ALGORYTM HEURYSTYCZNY DO OPTYMALIZACJI ROZDZIAŁU PROGRAMÓW W WIELOPROCESOROWYM SYSTEMIE INFORMATYCZNYM

Zbigniew BUCHALSKI

Streszczenie: W artykule przedstawiono zagadnienie czasowo-optymalnego przydziału *n* programów niezależnych niepodzielnych i *N* stron pamięci operacyjnej do procesorów pracujących równolegle. Czas wykonania *i*-tego programu na *k*-tym procesorze określony jest przez pewną funkcję zależną od liczby stron pamięci operacyjnej przydzielonych *k*-temu procesorowi oraz od parametrów charakteryzujących wykonywany program. Dla zadanej funkcji czasu realizacji programów zaproponowano pewien algorytm heurystyczny wyznaczający czasowo-optymalne szeregowanie programów i przydział stron pamięci operacyjnej do procesorów w wieloprocesorowym systemie informatycznym. Przedstawiono wyniki eksperymentów obliczeniowych przeprowadzonych na tym algorytmie.

Słowa kluczowe: wieloprocesorowy system informatyczny, szeregowanie programów, rozdział zasobów, algorytmy heurystyczne.

1. Wstęp

Rozwój równoległych systemów przetwarzania informacji pociągnął za sobą intensywny wzrost zainteresowania problematyką szeregowania zadań i rozdziału zasobów. Szczególnego znaczenia nabiera problem minimalizacji długości uszeregowania zadań na ma-szynach. Zadania optymalizacji zarówno dyskretnej, jak i ciągłej należą do klasy problemów bardzo trudnych zarówno z teoretycznego, jak i obliczeniowego punktu widzenia i najczęściej należą do klasy problemów *NP*-zupełnych, a więc są dość skomplikowane. Przy rozwiązywaniu tych problemów występują istotne trudności natury obliczeniowej.

Wyniki teorii złożoności obliczeniowej oraz rozmiar problemów praktycznych w sposób jednoznaczny eliminują z rozważań algorytmy dokładne, pozostawiając do zastosowania praktycznego jedynie algorytmy heurystyczne umożliwiające rozwiązanie postawionych problemów w krótkim czasie z zadowalająca dokładnością. Fakt ten jest typowy dla tej klasy problemów optymalizacji dyskretnej i w przypadku, kiedy zależy nam na krótkim czasie obliczeń, jedynym podejściem jest zastosowanie algorytmów heurystycznych. Ba-dania nad algorytmami heurystycznymi, dostarczającymi rozwiązań zagadnień, w których zastosowanie metod dokładnych jest nieefektywne lub wręcz niemożliwe, stanowią jedną z najszybciej rozwijających się gałęzi nauki [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20].

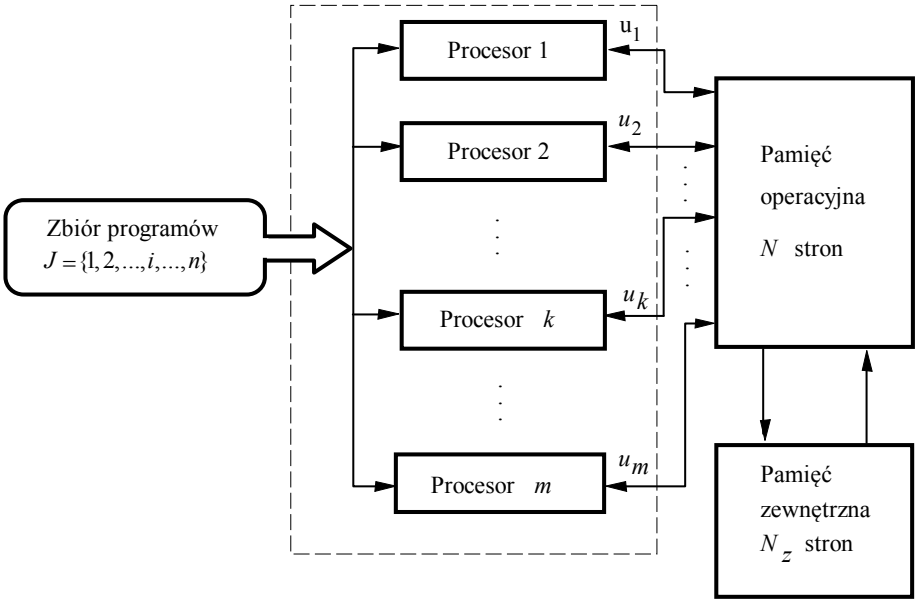
Celem niniejszej pracy jest znalezienie takiego uszeregowania *n* niezależnych niepodzielnych programów na *m* procesorach pracujących równolegle oraz takiego przydziału *N* stron pamięci operacyjnej do tych procesorów, aby minimalizować czas T_{zak} zakończenia wykonywania wszystkich programów. Sposób szeregowania programów

określa niezbędna do tego ilość zasobu w postaci stron pamięci operacyjnej. Tematyka ta poruszana już była we wcześniejszych pracach autora [21, 22, 23, 24].

W niniejszym artykule zaprezentowano pewien algorytm heurystyczny wyznaczający czasowo-optymalne szeregowanie n programów niezależnych niepodzielnych i N stron pamięci operacyjnej do m procesorów w wieloprocesorowym systemie informatycznym. Przedstawiono wyniki badań numerycznych przeprowadzonych na tym algorytmie dla losowo generowanych danych.

2. Sformułowanie problemu optymalizacji

Rozpatrzmy wieloprocesorowy system informatyczny przedstawiony na poniższym rysunku:



Rys. 1. Wieloprocesorowy system informatyczny

Na wieloprocesorowy system informatyczny nakładamy następujące założenia:

- pamięć operacyjna składa się z N stron o jednakowej objętości,
- każdy procesor może wykonywać dowolny program i ma dostęp do dowolnej strony pamięci operacyjnej,
- pamięć zewnętrzna zawiera N_z stron; $N_z > N$,
- liczba programów do wykonania jest większa od liczby procesorów; $n > m$ i każdy program wykonywany jest bez przerwań,
- strony pamięci operacyjnej są przydzielane procesorom i podczas wykonywania programów k-ty procesor może korzystać tylko z u_k stron pamięci operacyjnej jemu przydzielonych; $\sum_{k=1}^{m} u_k \le N, \quad u_k \ge 0, \quad 1 \le k \le m,$

realizacja każdego z programów na procesorach musi następować niezwłocznie po zakończeniu wykonywania poprzedniego programu lub nastąpić w chwili zerowej, gdy program realizowany jest jako pierwszy na dowolnym z procesorów.

Niech $J = \{1,2,...,i,...,n\}$ oznacza zbiór programów, $P = \{1,2,...,k,...,m\}$ - zbiór procesorów, a $U = \{1,2,...,N\}$ - zbiór stron pamięci operacyjnej. Czas wykonywania *i*-tego programu na *k*-tym procesorze określony jest przez następującą funkcję $T_i(u_k,k)$:

$$T_i(u_k,k) = a_{ik} + \frac{b_{ik}}{u_k}, \quad u_k \in U, \quad k \in P, \quad i \in J, \tag{1}$$

gdzie $a_{ik} > 0$, $b_{ik} > 0$ są parametrami charakteryzującymi *i*-ty program i *k*-ty procesor.

Przedstawiony do rozwiązania problem minimalizacyjny można scharakteryzować następująco: znajdź takie uszeregowanie programów na procesorach i taki przydział stron pa-mięci operacyjnej do procesorów, aby minimalizować czas T_{opt} zakończenia wykonywania całego zbioru programów *J*.

Jeżeli przez J_1, J_2, ... , J_k, ... , J_m oznaczymy zbiory programów realizowanych odpowiednio na 1, 2, ... , *k*, ... , *m* procesorze, to problem polega na znalezieniu takich zbiorów J_1, J_2, ... , J_k, ... , J_m i takich ilości stron pamięci operacyjnej u_1, u_2, ... , u_k, ... , u_m przydzielonych poszczególnym procesorom, które minimalizują następujące kryterium optymalizacji:

$$T_{opt} = \min_{\substack{J_1,J_2,...,J_m \\ u_1,u_2,...,u_m}} \max_{1\le k\le m}\left\{\sum_{i\in J_k} T_i(u_k,k)\right\} \tag{2}$$

przy następujących założeniach:

$$(i) \quad J_s \cap J_t = \varnothing; \quad s,t = 1,2,...,m, \quad s \neq t, \quad \bigcup_{k=1}^{m} J_k = J,$$

$$(ii) \quad \sum_{k=1}^{m} u_k \le N; \quad u_k \in U, \quad k = 1,2,...,m,$$

$$(iii) \quad u_1,u_2,...,u_m - \text{calkowite dodatnie.}$$

Założenie (iii) do kryterium (2) sprawia, że postawiony problem jest dość skomplikowany. W celu uproszczenia problemu przyjmiemy najpierw, że strony pamięci operacyjnej u_1, u_2, ... , u_k, ... , u_m są typu ciągłego i przy tym założeniu będziemy wyznaczać rozwiązanie problemu. Zaokrąglimy następnie otrzymane wartości optymalne u_1, u_2, ... , u_k, ... , u_m do najbliższych liczb naturalnych i wówczas rozpatrywany problem sprowadzi się do następującego problemu minimalizacji dyskretno-ciągłej:

$$T_{opt} = \min_{\substack{J_1,J_2,...,J_m \\ u_1,u_2,...,u_m}} \max_{1\le k\le m}\left\{\sum_{i\in J_k} \tilde{T}_i(u_k,k)\right\} \tag{3}$$

przy następujących ograniczeniach:

$$(i) \quad J_s \cap J_t = \varnothing; \quad s,t = 1,2,...,m, \quad s \neq t, \quad \bigcup_{k=1}^{m} J_k = J,$$

$$(ii) \quad \sum_{k=1}^{m} u_k \leq N; \quad u_k \geq 0, \quad u_k \in U, \quad k = 1,2,...,m,$$

gdzie: $\widetilde{T}_i : [0, N] \times \{1, 2, ..., m\} \to R^+$ jest rozszerzeniem następującej funkcji: $T_i : [1,2,...,N] \times \{1,2,...,m\} \to R^+$ i określony jest przez funkcję:

$$\widetilde{T}_i(u_k, k) = a_{ik} + \frac{b_{ik}}{u_k}, \quad u_k \in [0, N], \quad k \in P, \quad i \in J \,. \tag{4}$$

Przez u_k^*, J_k^*, k = 1, 2, ... , m oznaczmy rozwiązania zadania (3). W celu znalezienia tych rozwiązań pomocne będzie wykorzystanie poniższego **lematu**:

Lemat 1

Jeżeli u_k^*, J_k^*, k = 1, 2, ... , m są rozwiązaniami optymalnymi zadania (3), to:

$$(i) \quad \sum_{k=1}^{m} u_k^* = N; \quad u_k^* > 0, \qquad k : J_k^* \neq \varnothing, \quad k = 1,2,...,m,$$

$$u_k^* = 0, \qquad k : J_k^* = \varnothing, \quad k = 1,2,...,m,$$

$$(ii) \quad \sum_{i \in J_k^*} \widetilde{T}_i(u_k^*, k) = \text{const}; \quad k : J_k^* \neq \varnothing, \quad k = 1,2,...,m.$$

Warunek (*i*) w **Lemacie 1** mówi, że w przydziale czasowo-optymalnym stron pamięci operacyjnej i programów do procesorów wykorzystuje się wszystkie N stron, a warunek (*ii*), że czasy pracy tych procesorów, które wykonują jakieś programy są identyczne.

Zdefiniujmy funkcję $F(J_1, J_2, ... , J_m)$ określoną dla zbiorów $J_1, J_2, ... , J_m$, dla których zachodzi ograniczenie (*i*) dla kryterium optymalizacji (3). Wartość tej funkcji jest rozwiązaniem następującego układu równań:

$$\begin{cases} \sum_{i \in J_k} a_{ik} + \dfrac{\sum_{i \in J_k} b_{ik}}{u_k} \overset{\Delta}{=} F(J_1, J_2, ..., J_m); \qquad k : J_k \neq \varnothing, \quad k = 1,2,...,m \\ \sum_{k : J_k \neq \varnothing} u_k = N; \quad u_k > 0, \qquad k : J_k \neq \varnothing, \quad k = 1,2,...,m. \end{cases} \tag{5}$$

Wykorzystując **Lemat 1** oraz (5) zadanie (3) przyjmie ostatecznie następującą postać:

$$T_{opt} = \min_{J_1, J_2, ..., J_m} F(J_1, J_2, ..., J_m) \tag{6}$$

przy następujących ograniczeniach:

$$(i) \quad J_s \cap J_t = \varnothing; \quad s,t = 1,2,...,m, \quad s \neq t,$$

$$(ii) \quad \bigcup_{k=1}^{m} J_k = J; \quad k = 1,2,...,m.$$

Jeżeli $J_1^*, J_2^*, ..., J_m^*$ jest rozwiązaniem zadania (6), to u_k^*, J_k^*, $k = 1,2,...,m$, gdzie:

$$u_k^* = \begin{cases} \dfrac{\sum_{i \in J_k^*} b_{ik}}{F(J_1^*, J_2^*, ..., J_m^*) - \sum_{i \in J_k^*} a_{ik}}; & k : J_k^* \neq \varnothing, \quad 1 \leq k \leq m \\ 0 \qquad\qquad\qquad\qquad ; & k : J_k^* = \varnothing, \quad 1 \leq k \leq m \end{cases} \tag{7}$$

jest rozwiązaniem zadania (3).

3. Algorytm heurystyczny

Procesory wchodzące w skład wieloprocesorowego systemu informatycznego różnią się pod względem szybkości wykonywania programów. Decyduje o tym liczba stron pamięci operacyjnej przydzielonych poszczególnym procesorom. Dlatego też *k*-ty procesor będzie tym szybszy, im więcej stron pamięci operacyjnej u_k zostanie mu przydzielonych. Poniżej przedstawiony zostanie algorytm heurystyczny, który najpierw szereguje pro-gramy na jednakowych procesorach, tj. takich, do których przydzielona została jednakowa liczba dostępnych stron $u_k = \frac{N}{m}, k \in P$. Po tym uszeregowaniu następuje zróżnicowanie procesorów pod względem liczby przydzielonych im stron pamięci operacyjnej i sprawdzenie, czy skrócony został czas zakończenia wykonywania wszystkich programów $T_{opt.}$. Strony pamięci operacyjnej przydzielone zostają do procesorów w następujący sposób: miarą szybkości realizacji *i*-tego programu przez *k*-ty procesor jest tzw. współczynnik podziału stron pamięci operacyjnej β; $\beta > 1$, zakładamy, że procesorem najszybszym jest procesor pierwszy, a procesorem najwolniejszym jest procesor *m*-ty. Jeżeli procesorowi najwolniejszemu przydzielimy u_m stron pamięci operacyjnej, to do pozostałych procesorów przydział tych stron będzie wyglądał następująco:

$u_1 = (m-1) \cdot \beta \cdot u_m$

$u_2 = (m-2) \cdot \beta \cdot u_m$

$\vdots$

$u_k = (m-k) \cdot \beta \cdot u_m$

$\vdots$

$u_{m-2} = [m-(m-2)] \cdot \beta \cdot u_m = 2\beta \cdot u_m$

$u_{m-1} = [m-(m-1)] \cdot \beta \cdot u_m = \beta \cdot u_m$.

Jak wiadomo, pojemność pamięci operacyjnej wynosi *N* stron, a zatem:

$$\sum_{k=1}^{m} u_k = N. \tag{8}$$

Rozwijając sumę (8) oraz wprowadzając do niej parametr β otrzymujemy:

$$(m-1)\cdot\beta\cdot u_m+(m-2)\cdot\beta\cdot u_m+\ldots\ldots+(m-k)\cdot\beta\cdot u_m+\ldots+2\cdot\beta\cdot u_m+ \\ +\beta\cdot u_m+u_m=N. \tag{9}$$

Z zależności (9) wyliczamy wartość u_m dla *m*-tego procesora, czyli procesora najwolniejszego:

$$u_m+\sum_{k=1}^{m-1}[(m-k)\cdot\beta\cdot u_m]=N,$$

a zatem procesorowi *m*-temu przydzielimy następującą liczbę stron pamięci operacyjnej:

$$u_m=\frac{N}{1+\sum_{k=1}^{m-1}[(m-k)\cdot\beta]}. \tag{10}$$

Pozostałe procesory otrzymują liczbę stron pamięci operacyjnej określoną następującą zależnością:

$$u_k=(m-k)\cdot\beta\cdot u_m,\quad k=1,2,\ldots,m-1. \tag{11}$$

Przedstawiony powyżej sposób przydziału stron pamięci operacyjnej do procesorów wykorzystany zostanie w zaproponowanym w niniejszej pracy algorytmie heurystycznym.

Kolejne kroki algorytmu heurystycznego są następujące:

Krok 1. Oblicz czasy wykonywania programów na poszczególnych procesorach $T_i(u_k,k)=a_{ik}+\frac{b_{ik}}{u_k}$, $i\in J$, $k\in P$ dla zadanej wartości $u_k=\frac{N}{m}$ i losowo generowanych parametrów a_{ik}, b_{ik}.

Krok 2. Uszereguj malejąco czasy wykonywania poszczególnych programów i utwórz listę *L* tych programów.

Krok 3. Oblicz średni czas $T_{śr}$ wykonywania programów przez każdy z procesorów wg wzoru:

$$T_{śr}=\frac{\sum_{i=1}^{n}T_i(u_k,k)}{m};\quad i\in J,k\in P,u_k=\frac{N}{m}.$$

Krok 4. Przydzielaj kolejno najdłuższe programy z listy *L* do kolejnych procesorów (od pierwszego poczynając) aż do momentu, gdy suma czasów wykonywania programów przydzielonych kolejnym procesorom nie przekroczy czasu $T_{śr}$. Przydzielone programy usuń z listy *L*.

Krok 5. Jeżeli lista *L* się jeszcze nie wyczerpała to przydziel na przemian najkrótszy program z listy *L* do procesora, który ma najdłuższy czas wykonywania programów i najdłuższy program z listy *L* do procesora, który ma najkrótszy czas wykonywania programów jemu przydzielonych. Usuń te dwa ostatnio przydzielone programy z listy *L*.

Krok 6. Jeżeli lista *L* nie została jeszcze wyczerpana to wróć do **Kroku 5**. W przeciwnym wypadku przejdź do **Kroku 7**.

Krok 7. Oblicz czas zakończenia wykonywania wszystkich programów T_{opt} dla uszeregowania programów na procesorach utworzonego w **Krokach 4÷6** i dla $u_k = \frac{N}{m}$.

Krok 8. Oblicz sumaryczne czasy wykonywania programów uszeregowanych na poszczególnych procesorach.

Krok 9. Dla zadanego współczynnika β przydziel strony u_k, $k \in P$ poszczególnym procesorom wyliczone z zależności (10) i (11).

Krok 10. Dla uszeregowania programów na procesorach utworzonego **w Krokach 4÷8** i dla liczby stron u_k przydzielonych procesorom w **Kroku 9** oblicz czas zakończenia wykonywania wszystkich programów T_{opt}

Krok 11. Powtórz **Krok 9** i **Krok 10** dla następnych sześciu zwiększających się kolejno wartości współczynnika β . Po zakończeniu tych prób przejdź do **Kroku 12.**

Krok 12. Porównaj wartości czasów zakończenia wykonywania programów T_{opt} z kolejnych prób i wybierz najkrótszy z tych czasów.

Krok 13. Wyznacz dyskretne liczby stron $\hat{u}_k$, $k \in P$ według zależności:

$$\hat{u}_{\alpha(k)} = \begin{cases} \lfloor u_{\alpha(k)} \rfloor + 1; & k = 1, 2, \ldots, \Delta, \\ \lfloor u_{\alpha(k)} \rfloor & ; \quad k = \Delta + 1, \Delta + 2, \ldots, m, \end{cases}$$

gdzie $\Delta = N - \sum_{j=1}^{m} \lfloor u_j \rfloor$ oraz α jest permutacją elementów zbioru $P = \{1,2,\ldots,m\}$ taką, że $u_{\alpha(1)} - \lfloor u_{\alpha(1)} \rfloor \geq u_{\alpha(2)} - \lfloor u_{\alpha(2)} \rfloor \geq \ldots \geq u_{\alpha(m)} - \lfloor u_{\alpha(m)} \rfloor$. Jeżeli istnieją takie procesory, którym przydzielono zerowe liczby stron pamięci operacyjnej, to przydziel każdemu z tych procesorów po jednej stronie pobierając je z kolejnych procesorów poczynając od procesora, któremu przydzielono największą liczbę stron pamięci operacyjnej.

4. Wyniki eksperymentów obliczeniowych

Przedstawiony w pracy algorytm heurystyczny poddano ocenie dla siedmiu zwiększających się kolejno wartości współczynnika podziału stron pamięci operacyjnej β ze zbioru {3.0, 6.0, … , 21.0}. Parametry a_{ik}, b_{ik} charakteryzujące *i*-ty program i *k*-ty procesor wylosowane zostały ze zbioru {4.0, 8.0, … , 80.0} przez generator o jednostajnym rozkładzie prawdopodobieństwa. Zadano liczbę programów n = 30, 60, 90, 120, 150 i liczbę procesorów m = 4, 8, 12, 16, 20 oraz liczbę stron pamięci operacyjnej N = 10.000. Dla każdej kombinacji n i m wygenerowano 30 instancji. Rezultaty analizy porównawczej algo-rytmu heurystycznego przedstawionego w niniejszej pracy i znanego z literatury algorytmu *LPT* przedstawione zostały w Tab.1.

Tab. 1.Wyniki analizy porównawczej algorytmu heurystycznego i algorytmu *LPT*

n/m	Liczba instancji, dla których:			Δ^H	S^H	S^{LPT}
	$T_{opt}^H < T_{opt}^{LPT}$	$T_{opt}^H = T_{opt}^{LPT}$	$T_{opt}^H > T_{opt}^{LPT}$	%	sek	sek
30/4	16	1	13	1,7	1,7	1,6
60/4	16	0	14	2,6	2,6	2,4
90/4	17	1	12	3,3	4,1	3,9
120/4	18	2	10	3,7	5,9	4,7
150/4	19	2	9	3,9	7,1	5,9
30/8	16	0	14	1,8	2,7	2,1
60/8	15	1	14	2,8	3,3	2,7
90/8	17	1	12	3,7	5,2	4,6
120/8	17	2	11	4,2	6,6	5,5
150/8	18	2	10	4,6	7,8	5,9
30/12	15	0	15	2,0	3,0	2,2
60/12	16	0	14	3,7	4,1	2,9
90/12	16	1	13	4,2	6,0	4,9
120/12	17	2	11	4,9	7,8	6,4
150/12	18	3	9	5,1	8,5	6,9
30/16	16	0	14	2,4	3,4	2,8
60/16	16	1	13	2,9	4,9	3,7
90/16	18	2	10	3,8	6,7	4,8
120/16	20	3	7	4,9	9,6	7,8
150/16	21	2	7	5,8	12,8	10,9
30/20	17	0	13	2,6	3,9	3,2
60/20	19	1	10	3,9	6,8	5,6
90/20	20	2	8	4,7	9,8	7,8
120/20	21	2	7	5,3	12,6	10,7
150/20	22	3	5	5,9	14,9	12,8

W Tab.1. występują następujące wielkości:

m – liczba procesorów,
n – liczba programów,
T_{opt}^H – czas zakończenia wykonywania wszystkich programów ze zbioru J przy wykorzystaniu algorytmu heurystycznego,
T_{opt}^{LPT} – czas zakończenia wykonywania wszystkich programów ze zbioru J przy wykorzystaniu algorytmu *LPT*,
Δ^H – średnia procentowa poprawa czasu T_{opt}^H w stosunku do czasu T_{opt}^{LPT} wyrażona następującym wzorem: $\Delta^H = \frac{T_{opt}^{LPT} - T_{opt}^H}{T_{opt}^H} \cdot 100\%$,
S^H – średni czas obliczeń dla algorytmu heurystycznego,
S^{LPT} – średni czas obliczeń dla algorytmu *LPT*.

5. Uwagi końcowe

Przedstawione w poprzednim punkcie pracy eksperymenty obliczeniowe wykazały, że efektywność szeregowania programów na równoległych procesorach na bazie zaproponowanego w pracy algorytmu heurystycznego uległa poprawie w stosunku do szeregowania za pomocą znanego z literatury algorytmu *LPT*. Kilkuprocentowa poprawa czasu T_{opt}^{H} w stosunku do T_{opt}^{LPT} może być zachętą do dalszych prac nad efektywnymi algorytmami heurystycznymi.

Zastosowanie przedstawionego algorytmu heurystycznego jest wskazane przede wszystkim dla systemów o dużej liczbie programów, gdyż wówczas średnia procentowa poprawa Δ^{H} jest największa. Zaproponowany w pracy algorytm może służyć zarówno do szeregowania programów w wieloprocesorowych systemach komputerowych, jak i do rozdziału operacji na stanowiska produkcyjne wyposażone w odpowiednie maszyny w dyskretnym systemie produkcyjnym.

Literatura

1. Bianco L. i in.: Linear algorithms for preemptive scheduling of multiprocessor tasks subject to minimal lateness. Discrete Applied Mathematics, 72, 1997, 25-46.
2. Błażewicz J., Dell'Olmo P., Drozdowski M., Speranza M. G.: Scheduling multiprocessor tasks on three dedicated processors. Information Processing Letters 41, 1992, 275-280.
3. Błażewicz J. i in.: Scheduling in Computer and Manufacturing Systems. Springer-Verlag, Berlin-Heidelberg 1993.
4. Błażewicz J. i in.: Scheduling independent multiprocessor tasks before deadlines. Discrete Applied Mathematics 65 (1-3), 1996, 81-96.
5. Błażewicz J., Liu Z.: Scheduling multiprocessor tasks with chain constraints. European Journal of Operational Research, 94, 1996, 231-241.
6. Boctor F. F.: A new and efficient heuristic for scheduling projects with resources restrictions and multiple execution models. European Journal of Operational Research, Vol. 90, 1996, 349-361.
7. Brah S.A., Loo L.L.: Heuristics for scheduling in a flow shop with multiple processors, European Journal of Operational Research, Vol. 113, No. 1, 1999, 113-122.
8. Cheng J., Karuno Y., Kise H.: A shifting bottleneck approach for a parallel-machine flow-shop scheduling problem, Journal of the Operational Research Society of Japan, Vol. 44, No. 2, 2001, 140-156.
9. Gupta J.N.D., Hariri A.M.A., Potts C.N.: Scheduling a two-stage hybrid flow shop with parallel machines at the first stage, Annals of Operations Research, Vol. 69, No. 0, 1997, 171-191.
10. Hoogeveen J.A., Lenstra J.K., Veltman B.: Preemptive scheduling in a two-stage multi-processor flow shop in NP-hard, European Journal of Operational Research, Vol. 89, No.1, 1996, 172-175.
11. Janiak A.: Single machine scheduling problem with a common deadline and resource dependent release dates. European Journal of Operational Research, Vol. 53, 1991, 317-325.

12. Janiak A., Kovalyov M.: Single machine scheduling subject to deadlines and resources dependent processing times. European Journal of Operational Research, 1996, Vol. 94, 284-291.
13. Józefowska J. i in.: Discrete-continuous scheduling to minimize maximum lateness, Proceedings of the Fourth International Symposium on Methods and Models in Automation and Robotics MMAR'97, Międzyzdroje, Poland 1997, 947-952.
14. Józefowska J., Węglarz J.: On a methodology for discrete-continuous scheduling. European Journal of Operational Research, Vol. 107, 1998, 338-353.
15. Józefowska J. i in.: Local search metaheuristics for discrete-continuous scheduling problems, European Journal of Operational Research, 107, 1998, 354-370.
16. Józefowska J., Mika M., Różycki R., Waligóra G., Węglarz J.: Rozwiązywanie dyskretno-ciągłych problemów rozdziału zasobów przez dyskretyzację zasobu ciągłego. Zeszyty Naukowe Politechniki Śląskiej Nr 1474, seria Automatyka, Gliwice 2000, z. 129, 221-229.
17. Kubale M., Giaro K.: Złożoność zwartego szeregowania zadań jednostkowych w systemie otwartym, przepływowym i mieszanym. Uczelniane Wydawnictwo Naukowo-Dydaktyczne AGH, seria-Automatyka, półrocznik, tom 5, zeszyt ½, Kraków 2001, 329-334.
18. Ng C.T. i in.: Group scheduling with controllable setup and processing times: minimizing total weighted completion time, Ann. Oper. Res., 133, 2005, 163-174.
19. Nowicki E., Smutnicki C.: The flow shop with parallel machines. A tabu search approach. European Journal of Operational Research 106, 1998, 226-253..
20. Węglarz J.: Multiprocessor scheduling with memory allocation - a deterministic approach. IEEE Trans. Comput., C-29, 1980, 703-710.
21. Buchalski Z.: An heuristic solution procedure to minimize the total processing time of programs in multiprocessing computer system. Information Systems Architecture and Technology ISAT 2005, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2005, 20-26.
22. Buchalski Z.: A Program Scheduling Heuristic Algorithm in Multiprocessing Computer System with Limited Memory Pages. Polish Journal of Environmental Studies, Vol. 15, No. 4C, 2006, 26-29.
23. Buchalski Z.: Minimising the Total Processing Time for the Tasks Scheduling on the Parallel Machines System. Proc. of the 12th IEEE International Conference on Methods and Models in Automation and Robotics, Domek S., Kaszyński R. (Eds.), Międzyzdroje, Poland, MMAR 2006, 28-31 August 2006, 1081-1084.
24. Buchalski Z.: An Heuristic Algorithm for Solving the Scheduling Problem in Multiprocessing Computer System. Polish Journal of Environmental Studies, Vol. 16, No. 4A, 2007, 44-48.

Dr inż. Zbigniew BUCHALSKI
Instytut Informatyki, Automatyki i Robotyki
Politechnika Wrocławska
50-372 Wrocław, Janiszewskiego 11/17
tel.: (0 71) 320 32 92
e-mail: zbigniew.buchalski@pwr.wroc.pl