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

Wojciech BOŻEJKO[1], Paweł RAJBA[2], Mieczysław WODECKI[3]
[1]Politechnika Wrocławska, Wyższa Szkoła Zarządzania „Edukacja", Wrocław
[2] Politechnika Wrocławska
[3]Uniwersytet Wrocławski, Wyższa Szkoła Zarządzania „Edukacja", Wrocław

METODA ANALIZY MINIMÓW LOKALNYCH DO ROZWIĄZYWANIA PERMUTACYJNYCH PROBLEMÓW OPTYMALIZACYJNYCH

Streszczenie. W pracy przedstawiono metodę konstrukcji algorytmów rozwiązywania problemów optymalizacyjnych opartą na analizie minimów lokalnych. Najlepsze cechy tych minimów są dziedziczone przez następną populację rozwiązań. Wykonano eksperymenty obliczeniowe, które potwierdziły efektywność proponowanej metody.

THE METHOD OF LOCAL MINIMA ANALYSIS FOR SOLVING PERMUTATIONAL OPTIMIZATION PROBLEMS

Summary. In the paper we present a method of algorithms construction based on analyzing local minima for solving optimization problems. The best properties of these minima are succeeded by a next generation of solutions. Computational experiments, which has been done, affirmed the efficiency of the proposed method.

1. Wstęp

Wiele praktycznych problemów związanych z procesem podejmowania decyzji z dziedziny planowania, zarządzania oraz sterowania sprowadza się do rozwiązania pewnego problemu optymalizacji dyskretnej. Ponieważ zazwyczaj są to problemy NP-trudne, stąd w praktyce do ich rozwiązywania stosuje się niemal wyłącznie algorytmy przybliżone. Obecnie najlepsze znane w literaturze i najczęściej stosowane są metaheurystyki: poszukiwania z zabronieniami (*tabu search*), symulowane wyżarzanie (*simulated anealing*) oraz algorytmu genetycznego (*genetic algorithm*). Dla wielu problemów, przede wszystkim tych z regularnymi funkcjami celu, wyznaczone przez nie rozwiązania różnią się zaledwie o kilka procent od optymalnych. Zdecydowanie gorzej radzą one sobie z problemami, dla których nie są obecnie znane własności umożliwiające przegląd pośredni rozwiązań (takie jak np. własności eliminacyjne bloków [2], [8]). Zazwyczaj, wpadają one bowiem w „pułapkę" lokalnego minimum i jedynym wyjściem jest wówczas restart algorytmu.

W pracy przedstawimy metodę konstrukcji algorytmów przybliżonych dla problemów optymalizacyjnych, których rozwiązaniami dopuszczalnymi są permutacje. Jej idea jest opata 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*).

Skuteczność proponowanej metody sprawdzimy, wykonując obliczenia na pewnej grupie jednomaszynowych problemów szeregowania zadań z sumokosztowymi funkcjami celu. Pomimo prostoty sformułowania należą one do klasy problemów silnie NP-trudnych. Łatwość aplikacji powoduje, że problemy te są jednymi z częściej analizowanych i są powszechnie wykorzystywane do weryfikowania nowych podejść i algorytmów. Przedstawione w ostatnim rozdziale wyniki w pełni potwierdziły praktyczną przydatność prezentowanej metody konstrukcji algorytmów. Zastosowanie lepszej procedury wyznaczania minimów lokalnych (np. algorytmu poszukiwania z zabronieniami) zapewne poprawi zbieżność algorytmu oraz wartości wyznaczanych rozwiązań, niestety, kosztem zwiększenia się czasu obliczeń.

2. Metoda analizy minimów lokalnych

Niech Π będzie zbiorem permutacji elementów ze zbioru $J=\{1,2,...,n\}$. Rozpatrujemy *Problem Optymalizacji Dyskretnej* (*POD*) polegający na wyznaczeniu permutacji $\pi^* \in \Pi$ takiej, że $F(\pi^*) = \min\{\beta \in \Pi : \beta \in \Pi\}$, gdzie $F : \Pi \to R^+ \cup \{0\}$.

Działanie algorytmu rozwiązywania *POD* opartego na metodzie analizy minimów lokalnych rozpoczyna się od wyznaczenia 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 następująco. Dla bieżącej populacji P^i wyznacza się zbiór minimów lokalnych LM^i (dla każdego elementu $\pi \in P^i$ wykonując procedurę $LpcalOpt(\pi)$). Następnie ustala się elementy występujące na tych samych pozycjach w minimach lokalnych, tworząc zbiór elementów i pozycji ustalonych FS^{i+1}. Jednocześnie zwiększa się o jeden „wiek" każdego elementu zbioru FS^{i+1}. Aby zapobiec zbyt szybkiemu ustaleniu wszystkich elementów, po osiągnięciu pewnego wieku (który jest parametrem algorytmu) element jest zwalniany. 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 (kryterium zatrzymania).

Algorytm analizy minimów lokalnych *AAM*

Inicjalizacja: wyznaczyć losową populację początkową: $P^0 = \{\pi_1, \pi_2, \ldots, \pi_\eta\}$;

$\pi^* \leftarrow$ najlepszy element populacji P^0;

i=0; $FS^0 \leftarrow \varnothing$;

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*;

W każdej iteracji algorytmu *AAM*, po wyznaczeniu minimów lokalnych, modyfikujemy zbiór elementów i pozycji ustalonych, wykonując następujące operacje:

a. ustalenia nowych elementów,
b. zmiany „wieku" każdego z ustalonych elementów,
c. usunięcia (zwolnienia) najstarszych ustalonych elementów.

Niech $LM^i = \{\hat{\pi}_1, \hat{\pi}_2, \ldots, \hat{\pi}_\eta\}$ będzie zbiorem minimów lokalnych w i-tej iteracji algorytmu. 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 występuje a. Jeżeli $a \in J$ jest elementem wolnym oraz $\frac{nr(a,l)}{\eta} \geq \Phi(i)$ (gdzie $\Phi(i)$ jest parametrem), to element a ustalamy na pozycji l. Następnie wiek każdego ustalonego elementu zwiększamy o 1. Jeżeli $a \in J$ jest elementem ustalonym oraz jego wiek $\chi(a) > \Gamma(i)$ (gdzie $\Gamma(i)$ jest parametrem), to element a zwalniamy (usuwamy ze zbioru FS^i). Szczegółowy opis omawianej metody jest zamieszczony w pracy [1].

Do wyznaczania minimów lokalnych będzie stosowany algorytm zstępujący oparty na metodzie lokalnych popraw. Algorytm rozpoczyna obliczenia od pewnego rozwiązania startowego β. W każdej iteracji, dla bieżącego rozwiązania π, wyznacza się jego otoczenie $N(\pi)$ - podzbiór zbioru rozwiązań dopuszczalnych. Otoczenie jest generowane przez ruchy, tj. ustalone przekształcenia rozwiązania π. Następnie, z otoczenia jest wyznaczany najlepszy element β, który przyjmuje się za bieżące rozwiązanie w następnej iteracji.

Algorytm zstępujący

Niech $\beta \in \Pi$ będzie rozwiązaniem startowym;
repeat
$\pi \leftarrow \beta$;
Wyznaczyć permutację $\beta \in N(\pi)$ taką, że $F(\beta) = \min\{F(\delta) : \delta \in \Pi\}$;
until $F(\pi) \geq F(\beta)$.

Wyznaczona przez algorytm zstępujący permutacja π jest pewnym minimum lokalnym. Poniżej zamieszczamy opis najważniejszych elementów tego algorytmu.

Ruch i otoczenie

W każdej iteracji algorytmu zstępującego jest wyznaczane otoczenie – podzbiór zbioru rozwiązań dopuszczalnych. Jest ono generowane przez ruchy – przekształcenia

określone na zbiorze wszystkich permutacji. Niech k i l $(k<l)$ będzie parą pozycji w permutacji:

$$\pi=(\pi(1),\pi(2),\ldots,\pi(k-1),\pi(k),\pi(k+1),\ldots,\pi(l-1),\pi(l),\pi(l+1),\ldots,\pi(n)).$$

Ruch typu *wstaw* (*insert*), oznaczany przez i_l^k, polega na przestawieniu zadania $\pi(k)$ z pozycji k na pozycję l. Generuje on permutację $i_l^k(\pi)=\pi_l^k\in\Pi$, gdzie:

$$\pi_l^k(i)=\begin{cases}\pi(k), & i=l,\\ \pi(i), & l<i<k,\\ \pi(i+1), & i=k,k+1,\ldots,l-1.\end{cases}$$

Ruch i_l^k, w przypadku gdy $k>l$, definiujemy podobnie. Wszystkich takich ruchów (moc otoczenia) jest $n(n-1)$.

3. Szeregowanie zadań z terminami zakończenia

Efektywność metody analizy minimów lokalnych zweryfikujemy, wykonując obliczenia dla pewnych, posiadających dane referencyjne, NP-trudnych problemów szeregowania zadań na jednej maszynie z sumokosztowymi funkcjami celu.

Niech $J=\{1,2,\ldots,n\}$ będzie zbiorem zadań. Z i-tym zadaniem jest związany *czas wykonywania* p_i, *najwcześniejszy* e_i oraz *najpóźniejszy* d_i *termin* zakończenia oraz *współczynniki funkcji kary* za zbyt wczesne u_i lub zbyt późne w_i zakończenie. Jeżeli C_i jest momentem zakończenia wykonywania zadania, wówczas $E_j=\max\{0,e_j-C_j\}$ jest *przyśpieszeniem*, $T_j=\max\{0,C_j-d_j\}$ *opóźnieniem*, a $u_jE_j+w_jT_j$ *kosztem* (karą za spóźnienie) wykonania zadania.

Problem minimalizacji sumy kosztów wykonania zadań z najwcześniejszymi i najpóźniejszymi terminami zakończenia (*Total Weighted Earliness/Tardiness* problem - w skrócie oznaczanym przez *TWET*) polega na zminimalizowaniu funkcji celu $\sum_{j=1}^{n}(u_jE_j+w_jT_j)$. W literaturze jest on oznaczany przez $1\|\sum(u_jE_j+w_jT_j)$ i należy do klasy problemów silnie NP-trudnych. Opublikowano wiele prac poświęconych temu problemowi. Obecny stan badań jest przedstawiony w pracach Lauffa i Wernera [6] oraz Hoogeveena [5]. Łatwo zauważyć, że w optymalnym rozwiązaniu mogą występować przestoje maszyny (zadania nie muszą być wykonywane bezpośrednio po sobie). Rozwiązanie problemu *TWET* sprowadza się więc do wyznaczenia takich momentów rozpoczęcia zadań, aby spełnione były wszystkie ograniczenia.

Problem, w którym dla każdego zadania $i\in J$, $e_i=d_i$, tj. problem z jednakowymi najwcześniejszymi i najpóźniejszymi terminami zakończenia zadań (*Common Due Date*, w skrócie *TWcdd*), jest oznaczany przez $1\,|\,e_i=d_i=d\,|\sum_i(u_iE_i+w_iT_i)$. Bogaty przegląd wyników badań nad problemami szeregowania z jednakowymi terminami zakończenia zadań jest przedstawiony w pracy Gordona i in.[3], a najnowsze wyniki w pracach Hendela i Soursa [4].

W ostatnim, jednym z najczęściej badanych problemów szeregowania (*Total Weighted Tardiness* problem, w skrócie *TWT*) przyjmujemy, że $u_i=0$. Polega więc on na wyznaczeniu takiej kolejności wykonywania zadań, która minimalizuje *sumę*

kosztów opóźnień, tj. $\sum w_i T_i$. W literaturze jest on oznaczany przez $n|1|\sum w_i T_i$. Algorytmy optymalne pozwalają rozwiązywać (w rozsądnym czasie) przykłady, w których liczba zadań jest nie większa niż 80 (Wodecki [9]). Najlepszy obecnie algorytm przybliżony, oparty na metodzie przeszukiwania z tabu, przedstawiono w pracy Bożejko i in. [2].

4. Eksperymenty obliczeniowe

Dla problemów *TWcdd* oraz *TWT* przykładowe dane oraz wartości rozwiązań są zamieszczone na stronie internetowej [7]. Ponieważ dla problemu *TWET* brak jest ogólnie dostępnych przykładów, więc zostały one wygenerowane losowo, tak jak to opisano w pracy [8]. Ich rozwiązania wyznaczone przez algorytm oparty na metodzie przeszukiwania z tabu (zamieszczony w [8]) zostały porównane z wynikami algorytmu *AAM*. Dla każdego przykładu rozwiązywanego przez algorytm *AAM* wyznaczono procentowy błąd względny (względne odchylenie) $\delta = \frac{F^{AAM} - F^*}{F^*} \cdot 100\%$, gdzie F^{MMA} jest wartością rozwiązania wyznaczonego przez algorytm *AAM*, a F^* wartością optymalną lub najlepszą obecnie znaną. Średnie wartości tych błędów zamieszczono w tabeli 1. Na podstawie otrzymanych wyników można stwierdzić, że algorytm analizy minimów lokalnych daje wyniki zdecydowanie lepsze dla problemu ***TWcdd*** oraz niewiele gorsze dla ***TWET*** i ***TWT*** (chociaż błędy względne są mniejsze od 1%). Najgorsze wyniki otrzymano dla problemu z oknami czasowymi *TWcdd*. Głównym tego powodem jest słabo działający algorytm wyznaczania minimów lokalnych. Zastosowanie „prostej" wersji algorytmu przeszukiwania z tabu powoduje, że średni błąd względny jest także mniejszy od 1% (jednak przy znacznym wzroście czasu obliczeń). Maksymalny czas obliczeń największych przykładów nie przekraczał kilku sekund.

Tabela 1

Średni błąd względny δ_{aprd} w stosunku do rozwiązań referencyjnych

Problem	*Liczba zadań n*	*Liczba przykładów*	*Algorytm AAM* (δ_{aprd})	*Liczba uzyskanych najlepszych rozwiązań*
TWET (dane [8])	40	125	0.24	104
	50	125	0.37	85
	80	125	1.83	59
	średnio	-	**0.80**	-
	najlepszy znany algorytm [8]		**0.53**	
TWcdd (dane: OR-Library [7])	20	40	1.57	24
	50	40	3.45	1
	100	40	4.07	0
	średnio	-	**2.64**	-
	najlepszy znany algorytm [4]		**4.36**	
TWT (dane: OR-Library [7])	40	125	0.00	119
	50	125	0.19	99
	100	125	0.00	59
	średnio	-	**0.06**	-
	najlepszy znany algorytm [2]		**0.00**	

BIBLIOGRAFIA

1. Bożejko W., Wodecki M.: A hybrid evolutionary algorithm for the permutation optimization problem. ISDA 05, IEEE Computer Society, 2005, p.326-331.
2. Bożejko W., Grabowski J., Wodecki M.: Block approach tabu search algorithm for single machine total weighted tardiness problem, Computers & Industrial Engineering, 50,1/2, 2006, p.1-14.
3. Gordon V., Proth J.P., Chu C.: A survey of the state-of-art of common due date assignment and scheduling research, European Journal of Operational Research, 139, 2002, p.1-25.
4. Hendel Y., Sourd F.: Efficient neighborhood search for the one-machine earliness-tardiness scheduling problem, European Journal of Operational Research, 173, 1, 2006, p.108-119.
5. Hoogeveen J.A.: Multicriteria scheduling, European Journal of Operational Research, 167, 2005, p.592-623.
6. Lauff V., Werner F.: Scheduling with common due date, earliness and tardiness penalties for multimachine problems: A survey, Mathematical and Computer Modelling, 40, 2004, p.637-655.
7. OR Library http://people.brunel.ac.uk/~mastjjb/jeb/orlib/schinfo/html
8. Wodecki M.: A block approach to earliness-tardiness scheduling problems *International Journal on Advanced Manufacturing Technology*, (DOI: 10.1007/s00170-008-1395-7, 2008).
9. Wodecki M.: A Branch-and-Bound Parallel Algorithm for Single-Machine Total Weighted Tardiness Problem, Advanced Manufacturing Technology, (DOI: 10.1007/s00170-008-1023-y), 2008.

Recenzent: Prof. dr hab. inż. Eugeniusz Toczyłowski

Abstract

In the paper we present a method of algorithms construction based on analyzing local minima for solving optimalization problems in which feasible solutions are permutations. Its idea is based on the following state. If there are the same elements in the same positions in different local minima, therefore they can be on these positions in the optimal solution. The best properties of these minima are succeeded by a next generation of solutions. This method was used to solve some NP-hard single machine scheduling problems. Computational experiments, which has been done, affirmed the efficiency of the proposed method on the representative group of benchmark instances taken from the literature.