ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2006
Seria: AUTOMATYKA z. 143 Nr kol. 1726

Wojciech BOŻEJKO, Mieczysław WODECKI
Politechnika Wrocławska, Uniwersytet Wrocławski

JEDNOMASZYNOWY PROBLEM SZEREGOWANIA ZADAŃ Z PRZEZBROJENIAMI

Streszczenie. W pracy przedstawiamy algorytm populacyjny rozwiązywania jednomaszynowego problemu szeregowania zadań z przezbrojeniami, w którym należy zminimalizować sumę kosztów opóźnień. W literaturze jest on oznaczany przez $1|s_{ij}|\Sigma w_i T_i$ i należy do klasy problemów silnie NP-trudnych. Wykonaliśmy obliczenia na reprezentatywnej grupie danych testowych, a otrzymane wyniki porównujemy z najlepszymi znanymi w literaturze. Dla wielu przykładów uzyskaliśmy poprawę najlepszych rozwiązań.

SINGLE MACHINE SCHEDULING PROBLEM WITH SEQUENCE-DEPENDENT SETUP TIMES

Summary. In the paper we propose a population-based algorithm for solving single machine scheduling problem with total tardiness criterion and sequence-dependent setup times. It is represented by $1|s_{ij}|\Sigma w_i T_i$ in literature and it belongs to the strongly NP-hard class. Calculations on the representative group of benchmark instances were done and results were compared with the best known from literature. Obtained solutions were better than benchmark ones in many instances.

1. Wprowadzenie

Rozważany w pracy jednomaszynowy problem szeregowania zadań z przezbrojeniami zależnymi od kolejności wykonywania zadań z kryterium minimalizacji sumy kosztów opóźnień (oznaczany przez $1|s_{ij}|\Sigma w_i T_i$) należy do klasy problemów silnie NP-trudnych. W przeglądowej pracy Panwalkar, Dudek i Smith [6], dotyczącej szeregowania zadań, zauważono, że 75% występujących w praktyce problemów wymaga przynajmniej jednego przezbrojenia zależnego od kolejności wykonywania zadań. Natomiast w 15% problemów należy uwzględnić przezbrojenia pomiędzy wszystkimi zadaniami. Jednakże znacząca większość prac dotyczących sumo-kosztowych problemów jednomaszynowych, skupia się na klasycznym problemie $1||\Sigma w_i T_i$ (bez przezbrojeń), dla którego zaproponowano szereg algorytmów dokładnych i przybliżonych. Te pierwsze pozwalają na rozwiązywanie (w rozsądnym czasie) przykładów, w których liczba zadań jest nie większa niż 40 oraz dla pewnych

specyficznych danych 50. Ze względu na ich małą efektywność w praktyce stosuje się niemal wyłącznie algorytmy przybliżone (konstrukcyjne oraz popraw). Dla problemu $1|s_{ij}|\Sigma w_i T_i$ algorytmy konstrukcyjne [5] wyznaczają rozwiązania różniące się nawet o kilkaset procent od optymalnych. Z kolei bardzo obiecujące wyniki, otrzymane przy zastosowaniu metaheurystyk: algorytmu genetycznego [1], symulowanego wyżarzania [8], poszukiwań z tabu [7], algorytmu mrówkowego [4] oraz próbkowania stochastycznego [3] były dla nas inspiracją do konstrukcji nowego algorytmu bazującego na metodzie algorytmu ewolucyjnego. Polega on na wyznaczaniu wielu minimów lokalnych (przez poprawianie losowych rozwiązań) i następnie ich badaniu.

Ogólna idea jest oparta na następującym przypuszczeniu. Jeżeli w różnych permutacjach, będących minimami lokalnymi, na pewnych pozycjach są te same elementy, to w rozwiązaniu optymalnym elementy te będą być może także na tych samych pozycjach. Do wyznaczania minimów lokalnych stosujemy algorytm poszukiwania zstępującego (descending search). Zbiór punktów startowych w kolejnych iteracjach algorytmu jest wyznaczany w oparciu o pewne elementy algorytmu ewolucyjnego.

Algorytm startuje z dowolnej populacji – podzbioru zbioru rozwiązań. Następnie, dla każdego elementu z populacji, stosując algorytm lokalnej optymalizacji, wyznacza się minimum lokalne. W ten sposób otrzymujemy zbiór permutacji – minimów lokalnych. Jeżeli pewien element występuje w wielu minimach (jest to jeden z parametrów algorytmu) na tej samej pozycji, to zostaje on na tej pozycji ustalony (pozostałe elementy i pozycje są wolne). Nową populację (zbiór permutacji w następnej iteracji) generujemy, losując elementy wolne na pozycje wolne (na pozycjach ustalonych są już elementy ustalone). W ten sposób w każdej iteracji zwiększamy liczbę elementów ustalonych. Aby algorytm zbyt szybko nie zakończył działania (przez ustalenie wszystkich elementów), w każdej iteracji pewne „najstarsze" z ustalonych elementów są zwalniane.

2. Jednomaszynowy problem szeregowania zadań z przezbrojeniami

Dany jest zbiór n ponumerowanych zadań $N=\{1,2, \ldots ,n\}$, które należy wykonać bez przerywania, na jednej maszynie. Maszyna ta, w dowolnej chwili, może wykonywać co najwyżej jedno zadanie. Dla zadania i ($i=1,2, \ldots ,n$) niech p_i, w_i, d_i będą odpowiednio: *czasem wykonywania*, *wagą funkcji kosztów* oraz *żądanym terminem zakończenia*. Dane są także przezbrojenia s_{ij}, $i, j \in N$, reprezentujące czas potrzebny na przygotowanie maszyny do wykonywania zadania j, jeżeli bezpośrednio przed j było wykonywane zadanie i. Ponadto, s_{0i} jest czasem przygotowania maszyny, jeżeli zadanie i jest wykonywane jako pierwsze.

Dla ustalonej kolejności wykonywania zadań niech C_i ($i=1,2,...,n$) będzie terminem zakończenia wykonywania zadania i. Wówczas $T_i = \max\{0, C_i - d_i\}$ nazywamy *opóźnieniem*, a $f_i(C_i)=w_i \cdot T_i$ *kosztem opóźnienia*. Rozważany problem sprowadza się do wyznaczenia takiej kolejności wykonywania zadań, która minimalizuje *sumę kosztów opóźnień*, tj. $\sum w_i T_i$.

Niech Π będzie zbiorem permutacji elementów z N. Dla permutacji $\pi \in \Pi$, przez

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

oznaczamy *koszt permutacji* (tj. sumę kosztów opóźnień, gdy zadania są wykonywane w kolejności występowania w π), gdzie $C_{\pi(i)} = \sum_{j=1}^{i}(s_{\pi(j-1)\pi(j)} + p_{\pi(j)})$, $\pi(0) = 0$. Rozwiązanie problemu sprowadza się więc do wyznaczenia permutacji optymalnej (o minimalnym koszcie) w zbiorze wszystkich permutacji Π.

3. Hybrydowy algorytm ewolucyjny

Działanie algorytmu rozpoczyna się od wyznaczenia (zazwyczaj losowo) populacji początkowej P^0. Za suboptymalne rozwiązanie π^* przyjmujemy najlepszy element populacji P^0. Niech i będzie numerem iteracji algorytmu. Nowa i+1 populacja (tj. zbiór P^{i+1}) jest generowana w następujący sposób. Dla bieżącej populacji P^i wyznacza się zbiór minimów lokalnych LM^i (dla każdego elementu $\pi \in P^i$ wykonując procedurę $LocalOpt(\pi)$). Ustala się elementy występujące na tych samych pozycjach w minimach lokalnych, tworząc zbiór elementów i pozycji ustalonych FS^{i+1}. Każda permutacja nowej populacji P^{i+1} ma ustalone elementy (na ustalonych pozycjach) ze zbioru FS^{i+1}. Na pozostałe (wolne) pozycje są losowo wyznaczane elementy wolne. Jeżeli istnieje permutacja $\beta \in LM^i$ oraz $F(\beta) < F(\pi^*)$, to za π^* przyjmujemy β. Algorytm kończy działanie po wykonaniu z góry ustalonej liczby iteracji.

Algorytm badania minimów lokalnych *ATP*

Inicjalizacja. Wyznaczyć losową populację początkową: $P^0 = \{\pi_1, \pi_2, \ldots, \pi_\eta\}$;
$\pi^* \leftarrow$ najlepszy element populacji P^0;
i=0; {*numer iteracji*}
$FS^0 = \varnothing$; {*zbiór elementów i pozycji ustalonych*}
repeat
  Wyznaczyć zbiór minimów lokalnych $LM^i = \{\hat{\pi}_1, \hat{\pi}_2, \ldots, \hat{\pi}_\eta\}$, gdzie:
  $$\hat{\pi}_j = LocalOpt(\pi_j),\ \pi_j \in P^i;$$
  for j:=1 **to** η **do**
    if $F(\hat{\pi}_j) < F(\pi^*)$ **then** $\pi^* \leftarrow \hat{\pi}_j$;
    Wyznaczyć zbiór elementów i pozycji ustalonych oraz
    wygenerować nową populację P^{i+1};
  i=i+1;
until not Kryterium zatrzymania,

Do wyznaczania minimów lokalnych są stosowane szybkie algorytmy oparte na metodzie lokalnych popraw. Generalnie metoda ta polega na iteracyjnym polepszaniu bieżącego rozwiązania poprzez lokalne przeszukiwanie. Rozpoczyna się od pewnego rozwiązania początkowego (startowego) x^0. W każdej iteracji dla bieżącego rozwiązania x^i, wyznacza się jego sąsiedztwo $N(x^i)$ – podzbiór zbioru rozwiązań dopuszczalnych. Sąsiedztwo jest generowane przez ruchy, tj. ustalone przekształcenia rozwiązania x^i. Następnie z sąsiedztwa wyznaczany jest najlepszy element x^{i+1}, który przyjmuje się za bieżące rozwiązanie w następnej iteracji. Liczba iteracji takiego algorytmu nie powinna być większa niż n, a jego złożoność obliczeniowa $O(n^3)$. Dzięki temu będzie

można badać różne minima lokalne. W każdej iteracji algorytmu *ATP*, po wyznaczeniu minimów lokalnych (procedura *LocalOpt* – oparta na metodzie zstępującej), modyfikujemy zbiór elementów i pozycji ustalonych, przyjmując na początek $FS^{i+1} = FS^i$. Następnie, są wykonywane są operacje:

a) zmiany „wieku" każdego ustalonego elementu,
b) usunięcia najstarszych ustalonych elementów (ich liczba jest parametrem),
c) ustalenia nowych elementów.

Niech $P^i = \{\pi_1, \pi_2, \ldots, \pi_\eta\}$ będzie η – elementową populacją w i-tej iteracji algorytmu. Dla każdej permutacji $\pi_j \in P^i$, stosując algorytm poszukiwań lokalnych (procedura $LocalOpt(\pi_j)$) wyznaczamy zbiór minimów lokalnych $LM^i = \{\hat{\pi}_1, \hat{\pi}_2, \ldots, \hat{\pi}_\eta\}$, gdzie permutacja $\hat{\pi}_j = (\hat{\pi}_j(1), \hat{\pi}_j(2), \ldots, \hat{\pi}_j(n))$, $j = 1,2,\ldots\eta$. Wówczas

$$nr(a,l) = \left|\{\hat{\pi}_j \in LM^i : \ \hat{\pi}_j(l) = a\}\right|,$$

jest liczbą permutacji ze zbioru LM^i, w których na pozycji l znajduje się element a. Jeżeli $a \in N$ jest elementem wolnym oraz $\frac{nr(a,l)}{\eta} \geq \Phi(i)$, to element a ustalamy na pozycji l. Bardziej szczegółowy opis omawianej metody jest zamieszczony w [2].

Wyniki obliczeniowe

Algorytm *ATP* został zaimplementowany w języku C++ i uruchomiony na komputerze wyposażonym w procesor Pentium IV 3,0 GHz z 512 MB pamięci RAM. Eksperymenty obliczeniowe przeprowadzono w celu porównania otrzymanych wyników (F_{ATP}) z najlepszymi (F_{bench}) znanymi w literaturze [3], zamieszczonymi także na stronie http://www.ozone.ri.cmu.edu/benchmarks.html.

Tabela 1

Porównanie wyników algorytmu *ATP* z zamieszczonymi w pracy [3]

nr	F_{bench}	F_{ATP}	*czas*	*PRD*	nr	F_{bench}	F_{ATP}	*czas*	*PRD*
1	978	**819**	83,484	-16,26%	61	79884	**78413**	134,5	-1,84%
2	6489	**5989**	98,813	-7,71%	62	47860	**45652**	156,484	-4,61%
3	2348	**2025**	82,469	-13,76%	63	78822	**78241**	128,078	-0,74%
4	8311	**6855**	84,437	-17,52%	64	96378	**94922**	151,297	-1,51%
5	5606	**5233**	94,031	-6,65%	65	134881	**130560**	155,968	-3,20%
6	8244	**8056**	78,657	-2,28%	66	64054	**59832**	101,109	-6,59%
7	4347	**3891**	72,515	-10,49%	67	34899	**30127**	169,078	-13,67%
8	327	**291**	77,516	-11,01%	68	26404	**22998**	146,75	-12,90%
9	7598	**7215**	76,469	-5,04%	69	75414	**71496**	104,235	-5,20%
10	2451	**2369**	79,906	-3,35%	70	81200	**76799**	152,25	-5,42%
11	**5263**	5339	95,407	1,44%	71	161233	**156828**	136,078	-2,73%
12	0	0	82,703	0,00%	72	56934	**53486**	104,516	-6,06%
13	**6147**	7026	90,64	14,30%	73	36465	**32361**	141,734	-11,25%
14	**3941**	4128	71,875	4,74%	74	**38292**	38344	130,703	0,14%
15	2915	**2421**	72,266	-16,95%	75	30980	**26044**	128,203	-15,93%
16	**6711**	6762	68,844	0,76%	76	67553	**61543**	98,156	-8,90%
17	462	**433**	81,75	-6,28%	77	40558	**38255**	124,75	-5,68%
18	2514	**2363**	82,125	-6,01%	78	25105	**24864**	126,312	-0,96%

19	**279**	308	74,375	10,39%	79	**125824**	126248	116,312	0,34%
20	4193	**4065**	74,5	-3,05%	80	31844	**26743**	119,25	-16,02%
21	0	0	86,657	0,00%	81	**387148**	389106	84,75	0,51%
22	0	0	59,093	0,00%	82	413488	**412092**	117,75	-0,34%
23	0	0	39,875	0,00%	83	466070	**463334**	109,719	-0,59%
24	1791	**1119**	85,141	-37,52%	84	**331659**	331828	105,313	0,05%
25	0	0	91,969	0,00%	85	**558556**	559689	92,015	0,20%
26	0	0	76,515	0,00%	86	**365783**	365843	101,89	0,02%
27	229	**78**	110,329	-65,94%	87	403016	**402798**	101,281	-0,05%
28	72	**0**	106,578	-100,00%	88	**436855**	437736	79,297	0,20%
29	0	0	66,063	0,00%	89	416916	**415877**	136,547	-0,25%
30	575	**318**	92,265	-44,70%	90	406939	**404482**	126,953	-0,60%
31	0	0	59,313	0,00%	91	**347175**	355818	86,984	2,49%
32	0	0	60,859	0,00%	92	**365779**	366234	104,875	0,12%
33	0	0	89,625	0,00%	93	**410462**	420098	120,031	2,35%
34	0	0	73,109	0,00%	94	**336299**	340958	94,829	1,39%
35	0	0	68,609	0,00%	95	**527909**	539626	100,922	2,22%
36	0	0	88,375	0,00%	96	**464403**	477198	111,343	2,76%
37	2407	**2014**	95,688	-16,33%	97	**420287**	425184	107,281	1,17%
38	0	0	73,656	0,00%	98	**532519**	542264	70,891	1,83%
39	0	0	103,156	0,00%	99	**374781**	380745	99,406	1,59%
40	0	0	77,094	0,00%	100	**441888**	451029	115,563	2,07%
41	73176	**72244**	103,266	-1,27%	101	**355822**	355868	143,719	0,01%
42	61859	**60397**	84,578	-2,36%	102	**496131**	497805	120,531	0,34%
43	149990	**149655**	107,797	-0,22%	103	**380170**	383464	119,469	0,87%
44	38726	**36081**	129,875	-6,83%	104	362008	**360040**	122,813	-0,54%
45	62760	**60364**	146,422	-3,82%	105	456364	**454426**	112,407	-0,42%
46	37992	**37331**	100,235	-1,74%	106	459925	**457988**	99,578	-0,42%
47	77189	**75528**	76,937	-2,15%	107	356645	**355131**	114,687	-0,42%
48	68920	**67164**	183,797	-2,55%	108	468111	**465526**	111,406	-0,55%
49	84143	**80928**	112,438	-3,82%	109	415817	**414835**	111,109	-0,24%
50	36235	**34021**	107,438	-6,11%	110	**421282**	421305	122,75	0,01%
51	58574	**57127**	129,016	-2,47%	111	**350723**	351653	144,203	0,27%
52	**105367**	109340	140,844	3,77%	112	**377418**	382863	120,969	1,44%
53	**95452**	100496	115,969	5,28%	113	**263200**	263273	104,766	0,03%
54	**123558**	136857	135,922	10,76%	114	**473197**	480110	74,406	1,46%
55	**76368**	77273	113,578	1,19%	115	**460225**	471697	109,547	2,49%
56	88420	**84078**	137,203	-4,91%	116	**540231**	549506	100,421	1,72%
57	**70414**	77981	116,375	10,75%	117	**518579**	518659	113,563	0,02%
58	55522	**52331**	130,203	-5,75%	118	**357575**	366455	145,75	2,48%
59	**59060**	59399	124,234	0,57%	119	**583947**	588287	109,75	0,74%
60	73328	**71701**	109,421	-2,22%	120	**399700**	415623	136,359	3,98%
					Średnio			**105,53**	**-3,88%**

Na podstawie wyników zamieszczonych w tabeli 1 można jednoznacznie stwierdzić, że metoda algorytmu hybrydowego (ewolucja oraz lokalna optymalizacja) okazała się bardzo obiecującym podejściem do rozwiązywania rozpatrywanego problemu. Wyznaczono 62 nowe górne ograniczenia dla optymalnej wartości funkcji celu. Średnie względne odchylenie (*PRD*) od wartości rozwiązań referencyjnych wyniosło - 3,88%, a średni czas obliczeń dla jednego przykładu około 1,5 minuty.

LITERATURA:

1. Armentano V.A., Mazzini R.: A genetic algorithm for scheduling on a single machine set-up times and due dates. Production Planning & Control, 11(7), 2000, p. 713-720.
2. Bożejko W., Wodecki M.: A hybrid evolutionary algorithm for the permutation optimization problem. ISDA 05, IEEE Computer Society, 2005, p. 326-331.
3. Cicirello V.A., Smith S.F.: Enhancing stochastic search performance by value-based randomization of heuristics. Journal of Heuristics, 11, 2005, p. 5-34.
4. Gagné C., Price W.L., Gravel M.: Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times. Journal of the Operational Research Society, 53, 2002, p.895-906.
5. Lee Y.H., Bhaskaran K., Pinedo M.: A heuristic to minimize the total weighted tardiness with sequence-dependent setups. IIE Transactions, 29, 1997, p. 45-52.
6. Panwalkar S.S., Dudek R.A., Smith M.L.: Sequencing research and the industrial scheduling problem. Symposium on the theory of scheduling and its applications (ed. Elmaghraby S.E.), Springer-Verlag, Berlin 1973.
7. Sun X., Noble J.S., Klein C.M.: Single-machine scheduling with sequence dependent setup to minimize total weighted squared tardiness. IIE Transactions, 31, 1999, p. 113-124.
8. Tan K.C., Narasimban R., Rubin P.A., Ragatz G.L.: A comparison on four methods for minimizing total tardiness on a single processor with sequence dependent setup times. Omega, 28, 2000, p. 313-326.

Recenzent: Prof. dr hab. inż. Joanna Józefowska

Abstract

In the survey paper of Panwalker, Dudek i Smith [6] of scheduling in the industry there is a notice, that 75% of encountered problems requires at least one setup time, and 15% of problems should be modeled by setups between all jobs. However majority of papers concern a single machine scheduling problem without setup times. In this paper we propose a population-based algorithm for solving a single machine scheduling problem with total tardiness criterion and sequence-dependent setup times. It is represented by $1|s_{ij}|\Sigma w_i T_i$ in the literature and it belongs to the strongly NP-hard class. Calculations on the representative group of benchmark instances were done and the results were compared with the best known from the literature. Obtained solutions were better than benchmark ones for many instances.