AUTOMATYKA • 2005 • Tom 9 • Zeszyt 1–2

Józef Grabowski*, Jarosław Pempera*

Zagadnienie przepływowe z ograniczeniami „bez magazynowania". Algorytm tabu search z multiruchami

1. Wprowadzenie

Zdolności wytwórcze większości spotykanych w praktyce systemów produkcyjnych z reguły znacznie przewyższają możliwości „wchłonięcia" przez rynek wyprodukowanych wyrobów. Między innymi z tego powodu coraz częściej projektuje się systemy produkcyjne, w których świadomie rezygnuje się z maksymalizacji wykorzystania maszyn na korzyść systemów elastycznych pozwalających na dostosowanie produkowanego asortymentu do zapotrzebowania rynkowego. Dużą elastyczność systemu można osiągnąć przez ograniczenie pojemności lub całkowitą rezygnację z buforów magazynujących przetwarzane detale pomiędzy obróbką na kolejnych stanowiskach. W takich systemach znacznie prostsze jest automatyczne sterowanie przebiegiem procesu produkcyjnego, zwiększa się niezawodność oraz zmniejsza koszty samego sterowania (brak urządzeń obsługujących bufory), koszty produkcji w toku.

W pracy rozważa się klasyczny problem przepływowy $F||C_{\max}$ z ograniczeniem „bez magazynowania", którym objęte są wszystkie pary maszyn sąsiadujących w ciągu technologicznym. Zagadnienie optymalizacji polega na określeniu takiej kolejności wykonywania zadań, aby minimalizować termin zakończenia realizacji wszystkich zadań. Zagadnienie to dla $m \geq 3$ jest *NP*-trudne (dwumaszynowy problem jest rozwiązywalny w czasie O($n$log n) algorytmem Gilmore-Gomory [1]). Silna *NP*-trudność problemu ogranicza zakres stosowania algorytmów dokładnych do instancji o małej liczbie zadań. Z tego powodu do rozwiązania (znalezienia dobrych rozwiązań przybliżonych) stosuje się najczęściej szybkie algorytmy heurystyczne oparte na metodach przeszukiwań lokalnych.

Problem przepływowy z ograniczeniami bez magazynowania cieszy się dużym zainteresowaniem badaczy zaledwie od kilkunastu lat. Wiąże się to przede wszystkim z faktem, że dopiero od niedawna pojawiły się w pełni zautomatyzowane elastyczne systemy produkcyjne umożliwiające ograniczenie pojemności lub wyeliminowanie buforów międzystadialnych. Spośród najnowszych prac poświęconych rozważanemu problemowi należy wymienić prace: Abaki [2], Caraffa i in. [3], Grabowski i Pempera [4], Leistein [5], McCormick i in.

* Instytut Cybernetyki Technicznej Politechniki Wrocławskiej

[6], Nowicki [7], Reddi and Ramamoorthy [8], Ronconi [9,10], Ronconi i Armentano [11], Smutnicki [12]. W pracy Hall i Sriskandarayah [13], przedstawiono przegląd problemów i algorytmów z ograniczeniami bez magazynowania i bez czekania.

W pracy przedstawiono model grafowy, własności problemu, algorytm lokalnej optymalizacji oparty na technice tabu search. W proponowanym algorytmie wykorzystano idee bloków zadań oraz zastosowano całkowicie nowe pojęcie nazywane multiruchem. Polega ono na tym, że kilka odpowiednio wybranych „pojedynczych" ruchów wykonuje się jednocześnie w jednej iteracji algorytmu. Umożliwia to uzyskanie znacznej poprawy wartości funkcji celu (w jednej iteracji) poprzez połączenie popraw poszczególnych pojedynczych ruchów wchodzących w skład multiruchu.

2. Opis problemu

Rozpatrywane zagadnienie może być sformułowane następująco: dany jest zbiór zadań $J = \{1, 2, ..., n\}$, który należy wykonać przy użyciu m maszyn ze zbioru 1, 2, ..., m. Każde zadanie $j \in J$ składa się z sekwencji m operacji $O_{j1}, O_{j2}, ..., O_{jm}$ wykonywanych kolejno na maszynach. Operacja O_{jk} odpowiada wykonywaniu zadania j na k-tej maszynie w czasie $p_{jk}>0$. Ze względu na brak możliwości magazynowania pomiędzy obróbką na kolejnych maszynach, zadanie $j \in J$, pomimo że zostało już zakończone, pozostaje na maszynie k, aż do chwili zwolnienia maszyny następnej w ciągu technologicznym, tj. $k + 1$, dla $k = 1, \ldots, m - 1$. Należy znaleźć uszeregowanie określone przez terminy rozpoczęcia (zakończenia) wykonywania zadań na poszczególnych maszynach takie, aby czas wykonywania zadań był najkrótszy.

W pracy [4] pokazano, że dopuszczalna kolejność wykonywania zadań w problemie przepływowym z ograniczeniem „bez magazynowania'' jest jednakowa na wszystkich maszynach. Będziemy ją opisywali za pomocą n-elementowej permutacji π elementów ze zbioru $\{1, \ldots, n\}$. Zbiór wszystkich takich permutacji będziemy oznaczali symbolem Π.

Dla ustalonej kolejności π dopuszczalny harmonogram wykonywania zadań na maszynach określony przez terminy rozpoczęcia wykonywania zadań $S_{jk}, j = 1, \ldots, n, k = 1, \ldots, m$ musi spełniać następujące ograniczenia:

$$S_{j1} \geq 0, \quad j = 1, \ldots, n \tag{1}$$

$$C_{jk} = S_{jk} + p_{jk}, \quad j = 1, \ldots, n, k = 1, \ldots, n \tag{2}$$

$$S_{jk} \geq C_{j,k-1}, \quad j = 1, \ldots, n, k = 2, \ldots, m \tag{3}$$

$$S_{\pi(j),k} \geq C_{\pi(j-1),k}, \quad j = 2, \ldots, n, k = 1, \ldots, m \tag{4}$$

$$Z_{jk} \geq C_{jk}, \quad j = 1, \ldots, n, k = 1, \ldots, m \tag{5}$$

$$S_{\pi(j),k} \geq Z_{\pi(j-1),k}, \quad j = 2, \ldots, n, k = 1, \ldots, m \tag{6}$$

$$S_{j,k} \geq Z_{jk-1}, \quad j = 1, \ldots, n, k = 2, \ldots, m \tag{7}$$

gdzie:

C_{jk} – moment zakończenia wykonywania zadania j na maszynie k,

Z_{jk} – moment opuszczenia (zwolnienia) tej maszyny przez to zadanie.

Nierówność (1) i równość (2) są oczywiste. Ograniczenia (3) i (4) modelują klasyczne wymagania kolejnościowe, odpowiednio technologiczne i maszynowe. Nierówność (5) oznacza, że moment zwolnienia maszyny k przez zadanie j nie może być wcześniejszy od momentu zakończenia obróbki tego zadania na tej maszynie. Nierówność (6) odpowiedzialna jest za „blokowanie” zadania $\pi(j)$ na maszynie k (odroczenie rozpoczęcia obróbki) aż do chwili zwolnienia maszyny k przez zadanie wykonywane przed zadaniem $\pi(j)$ w kolejności π. Ostatnia nierówność wiąże moment zwolnienia maszyny z terminem rozpoczęcia obróbki na następnej maszynie.

Po wyeliminowaniu zmiennych Z_{jk} i C_{jk} otrzymujemy równoważny zestaw nierówności:

$$S_{j1} \geq 0, \quad j = 1, \ldots, n \tag{8}$$

$$S_{jk} \geq S_{j,k-1} + p_{j,k-1}, \quad j = 1, \ldots, n, k = 2, \ldots, m \tag{9}$$

$$S_{\pi(j),k} \geq S_{\pi(j-1),k} + p_{\pi(j-1),k}, \quad j = 2, \ldots, n, k = 1, \ldots, m \tag{10}$$

$$S_{\pi(j),k} \geq S_{\pi(j-1),k+1} + p_{\pi(j-1),k+1}, \quad j = 2, \ldots, n, k = 1, \ldots, m \tag{11}$$

Chcemy znaleźć permutację $\pi^* \in \Pi$, taką że

$$C_{\max}(\pi^*) = \min_{\pi \in \Pi} C_{\max}(\pi) \tag{12}$$

gdzie $C_{\max}(\pi)$ jest czasem niezbędnym do zrealizowania wszystkich zadań na maszynach wykonywanych zgodnie z kolejnością określoną przez π. Wiadome jest, że dla problemu przepływowego $C_{\max}(\pi)=C_{\pi(n),m}$. Wartość $C_{\pi(n),m}=S_{\pi(n),m}+p_{\pi(n),m}$ może być wyznaczona przy wykorzystaniu następującej formuły:

$$S_{\pi(j)k} = \begin{cases} \max\{S_{\pi(j)k-1} + p_{\pi(j)k-1}, S_{\pi(j-1)k+1}\}, & j = 1,\ldots,n,\ k = 1,\ldots,m-1, \\ \max\{S_{\pi(j),m-1} + p_{\pi(j),m-1}, S_{\pi(j-1),m} + p_{\pi(j-1),m}\}, & j = 1,\ldots,n. \end{cases} \tag{13}$$

gdzie:

$\pi(0) = 0$,

$S_{0,k} = 0,\ k = 1, \ldots, m$,

$S_{j,0} = 0,\ j = 1, \ldots, n$.

3. Model grafowy

Dla danej kolejności wykonywania zadań π, definiujemy graf $G(\pi) = (N, R \cup F^0(\pi) \cup F^-(\pi))$ ze zbiorem węzłów N i zbiorem łuków $R \cup F^0(\pi) \cup F^-(\pi)$, gdzie:

– $N = \{1, ..., n\} \times \{1, ..., m\}$,

węzeł (j, k) reprezentuje k-tą operację zadania p(j); waga węzła $(j, k) \in N$ wynosi $p_{\pi(j)k}$;

– $R = \bigcup_{j=1}^{n} \bigcup_{k=1}^{m} \{((j,k),(j,k+1))\}$,

zbiór R zawiera łuki łączące operacje tego samego zadania;

– $F^o(\pi) = \bigcup_{j=1}^{n-1} \bigcup_{k=1}^{m} \{((\pi(j),k),(\pi(j+1,k))\}$,

zbiór $F^o(\pi)$ zawiera łuki łączące operacje wykonywane na tej samej maszynie; łuki ze zbiorów R i $F^o(\pi)$ mają wagę równą zero;

– $F^-(\pi) = \bigcup_{j=1}^{n-1} \bigcup_{k=1}^{m} \{((\pi(j),k+1),(\pi(j+1,k))\}$.

Każdy łuk $((\pi(j),k+1),(\pi(j+1,k)) \in F^-(\pi)$ ma wagę minus $p_{\pi(j),k+1}$ i powoduje opóźnienie wykonywania zadania $\pi(j + 1)$ na maszynie k, aż do chwili zwolnienia maszyny k+1 przez zadanie $\pi(j)$ (ograniczenie „bez magazynowania). Łuki ze zbiorów R, $F^0(\pi)$, $F^-(\pi)$ modelują ograniczenia odpowiednio (9), (10) i (11).

Własność 1. *Wyznaczenie najwcześniejszego terminu zakończenia wykonywania operacji o_{jk} równoważne jest wyznaczeniu długości najdłuższej drogi dochodzącej do węzła (j, k) (z obciążeniem tego węzła) w grafie G(π).*

Długość najdłuższej drogi do węzła $(\pi(n),m)$ określa najwcześniejszy termin zakończenia wykonywania zadań. Najdłuższa droga w $G(\pi)$ nazywana jest ścieżką krytyczną w $G(\pi)$. Ścieżka krytyczna, z oczywistych względów rozpoczynająca się w węźle $(\pi(1), 1)$ i kończąca w węźle $(\pi(n), m)$, może być reprezentowana przez sekwencję węzłów. Niech $u = (u_1, u_2, \ldots, u_w)$ oznacza jedną dowolnie wybraną ścieżkę krytyczną w $G(\pi)$, gdzie $u_i = (j_i, k_i) \in N$, $1 \le i \le w$, w jest liczbą węzłów w tej ścieżce.

W ścieżce u można wyróżnić dwa typy podciągów.

Pierwszy typ podciągów jest określony przez maksymalny podciąg $(u_g, \ldots, u_h) = ((j_g, k_g), \ldots, (j_h, k_h))$ ścieżki u taki, że $k_g = k_{g+1} = \ldots = k_h$ i $((j_i, k_i), (j_{i+1}, k_{i+1})) \in F^0(\pi)$ dla $i = g, \ldots, h–1$, $g < h$. Każdy taki podciąg określa sekwencję zadań $B_{gh} = (j_g, j_{g+1}, \ldots, j_{h-1}, j_h)$, którą będziemy nazywali *blokiem zadań* lub krótko *blokiem*. Zadania należące do bloku B_{gh} wykonywane są na maszynie k_g bez przerw.

Drugi typ podciągów jest określony przez maksymalny podciąg $(u_s,\ldots,u_t) = ((j_s,k_s), \ldots, (j_t,k_t))$ ścieżki u taki, że $((j_i, k_i), (j_{i+1}, k_{i+1})) \in F^-(\pi)$ dla $i = s, \ldots, t–1$, $s < t$. Każdy taki

podciąg określa sekwencję zadań $A_{st} = (j_s, j_{s+1}, \ldots, j_{t-1}, j_t)$, którą będziemy nazywali *antyblokiem zadań* lub krótko *antyblokiem*. Rozpoczęcie wykonywania zadań należących do antybloku A_{st} jest opóźnione ze względu na blokowanie maszyn przez zadania czekające na obróbkę na następnej maszynie – ograniczenie bez magazynowania.

Dla bloku $B_{gh} = (j_g, j_{g+1}, \ldots, j_{h-1}, j_h)$ definiujemy: *pierwsze* zadanie w bloku jako j_g, *ostatnie* zadanie w bloku jako j_h oraz *blok wewnętrzny* jako podsekwencję $B^*_{gh} = (j_{g+1}, \ldots, j_{h-1})$. W analogiczny sposób definiujemy dla antybloku $A_{st} = (j_s, j_{s+1}, \ldots, j_{t-1}, j_t)$ *pierwsze* j_s i *ostatnie* j_t zadanie oraz *antyblok wewnętrzny* jako $A^*_{st} = (j_{s+1}, \ldots, j_{t-1})$.

Własność 2. *Dla dowolnego bloku B_{gh} w π, niech α będzie permutacją otrzymaną z π przez dowolną modyfikację kolejności wykonywania zadań należących do bloku wewnętrznego B^*_{gh} wewnątrz tej sekwencji. Wówczas otrzymujemy: $C_{\max}(\alpha) \geq C_{\max}(\pi)$.*

Własność 3. *Dla dowolnego antybloku A_{st} w π, niech α będzie permutacją otrzymaną z π przez dowolną modyfikację kolejności wykonywania zadań należących do antybloku wewnętrznego A^*_{st} wewnątrz tej sekwencji. Wówczas otrzymujemy: $C_{\max}(\alpha) \geq C_{\max}(\pi)$.*

Dowody własności 2 i 3 przedstawiono w pracy [4].

4. Algorym tabu search z multiruchami (TS + M)

W każdej iteracji algorytmu opartego na metodzie przeszukiwania z zabronieniami, podobnie jak w algorytmie przeszukiwania zstępującego, dla pewnego rozwiązania zwanego bazowym, generowany jest podzbiór przestrzeni rozwiązań zwany sąsiedztwem. Z sąsiedztwa wybierane jest rozwiązanie najlepsze, które staje się rozwiązaniem bazowym w następnej iteracji, a najlepsze do tej pory znalezione rozwiązanie jest aktualizowane.

W celu zapobieżenia powrotu do rozwiązań już sprawdzonych wprowadza się mechanizm zabronień (tabu), który zwykle realizowany jest za pomocą listy cyklicznej. Elementy tej listy określają zabronione rozwiązania, które usuwane są z sąsiedztwa. Algorytm oparty na tej metodzie kończy swoje działanie, gdy zostanie spełniony określony warunek zatrzymania. Zwykle następuje to wtedy gdy zostanie wykonana z góry zadana liczba iteracji lub zostanie przekroczony zadany czas obliczeń.

Mechanizm zabronień zwykle pełni też rolę dywersyfikacyjną, tj. przeprowadza proces przeszukiwania do różnych obszarów przestrzeni rozwiązań problemu. Innym sposobem prowadzącym do dywersyfikacji jak i intensyfikacji jest wykorzystanie tzw. multiruchów. Stanowią one główny komponent algorytmu TS zaproponowanego przez Grabowskiego i Wodeckiego dla permutacyjnego problemu przepływowego [14] i problemu gniazdowego [15]. W dalszej części przedstawimy definicję multiruchów dla rozważanego problemu oraz przedstawimy schemat algorytmu.

4.1. Ruchy i ruchy złożone (multiruchy)

Dla bloku B_{gh} w π definiujemy zbiór $RB_{gh} = \{(j, j_h) \mid j \in B_{gh} - \{j_h\}\}$ oraz $LB_{gh} = \{(j, j_g) \mid j \in B_{gh} - \{j_g\}\}$. Elementy zbiorów RB_{gh} i LB_{gh} definiują ruch w π, przy czym $(j, j_h) \in RB_{gh}$ definiuje ruch „w prawo” polegający na przesunięciu zadania j bezpośrednio za zadanie j_h,

natomiast $(j, j_g) \in LB_{gh}$ definiuje ruch „w lewo'' polegający na przesunięciu zadania j bezpośrednio przed zadanie j_g. W analogiczny sposób definiujemy zbiór ruchów w prawo $RA_{st} = \{(j, j_t) \mid j \in A_{st} - \{j_t\}\}$ i w lewo $LA_{st} = \{(j, j_s) \mid j \in A_{st} - \{j_s\}\}$ dla antybloku A_{st}. Zauważmy, że w zbiorach RB_{gh}, LB_{gh}, RA_{st} i LA_{st} nie znajdują się ruchy, które są modyfikacją kolejności wykonywania zadań w sensie założeń własności 2 lub własności 3.

Niech l_B (l_A) oznacza liczbę bloków (antybloków) w π. Każdy z bloków może być opisany przez parę (g_i, h_i), $i = 1, \ldots, l_B$. W analogiczny sposób opisujemy każdy z antybloków jako parę (s_i, t_i), $i = 1, \ldots, l_A$. Ostatecznie zbiór ruchów V definiujemy następująco

$$V = \bigcup_{(g,h)\in GH} [RB_{gh} \cup LB_{gh}] \cup \bigcup_{(s,t)\in ST} [RA_{st} \cup LA_{st}],$$

gdzie:

$GH = \{(g_i, h_i) \mid i = 1, \ldots, l_B\}$,

$ST = \{(s_i, t_i) \mid i = 1, \ldots, l_A\}$.

Multiruchy

Dla każdego zadania należącego do wnętrza bloku B_{gh} definiujemy najlepsze ruchy: wykonywany w prawo $v_{R(gh)}$ oraz w lewo $v_{L(gh)}$, takie że:

$$C_{\max}(\pi_{v_{R(gh)}}) = \min_{v\in RB_{gh}\setminus\{(j,j_h)\}} C_{\max}(\pi_v),$$

$$C_{\max}(\pi_{v_{L(gh)}}) = \min_{v\in LB_{gh}\setminus\{(j,j_g)\}} C_{\max}(\pi_v).$$

Oczywiście, gdy blok B_{gh} nie zawiera bloku wewnętrznego to oba ruchy nie są definiowane. Podobna sytuacja ma miejsce, gdy oba ruchy dotyczą tego samego zadania, wówczas ruch generujący gorszą permutację nie jest definiowany.

Niech $BB = \{v_{R(gh)} \mid (g, h) \in GH\} \cup \{v_{L(gh)} \mid (g,h) \in GH\}$ oraz

$$BB^{(-)} = \{v \in BB \mid C_{\max}(\pi_v) < C_{\max}(\pi)\} = \{v_1, v_2, ..., v_p\}$$

będzie zbiorem ruchów profitujących. Zgodnie z definicją wykonanie dowolnego z ruchów należącego do $BB^{(-)}$ powoduje wygenerowanie permutacji lepszej od π. Podobny tok rozumowania przeprowadzony dla antybloków prowadzi do otrzymania następującego zbioru ruchów profitujących wykonywanych zadaniami z wnętrza antybloków

$$BA^{(-)} = \{v \in BA \mid C_{\max}(\pi_v) < C_{\max}(\pi)\} = \{v_1, v_2, ..., v_q\}.$$

Ostatecznie otrzymujemy zbiór wszystkich profitujących ruchów $BM^{(-)} = BB^{(-)} \cup BA^{(-)}$.

Wykonanie multiruchu oznacza wykonanie wszystkich ruchów ze zbioru $BM^{(-)}$ równocześnie, generując permutację $\pi_{\bar{v}}$, gdzie $\bar{v} = BM^{(-)}$. Zauważmy, że permutacja $\pi_{\bar{v}}$ nie

należy do otoczenia $N(V, \pi)$, poza sytuacją, gdy $|\bar{v}| = 1$, ponadto definicja $\bar{v}$ gwarantuje uzyskanie jednoznacznego rozwiązania niezależnie od kolejności wykonywania ruchów składowych.

4.2. Schemat – algorytm

Symbolem (*) oznaczono “najlepsze” wartości znalezione w trakcie przebiegu algorytmu, zerem (^{o}) wartości początkowe, natomiast bez indeksu oznaczono wartości bieżące.

INICJACJA

Podstaw $\pi=\pi^o$, $\pi^*=\pi^o$, $iter = 0$, $l = 0$, $T = \varnothing$.

PRZESZUKIWANIE

Podstaw $iter = iter + 1$, $\bar{v} = \varnothing$. Wyznacz zbiór V.
Wyznacz najlepszy ruch v ze zbioru V, tzn. taki że $C_{\max}(\pi_v) = \min_{w \in V} C_{\max}(\pi_w)$.
Jeżeli $C_{\max}(\pi_v) < C_{\max}(\pi^*)$ to podstaw $C_{\max}(\pi_v) < C_{\max}(\pi^*)$, idź do KRYTERIUM ZATRZYMANIA.
Jeżeli $l \geq Piter$, to
Wyznacz multiruch $\bar{v}$ oraz najlepszy ruch v składowy $\bar{v}$.
Jeżeli $\bar{v} \neq \varnothing$ idź do KRYTERIUM ZATRZYMANIA.
Wyznacz najlepszy niezabroniony ruch v ze zbioru V.

WYKONAJ RUCH

Na podstawie ruchu v, π oraz $iter$ odpowiednio zmodyfikuj listę T.
Jeżeli $\bar{v} \neq \varnothing$, to podstaw $\pi' = \pi_{\bar{v}}$, w przeciwnym wypadku podstaw $\pi' = \pi_v$.
Jeżeli $C_{\max}(\pi') \geq C_{\max}(\pi)$, to podstaw $l = l + 1$, w przeciwnym wypadku podstaw $l = 0$.
Podstaw $\pi = \pi'$.

KRYTERIUM ZATRZYMANIA

Jeżeli $iter \leq Maxiter$ to idź do PRZESZUKIWANIE.

5. Eksperyment komputerowy

Celem przeprowadzonego eksperymentu komputerowego było przebadanie wpływu proponowanych rozszerzeń klasycznego algorytmu opartego na przeszukiwaniu z zabronieniami. Prezentowane w pracy algorytmy zostały zaprogramowane w języku C++ oraz były testowane na komputerze PC z procesorem Pentium IV 1000 MHz. na zestawie literaturowych przykładów testujących zaproponowanych przez Tailarda [16]. Zestaw ten składa 120 ,,trudnych” instancji permutacyjnego problemu przepływowego podzielonych na 12 grup zawierających 10 instancji o tej samej liczbie maszyn m i tej samej liczbie zadań n.

Do wyznaczenia permutacji początkowej dla algorytmu TS został użyty odpowiednio zaadoptowany algorytm NEH [17]. Adaptacja tego algorytmu do potrzeb rozważanego problemu polegała na uwzględnieniu ograniczeń ,,bez magazynowania’’ podczas wyznacza-

nia wartości funkcji celu dla każdej permutacji częściowej przetwarzanej w algorytmie. Algorytmy testowane były z różnymi wartościami parametrów sterujących celem wyboru najlepszych. Ostatecznie przyjęto następujące parametry algorytmów: TS – $L = 7$, TS + M – $LB = 7$, $LM = 4$, $PG = 2$.

Dla każdej instancji problemu i każdego z testowanych algorytmów (TS, TS+M) wyliczono następujące wielkości:

– PRD = 100%(C^*–C)/C^* – procentową różnicę wartości referencyjnej C^* z pracy [10] i wartości funkcji celu C uzyskanej testowanym algorytmem względem wartości referencyjnej C^*,
– CPU – czas obliczeń w sekundach.

Następnie dla każdej grupy instancji testowych obliczono:

– APRD – wartość średnią PRD (wyznaczoną dla 10 instancji),
– ACPU – wartość średnią CPU (wyznaczoną dla 10 instancji).

W tabeli 1 przedstawiono wyniki porównania testowanych algorytmów oraz dodatkowo algorytmu NEH. Z prezentowanych w tabeli rezultatów wynika, że algorytmy oparte na metodzie przeszukiwania z zabronieniami dostarczają dla większości grup instancji znacznie lepsze rozwiązania od rozwiązań referencyjnych. Algorytmy TS i TS+M generują rozwiązania o podobnej jakości tylko w zakresie pierwszych 1000 iteracji. Średnia względna poprawa wyznaczona dla wszystkich grup dla TS wynosi 0,36, natomiast dla TS+M jest trochę większa i wynosi 0,46. Dla większej liczby iteracji efektywność algorytmu TS+M jest ewidentnie większa. Dla 30 000 średnia poprawa algorytmu TS+M wynosi 1,68 podczas, gdy dla algorytmu TS wynosi ona tylko 0,69.

Tabela 1
Zestawienie wartości APRD testowanych algorytmów

n×*m*	NEH	TS			TS+M		
Iter		1000	10000	30000	1000	10000	30000
20×5	–4,75	–2,40	–1,88	–1,88	–1,86	–0,86	–0,34
20×10	–2,99	0,84	1,32	1,32	0,33	1,52	1,76
20×20	–0,01	2,36	2,76	2,88	2,26	2,66	2,94
50×5	–2,59	–0,82	–0,68	–0,68	–0,50	0,38	0,55
50×10	–0,94	1,44	1,86	1,86	1,85	3,23	3,52
50×20	0,11	2,21	3,23	3,54	2,11	3,58	4,26
100×5	–4,19	–3,28	–3,28	–3,28	–3,12	–2,64	–2,62
100×10	–0,27	1,54	1,57	1,57	1,52	2,44	2,66
100×20	–0,02	1,65	1,99	1,99	1,73	2,65	3,03
200×10	–1,46	–0,74	–0,74	–0,74	–0,51	0,30	0,58
200×20	–0,19	0,94	1,11	1,11	1,06	1,99	2,31
500×20	0,26	0,57	0,58	0,58	0,76	1,13	1,47
Razem	–1,42	0,36	0,65	0,69	0,47	1,36	1,68

W tabeli 2 przedstawiono rozszerzone wyniki eksperymentu dotyczące algorytmu TS+M. Zawiera one wyniki dla mniejszej liczby iteracji oraz średni czas obliczeń. Z analizy wyników zamieszczonych w tej tabeli wynika, że już po wykonaniu 100 iteracji algorytm TS+M w większości grup przykładów znalazł rozwiązania lepsze od referencyjnych. W przypadku grupy 50×20 dla 100 iteracji wartość APRD wynosi przeszło 1%. Poprawę tą osiągnięto w czasie mniejszym od 0,1 s. Algorytm TS+M wyjątkowo skutecznie działa dla instancji o małym stosunku liczby zadań do liczby maszyn. Wartość współczynnika APRD jest największa dla omawianej grupy 50×20 (n/m = 2,5) i wynosi 4,26.

Tabela 2
Szczegółowe wyniki dla algorytmów TS+M

Maxiter	100	200	500	1000		10000		20000		30000	
N×*m*	APRD	APRD	APRD	APRD	ACPU	APRD	ACPU	APRD	ACPU	APRD	ACPU
20×5	–3,04	–2,52	–2,02	–1,86	0,1	–0,86	0,9	–0,68	1,8	–0,34	2,7
20×10	–0,95	–0,70	–0,16	0,33	0,2	1,52	1,6	1,70	3,1	1,76	4,6
20×20	1,32	1,49	1,96	2,26	0,3	2,66	2,6	2,82	5,1	2,94	7,6
50×5	–1,51	–1,21	–0,74	–0,50	0,2	0,38	2,0	0,55	4,0	0,55	6,2
50×10	0,73	0,95	1,46	1,85	0,4	3,23	3,6	3,38	7,1	3,52	10,8
50×20	1,08	1,25	1,91	2,11	0,6	3,58	6,4	3,97	12,9	4,26	19,1
100×5	–3,54	–3,40	–3,28	–3,12	0,4	–2,64	4,1	–2,64	8,3	–2,62	12,3
100×10	0,91	1,07	1,31	1,52	0,7	2,44	7,3	2,63	14,5	2,66	22,0
100×20	0,98	1,11	1,40	1,73	1,2	2,65	13,0	2,90	26,1	3,03	39,2
200×10	–0,89	–0,81	–0,63	–0,51	1,6	0,30	15,0	0,50	29,8	0,58	44,1
200×20	0,52	0,68	0,82	1,06	2,7	1,99	26,5	2,16	53,1	2,31	79,3
500×20	0,56	0,62	0,67	0,76	7,1	1,13	69,8	1,32	140,2	1,47	209,3
Razem	–0,32	–0,12	0,23	0,47		1,36		1,55		1,68	

Czas obliczeń algorytmu rośnie proporcjonalnie do liczby maszyn m, do liczby zadań n oraz liczby iteracji i waha się w pobliżu średniego czasu działania algorytmu TS.

6. Wnioski i uwagi

Dla rozważanego w pracy problemu przepływowego z ograniczeniami bez magazynowania, można zaobserwować ewidentną poprawę efektywności algorytmu TS spowodowaną zastosowaniem mechanizmu dywersyfikacji i intensyfikacji procesu obliczeń zwanego multiruchem. Zastosowany w algorytmie TS+M omawiany mechanizm pozwala na znalezienie lepszych rozwiązań w tej samej liczbie iteracji oraz na skuteczną dywersyfikację procesu przeszukiwań prowadzącą do penetracji różnych obszarów przestrzeni rozwiązań i w konsekwencji do znalezienia lepszych rozwiązań (patrz wyniki dla 30 000 iteracji). Klasyczny algorytm TS po wykonaniu 1000 iteracji osiąga zaledwie 20% poprawy, co wię-

cej 30-krotne zwiększenie liczby iteracji nie zmienia w zasadniczy sposób tej sytuacji. Dla kontrastu, w przypadku problemów permutacyjnego przepływowego i przepływowego z ograniczeniami bez czekania w ciągu pierwszych 1000 iteracji algorytm TS osiąga około 90% uzyskanej poprawy.

Literatura

[1] Gilmore P.C., Gomory R.E.: *Sequencing a state-variable machine: a solvable case of the traveling salesman problem*. Operations Research, 12, 1964, 655–679

[2] Abadi I.N.K., Hall N.G., Sriskandarayah C.: *Minimizing Cycle Time in a Blocking Flowshop*. Operations Research, 48, 2000, 177–180

[3] Caraffa V., Ianes S., Bagchi T.P., Sriskandarayah C.: *Minimizing makespan in a blocking flowshop using genetic algorithms*. International Journal of Production Economics, 70, 2001, 101–115

[4] Grabowski J., Pempera J.: *Sequencing of jobs in some production system.* European Journal of Operational Research, 125, 2000, 535–550

[5] Leistein R.: *Flowshop sequencing with limited buffer storage*. International Journal of Production Research, 28, 1990, 2085–2100

[6] McCormick M.L., Pinedo M.L., Shenker S., Wolf B.: *Sequencing in an assembly line with blocking to minimize cycle time*. Operations Research, 37, 1989, 925–935

[7] Nowicki E.: *The permutation flow shop with buffers: A tabu search approach.* European Journal of Operational Research, 116, 1999, 205–219

[8] Reddi S.S., Ramamoorthy C.V.: *On flowshop sequencing problems with no-wait in process.* Operational Research Quarterly, 23, 1972, 323–331

[9] Ronconi D.P.: *A note on constructive heuristics for the flowshop problem with blocking.* International Journal of Production Economics, 87, 2004, 39–48

[10] Ronconi D.P.: *A branch-and-bound algorithm to mnimize the makespan in a flowshop problem with blocking*. Annals of Operations Research (in print)

[11] Ronconi D.P., Armentano V.A.: *Lower bounding schemes for flowshops with blocking in-process.* Journal of the Operational Research Society, 52, 2001, 128912–97

[12] Smutnicki C.: *Some properties of scheduling problem with storing constraints*. Zeszyty Naukowe AGH: Automatyka, 34, 1983, 223–232

[13] Hall N.G., Sriskandarayah C.: *A survey of machine scheduling problems with blocking and no-wait in process*. Operations Research, 44, 1996, 510–525

[14] Grabowski J., Wodecki M.: *A very fast tabu search algorithm for the job shop problem*. In: C. Rego, B. Alidaee (Eds), Metaheuristic Optimization via Memory and Evolution; Tabu Search and Scatter Search, Kluwer Academic Publishers, 2004, (w druku)

[15] Grabowski J., Wodecki M.: *A very fast tabu search algorithm for the flow shop problem with makespan criterion*. Computers and Operations Research, 11, 2004, 1891–1909

[16] Taillard E.: *Benchmarks for basic scheduling problems.* European Journal of Operational Research, 64, 1993, 278–285

[17] Nawaz M., Enscore E., Ham I.: *A Heuristic algorithm for the m-machine, n-job flowshop sequencing problem*. OMEGA The International Journal of Management Science, 11, 1983, 91–95