ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2002
Seria: AUTOMATYKA z. 134 Nr kol. 1554

Adam JANIAK, Tomasz KRYSIAK, Maciej LICHTENSTEIN
Politechnika Wrocławska

SZEREGOWANIE ZADAŃ Z PRZEZBROJENIAMI I CZASAMI WYKONYWANIA ZADAŃ ZALEŻNYMI OD ZASOBÓW - MINIMALIZACJA SUMY WAŻONYCH CZASÓW ZAKOŃCZENIA WYKONYWANIA ZADAŃ

Streszczenie. W pracy rozpatrzono dwa jednomaszynowe problemy szeregowania przy kryterium minimalizacji sumy ważonych czasów zakończenia wykonywania zadań oraz ograniczeniu na całkowitą ilość dostępnego zasobu. W pierwszym problemie czasy przezbrojeń są zależne od ilości przydzielonego zasobu, występuje tutaj także wymóg technologii grupowej. W drugim problemie czasy wykonywania zadań są zależne od zasobu, natomiast przezbrojenia są równe zero. Wykazano równoważność obu problemów dla malejącej funkcji liniowej opisującej zależność czasu przezbrojenia oraz czasu wykonywania od zasobu. Dla rozpatrywanych problemów wykazano szereg własności określających optymalne rozwiązania ich szczególnych przypadków oraz własności pomocnicze wykorzystane przy konstrukcji algorytmów heurystycznych rozwiązujących te problemy.

SCHEDULING JOBS WITH RESOURCE DEPENDENT SETUP AND PROCESSING TIMES – TOTAL WEIGHTED COMPLETION TIME MINIMIZATION

Summary. The paper deals with two single machine scheduling problems considered for the sum of weighted completion times minimization. In the first problem, the setup times are resource-dependent and there is group technology restriction. In the second one the processing times are resource-dependent and the setup times are equal to zero. It is shown that both problems are equivalent if both the setup and processing times are decreasing resource dependent linear functions. We prove some properties for the considered problems, based on which some special cases of the considered problem can be solved optimally. These properties are also used to construct some heuristic algorithms solving the general cases of the problems under considerations. The efficiency of the heuristic algorithms is tested experimentally.

1. Wprowadzenie

Optymalizacja procesu produkcyjnego jest bardzo często związana z przydziałem pewnego ograniczonego zasobu do wykonywanych zadań lub maszyn. Zasób może mieć charakter ciągły (np. paliwo, energia), lub dyskretny (np. dodatkowe narzędzia, pracownicy). Sposób rozdziału zasobu wpływa na wartość kryterium charakteryzującego jakość optymalizowanego procesu, dlatego też znaczącą rolę wśród problemów szeregowania zadań pełnią problemy z rozdziałem zasobów. W ostatnich latach problemy te były często rozważane w literaturze naukowej. W pracy [2] przedstawiono szeroki przegląd problemów z rozdziałem zasobów. Opisano tam jednomaszynowe i wielomaszynowe problemy, w których czasy wykonywania zadań, czasy przezbrojeń lub terminy dostępności zadań zależą od dodatkowego zasobu. Przykłady procesów, w których mamy do czynienia z problemem rozdziału zasobów, można znaleźć opisując np. proces produkcji elementów plastikowych [3] czy też proces elektrorafinacji miedzi [2].

W niniejszej pracy rozważane są dwa jednomaszynowe problemy z rozdziałem zasobów. W pierwszym z nich liniowa zależność od dodatkowego zasobu charakteryzuje czasy przezbrojeń, natomiast w drugim czasy wykonywania zadań. Celem obu problemów jest minimalizacja sumy ważonych czasów zakończenia wykonywania zadań.

Dalsza część pracy jest zorganizowana następująco. Rozdział 2 zawiera sformułowania dwóch rozważanych w pracy problemów. W rozdziale 3 wykazano, że oba problemy są sobie równoważne. W rozdziale 4 zamieszczono własności określające optymalne rozwiązania dla szczególnych przypadków badanych problemów oraz własności pomocnicze wykorzystane przy konstrukcji algorytmów heurystycznych rozwiązujących te problemy. W rozdziale 5 opisano skonstruowane algorytmy, natomiast w rozdziale 6 dokonano eksperymentalnej analizy ich efektywności. Rozdział 7 zawiera krótkie podsumowanie.

2. Sformułowanie problemów

Dany jest zbiór n niepodzielnych i niezależnych zadań dostępnych w chwili $t=0$, które mają być wykonane na pojedynczej maszynie. Każde zadanie jest opisane przez wagę $w_j>0$ oraz czas wykonywania $p_j(u_j)$. Czas wykonywania zadania p_j dany jest jako nierosnąca liniowa funkcja zależna od ilości przydzielonego zasobu u_j, tj.

$p_j(u_j) = b_j - a_j u_j$, gdzie $b_j>0$ oraz $a_j>0$ są zadanymi parametrami modelu. Ze względów technologicznych przyjmuje się, że $0 \le u_j \le \bar{u}_j$, natomiast w celu zapewnienia dodatniej wartości czasu wykonywania zadania zakłada się, że $0 \le \bar{u}_j \le b_j / a_j$, gdzie $\bar{u}_j$ jest maksymalną ilością zasobu, jaką można przydzielić do zadania *j*. Zakłada się również, że istnieje ograniczenie na całkowitą ilość dostępnego zasobu $\sum_{j=1}^{n} u_j \le \hat{R}$. Każde zadanie należy do jednej rodziny *f*, $f = 1,\ldots,B$. Zadania z tej samej rodziny są wykonywane łącznie, tzn. istnieje wymóg technologii grupowej (GT). Wykonanie pierwszego zadania z danej rodziny wymaga wcześniejszego przygotowania (przezbrojenia) maszyny. W związku z tym, dla każdej rodziny *f* określa się czas przezbrojenia maszyny s_f jako nierosnącą liniową funkcję zależną od ilości przydzielonego zasobu v_f, tj. $s_f(v_f) = g_f - h_f v_f$, gdzie $g_f>0$ oraz hf>0 są zadanymi parametrami modelu. Ponadto, ze względów technologicznych, zachodzą następujące warunki: $0 \le v_f \le \bar{v}_f$, $\bar{v}_f \le g_f / h_f$, $0 \le \sum_{f=1}^{B} v_f \le \hat{Z} \le \sum_{f=1}^{B} \bar{v}_f$, gdzie $\bar{v}_f$ oraz $\hat{Z}$ oznaczają odpowiednio maksymalną ilość zasobu, jaką można przydzielić do przezbrojenia s_f oraz całkowitą ilość zasobu dostępną do realizacji przezbrojeń zadań.

Problem polega na znalezieniu kolejności wykonywania rodzin, kolejności wykonywania zadań w obrębie tych rodzin oraz takiego rozdziału zasobu, które minimalizują wartość sumy ważonych czasów zakończenia wykonywania zadań.

W niniejszej pracy rozpatrywane będą dwa problemy, które w notacji trójpolowej [1] są opisane następująco:

$$1 \mid s_f(v_f) = g_f - h_f v_f, GT, \sum v_j \le \hat{Z} \mid \sum w_j C_j \tag{1}$$

$$1 \mid p_j(u_j) = b_j - a_j u_j, \sum u_j \le \hat{R} \mid \sum w_j C_j . \tag{2}$$

Zakłada się przy tym, że w problemie (1) czasy wykonywania zadań są z góry zadanymi wartościami i nie zależą od ilości przydzielonego zasobu, natomiast w problemie (2) liczba rodzin jest równa liczbie zadań, a przezbrojenia maszyny są równe zero. W obu problemach minimalizacji podlega suma ważonych czasów zakończenia wykonywania zadań.

3. Tożsamość problemów

Ze względu na to, że w problemie (1) występuje ograniczenie technologii grupowej, całe rodziny zadań – łącznie z czasem przezbrojenia – można traktować jak pojedyncze zadania. Dzięki temu można wykazać, że problem (1) jest tożsamy z problemem (2).

Twierdzenie 1. Problem $1|s_f(v_f)=g_f-h_f v_f, GT, \sum v_j \le \hat{Z} | \sum w_j C_j$ jest równoważny problemowi $1|p_j(u_j)=b_j-a_j u_j, \sum u_j \le \hat{R} | \sum w_j C_j$.

Dowód. Załóżmy, że dla problemu (1) w obrębie rodziny f zadania są uszeregowane według niemalejących wartości p_j/w_j (reguła SWPT [4]). Bez straty ogólności możemy założyć, że kolejność uszeregowania zadań w obrębie rodziny f odpowiada permutacji naturalnej, tj. $1,2,\ldots,n_f$. Załóżmy ponadto, że maszyna rozpoczyna wykonywać zadania należące do rodziny f w momencie S_f. Wykonanie wszystkich zadań z rodziny f spowoduje, że wartość kryterium wzrośnie o $W_f=\sum_{j=1}^{n_f} w_j C_j = \sum_{j=1}^{n_f} w_j \left(S_f + s_f + \sum_{i=1}^{j} p_i\right)$. Wielkość W_f można przekształcić następująco: $W_f = \left(S_f + s_f + \sum_{i=1}^{n_f} p_i\right) w'_f - A_f$, gdzie $w'_f = \sum_{i=1}^{n_f} w_i$ oraz $A_f = \sum_{i=2}^{n_f} p_i \sum_{j=1}^{i-1} w_j$.

Łatwo zauważyć, że wartość wyrażenia A_f zależy jedynie od uszeregowania zadań w obrębie rodziny f (z założenia są one uszeregowane wg reguły SWPT). Zatem wykonanie wszystkich zadań z rodziny f w momencie S_f jest równoważne (ze względu na wartość kryterium) wykonaniu pojedynczego zadania o następujących parametrach $p'_f = s_f + \sum_{i=1}^{n_f} p_i = g_f - h_f v_f + \sum_{i=1}^{n_f} p_i$ oraz $w'_f = \sum_{i=1}^{n_f} w_i$. Przyjmując $u_f = v_f$, $a_f = h_f$ oraz $b_f = g_f + \sum_{i=1}^{n_f} p_i$, otrzymujemy czas wykonywania zadania $p'_f = b_f - a_f u_f$. Szeregowanie B takich zadań odpowiada problemowi (2). Rozwiązanie problemu (2) jest więc równoważne rozwiązaniu problemu (1), różnią się one jedynie wartością kryterium (dokładnie o $\sum_{i=1}^{B} A_i$).

W dalszej części pracy będziemy rozpatrywali problem (2), ponieważ jest on łatwiejszy w analizie w stosunku do problemu (1). Jednak, na podstawie Twierdzenia 1, wszystkie wnioski dotyczące problemu (2) odnoszą się także do problemu (1).

4. Własności problemu

Własność 1 [2]. Dla problemu $1 \mid p_j = b_j - a_j u_j, \sum u_j \le \hat{R} \mid \sum w_j C_j$, przy ustalonej permutacji zadań, optymalny rozdział zasobu można wyznaczyć w czasie $O(n^2)$ stosując następujący algorytm:

Algorytm 1.

Krok 0: Inicjalizacja: $J := \{1,2,...,n\}$, $R := \hat{R}$.

Krok 1: Znajdź j^* takie, że: $a_{\pi(j^*)} \sum_{i=j^*}^{n} w_{\pi(i)} = \max_{j \in J} \left\{ a_{\pi(j)} \sum_{i=j}^{n} w_{\pi(i)} \right\}$.

Krok 2: Podstaw $u_{\pi(j^*)} = \min\{\overline{u}_{\pi(j^*)}, R\}$, $R := R - u_{\pi(j^*)}$ oraz $J := J \setminus \{j^*\}$.

Krok 3: Jeżeli $J = \varnothing$ lub $R = 0$, to STOP, w przeciwnym wypadku przejdź do Kroku 1.

Własność 2. Dla problemu $1 \mid p_j = b_j - a_j u_j, \sum u_j \le \hat{R} \mid \sum w_j C_j$, przy ustalonym rozdziale zasobu, uszeregowanie zadań według niemalejących wartości p_j/w_j (reguła SWPT [4]) jest optymalne.

Rozpatrywany problem przy ustalonym rozdziale zasobu sprowadza się do klasycznego problemu szeregowania zadań rozpatrywanego w [4] i rozwiązywanego za pomocą reguły SWPT.

Własność 3. W rozwiązaniu optymalnym problemu $1 \mid p_j = b_j - a_j u_j, \sum u_j \le \hat{R} \mid \sum w_j C_j$, wszystkim zadaniom (z wyjątkiem jednego) przydzielona jest maksymalna lub minimalna ilość zasobu jedno zadanie może otrzymać pośrednią ilość zasobu.

Dowód. Dowód powyższej własności wynika z zasady działania Algorytmu 1. W algorytmie tym, jeżeli tylko można, przydzielamy zadaniom maksymalną możliwą ilość zasobu, tj. $\overline{u}_j$. Zatem, tylko jedno zadanie (ostatnie spośród tych, które otrzymają zasób) może dostać wartość mniejszą, niż wspomniane $\overline{u}_j$. Z Własności 1 wynika, że Algorytm 1 jest optymalny, zatem każdy optymalny rozdział zasobu spełnia Własność 3.

Własność 4. Jeżeli $\overline{u}_j = \overline{u}$ dla $j = 1,\dots,n$ oraz $\hat{R} = k \cdot \overline{u}$ dla $k \in Z_+$, wtedy problem $1 \mid p_j = b_j - a_j u_j, \sum u_j \le \hat{R} \mid \sum w_j C_j$ można rozwiązać optymalnie w czasie $O(n^k)$.

Dowód. Z Własności 3 wynika, że w rozpatrywanym przypadku każde zadanie albo dostanie dokładnie $\bar{u}$ zasobu, albo nie dostanie go wcale. Oczywiste jest też, że zasób dostanie dokładnie k zadań z n. Liczba wszystkich możliwych kombinacji rozdziału zasobu wynosi $\binom{n}{k}$, a złożoność obliczeniowa procedury sprawdzającej te kombinacje wynosi $O(n^k)$.

5. Algorytmy

W celu rozwiązania problemu $1 \mid p_j = b_j - a_j u_j, \sum u_j \le \hat{R} \mid \sum w_j C_j$ skonstruowano trzy algorytmy heurystyczne H_1, H_2 oraz H_3 wykorzystujące własności wykazane w poprzednim rozdziale. Zasadę działania algorytmu H_1 można przedstawić następująco: przydzielaj maksymalną możliwą ilość zasobu do zadań (Własność 3) oraz przez cały czas działania algorytmu utrzymuj warunek uszeregowania zadań wg reguły SWPT (Własność 2). Jak łatwo zauważyć, przydział zasobu $u_{\pi(j)}$ do zadania $\pi(j)$ może spowodować zaburzenie reguły SWPT. Wobec tego zachowanie reguły SWPT wymaga przesunięcia zadania $\pi(j)$ na odpowiednią pozycję, dla której reguła SWPT jest nadal spełniona. Wykorzystując regułę SWPT, można dla każdego zadania wyznaczyć progową wartość zasobu $u^k_{\pi(j)}$, przydzielenie której spowoduje, że zadanie $\pi(j)$ zostanie przesunięte w permutacji π o k pozycji w lewo $u^k_{\pi(j)} = \frac{b_{\pi(j)}}{a_{\pi(j)}} - \frac{w_{\pi(j)}}{a_{\pi(j)}} \frac{p_{\pi(j-k)}}{w_{\pi(j-k)}}$. Jeżeli zadanie $\pi(j)$, otrzymując u zasobu, zostanie przesunięte o k pozycji w lewo, to wartość kryterium zmniejszy się o wartość $z^k_{\pi(j)}(u) = u a_{\pi(j)} \sum_{i=j-k}^{n} w_{\pi(i)} + w_{\pi(j)} \sum_{i=1}^{k} p_{\pi(j-i)} - p_{\pi(j)} \sum_{i=1}^{k} w_{\pi(j-i)}$. W algorytmie H_1 powyższe rozważania wykorzystano w taki sposób, że w danej iteracji algorytmu wybieramy takie zadanie j oraz przydzielamy mu taką ilość zasobu u, dla których wartość wyrażenia $z_j(u)/u$ jest największa. Jednocześnie modyfikujemy permutację tak, aby spełniona była reguła SWPT. Formalny opis algorytmu H_1 jest dany poniżej:

Algorytm H_1

Krok 0: Podstaw: $u_j = 0$ dla $j = 1,2,...,n$. Uszereguj zadania według reguły SWPT.

Krok 1: Dla uzyskanej permutacji wyznacz wartość funkcji celu.

Krok 2: Wyznacz progowe wartości zasobów:

$$u^k_{\pi(j)} = \frac{b_{\pi(j)}}{a_{\pi(j)}} - \frac{w_{\pi(j)}}{a_{\pi(j)}} \frac{p_{\pi(j-k)}}{w_{\pi(j-k)}} \text{ dla } j = 1,2,...,n;\ k = 1,2,...,j-1.$$

Krok 3: Wyznacz zysk w wartości funkcji celu po przydzieleniu zadaniu $\pi(j)$ zasobu u:

$$z_{\pi(j)}(u) = \begin{cases} z^k_{\pi(j)} & gdy \quad u \in \left[u^k_{\pi(j)}; u^{k+1}_{\pi(j)}\right) \\ 0 & gdy \quad u > \overline{u}_{\pi(j)} \end{cases},$$

gdzie $z^k_{\pi(j)}(u) = ua_{\pi(j)} \sum_{i=j-k}^{n} w_{\pi(i)} + w_{\pi(j)} \sum_{i=1}^{k} p_{\pi(j-i)} - p_{\pi(j)} \sum_{i=1}^{k} w_{\pi(j-i)}$.

Krok 4: Wyznacz $u^* = \arg\max_{0 \le u \le \hat{R}} \left\{ \max_{1 \le j \le n} \frac{z_j(u)}{u} \right\}$, j^* oraz k^*.

Krok 5: Jeżeli $u^* = \overline{u}_{\pi(j^*)}$, zadanie $\pi(j^*)$ dostaje u^* zasobu, w przeciwnym przypadku szukamy zadania, któremu można przydzielić u^* zasobu i przyniesie ono największy zysk – zadanie takie oznaczamy jako j^*. Przesuń zadanie j^* o k^* pozycji w permutacji w lewo. Wartość funkcji celu maleje o $z^{k^*}_{\pi(j^*)}(u^*)$.

Krok 6: $\hat{R} := \hat{R} - u^*$.

Krok 7: Jeżeli $\hat{R} > 0$, to przejdź do Kroku 2, w przeciwnym razie STOP.

Złożoność obliczeniowa poszczególnych kroków algorytmu H_1 jest następująca: Krok 0 wyznaczamy w czasie $O(n\log n)$, Krok 1 – $O(n)$, Krok 2 – $O(n^2)$, Krok 3 – $O(n\log n)$, Krok 4 – $O(n)$, Kroki 5 i 6 to $O(1)$. Kroki 2 – 6 wykonujemy co najwyżej n razy (każdemu zadaniu zasób możemy przydzielić najwyżej raz – Własność 3), a więc złożoność obliczeniowa całego algorytmu wynosi co najwyżej $O(n^3)$.

Działanie algorytmu H_2 można przedstawić następująco.

Algorytm H_2

Krok 0: Podstaw $u_j = 0$ dla $j = 1,2,...,n$.

Krok 1: Uszereguj zadania według reguły SWPT.

Krok 2: Przydziel cały dostępny zasób optymalnie według Algorytmu 1. Otrzymana permutacja po przydziale zasobu nie spełnia reguły SWPT.

Kroki 1 i 2 powtarzaj, dopóki następuje zmniejszenie wartości kryterium.

Doświadczalnie stwierdzono, że Algorytm H_2 działa najlepiej, gdy jako kryterium stopu przyjmuje się brak poprawy wartości funkcji celu w dwóch następujących po sobie iteracjach (za brak poprawy przyjmuje się zmianę wartości kryterium o nie więcej niż $\varepsilon = 0{,}0001$). Zastosowane kryterium stopu nie pozwala określić złożoności obliczeniowej powyższego algorytmu.

Dokładny opis działania algorytmu H_3 jest dany poniżej.

Algorytm H_3

Krok 1: Uszereguj zadania według reguły SWPT.

Krok 2: Dla otrzymanej permutacji przydziel zasób optymalnie według Algorytmu 1.

Złożoność obliczeniowa algorytmu H_3 wynosi $O(n^2)$.

6. Analiza eksperymentalna

W eksperymentach generowano instancje dla trzech wartości $\hat{R}$: $\hat{R}_1 \in (0;0{,}25U]$, $\hat{R}_2 \in (0;0{,}5U]$ oraz $\hat{R}_3 \in (0;0{,}75U]$, gdzie $U = \sum_{j=1}^{n} \overline{u}_j$. Dla każdej wartości $\hat{R}$ generowano instancje problemu dla n = 9, 20, 100 i 200 zadań. Wartości parametrów problemu losowano z następujących przedziałów: $a_j \in [1;10]$, $b_j \in [0;100]$, $w_j \in [1;10]$ oraz $\overline{u}_j \in \lfloor 0;b_j/a_j \rfloor$. Dla n=9 zadań losowano po 20 instancji problemu i dla każdej obliczano błąd względny $\eta_{Hj} = \frac{F_{Hj} - F_o}{F_o} 100\%$, gdzie F_{Hj} oznacza rozwiązanie dostarczone przez algorytm H_j, ($j = 1,2,3$), natomiast F_o oznacza rozwiązanie optymalne uzyskane za pomocą przeglądu zupełnego. W tablicy 1 przedstawiono uśrednione wartości błędu względnego.

Tablica 1

Uśrednione wartości błędu względnego η_{Hj} dla n=9 zadań i różnych wartości $\hat{R}$

		$\hat{R}$	$\hat{R}$	$\hat{R}$	śr.
η_{H1}	[%]	0,014	0,14	0,11	0,088
η_{H2}	[%]	0,304	1,15	0,561	0,67
η_{H3}	[%]	2,65	6,99	5,76	5,13

Dla n=9 wszystkie trzy algorytmy dostarczają rozwiązań bliskich optymalnemu, przy czym algorytmy H_1 i H_2 wydają się być wyraźnie dokładniejsze od algorytmu H_3. Ponadto, wielkość dostępnego zasobu $\hat{R}$ nie wpływa tutaj na jakość rozwiązań. Dla n>9 algorytmy H_2 i H_3 były porównywane z algorytmem H_1 według wzoru: $\rho_{Hj} = \frac{F_{H1}}{F_{Hj}} 100\%$. Dla każdej wartości n poszczególne algorytmy były uruchamiane dla 10 różnych instancji, następnie uzyskane wyniki były uśredniane. Otrzymane rezultaty przedstawiono w tablicy 2.

Tablica 2

Porównanie jakości algorytmów H_2 i H_3, w odniesieniu do algorytmu H_1, dla różnych wartości zasobu $\hat{R}$, przy $n>9$

n:		20			100			200		
		$\hat{R}$	$\hat{R}$	$\hat{R}_3$	$\hat{R}$	$\hat{R}$	$\hat{R}_3$	$\hat{R}$	$\hat{R}$	$\hat{R}_3$
ρ_{H2}	[%]	99,80	99,77	98,63	98,81	98,14	99,07	98,90	99,16	99,34
ρ_{H3}	[%]	97,92	95,49	91,14	94,74	88,59	90,13	91,83	95,61	87,62

Dla dużych rozmiarów problemu algorytm H_2 jest prawie tak dokładny, jak algorytm odniesienia H_1 (współczynnik ρ_{H2} rzędu 99%), natomiast algorytm H_3 jest zdecydowanie mniej dokładny niż algorytm H_1 (współczynnik ρ_{H3} rzędu 93%). Można więc stwierdzić, że relacja między algorytmami H_1 i H_2 a algorytmem H_3 jest podobna jak dla $n=9$, tzn. algorytmy H_1 i H_2 są dokładniejsze od algorytmu H_3. Prezentowane w tablicy 2 wyniki nie wykazują wpływu liczby zadań ani ilości dostępnego zasobu $\hat{R}$ na dokładność algorytmów.

Spośród badanych algorytmów bezkonkurencyjny pod względem czasu działania okazuje się być algorytm H_3 (czasy działania poniżej 1ms). Poza nim warto zwrócić uwagę na algorytm H_2 (średni czas działania rzędu kilkudziesięciu milisekund). Algorytm H_1 może być użyteczny dla problemów o niedużym rozmiarze (praktycznie dla $n<100$ czas działania poniżej 1s). Biorąc pod uwagę dokładność badanych algorytmów i czas ich działania, należy stwierdzić, że najbardziej uniwersalny jest algorytm H_2 – szybki i dokładny. Algorytm H_1 jest nieco dokładniejszy, ale nieefektywny pod względem czasu działania dla dużych problemów (dla $n>100$), natomiast algorytm H_3 jest bardzo szybki, ale najmniej dokładny.

7. Podsumowanie

W niniejszej pracy rozpatrywano dwa problemy szeregowania zadań z rozdziałem zasobów. W jednym z nich czasy przezbrojeń są liniowo zależne od zasobu oraz istnieje wymóg technologii grupowej. W drugim natomiast czasy wykonywania zadań zależą liniowo od zasobu. W obu przypadkach minimalizowano sumę ważonych czasów zakończenia wykonywania zadań. Wykazano, że problemy te są sobie równoważne. Skonstruowano także algorytm optymalnego rozdziału zasobu przy ustalonej permutacji zadań. Ponadto wykazano szereg własności określających wielomianowe rozwiązania optymalne dla szczególnych przypadków rozpatrywanych problemów oraz własności pozwalające skonstruować trzy efektywne obliczeniowo algorytmy heurystyczne rozwiązujące badane problemy. Na koniec zaprezentowano wyniki analizy eksperymentalnej porównującej powyższe algorytmy.

Analiza ta wykazuje, że dla badanych problemów można skonstruować efektywne obliczeniowo algorytmy heurystyczne – dostarczające w krótkim czasie wyniki bliskie optymalnym.

LITERATURA

1. Graham R.L., Lawler E.L., Lenstra J.K., Rinnooy Kann A.H.G.: Optimization and approximation in sequencing and scheduling: a survey. Annals of Discrete Mathematics, 5, 1979, pp. 287-326.
2. Janiak A.: Wybrane problemy i algorytmy szeregowania zadań i rozdziału zasobów. Akademicka Oficyna Wydawnicza PLJ, Warszawa 1999.
3. Potts C.N.: van Wassenhove L.N., Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity. Journal of the Operational Research Society, No 43, 1992, pp. 395-406.
4. Smith W.E.: Various optimizers for single-stage production. Naval Research Logistics Quarterly, 3, 1956, pp. 59-66.

Recenzent: Prof. zw. dr hab. inż. Jan Węglarz

Abstract

Two single machine scheduling problems with distribution of the resources were considered in this paper. In the first problem the setup times are resource-dependent and there is group technology. However, in the second one the processing times are resource-dependent. In both cases the sum of weighted completion times is minimized. It is shown that these problems are equivalent each other if the resource-dependence of setup time and processing time is the decreasing linear function. Manufacturing of plastic components and electro refining of copper are examples of the situation in which distribution of the resources occurs. We construct some optimal algorithms which solve special cases of the general problems in polynomial time. We also prove the properties which are helpful in constructing an effective heuristic algorithms. Finally, we show the results of an experimental analysis which compares these algorithms. The analysis shows that there is an exact heuristic algorithm which provides the solutions with relative errors less than 1% in a short time (in practice its working time do not exceed 1s, even for large quantity of jobs). It is also possible to construct two others algorithms. One of tem is more exact but slowest than the first one – it generates the solutions with relative errors less than 0.1% but it is practically unusable for problem of more than 100 jobs because of its working time. And the other algorithm is much faster – its computational complexity is $O(n^2)$ – but less exact – its relative error is a few percent.