polega na przydzieleniu zadania dopuszczalnego ω_n dla A_o oraz odpowiedniego A_m. Zadanie ω_n jest dopuszczalne dla A_o, jeżeli nie powoduje przestoju celowego. Warto podkreślić, że zadanie ω_n może być równocześnie przyczielone na A_m. Stąd liczba etapów decyzyjnych wynosi N. Zadanie ω_n może być przydzielone na A_m, jeżeli w konsekwencji nie spowoduje celowego przestoju A_o. Taki przypadek występuje wtedy, gdy A_m jest zajęty realizacją zadania a jego punkt przeładunkowy jest również zajęty przez kolejny kontener. W takiej sytuacji A_o rozpoczyna transport kontenera w momencie zależnym od chwili zwolnienia punktu przeładunkowego. Przestój A_o może być dopuszczalny (jeżeli wcześniej dostępne są inne zadania). Przestój tego rodzaju byłby dopuszczalny jedynie wtedy, gdy wszystkie punkty przeładunkowe są zajęte.

Momenty zwolnienia wózków określamy z formuł:

$$T_o^{l,\varphi-1} = \max(r_o;\ \max_{1\leq i\leq N} p_{i,1}^{l,\varphi-1}) \tag{20}$$

$$T_m^{l,\varphi-1} = \max(r_m;\ \max_{1\leq i\leq N} p_{i,3}^{l,\varphi-1}) \quad (m=\overline{1,M}) \tag{21}$$

Zakładamy, że w stanie początkowym wszystkie punkty przeładunkowe są puste.

Wyróżnimy cztery procedury generowania stanów. Procedura pierwsza dla przypadku, gdy nie ma wymuszonego przestoju A_o; procedura druga, gdy przestój A_o jest wymuszony brakiem zadania; procedura trzecia, gdy przestój A_o jest wymuszony brakiem reguły; procedura czwarta, gdy przestój A_o jest spowodowany brakiem zadania i reguły. Dla wyboru jednej z tych procedur wprowadzimy prognozowane terminy rozpoczęcia pracy przez A_o w danym stanie $P^{l,\varphi-1}$, a zatem:

$$\varphi_o^* = \min_{n\in\alpha^{l,\varphi-1}} \varphi_n \tag{22}$$

gdzie:

φ_o^* - najwcześniejszy termin dostępności zadania, które nie zostało jeszcze zrealizowane,

przy tym:

$$\bigvee_n (p_{n,1}^{l,\varphi-1} = 0) \Rightarrow (n\in\alpha^{l,\varphi-1}) \tag{23}$$

Jeżeli spełniona jest nierówność:

$$\varphi_o^* < T_o^{l,\varphi-1} \tag{24}$$

to A_o nie ma wymuszonego przestoju wynikającego z braku zadania. Z kolei wprowadzamy termin:

$$\varrho_o^{**} = \min_{1 \leqslant m \leqslant M} \min_{n \in \beta_m^{l,\varrho-1}} (T_m^{l,\varrho-1} - 2\vartheta_{m,\vartheta_m} - \tau_m) \tag{25}$$

gdzie:

ϱ_o^{**} - najwcześniejszy termin dostępności regułu dla zadania.

przy tym:

$$\bigwedge_n (p_{n,1}^{l,\varrho-1} = 0) \wedge (\varphi_n \leqslant T_o^{l,\varrho-1}) \wedge (u_{m,n} = 1) \Rightarrow (n \in \beta_m^{l,\varrho-1}) \tag{26}$$

oraz:

$$\bigwedge_{1 \leqslant m \leqslant M} (p_{i,2}^{l,\varrho-1} = m) \wedge (T_m^{l,\varrho-1} = p_{i,3}^{l,\varrho-1}) \Rightarrow (i = \vartheta_m) \tag{27}$$

Jeżeli spełniona jest nierówność:

$$\varrho_o^{**} \leqslant T_o^{l,\varrho-1} \tag{28}$$

to A_o nie ma wymuszonego przestoju z uwagi na brak regału

Termin ϱ_o^{**} jest dostępnością regału w punkcie startowym A_o.

Jeżeli (24) i (28) są spełnione, to stosujemy następującą procedurę generowania stanów:

$$\bigwedge_n \bigwedge_m (p_{n,1}^{l,\varrho-1} = 0) \wedge (\varphi_n \leqslant T_o^{l,\varrho-1}) \wedge (u_{m,n} = 1) \wedge (T_m^{l,\varrho-1} - 2\vartheta_{m,\vartheta_m} - \tau_m \leqslant T_o^{l,\varrho-1})$$

$$\Rightarrow (P^{\lambda,\varrho} = P^{l,\varrho-1} + \Delta P) \tag{29}$$

Elementy macierzy ΔP mają następujące znaczenia:

$$\Delta p_{i,1} = \begin{cases} t_{o,n} & : \text{dla } i = n \\ 0 & : \text{w przeciwnym przypadku} \end{cases} \tag{30}$$

$$\Delta p_{i,2} = \begin{cases} m : \text{dla } i = n; \\ 0 : \text{w przeciwnym przypadku} \end{cases} \tag{31}$$

$$\Delta p_{i,3} = \begin{cases} t_{m,n} : \text{dla } i = n; \\ 0 : \text{w przeciwnym przypadku} \end{cases} \tag{32}$$

Momenty $t_{o,n}$ oraz $t_{m,n}$ wynoszą:

$$t_{o,n} = T_o^{l,\varrho-1} + 2\tilde{\tau}_m \tag{33}$$

$$t_{m,n} = \max(T_o^{l,\varrho-1} + \tilde{\tau}_m; T_m^{l,\varrho-1}) + 2\vartheta_{m,n} \tag{34}$$

A zatem pierwsza procedura generowania stanów została wyjaśniona.

Jeżeli zachodzi warunek:

$$\varrho_o^* \leqslant T_o^{l,\varrho-1} < \varrho_o^{**} \tag{35}$$

to realizujemy tylko jedno zadanie, które można wykonać najwcześniej (wynika to z wykluczenia celowych przestojów A_o).

Procedura generowania stanów ma postać:

$$\exists_n \left[\min_{i \in \varphi^{l,\varrho-1}} (\varphi_i; \varrho_{o,i}^{**}) = \min(\varphi_n, \varrho_{o,n}^{**})\right] \Rightarrow (P^{\lambda,\varrho} = P^{l,\varrho-1} + \Delta P) \tag{36}$$

przy tym:

$$\varrho_{o,i}^{**} = \min_{\mu \in \gamma_i} (T_\mu^{l,\varrho-1} - 2\vartheta_{\mu,\vartheta_\mu} - \tilde{\tau}_\mu) \tag{37}$$

oraz:

$$\forall_\mu (u_{\mu,i} = 1) \Rightarrow (\mu \in \gamma_i) \tag{38}$$

W przypadku gdy:

$$\varphi_n \leqslant \varrho_{o,n}^{**} \tag{39}$$

elementy wektora ΔP wyznaczamy zgodnie z: (30), (31) i (32) następująco:

$$t_{o,n} = \varrho^{**}_{o,n} + 2\tilde{\tau}_m \tag{40}$$

$$t_{m,n} = \max(\varrho^{**}_{o,n} + \tilde{\tau}_m ; T^{1,\mathcal{P}-1}_m) + 2\vartheta_{m,n} \tag{41}$$

przy tym korzystając z (37) otrzymamy agregat A_m:

$$\varrho^{**}_{o,i} = T^{1,\mathcal{P}-1}_m - 2\vartheta_{m,\vartheta_m} + \tilde{\tau}_m \tag{42}$$

W przypadku gdy:

$$\varrho^{**}_{o,n} < \varphi_n \tag{43}$$

Zadanie ω_n może być przydzielone do każdego A_j spełniającego warunek:

$$T^{1,\mathcal{P}-1}_j - 2\vartheta_{j,\vartheta_j} - \tilde{\tau}_j < \varphi_n \tag{44}$$

Zatem można wygenerować stan dla każdego j, obliczając elementy wektora ΔP zgodnie z: (30), (31) i (32) następująco:

$$t_{o,n} = \varphi_n + 2\tilde{\tau}_j \tag{45}$$

$$t_{j,n} = \max(\varphi_n + \tilde{\tau}_j ; T^{1,\mathcal{P}-1}_j) + 2\vartheta_{j,n} \tag{46}$$

A zatem druga procedura generowania stanów została wyjaśniona.

Jeżeli zachodzi warunek:

$$T^{1,\mathcal{P}-1}_o < \varrho^{*}_o \tag{47}$$

to generowanie stanów przebiega tak jak w poprzednim przypadku, tzn.zgodnie z (36). Należy zauważyć, że warunek (47) obejmuje dwa przypadki.Pierwszy, gdy zadanie $\omega_n \in \alpha^{1,\mathcal{P}-1}$ są niedostępne w chwili $T^{1,\mathcal{P}-1}_o$ lecz dostępne są niektóre regały dla niektórych z tych zadań. W drugim przypadku zadania $\omega_n \in \alpha^{1,\mathcal{P}-1}$ nie są dostępne w chwili $T^{1,\mathcal{P}-1}_o$ i również reguły dla tych zadań nie są dostępne. W obydwu przypadkach należy wyznaczyć

zadanie ω_n, które można rozpocząć realizować najwcześniej (ponieważ wykluczone są celowe przestoje A_o).

Tak więc z analizy możliwych przypadków wynika, że stany generowane są na podstawie procedury (29), jeżeli spełniony jest warunek (24) i (28) lub na podstawie procedury (36), jeżeli warunek (24) lub (28) nie jest spełniony.

2.3. Eliminowanie stanów

W trakcie generowania stanów niektóre z nich można wyeliminować, jeżeli nie prowadzą do rozwiązania optymalnego. Jeżeli optymalne rozwiązanie uzyskane ze stanu $P^{\lambda_1,\varphi}$ jest lepsze niż ze stanu $P^{\lambda_2,\varphi}$, to będziemy mówić, że stan $P^{\lambda_1,\varphi}$ dominuje nad stanem $P^{\lambda_2,\varphi}$.

Eliminacja stanów przez dominację jest oparta na następującym twierdzeniu:

Twierdzenie: Stan $P^{\lambda_1,\varphi}$ dominuje nad stanem $P^{\lambda_2,\varphi}$, jeżeli jest spełniony warunek:

$$\forall_n \; \forall_{1\le m\le M} \left[(p_{n,1}^{\lambda_1,\varphi}=0)\Leftrightarrow(p_{n,1}^{\lambda_2,\varphi}=0)\right] \wedge (T_o^{\lambda_1,\varphi} \le T_o^{\lambda_2,\varphi}) \wedge (T_m^{l_1,\varphi-1} \le T_m^{l_2,\varphi-1}) \tag{48}$$

Terminy $T_m^{l,\varphi-1}$ można wyznaczyć odejmując od $T^{\lambda,\varphi}$ czas realizacji ostatniego zadania:

$$T_m^{l,\varphi-1} = T_m^{\lambda,\varphi} - 2\vartheta_{m,\vartheta_m} \tag{49}$$

Dowód tego twierdzenia jest łatwy. Można zauważyć, że optymalny harmonogram od stanu $P^{\lambda_2,\varphi}$ można zrealizować również od stanu $P^{\lambda_1,\varphi}$. Ponadto, ponieważ w stanie $P^{\lambda_1,\varphi}$ agregaty są dostępne wcześniej niż w stanie $P^{\lambda_2,\varphi}$, stąd można przyspieszyć realizację tego harmonogramu od stanu $P^{\lambda_1,\varphi}$. Stąd wnioskujemy, że ze stanu $P^{\lambda_1,\varphi}$ uzyskujemy lepsze rozwiązanie końcowe, aniżeli ze stanu $P^{\lambda_2,\varphi}$. Stany zdominowane w obliczeniach pomijamy.

3. UWAGI KOŃCOWE I WNIOSKI

Przedstawiony w referacie model matematyczny załadunku kontenerów w PMWS interpretuje sytuację, w której fazy załadunku i rozładunku są od-

dzielone od siebie. W praktyce taka sytuacja występuje w przypadku, gdy załadunek ma wyższy priorytet niż rozładunek. Jest to związane z karami umownymi płaconymi za przestój środków transportowych (np. wagonów kolejowych). Jeżeli załadunek PMWS odbywa się równocześnie z różnych środków transportowych, to dla optymalnego harmonogramowania należałoby wprowadzić kryterium minimalizacji sumarycznej kary.

W rozpatrywanej fazie załadunku PMWS założono, że w każdym regale ustalona jest lokalizacja kontenera określonego typu (określonego zadania). Drugi skrajny przypadek, to nieustalona lokalizacja kontenerów w każdym regale. W przypadku tym liczba zadań, jakie mogą być ulokowane w każdym regale, jest ograniczona. Czasy realizacji zadań przez agregaty A_m, ($m = \overline{1,M}$) są zależne od kolejności załadunku ($\vartheta_{k,m}$) a nie od zadania (o ustalonej lokalizacji). Rozwiązanie problemu harmonogramowania przy takim założeniu jest analogiczne do przedstawionego w referacie. Różnice są zawarte w procedurze generowania stanów (trzeba wybierać A_m, w których jest miejsce na kontener) i dominacji stanów (w każdym regale A_m testowanych stanów winno być ulokowanych tyle samo kontenerów).

Problem harmonogramowania rozładunku PMWS może być rozpatrywany jako "przepływ" zadań w systemie o strukturze antydrzewa. W najprostszym przypadku lokalizacja kontenerów (zadań) jest ustalona. Ma to miejsce, gdy każde zadanie (kontener) jest innego typu. W złożonym przypadku w ramach każdego typu może wystąpić kilka kontenerów (o znanej lokalizacji). Zbiór zadań nie obejmuje wszystkich kontenerów, stąd lokalizacja zadań nie jest ustalona. Harmonogramowanie rozładunku PMWS można przeprowadzić analogicznie jak załadunku kontenerów (uwzględniając jedynie dodatkowe ograniczenia).

Proponowany algorytm programowania wieloetapowego jest algorytmem podziału i ograniczeń bez powrotów. Za pomocą tego algorytmu uzyskujemy rozwiązanie optymalne, jeżeli nie zostanie przekroczona dopuszczalna liczba stanów w pewnym etapie. W przeciwnym przypadku otrzymujemy rozwiązanie heurystyczne. Dla ograniczenia czasu obliczeń w algorytmie deklaruje się odpowiednią liczbę stanów, jaką można zapamiętywać na poszczególnych etapach decyzyjnych.

LITERATURA

[1] LORECKI M., PUCHAŁKA T.: Identyfikacja magazynów wysokiego składowania w ujęciu systemowym, ZN Pol. Śl., s. Automatyka, z. 54, Gliwice 1980, ss. 109-117.

[2] NIEDERLIŃSKI A.: Komputerowe systemy sterowania magazynów wysokościowych. Przegląd Problematyki. ZN Pol. Śl., s. Automatyka, z. 45, Gliwice, 1978, ss. 96-104.

[3] NIEDERLIŃSKI A.: Harmonogramowanie produkcji a wielopoziomowe wielowymiarowe dyskretne układy regulacji nadążej. ZN Pol. Śl., s. Automatyka, z. 55, ss. 63-69.

[4] COFFMAN E.G.: Teoria szeregowania zadań. WNT, Warszawa 1980.

Recenzent: Doc. dr hab. inż. Ernest CZOGAŁA

Wpłynęło do Redakcji 15.05.1982 r.

СОСТАВЛЕНИЕ ГРАФИКОВ ПОГРУЗКИ КОНТЕЙНЕРОВ В ПРОМЫШЛЕННОМ ВЫСОКОСТЕЛДАЖНОМ СКЛАДСКОМ МАГАЗИНЕ

Р е з ю м е

В работе представлено математическую модель процесса погрузки в промышленном высокостеллажном складском магазине. Для оптимального расписания графика принято критерий минимализации времени реализации всех заданий. Сформулированную проблему решено методом многомагового програмирования.

SCHEDULING OF CONTAINER LOADING IN AN INDUSTRIAL HIGH STARAGE WAREHOUSE

S u m m a r y

We present a mathematical model of container loading in an industrial high storage warehouse. For optimal process scheduling we accepted the criterion of the minimum total time of task performance. The problem is solved with the aid of a multistage programming algorithm.