ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2008
Seria: AUTOMATYKA z. 150 Nr kol. 1796

Mariusz MAKUCHOWSKI
Politechnika Wrocławska

PROBLEM GNIAZDOWY Z OGRANICZENIEM BEZ CZEKANIA. ALGORYTMY KONSTRUKCYJNE

Streszczenie. W pracy opisana jest pewna procedura konstrukcji rozwiązania dla problemu gniazdowego z ograniczeniem bez czekania. Pokazano także sposób jej wykorzystania do budowy bardzo wydajnych algorytmów zarówno konstrukcyjnych, jak i poprawiających. W pracy prezentuje się algorytmy konstrukcyjne, bazujące na wspomnianej procedurze. Ich efektywność przebadano na dobrze znanych w literaturze przykładach testowych.

THE NO–WAIT JOB SHOP PROBLEM. HEURISTIC ALGORITHMS

Summary. This paper describes some solution constructive procedure dedicated to a job shop problem with the no–wait constraint and a makespan criterion. Very efficient heuristic algorithms based on the presented procedure are discussed. The efficiency of the algorithms is tested on well–known literature benchmarks.

1. Wprowadzenie

W pracy analizuje się silnie NP–trudny problem gniazdowy z dodatkowym ograniczeniem bez czekania z kryterium optymalizacji będącym momentem zakończenia wykonywania wszystkich zadań. Omawiany problem wywodzi się z klasycznego problemu gniazdowego przez nałożenie dodatkowych wymogów dotyczących rozwiązania dopuszczalnego. Różnica pomiędzy analizowanym problemem a jego klasycznym odpowiednikiem polega na konieczności wykonywania kolejnych operacji danego zadania dokładnie w momencie zakończenia się ich poprzedników technologicznych. Wymóg ten jest często spotykany w rzeczywistych procesach produkcyjnych, w których to obrabiane elementy z biegiem czasu zmieniają swoje parametry fizyczno–chemiczne. Przykładowo, podczas wyrobu elementów stalowych należy podgrzać surowce do odpowiednio wysokiej temperatury, aby następnie przystąpić do formowania gotowych elementów. Proces formowania musi rozpocząć się dokładnie w momencie otrzymania gorącego półproduktu [1]. Podobnie podczas produkcji niektórych produktów spożywczych, ze względów zarówno sanitarnych, jak i zachodzących procesów chemiczno–biologicznych, niezbędne jest wykonywanie jednej czynności bezpośrednio po drugiej [2]. Teoria złożoności obliczeniowej klasyczny problem gniazdowy klasyfikuje do grupy problemów silnie NP–trudnych; dowód zawarty jest w pracy [3]. W tej samej pracy pokazane jest, iż w przypadku dodania ograniczenia bez czekania problem także

należy do tej samej klasy – problemów silnie NP–trudnych. Choć teoretyczna złożoność obu wariantów analizowanego problemu jest identyczna, to wielu badaczy uważa, z praktycznego punktu widzenia, problem z ograniczeniem bez czekania za znacznie trudniejszy. Wynika to między innymi z faktu, iż rozmiar instancji, które udaje się rozwiązać dokładnie, np. metodą podziału i ograniczeń, jest znacznie mniejszy dla wariantu z ograniczeniem bez czekania. Przykładowo, w pracy [4] pokazano algorytm znajdujący rozwiązanie optymalne w sensownym czasie dla instancji problemu bez czekania, w których rozmiar nie przekraczał 15 zadań wykonywanych na 5 maszynach lub 10 zadań wykonywanych na 10 maszynach. Ponadto różnego rodzaju algorytmy przybliżone, algorytm oparty na technice poszukiwań z zabronieniami [5], algorytm oparty na technice symulowanego wyżarzania [6] oraz algorytm genetyczny z elementami symulowanego wyżarzania [7] dostarczają rozwiązań z dość dużym błędem liczonym względem rozwiązań referencyjnych.

Na początku pracy omawia się pewną procedurę (zwaną procedurą H) dedykowaną problemowi gniazdowemu z ograniczeniem bez czekania. Prezentowana procedura H ma szerokie zastosowanie przy budowie zarówno algorytmów konstrukcyjnych, jak i algorytmów popraw. Procedura ta przyjmuje jako parametr permutację zadań zwaną dalej permutacją ładującą, może więc ona umożliwić uzyskanie do $r!$ różnych rozwiązań, gdzie r jest liczbą szeregowanych zadań. Stosowanie procedury H gwarantuje, iż dowolne ustawienie parametru sterującego uzyskanego przez algorytm nadrzędny zawsze wygeneruje rozwiązanie dopuszczalne. Generowane rozwiązania są stosunkowo wysokiej jakości w sensie wartości funkcji celu. Niestety, zdarza się, iż podejście takie uniemożliwia znalezienie rozwiązania optymalnego, nawet dla najlepiej dobranej permutacji ładującej.

W pracy proponuje się dwa algorytmy HBB i $HNEH$ bazujące na wspomnianej procedurze H. Prezentowane algorytmy zostały przebadane numerycznie na literaturowej grupie przykładów testowych. Pracę kończą wnioski autora na temat procedury H oraz proponowanych algorytmów.

2. Sformułowanie problemu

Problem gniazdowy z ograniczeniem bez czekania można zamodelować jak klasyczny problem gniazdowy, wprowadzając dodatkowy warunek dotyczący dopuszczalności rozwiązania. Klasyczny problem gniazdowy można sformułować w następujący sposób:

Dany jest zbiór zadań $J = \{1, \ldots, r\}$, zbiór operacji $O = \{1, \ldots, n\}$ oraz zbiór maszyn $M = \{1, \ldots, m\}$. Zbiór operacji podzielony jest na r rozłącznych podzbiorów odpowiadających poszczególnym zadaniom. Zadanie $k \in J$ utożsamia się z zadaną sekwencją o_k, operacji; $\sum_{k \in J} o_k = n$. Sekwencja ta określa wymaganą kolejność wykonywania operacji zadania o numerze k i dla uproszczenia notacji przyjmuje się, że ma ona następującą postać: $j_k + 1, j_k + 2, \ldots, j_k + o_k$, gdzie $j_k = \sum_{i=1}^{k-1} o_i$ jest liczbą operacji w zadaniach $1, 2, \ldots, k-1$ oraz $j_0 = 0$. Dodatkowo dla ułatwienia dalszej notacji ciąg operacji zadania k, tj. $(j_k + 1, j_k + 2, \ldots, j_k + o_k)$, oznaczmy przez O_k. Ponadto dla każdej operacji $h \in O$ określona jest jedna maszyna $\mu(h)$, na której ma być ona wykonywana w czasie $p_h > 0$. Dodatkowo w modelu obowiązują trzy typowe, dla tego rodzaju problemów, ograniczenia:

- każda maszyna może wykonywać w danej chwili nie więcej niż jedną operację,
- nie można jednocześnie wykonywać więcej niż jednej operacji danego zadania w każdej chwili,
- wykonywanie operacji na maszynie nie może być przerywane.

Uszeregowanie dopuszczalne definiowane jest przez momenty rozpoczęcia $S(h) \geqslant 0$ oraz momenty zakończenia $C(h) = S(h) + p(h)$ wykonywania operacji $h \in O$ takie, że wszystkie powyższe ograniczenia są spełnione. Problem polega na znalezieniu uszeregowania dopuszczalnego minimalizującego moment wykonania wszystkich operacji $\max_{h \in O} C(h)$. Zakładając dodatkowy wymóg:

- bez czekania – każda nie pierwsza operacja danego zadania musi rozpocząć się dokładnie w momencie zakończenia wykonywania operacji wcześniejszej tego samego zadania,

otrzymujemy model problemu gniazdowego z ograniczeniem bez czekania oznaczanego w notacji Grahama [8] przez $J|no - wait|C_{\max}$.

Zauważmy teraz, że termin t_k rozpoczęcia wykonywania zadania k (moment rozpoczęcia wykonywania pierwszej operacji tego zadania; $t_k = S(j_k)$) określa precyzyjnie terminy rozpoczęć oraz zakończeń wykonywania wszystkich operacji $h \in O_k$ tego zadania. Dla przypadku bez czekania rozwiązanie problemu może więc być reprezentowane przez wektor $T = (t_1, t_2, \ldots, t_r)$ terminów rozpoczęcia wykonywania wszystkich zadań.

3. Procedura wykonawcza H

Ogólna idea proponowanego algorytmu polega na tworzeniu rozwiązania przez systematyczne dokładanie zadań do częściowo utworzonego we wcześniejszych iteracjach uszeregowania. Parametrem sterującym, mogącym być jednocześnie zmienną decyzyjną dla algorytmu nadrzędnego, jest kolejność szeregowania zadań. Kolejność ta zwana jest dalej permutacją ładującą i oznaczana przez $\pi = \pi(1), \pi(2), \ldots, \pi(r)$, $\pi(i) \in J$, $1 \leqslant ß \leqslant r$. Zbiór wszystkich możliwych permutacji ładujących oznaczamy przez Π. W i–tej iteracji szeregowaniu poddawane zostaje zadanie o numerze $\pi(i)$. Proces dodania zadania $\pi(i)$ do częściowego harmonogramu polega na wyznaczeniu wszystkich momentów rozpoczęcia $S(h)$ oraz momentów zakończenia $C(h)$ wykonywania się każdej operacji tego zadania $h \in O_{\pi(i)}$. Raz uszeregowane zadanie nie zmienia już swego położenia w harmonogramie, tzn. wyznaczone w i–tej iteracji momenty $S(h)$ i $C(h)$, $h \in O_{\pi(i)}$ będą momentami rozpoczęcia i zakończenia wykonywania się tych operacji w końcowym uszeregowaniu. W i–tej iteracji sposób wyznaczenia momentów rozpoczęcia i zakończenia operacji zadania oblicza się względem możliwie najmniejszego (spełniającego wszystkie wymagane ograniczenia) momentu $t_{\pi(i)} \geqslant 0$ rozpoczęcia wykonywania się zadania. Zauważmy, że każda permutacja ładująca (permutacja zbioru J) definiuje dokładnie jedno uszeregowanie, co więcej, uszeregowanie to jest zawsze dopuszczalne, a wynika to bezpośrednio ze sposobu jego konstrukcji. Złożoność obliczeniowa przedstawionej procedury H wynosi $O(n^2N)$, gdzie N oznacza największą liczbę operacji wykonywanych na tej samej maszynie w pojedynczym zadaniu; $N = max_{j \in J}\, max_{k \in M} |\{h : \mu(h) = k, h \in O_j\}|$. W literaturowych przykładach testowych $N = 1$.

4. Algorytm HBB

Algorytm HBB jest algorytmem dwupoziomowym, w którym procedura nadrzędna, oparta na metodzie podziału i ograniczeń, wybiera najlepsze możliwe wysterowanie dla procedury H. Złożoność samego algorytmu sterującego jest ponadwielomianowa. Praktyczne zastosowanie algorytmu HBB kończy się na przykładach o rozmiarze do 10 zadań. Niemniej dla niniejszej pracy opisywany tu algorytm jest bardzo ważny, bo za jego pomocą można oszacować średnią odległość (w sensie wartości funkcji celu od rozwiązania optymalnego), na jaką mogą się zbliżać się algorytmy oparte na rozwiązaniach generowanych procedurą H.

5. Algorytm $HNEH$

Konstrukcyjny algorytm $HNEH$ jest także algorytmem dwupoziomowym. Procedura H generująca rozwiązanie została opisana powyżej, natomiast członem sterującym jest algorytm NEH [10]. Algorytm NEH dedykowany jest permutacyjnemu problemowi przepływowemu, w którym to problemie (przy typowym modelu permutacyjno–grafowym) zmienną decyzyjną jest permutacja zadań. Ponieważ parametrem sterującym algorytmu wykonawczego H jest także permutacja zadań, możliwe było bezpośrednie zastosowanie algorytmu NEH w jego oryginalnej formie. Niestety, w związku z brakiem wykrycia praktycznych relacji pomiędzy permutacjami sterującymi a długością otrzymywanych uszeregowań autorowi nie udało się stworzyć akceleracji analogicznych do akceleracji powszechnie stosowanych w przypadku permutacyjnego problemu przepływowego. Ostatecznie złożoność obliczeniowa algorytmu $HNEH$ wynosi $O(r^2n^2N)$.

6. Badania numeryczne

Jak już wcześniej wspominano, nawet sprawdzenie wszystkich możliwych permutacji ładujących prezentowanej procedury H nie daje gwarancji znalezienia rozwiązania optymalnego. Niestety, autorowi nie udało się przeprowadzić teoretycznej analizy najgorszego przypadku tego zagadnienia. Jednakże wykonane badania numeryczne dają wyobrażenie o skali tego zjawiska. Interesujące jest zarówno, jak często zdarza się, aby najlepsze możliwe do otrzymania rozwiązanie (oznaczmy je α^{**}) nie było rozwiązaniem optymalnym, oznaczonym α^{*} oraz jaki jest średni błąd wartości funkcji celu rozwiązań α^{**} względem wartości funkcji celu rozwiązań optymalnych. Żeby odpowiedzieć sobie na tak postawione pytania, należy dysponować możliwie dużą serią instancji o znanych wartościach funkcji celu rozwiązań optymalnych oraz należy znać wartości funkcji celu najlepszych możliwych do wygenerowania rozwiązań α^{**}. W tym celu posłużono się 22 literaturowymi przykładami znanymi pod nazwami La01–La05, La16–La20, Orb01–Orb10, Ft06 i Ft10. Dla tych przykładów znane są rozwiązania optymalne osiągnięte algorytmem opartym na technice podziału i ograniczeń [4]. Równocześnie liczba zadań w tych przykładach nie przekracza 10, $r \leqslant 10$, co umożliwia zastosowanie algorytmu HBB.

Niech C^{alg} oznacza wartość długości uszeregowania rozwiązania otrzymanego algorytmem alg. Badano otrzymane wartości funkcji celu względem wartości C^{OPT}

Tabela 1

Wartości funkcji celu oraz błąd $\delta[\%]$ względem rozwiązań optymalnych

Przykład	$r \times m$	C^{OPT}	C^{HBB}	C^{VNS}	C^{HNEH}	δ^{HBB}	δ^{VNS}	δ^{HNEH}
ft06	6×6	73	73	73	73	0,0	0,0	0,0
ft10	10×10	1607	1810	1607	1683	12,6	0,0	4,7
la01		971	1064	975	1273	9,6	0,4	31,1
la02		937	1011	961	1116	7,9	2,6	19,1
la03	10×5	820	973	820	898	18,7	0,0	9,5
la04		887	1020	887	973	15,0	0,0	9,7
la05		777	861	781	861	10,8	0,5	10,8
la16		1575	1673	1575	1905	6,2	0,0	21,0
la17		1371	1556	1384	1492	13,5	0,9	8,8
la18	10×10	1417	1713	1417	1910	20,9	0,0	34,8
la19		1482	1786	1482	1903	20,5	0,0	28,4
la20		1526	1795	1526	1526	17,6	0,0	0,0
orb01		1615	1927	1615	1799	19,3	0,0	11,4
orb02		1485	1814	1485	1684	22,2	0,0	13,4
orb03	10×10	1599	1603	1599	1871	0,3	0,0	17,0
orb04		1653	1898	1653	2006	14,8	0,0	21,4
orb05		1363	1490	1370	1499	9,3	0,5	10,0
orb06		1555	1790	1555	1912	15,1	0,0	23,0
orb07		689	777	705	823	12,8	2,3	19,4
orb08	10×10	1319	1319	1319	1420	0,0	0,0	7,7
orb09		1445	1742	1445	1744	20,6	0,0	20,7
orb10		1557	1843	1557	1760	18,4	0,0	13,0
Średnia						**13,0**	**0,3**	**15,2**

optymalnych rozwiązań odczytanych z pracy [4, 7]. Wartości C^{VNS} dla poszczególnych instancji zaczerpnięto z prac [9, 7], a wartość C^{HBB} i C^{HNEH} otrzymano, stosując prezentowane algorytmy. Następnie dla każdej instancji policzono względne błędy rozwiązań uzyskane kolejnymi algorytmami dla $C^{REF} = C^{OPT}$,

$$\delta^{alg} = \frac{C^{alg} - C^{REF}}{C^{REF}} \cdot 100\%, \quad alg \in \{VNS, HBB, HNEH\}. \tag{1}$$

Otrzymane rezultaty zostały zawarte w tabeli 1. Dodatkowo w tabeli tej zawarta jest informacja o rozmiarze instancji oraz średnie wartości analizowanych błędów.

W drugim eksperymencie porównano jakość generowanych rozwiązań otrzymanych algorytmem $HNEH$ w stosunku do rozwiązań uzyskanych przez algorytm VNS. Badania przeprowadzone zostały na 50 instancjach testowych zwanych w literaturze la06–la15, la21–la40, swv01–swv20. Instancje te są znacznie większe niż instancje użyte w pierwszym teście, liczba zadań zmienia się w nich od 15 do 50. Ze względu na nieznaną wartość rozwiązań optymalnych jakość algorytmów mierzona jest względem rozwiązań referencyjnych zaczerpniętych z pracy [7]. Brak przeprowadzonej analizy czasów pracy algorytmu $HNEH$ wynika z ograniczonego rozmiaru publikacji. Napiszę tylko, iż sumaryczny czas obliczeń wszystkich prezentowanych w pracy przykładów wynosił około 5 s (na komputerze klasy PC z procesorem PentiumD 2.8GHz).

Tabela 2

Wartości funkcji celu oraz błąd δ[%] względem rozwiązań referencyjnych

Przykład	C^{REF}	C^{VNS}	C^{HNEH}	δ^{VNS}	δ^{HNEH}	Przykład	C^{REF}	C^{VNS}	C^{HNEH}	δ^{VNS}	δ^{HNEH}
15 × 5						15 × 15					
la06	1863	1431	1453	-23.2	-22.0	la36	4526	3199	3140	-29.3	-30.6
la07	1965	1366	1358	-30.5	-30.9	la37	4766	3665	3778	-23.1	-20.7
la08	1910	1390	1424	-27.2	-25.4	la38	4426	3002	3300	-32.2	-25.4
la09	2189	1586	1580	-27.5	-27.8	la39	4295	3347	3401	-22.1	-20.8
la10	2045	1527	1578	-25.3	-22.8	la40	4099	3365	3265	-17.9	-20.3
20 × 5						20 × 10					
la11	2821	1915	1906	-32.1	-32.4	swv01	3824	2631	2558	-31.2	-33.1
la12	2434	1694	1623	-30.4	-33.3	swv02	3800	2705	2751	-28.8	-27.6
la13	2650	1907	1843	-28.0	-30.5	swv03	3797	2703	2636	-28.8	-30.6
la14	2662	2313	1893	-13.1	-28.9	swv04	3895	2735	2892	-29.8	-25.8
la15	2765	1898	1996	-31.4	-27.8	swv05	3836	2734	2627	-28.7	-31.5
15 × 10						20 × 15					
la21	3574	2496	2451	-30.2	-31.4	swv06	5163	3757	3905	-27.2	-24.4
la22	2879	2210	2213	-23.2	-23.1	swv07	5076	3505	3813	-30.9	-24.9
la23	3228	2397	2506	-25.7	-22.4	swv08	5547	3943	4022	-28.9	-27.5
la24	3075	2455	2339	-20.2	-23.9	swv09	5117	3705	3647	-27.6	-28.7
la25	2985	2344	2332	-21.5	-21.9	swv10	5347	3898	4104	-27.1	-23.2
20 × 10						50 × 10					
la26	4268	3194	3228	-25.2	-24.4	swv11	9258	5878	6107	-36.5	-34.0
la27	4674	3286	3384	-29.7	-27.6	swv12	9243	6023	6156	-34.8	-33.4
la28	4478	3134	3210	-30.0	-28.3	swv13	9480	5967	6418	-37.1	-32.3
la29	4117	2838	2913	-31.1	-29.2	swv14	9450	5834	6263	-38.3	-33.7
la30	4097	3217	3167	-21.5	-22.7	swv15	9247	5743	6078	-37.9	-34.3
30 × 10						50 × 10					
la31	6519	4070	4286	-37.6	-34.3	swv16	10659	6899	6773	-35.3	-36.5
la32	6912	4476	4918	-35.2	-28.8	swv17	10003	6522	6380	-34.8	-36.2
la33	6454	4361	4472	-32.4	-30.7	swv18	10102	6470	6618	-36.0	-34.5
la34	6380	4447	4294	-30.3	-32.7	swv19	10055	6697	6888	-33.4	-31.5
la35	6563	4378	4356	-33.3	-33.6	swv20	10663	6509	6615	-39.0	-38.0
Średnia										**-29.4**	**-28.7**

7. Podsumowanie

Z przeprowadzonych badań wynika, iż sytuacje, w których nie można osiągnąć rozwiązania optymalnego procedurą H, to około 30% przykładów. Jednakże średnia odległość w sensie wartości funkcji celu rozwiązań najlepszych możliwych do osiągnięcia przez procedurę H do rozwiązań optymalnych jest mała i wynosi około 0,3%. Przeprowadzone testy dają wyobrażenie o stosunkowo niewielkiej niedogodności zjawiska pomijania rozwiązań optymalnych przez algorytmy bazujące na tej procedurze. Przykładowy algorytm $HNEH$ powstały ze znanego algorytmu NEH i procedury H daje rozwiązania prawie 30% lepsze niż literaturowe heurystyki konstrukcyjne. Ponadto jakość generowanych rozwiązań jest na poziomie algorytmu zstępującego, jakim jest VNS.

BIBLIOGRAFIA

1. Wismer D.A.: Solution of the flowshop scheduling-problem with no intermediate queues. Operations Research. 20, 1972, p. 689.
2. Hall N., Sriskandarajah C.: A survey of machine scheduling-problems with blocking and no-wait in process. Operations Research. 44(3), 1996, p. 510.
3. Lenstra J., Rinnooy Kan A.: Computational complexity of discrete optimization problems. Annals of Discrete Mathematics, 4, 1979, p. 121.
4. Mascis A., Pacciarelli D.: Job shop scheduling with blocking and no-wait constraints, European Journal of Operational Research. 142(3), 2002, p. 498.
5. Macchiaroli R., Mole S., Riemma S.: Modelling and optimization of industrial manufacturing processes subject to no-wait constrains, International Journal of Production Research, 37(11), 1999, p. 2585.
6. Raaymakers W., Hoogeveen J.: Scheduling multipurpouse batch process industries with no-wait restrictions by simulated annealing, European Journal of Operational Research. 126, 2000, p. 131.
7. Schuster C., Framinan J.: Approximative procedures for no-wait job shop scheduling, Operations Research Letters. 31, 2003, p. 308.
8. Graham R., Lawler E., Lenstra J., Rinnooy Kan A.: Optimization and approximation in deterministic sequencing and scheduling: a survey, Annals of Discrete Mathematics, 5, 1979, p. 287.
9. Hansen P., Maldenovic N.: Variable neighborhood search. Principles and applications, European Journal of Operational Research. 130, 2001, p. 449
10. Nawaz M., Enscore Jr.E.E., Ham I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, OMEGA International Journal of Management Science, 11, 1983, p. 91

Recenzent: Prof. dr hab. inż. Marek Kubale

Abstract

The paper deals with some solution constructive procedure for a job shop problem with the no–wait constraint and a makespan criterion. The solution constructive procedure, called H procedure, is controlled by a job permutation, called loading permutation. The H procedure generates $r!$ (r – is the number of tasks) high quality, in criterion sense, solutions.

Algorithms given in the paper (HBB and $HNEH$) consist of two levels. At the high level, there is a certain heuristic algorithm called the master algorithm and at the low level, there is the H procedure. The master algorithm chooses a loading permutation for the H procedure. In HBB the master algorithm is based on a brand and bound method, in $HNEH$ the master algorithm is a well–known from literature the NEH algorithm. Computational results show very high efficiency of the proposed algorithms.