ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1984
Serie: AUTOMATYKA z. 74 Nr kol. 810

Józef Grabowski, Eugeniusz Nowicki,
Stanisław Zdrzałka
Politechnika Wrocławska

SZEREGOWANIE ZADAŃ W DYSKRETNYM PROCESIE PRODUKCYJNYM Z GNIAZDEM KRYTYCZNYM*

Streszczenie. Praca zawiera nowy algorytm zagadnienia kolejnościowego w procesie produkcyjnym z jednym gniazdem krytycznym. Algorytm ten jest oparty na idei bloków i może być stosowany w systemach uwarunkowanych czasowo. Oznacza to, że w każdej iteracji istnieje pewne rozwiązanie i w przypadku przerwania obliczeń, można uznać je za rozwiązanie suboptymalne. Przedstawiono również wyniki obliczeń numerycznych dla znacznej liczby losowych przykładów.

1. Wstęp

W praktyce zagadnienia kolejnościowe, formułowane jako zadania optymalizacji procesu, są dość skomplikowane i najczęściej są zagadnieniami NP-zupełnymi. Dla tych zagadnień, chcąc uzyskać dobre rozwiązanie, musimy stosować metody przeglądu /metoda podziału i ograniczeń, programowanie dynamiczne/, które są czasochłonne, szczególnie w sytuacji, gdy liczba agregatów /i zadań/ jest stosunkowo duża. Zmniejszenie liczby agregatów, na których należy wyznaczać optymalną kolejność przyspiesza w istotny sposób działanie algorytmów opartych na tych metodach. Stąd też nasuwa się wniosek, aby ustalać optymalną kolejność zadań /lub operacji/ na agregatach stanowiących tzw. "wąskie gardło" /gniazdo krytyczne/. W konkretnych procesach produkcyjnych identyfikacja gniazd krytycznych przez służby techniczno-organizacyjne nie nastręcza większych trudności. Zwykle gniazdami krytycznymi są urządzenia, których koszty zakupu i eksploatacji są szczególnie duże.

W niniejszym artykule przedstawimy pewne zagadnienie kolejnościowe, które może wystąpić w przypadku zaistnienia wąskiego gardła na jednym agregacie technologicznym.

Dany jest zbiór zadań $J = \{J_1, J_2, \ldots, J_n\}$, które mają być wykonane przy użyciu jednej maszyny M_1 stanowiącej gniazdo krytyczne. Każde składa się z ciągu operacji, tzn. $J_i = \langle O_{i1}, O_{i2}, \ldots, O_{in_i} \rangle$, przy czym tylko jedna z tych operacji /krytyczna/ jest wykonywana na maszynie krytycznej M_1 w czasie $p_i > 0$. Pozostałe operacje są wykonywane na maszynach niekrytycznych /nie musimy ich szeregować/ i niech:

r_i - będzie sumą czasów trwania wszystkich operacji występujących przed operacją krytyczną,

* Praca wykonana w ramach problemu resortowego R.I.21.01

q_i - będzie sumą czasów trwania wszystkich operacji występujących po operacji krytycznej.

Jeżeli kolejność wykonywania zadań na maszynie M_1 jest określona, to dla każdego zadania J_i możemy wyznaczyć terminy rozpoczęcia wykonywania S_i oraz zakończenia C_i na maszynie /przy czym musi być $S_i \geq r_i$/. Zagadnienie optymalizacji polega na ustaleniu takiej kolejności wykonywania zadań na maszynie M_1, aby łączny czas wykonywania wszystkich zadań $C_{max} = \max_i\{C_i+q_i\}$ osiągnął wartość minimalną. Zagadnienie to jest oznaczane symbolem $n|1|r_i \geq 0, q_i \geq 0|C_{max}$ i jest równoważne zagadnieniu $n|1|r_i \geq 0|L_{max}$. Związek z nieterminowością L_{max} jest następujący; przyjmując $K = \max_i q_i$, $d_i = K-q_i$, otrzymujemy: $C_{max}=\max_i\{C_i+q_i\} =\max_i\{C_i-d_i\} +K=\max_i L_i+K=L_{max}+K$ i zamiast minimalizacji C_{max} możemy równoważnie minimalizować L_{max}.

Zagadnienie to było przedmiotem rozważań wielu autorów. Dokładne algorytmy rozwiązania tego zagadnienia były prezentowane w pracach [2], [6], [7]. Algorytmy heurystyczne zaś w artykułach [1], [6], [8]. W niniejszej pracy przedstawimy nowy algorytm dokładny wraz z wynikami obliczeniowymi na maszynie cyfrowej. Algorytm ten ma tę zaletę, że może współpracować z dowolnym algorytmem heurystycznym w przeciwieństwie do wcześniej proponowanych w literaturze algorytmów, w których bazuje się na algorytmie Schragego [6], [8] i użycie innych algorytmów jest wykluczone. Wyniki obliczeniowe wykazały wysoką efektywność algorytmu.

2. Pewne własności zagadnienia $n|1|r_i \geq 0, q_i \geq 0|C_{max}$

Niech $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ będzie dowolną permutacją liczb $\{1,2,\ldots,n\}$. Niech Π będzie zbiorem wszystkich tych permutacji. Każda permutacja $\pi \in \Pi$ określa kolejność wykonywania zadań $\langle J_{\pi(1)}, J_{\pi(2)}, \ldots, J_{\pi(n)}\rangle$ na maszynie M_1. Zadanie optymalizacji polega na znalezieniu takiej permutacji π^*, dla której zachodzi

$$C_{max}(\pi^*) = \min_{\pi\in\Pi} C_{max}(\pi),$$

gdzie

$$C_{max}(\pi) = \max_i \left\{ C_{\pi(i)} + q_{\pi(i)} \right\}.$$

Wartość $C_{max}(\pi)$ oznacza wartość funkcji celu dla kolejności określonej przez permutację π. Aby wyznaczyć wartość $C_{max}(\pi)$ posłużymy się wyrażeniem:

$$C_{max}(\pi) = \max_{1\leq i_1\leq i_2\leq n} \left\{ r_{\pi(i_1)} + \sum_{i=i_1}^{i_2} p_{\pi(i)} + q_{\pi(i_1)} \right\}.$$

Wyrażenie to jest szczególnym przypadkiem analogicznego wyrażenia z pracy [5], w którym należy przyjąć $m=1$ /gdzie m jest liczbą maszyn/.

Każdą dwójkę liczb całkowitych $\langle i_1, i_2\rangle$ spełniających nierówność $1 \leq i_1 \leq i_2 \leq n$ będziemy nazywać drogą w π. Drogą $\langle u_1, u_2\rangle$ taką, dla której zachodzi:

$$C_{max}(\pi) = r_{\pi(u_1^\pi)} + \sum_{i=u_1^\pi}^{u_2^\pi} p_{\pi(i)} + q_{\pi(u_2^\pi)}$$

będziemy nazywać drogą krytyczną.

Wprowadźmy teraz kilka pojęć i oznaczeń dla każdej permutacji $\pi \in \Pi$.

Ciąg zadań $\langle J_{\pi(u_1^\pi)}, J_{\pi(u_1^\pi+1)}, \ldots, J_{\pi(u_2^\pi)} \rangle$ będziemy nazywać blokiem w permutacji π. Zadanie $J_{\pi(u_1^\pi)}$ oraz $J_{\pi(u_2^\pi)}$ będziemy nazywać odpowiednio zadaniem pierwszym oraz ostatnim w bloku w π. Warto zauważyć, że w czasie wykonywania zadań z bloku, maszyna wykonuje te zadania bez przestoju /rys. 1/.

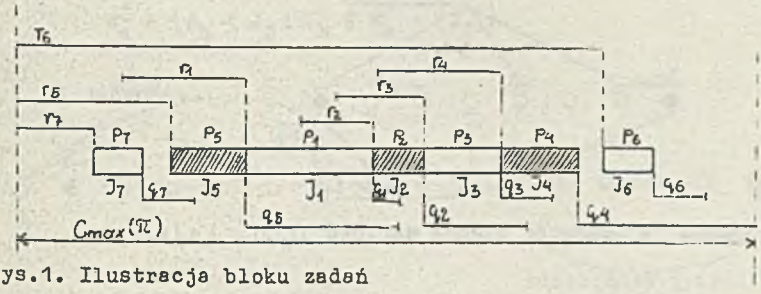
Dalej niech

$$P^\pi = \{\pi(j) : u_1^\pi \leq j \leq u_2^\pi\}$$

będzie zbiorem wskaźników zadań bloku w π. W dalszych rozważaniach będziemy opuszczać wskaźnik π w oznaczeniach u_1^π, u_2^π, P^π. Wszystkie zdefiniowane pojęcia są zilustrowane na rys. 1.

Przykład /rys. 1/

Rozważmy przykład $7|1|r_i \geq 0, q_i \geq 0|C_{max}$, dla którego mamy: $\pi = \langle 7,5,1,2,3,4,6 \rangle$, $C_{max}(\pi) = r_5+p_5+p_1+p_2+p_3+p_4+q_4$ /zaznaczono linią przerywaną/, $u_1=2$, $u_2=6$, $J_{\pi(u_1)} = J_5$, $J_{\pi(u_2)} = J_4$, $P = \{5,1,2,3,4\}$.



Rys.1. Ilustracja bloku zadań

Twierdzenie 1

Niech $\pi \in \Pi$ będzie dowolną permutacją określającą kolejność wykonywania zadań z drogą krytyczną $\langle u_1, u_2 \rangle$. Jeżeli istnieje permutacja $\beta \in \Pi$ taka, że $C_{max}(\beta) < C_{max}(\pi)$, wtedy w β przynajmniej jedno zadanie z bloku w π poprzedza pierwsze zadanie lub występuje za ostatnim zadaniem tego bloku.

Z twierdzenia 1 wynika, że jeżeli permutacja β została otrzymana z permutacji π przez zamianę kolejności wykonywania zadań w π oraz jeżeli $C_{max}(\beta) < C_{max}(\pi)$, wtedy w β przynajmniej jedno zadanie z bloku w π zostało przesunięte przed pierwsze zadanie /lub za ostatnie zadanie/ tego bloku.

3. Schemat przebiegu algorytmu

Algorytm rozwiązania zagadnienia $n|1|r_i \geqslant 0, q_i \geqslant 0|C_{max}$ jest oparty na metodzie podziału i ograniczeń z mieszaną strategią podziału /z cofaniem/.

Rozpoczynając od permutacji $\alpha \in \Pi$ generujemy ciąg permutacji $\pi \in \Pi$. Dla każdej permutacji z ciągu obliczamy wartość $C_{max}(\pi)$, wyznaczamy drogę krytyczną $\langle u_1, u_2 \rangle$ oraz określamy blok zadań. Jeżeli aktualne górne ograniczenie C^* jest większe niż $C_{max}(\pi)$, to należy przyjąć $C^* := C_{max}(\pi)$. Każdą nową permutację $\beta \in \Pi$ otrzymujemy z permutacji $\pi \in \Pi$ przez przesunięcie jednego zadania J_j przed pierwsze /lub za ostatnie/ zadanie bloku w π. Zadania, które są przesuwane przed pierwsze zadanie bloku /tzn. na pozycję u_1/, są określone w zbiorze kandydatów /rys. 2/:

$$E_b \triangleq \{J_j : j \in P,\ \Delta_b(j) < 0\},$$

gdzie:

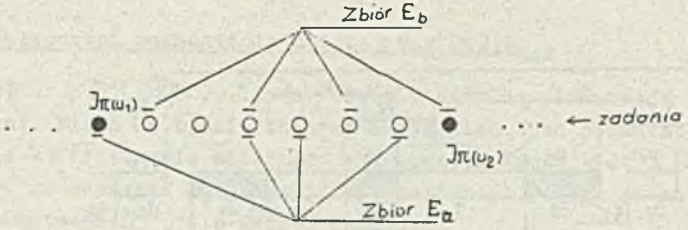
$$\Delta_b(j) \triangleq r_{\pi(j)} - r_{\pi(u_1)}, \quad j \in P.$$

Natomiast zadania, które są przesuwane za ostatnie zadanie bloku /tzn. na pozycję u_2/, są określone w zbiorze kandydatów /rys. 2/:

$$E_a \triangleq \{J_j : j \in P,\ \Delta_a(j) < 0\},$$

gdzie:

$$\Delta_a(j) \triangleq q_{\pi(j)} - q_{\pi(u_2)}, \quad j \in P.$$



Symbol "-" oznacza ujemną wartość $\Delta_\lambda(j)$, $\lambda \in \{a, b\}$.

Rys.2. Zbiory kandydatów

Liczba możliwych permutacji β, które możemy bezpośrednio wygenerować z π wynosi $s = e_b + e_a$, gdzie $e_b = |E_b|$, $e_a = |E_a|$.
Jeżeli zadanie J_j jest przesuwane na pozycję u_1, wówczas jest ono wykonywane w β jako zadanie pierwsze spośród wszystkich zadań ze zbioru

$$E_b' = E_b \cup \{J_{\pi(u_1)}\}.$$

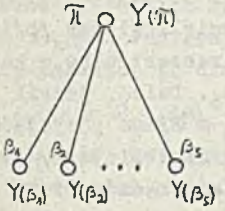
Natomiast w przypadku przesunięcia zadania J_j na pozycję u_2, jest ono wykonywane jako zadanie ostatnie spośród wszystkich ze zbioru

$$E_a' = E_a \cup \{J_{\pi(u_2)}\}.$$

Proces generowania permutacji przedstawimy w postaci drzewa H. Każdy węzeł w H będzie przedstawiał pewną permutację π, a łuk w H będzie odpowiadał pewnej parze $\langle \pi, \beta \rangle$, gdzie β został otrzymany przez przesunięcie jednego zadania w π. Generowanie bezpośrednich następników β_r /rys. 3/ względem π powoduje, że zbiór rozwiązań $Y(\pi)$ odpowiadający węzłowi π zostaje rozbity na podzbiory $Y(\beta_r)$ /niekoniecznie rozłączne/. Jeżeli β_r jest generowany przez przesunięcie $J_j \in E_b$, wówczas z podzbiorem $Y(\beta_r)$ związane jest zdanie logiczne wskazujące, że zadanie J_j ma być wykonane jako pierwsze spośród wszystkich zadań zbioru

$$E_b' = E_a \cup \{J_{\pi(u_1)}\}.$$

Zdanie to jest równoważne wymaganiom częściowego porządku /rys. 4a/:



Rys.3. Podział węzła

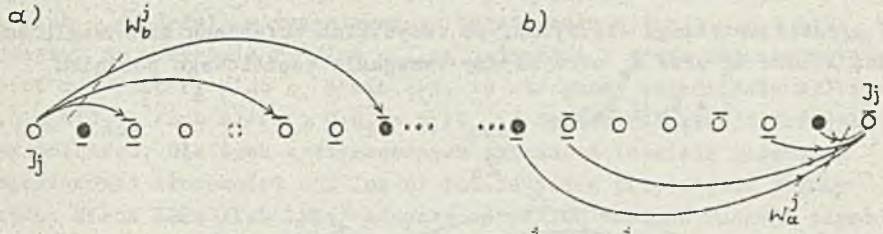
$$W_b^j = \{J_j \prec J_h : J_h \in E_b' - \{J_j\}\}.$$

Natomiast, jeżeli β_r jest generowany przez przesunięcie zadania $J_j \in E_a$, wówczas z podzbiorem $Y(\beta_r)$ związane jest zdanie logiczne, że zadanie J_j ma być wykonywane jako ostatnie spośród wszystkich zadań zbioru

$$E_a' = E_a \cup \{J_{\pi(u_2)}\}.$$

Zdanie to jest równoważne wymaganiom częściowego porządku /rys.4b/

$$W_a^j = \{J_h \prec J_j : J_h \in E_a' - \{J_j\}\}.$$



Rys.4. Wymagania częściowego porządku W_b^j i W_a^j

Dla ustalenia uwagi, niech β_1 będzie pierwszym następnikiem wygenerowanym z π przez przesunięcie $J_j \in E_b$. Po dokonaniu cofania z β_1 do π otrzymujemy zdanie logiczne dopełniające, że J_j nie może być wykonywane jako pierwsze spośród zadań zbioru E_b'. Dla zapewnienia rozłączności zbiorów rozwiązań $Y(\beta_r)$ /r = 1,2,...,s/, zdanie to powinno być respektowane w innych następnikach generowanych z π, tzn. we wszystkich β_r, $r \geqslant 2$. Jeżeli wszystkie zadania ze zbioru E_b były przesuwane, wtedy po wszystkich cofaniach do π w H względem tych zadań, otrzymujemy zdanie logiczne dopełniające:

/a/ Każde zadanie ze zbioru E_b nie może być wykonywane jako pierwsze spośród wszystkich zadań zbioru E_b'.

Z tego zdania logicznego wynika bezpośrednio następujące inne zdanie lo-

giczne

/b/ Zadanie $J_{\pi(u_1)} \in E_b'$ musi być wykonywane jako pierwsze spośród wszystkich zadań zbioru E_b'.

Zdanie logiczne /b/ możemy wyrazić poprzez wymagania częściowego porządku /rys. 5/:

$$Z_b = \{J_{\pi(u_1)} \prec J_h : J_h \in E_b\}.$$

Zauważmy, że po przebiegu tej części algorytmu, zdanie logiczne /b/ prezentowane w postaci zbioru Z_b, może być ustalone i respektowane w kolejnych następnikach generowanych z π, tzn. w β_r, $r > e_b$. Dalej, jeżeli β_{e_b+1} został wygenerowany z π przez przesunięcie $J_j \in E_a$ na pozycję u_2, wtedy, po cofaniu otrzymamy kolejne zdanie logiczne dopełniające, że zadanie J_j nie może być wykonywane jako ostatnie spośród zadań zbioru E_a'. Jeżeli teraz wszystkie zadania ze zbioru E_a były przesuwane, wtedy, po wszystkich cofaniach do π względem tych zadań, otrzymujemy zdanie logiczne dopełniające

/c/ Każde zadanie ze zbioru E_a nie może być wykonywane jako ostatnie spośród wszystkich zadań zbioru E_a'.

Z tego zdania logicznego wynika bezpośrednio następujące zdanie logiczne

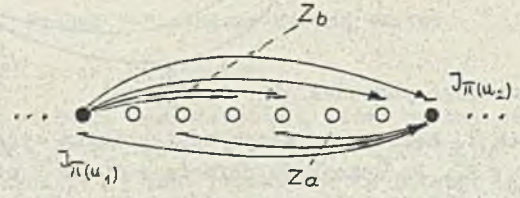
/d/ Zadanie $J_{\pi(u_2)} \in E_a'$ musi być wykonywane jako ostatnie spośród wszystkich zadań zbioru E_a'.

Zdanie logiczne /d/ możemy wyrazić poprzez wymagania częściowego porządku /rys. 5/:

$$Z_a = \{J_h \prec J_{\pi(u_2)} : J_h \in E_a\}.$$

W efekcie przebiegu algorytmu, po wszystkich cofaniach do π względem zadań zbioru E_b oraz E_a otrzymujemy wymagania częściowego porządku

$$Z = Z_b \cup Z_a.$$



Rys.5. Wymaganie częściowego porządku Z_a i Z_b

Twierdzenie 2

Niech $\pi \in \Pi$ będzie dowolną permutacją a Z_b oraz Z_a będą odpowiadającymi jej wymaganiami częściowego porządku. Wtedy dla dowolnej permutacji $\gamma \in \Pi$ spełniającej wymagania Z_b i Z_a, zachodzi

$$C_{max}(\gamma) \geqslant C_{max}(\pi).$$

Z twierdzenia 2 wynika, że dalsze generowanie następników z π nie doprowadzi nas do otrzymania "lepszej" permutacji niż permutacja π.

W tego rodzaju algorytmach istotną jest kolejność wyboru zadań ze zbioru E_b lub E_a celem dokonania przesunięcia. Będziemy dążyli do wyboru takiego zadania, którego przesunięcie wygeneruje następnik o możliwie najmniejszej wartości C_{max}.

Ocena wszystkich kandydatów jest dokonywana w oparciu o wartości wyrażeń $\Delta_\lambda(j)$. Postępując podobnie jak w pracy [4] /przy założeniu m = 1/ możemy pokazać, że dla każdej permutacji $\pi \in \Pi$ mamy:

$$C_{max}(\pi) \geq C_{max}(\beta) + \Delta_\lambda(j), \quad J_j \in E_\lambda, \quad \lambda \in \{a,b\}.$$

Z nierówności tej wynika, że zadania do przesunięcia powinny być wybierane w kolejności niemalejących wartości $\Delta_\lambda(j)$.

4. Dolne ograniczenia

Dla każdego węzła π w H należy znaleźć dolne ograniczenie wartości C_{max} dla wszystkich następników β /niekoniecznie bezpośrednich/ wygenerowanych z węzła π.

Pierwszym i oczywistym dolnym ograniczeniem jest wartość wyrażenia

$$LB1 = \max_{1 \leq j \leq n} \{r_j + p_j + q_j\}.$$

Wyznaczenie wartości LB1 wymaga O(n) iteracji.

Kolejne dolne ograniczenia możemy otrzymać poprzez relaksację wartości r_i /lub q_i/. Jeżeli w rozpatrywanym zagadnieniu $n|1|r_i \geq 0, q_i \geq 0|C_{max}$ przyjmiemy, że wszystkie r_i /lub q_i/ są jednakowe i równe najmniejszej wartości $r_* = \min_i r_i$ /lub $q_* = \min_i q_i$/, to otrzymamy zagadnienie $n|1|r_i = r_*, q_i \geq 0|C_{max}$ /lub $n|1|r_i \geq 0, q_i = q_*|C_{max}$/ będące relaksacją zagadnienia wyjściowego. Dla tych zrelaksowanych zagadnień istnieją algorytmy o wielomianowej złożoności O(n log n) rozwiązywane przy użyciu reguły Jacksona. Niech LB2r /lub LB2q/ oznacza optymalną wartość funkcji zagadnienia zrelaksowanego. W naszej implementacji stosowaliśmy ograniczenie LB2r jeżeli $\max_i r_i - \min_i r_i \geq \max_i q_i - \min_i q_i$ oraz ograniczenie LB2q jeżeli warunek ten nie był spełniony.

5. Algorytm

Przedstawimy teraz zasadnicze kroki algorytmu rozwiązania zagadnienia $n|1|r_i \geq 0, q_i \geq 0|C_{max}$. Niech korzeniem drzewa rozwiązań H będzie permutacja α oraz niech $C^* = \infty$. Dalej, niech π będzie aktualną permutacją /węzłem w H / w π -tej permutacji algorytmu.

Krok 1 - obliczyć dolne ograniczenie $LB = \max\{LB1, LB2(\cdot)\}$. Jeżeli $LB \geq C^*$, to przejść do kroku 4.

Krok 2 - Obliczyć $C_{max}(\pi)$. Jeżeli $C_{max}(\pi) < C^*$, to przyjąć $C^*:=C_{max}(\pi)$. Ustalić zbiory kandydatów E_b oraz E_a. Obliczyć $\Delta_\lambda(j)$ dla $J_{\pi(j)} \in E_\lambda$, $\lambda \in \{a,b\}$.

Krok 3 - Spośród wszystkich kandydatów w π wybrać zadanie $J_{\pi(j)}$, które ma najmniejszą wartość $\Delta_b(j)$ /lub $\Delta_a(j)$/. Wygenerować nową permutację β /węzeł w H / przez przesunięcie zadania $J_{\pi(j)}$ na pozycję u_1 /lub u_2/. Następnie przyjąć $\pi:=\beta$ i przejść do kroku 1.

Krok 4 - Cofnąć się do poprzednika γ permutacji π w H . Jeżeli należy cofnąć się od korzenia w H to stop. W przeciwnym przypadku przyjąć $\pi:=\gamma$ i przejść do kroku 3.

6. Wyniki obliczeniowe

W konkretnej implementacji algorytmu, istotnym problemem jest sposób wykorzystania wymagań częściowego porządku ze zbiorów W_b^j, W_a^j oraz Z_b. W naszym algorytmie, zbiory te były implementowane poprzez zwiększanie wartości r_i oraz q_i /zasady zwiększania tych wartości są takie same jak w pracy [6], przy założeniu m = 1/, co ma istotny wpływ na uzyskanie silniejszych dolnych ograniczeń.

Zaprezentowany algorytm został zaprogramowany w języku FORTRAN-1300 i sprawdzony na maszynie cyfrowej ODRA-1325. W celu empirycznego oszacowania czasu obliczeń przeprowadzono następujące badania numeryczne. Dla n = 20,40,80,150,200 generowano liczby r_i, p_i, q_i wg rozkładu jednostajnego odpowiednio z przedziałów [0,RMAX], [1,PMAX], [0,QMAX]. Przyjęto zakresy: RMAX = R · PMAX, PMAX = 50, QMAX = Q · PMAX, gdzie R = 0.5,2,0.5n,2n, Q = 0.5,2,0.5n,2n. Dla ustalonego n, dla każdej kombinacji R oraz Q /jest ich łącznie 16/ przeprowadzono obliczenia dla 5 wygenerowanych przykładów. Łącznie dla ustalonego n przeprowadzono obliczenia dla 80 przykładów testujących.

W celu wyznaczenia rozwiązania początkowego α /korzenia w H / posłużono się algorytmem Schragego przedstawionym w pracach [6], [8].

Wyniki obliczeniowe przedstawiono w tabeli:

n	Czasy obliczeń [sec]		Ilość węzłów		Liczba przykładów, w których wygenerowano tylko jeden węzeł
	Mediana	Średnia	Mediana	Średnia	
20	0.06	0.07	1	1.24	75
40	0.25	0.23	1	1.21	74
80	0.50	0.59	1	1.18	73
150	1.75	1.87	1	1.15	71
200	2.75	3.13	1	1.07	71

Omawiając wyniki obliczeniowe należy podkreślić, że dla większości przykładów /90%/ testowany algorytm wymagał tylko jednej iteracji. Co

więcej, rozmiar zagadnienia nie wpływał znacząco na liczbę tych przykładów. Warto zaznaczyć, że prezentowany algorytm wymaga wykonania co najmniej jednej iteracji, jeżeli permutacja początkowa jest optymalna. Wynika stąd, że dla 90% przykładów algorytm heurystyczny /Schragego/ wygenerował rozwiązanie optymalne a badany algorytm wykorzystując swoje własności nie wymagał generowania następnych rozwiązań. Można się spodziewać, że efektywność algorytmu można poprawić stosując do wyznaczenia δ algorytm heurystyczny Pottsa [7], który generuje rozwiązania nie gorsze niż algorytm Schragego.

Na szczególną uwagę zasługuje bardzo mała średnia liczba generowanych węzłów. Co więcej, liczba ta maleje wraz ze wzrostem rozmiaru zagadnienia, dla n = 200 wynosi ona zaledwie 1.07.

LITERATURA

[1] Baker K.R., Su Z.S., "Sequencing with due-dates and early start times to minimize maximum tardiness", Naval Res. Logist. Quart., vol. 21, 1974, str. 171-176.

[2] Carlier J., "The one-machine sequencing problem", European J. of Opnal. Res., vol. 11, 1982, str. 42-47.

[3] Grabowski J., Janiak A., Nowicki E., Smutnicki C., Zdrzałka S., "Wybrane problemy i programy optymalizacji dyskretnej dla szeregowania zadań", Część I, Etap 2, Podstawy Teoretyczne, Raport serii Sprawozdania nr 7/83, Wydawnictwo Politechniki Wrocławskiej, 1983 r.

[4] Grabowski J., "On two-machine scheduling with release and due dates to minimize maximum lateness", OPSEARCH 17, 1980, str. 133-154.

[5] Grabowski J., Skubalska E., Smutnicki C., "On flow-shop scheduling with release and due dates to minimize maximum lateness; J. Opl. Res. Soc., vol. 34, No. 7, 1983, str. 615-620.

[6] Lageweg B.J., Lenstra J.E., Rinnooy Kan A.H.G. "Minimizing maximum lateness on one machine: computational experience and some applications", Statist. Neerlandica 30, 1976, str. 25-41.

[7] McMahon G., Florian M., "On scheduling with ready times and due dates to Minimize maximum lateness", Opns. Res., vol. 23, 1975, str. 475--482.

[8] Potts C.N., "Analysis of a heuristic for one machine with release dates and delivery times", Opns. Res., vol. 28, 1980, str. 1436-1441.

Recenzent: Doc.dr hab.inż. Konrad [illegible]
Wpłynęło do Redakcji do 30.03.1984r.

РАСПИСАНИЕ ЗАДАЧ В ДИСКРЕТНОМ ПРОИЗВОДСТВЕННОМ ПРОЦЕССЕ С КРИТИЧЕСКИМ ГНЕЗДОМ

Р е з ю м е

В работе представлен новый алгоритм для проблемы упорядочения задач в производственном процессе с одним критическим гнездом. Основой алгорит-

ма является идея блоков. В каждой итерации алгоритма получается некоторое субоптимальное решение и поэтому алгоритм может применяться в системах реального времени . В работе представлены результаты вычислительных экспериментов проведённых для случайно избранных примеров.

SCHEDULING JOBS IN A DISCRETE PRODUCTION PROCESS WITH BOTTLE-NECK

S u m m a r y

In the paper a new algorithm for a problem of jobs scheduling in a production process with a single bottle-neck is proposed. The algorithm is based on a block idea and can be applied in on-line control systems. It is possible because it yields a suboptimal solution in each iteration. The results of numerical experiments for a large number of test problems generated from uniform distributions are also presented.