ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1982

Seria: AUTOMATYKA z. 64 Nr kol. 736

Franciszek MARECKI
Irena ŚLESIŃSKA

PROBLEMY HARMONOGRAMOWANIA PROCESU LAKIEROWANIA KAROSERII

Streszczenie. W referacie przedstawiono model matematyczny i algorytm optymalnego harmonogramowania procesu lakierowania karoserii samochodowych. Do rozwiązania problemu wykorzystano metodę programowania wieloetapowego.

1. WPROWADZENIE

Produkcja karoserii samochodowych jest realizowana na wydziale spawalniczo-montażowym oraz lakierni. Elementy składowe karoserii (tzw.wytłoczki) są łączone (spawane lub zgrzewane) w gniazdach technologicznych.Z uwagi na przezbrojenia tych gniazd karoserie wytwarza się partiami różnych wersji. Wersja formalnie jest określonym zbiorem wytłoczek. Harmonogram produkcji wydziału spawalniczo-montażowego określa przedziały czasu (lub inaczej kolejność) realizacji wersji karoserii.

Z wydziału spawalniczo-montażowego karoserie są przekazywane (transportem podwieszanym) do lakierni. Na każdej zawieszce znajduje się jedna karoseria. Zasadniczym agregatem lakierni jest tunel. W tunelu tym karoserie przesuwają się na transporterze ze stałą prędkością w trakcie procesu lakierowania. Przy zmianie koloru, ze względów technologicznych,określona liczba zawieszek jest pusta. Liczba pustych miejsc jest zależna od poprzedniego i następnego koloru.

W ogólnym przypadku karoserie tej samej wersji są lakierowano na różne kolory. Stąd w lakierni wyróżnia się tzw. odmiany kolorystyczne karoserii. Odmianę kolorystyczną charakteryzuje wersja i kolor. Należy zwrócić uwagę na fakt, że do lakierni podawane są karoserie partiami różnych wersji, o ustalonej kolejności. W ramach różnych wersji mogą występować te same kolory. Grupując odmiany kolorystyczne o identycznym kolorze zmniejszamy liczbę pustych zawieszek w tunelu lakierni. Zatem celem harmonogramowania procesu lakierowania karoserii jest maksymalizacja wydajności tunelu lakierni (czyli minimalizacja sumy pustych zawieszek).

W procesie lakierowania karoserii występują charakterystyczne ograniczenia logiczne. Jeżeli kolejność wersji jest ustalona, to każda odmiana kolorystyczna wcześniejszej wersji poprzedza każdą odmianę kolorystyczną

późniejszej wersji. Ponadto w ramach wersji kolejność odmian kolorystycznych jest dowolna. Sytuację komplikuje nieco magazyn buforowy (typu nitkowego) znajdujący się pomiędzy wydziałem spawalniczo-montażowym a lakiernią. Uwzględniając tzw. ekonomiczną długość serii karoserii tej samej odmiany kolorystycznej założymy, że w buforze może znajdować się co najwyżej partia jednej odmiany kolorystycznej. W tym przypadku ograniczenia procesu lakierowania są bardziej złożone (aniżeli typowe ograniczenia kolejnościowe). Są to ograniczenia kolejnościowe zmienne w czasie (uzależnione od zawartości magazynu buforowego).

Problematyka harmonogramowania procesu produkcji karoserii samochodowych była przedmiotem prac [1] i [2]. W pracach tych przedstawiono modele ogólne wydziału spawalniczo-montażowego i lakierni. Szczegółową analizę procesu spawalniczo-montażowego i lakierowania karoserii (na przykładzie FSM Tychy) przedstawiono w [3] i [4]. W niniejszym referacie zostanie pokazany model matematyczny i algorytm harmonogramowania procesu lakierowania. Z formalnego punktu widzenia jest to problem realizacji zadań (z ograniczeniami logicznymi) na jednym agregacie. Ponieważ jest to problem NP-zupełny - w sensie złożoności obliczeniowej - dlatego proponowany algorytm ma złożoność wykładniczą.

2. SFORMUŁOWANIE PROBLEMU - MODEL MATEMATYCZNY

Załóżmy, że dany jest zbiór wersji karoserii, które powstają na wydziale spawalniczo-montażowym:

$$W = \{w_r\}_{(r=\overline{1,R})} \tag{1}$$

gdzie:

w_r - r-ta wersja,

R - liczba wersji.

Poszczególne wersje napływają przed tunel lakierni w określonej kolejności, w znanych chwilach czasu.

Dany jest zbiór kolorów, na które karoserie mogą być malowane:

$$C = \{c_k\}_{(k=\overline{1,K})} \tag{2}$$

gdzie:

c_k - k-ty kolor,

K - liczba kolorów.

Załóżmy, że dane są czasy potrzebne na zmianę koloru w tunelu lakierni:

$$T = \left[\tau_{\ell,k}\right]_{\substack{(\ell=\overline{1,K})\\(k=\overline{1,K})}} \tag{3}$$

gdzie:

$\tau_{\ell,k}$ - czas zmiany koloru ℓ-tego na k-ty.

Niechaj przyporządkowanie kolorów do wersji przedstawia macierz:

$$A = \left[a_{rk}\right] \tag{4}$$

gdzie:

$$a_{rk} = \begin{cases} 1 : \text{gdy w wersji } w_r \text{ występuje malowanie na kolor } c_k \\ 0 : \text{w przypadku przeciwnym} \end{cases}$$

Z każdą wersję związana jest zatem pewna liczba odmian kolorystycznych N_r, określona następująco:

$$N_r = \sum_{k=1}^{k=K} a_{rk} \tag{5}$$

Interpretując zadania jako odmiany kolorystyczne, możemy zdefiniować zbiór zadań lakierni:

$$\Omega = \left\{\omega_n\right\}_{(n=\overline{1,N})} \tag{6}$$

gdzie:

ω_n - n-te zadanie,

N - liczba zadań.

Ogólna liczba zadań wynosi:

$$N = \sum_{r=1}^{r=R} N_r \tag{7}$$

Zadania numerujemy tak, by z numeru zadania ω_n wynikała wersja karoserii i odmiana kolorystyczna. Zatem:

$$\bigwedge_{1 \leqslant r \leqslant R} \left(\sum_{j=0}^{j=r-1} N_j < n \leqslant \sum_{j=0}^{j=r} N_j \right) \Rightarrow (\omega_n \hat{=} w_r) \tag{8}$$

przy czym: $\hat{=}$ symbol odpowiedniości, $N_o = 0$.

Numer koloru c_k dla zadania ω_n wyznaczamy z warunku:

$$\bigwedge_{1 \leqslant r \leqslant R} \left\{ \left(\sum_{j=0}^{j=r-1} N_j < n \leqslant \sum_{r=0}^{j=r} N_j \right) \wedge \left(J = n - \sum_{j=0}^{j=r-1} N_j \right) \wedge \right.$$

$$\left. \wedge \left(J = \sum_{i=1}^{i=J^*} a_{ri} \right) \wedge \left(\sum_{i=1}^{i=J^*-1} a_{ri} = J - 1 \right) \right\} \Rightarrow (J^* = c_k) \wedge (\omega_n \hat{=} c_k) \tag{9}$$

Załóżmy, że czasy realizacji zadań dane są wektorem:

$$\Theta = [\vartheta_n] \tag{10}$$

gdzie:

ϑ_n - czas realizacji zadania ω_n.

Dla każdego zadania określone są terminy najwcześniejszego rozpoczęcia realizacji:

$$\Phi = [\varphi_n] \tag{11}$$

gdzie:

φ_n - termin najwcześniejszego rozpoczęcia zadania ω_n

oraz terminy najpóźniejszego zakończenia zadań:

$$\Psi = [\psi_n] \tag{12}$$

gdzie:

ψ_n - termin najpóźniejszego zakończenia ω_n.

Ponadto założymy, że w buforze przed tunelem lakierni może być pozostawione co najwyżej jedno zadanie, jeżeli czas realizacji tego zadania spełnia warunek:

$$\vartheta_n < \vartheta_o \tag{13}$$

gdzie:

ψ_0 - dopuszczalny czas realizacji zadania, które może być pozostawione w buforze.

Oznaczmy przez t_n moment zakończenia realizacji zadania ω_n. Dla optymalizacji harmonogramowania przyjmiemy kryterium minimalizacji maksymalnego opóźnienia realizacji zadań:

$$Q = \max_{1 \leq n \leq N} (t_n - \psi_n) \rightarrow \min \tag{14}$$

A zatem dopuszczamy opóźnienie realizacji zadań. Harmonogram winien określać przedział czasu realizacji każdego zadania.

3. ALGORYTM

Do rozwiązania problemu zostanie wykorzystany algorytm programowania wieloetapowego. Istota optymalizacji wieloetapowej polega na tym, że proces znajdowania rozwiązań jest wieloetapowym procesem decyzyjnym. Każda decyzja podejmowana na φ-tym etapie ($\varphi = \overline{0,N}$) wiąże się z przydziałem jednego zadania do realizacji. Stany etapu φ-tego są wyznaczane na podstawie stanów etapu $(\varphi-1)$-ego. Początkowy stan procesu interpretuje sytuację, gdy nie przydzielono do realizacji żadnego zadania. Stany końcowe przedstawiają dopuszczalne warianty przydziału wszystkich zadań. Z każdym stanem związana jest jego wartość, wynikająca z przyjętego kryterium optymalizacji.

3.1. Procedura generowania stanów

Generowanie stanów jest procedurą pozwalającą wygenerować na podstawie pewnego stanu etapu $(\varphi-1)$-ego stany etapu φ-tego.

Wprowadzamy następującą definicję stanu:

Definicja 1: Stanem procesu decyzyjnego jest wektor:

$$p^{\lambda,\varphi} = \left[p_n^{\lambda,\varphi}\right]_{\substack{(n=\overline{1,N}) \\ (\lambda=\overline{1,L_\varphi}) \\ (\varphi=\overline{0,N})}} \tag{15}$$

przy czym:

φ - numer etapu decyzyjnego,

λ - numer stanu w ramach etapu,

L_φ - liczba stanów na φ-tym etapie.

Elementy wektora stanu określamy następująco:

$$p_n^{\lambda,\varrho} = \begin{cases} t_n & : \text{jeśli zadanie } \omega_n \text{ zostało} \\ & \quad \text{przydzielone do realizacji,} \\ 0 & : \text{w przypadku przeciwnym} \end{cases}$$

Stan początkowy $P^{1,0}$ spełnia warunek:

$$\bigvee_{1 \leq n \leq N} p_n^{1,0} = 0. \tag{16}$$

natomiast w każdym stanie końcowym $P^{\lambda,N}$:

$$\bigvee_{1 \leq n \leq N} p_n^{\lambda,N} > 0 \tag{17}$$

Z każdym stanem wiążemy jego wartość, zdefiniowaną następująco:

Definicja 2: Wartością stanu $P^{\lambda,\varrho}$ jest liczba $V^{\lambda,\varrho}$ określona zależnością:

$$\bigvee_{\mu} \left[(p_\mu^{\lambda,\varrho} > 0) \Rightarrow (\mu \in \beta) \right] \Rightarrow V^{\lambda,\varrho} = \max_{\mu \in \beta} (p_\mu^{\lambda,\varrho} - \psi_n) \tag{18}$$

Wartość stanu $P^{\lambda,\varrho}$ obliczamy na podstawie wartości stanu bezpośrednio go poprzedzającego w sieci generacji, a więc:

$$V^{\lambda,\varrho} = \max(V^{\lambda,\varrho-1}; \Delta V^{\lambda,\varrho-1;\lambda,\varrho}) \tag{19}$$

gdzie:

$\Delta V^{\lambda,\varrho-1;\lambda,\varrho}$ - przyrost wartości wynikający z przejścia od stanu $P^{\lambda,\varrho-1}$ do $P^{\lambda,\varrho}$.

Procedura generowania stanów polega na uzupełnieniu stanu $P^{\lambda,\varrho-1}$ o dopuszczalne zadanie ω_n, co prowadzi do stanu $P^{\lambda_n,\varrho}(1 \leq \lambda_n \leq L_\varrho)$. Uzupełnienie stanu o zadanie ω_n wiąże się ze zmianą odpowiedniej współrzędnej, co zapisujemy następująco:

$$P^{\lambda_n,\varrho} = P^{\lambda,\varrho-1} + \Delta P^{\lambda,\varrho-1;\lambda_n,\varrho} \tag{20}$$

Wektor $\Delta P^{1,\varphi-1;\lambda_n,\varphi}$ posiada następujące współrzędne:

$$\Delta P_\mu^{1,\varphi-1;\lambda_n,\varphi} = \begin{cases} t_n & : \mu = n \\ 0 & : \mu \neq n \end{cases} \tag{21}$$

Zadanie ω_n jest dla stanu $P^{1,\varphi-1}$ dopuszczalne, jeżeli spełnia wszystkie ograniczenia rozważanego problemu, a więc:

1) ω_n nie może należeć do stanu $P^{1,\varphi-1}$, tzn.:

$$p_n^{1,\varphi-1} = 0 \tag{22}$$

2) moment pojawienia się zadania przed tunelem lakierni - φ_n nie powinien być późniejszy niż moment zakończenia realizacji ostatniego zadania ze stanu $P^{1,\varphi-1}$

$$\varphi_n \leqslant \max_{1 \leqslant \mu \leqslant N} p_\mu^{1,\varphi-1} \tag{23}$$

wynika to z przyjętego założenia o niedopuszczalnych przestojach tunelu;

3) zadanie ω_n musi być możliwe do wykonania, co zapiszemy w postaci:

$$n \in \alpha^{1,\varphi-1} \tag{24}$$

gdzie:

$\alpha^{1,\varphi-1}$ - zbiór zadań, które mogą być wykonane bezpośrednio ze stanu $P^{1,\varphi-1}$.

Sposób określenia zbioru $\alpha^{1,\varphi-1}$ zostanie przedstawiony w następnym punkcie pracy.

Ogólna procedura generowania stanów ma postać:

$$\bigvee_n (n \in \alpha^{1,\varphi-1}) \wedge (p_n^{1,\varphi-1} = 0) \wedge \varphi_n \leqslant \max_{1 \leqslant \mu \leqslant N} p_\mu^{1,\varphi-1}) \Rightarrow$$

$$\Rightarrow (P^{\lambda_n,\varphi} = P^{1,\varphi-1} + \Delta P^{1,\varphi-1;\lambda_n,\varphi}) \tag{25}$$

Wektor ΔP określa zależność (21), przy czym:

$$t_n = \vartheta_n + \max_{1 \leqslant \mu \leqslant N} p_\mu^{1,\varphi-1} + \tilde{\tau}_{\varkappa,k} \tag{26}$$

gdzie:

$\mathcal{H}$ - numer ostatniego zadania w stanie $P^{1,\eta-1}$.

Po wygenerowaniu stanów etapu ostatniego wyznaczamy stan optymalny o najmniejszej wartości (ze względu na przyjęte kryterium):

$$\left(\min_{1\leqslant\lambda\leqslant L_N} V^{\lambda,N} = V^{\lambda^o,N}\right) \Rightarrow \left(P^{\lambda_o,N} = P^o\right) \tag{27}$$

gdzie:

P^o - stan optymalny na N-tym etapie.

Ze stanu P^o wyznaczamy optymalny harmonogram realizacji zadań w postaci ciągu uporządkowanych N dwójek:

$$H^o = \langle\langle \varrho_1, t_1\rangle; \langle \varrho_2, t_2\rangle; \ldots \langle \varrho_n, t_n\rangle; \ldots \langle \varrho_N, t_N\rangle\rangle \tag{28}$$

gdzie:

ϱ_n - moment rozpoczęcia realizacji zadania ω_n;

t_n - moment zakończenia realizacji ω_n

oraz:

$$\bigwedge_{1\leqslant n\leqslant N} (t_n = p_n^o) \wedge (\varrho_n = t_n - \vartheta_n) \tag{29}$$

3.2. Interpretacja zbioru $\alpha^{1,\eta-1}$

Zgodnie z (24) $\alpha^{1,\eta-1}$ oznacza zbiór zadań, które można wykonać bezpośrednio ze stanu $P^{1,\eta-1}$. Z definicji (15) wynika, że w stanie $P^{1,\eta-1}$ znany jest podzbiór $(\eta-1)$ zadań, które zostały już skierowane do tunelu lakierni. Wybór następnego zadania jest uzależniony od stanu magazynu buforowego. Rozpoczynając generowanie ze stanu $P^{1,0}$ rozważmy najpierw pierwsze N_1 zadań należących do pierwszej wersji, ze względu na ustaloną kolejność wersji. Wyróżnimy tu dwa przypadki:

1) z pierwszą wersją związane jest jedno zadanie ($N_1=1$). W tym przypadku zbiór $\alpha^{1,0}$ będzie zawierał tylko jedno zadanie ω_1, jeśli zadanie to nie mieści się w buforze. W przeciwnym przypadku do zbioru $\alpha^{1,0}$ należeć będą również zadania odpowiadające następnej wersji (drugiej);

2) w pierwszej wersji występują co najmniej dwie odmiany kolorystyczne ($N_1 > 2$). Wówczas do zbioru $\alpha^{1,0}$ wchodzą wszystkie zadania o numerach od 1 do N_1.

W ogólnym przypadku zasady tworzenia zbioru $\alpha^{1,\eta-1}$ dla dowolnego stanu $P^{1,\eta-1}$ można przedstawić następująco:

- zbiór $\alpha^{1,\eta-1}$ zdefiniujemy w postaci sumy logicznej trzech rozłącznych zbiorów:

$$\alpha^{1,\eta-1} = \alpha_1^{1,\eta-1} \cup \alpha_2^{1,\eta-1} \cup \alpha_3^{1,\eta-1} \tag{30}$$

Liczebność zbioru $\alpha_1^{1,\eta-1}$ jest co najwyżej równa jedności, tzn.:

$$\left|\alpha_1^{1,\eta-1}\right| \leqslant 1 \tag{31}$$

Do zbioru tego należą zadania przebywające w buforze. Zadanie ω_n należy do $\alpha_1^{1,\eta-1}$, jeżeli jest spełniony warunek:

$$\exists_n \; \exists_i \left[(p_n^{1,\eta-1} = 0) \wedge (i>n) \wedge (p_i^{1,\eta-1} > 0)\right] \wedge (r_n < r_i) \Rightarrow (n \in \alpha_1^{1,\eta-1}) \tag{32}$$

gdzie:

r_n - numer wersji ω_n.

Powyższy zapis oznacza, że pewne zadanie ω_n o numerze wcześniejszym niż pozostałe zadania już zrealizowane znajduje się w buforze.

Mówimy, że zadanie ω_ϑ należy do zbioru $\alpha_2^{1,\eta-1}$, jeżeli:

$$\exists_r \forall_\vartheta \left[(\max_{1 \leqslant i < N} p_i^{1,\eta-1} = p_\mu^{1,\eta-1}) \wedge (\sum_{j=0}^{j=r-1} N_j < \mu \leqslant \sum_{j=0}^{j=r} N_j) \wedge \right.$$

$$\left. \wedge (\sum_{j=0}^{j=r-1} N_j < \vartheta \leqslant \sum_{j=0}^{j=r} N_j) \wedge (p_\vartheta^{1,\eta-1} = 0)\right] \Rightarrow (\vartheta \in \alpha_2^{1,\eta-1}) \tag{33}$$

Na podstawie numeru zadania realizowanego jako ostatnie w stanie $P^{1,\eta-1}$ ustalamy wersję, do jakiej ono należy. Zadania tej wersji, które jeszcze nie zostały zrealizowane, należą do zbioru $\alpha_2^{1,\eta-1}$.

O istnieniu niepustego zbioru $\alpha_3^{1,\eta-1}$ mówimy wówczas, gdy zbiór $\alpha_1^{1,\eta-1}$ jest pusty, natomiast do zbioru $\alpha_2^{1,\eta-1}$ należy jedno zadanie, które mie-

ści się w buforze. Istnieje wtedy możliwość przydziału zadań z następnej w kolejności wersji. Warunek ten zapiszemy w postaci:

$$(|\alpha_1^{1,\eta-1}| = 0) \wedge (|\alpha_2^{1,\eta-1}| = 1) \wedge (\vartheta_\mu < \vartheta_0) \Rightarrow (\omega_\mu \in \alpha_2^{1,\eta-1}) \qquad (34)$$

3.3. Eliminacja stanów

Zasadniczą trudnością wykorzystania programowania wieloetapowego w praktyce jest duża zajętość pamięci operacyjnej dla zapisu stanów danego etapu. Z tego względu istotne znaczenie mają procedury pozwalające wykryć, że pewien stan $P^{\lambda_1,\eta}$ jest nieperspektywiczny, tzn. nie prowadzi do rozwiązania optymalnego.

W rozważanym problemie do eliminacji pewnych stanów wykorzystamy regułę dominacji. Regułę tę można stosować w odniesieniu do stanów alternatywnych.

Definicja 3: Dwa stany $P^{\lambda_1,\eta}$ i $P^{\lambda_2,\eta}$ są alternatywne, jeżeli spełniają warunki:

$$\bigwedge_\mu [(p_\mu^{\lambda_1,\eta} > 0) \Leftrightarrow (p_\mu^{\lambda_2,\eta} > 0)] \wedge [(\max_{1 \leq i \leq N} p_i^{\lambda_1,\eta} = p_\vartheta^{\lambda_1,\eta}) \Leftrightarrow$$

$$\Leftrightarrow (\max_{1 \leq i \leq N} p_i^{\lambda_2,\eta} = p_\vartheta^{\lambda_2,\eta})] \Rightarrow (P^{\lambda_1,\eta} \cong P^{\lambda_2,\eta}) \qquad (35)$$

gdzie:

$\cong$ - symbol alternatywności.

Oznaczmy przez $P^{\lambda_1^0,N}$ najlepszy (lokalnie optymalny) stan końcowy uzyskany za stanu $P^{\lambda_1,\eta}$, natomiast przez $P^{\lambda_2^0,N}$ stan uzyskany z $P^{\lambda_2,\eta}$.

Definicja 4: Stan $P^{\lambda_1,\eta}$ dominuje nad stanem $P^{\lambda_2,\eta}$, jeżeli jest spełniony warunek:

$$(V^{\lambda_1^0,N} < V^{\lambda_2^0,N}) \Rightarrow (P^{\lambda_1,\eta} \rightarrowtail P^{\lambda_2,\eta}) \qquad (36)$$

gdzie:

$\rightarrowtail$ - symbol dominacji stanów.

W algorytmie programowania wieloetapowego zapamiętywane są stany tylko jednego etapu. O dominacji trzeba zadecydować na η-tym etapie, w trakcie generowania stanów.

Reguła dominacji opiera się na następującym twierdzeniu:

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

$$\bigwedge_n (P^{\lambda_1,?} \cong P^{\lambda_2,?}) \wedge (V^{\lambda_1,?} < V^{\lambda_2,?}) \Rightarrow (P^{\lambda_2,?} \mapsto P^{\lambda_1,?}) \qquad (37)$$

Dowód tego twierdzenia można przeprowadzić nie wprost, zakładając, że implikacja (37) ma przeciwny sens, tzn. $P^{\lambda_1,?} \mapsto P^{\lambda_2,?}$. Z definicji 4 wynika, że będzie zachodziła zależność:

$$V^{\lambda_1^o,N} > V^{\lambda_2^o,N} \qquad (38)$$

Niech H_o oznacza optymalny harmonogram realizowany od stanu $P^{\lambda_2,?}$. Harmonogram ten może być realizowany od stanu $P^{\lambda_1,?}$ ze względu na alternatywność tych stanów. A zatem:

$$V^{\lambda_1^o,N} = \max\left[V^{\lambda_1,?};\ \Delta V(H_o)\right] \qquad (39)$$

$$V^{\lambda_2^o,N} = \max\left[V^{\lambda_2,?};\ \Delta V(H_o)\right] \qquad (40)$$

gdzie:

$\Delta V(H_o)$ - przyrost wartości stanu uzyskany w wyniku zastosowania H_o.

Biorąc pod uwagę (36), (39) i (40) otrzymujemy:

$$V^{\lambda_1^o,N} < V^{\lambda_2^o,N},$$

co jest sprzeczne z założeniem (38).

Tym samym dowód został przeprowadzony.

4. UWAGI KOŃCOWE

Rozpatrywany w referacie problem harmonogramowania montażu i lakierowania karoserii samochodowych został przedstawiony w postaci przepływu zadań przez system dwóch agregatów połączonych szeregowo. Założono, że pomiędzy agregatami istnieje bufor o pojemności jednego zadania. Realizacja zadań na drugim agregacie wymaga przezbrojeń. Z tego względu problem har-

monogramowania zadań na drugim agregacie (problem komiwojazera) należy do klasy NP - zupełnych. Stąd harmonogramowanie optymalne zadań w rozpatrywanym systemie nie może być rozwiązane za pomocą algorytmu o wielomianowej złożoności obliczeniowej.

Do rozwiązania sformułowanego problemu zaproponowano algorytm programowania wieloetapowego. Za pomocą tego algorytmu można uzyskać rozwiązanie optymalne lub heurystyczne, gdy ograniczony jest czas lub pamięć operacyjna komputera. Rozwiązanie optymalne jest osiągalne, gdy liczba zadań jest względnie mała lub liczba ograniczeń względnie duża. Dla podwyższenia efektywności algorytmu wprowadzono reguły eliminacji stanów nieperspektywicznych. Ocena tych reguł będzie przedmiotem dalszych prac - poprzez przeprowadzenie testów komputerowych.

LITERATURA

[1] ŚLESIŃSKA I., MARECKI F.: Algorytm harmonogramowania niezależnych zadań na dwóch szeregowych maszynach. ZN Pol. Śl. s. Automatyka (oddano do druku).

[2] ŚLESIŃSKA I., MARECKI F.: Algorytm harmonogramowania zależnych zadań na dwóch szeregowych maszynach z magazynem buforowym. ZN Pol. Śl. s. Automatyka (oddano do druku).

[3] Raport z pracy n-b nt. "Optymalizacja harmonogramowania produkcji Wydziału Tłoczni w Zakładzie nr 2 w Tychach (etap IV) rozdział VI "Sterowanie Lakiernią", ss. 310-354, Instytut Automatyki, Gliwice 1981.

[4] ZGORZYK M.: Modelowanie cyfrowe lakierni samochodów małolitrażowych FIAT-126p. Praca dyplomowa, Instytut Automatyki, Gliwice 1980.

Recenzent: Prof. dr hab. inż. Zdzisław TRYBALSKI

Wpłynęło do Redakcji 15.05.1982 r.

ПРОБЛЕМЫ СОСТАВЛЕНИЯ ГРАФИКОВ ОКРАСКИ КУЗОВОВ

Р е з ю м е

В статье показана математическая модель и алгоритм оптимального составления графиков процесса окраски автомобильных кузовов. Для решения задачи использовано метод многошагового программирования.

THE PROBLEM OF SCHEDULING OF THE CAR-BODY LAQUERING PROCESS

S u m m a r y

We present the mathematical model and the algorithm of optimal scheduling of the car-body laquering process. To solve this problem,we used the multistage programming method.