ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1980

Seria: AUTOMATYKA z. 55 Nr kol. 651

Czesław Smutnicki, Józef Grabowski
Instytut Cybernetyki Technicznej
Politechniki Wrocławskiej

METODY I ALGORYTMY OPTYMALIZACJI KOLEJNOŚCI OPERACJI Z KRYTERIUM MINIMALIZACJI KOSZTÓW

Streszczenie. W pracy przedstawiono ocenę metod i algorytmów szeregowania niepodzielnych zadań na realizatorach z uwzględnieniem kosztów. Zaproponowano nowe algorytmy i podano przykłady zastosowań.

1. Wstęp

W artykule jest rozpatrywany dyskretny system produkcyjny realizujący niepodzielne zadania w ustalonym porządku technologicznym. Zadania te należy wykonać dysponując pewnymi niezużywalnymi środkami (maszyny, brygady ludzi itp.). Należy dokonać takiego rozdziału środków pomiędzy czynności (zadania) oraz określić taką kolejność ich wykonywania, aby całkowity koszt, związany z użyciem środków i terminowością realizacji zadań, był minimalny.

Problemy szeregowania uwzględniające koszty stanowią dość ważną grupę zagadnień z praktycznego (i ekonomicznego) punktu widzenia. Fakt, że stosunkowo niedawno uzyskano istotne wyniki w tej dziedzinie, można usprawiedliwić dużymi trudnościami przy konstrukcji "efektywnych" algorytmów.

W pracy przedstawiono w skrócie aktualny stan metod i algorytmów służących do ich rozwiązania. Ponadto zaproponowano nowe algorytmy zagadnień.

2. Podstawowe definicje, klasyfikacja problemów

Niech $J = \{J_1, J_2, \ldots, J_n\}$ będzie zbiorem zadań, które mają być wykonane przy użyciu zbioru maszyn różnych typów

$$M = \{M_1, M_2, \ldots, M_m\}.$$

Każde zadanie $J_i \in J$ składa się z ciągu operacji

$$J_i = \langle O_{i1}, O_{i2}, \ldots, O_{in_i} \rangle .$$

Operacje mają być wykonywane w zadanym porządku technologicznym, tzn. każda operacja O_{in} ma być wykonywana po wykonaniu operacji O_{in-1}, a przed wykonaniem operacji O_{in+1}. Każda maszyna może wykonywać co najwyżej jedno zadanie w danej chwili czasowej. Dla każdego zadania są okreś-

lone następujące pojęcia: n_i - liczba operacji, p_{in} - czas wykonywania operacji O_{in}, r_i - najwcześniejszy możliwy czas rozpoczęcia wykonywania zadania, d_i - żądany czas zakończenia zadania, S_i - czas rozpoczęcia wykonywania zadania, $E_i = \max[0, r_i - S_i]$ - przyspieszenia wykonania zadania, $T_i = \max[0, C_i - d_i]$ - spóźnienia wykonania zadania, $f_i : C_i \to \mathcal{R}$ - niemalejąca funkcja reprezentująca koszt związany z wykonaniem zadania w czasie C_i, $\mathcal{R}$ - zbiór liczb rzeczywistych, w_i - współczynnik wagowy zadania.

Zagadnienia sklasyfikowano [8] oznaczając symbolami

$$n|m|\gamma,\Gamma|\delta$$

gdzie: n - liczba zadań, m - liczba maszyn, $\gamma \in \{P, F, G, I\}$ - oznacza typ zagadnienia: F-zagadnienie taśmowe, P-permutacyjne zagadnienie taśmowe, G-zagadnienie gniazdowe, I-zagadnienie z równoległymi maszynami, $\Gamma \subset \{$przerywać, $r_i \geq 0$, $p_i \leq p_*$, $p_i = 1$, drzewo, $\prec\}$ - - oznacza istnienie dodatkowych założeń ograniczających, δ - postać funkcji celu $\sum f_i(C_i) = \sum_{i=1}^{n} f_i(C_i)$ albo $f_{max} = \max_i f_i(C_i)$.

W tablicy 1 zebrano problemy, dla których znane są algorytmy. Podano także źródło literaturowe i złożoność obliczeniową. Ze względu na istotne trudności napotykane przy konstrukcji algorytmów w literaturze rozważa się głównie problemy jednomaszynowe.

Tablica 1

problem	autorzy algorytmu	złożoność obliczeniowa	źródło
$n\|1\|\prec\|f_{max}$	Lawler [1973]	$o(n^2)$	[12]
$n\|1\|\prec^{(1)}\|\sum C_i$	Conway, Maxwell, Miller [1967]		[4]
$n\|1\|\|\sum w_i C_i$	Smith [1956]	o(nlogn)	[21]
$n\|1\|$drzewo$\|\sum w_i C_i$	Horn [1972]	o(nlogn)	[10]
$n\|1\|\prec^{(1)}\|\sum w_i C_i$	Sidney [1975]	$o(n^2)$	[18]
$n\|1\|r_i \geq 0, p_i = 1\|\sum w_i C_i$	Rinnooy [1976]	o(nlogn)	[16]
$n\|1\|r_i \geq 0, p_i = 1\|\sum w_i T_i$	Lawler [1964]	$o(n^2)$	[11]
$n\|1\|\|\sum T_i$	(i) Elmaghraby [1968]	NP-zupełny	[5]
	(ii) Emmons [1969]		[6]
	(iii) Srinivasan [1971]		[22]
	(iv) Baker, Martin [1974]		[2]
	(v) Fisher [1976]		[7]
$n\|1\|\|\sum w_i T_i$	Picard, Queyranne [1978]	NP-zupełny	[14]
$n\|1\|\|\sum f_i(C_i)$	(i) Schild, Fredman [1962]	NP-zupełny	[17]
	(ii) Rinnooy Kan, Lageweg, Leustra [1975]		[15]
	(iii) Shwimer [1972]		[16]

Tablica 1 - cd.

problem	autorzy algorytmu	złożoność obliczeniowa	źródło
$n\|1\|<\|\sum f_i(C_i)$	Baker, Schrage [1978]	NP-zupełny	[3]
$n\|1\|\|\sum f_i(C_i)$ (2)	Lawler, Sivazlian [1978]	o(nlogn)	[13]
$n\|1\|\|h_{max}$ (3)	Sidney [1977]	$o(n^2)$	[19]
$n\|m\|P\|h_{max}$	Achuthan, Grabowski, Sidney [1978]	NP-zupełny	[1]
$n\|m\|P\|f_{max}$	Townsend [1977]	NP-zupełny	[23]

(1) relacja poprzedzania przedstawia graf szeregowo-równoległy ; dla arbitralnie przyjętej relacji poprzedzania problem jest NP-zupełny

(2) wszystkie funkcje $f_i(C_i)$ są postaci $f_i(C_i) = \int_{C_i-p_i}^{C_i} k_i(u)du$, gdzie $k_i(u) = a_i\varphi(u) + b_i$

(3) przy założeniu $r_i \leqslant r_j \Rightarrow d_i \leqslant d_j$. W ogólnym przypadku zagadnienie jest NP-zupełne.

3. Metody rozwiązywania zagadnienia $n|1||\sum f_i(C_i)$

Problem ten w ogólnym przypadku, bez dodatkowych założeń o rodzaju ograniczeń i postaci funkcji celu, jest problemem NP-zupełnym. W przypadku, gdy funkcje $f_i(C_i)$ są postaci

$$f_i(C_i) = \int_{C_i-p_i}^{C_i} k_i(u)du \tag{3.1}$$

gdzie $k_i(u) = a_i\varphi(u) + b_i$ są niemalejącymi (nierosnącymi), to rozwiązaniem optymalnym jest wykonywanie zadań w porządku nierosnących (niemalejących) a_i. Funkcje celu $f_i(C_i) = C_i$, $f_i(C_i) = w_iC_i$ są szczególnymi przypadkami (3.1) [13].

Dodanie ograniczeń w postaci arbitralnie przyjętej relacji poprzedzeń pomiędzy zadaniami powoduje, że problem staje się NP-zupełny nawet, gdy wszystkie $f_i(C_i) = C_i$, dlatego też udało się skonstruować efektywne algorytmy jedynie dla przypadków, gdy relacja poprzedzeń jest drzewem [10] lub grafem szeregowo-równoległym [4], [18].

Szczególnie ważne z praktycznego punktu widzenia są problemy, których funkcja celu przedstawia karę (lub straty) związaną ze spóźnieniem zadań T_i. W tym przypadku $f_i(C_i)$ są niemalejącymi funkcjami spóźniania

$$f_i(C_i) = h_i(\max[0, C_i-d_i]) = h_i(C_i) \tag{3.2}$$

Ponieważ tak sformułowany problem jest NP-zupełny [8], do jego rozwiązania stosowane są najczęściej następujące podejścia:

1) programowanie dynamiczne [2], [3],

2) algorytmy oparte na metodzie podziału i ograniczeń [5], [6], [7], [14], [17], [22],

3) metody subgradientowej optymalizacji [7], [9],

4) metody heurystyczne [24].

3.1. Programowanie dynamiczne. Niech J oznacza zbiór wszystkich zadań i niech I oznacza pewien podzbiór $I \subset J$. Niech $I' = J - I$ oznacza uzupełnienie I oraz $q_I = \sum_{k \in I'} p_k$ oznacza całkowity czas potrzebny do wykonania zadań z I'. Niech F(I) oznacza minimalną wartość funkcji celu, dla I $F(I) = \min \sum_{k \in I'} f_k(C_k)$, przy warunku, że wszystkie zadania zaczynają się nie wcześniej niż q_I. Należy znaleźć F(J) przy pomocy wzoru rekurencyjnego:

$$F(I) = \min_{k \in I'} \{f_k(q_I) + F(I-\{k\})\}$$
$$F(\emptyset) = 0. \tag{3.3}$$

Zręczna implementacja programowa [3] może skompensować pewne wady metody, jednakże w przypadku zagadnień o dużym wymiarze metoda staje się mało skuteczna. Do wad DP można zaliczyć:

a) znaczny obszar pamięci potrzebny do przechowywania informacji,

b) dużą liczbę kroków podstawowych potrzebnych do przeglądnięcia zbiorów I.

Do zalet tej metody można zaliczyć:

a) niewielką złożoność kroku podstawowego,

b) możliwość redukcji wymiaru pamięci w przypadku przyjęcia relacji poprzedzeń [3],

c) możliwość zastosowania dowolnej nieaddytywnej, niemalejącej funkcji celu typu min, max, iloczyn $f_i(\cdot)$,

d) łatwość implementacji programowej.

3.2. Metoda podziału i ograniczeń. O skuteczności metody decydują głównie:

a) kryteria eliminacji pozwalające odrzucać a priori pewne zbiory rozwiązań,

b) moc testu określającego dolne ograniczenie funkcji celu LB pozwalające odrzucać rozwiązania, gdy $LB \geq UB$, gdzie UB jest górnym ograniczeniem funkcji celu,

c) szybkie znalezienie dobrego rozwiązania w celu lepszego określenia wartości UB.

Najczęściej do konstrukcji testów używa się:

1) zagadnienia postaci $n|1|d_i = d_*, f_i = f_*|\sum f_i(C_i)$, gdzie $d_* = \min_i d_i$, $f_*(u) = \min_i f_i(u)$,

2) zagadnienia $n|1|$przerywać$|\sum f_i(C_i)$; zagadnienie to jest sprowadzalne do problemu $n|1|p_i|=p^*|\sum f_i(C_i)$, gdzie $n' = \sum_{k=1}^{n} p/p^*$ [11], [13],

3) zagadnienia przydziału pracy (AP) z odpowiednio określonymi współczynnikami kosztu [15],

4) zagadnienia czasowo-zależnego komiwojażera (TDTSP) z odpowiednio określoną macierzą kosztu [14]. Ponieważ problem TDTSP jest NP-zupełny, korzysta się z dolnego ograniczenia problemu znalezionego np. metodą subgradientową [14],[9],

5) problemu Lagrange'a [7], gdzie zagadnienie dualne jest rozwiązywane metodą subgradientową [9].

Kryteria eliminacji dla problemów kosztowych bazują zwykle na szczególnych postaciach funkcji $f_i(C_i)$ [6], [14], [15], [20], [22]. Biorąc pod uwagę znane rezultaty, metoda B-and-B jest aktualnie najbardziej skuteczną metodą rozwiązywania zagadnień tego typu.

3.3. Metoda subgradientowa. Ze względu na swe własności metoda rzadko jest używana samodzielnie, częściej natomiast jest wykorzystywana przy konstrukcji testów [9].

4. Nowy algorytm zagadnienia $n|1||\sum f_i(C_i)$

W niniejszym rozdziale przedstawimy algorytm oparty na metodzie podziału i ograniczeń posiadający następujące zalety:

a) w każdym kroku jest dane pewne dopuszczalne rozwiązanie,

b) rozwiązanie optymalne jest znajdowane stosunkowo szybko.

Algorytm ustala porządek zadań począwszy od końca do początku. Niech będzie permutacją, F^* - najlepszą wartością funkcji celu, I - zbiorem zadań, których porządek został ustalony, J - zbiorem wszystkich zadań. Niech będzie dana pewna permutacja początkowa π^0.

Algorytm

Przyjąć $F^* = \infty$, $\pi = \pi^0$, $I = \emptyset$, $k = n$, $l = 1$,

krok 1 - obliczyć dolne ograniczenia LB wartości funkcji celu, np. stosując metodę opisaną w [7] lub [14]. Jeśli $LB \geq F^*$, to przejść do kroku[4], inaczej przejść do kroku[2].

krok 2 - obliczyć wartość funkcji celu dla permutacji

$$F(\pi) = \sum_{i=1}^{n} f_{\pi(i)}(C_{\pi(i)})$$

$$C_{\pi(i)} = \sum_{j=1}^{i} p_{\pi(j)}. \tag{4.1}$$

Jeśli $F(\pi) < F^*$, to $F^* := F(\pi)$. Określić $I' = J - I$. Jeśli $I' = \emptyset$, to

przejść do kroku 4, inaczej obliczyć lokalną ocenę zmiany kosztów przy przeniesieniu pracy $\pi(i) \in I_1'$ z pozycji i-tej na pozycję k-tą według wzoru

$$\Delta_{ik} = f_{\pi(i)}(C_{\pi(i)}) - f_{\pi(i)}(C_{\pi(k)}) +$$

$$+ \sum_{j=i+1}^{k} \left[f_{\pi(j)}(C_{\pi(j)}) - f_{\pi(j)}(C_{\pi(j)} - p_{\pi(i)}) \right] \quad \pi(i) \in I_1'$$

$$C_{\pi(i)} = \sum_{j=1}^{i} p_{\pi(j)} \tag{4.2}$$

i przejść do kroku 3.

krok 3 - Jeśli $I_1' = \emptyset$, to przejść do kroku 4, inaczej znaleźć

$$\Delta_{i^*k} = \min_{\pi(i) \in I_1'} \Delta_{ik} \tag{4.3}$$

i sprawdzić kryterium eliminacji dla prac $\pi(i^*)$ i $\pi(k)$ $i^* \neq k$ w przedziale $\left[C_{\pi(k-1)} - p_{\pi(i)}, C_{\pi(k)}\right]$. Jeśli eliminacja zachodzi, to przyjąć $I_1' = I_1' - \{\pi(i)\}$ i przejść do kroku 3. Inaczej wykonać zamianę w permutacji przenosząc pracę $\pi(i^*)$ z miejsca i^*-tego na k-te w permutacji oraz przyjąć $k = k-1$, $l = l+1$, $I = I \cup \{\pi(i^*)\}$, $I_1' = I_1' - \{\pi(i^*)\}$ i przejść do kroku 1.

krok 4 - Jeśli $l=1$, to stop F^* oraz π^* są odpowiednio wartością funkcji celu oraz permutacją rozwiązania optymalnego. Inaczej wykonać $l=l-1$, $k=k+1$, $I=I-\pi(i^*)$, gdzie $\pi(i^*)$ - jest ostatnio przenoszoną pracą z miejsca i^*-tego na k-te. Wykonać zamianę odwrotną przenosząc pracę $\pi(i^*)$ z miejsca k-tego na i^*-te. Przejść do kroku 3.

Powyższy algorytm można implementować w prosty sposób na problem $n|1|r_j \geq 0|\sum f_i(C_i)$.

5. Algorytm zagadnienia $n|1||\sum(g_i(E_i)+h_i(T_i))$

Jednym z zagadnień posiadających duże znaczenie praktyczne jest problem, w którym funkcje strat (kary) są związane z przyspieszeniem wykonywania zadań E_i oraz z ich spóźnieniem T_i. W ogólnym przypadku w rozwiązaniu optymalnym mogą istnieć przerwy (czas przestoju maszyn) pomiędzy wykonywanymi zadaniami. Ze względu na NP-zupełność problemu proponuje się metodę podziału i ograniczeń do jego rozwiązania. Stosowana jest mieszana strategia przeglądania drzewa rozwiązań, przy czym w każdym węźle istnieje pewne rozwiązanie dopuszczalne.

Niech π będzie pewną ustaloną permutacją, wówczas wartość funkcji celu można znaleźć przez rozwiązanie problemu

$$\min_{E_i,T_i} \sum_{i=1}^{n} (g_i(E_i) + h_i(T_i))$$

$$E_i + T_i \geqslant \Delta_{ii} \quad i = 1,\dots,n \tag{5.1}$$

$$E_{\pi(i)} + T_{\pi(j)} \geqslant \Delta_{ij} \quad i,j=1,\dots,n \quad i<j$$

$$E_i, T_i \geqslant 0$$

gdzie: $\Delta_{ii} = p_i - (d_i - r_i)$

$$\Delta_{ij} = \sum_{k=i}^{j} p_{\pi(k)} - (d_{\pi(j)} - r_{\pi(i)}). \tag{5.2}$$

Korzystając ze specyficznej postaci ograniczeń można pokazać, że gdy funkcje g_i, h_i są liniowe, to algorytm wyznaczający wartość funkcji celu wymaga $o(n^2 \log n)$ iteracji.

Algorytm

Przyjąć $F^* = \infty$, $\pi = \pi^0$, $I = \emptyset$, $k = n$, $l = 1$,

krok 1 - obliczyć dolne ograniczenie LB wartości funkcji celu stosując podejście [7], [14], [15]. Jeśli LB $\geqslant F^*$, to przejść do kroku 4, inaczej przejść do kroku 2.

krok 2 - obliczyć wartość funkcji celu dla permutacji π wg (5.1)-(5.2). Jeśli $F(\pi) < F^*$, to $F^* := F(\pi)$. Określić $I_1' = J - I$. Jeśli $I_1' = \emptyset$, to przejść do kroku 4, inaczej obliczyć lokalną ocenę zmiany kosztów Δ_{ik} przy przeniesieniu pracy $\pi(i) \in I_1'$ z pozycji i-tej na pozycją k-tą i przejść do kroku 3.

krok 3 - jeśli $I_1' = \emptyset$, to przejść do kroku 4, inaczej znaleźć

$$\Delta_{i^*k} = \min_{\pi(i) \in I_1'} \Delta_{ik}$$

i sprawdzić kryterium eliminacji dla prac $\pi(i^*)$ i $\pi(k)$. Jeśli eliminacja zachodzi, to przyjąć $I_1' = I_1' - \{\pi(i)\}$ i przejść do kroku 3. Inaczej wykonać zamianę w permutacji przenosząc pracę $\pi(i^*)$ z miejsca i^*-tego na k-te oraz przyjąć $k=k-1$, $l=l+1$, $I=I \cup \{\pi(i^*)\}$, $I_1'=I_1' - \{\pi(i^*)\}$ i przejść do kroku 1.

krok 4 - jeśli $l=1$, to stop F^* oraz π^* są odpowiednio wartością funkcji celu oraz permutacją rozwiązania optymalnego, w przeciwnym przypadku wykonać $l=l-1$, $k=k+1$, $I=I-\{\pi(i^*)\}$, gdzie $\pi(i^*)$ jest ostatnio przenoszoną pracą z miejsca i^*-tego na k-te. Wykonać zamianę odwrotną przenosząc pracę $\pi(i^*)$ z miejsca k-tego na i^*-te i przejść do kroku 3.

Należy się spodziewać, że podejście [7] da najlepsze wyniki w konstrukcji testu.

LITERATURA

[1] Achuthan N.R., Grabowski J., Sidney J.B.: Optimal Flow-Shop Scheduling with Earliness and Tardiness Penalties, Opsearach, to appear.

[2] Baker K.R., Martin J.B.: An Experimental Comparison of Solutions Algorithms for the Single-Machine Tardiness Problem, Naval Res. Log. Quart. 21, 187-199 (1974).

[3] Baker K.R., Schrage L.E.: Finding an Optimal Sequence by Dynamic. Programming: An Extension to Precedence-Related Tasks, OR 26, 111-120 (1978).

[4] Conway R.W., Maxwell W.L., Miller L.W.: Theory of Scheduling, Addison--Wesley, 1967.

[5] Elmaghraby S.E.: The One Machine Sequencing Problem with Delay Costs, J.Indust.Eng. 19, 105-108 (1968).

[6] Emmons H.: One Machine Sequencing to Minimize Certain Functions of Job.Tardiness, OR 17, 701-715 (1969).

[7] Fisher M.L.: A Dual Algorithm for the One-Machine Scheduling Problem, Math.Progr.11, 229-251 (1976).

[8] Grabowski J., Smutnicki C.: Zagadnienia kolejnościowe. Klasyfikacja i złożoność obliczeniowa algorytmów. Przegląd statystyczny. W druku.

[9] Held M., Wolfe P., Crowder H.: Validation of Subgradient Optimization, Math.Progr. 6, 62-88 (1974).

[10] Horn W.A.: Single Machine Job. Sequencing with Treelike Precedence Ordering and Linear Delay Penalties, SIAM J. Appl. Math. 23, 189-202 (1972).

[11] Lawler E.L.: On Scheduling Problems with Deferral Cost, Mgmt Sci 11, 280-288 (1964).

[12] Lawler E.L.: Optimal Sequencing of a Single Machine Subject to Precedence Constraints, Mgmt Sci 19, 544-546 (1973).

[13] Lawler E.L.; Sivazlian B.D.: Minimization of Time-Varying Costs in Single Machine Scheduling, OR 26, 563-569 (1978).

[14] Picard J.C., Queyranne M.: The TDTSP and its Application to the Tardiness Problem in One-Machine Scheduling, OR 26, 86-110 (1978).

[15] Rinnoy Kan A.H.G., Lenstra J.K., Lagaweg B.J.: Minimizing Total Costs in One-Machine Scheduling, OR 23, 908-927 (1975).

[16] Rinnoy Kan A.H.G.: Machine Scheduling Problems: Classification, Complexity and Computations, Nijhoff, The Hague, (1976).

[17] Schild A., Fredman F.J.: Scheduling tasks with deadlines and non-linear, loss functions, Mgmt Sci 9, 73-81 (1962).

[18] Sidney J.B.: Decomposition Algorithms for Single-Machine Sequencing with Precedence Relations and Deferral Costs, OR 23, 283-298 (1975).

[19] Sidney J.B.: Optimal Single-Machine Scheduling with Earliness and Tardiness Penalties, OR 25, 62-69 (1977).

[20] Shwimer J.: On the N-Job, One-Machine, Sequence Independent Scheduling Problem with Tardiness Penalties: A Branch-and-Bound Solution, Mgmt Sci 18, 301-313 (197).

[21] Smith W.E.: Various Optimizers for Single-Stage Production, Nav. Res. Log. Quart. 3, 59-66 (1956).

[22] Srinivasan V.: A Hybrid Algorithm for the One-Machine Sequencing Problem to Minimize Total Tardiness, Nav. Res. Log. Quart. 18, 317--327 (1971).

[23] Townsend W.: Minimizing the Maximum Penalty in the m-Machine Sequencing Problem, OR Quart. 28, 727-734 (1977).

[24] Wilkerson L.J., Irwin J.D.: An Improved Method for Scheduling Independent Tasks, AIIE Trans. 3, 239-245, (1971).

МЕТОДЫ И АЛГОРИТМЫ ОПТИМАЛИЗАЦИИ ПОСЛЕДОВАТЕЛЬНОСТИ ОПЕРАЦИЙ С МИНИМАЛЬНО--СТОИМОСТНЫМ КРИТЕРИЕМ

Р е з ю м е

В работе представлено оценку методов алгоритмов упорядочнения нераздельных задач на реализаторах с учётом минимальной стоимости. Представлено новые алгоритмы и их применения.

METHODS AND ALGORITHMS OF OPERATIONS SEQUENCE OPTIMIZATION WITH MINIMUM COST CRITERION

S u m m a r y

The review of methods and algorithms of scheduling nonpreemptive jobs with the minimum-cost criterion is presented. A new algorithms for one-machine case are proposed.