AUTOMATYKA • 2009 • Tom 13 • Zeszyt 2

Józef Grabowski*, Jarosław Pempera*

Hybrydowy algorytm tabu dla niepermutacyjnego problemu przepływowego z kryterium sumacyjnym

1. Wprowadzenie

W przepływowym systemie produkcyjnym należy wykonać określoną liczbę zadań na maszynach zadań przy użyciu określonej liczby maszyn stanowiących park maszynowy. Każde zadanie wykonywane jest na każdej maszynie w kolejności zgodnej z numeracją tych maszyn. Planowanie operacyjne w takim systemie polega na wyznaczeniu harmonogramu wykonywania zadań na maszynach, który minimalizuje (maksymalizuje) zadane kryterium optymalizacyjne. Harmonogram wykonywania zadań określony jest przez momenty rozpoczęcia oraz zakończenia wykonywania zadań na poszczególnych maszynach. Dopuszczalny harmonogram wykonywania cechuje się tym, że w dowolnej chwili dowolna maszyna wykonuje tylko jedno zadanie oraz dowolne zadanie wykonywane jest tylko na jednej maszynie.

Zdecydowana większość prac poświęcona problemowi przepływowemu FS (*flow shop problem*) dotyczy jego szczególnego przypadku jakim jest permutacyjny problem przepływowy PFS (*permutation flow shop problem*). W permutacyjnym problemie przepływowym dodatkowo wymaga się, aby kolejność wykonywania zadań na wszystkich maszynach była identyczna. W obu przypadkach dla zadanej kolejności wykonywania zadań na maszynach można w szybki i jednoznaczny sposób wyznaczyć dopuszczalny harmonogram wykonywania zadań. Problem PFS jest jednym ze sztandarowych problemów harmonogramowania zadań i w związku z tym za każdym razem, gdy pojawi się jakaś nowa metoda konstruowania algorytmów heurystycznych dla problemów kombinatorycznych, opracowywany jest algorytm dla tego właśnie problemu. W pracy [1] można zapoznać się z zarysem historycznym dotyczącym rozwoju metod rozwiązywania problemu PFS.

Problemy przepływowe z kryteriami sumowo-kosztowymi, takimi jak suma czasów zakończenia zadań, przepływ całkowity lub średni są rzadko rozważane w literaturze. Często problemy te uważane są za znacznie trudniejsze. Dla problemu PFS opracowano algorytmy

* Instytut Automatyki, Informatyki i Robotyki, Politechnika Wrocławska

dokładne [2] pozwalające na rozwiązanie problemów o niewielkie liczbie zadań i/lub maszyn. Do rozwiązania problemów o rzeczywistych rozmiarach opracowano algorytmy heurystyczne oparte na metodach przeszukiwania lokalnego [3], mrówkowego [4], genetycznego [5], stadnego [6] oraz przeszukiwania z zabronieniami [7]. Z powodu braku własności pozwalających na efektywną obliczeniowo predykcję rozwiązań nie lepszych od rozwiązania bazowego, znalezienie dobrych rozwiązań przez te algorytmy związane jest z przeglądnięciem bardzo dużej liczby rozwiązań.

W przypadku niepermutacyjnego problemu przepływowego z kryterium minimalizacji czasu zakończenia zadań wykazano, że stosowanie niepermutacyjnej strategii wytwarzania może znacząco skrócić czas wykonania zadań [8].

2. Opis formalny problemu

W niepermutacyjnym problemie przepływowym dany jest zbiór zadań $J = \{1, 2, ..., n\}$, który należy wykonać przy użyciu m maszyn ze zbioru $M = \{1, 2, ..., m\}$. Zadanie $j \in J$ wykonywane jest kolejno na każdej maszynie. Czas wykonania zadania $j \in J$ na maszynie $k \in M$ wynosi $p_{jk} > 0$. Kolejność wykonywania zadań na maszynach można opisać za pomocą zestawu składającego z m permutacji $\pi = (\pi_1, \ldots, \pi_m)$. Permutacja π_k, określa kolejność wykonywania zadań na maszynie $k \in M$. Oczywiście każda permutacja π_k, $k \in M$, określona jest na zbiorze $\{1, \ldots, n\}$.

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

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

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

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

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

Nierówność (1) i równość (2) są oczywiste. Nierówność (3) modeluje ograniczenie technologiczne i oznacza, że moment rozpoczęcia wykonywania danego zadania na danej maszynie nie może być wcześniejszy od momentu zakończenia realizacji tego zadania na maszynie poprzedniej. Nierówność (4) modeluje ograniczenie wynikające z jednostkowej przepustowości maszyn.

Dla zadanej kolejności wykonywania zadań π najwcześniejsze momenty zakończenia realizacji zadań na maszynach można wyznaczyć z następującego wzoru rekurencyjnego:

$$C_{\pi k(j),k} = \max(C_{\pi k(j-1),k}, C_{\pi k(j),k-1}) + p_{\pi k(j),k} \tag{5}$$

gdzie $\pi_k(0) = 0$, $C_{0,k} = 0$ dla $k \in M$, oraz $C_{j,0} = 0$ dla $j \in J$.

Niech Π będzie zbiorem wszystkich kolejności. Zbiór Π zawiera $(n!)^m$ zestawów permutacji. Problem optymalizacji polega na wyznaczeniu takiej kolejności π^*, dla której suma czasów zakończenia zadań $C_{sum}(\pi^*)$ będzie jest najmniejsza, precyzyjniej

$$C_{sum}(\pi^*) = \min_{\alpha \in \Pi} C_{sum}(\alpha) \tag{6}$$

gdzie $C_{sum}(\pi) = \sum_{s=1}^{n} C_{\pi_m(s),m}$.

3. Model grafowy

Problem wyznaczenia dopuszczalnego harmonogramu wykonywania zadań dla ustalonej kolejności ich wykonywania π można sprowadzić do problemu wyznaczenia długości najdłuższych dróg w grafie skierowanym $G(\pi) = (N, T \cup F(\pi))$ ze zbiorem obciążonych węzłów N i zbiorem nieobciążonych łuków $T \cup F(\pi)$, gdzie

$$N = \{1, ..., n\} \times \{1, ..., m\} \tag{7}$$

węzeł $(j, k) \in N$ obciążony jest wagą $p_{\pi(j)k}$,

$$T = \bigcup_{j=1}^{n} \bigcup_{k=1}^{m-1} \{((j,k),(j,k+1))\} \tag{8}$$

zbiór T zawiera modelujące ograniczenia technologiczne (3)

$$F(\pi) = \bigcup_{k=1}^{m} \bigcup_{j=1}^{n-1} \{((\pi_k(j),k),(\pi_k(j+1),k))\} \tag{9}$$

łuk $((\pi_k(j),k), (\pi_k(j + 1),k)) \in F(\pi)$ modeluje ograniczenia maszynowe (4).

Własność 1. Najwcześniejszy moment zakończenia realizacji zadania $\pi_k(j)$ na maszynie k na maszynie jest równy długości najdłuższej drogi (ścieżki) dochodzącej do węzła (j,k) (z obciążeniem tego węzła) w grafie $G(\pi)$.

Najdłuższą drogę dochodzącą do węzła $x = (j,k)$ można jednoznacznie opisać za pomocą ciągu węzłów $u^x = (u_1^x, u_2^x, ..., u_{w^x}^x)$ w grafie $G(\pi)$, gdzie $u_i^x = (j_i^x, k_i^x) \in N$, natomiast w^x jest liczbą węzłów w tej ścieżce. Każda taka ścieżka z oczywistych względów, rozpoczyna się w węźle $(\pi_1(1),1)$.

Własność 2. Problem wyznaczenia sumy czasów zakończenia wykonywania zadań możemy sprowadzić do wyznaczenia sumy długości najdłuższych dróg dochodzących do węzłów reprezentujących zadania wykonywane na ostatniej maszynie.

Własność 3. Długości najdłuższych dróg dochodzących do węzłów dla wszystkich węzłów można wyznaczyć w czasie $O(nm)$.

Najdłuższą drogę u^x można podzielić na podciągi węzłów reprezentujących zadania wykonywane na tej samej maszynie. Sekwencję zadań $B^x_{gh} = (j^x_g,\ j^x_{g+1}, ..., j^x_{h-1},\ j^x_h)$ składającą się z co najmniej dwóch zadań odpowiadającą maksymalnemu podciągowi $(u^x_g, ..., u^x_h) = ((j^x_g, k^x_g), ..., (j^x_h, k^x_h))$ ścieżki u^x takiemu, że $k^x_g = ... = k^x_h$ będziemy nazywali *blokiem zadań*. Dla bloku $B^x_{gh} = (j^x_g,\ j^x_{g+1}, ..., j^x_{h-1},\ j^x_h)$ definiujemy: *pierwsze* zadanie w bloku jako j^x_g, *ostatnie* zadanie w bloku jako j^x_h oraz *blok wewnętrzny* jako podsekwencję $(j^x_g,\ j^x_{g+1}, ...,\ j^x_{h-1},\ j^x_h)$.

Własność 4. Niech $\pi \in \prod$ będzie dowolną kolejnością wykonywania zadań, niech u^x będzie najdłuższą drogą do węzła x, natomiast $L^x(\pi)$ długością tej drogi. Jeżeli istnieje kolejność wykonywania zadań $\beta \in \prod$ taka, że $L^x(\beta) < L^x(\pi)$, wtedy β przynajmniej jedno zadanie z przynajmniej jednego bloku zadań wykonywane jest przed pierwszym lub za ostatnim zadaniem z tego bloku.

Własność 4 bezpośrednio wywodzi się z teorii blokowej [10].

Własność 5. Niech $\pi \in \prod$ będzie dowolną kolejnością wykonywania zadań. Jeżeli istnieje kolejność wykonywania zadań $\beta \in \prod$ taka, że $C_{\text{sum}}(\beta) < C_{\text{sum}}(\pi)$, wtedy β przynajmniej jedno zadanie z przynajmniej jednego bloku spośród wszystkich bloków wyznaczonych dla węzłów reprezentujących wykonywanie zadań na ostatniej maszynie wykonywane jest przed pierwszym lub za ostatnim zadaniem z tego bloku.

Własność 5 jest prostą konsekwencją własności 2 oraz 4.

4. Algorytmy metaheurystyczne

Obecnie najskuteczniejsze algorytmy metaheurystyczne dla problemów szeregowania zadań oparte są na szeroko rozumianych metodach przeszukiwań lokalnych. W każdej iteracji tego typu algorytmów przeglądany jest niewielki podzbiór przestrzeni rozwiązań problemu celem wyznaczenia rozwiązania najlepszego oraz wyznaczany jest podzbiór przeszukiwany w następnej iteracji. Na efektywność tego typu algorytmów decydujący wpływ ma typ ruchów generujący sąsiedztwo.

W wielopermutacyjnych problemach harmonogramowania zadań jednym z najskuteczniejszych otoczeń jest zredukowane otoczenie typu zamień sąsiednie zaproponowane dla problemu gniazdowego w pracy [11]. Redukcja otoczenia polega na wyeliminowaniu ruchów, o których wiemy apriorycznie, że nie przyniosą poprawy funkcji celu. Są to w szczególności zmieniające kolejność wykonywania zadań nienależących do bloków oraz zadań wewnętrznych bloków. Wysoka skuteczność tego otoczenia została potwierdzona badaniami prezentowanymi w pracy [12] oraz z powodzeniem zastosowana do konstrukcji hybrydowego algorytmu opartego na metodach przeszukiwań lokalnych dla niepermutacyjnego problemu przepływowego z kryterium minimalizacji czasu zakończenia zadań [9].

4.1. Otoczenie

Dla rozważanego problemu, ruch typu zamień sąsiednie możemy opisać za pomocą trójki $v = (k, a, a + 1)$. W wyniku wykonania ruchu $v = (k, a, a + 1)$ w kolejności π otrzymujemy nową kolejność β, w której $\beta_s = \pi_s$ dla $s \neq k$, $s = 1, \ldots, m$ oraz $\beta_k = (\pi_k(1), \ldots, \pi_k(a - 1), \pi_k(a + 1), \pi_k(a), \ldots, \pi_k(n))$. Sąsiedztwo zamień sąsiednie składa się z $(n - 1)m$ rozwiązań sąsiednich i wymaga $O(n^2m^2)$ czasu. Czas przeglądania otoczenia możemy zmniejszyć, jedynie zmniejszając liczbę przeglądanych rozwiązań sąsiednich. Jak dotąd, podobnie jak w przypadku permutacyjnego problemu przepływowego z kryterium sumacyjnym, nie są znane efektywne metody akceleracji obliczeń. Na podstawie własności 5 możemy apriorycznie wskazać ruchy, które wygenerują rozwiązania na pewno nie lepsze od rozwiązania bazowego.

W tym celu weźmy pod uwagę najdłuższą drogę prowadzącą do węzła reprezentującego wykonywanie na ostatniej maszynie jednego dowolnie wybranego zadania. Zauważmy, że ruchy zmieniające kolejność wykonywania zadań nienależących do żadnego z bloków wyznaczonych dla tej drogi oraz zadań wewnętrznych bloku nie przyniosą skrócenia jej długości. Jedynymi ruchami typu zamień sąsiednie dającymi szansę na zmniejszenie jej długości i poprawę wartości funkcji celu są ruchy przesuwające drugie zadanie z bloku przed pierwsze lub przedostatnie za ostatnie. Można pokazać, że dla dowolnej kolejności wykonywania zadań, najdłuższa droga do tego typu węzłów odwiedza każdą maszynę dokładnie jeden raz, zatem dla każdego zadania mamy co najwyżej $2m$ ruchów rokujących poprawę i łącznie $2nm$ ruchów, jeżeli weźmiemy pod uwagę wszystkie węzły na ostatniej maszynie. W rzeczywistości liczba węzłów jest zdecydowanie mniejsza, ponieważ niektóre ruchy mogą być identyczne dla wielu dróg, bloki dwuelementowe generują tylko jeden ruch, natomiast fragmenty drogi składające się tylko z jednego zadania na maszynie nie generują żadnego ruchu rokującego poprawę wartości funkcji celu. Oczywiście liczba ruchów rokujących poprawę wartości funkcji celu zależy od dystrybucji bloków.

4.2. Hybrydowy algorytm przeszukiwania z zabronieniami (HTS)

Technika przeszukiwania z zabronieniami jest jedną z najbardziej efektywnych metod konstruowania algorytmów heurystycznych dla problemów szeregowania zadań. Jej wysoką efektywność dla problemu przepływowego z kryterium sumacyjnym potwierdzają wyniki badań zawarte w pracy [7]. W każdej iteracji algorytmu wyznaczana jest wartość funkcji celu dla wszystkich rozwiązań z sąsiedztwa rozwiązania bazowego celem wyboru najlepszego. Istotnym elementem metody jest mechanizm zabronień, który zabezpiecza proces przeszukiwań przed nieustannym powtarzaniem fragmentu trajektorii przeszukiwań.

Do rozwiązania problemu proponujemy hybrydowy (dwuczłonowy) algorytm oparty na metodzie przeszukiwania z zabronieniami. Zadaniem pierwszego członu algorytmu jest wyznaczenie dobrego rozwiązania permutacyjnego. Drugi (właściwy) człon rozpoczyna przeszukiwania od otrzymanego rozwiązania permutatycnego i wyszukuje możliwie najlepsze rozwiązanie niepermutacyjne.

Do konstrukcji pierwszego członu wykorzystano uproszczony algorytm tabu search z pracy [7]. Uproszczenie polegało na usunięciu wszystkich mechanizmów dywersyfikacyjnych, tj. dynamicznej długości zabronień, mechanizmu multiruchu. Algorytm oparty jest na sumie sąsiedztw typu wstaw i zamień. Na potrzeby dalszego opisu algorytm ten będziemy oznaczali symbolem TSP.

Drugi człon algorytmu, algorytm TSNP, operuje na rozwiązaniach niepermutacyjnych (zestawach permutacji). Zastosowano w nim zredukowane sąsiedztwo typu zamień sąsiednie oraz mechanizm zabronień wzorowany na rozwiązaniach stosowanych m.in. w pracach [7] i [11]. Precyzyjniej, niech β będzie najlepszym niezabronionym rozwiązaniem w otoczeniu rozwiązania bazowego π wygenerowanym przez wykonanie ruchu $v = (k, a, a + 1)$ w tej kolejności. Do listy zabronień dodawana jest trójka $(k, \pi_k(a), \pi_k(a + 1))$. Element listy zabronień (k, a, b) zabrania wszystkich kolejności, w których zadanie a znajduje się przed zadaniem b na maszynie k. Na liście pamiętanych jest L ostatnich trójek. Algorytm TSNP kończy działanie po wykonaniu zadanej liczby iteracji.

5. Badania eksperymentalne

W celu oceny jakości omawianego otoczenia oraz efektywności opracowanego hybrydowego algorytmu tabu przeprowadzono eksperyment komputerowy na pierwszych pięciu grupach instancji problemu przepływowego, które zostały zaproponowane przez Taillarda [13]. W zestawie tym dla każdej pary $n \times m$: 20×5, 20×10, 20×20, 50×5, 50×10, 50×20 znajduje się 10 przykładów testujących.

Tabela 1
Wyniki badań eksperymentalnych

Grupa	$PRD(\pi^{HTS1})$	$PRD(\pi^{HTS2})$	CPU^{TSNP}	$DIV(\pi^{NAT})$	$DIV(\pi^{HTS1})$	$DIV(\pi^{TSP})$	$DIV(\pi^{HTS2})$
20 × 5	8,70	0,04	0,39	33,50	21,87	0,03	– 0,09
20 × 10	3,48	0,10	0,92	28,41	23,93	0,04	– 0,24
20 × 20	1,48	0,24	2,29	19,78	17,99	0,00	– 0,11
50 × 5	2,81	0,20	1,88	32,43	28,65	2,08	2,00
50 × 10	1,61	0,51	4,99	30,91	28,77	1,53	1,34
Średnio	3,62	0,22		29,01	24,24	0,74	0,58

Algorytm został zaprogramowany w środowisku Visual C++ 2005 i uruchomiony na komputerze z procesorem Intel Core 2 Duo 2.60 GHz. Rozwiązanie początkowe dla algorytmu TSP zostało wygenerowane algorytmem NEH [14]. Dla każdej instancji problemu zarówno algorytm TSP jak i TSNP wykonał 1000 iteracji. Długość listy zabronień wynosiła odpowiednio 7 oraz 25.

Dla każdego przykładu wyznaczono 4 rozwiązania: naturalną kolejność wykonywania zadań $-\pi^{NAT}$, kolejność wygenerowaną algorytmem TSP $-\pi^{TSP}$, kolejność wygenerowaną algorytmem HTS z naturalną kolejnością początkową $-\pi^{HTS1}$, kolejność wygenerowaną algorytmem HTS z rozwiązaniem początkowym wygenerowanym algorytmem TSP $-\pi^{HTS2}$. Następnie obliczono następujące wielkości:

- $PRD(\pi)$ – $100\%\ (C_{sum}(\pi^0) - C_{sum}(\pi))/C_{sum}(\pi^0)$ względna poprawa rozwiązania początkowego π^0,
- $DIV(\pi)$ – $100\%\ (C_{max}(\pi) - C^{reft})/C^{ref}$ względna różnica wartości funkcji celu rozwiązania wygenerowanego algorytmem TS i wartości referencyjnej C^{ref} dla permutacyjnego problemu przepływowego z pracy [5],
- CPU^A – czas obliczeń algorytmu A.

Wyniki eksperymentu komputerowego przedstawiono w tabeli 1. Dodatkowo dla każdej grupy wyznaczono najmniejszą wartość $DIV(\pi^{HTS2})$. Wartości te odpowiednio wynosiły – 0,45; – 1,21; – 0,28; 0,89; 0,42%. Algorytm HTS został uruchomiony również na większą liczbę iteracji jednakże dla większości instancji otrzymywane rozwiązania końcowe nie były lepsze.

Z analizy wyników eksperymentu komputerowego wynika, że jakość rozwiązań generowanych przez algorytm HTS istotnie zależy od rozwiązania początkowego. Dla naturalnych rozwiązań początkowych, znacznie oddalonych od rozwiązań optymalnych średnio 24%, obserwuje się największą poprawę uzyskiwaną przez algorytm TSNP, która wynosiła średnio w zależności od grupy od 1,4–8,7%.

W przypadku uruchomienia algorytmu TSNP z rozwiązaniem początkowym wygenerowanym algorytmem TSP, obserwuje się znacznie mniejsze poprawy tj. od 0,04–0,51%. Ujemne wartości DIV świadczą o znalezieniu rozwiązań lepszych od referencyjnych rozwiązań permutacyjnych (w przypadku pierwszych trzech grup najprawdopodobniej optymalnych). Zatem istnieją harmonogramy niepermutacyjne o nawet 1,21% lepszej wartości funkcji celu.

6. Podsumowanie

W pracy został przedstawiony hybrydowy algorytm tabu search dla niepermutacyjnego problemu przepływowego bazujący na jednym najefektywniejszych otoczeń dla problemów wielopermutacyjnych. Wyniki badań eksperymentalnych wskazują, że w klasie rozwiązań niepermutacyjnych znajdują się rozwiązania znacząco lepsze od rozwiązań permutacyjnych. Niestety efektywność tego otoczenia ogólnie należy ocenić nisko. Uzyskiwane poprawy są relatywnie niewielkie. Dalsze kierunki badań powinny koncentrować się na opracowaniu innnego typu otoczeń.

Literatura

[1] Gupta J.N.D., Stafford S., *Flowshop scheduling research after five decades*. European Journal of Operational Research, 169, 2006, 699–711.
[2] Bansal S.P., *Minimizing the sum of completion times of n-jobs over m-machines in a flowshop – a branch and bound approach*. AIIE Transactions, 9, 1977, 306–311.
[3] Allahverdi A., Aldowaisan T., *New heuristics to minimize total completion time in m-machine flowshops.* International Journal of Production Economics, 77, 2002, 71–83.
[4] Rajendran C., Ziegler H., *Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs*. European Journal of Operational Research, 155, 2004, 426–438.
[5] Reeves C.R., Yamada T., *Genetic algorithms, path relinking and the flowshop sequencing problem*. Evolutionary Computation, 6, 1998, 45–60.
[6] Tasgetiren M.F., Liang Y., Sevkli M., Gencyilmaz G., *A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem.* European Journal of Operational Research, 177, 2007, 1930–1947.
[7] Bożejko W., Pempera J., Smutnicki A., *Parallelization of metaheuristics for the flow shop scheduling problem.* Lecture Notes in Computer Science (w druku).
[8] Tandon M., Cummings P.T., Levan M.D., *Flowshop sequencing with non-permutation schedules*. Computers and Chemical Engineering, 15, 1991, 601–607.
[9] Lin S.W., Ying K.C., *Applying a hybrid simulated annealing and tabu search approach to non-permutation flowshop scheduling problems*. International Journal of Production Research, 45, 5, 2009, 1411–1424.
[10] Grabowski J., Nowicki E., Smutnicki C., *Metoda blokowa w zagadnieniach szeregowania zadań*. EXIT, Warszawa 2003.
[11] Nowicki E., Smutnicki C., *A fast taboo search algorithm for job shop problem*. Management Science, 42,6, 1996, 797–813.
[12] Watson J.P., Howe A.E., Whitley L.D., *Deconstruction Nowicki and Smutnicki's i-TSAB algorithm for the job shop problem*. Computers and Operations Research, 33,9, 2006, 2623–2644.
[13] Taillard E., *Benchmarks for basic scheduling problems*. European Journal of Operational Research, 64, 1993, 278–285.
[14] Nawaz E., Enscore E.E., Ham I., *A heuristic algorithm for the m–machine, n–job flow–shop sequencing problem*. Omega International Journal of Management Science, 11, 1993, 91–95.