ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1988
Seria: AUTOMATYKA z.95 Nr kol.971

Konrad Wala
AGH Kraków

METODY LOKALNEJ OPTYMALIZACJI W ZAGADNIENIU HARMONOGRAMOWANIA(*)

Streszczenie. W pracy zaprezentowano algorytm optymalizacji lokalnej dla zagadnień harmonogramowania, w którym zastosowano ideę poszukiwania o zmiennej głębokości B.W.Kernighana i S.Lina. Porównano efektywność tego algorytmu ze standardowymi metodami optymalizacji lokalnej.

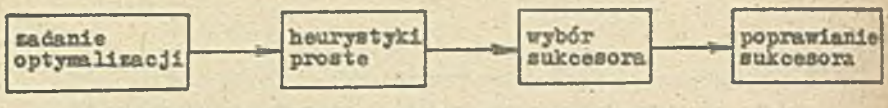
1. Wprowadzenie

W Instytucie Automatyki AGH jest realizowany komputerowy system wspomagania syntezy przybliżonych algorytmów dla potrzeb operatywnego sterowania dyskretnych procesów przemysłowych nazwany systemem KWSP. System ten umożliwia użytkownikowi pracującemu nad algorytmem przybliżonym obliczania decyzji sterujących dla konkretnego zagadnienia na wygenerowanie potrzebnego modelu komputerowego procesu produkcji za pomocą bloku zarządzającego systemu. Wystarczy, że w tym celu poda on nazwy i parametry charakteryzujące proces i tym samym ustali funkcje symulatora procesu [4]. Ponadto użytkownik ma do dyspozycji dwa pakiety programów optymalizacji, które pozwalają na syntezę przybliżonego algorytmu sterowania procesem w trybie pracy interakcyjnej użytkownika z komputerem, przy czym synteza polega na wyborze i składaniu algorytmów z modułów programów optymalizacji oraz wyborze parametrów algorytmów. Oczywiście użytkownik może także uzupełnić pakiety programów własnymi rozwiązaniami.

Przyjęto, że podstawowy schemat optymalizacji stosowany w systemie KWSP polega na realizacji dwuetapowego procesu obliczeń (rys.1). W pierwszym etapie obliczeń wyznaczane są rozwiązania za pomocą pewnej liczby "prostych" heurystyk wybranych przez użytkownika z pierwszego pakietu programów optymalizacji. Pod nazwą heurystyka prosta rozumie się tu procedurę, która umożliwia wyznaczenie jednego rozwiązania, przy czym każda iteracja procedury wyznacza jedną składową tego rozwiązania. Spośród

(*) Praca została wykonana w ramach programu resortowego RP.I.02 "Teoria sterowania i optymalizacji ciągłych układów dynamicznych i procesów dyskretnych".

rozwiązań etapu pierwszego wybierane jest rozwiązanie najlepsze (jedno lub kilka), zwane dalej sukcesorem (sukcesorami).



Rys.1. Schemat dwuetapowego procesu optymalizacji
Fig.1. Diagram of two-stage optimization process

Etap drugi optymalizacji polega na zastosowaniu wieloiteracyjnego procesu poprawiania sukcesora za pomocą procedur drugiego pakietu optymalizacji, przy czym iteracja polega na wygenerowaniu jednego "nowego rozwiązania" i porównaniu go z sukcesorem.

Zakłada się, że użytkownik systemu KWSP na podstawie eksperymentu komputerowego w trakcie pracy interakcyjnej wybierze jedną lub kilka procedur z pakietu pierwszego oraz jedną z drugiego i w ten sposób dokona syntezy algorytmu optymalizacji dla rozwiązywanego zagadnienia.

Podstawowymi metodami optymalizacji drugiego pakietu są różne warianty metod optymalizacji lokalnej. Najczęściej stosowanym, ze względu na efektywność procesu optymalizacji, jest podejście polegające na szukaniu rozwiązania lepszego najpierw w mniejszym otoczeniu sukcesora, a następnie - w przypadku braku sukcesu w poprawie rozwiązania - zwiększa się otoczenie sukcesora i kontynuuje szukanie. Zauważmy, że zagadnienia rozwiązywane z zakresu operatywnego sterowania dyskretnych procesów przemysłowych są zwykle zagadnieniami złożonymi i czas wygenerowania jednego rozwiązania dopuszczalnego jest stosunkowo duży. Tak więc poprawianie sukcesora należy wykonywać w małym otoczeniu i jego powiększanie jest zwykle niedopuszczalne ze względu na czas obliczeń.

Powstała stąd potrzeba opracowania nowej - czasowo oszczędnej - idei lokalnej optymalizacji. Sprawdzono w tym celu możliwość przystosowania do zagadnień harmonogramowania interesującego podejścia, które B.W.Kernighan i S.Lin zaproponowali dla zagadnienia podziału grafu.

2. Opis algorytmów lokalnej optymalizacji

Istotą metody lokalnej optymalizacji jest próba poprawienia sukcesora x (najlepszego znanego rozwiązania zagadnienia) na drodze przeszukiwania rozwiązań tylko z jego otoczenia $N_r(x)$, gdzie $N_r(x) \subset X$, X - zbiór dyskretny, x - sukcesor, r - parametr. O rodzinie zbiorów $N_r(x)$ zakłada się, że:

1. dla każdego $x \in X$ istnieje dokładnie jeden zbiór $N_r(x)$,
2. $x \in N_r(x)$ dla każdego $x \in X$,

oraz zwykle zachodzi relacja $N_{r1}(x) \subset N_{r2}(x)$ dla $r1 < r2$.

W pierwszej kolejności przypomnijmy podstawowe - standardowe - warianty algorytmów lokalnej optymalizacji. Niech $S \subset X$ jest zbiorem dopuszczalnych rozwiązań zagadnienia minimalizacji funkcji celu $f(x)$.

Algorytm A1(r)

Podstawiamy $x_i = x$, $i = 1$ oraz określamy r . Rozwiązujemy zagadnienie $f(x_{i+1}) = \{\min f(y) : y \in N_r(x_i) \cap S\}$

Jeżeli $x_i = x_{i+1}$, to kończymy obliczenia i za rozwiązanie lokalnie optymalne zagadnienia przyjmujemy x_i . W przeciwnym przypadku zamieniamy i na $i+1$ i powtarzamy obliczenia.

W algorytmie A1(r) realizowany jest pełny przegląd otoczenia sukcesora $N_r(x)$ i najlepsze z rozwiązań jest porównywane z sukcesorem.

Algorytm A2(r)

Ustalamy r i kolejność przeglądania rozwiązań zbioru $N_r(x)$ oraz podstawiamy $x_i = x$, $i = 1$. Dla kolejnych rozwiązań y zbioru $N_r(x_i) \cap S$ obliczamy $f(y)$. Jeżeli $f(y) < f(x_i)$, to podstawiamy $x_{i+1} = y$ oraz zamieniamy i na $i+1$ i powtarzamy obliczenia. Jeżeli natomiast w zbiorze $N_r(x_i)$ nie znajdziemy rozwiązania lepszego, to za rozwiązanie lokalnie optymalne zagadnienia przyjmujemy x_i .

W algorytmie A2(r) jest realizowany przegląd otoczenia sukcesora do chwili uzyskania pierwszego lepszego rozwiązania.

Algorytm A3(r1,r2,...,rK)

1. Ustalamy liczbę parametrów K , parametry $r1,\dots,rK$ i kolejność przeglądania rozwiązań zbiorów $N_{rk}(x)$, $k = 1,\dots,K$ oraz podstawiamy $x_i = x$, $i = 1$.
2. Dla kolejnych rozwiązań $y \in N_{r1}(x_i) \cap S$ obliczamy $f(y)$. Jeżeli $f(y) < f(x_i)$, to podstawiamy $x_{i+1} = y$, $i = i+1$ oraz powtarzamy krok 2. Jeżeli sprawdziliśmy z negatywnym skutkiem ostatni element zbioru $N_{r1}(x_i) \cap S$, to przechodzimy do kroku 3.
3. Dla każdego otoczenia $N_{rk}(x_i)$, $k = 2,3,\dots,K$ próbujemy ponownie znaleźć rozwiązanie $y \in N_{rk}(x_i) \cap S$ spełniające nierówność:

$$f(y) < f(x_i) \qquad (*)$$

Jeżeli y^* jest rozwiązaniem nierówności $(*)$ otrzymanym dla pewnego k , to podstawiamy $x_{i+1} = y^*$, $i = i+1$ i przechodzimy do kroku 2 . Jeżeli nie znaleziono żadnego rozwiązania dla $(*)$, to za rozwiązanie lokalnie optymalne zagadnienia przyjmujemy x_i .

W algorytmie A3(r1,...,rK) poszukiwanie rozwiązania lepszego przeprowadza się najpierw w najmniejszym otoczeniu i w przypadku niepowodzenia przechodzi się do poszukiwania w coraz większych otoczeniach sukcesora.

Ideą lokalnej optymalizacji dotychczas nie wypróbowaną w rozwiązywa-

niu zagadnień harmonogramowania jest interesująca metoda,którą B.W.Kernighan i S.Lin zaproponowali dla zagadnienia podziału grafu [1] .Nazwali oni tę metodę poszukiwaniem o zmiennej głębokości.W zagadnieniu równomiernego podziału niezorientowanego grafu (V, E) z $V = 2n$ węzłami poszukuje się rozwiązania $A \cup B = V$, $|A| = |B|$,którego koszt $c(A,B) = \sum_{i,j} d_{ij}$, $i \in A$, $j \in B$, jest najmniejszy,gdzie $[d_{ij}]$ jest tablicą kosztów określonych na zbiorze E krawędzi grafu.

Niech dany jest równomierny podział A, B i elementy $a \in A$ i $b \in B$ .Wtedy nowy równomierny podział A', B' otrzymamy realizując operację wymiany:

$$A' = (A - \{a\}) \cup \{b\} \quad , \quad B' = (B - \{b\}) \cup \{a\} \quad .$$

Można teraz wprowadzić definicję otoczenia dla zagadnienia podziału grafu:

$N(A,B) = \{$wszystkie równomierne podziały A', B',które można otrzymać z równomiernego podziału A, B za pomocą jednej wymiany$\}$.

Idea poszukiwania o zmiennej głębokości polega na zamianie poszukiwania jednej korzystnej wymiany (tj.jednego korzystnego wyboru) w otoczeniu $N(A,B)$ na poszukiwanie <u>ciągu</u> wymian z wykorzystaniem wartości funkcji celu w charakterze przewodnika.Ciąg L wymian znajduje się tu w następujący sposób:

1. Wybieramy parę $a_1 \in A$, $b_1 \in B$ tak,ażeby funkcja celu była możliwie najmniejsza (niekoniecznie mniejsza niż dla sukcesora!) .
2. Wymieniamy a_1 i b_1 ,obliczamy wartość funkcji celu
$$f_1 = c\left[(A - \{a_1\}) \cup \{b_1\}, (B - \{b_1\}) \cup \{a_1\}\right]$$
i określamy zbiory A i B :
$$A := A - \{a_1\} \quad , \quad B := B - \{b_1\}$$
3. Powtarzamy kroki 1 i 2 ustaloną liczbę L razy,otrzymując ciąg wymienianych par $(a_1,b_1),(a_2,b_2),\ldots,(a_L,b_L)$ .Po tym,jak pewna para jest wymieniona,<u>dalej nie jest brana pod uwagę przy wymianie w kroku 1</u> (jej miejsce w zbiorach jest już ustalone).W rezultacie otrzymujemy ciąg wartości funkcji celu $f_1,f_2,\ldots,f_L$.Wybieramy teraz takie l ,ażeby wartość funkcji celu była minimalna: $f_l = \min\{f_i : i \in \{1,2,\ldots,L\}\}$ oraz wymieniamy zbiory $X = \{a_1,\ldots,a_l\}$ i $Y = \{b_1,\ldots,b_l\}$ .W przypadku uzyskania poprawy rozwiązania procedurę można znów rozpocząć od kroku 1 .

Zauważmy,że wymianę X, Y trudno byłoby znaleźć rozpatrując tylko po jednej wymianie w każdym kroku (w każdym otoczeniu). W ten sposób wykonuje się "głębokie przerwanie" od punktu,w którym się znajdujemy,bez konieczności rozpatrzenia wszystkich takich ciągów.Kernighan i Lin stwierdzili wysoką efektywność tego podejścia dla zagadnienia podziału grafu.

Podjęto próbę przeniesienia tej idei do zagadnień kolejnościowych.

Niech sukcesorem zagadnienia kolejnościowego jest ciąg $\pi = (j_1, j_2, \ldots, j_n)$, (na przykład permutacja n liczb całkowitych). Rozwiązania z jego otoczenia otrzymuje się zmieniając położenie poszczególnych elementów ciągu. Przyjęto, że jeżeli zmiana położenia elementu j_i w ciągu daje najlepsze rozwiązanie w otoczeniu sukcesora, to przy dalszym poszukiwaniu "w głąb" za lepszymi rozwiązaniami ustala się położenie tego elementu wśród jego sąsiadów w ciągu (i to zarówno z lewej, jak i prawej strony), przy czym można ustalić położenie wśród dwóch sąsiadów:

$$\ldots, j_{r-1}, \underline{j_r, j_i, j_{r+1}}, j_{r+2}, \ldots$$

czterech sąsiadów:

$$\ldots, j_{r-2}, \underline{j_{r-1}, j_r, j_i, j_{r+1}, j_{r+2}}, j_{r+3}, \ldots \qquad \text{itd.}$$

W czasie generowania nowych rozwiązań w otoczeniu sukcesora, element j_i i jego sąsiedzi (tj. trójki (j_r, j_i, j_{r+1}), piątki $(j_{r-1}, j_r, j_i, j_{r+1}, j_{r+2})$, itd.) traktowani są dalej jako jeden element, tj. przy kolejnych zmianach położenia elementów sukcesora ulega przesunięciu cała trójka czy piątka. Zauważmy, że ustalenie s sąsiadów wokół elementu j_i powoduje zmniejszenie liczby rozwiązań w otoczeniu sukcesora dokładnie o s rozwiązań. Możemy teraz podać algorytm lokalnej optymalizacji z szukaniem w głąb.

Algorytm A4(r,s,L)

1. Ustalamy parametr r, liczbę sąsiadów s i głębokość szukania L ($r = 1,2,\ldots$; $s = 2,4,6,\ldots$; $L = 2,3,\ldots$) oraz podstawiamy $\pi_{l-1} = \pi$, $l = 1$.
2. Rozwiązujemy zagadnienie
$$f(\pi_l) = \min\{f(\pi) : \pi \in (N_r(\pi_{l-1}) - \{\pi_{l-1}\}) \cap s\}$$
3. Niech j_i jest elementem (jednym z elementów), który zmienił położenie w czasie generowania rozwiązania π_l z otoczenia $N_r(\pi_{l-1})$. W rozwiązaniu π_l element j_i razem z jego s sąsiadami jest dalej uważany za jeden element. Określamy otoczenie wokół rozwiązania π_l: $N_r(\pi_l)$.
4. Powtarzamy kroki 2 i 3 dla $l = 2,3,\ldots,L$. Wybieramy takie $k \in \{1,2,\ldots,L\}$, dla którego funkcja celu $f(\pi_k)$ jest minimalna.
5. Jeżeli $f(\pi_k) < f(\pi_0)$, to podstawiamy $\pi_0 = \pi_k$ (rozwiązanie π_k jest nowym sukcesorem) oraz powtarzamy obliczenia od kroku 1, w przeciwnym przypadku algorytm kończy działanie.

Uwagi:

- Znalezione w kroku 2 rozwiązanie π_l nie musi być zawsze lepsze od sukcesora.
- W miarę jak rośnie głębokość szukania, zmniejsza się liczba elementów otoczenia. W otoczeniu sukcesora poziomu l mamy o $s(l-1)$ rozwiązań mniej w stosunku do wyjściowej liczby rozwiązań w otoczeniu.

Opracowano także algorytm łączący czynności algorytmu A2 i A4:

Algorytm A5(r,s,L)

1. Ustalamy r,s,L .
2. Poprawiamy sukcesor π za pomocą algorytmu A2(r) .
3. Rozwiązanie lokalnie optymalne (nowy sukcesor) poprawiamy za pomocą czterech początkowych kroków algorytmu A4(r,s,L).Jeżeli uzyskano rozwiązanie lepsze,podstawiamy je za sukcesor i przechodzimy do kroku 2, w przeciwnym przypadku algorytm kończy działanie.

3. Wyniki testowania algorytmów

Efektywność algorytmów lokalnej optymalizacji testowano dla dwóch podstawowych zagadnień harmonogramowania:

1. Minimalizacja czasu wykonania n zadań na m identycznych maszynach równoległych,tj.zagadnienia $n|m|I|c_{max}$ .Do wyznaczenia pierwszego sukcesora zastosowano heurystykę LPT ,która szereguje zadania w ciąg π zgodnie z niemalejącymi wartościami czasu wykonania zadań.
2. Minimalizacja czasu wykonania n zadań na m maszynach ustawionych w szereg - permutacyjne zagadnienie taśmowe $n|m|P|c_{max}$ .Do wyznaczenia pierwszego sukcesora zastosowano heurystykę zaproponowaną przez D.S.Palmera [2].Heurystyka ta szereguje zadania w ciąg π zgodnie z niemalejącym współczynnikiem

$$\lambda_i = \sum_{k=1}^{m} \left| k - (m + 1/2) \right| p_{ik}$$

gdzie: i (i=1,2,...,n) - numer zadania, m - liczba maszyn, p_{ik} -czas operacji zadania i na maszynie k .

Obliczano dwa wskaźniki oceniające efektywność algorytmów lokalnej optymalizacji.Pierwszy ocenia poprawę wartości funkcji celu:

$$q1 = \frac{f(\text{początkowy sukcesor}) - f(\text{rozwiązanie lokalnie optymalne})}{f(\text{początkowy sukcesor})}$$

Natomiast drugi wskaźnik - ważniejszy ze względu na ocenę przydatności algorytmu dla systemu KWSP (patrz dyskusja w punkcie 1) - ma postać:

q2 = q1 / liczba rozw. wygenerowanych przez alg.lokalnej optymalizacji.

Optymalizacja lokalna była realizowana w następujących otoczeniach sukcesora $\pi = (j_1, j_2, \ldots, j_n)$:

$N1_r(\pi) = \{$ciągi π',w których co najwyżej r sąsiednich elementów ciągu π zamieniło się pozycją $\}$,

$N2_r(\pi) = \{(j_1,\ldots,j_r,j_{r+1},\ldots,j_n),(j_{r+1},j_1,\ldots,j_r,j_{r+2},\ldots,j_n)$,
$(j_{r+1},j_{r+2},j_1,\ldots,j_r,j_{r+3},\ldots,j_n),\ldots,(j_{r+1},\ldots,j_n,j_1,\ldots,j_r)\}$.

W tablicach 1 i 2 przedstawiono średnie wartości wskaźników q1 i q2 ,przy czym dla każdego zagadnienia określonego parą liczb (n,m) rozwiązano 30 przykładów.Czasy operacji każdego przykładu otrzymano na

drodze losowania liczb z przedziału (1,100) przy równomiernym rozkładzie. W algorytmie A4 były realizowane tylko cztery początkowe kroki, tj. realizowany był tylko jeden proces poszukiwania w głąb - osiągano wtedy znacznie wyższą wartość wskaźnika q2.

Tablica 1

Wyniki testowania zagadnienia $n|m|I|c_{max}$

n	algorytm otoczenie	m=2		m=4		m=6		m=8	
		q1	q2= 100	q1	q2= 100	q1	q2= 100	q1	q2= 100
20	A1(2) N1	1.05	1.52	2.27	2.26	2.18	2.86	1.77	2.68
	A2(2)	1.05	2.51	2.37	3.43	2.22	3.99	1.73	3.79
	A4(2,2,5)	1.08	1.44	2.70	3.59	2.76	3.68	1.95	2.61
	A4(2,4,5)	1.06	1.92	1.83	3.33	1.89	3.44	1.59	2.89
	A5(2,2,4)	1.41	1.10	3.46	1.94	3.14	2.27	2.38	2.18
	A5(2,4,4)	1.40	1.28	2.57	2.17	3.27	2.51	2.19	2.27
20	A1(1) N2	1.18	3.04	2.83	6.78	3.49	8.71	3.77	9.92
	A2(1)	1.17	5.47	2.78	10.8	3.50	13.2	3.77	13.9
	A4(1,2,5)	1.20	1.60	3.27	4.36	4.32	5.76	4.51	6.02
	A4(1,4,5)	1.18	2.14	3.20	5.81	3.92	7.13	4.88	8.87
	A5(1,2,4)	1.22	1.51	3.27	3.48	4.43	4.89	4.74	5.12
	A5(1,4,4)	1.17	2.01	3.21	4.43	4.12	5.88	4.87	7.17
30	A1(2) N1	0.41	0.28	1.02	0.45	1.67	0.58	1.96	0.82
	A2(2)	0.40	0.46	1.00	0.62	1.55	0.86	2.10	1.26
	A4(2,2,5)	0.41	0.33	1.23	0.98	1.69	1.35	1.60	1.28
	A4(2,4,5)	0.39	0.38	0.93	0.89	1.19	1.14	1.26	1.20
	A5(2,2,4)	0.58	0.25	1.51	0.39	1.99	0.56	2.22	0.74
	A5(2,4,4)	0.55	0.25	1.08	0.39	1.93	0.59	2.37	0.85
30	A1(1) N2	0.46	0.75	1.20	1.81	1.72	2.42	1.87	2.89
	A2(1)	0.44	1.38	1.14	2.97	1.63	3.81	1.88	4.45
	A4(1,2,5)	0.46	3.70	1.31	1.05	2.28	1.82	2.93	2.34
	A4(1,4,5)	0.46	4.40	1.37	1.31	2.24	2.13	2.94	2.80
	A5(1,2,4)	0.45	0.34	1.35	0.89	2.33	1.44	3.15	1.76
	A5(1,4,4)	0.45	0.41	1.42	1.05	2.42	1.51	3.12	2.10

Z eksperymentów obliczeniowych wynika generalny i ważny wniosek dla zastosowań algorytmów lokalnej optymalizacji w systemie KWSP: największy wskaźnik q2 otrzymano dla algorytmu A4 (bez kroku 5). Algorytm A2 zachowywał się pod tym względem nieznacznie gorzej.

Biorąc pod uwagę wskaźnik q2, algorytm A4 (bez kroku 5) jest znacznie efektywniejszy od algorytmu A5, tj. maksymalną efektywność poprawy wartości funkcji celu otrzymano stosując jeden proces poszukiwania w głąb. W czasie testowania algorytmów stwierdzono także przewagę pełnego algory-

tmu A4 (tj. algorytmu z krokiem 5) nad pozostałymi pod względem wartości wskaźnika q1.

Tablica 2

Wyniki testowania zagadnienia $n|m|P|c_{max}$

n	algorytm otoczenie	m=2		m=4		m=6		m=8	
		q1	q2=100	q1	q2= 100	q1	q2= 100	q1	q2= 100
20	A1(2) N1	1.07	2.12	4.76	3.89	5.92	4.14	6.45	4.41
	A2(2)	1.07	2.46	4.92	5.21	5.50	5.34	5.98	6.54
	A4(2,2,5)	1.93	2.57	4.44	5.92	4.94	6.58	5.55	7.40
	A4(2,4,5)	0.99	1.80	3.70	6.73	4.39	7.99	4.77	8.67
	A5(2,2,4)	1.87	1.56	5.13	2.98	6.35	3.31	6.75	3.71
	A5(2,4,4)	1.07	1.27	5.40	3.53	6.03	3.90	7.19	4.35
20	A1(1) N2	0.17	0.39	1.77	2.81	1.79	2.52	2.48	3.53
	A2(1)	0.16	0.60	1.20	3.05	0.96	2.58	1.69	4.45
	A4(1,2,5)	0.27	0.36	2.43	3.24	2.76	3.69	4.60	6.14
	A4(1,4,5)	0.50	0.92	1.95	3.56	2.16	3.39	3.88	7.06
	A5(1,2,4)	0.28	0.30	3.04	2.33	3.45	2.47	4.87	3.95
	A5(1,4,4)	1.57	1.26	3.18	2.69	2.80	2.55	4.77	4.08
30	A1(2) N1	0.51	0.61	3.61	1.41	4.68	2.15	5.69	2.01
	A2(2)	0.51	0.78	3.59	2.04	4.66	3.12	5.16	2.62
	A4(2,2,5)	1.23	0.98	3.18	2.54	4.11	3.29	4.49	3.59
	A4(2,4,5)	0.48	0.46	2.68	2.55	3.47	3.30	4.07	3.87
	A5(2,2,4)	1.14	0.57	4.10	1.24	5.45	1.76	5.94	1.56
	A5(2,4,4)	0.53	0.35	3.99	1.30	5.15	1.81	5.94	1.73
30	A1(1) N2	0.15	0.20	1.12	1.09	1.65	1.64	1.61	1.48
	A2(1)	0.14	0.30	0.79	1.46	0.78	1.85	0.94	1.71
	A4(1,2,5)	0.21	0.16	1.68	1.34	2.35	1.88	2.68	2.14
	A4(1,4,5)	0.21	0.20	1.71	1.63	2.37	2.26	2.74	2.61
	A5(1,2,4)	0.26	0.17	1.90	0.96	2.65	1.30	3.95	1.62
	A5(1,4,4)	0.44	0.26	2.15	1.27	3.06	1.78	4.12	2.05

LITERATURA

[1] Kernighan B.W., Lin S.: An Efficient Heuristic Procedure for Partitioning Graphs, BSTJ, No.2 /February 1970/,291-307.

[2] Palmer D.S.: Sequencing Jobs Through a Multi-Storage Process in the Minimum Total Time - a Quick Method of Obtaining a Near Optimum, Opnal Res.Quart.,Vol.16,p.101-197 /1965/.

[3] Savage S.L., Weiner P., Krone M.J.: Convergent Local Search. Research Report No.14, Yale University, Computer Science Departament, 1973.

[4] Wala K.,Gądek-Wadeja H.:Komputerowy model symulacyjny dla potrzeb

optymalizacji dyskretnych procesów przemysłowych.Prace VI KK Automatyzacji Dyskretnych Procesów Przemysłowych,1988 (praca zgłoszona).

Recenzent:Doc.dr hab.inż.M.Zaborowski

Wpłynęło do Redakcji do 1988-04-30.

МЕТОДЫ ЛОКАЛЬНОЙ ОПТИМИЗАЦИИ ДЛЯ ПРОБЛЕМ КАЛЕНДАРНОГО ПЛАНИРОВАНИЯ

Р е з ю м е

В работе представлен алгоритм локальной оптимизации проблем календарного планирования , в котором использована идея поиска с переменной глубиной Кернигана и Лина. Дано сравнение эффективности этого алгоритма со стандартными алгоритмами локальной оптимизации.

THE LOCAL SEARCH METHODS FOR SCHEDULING PROBLEMS

S u m m a r y

A local search algorithm for scheduling problems based on B.W. Kernighan's and s. Lin's variable-depth search idea is presented.
The efficiency of the proposed algorithm is compared to the standard local search algorithms.