Jarosław PEMPERA
Politechnika Wrocławska

DOKŁADNY ALGORYTM BLOKOWY DLA PROBLEMU RPQ Z BRAKIEM PRZESTOJU MASZYNY

Streszczenie. W pracy rozważany jest jednomaszynowy problem szeregowania zadań, w którym wszystkie zadania mają określony moment dostępności i czas dostawy. W badanym wariancie, nie jest dozwolony przestój na maszynie. Celem optymalizacji jest wyznaczenie kolejności wykonywania zadań minimalizującej moment dostarczenia wszystkich zadań. Zaprezentowano szereg właściwości problemu, w szczególności sformułowano własności blokowe dla ograniczenia bez przestoju. Zaproponowano algorytm dokładny oparty na metodzie B&B oparty na własnościach blokowych.

AN EXACT BLOCK ALGORITHM FOR NO-IDLE RPQ PROBLEM

Summary. We consider an one machine scheduling problem in which all jobs have a release and delivery time. This problem is well known as a RPQ problem. In the studied variant, no idle time is allowed on machine. The objective is to determine an optimal sequence of job to minimize makespan. We present a several properties of the problem, especially we formulate block properties for no-idle constraint. Finally, we propose an exact Branch-and-Bound algorithm based on the block properties.

1. Wstęp

W jednomaszynowym problemie z terminami dostępności oraz czasami dostarczenia zadań (RPQ) należy wykonać określoną liczbę zadań na jednej maszynie. Brak możliwości przestoju maszyny oznacza, że maszyna od momentu rozpoczęcia wykonywania pierwszego zadania musi wykonywać następne zadania bez przerw aż do ukończenia zadania ostatniego. Należy wyznaczyć harmonogram wykonywania zadań minimalizujący moment dostarczenia wszystkich zadań.

Ograniczenie bez przestoju maszyny występuje w systemach produkcyjnych, w których występują procesy termiczne lub chemiczne. W takich systemach postoje maszyn mogą być przyczyną awarii maszyn, pogorszenia jakości produktów lub generować dodatkowe koszty związane z wydatkami energetycznymi. Ograniczenie bez przestoju może być również wykorzystane w planowaniu usług outsoursingowych, w których koszty uzależnione są od długości okresu umowy lub w przypadku wypożyczania drogich urządzeń na okres realizacji zleceń.

Problemy RPQ oraz RPQ z ograniczeniem bez przestoju maszyny (RPQ no-idle) są problemami NP-trudnym. W przypadku problemu RPQ istnieją efektywne algoryt-

my oparte na metodzie podziału i ograniczeń (B&B) zaproponowane przez Calriera [1] oraz Grabowskiego i innych [4]. W pierwszym algorytmie węzeł w drzewie przeszukiwań metody B&B tworzony jest w czasie $O(n \ln n)$ przy użyciu algorytmu Schage [10], natomiast w drugim algorytmie węzły tej metody tworzone są w oparciu o pojęcie bloku zadań. Ze względu na krótki czas działania tych algorytmów (kilka sekund dla 50–10000 zadań), można je wykorzystać do konstrukcji dolnych oszacowań dla problemów wielomaszynowych z kryterium minimalizacji czasu zakończenia wszystkich zadań: problemy przepływowe, gniazdowe, elastyczne przepływowe i gniazdowe.

Projektowanie algorytmów dla prostych problemów szeregowania ma istotne znaczenie badawcze, ponieważ własności tych problemów mogą być wykorzystane w konstrukcji algorytmów dokładnych i heurystycznych dla wielomaszynowych problemów szeregowania zadań. Odkryte przez Grabowskiego i innych własności blokowe zostały z powodzeniem wykorzystane do konstrukcji algorytmów dokładnych dla problemów przepływowego i gniazdowego (rozwiązania dokładne dla instancji o małych rozmiarach) oraz w konstrukcji jednych z najefektywniejszych algorytmów heurystycznych dla problemów: przepływowego [3, 7], gniazdowego [8], przepływowego z ograniczeniami magazynowania [9] itp.

Badania dotyczące szeregowania zadań z ograniczeniami bez przestoju maszyn głównie dotyczą problemu przepływowego. Opracowano algorytmy dokładne [11] oraz algorytmy heurystyczne oparte na metodach konstrukcyjnych [2], [5] oraz przeszukiwania przestrzeni rozwiązań [6].

W pracy przedstawiono nowe własności problemu z brakiem przestoju maszyny. Ze względu na istotne podobieństwo do własności blokowych problemu RPQ będziemy je również nazywali blokowymi. W oparciu o te własności zaproponowano algorytm dokładny oparty na metodzie B&B.

2. Opis problemu

Na maszynie należy wykonać n zadań ze zbioru $J = \{1, \ldots, n\}$. Każde zadanie ma znany czas wykonania $p_j > 0$, $j \in J$ i nie może rozpocząć się wcześniej niż w momencie jego dostępności $r_j \geqslant 0$. Maszyna nie może wykonywać więcej niż jedno zadanie w danej chwili. Zadania na maszynie muszą być wykonywane bez przerw. Po zakończeniu wykonywania zadania gotowy produkt dostarczany jest do klienta przez czas $q_j \geqslant 0$. Niech S_j, C_j oraz D_j, będą odpowiednio momentem rozpoczęcia, momentem zakończenia oraz momentem dostarczenia zadania j, $j \in J$. Należy wyznaczyć harmonogram wykonywania zadań minimalizujący moment dostarczenia wszystkich zadań.

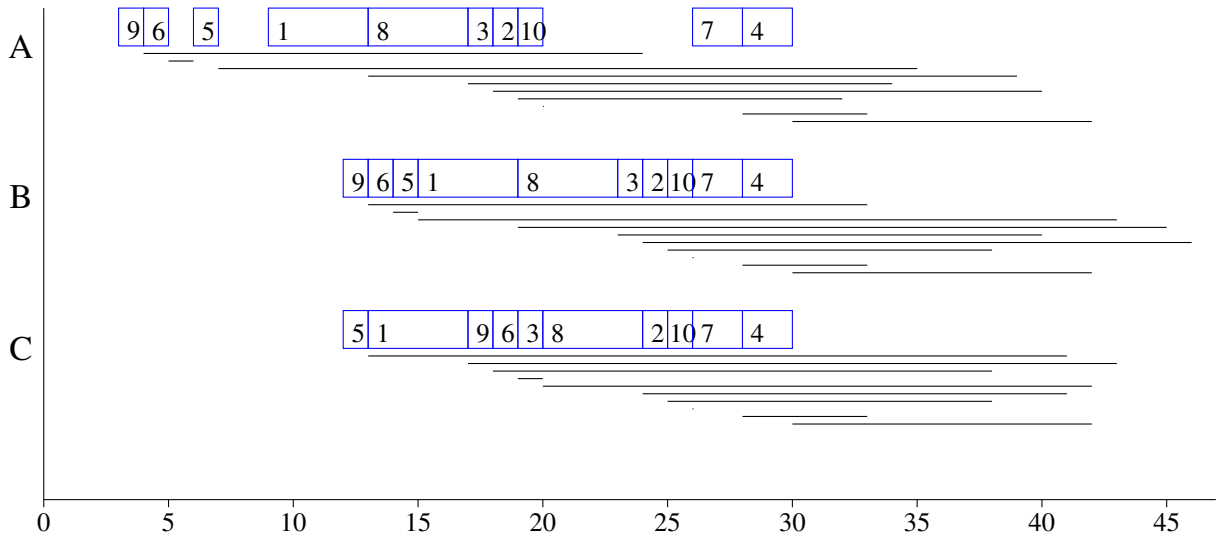
Niech π będzie permutacją zdefiniowaną na zbiorze J opisującą kolejność wykonywania zadań na maszynie. Dopuszczalny harmonogram wykonywania zadań w kolejności π musi spełniać następujące ograniczenia:

$$S_j \geqslant r_j \quad \text{dla } j \in J, \tag{1}$$

$$C_j = S_j + p_j \quad \text{dla } j \in J, \tag{2}$$

$$D_j = C_j + q_j \quad \text{dla } j \in J, \tag{3}$$

$$S_{\pi(j)} = C_{\pi(j-1)} \quad \text{dla } j = 2, \ldots, n. \tag{4}$$



Rys. 1. Wykres Gantta

Ograniczenia (1–3) są oczywiste, natomiast równanie (4) narzuca wykonywanie zadań na maszynie w kolejności π bez przerw.

Niech $D_{\max} = \max_{j \in J} D_j$ będzie momentem dostarczenia wszystkich zadań. Chcemy znaleźć kolejność wykonywania zadań na maszynie π^*, taką że

$$D_{\max}(\pi^*) = \min_{\pi \in \Pi} D_{\max}(\pi), \tag{5}$$

gdzie Π jest zbiorem wszystkich permutacji określonych na zbiorze J.

2.1. Przykład

Na maszynie należy wykonać $n = 10$ zadań. Terminy dostępności, czasy wykonania oraz czasy dostarczenia zadań zebrano w tabeli 1. Na rysunku 1 przedstawiono trzy harmonogramy wykonywania zadań w postaci wykresu Gantta. Harmonogramy (A) i (B) zostały wyznaczone dla permutacji wygenerowanej algorytmem Schrage (A-RPQ, B-RPQ no-idle), natomiast harmonogram (C) jest harmonogramem optymalnym dla problemu RPQ no-idle. Moment dostarczenia wszystkich zadań wynosi: (A)–42, (B)–46, (C)–43.

Tabela 1

Dane testowe

Parametr	1	2	3	4	5	6	7	8	9	10
termin dostępności – r_i	9	16	15	27	6	4	26	12	3	11
czas wykonania – p_i	4	1	1	2	1	1	2	4	1	1
czas dostarczenia –q_i	26	13	22	12	28	1	5	17	20	0

3. Model grafowy

Przed przystąpieniem do prezentacji modelu grafowego zauważmy, że równanie (4) możemy zastąpić przez dwie nierówności:

$$S_{\pi(j)} \geqslant S_{\pi(j-1)} + p_{\pi(j-1)} \quad \text{dla } j = 2, \ldots, n, \tag{6}$$

$$S_{\pi(j-1)} \geqslant S_{\pi(j)} - p_{\pi(j-1)} \quad \text{dla } j = 2, \ldots, n. \tag{7}$$

Dla permutacji π definiujemy graf $G(\pi) = (V, E(\pi))$ ze zbiorem węzłów V oraz zbiorem skierowanych łuków $E(\pi)$. Zbiór węzłów $V = J \cup \{s, t\}$ składa się z n węzłów reprezentujących zadania oraz dwóch fikcyjnych węzłów s (początkowy, źródłowy) i t (końcowy). Węzeł reprezentujący zadanie j obciążony jest wagą równą czasowi trwania zadania p_j, $j \in J$, natomiast węzły fikcyjne wagą równą zero.

Zbiór łuków $E(\pi)$ składa się z czterech podzbiorów łuków:

$$E^s = \{(s, j) : j \in J\}, \tag{8}$$

łuk $(s, j) \in E^s$ odpowiada ograniczeniu (1) i jest obciążony wagą r_j,

$$E^t = \{(j, t) : j \in J\}, \tag{9}$$

łuk $(j, t) \in E^t$ odpowiada ograniczeniu (3) i jest obciążony wagą q_j,

$$E^1(\pi) = \{(\pi(j-1), \pi(j)) : j = 2, \ldots n\}, \tag{10}$$

łuk $(\pi(j-1), \pi(j)) \in E^1(\pi)$ odpowiada ograniczeniu (6) i jest obciążony jest wagą 0. Łuki ze zbioru $E^1(\pi)$ będziemy nazywali łukami wynikającymi z kolejności wykonywania zadań na maszynie lub krótko łukami maszynowymi. Ostatnim podzbiorem są łuki wynikające z ograniczenia bez przestoju

$$E^2(\pi) = \{(\pi(j), \pi(j-1)) : j = 2, \ldots n\}, \tag{11}$$

łuk $(\pi(j), \pi(j-1)) \in E^2(\pi)$ odpowiada ograniczeniu (7) i jest obciążony jest wagą $-p_{\pi(j)} - p_{\pi(j-1)}$, co wynika z przekształcenia (7) do postaci

$$S_{\pi(j-1)} \geqslant S_{\pi(j)} + p_{\pi(j)} - p_{\pi(j)} - p_{\pi(j-1)} \quad \text{dla } j = 2, \ldots, n. \tag{12}$$

Własność 1. *Dla zadanej kolejności π wykonywania zadań na maszynie, najwcześniejszy moment rozpoczęcia wykonywania zadania j, $j \in J$ jest równy długości najdłuższej drogi w grafie $G(\pi)$ do węzła reprezentującego to zadanie (bez obciążenia tego węzła).*

Własność 2. *Dla zadanej kolejności π, najwcześniejszy moment dostarczenia wszystkich zadań jest równy długości najdłuższej drogi do węzła t w grafie $G(\pi)$.*

Z konstrukcji grafu $G(\pi)$ wynika, że każda droga rozpoczyna się w węźle źródłowym s, co więcej, najdłuższa droga w grafie $G(\pi)$ do węzła t jest również najdłuższą drogą w tym grafie. Drogę tą będziemy nazywali *ścieżką krytyczną*.

W skład zbioru łuków grafu $G(\pi)$ wchodzą łuki z ujemnymi wagami, zatem wyznaczenie długości najdłuższych dróg wymaga użycia algorytmu Bellmana-Forda o złożności $|V| \cdot |E|$.

Własność 3. *Długości najdłuższych dróg w $G(\pi)$ można wyznaczyć w czasie $O(n)$.*

Dla dowodu wystarczy zauważyć, że długości najdłuższych dróg mają ustaloną wartość po wykonaniu jednej relaksacji łuków w następującej kolejności: (*i*) łuki ze zbioru E^s, (*ii*) łuki ze zbioru $E^1(\pi)$ w kolejności $(\pi(1),\pi(2)),\ldots,(\pi(n-1),\pi(n))$, (*iii*) łuki ze zbioru $E^2(\pi)$ w kolejności $(\pi(n),\pi(n-1)),\ldots,(\pi(2),\pi(1))$, (*iv*) łuki ze zbioru E^t.

Dowolna ścieżka krytyczna w grafie $G(\pi)$ rozpoczyna się łukiem należącym do E^s i kończy łukiem należącym do E^t. Z tego względu opisywać ją będziemy parą (a,b), gdzie $(s,a)\in E^s$ i $(b,t)\in E^t$. Dla $a=b$ rozwiązanie π jest rozwiązaniem optymalnym. Rozpatrzmy dwa przypadki: (*i*) $a<b$ oraz (*ii*) $a>b$.

W przypadku (*i*) ciąg $B_{a,b}=(\pi(a),\pi(a+1),\ldots,\pi(b))$ będziemy nazywali *blokiem zadań*. Twierdzenie blokowe [4] można sformułować w następujący sposób:

Własność 4. *Jeśli $B_{a,b}$ jest blokiem w π oraz $\beta\in\Pi$ jest dowolną permutacją taką, że $D_{\max}(\beta)<D_{\max}(\pi)$, to istnieje zadanie $\pi(k)\in\{\pi(a+1),\ldots,\pi(b)\}$ (lub $\pi(k)\in\{\pi(a),\ldots,\pi(b-1)\}$) takie, że w permutacji β zadanie $\pi(k)$ poprzedza wszystkie pozostałe zadania w bloku (występuje za wszystkimi pozostałymi zadaniami w bloku).*

W przypadku (*ii*) ciąg $B_{a,b}=(\pi(a),\ldots,\pi(n),\pi(1),\ldots,\pi(b))$ będziemy nazywali *i-blokiem zadań*. Można udowodnić, w sposób analogiczny do Własności 4., następującą Własność dla *i*-bloku $B_{a,b}$:

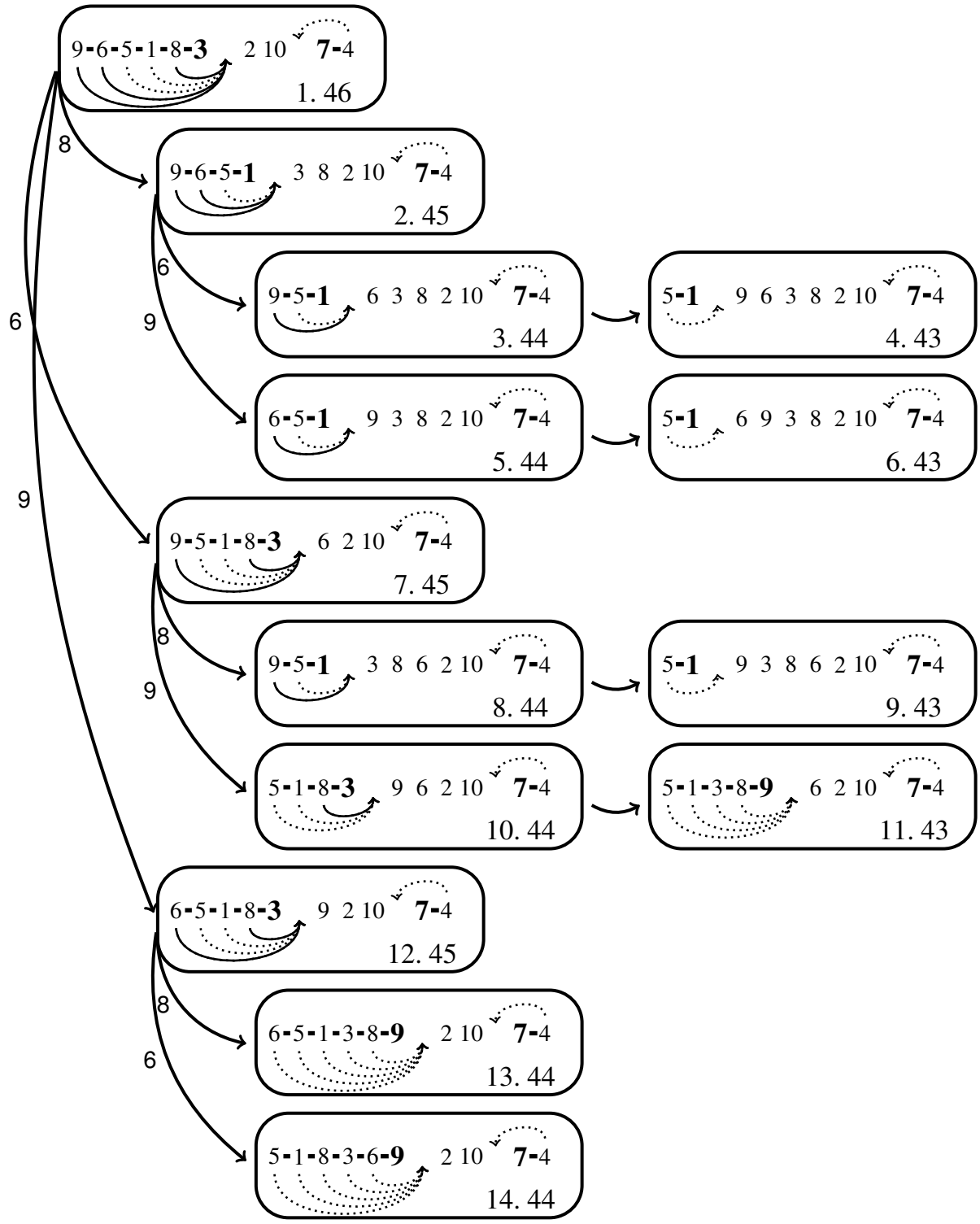
Własność 5. *Jeśli $B_{a,b}$ jest i-blokiem w π oraz $\beta\in\Pi$ jest dowolną permutacją taką, że $D_{\max}(\beta)<D_{\max}(\pi)$, to istnieje zadanie $\pi(k)\in\{\pi(a+1),\ldots,\pi(n)\}$ (lub $\pi(k)\in\{\pi(1),\ldots,\pi(b-1)\}$) takie, że w permutacji β zadanie $\pi(k)$ poprzedza wszystkie pozostałe zadania ze zbioru $\{\pi(a),\ldots,\pi(n)\}$ (występuje za wszystkimi pozostałymi zadaniami ze zbioru $\{\pi(1),\ldots,\pi(b)\}$).*

4. Dokładny algorytm blokowy

Z każdym węzłem metody podziału i ograniczeń skojarzona jest permutacja określają kolejność wykonywania zadań π, blok albo *i*-blok zadań $B_{a,b}$ oraz relacja częściowego porządku R. Relacja R implikuje ograniczenie "jeśli $(i,j)\in R$ to zadanie j może rozpocząć się dopiero po zakończeniu wykonywania zadania i". Podział przestrzeni w metodzie odbywa się na podstawie zadań znajdujących się w bloku (albo *i*-bloku) $B_{a,b}$.

Nowy węzeł drzewa przeszukiwań algorytmu powstaje przez wygenerowanie nowej permutacji oraz dodanie ograniczeń do relacji częściowego porządku R. Dla $k=a+1,\ldots,x$ ($x=b$ dla bloku oraz $x=n$ dla *i*-bloku) nowa permutacja powstaje przez przesunięcie zadania $\pi(k)$ na pozycję a w π. Do relacji R dodawane są ograniczenia $\{(\pi(k),\pi(j)) : j\neq k,\ j=a,\ldots,x\}$ co oznacza, że zadanie $\pi(k)$ musi być wykonywane przed wszystkimi pozostałymi zadaniami ze zbioru $\{a,\ldots,x\}$. Natomiast dla $k=x,\ldots,b-1$ ($x=a$ dla bloku oraz $x=1$ dla *i*-bloku) nowa permutacja powstaje przez przesunięcie zadania $\pi(k)$ na pozycję b w π. Do relacji R dodawane są ograniczenia $\{(\pi(j),\pi(k)) : j\neq k,\ j=x,\ldots,b\}$ co oznacza, że zadanie $\pi(k)$ musi być wykonywane za wszystkimi pozostałymi zadaniami ze zbioru $\{x,\ldots,b\}$.

Węzeł drzewa przeszukiwań jest odcinany jeżeli relacja częściowego porządku R jest sprzeczna. Dodatkowo, podobnie jak w dokładnym algorytmie blokowym dla



Rys. 2. Proces obliczeń dla danych z przykładu

problemu RPQ, węzeł generowany przez przesunięcie zadania $\pi(k)$ jest odcinany, jeżeli $r_{\pi(k)} > r_{\pi(a)}$ w przypadku wstawiania na pozycję a oraz jeżeli $q_{\pi(k)} > q_{\pi(b)}$ w przypadku wstawiania na pozycję b.

Na rysunku 2 przedstawiono przebieg proponowanego algorytmu dla danych z przykładu. Każdy węzeł opisano w osobnym prostokącie, w którym znajduje się per-

mutacja, numer węzła oraz $D_{\max}$ (w prawym dolnym rogu). Zadania tworzące blok (*i*-blok) zostały połączone grubą linią, dodatkowo zadania $\pi(a)$ oraz $\pi(b)$ zostały wyróżnione grubą czcionką. Przesunięcia zadań wynikające ze strategii podziału zilustrowano przy pomocy łuków, przy czym łuki prowadzące do węzłów odcinanych zostały zaznaczone linią przerywaną.

Permutacja początkowa π (węzeł 1) została wygenerowana algorytmem Schrage. W permutacji π znajduje się *i*-blok $B_{9,6}$. W przypadku zadań 4 oraz 5 i 1 spełnione są warunki $r_4 > r_7$ oraz $q_5 > q_3$ i $q_1 > q_3$ zatem odpowiadające węzły zostaną odcięte. Ostatecznie w wyniku podziału powstaną trzy węzły potomne powstałe przez przesunięcie: (*i*) zadania 8 na pozycję 6 (węzeł 2), (*ii*) zadania 6 na pozycję 6 (węzeł 7), (*iii*) zadania 9 na pozycję 6 (węzeł 12). Węzły potomne zostały połączone z węzłem rodzicem łukiem z przyporządkowanym numerem zadania przesuwanego w wyniku generowania nowej permutacji.

W każdym potomku liczba ograniczeń pamiętanych w relacji częściowego porządku wzrasta. Stanowi ona główny mechanizm odcinania węzłów potomnych. Można to zauważyć na przykładzie węzła 11, w którym węzły potomne generowane przez przesunięcie zadań 5, 1, 3 i 8 zostały odcięte z powodu ograniczeń dodanych podczas tworzenia węzła 10, tj. przesunięcia zadania 9, które generuje ograniczenie : zadanie 9 musi być wykonywane za zadaniami 5,1,8 i 3. Algorytm znalazł rozwiązanie optymalne w 4 węźle o wartości 43, łącznie wygenerował 14 węzłów (permutacji) wobec $10!$ wszystkich możliwych.

Przeprowadzono testy efektywności algorytmu blokowego na wielu instancjach danych wygenerowanych losowo o rozmiarze od 10 do 80 zadań. W zdecydowanej większości przypadków liczba węzłów generowanych przez algorym nie przekraczała kilkudziesięciu i czas działania nie przekraczał kilkuset milisekund. Ponadto podczas działania algorytmu, *i*-bloki były generowane stosunkowo rzadko. Oznacza to, że dla wielu instancji problemu RPQ w rozwiązaniu dokładnym usunięcie okresów przestoju maszyny nie zwiększa momentu dostarczenia wszystkich zadań.

5. Podsumowanie

W pracy, dla jednomaszynowego problemu szeregowania zadań z terminami dostępności i czasami dostarczenia oraz brakiem możliwości przestoju maszyny sformułowano pojęcie *i*-bloku zadań, który jest rozszerzeniem dobrze znanego bloku zadań. Zaproponowano algorytm dokładny oparty na metodzie podziału i ograniczeń, w który własności blokowe zostały wykorzystane w strategii podziału i/lub ograniczeń. Z badań eksperymentalnych algorytmu wynika, że rozwiązanie znaczącej liczby instancji o rozmiarach kilkudziesięciu zadań zajmuje nie więcej niż 1 sekundę na komputerze klasy PC. Autor pracy jest przekonany, że w oparciu o *i*-bloki zadań można skonstruować efektywne algorytmy przeszukiwań lokalnych dla wielomaszynowych problemów szeregowania zadań.

LITERATURA

1. Carlier J.: The one-machine sequencing problem. European Journal of Operational Research 11/1, (1982), p. 42-47.

2. Cepek, O., Okada, M., and Vlach, M.: Note: On the two-machine no-idle flowshop problem.. Naval Research Logistics, 47(4), (2000), p. 353—358.

3. Grabowski J. and Pempera J.: New Block Properties for the Permutation Flow Shop Problem with Application in Tabu Search. The Journal of the Operational Research Society 52/2, (2001), p. 210–220.

4. Grabowski J., Nowicki E. and Zdrzałka S.: A block approach for single-machine scheduling with release dates and due dates. European Journal of Operational Research 26, (1986), p. 278–285.

5. Kalczynski P.J. and Kamburowski J.: A heuristic for minimizing the makespan in no-idle permutation flow shops. Computers & Industrial Engineering. (2005), p. 146–154.

6. Liu B., Wang L., Jin Y.: An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers. Computers & Operations Research 35/9, (2008), p. 2791–2806.

7. Nowicki E., Smutnicki C.: A fast taboo search algorithm for the job shop problem. Management Science 24/5, (1996), p. 530–534.

8. Nowicki E., Smutnicki C.: A fast tabu search algorithm for the permutation flow-shop problem. European Journal of Operational Research 91/1, (1996), p. 160–175.

9. Pan, Q.-K. and Wang, L.: No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm. In press at International Journal of Advanced Manufacturing Technology.

10. Schrage L.: Obtaining optimal solution to resource constrained network scheduling problem. Unpublished manuscript, 1971.

11. Vachajitpan P.: Job sequencing with continuous machine operation. Computers & Industrial Engineering 6/3, (1982), p. 255–259.