ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994
Seria: AUTOMATYKA z.114 Nr kol. 1250

Stanisław ZDRZAŁKA
Politechnika Wrocławska

ALGORYTMY DLA NIEKTÓRYCH ZAGADNIEŃ SZEREGOWANIA ZADAŃ Z PRZEZBROJENIAMI MASZYN

Streszczenie: W pracy przedstawia się dokładne i aproksymacyjne algorytmy rozwiązywania zagadnień grupowania i porcjowania zadań dla pewnego jednomaszynowego modelu szeregowania z przezbrojeniami maszyny. Algorytmy oparte są na procedurze heurystycznej będącej uogólnieniem reguły Jacksona. Wyniki badań eksperymentalnych oraz analiza najgorszego przypadku potwierdzają ich przydatność jako praktycznych narzędzi rozwiązywania rozważanych problemów.

ALGORITHMS FOR SOME SCHEDULING PROBLEMS WITH MACHINE SETUPS

Summary: The paper presents exact and approximate algorithms for solving batching and lotsizing problems for a single-machine scheduling model with setups. The algorithms are based on a heuristic procedure which is a generalization of Jackson's rule. Results of computational experiments and the worst-case analysis show that they are practical tools for solving the problems considered.

ALGORITHMES DES CERTAINS PROBLÈMES DE MISE EN ROUTE

Resumé: Dans cet article on présente les algorithmes précis et approximatifs résolvant des problèmes du groupement et portionnement des tâches pour certain modèle d'ordonnancement avec la mise en route d'une machine. Algorithmes sont basés sur la procédure heuristique étant la généralisation de la régle de Jackson. Les résultats des expériences effectuées ainsi que l'analyse d'un cas le mains avantageux, confirment l'utilité des algorithmes proposés pour les solutions des problèmes pratiques.

1. Wstęp

W pracy przedstawiamy niektóre wyniki badań nad algorytmami optymalizacji dla zagadnień harmonogramowania z przezbrojeniami maszym. Standardowe modele harmonogramowania rozszerzone o przezbrojenia maszyn oraz elementy pozwalające ująć łącznie

aspekty kolejnościowe z aspektami grupowania lub porcjowania zadań są ostatnio przedmiotem intensywnych badań; patrz prace przeglądowe: Monma i Potts [4], Potts i Van Wassenhove [5]. Problemy z tej klasy można rozbić na następujące grupy: 1) zagadnienia, w których każde zadanie wymaga przezbrojenia maszyny, 2) zagadnienia grupowania, w których każda rodzina zadań potrzebuje innego uzbrojenia maszyny; jeżeli w harmonogramie jakieś zadanie poprzedzane jest przez inne z tej samej rodziny, wówczas nie wymaga ono przezbrojenia maszyny, 3) zagadnienia porcjowania, gdzie zadania dzielone są na porcje, a przezbrojenie występuje pomiędzy każdymi dwoma porcjami dwóch różnych zadań. W zagadnieniach grupowania wyróżnia się dodatkowo modele, w których obowiązuje "założenie technologii grupowej", to znaczy rozbijanie rodzin na grupy zadań wykonywanych oddzielnie jest zabronione, patrz np. [7]. W pracy przedstawiamy przegląd algorytmów rozwiązywania powyższych zagadnień dla dobrze znanego modelu szeregowania zadań na pojedynczej maszynie przy zadanych czasach gotowości do realizacji i zadanych czasach dostarczania zadań, w którym kryterium jest minimalny czas dostarczenia wszystkich zadań, patrz np. [2], [3], [6]. Kryterium to jest równoważne minimalizacji maksymalnej nieterminowości przy zadanych pożądanych czasach wykonania zadań.

2. Problemy jednomaszynowe

Dany jest zbiór zadań $J = \{1, 2, \ldots, n\}$, które mają być wykonane na pojedynczej maszynie o przepustowości jeden. Dla każdego zadania j dany jest *czas gotowości do realizacji* (*głowa zadania*) $r_j \geq 0$, *czas wykonywania* $p_j > 0$ oraz *czas dostarczania* (*ogon zadania*) $q_j \geq 0$; przyjmując, że C_j jest czasem zakończenia wykonywania zadania j na maszynie, $C_j + q_j$ jest czasem jego dostarczenia. Zbiór zadań J podzielony jest na B podzbiorów $I_1, \ldots, I_B$ zwanych *rodzinami*; przez $F = \{1, 2, \ldots, B\}$ oznaczamy zbiór indeksów rodzin, jak również zbiór rodzin, a przez n_b, liczbę zadań rodziny I_b. Z każdą rodziną I_b związany jest *czas przezbrojenia* $s_b \geq 0$ uwzględniany w dopuszczalnym harmonogramie w następujący sposób: jeżeli zadanie $j \in I_b$ wykonywane jest jako pierwsze lub bezpośrednio przed nim wykonywane jest zadanie z innej rodziny, wówczas bezpośrednio przed rozpoczęciem zadania j maszyna musi być wolna przez czas nie mniejszy niż s_b. Wygodnie jest interpretować s_b jako czas wykonywania dodatkowego zadania. Zakładamy, że *przezbrojenie maszyny związane z zadaniem j może być dokonane przed jego pojawieniem się, to znaczy przed momentem r_j.*

Wprowadźmy jeszcze następujące oznaczenia. Niech Π będzie zbiorem wszystkich permutacji zbioru J. Ciąg $\pi \in \Pi$ określa kolejność wykonywania zadań; $\pi(i)$ jest i-tym zadaniem ciągu π. Dla $\pi \in \Pi$, najwcześniejszy moment dostarczenia wszystkich zadań dany jest wzorem

$$C(\pi) = \max_{0 \le i_1 \le i_2 \le n} [r_{\pi(i_1)} + p_{\pi(i_1)} + \sum_{i=i_1+1}^{i_2} (g_{\pi,i} + p_{\pi(i)}) + q_{\pi(i_2)}], \qquad (1)$$

gdzie:

- $\pi(0) \equiv 0$ jest indeksem fikcyjnego zadania, dla którego $r_0 = p_0 = q_0 = 0$, należącego do fikcyjnej rodziny I_0 z zerowym czasem przezbrojenia, $s_0 = 0$,
- $g_{\pi,i}$ zdefiniowane jest dla $\pi \in \Pi$ i $1 \le i \le n$ następująco: $g_{\pi,i} = 0$, jeżeli $\pi(i-1)$ i $\pi(i)$ należą do tej samej rodziny; $g_{\pi,i} = s_b$, jeżeli $\pi(i-1) \in I_a$, $\pi(i) \in I_b$ oraz $a \neq b$,
- przyjmujemy, że $\Sigma_{h=k}^{l} x_h = 0$, jeżeli $k > l$.

Przy założeniu że wszystkie zadania każdej rodziny muszą być wykonywane jedno po drugim (rodziny zadań są niepodzielne), uszeregowanie dopuszczalne jest połączeniem permutacji π_b zbiorów I_b, $b \in F$, w kolejności określonej przez ϕ, permutację zbioru F. Oznaczając przez Π_b oraz Φ zbiory wszystkich, odpowiednio, π_b oraz ϕ, zbiór dopuszczalnych uszeregowań dla tego przypadku możemy zapisać następująco:

$$\Pi' = \{\pi = (\pi_{\phi(1)}, \ldots, \pi_{\phi(B)}) : \pi_{\phi(i)} \in \Pi_{\phi(i)} \text{ dla } 1 \le i \le B, \phi \in \Phi\}.$$

Podobnie jak poprzednio, $\pi_b(i)$, $1 \le i \le n_b$, jest i-tym zadaniem w π_b, zaś $\phi(i)$, $1 \le i \le B$, i-tą rodziną w ϕ. Oczywiście, $\Pi' \subset \Pi$. *Grupą* nazywać będziemy podzbiór zadań tej samej rodziny, które muszą być wykonywane jedno po drugim; wykonywanie grupy zadań wymaga co najwyżej jednego przezbrojenia maszyny. W dalszym ciągu rozpatrujemy następujące problemy:

Szeregowanie Rodzin (SR): Znajdź $\pi^* \in \Pi'$, dla którego C przyjmuje minimalną wartość na zbiorze Π'.

Grupowanie Zadań (GZ): Znajdź $\pi^* \in \Pi$, dla którego C przyjmuje minimalną wartość na zbiorze Π.

Ponieważ każde $\pi \in \Pi$ wyznacza automatycznie pewien podział rodzin na grupy, GZ jest zagadnieniem, w którym optymalizacja kolejności zadań połączona jest w naturalny sposób z szukaniem optymalnego, ze względu na to samo kryterium, podziału rodzin na grupy. W problemach SR i GZ obowiązuje założenie o niepodzielności zadań. Zakładając, że zadania są podzielne oraz przyjmując $B = n$, otrzymujemy zagadnienie (ciągłe) porcjowania zadań, w literaturze nazywane również harmonogramowaniem przez porcjowanie; każda porcja zadania j ma głowę r_j, ogon q_j i wymaga przezbrojenia trwającego s_j jednostek czasu, jeżeli tylko jest poprzedzana przez porcję innego zadania.

Porcjowanie Zadań (PZ): Założenie: $B = n$ i zadania są podzielne. Znajdź harmonogram, dla którego czas dostarczenia wszystkich zadań jest minimalny.

Ponieważ problem wyjściowy jest silnie NP-trudny, Lenstra *et al.* [3], zagadnienia SR i GZ, jako jego uogólnienia, są również silnie NP-trudne. NP-trudność problemu PZ pokazano w [8]; należy zauważyć, że specjalny przypadek tego problemu, gdy wszystkie czasy przezbrojeń są zerowe, jest rozwiązywany w wielomianowym czasie.

Ciąg $(\pi(u_1), \pi(u_1+1), \ldots, \pi(u_2))$, dla którego prawa strona (1) osiąga maksimum nazywamy dalej *ciągiem krytycznym* permutacji π; w przypadku niejednoznaczności u_1 jest wybierane jako najmniejsze z możliwych. Jeżeli $\pi = (\pi_{\phi(1)}, \ldots, \pi_{\phi(B)}) \in \Pi'$, to ciąg rodzin $(\phi(w_1), \phi(w_1+1), \ldots, \phi(w_2))$, taki że $\pi(\max\{u_1, 1\}) \in I_{\phi(w_1)}$ i $\pi(u_2) \in I_{\phi(w_2)}$, nazywany jest *ciągiem krytycznym rodzin* permutacji π. Pojęcia te, sformułowane dla SR, przenoszą się naturalnie na dowolne uszeregowanie $\pi \in \Pi$ w problemie GZ, jeżeli rodziny zadań zastąpić grupami wyznaczonymi przez π. Można je również przenieść na problem PZ traktując porcje jak zadania. Dla $V \subset J$ niech $Z(V)$ będzie zbiorem rodzin, których zadania należą do V, oraz niech $S(V) = \sum_{b \in Z(V)} s_b$ i $P(V) = \sum_{j \in V} p_j$. Każdej rodzinie I_b przyporządkowujemy *zadanie złożone (zagregowane)* b, w którym $R_b = \max\{\mathcal{R}(\pi'_b) - P(I_b) - s_b, 0\}$ jest głową, $T_b = P(I_b) + s_b$, czasem wykonywania, $Q_b = \mathcal{Q}(\pi''_b) - P(I_b)$, ogonem, gdzie dla permutacji zadań $\alpha = (\alpha(1), \ldots, \alpha(v))$

$$\mathcal{R}(\alpha) = \max_{1 \le i \le v} [r_{\alpha(i)} + \sum_{j=i}^{v} p_{\alpha(j)}], \qquad \mathcal{Q}(\alpha) = \max_{1 \le i \le v} [\sum_{j=1}^{i} p_{\alpha(j)} + q_{\alpha(i)}],$$

a π'_b i π''_b są uporządkowane według, odpowiednio, niemalejących r_j i nierosnących q_j. Zauważmy, że dla π'_b i π''_b, funkcje, odpowiednio, $\mathcal{R}$ i $\mathcal{Q}$ przyjmują wartości minimalne na Π_b. Ponieważ na ogół $\pi'_b \neq \pi''_b$, zadania złożone nie występują "fizycznie" w uszeregowaniu, lecz są wyłącznie parametrami charakteryzującymi rodziny.

Oznaczmy przez $\bar{C}^*$ minimum funkcji celu w problemie SR, a przez C^* minimum w zagadnieniu GZ. Następujące dolne ograniczenie jest prostym przeniesieniem z klasycznego zagadnienia

$$\bar{C}^* \ge C^* \ge \min_{j \in V} [\max\{r_j - S(\{j\}), 0\}] + P(V) + S(V) + \min_{j \in V} q_j \equiv H(V)$$

dla każdego $V \subset J$. W [10] pokazano, że dla każdego $W \subset F$ spełniającego warunek $|W| > 1$

$$\bar{C}^* \ge \min_{b \in W} R_b + \sum_{b \in W} T_b + \min_{b \in W} Q_b \equiv H'(W).$$

Łatwo jest się przekonać, że założenie $|W| > 1$ jest konieczne. Rzeczywiście, rozważmy przykład: $n = 2$, $B = 1$, $s_1 = 0$, $r_1 = 0$, $p_1 = 1$, $q_1 = L$, $r_2 = L$, $p_2 = 1$, $q_2 = 0$, gdzie $L > 1$. Parametry zadania złożonego 1 są następujące: $R_1 = L - 1$, $T_1 = 2$, $Q_1 = L - 1$. W rezultacie $\bar{C}^* = L + 1 < 2L = R_1 + T_1 + Q_1$.

3. Algorytm metody podziału i ograniczeń dla problemu SR

Algorytm oparty jest na procedurze heurystycznej nazywanej w literaturze rozszerzoną regułą Jacksona (EJR), [1], [2]. W naszym przypadku reguła ta stosowana jest do szeregowania zadań wewnątrz rodzin i równocześnie do szeregowania rodzin traktowanych jako zadania złożone. De facto stosujemy złożenie rozszerzonych reguł Jacksona nazywane dalej algorytmem CEJR.

procedure EJR(V, t, α)
  while $V \neq \emptyset$ **do**
    $t := \max\{t, \min_{j \in V} r_j\}$;
    wybierz i takie że $q_i = \max\{q_j : j \in V \text{ i } r_j \leq t\}$;
    $\alpha := (\alpha, i)$;
    $V := V \setminus \{i\}$;
    $t := t + p_i$;

Algorytm CEJR: *Inicjalizacja*: $W := F$; $t := 0$; $\alpha := \emptyset$;
  while $W \neq \emptyset$ **do**
    $\tau := \max\{t, \min_{b \in W} R_b\}$;
    wybierz a takie że $Q_a = \max\{Q_b : b \in W \text{ i } R_b \leq \tau\}$;
    $t := t + s_a$;
    EJR(I_a, t, α);
    $W := W \setminus \{a\}$;

Można łatwo sprawdzić, że czasowa złożoność obliczeniowa CEJR wynosi $O(n\log n + B\log B)$ oraz że $C_{\mathrm{CEJR}}/\bar{C}^* \leq 2$; ograniczenie to jest najlepsze z możliwych. Tutaj, jak również w dalszej części pracy. C_H oznacza wartość funkcji celu otrzymaną przez algorytm H. Konstrukcja algorytmu podziału i ograniczeń oparta jest na specjalnych własnościach zadań i rodzin krytycznych w uszeregowaniu wygenerowanym przez CEJR. Zadanie $\pi(c)$ jest *krytycznym zadaniem* ciągu $\pi \in \Pi'$, jeżeli: (i) $\max\{u_1, l\} \leq c < u_2$, gdzie $\pi(l) = \pi_{\phi(w_2)}(1)$ ($\pi(c)$ jest zadaniem ostatniej rodziny w ciągu krytycznym); (ii) $q_{\pi(c)} < q_{\pi(u_2)}$ oraz $q_{\pi(i)} \geq q_{\pi(u_2)}$ dla $c+1 \leq i \leq u_2$. Rodzina $\phi(e)$ jest *krytyczną rodziną* ciągu $\pi \in \Pi'$, jeżeli: (i) $w_1 \leq e < w_2$; (ii) $Q_{\phi(e)} < Q_{\phi(w_2)}$ oraz $Q_{\phi(i)} \geq Q_{\phi(w_2)}$ dla $e+1 \leq i \leq w_2$.

W pracy [10] pokazano, że permutacja $\pi = (\pi_{\phi(1)}, \ldots, \pi_{\phi(B)})$ wygenerowana przez CEJR posiada następujące własności:

- Jeżeli π nie posiada zadania krytycznego i rodziny krytycznej, to $C(\pi) = \bar{C}^*$.
- Jeżeli π ma zadanie krytyczne $\pi(c)$, to istnieje optymalne uszeregowanie, w którym $\pi(c)$ poprzedza albo następuje po wszystkich zadaniach zbioru $\{\pi(c+1), \ldots, \pi(u_2)\} \equiv K$.
- Jeżeli π nie ma zadania krytycznego, ale posiada rodzinę krytyczną $\phi(e)$, to istnieje uszeregowanie optymalne, w którym $\phi(e)$ poprzedza albo następuje po wszystkich zadaniach zbioru $\{\phi(e+1), \ldots, \phi(w_2)\} \equiv K'$.

Powyższe własności wyznaczają w naturalny sposób regułę podziału stosowaną w algorytmie. Stanowią one uogólnienie wyników uzyskanych przez Pottsa [6] i Carliera [1] dla standardowego problemu, i w rezultacie pozwalają rozszerzyć podejście zastosowane w [1] na problem SR.

Zakładamy, że ograniczenia kolejnościowe w problemie SR dane są w postaci zestawu relacji częściowego porządku $\Gamma = (\Gamma_0, \Gamma_1, \ldots, \Gamma_B)$, gdzie Γ_0 określa ograniczenia kolejnościowe w zbiorze rodzin F, natomiast Γ_b, $1 \leq b \leq B$, ograniczenia kolejnościowe wewnątrz rodziny I_b. Podział problemu odbywa się poprzez dodanie do Γ nowych ograniczeń; z każdym wierzchołkiem i drzewa rozwiązań wiążemy zatem podproblem z ograniczeniami Γ^i. Bieżący zbiór aktywnych wierzchołków oznaczamy przez N. Ponadto w opisie algorytmu stosujemy notację: jeżeli π nie ma zadania krytycznego (rodziny krytycznej), to kładziemy $c = 0$ ($e = 0$).

Algorytm A

(0) Inicjalizacja: $N := \{0\}$; $LB_0 := 0$; $UB := \infty$; $\Gamma = \emptyset$;

(1) Krok iteracyjny: Jeżeli $N = \emptyset$, to STOP. W przeciwnym przypadku znajdź w N wierzchołek i z najmniejszym dolnym ograniczeniem LB_i, wyznacz π stosując algorytm CEJR do podproblemu i, wylicz: $C(\pi)$, c, K, e, K', podstaw $UB := \min\{UB, C(\pi)\}$ oraz $N := N \setminus \{i\}$.

(1.1) Podział problemu i na podstawie $\pi(c)$: Jeżeli $c = 0$, przejdź do kroku (1.2). Jeżeli $c \neq 0$, wylicz $LB := \max\{LB_i, H(K)\}$ i wyznacz dwa nowe wierzchołki dodając do $\Gamma^i_{\phi(w_2)}$ ograniczenia kolejnościowe, odpowiednio, $\{(\pi(c), j) : j \in K\}$ i $\{(j, \pi(c)) : j \in K\}$. Dla każdego nowego wierzchołka z zmodyfikuj r_j i q_j w zadaniach należących do $I_{\phi(w_2)}$ stosując (2) i (3), wylicz $LB_z := \max\{LB, H(K \cup \{\pi(c)\})\}$ i jeżeli $LB_z < UB$, to: znajdź nowe $R_{\phi(w_2)}$, $Q_{\phi(w_2)}$ i następnie zmodyfikuj głowy i ogony pozostałych rodzin oraz zadań stosując (4) i (5), dodaj z do N. Przejdź do kroku (1).

(1.2) Podział problemu i na podstawie $\phi(e)$: Jeżeli $e = 0$ i $C(\pi) \leq LB_i$, to STOP (π *jest rozwiązaniem optymalnym*). Jeżeli $e = 0$, przejdź do kroku (1) (*zamknięcie wierzchołka i*). Jeżeli $e \neq 0$, wylicz $LB := \max\{LB_i, LB'_i\}$ i wyznacz dwa nowe wierzchołki dodając do Γ^i_0 ograniczenia kolejnościowe, odpowiednio, $\{(\phi(e), b) : b \in K'\}$ i $\{(b, \phi(e)) : b \in K'\}$. Dla każdego nowego wierzchołka z zmodyfikuj $R_{\phi(e)}$ i $Q_{\phi(e)}$ stosując (6), wylicz $LB_z := \max\{LB, H'(K' \cup \{\phi(e)\})\}$ i jeżeli $LB_z < UB$, to: zmodyfikuj głowy i ogony rodzin oraz zadań zgodnie z (4) i (5), dodaj z do N. Przejdź do kroku (1).

Ograniczenia kolejnościowe wymuszane są poprzez odpowiednią modyfikację głów i ogonów zadań oraz rodzin. W celu spełnienia ograniczeń $\{(\pi(c), j) : j \in K\}$ i $\{(j, \pi(c)) : j \in K\}$ podstawiamy, odpowiednio

$$q_{\pi(c)} := \sum_{j \in K} p_j + q_{\pi(u_2)} \quad \text{i} \quad r_{\pi(c)} := \min_{j \in K} r_j + \sum_{j \in K} p_j. \tag{2}$$

Po modyfikacji $q_{\pi(c)}$ ($r_{\pi(c)}$), ogony (głowy) zadań, które według $\Gamma_{\phi(w_2)}$ poprzedzają (następują po) $\pi(c)$, modyfikowane są zgodnie z zasadą: *jeżeli q_j (r_i) zostało powiększone oraz $(i, j) \in \Gamma_{\phi(w_2)}$, to podstaw*

$$q_i := \max\{q_i, p_j + q_j\} \qquad (\, r_j := \max\{r_j, r_i + p_i\}\,). \tag{3}$$

Dalej, jeżeli w rezultacie modyfikacji dokonanych w pierwszej fazie rośnie $Q_{\phi(w_2)}$ ($R_{\phi(w_2)}$), to ogony (głowy) rodzin, które zgodnie z Γ_0 poprzedzają (następują po) $\phi(w_2)$, modyfikowane są w następujący sposób: *jeżeli* Q_b (R_a) *zostało powiększone oraz* $(a,b) \in \Gamma_0$, *to podstaw*.

$$Q_a := \max\{Q_a, T_b + Q_b\} \qquad (\ R_b := \max\{R_b, R_a + T_a\}\ ). \tag{4}$$

W tej fazie modyfikacji, jeżeli Q_b (R_b), $b \neq \phi(w_2)$, powiększyło się o Δ, podstawiamy.

$$q_j := q_j + \Delta \qquad (\ r_j := r_j + \Delta\ ) \quad \text{dla } j \in I_b. \tag{5}$$

Podobnie ograniczenia kolejnościowe $\{(\phi(e), b) : b \in K'\}$ i $\{(b, \phi(e)) : b \in K'\}$ są implementowane przez podstawienie, odpowiednio

$$Q_{\phi(e)} := \sum_{b \in K'} T_b + Q_{\phi(w_2)} \quad \text{i} \quad R_{\phi(e)} := \min_{b \in K'} R_b + \sum_{b \in K'} T_b. \tag{6}$$

Za każdym razem kiedy modyfikowane jest $Q_{\phi(e)}$ ($R_{\phi(e)}$), ogony (głowy) rodzin, które zgodnie z Γ_0 poprzedzają (następują po) $\phi(e)$, są modyfikowane w oparciu o (4). Ogony (głowy) zadań zmodyfikowanych rodzin, włączając $\phi(e)$, przystosowywane są w oparciu o (5).

Algorytm wykorzystuje dolne ograniczenia $H(V)$ dla $V = K, K \cup \{\pi(c)\}$, oraz $H'(W)$ dla $W = K', K' \cup \{\phi(e)\}$. Ponieważ drugie z tych ograniczeń jest prawdziwe tylko dla $|W| > 1$, w wierzchołku i stosujemy dolne ograniczenie LB'_i zdefiniowane następująco:

$LB'_i = H'(K')$ dla $|K'| > 1$;
$LB'_i = \max\{H'(\{\phi(w_1), \ldots, \phi(w_2)\}),\ H'(\{\phi(w_2), \ldots, \phi(B)\}),\ H'(F)\}$
dla $|K'| = 1, w_2 < B$;
$LB'_i = \max\{H'(\{\phi(w_1), \ldots, \phi(w_2)\}),\ R_{\phi(w_2)} + T_{\phi(w_2)},\ H'(F)\}$ dla $|K'| = 1, w_2 = B$.

Eksperymenty obliczeniowe przeprowadzone dla n i B zmieniających się w zakresie, odpowiednio, od 20 do 400 i od 4 do 40 wykazały, że algorytm A można uważać za szybkie i skuteczne narzędzie do rozwiązywania problemu SR. Dla każdego n generowano łącznie 540 problemów testowych, zmieniając skokowo górne zakresy zmian poszczególnych danych; do generacji danych wykorzystywano rozkład jednostajny. Obliczenia dla problemu testowego były przerywane, gdy liczba wygenerowanych wierzchołków drzewa rozwiązań przekraczała 100. W takiej sytuacji najlepsze rozwiązanie otrzymane do momentu przerwania obliczeń było porównywane z rozwiązaniem optymalnym w następujący sposób: $(UB - \bar{C}^*)/\bar{C}^* \leq (UB - LB')/LB'$, gdzie LB' jest najmniejszym dolnym ograniczeniem na zbiorze aktywnych, w momencie przerwania, wierzchołków. Przykładowo, dla $n = 200$ uzyskano następujące wyniki: (1) w grupie obliczeń zakończonych sukcesem średnia liczba wygenerowanych wierzchołków wynosiła 4, (2) w 61 przykładach testowych obliczenia zostały przerwane, przy czym maksymalny błąd względny w tej grupie obliczeń wyniósł

1.58%. Omawiana grupa problemów testowych była, biorąc pod uwagę otrzymane wyniki, najgorsza dla badanego algorytmu.

4. Algorytmy aproksymacyjne dla problemu GZ

Problem grupowania zadań, GZ, różni się od poprzednio omawianego tym, że dopuszczalne jest w nim rozbijanie zadań na grupy. Załóżmy, że rodziny rozbite są na v ($v \geq B$) grup $L_1, \ldots, L_v$. W dalszym ciągu w odniesieniu do zbioru grup $\{L_1, \ldots, L_v\}$ używać będziemy notacji wprowadzonej dla zbioru rodzin F. W szczególności $\pi = (\pi_{\phi(1)}, \ldots, \pi_{\phi(v)})$ jest uszeregowaniem grup zadań, w którym ϕ określa porządek grup, zaś $\pi_{\phi(i)}$, $1 \leq i \leq v$, porządek zadań w grupie. Dodatkowo wprowadzamy jeszcze wielkość $\bar{r}_j = \max\{r_j - S(\{j\}), 0\}$, którą można interpretować jako rzeczywisty, bo uwzględniający czas przezbrojenia, moment gotowości zadania j do realizacji.

W [10] pokazano, że optymalne uszeregowanie rodzin dla SR, wykorzystane jako rozwiązanie przybliżone dla problemu GZ, może dać uszeregowanie trzykrotnie dłuższe od optymalnego. Nie jest to zatem podejście obiecujące. W dalszym cągu pokażemy algorytmy heurystyczne, które dzielą rodziny na grupy i następnie znajdują uszeregowanie grup, wykorzystując w tym celu jedną z technik rozwiązywania problemu SR, aproksymacyjną lub dokładną; grupy odgrywają w tym przypadku rolę rodzin. Ogólnie metoda polega na stosowaniu algorytmu szeregowania rodzin (grup) sukcesywnie, za każdym razem zmieniając bieżący podział na grupy poprzez usunięcie jednego zadania z grupy należącej do ciągu krytycznego i utworzenie nowej grupy lub dopisanie zadania do innej grupy tej samej rodziny. Podstawowy dla tej metody jest następujący algorytm aproksymacyjny.

Każdej rodzinie I_b przyporządkujmy dwa rozłączne podzbiory zadań (dwie grupy) M_b i X_b, takie że $I_b = M_b \cup X_b$, i połóżmy $W = \bigcup_{i=w_1}^{w_2} L_{\phi(i)}$ dla krytycznego ciągu grup $(\phi(w_1), \ldots, \phi(w_2))$. Niech I_0 będzie rodziną (fikcyjną) z jednym zadaniem o numerze 0.

Algorytm B

Inicjalizacja: Podstaw $M_b := I_b$ i $X_b := \emptyset$ dla $0 \leq b \leq B$; $C_{\mathrm{B}} := \infty$; $x := 0$; $a := 0$;

repeat

$M_a := M_a \setminus \{x\}$; $X_a := X_a \cup \{x\}$;

$V :=$ zbiór wszystkich niepustych grup typu M_b lub $X_b, 1 \leq b \leq B$;

znajdź $\pi = (\pi_{\phi(1)}, \ldots, \pi_{\phi(v)})$ stosując CEJR do zbioru grup V;

wyznacz $C(\pi)$, w_1, w_2 i znajdź zadanie $x \in W$, takie, że $\bar{r}_x = \min_{j \in W} \bar{r}_j$;

$a :=$ indeks rodziny, do której należy x;

$C_{\mathrm{B}} := \min\{C(\pi), C_{\mathrm{B}}\}$;

until $\bar{r}_x \geq R_{\phi(w_1)}$ **or** W zawiera przynajmniej jedną grupę typu X_b;

Ponieważ w każdym kroku iteracyjnym dokładnie jedno zadanie przesuwane jest z grupy typu M_b do grupy typu X_b, algorytm wykonuje nie więcej niż n kroków iteracyjnych, z

których każdy ma złożoność obliczeniową $O(n\log n + B\log B)$. Zatem złożoność obliczeniowa algorytmu B wynosi $O(n^2\log n + nB\log B)$. W [10] pokazano, że dla każdego problemu konkretnego

$$2C_B \leq 3C^* + 2(\max_{j\in J} q_j),$$

i że ograniczenie to jest ścisłe. Algorytm B jest podstawową precedurą dla dalszych heurez. W następnej stosowany jest najpierw do problemu pierwotnego, po czym do problemu odwróconego. Problem odwrócony otrzymujemy poprzez zamianę r_j z q_j, odwrócenie kolejności wykonywania zadań i odwrócenie kolejności przezbrojeń; przezbrojenie dokonywane jest po wykonaniu zadania. Zauważmy, że dla odwróconej wersji oszacowanie dokładności ma postać

$$2C_B \leq 3C^* + 2(\max_{j\in J} \bar{r}_j).$$

Algorytm C: Etap 1. *Podział rodzin na podstawie r_j's*: Zastosuj B do problemu ze zbiorem rodzin F. Niech V będzie zbiorem grup otrzymanych w pierwszym etapie.
Etap 2. *Podział rodzin na podstawie q_j's* : Zastosuj B do problemu odwróconego ze zbiorem rodzin V.
Wybierz najlepsze uszeregowanie.

Algorytm D: Zastosuj C, zastępując CEJR przez A.

Dla zbadania efektywności algorytmów C i D przeprowadzono eksperyment numeryczny, w którym problemy testowe były generowane według zasady zastosowanej uprzednio do testowania algorytmu metody podziału i ograniczeń A. W przypadku algorytmu D procedura A zatrzymywana była po osiągnięciu 50 wierzchołków w drzewie rozwiązań. Dla każdego problemu testowego wyliczano oszacowanie względnego błędu w następujący sposób: $(C_\alpha - LB)/LB$, α =C,D, gdzie LB wybierano jako maksymalne $H(V)$ na zbiorze odpowiednio dobranych podzbiorów V. Przykładowo, dla grupy problemów testowych z $n = 200$ otrzymano następujące wyniki: (1) średnie oszacowanie błędu względnego dla C wyniosło 4.6%, dla D, 3.7%, (2) maksymalne oszacowanie błędu względnego wyniosło 64.1% dla C i 43.2% dla D. Wyniki dla $n = 200$ był najgorsze w grupie badanych problemów testowych. Można zatem uznać obydwa algorytmy za praktyczne metody znajdowania "dobrych" uszeregowań w problemie GZ.

5. Algorytm aproksymacyjny dla problemu PZ

W problemie porcjowania zadań PZ każda rodzina ma tylko jedno zadanie. Dlatego zadania złożone przybierają postać: $R_j = \max\{r_j - s_j, 0\}$, $T_j = p_j + s_j$ oraz $Q_j = q_j$, $j \in J$.

Kiedy zadania nie są podzielone na porcje, wyrażenie (1) sprowadza się do standardowej postaci:

$$C(\pi) = \max_{1\le i_1\le i_2\le n}[R_{\pi(i_1)} + \sum_{i=i_1}^{i_2} T_{\pi(i)} + Q_{\pi(i_2)}] = R_{\pi(u_1)} + \sum_{i=u_1}^{u_2} T_{\pi(i)} + Q_{\pi(u_2)}. \qquad (7)$$

W dalszym ciągu przez $S^-(\beta)$ i $S^+(\beta)$ oznaczamy harmonogramy dla permutacji zadań β z, odpowiednio, najwcześniejszymi i najpóźniejszymi momentami rozpoczynania zadań.

Algorytm E

Etap I. *Szeregowanie bez podziału zadań na porcje*

Krok 1: Znajdź ciąg π zadań złożonych stosując EJR dla $V = J$, $t = 0$ i $\alpha = \emptyset$. Jeżeli nie istnieje zadanie krytyczne $\pi(c)$, to podstaw $C_E = C(\pi)$ i STOP (π *jest optymalnym uszeregowaniem*).

Krok 2: Wyznacz zbiory $A = \{j \in J : R_j \le Q_j, j \ne \pi(c)\}$ oraz $B = \{j \in J : R_j > Q_j, j \ne \pi(c)\}$. Znajdź ciąg zadań β kładąc najpierw zadania ze zbioru A w kolejności niemalejących R_j, następnie zadanie $\pi(c)$, i na końcu zadania ze zbioru B w kolejności nierosnących Q_j.

Etap II. *Rozbicie zadania krytycznego na porcje*

Krok 3: Jeżeli zadanie $\pi(c)$ nie należy do ciągu krytycznego $(\beta(u_1), \ldots, \beta(u_2))$ permutacji β, to podstaw $C_E = \min\{C(\pi), C(\beta)\}$ i STOP. W przeciwnym przypadku podziel $\pi(c)$ na porcje i włącz je do uszeregowania w następujący sposób. Dla $S^-(\beta)$, podstaw $D_1 :=$ łączny czas przestojów maszyny w przedziale $[R_{\pi(c)}, R_{\beta(u_1)}]$.

Jeżeli $D_1 > s_{\pi(c)}$, to podziel zadanie $\pi(c)$ na porcję j_1 z czasem wykonywania $T_{j_1} := \min\{D_1, T_{\pi(c)}\}$ oraz porcję $\pi(c)$ z czasem $T_{\pi(c)} := T_{\pi(c)} - T_{j_1} + s_{\pi(c)}$, gdy $T_{\pi(c)} > T_{j_1}$; $T_{\pi(c)} := 0$ oraz $R_{\pi(c)} := Q_{\pi(c)} := 0$, w przeciwnym przypadku. Podstaw $A := A \cup \{j_1\}$ i utwórz ciąg β' stosując regułę z Kroku 2.

Jeżeli $D_1 \le s_{\pi(c)}$, podstaw $\beta' := \beta$.

Krok 4: Jeżeli zadanie $\pi(c)$ nie należy do ciągu krytycznego $(\beta'(u_1'), \ldots, \beta(u_2'))$ permutacji β', to podstaw $C_E = \min\{C(\pi), C(\beta), C(\beta')\}$ i STOP. W przeciwnym przypadku podziel $\pi(c)$ na porcje i włącz je do uszeregowania w następujący sposób. Dla $S^+(\beta')$, podstaw $D_2 :=$ łączny czas przestojów maszyny w przedziale $[C(\beta') - Q_{\beta'(u_2')}, C(\beta') - Q_{\pi(c)}]$.

Jeżeli $D_2 > s_{\pi(c)}$, to podziel zadanie $\pi(c)$ na porcję j_2 z czasem wykonywania $T_{j_2} := \min\{D_2, T_{\pi(c)}\}$ oraz porcję $\pi(c)$ z czasem $T_{\pi(c)} := T_{\pi(c)} - T_{j_2} + s_{\pi(c)}$, gdy $T_{\pi(c)} > T_{j_2}$; $T_{\pi(c)} := 0$ oraz $R_{\pi(c)} := Q_{\pi(c)} := 0$, w przeciwnym przypadku. Podstaw $B := B \cup \{j_1\}$ i utwórz ciąg β'' stosując regułę z Kroku 2.

Jeżeli $D_2 \le s_{\pi(c)}$, podstaw $\beta'' := \beta'$.

Podstaw $C_E = \min\{C(\pi), C(\beta), C(\beta'), C(\beta'')\}$ i STOP.

Złożoność obliczeniowa algorytmu E wynosi $O(n\log n)$. W pracy [8] pokazano, że dla każdego problemu konkretnego

$$\frac{C_E}{C^*} \leq \frac{3}{2},$$

i że ograniczenie to jest ścisle.

LITERATURA:

[1] Carlier J.: The one-machine sequencing problem. European J. Oper. Res. 11, 42-47, 1982.

[2] Jackson J.R.: Scheduling a production line to minimize maximum tardiness. Research Report 43, Mgmt. Sci. Research Project, UCLA, 1955.

[3] Lenstra J.R., Rinnooy Kan A.H.G. and Brucker P.: Complexity of machine scheduling problems. Ann. Discrete Math. 1, 343-363, 1977.

[4] Monma C.L., Potts C.N.: On the complexity of scheduling with batch set-up times. Operations Research 37, 798-804, 1989

[5] Potts C.N., Van Wassenhove L.N.: Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity. J. Opl Res. Soc. 43, 395-406, 1992

[6] Potts C.N.: Analysis of heuristic for one machine sequencing with release dates and delivery times. Operations Research 28, 1436-1441, 1980.

[7] Sekiguchi Y.: Optimal schedule in a GT-type flow-shop under series-parallel precedence constraints. J. Oper. Res. Society of Japan 26, 226-251, 1983.

[8] Zdrzałka S.: Preemptive scheduling with release dates, delivery times and sequence independent setup times. Technical University of Wroclaw, Institute of Engineering Cybernetics, Wroclaw, (to appear in European J. Oper. Res.) 1992.

[9] Zdrzalka S.: Approximation algorithms for single machine sequencing with delivery times and unit batch set-up times. European J. Oper. Res. 51, 199-209, 1991.

[10] Zdrzałka S.: A sequencig problem with family setup times. Technical University of Wrocław, Institute of Engineering Cybernetics, Wroclaw, 1993.

Recenzent: Dr inż. Maciej Drozdowski

Wpłynęło do Redakcji do 30.04.1994 r.

Abstract

In many practical scheduling problems the set of jobs to be scheduled consists of many families requiring a machine setup between two consecutively scheduled jobs from two different families; no setups are needed between jobs in the same family. Also, in practical scheduling problems, setups are required between two consecutively scheduled sublots of different jobs (lots). In these problems, scheduling is combined with batching or lotsizing decisions. This paper discusses batching and lotsizing problems for the

well-known single-machine model of minimizing the makespan subject to given job release and delivery times. Basing on the concept of a composite job, a generalized Jackson's rule for a family sequencing problem is proposed and some results of its analysis are given; in the family sequencing problem, families cannot be divided into batches. This procedure forms a basis for a branch and bound algorithm for solving the
family sequencing problem, and then it is used as a basic tool in heuristic approaches to solving the batching version of our problem. Results of the worst-case analysis and computational experiments are given. We also propose an approximate algorithm for the lotsizing problem with the worst-case performance ratio of 3/2.