HEURYSTYCZNY ALGORYTM SZEREGOWANIA ZADAŃ W SYSTEMIE MASZYN RÓWNOLEGŁYCH Z KRYTERIUM MINIMALNO-CZASOWYM

Zbigniew BUCHALSKI

Streszczenie: Artykuł dotyczy zagadnienia czasowo-optymalnego przydziału zasobu podzielnego w sposób ciągły i *n* zadań do *m* maszyn równoległych. Założono, że zadania są niezależne i niepodzielne. Dla zadanej funkcji czasu realizacji zadań sformułowano model formalny zagadnienia i zaproponowano pewien algorytm heurystyczny wyznaczający czasowo-optymalne szeregowanie zadań i przydział zasobów do maszyn równoległych. Przed-stawiono wyniki eksperymentów obliczeniowych przeprowadzonych na tym algorytmie.

Słowa kluczowe: systemy maszyn równoległych, szeregowanie zadań, rozdział zasobów, algorytmy heurystyczne.

1. Wstęp

Odczuwalny od wielu lat gwałtowny rozwój równoległych systemów przetwarzania informacji pociągnął za sobą potrzebę rozwiązywania problemów czasowo-optymalnego szeregowania zadań i rozdziału zasobów [1, 2, 3, 4, 5]. Znaczenie praktyczne rozwiązywania tych problemów jest bezsporne w wielu dziedzinach życia. Problemy te mogą być związane przykładowo ze sterowaniem procesem produkcyjnym, wyznaczaniem kolejnych etapów montażu urządzeń, harmonogramowaniem zadań transportowych, itp.

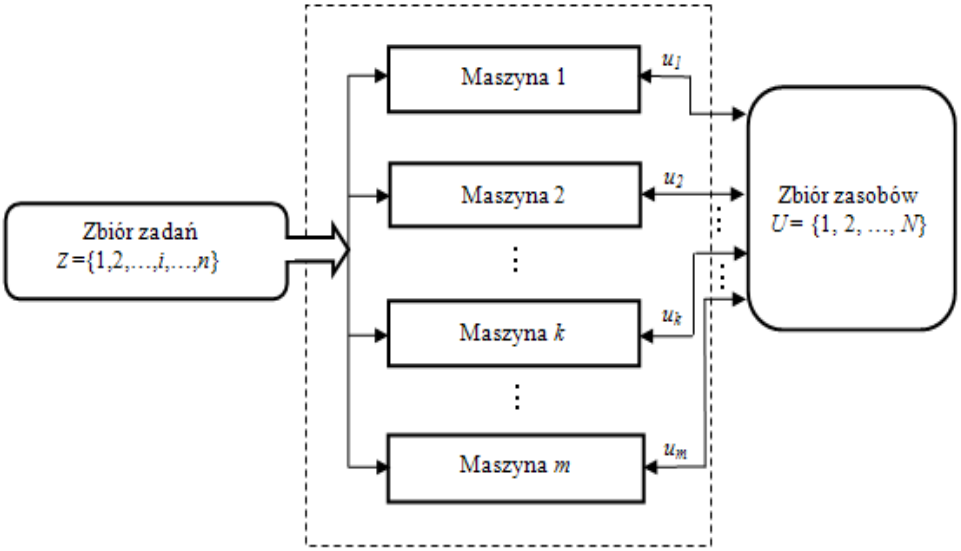
Zadania optymalizacji zarówno dyskretnej, jak i ciągłej należą do klasy problemów bardzo trudnych zarówno z teoretycznego, jak i obliczeniowego punktu widzenia i najczęściej należą do klasy problemów *NP*- zupełnych [6, 7, 8]. Doświadczenia z podejmowania prób optymalizacji tego typu problemów wskazują na możliwość osiągnięcia tą drogą znacznych efektów ekonomicznych. Istnieje więc pilna potrzeba prowadzenia badań w tym zakresie i to zarówno podstawowych, jak i stosowanych.

W systemach maszyn równoległych spotykamy się z szeregowaniem zadań na maszynach oraz przydziałem zasobów do maszyn. Przy rozwiązywaniu tych problemów występują istotne trudności natury obliczeniowej w związku z czym eliminuje się z rozważań algorytmy dokładne, pozostawiając do zastosowania praktycznego jedynie algorytmy heurystyczne umożliwiające rozwiązanie postawionych problemów w krótkim czasie z zadowalającą dokładnością [9, 10, 11].

W niniejszym artykule przedstawiono pewien algorytm heurystyczny wyznaczający czasowo-optymalny rozdział *n* zadań niezależnych niepodzielnych i *N* jednostek zasobu nieodnawialnego podzielnego w sposób ciągły do *m* maszyn pracujących równolegle. Przedstawiono wyniki eksperymentów obliczeniowych przeprowadzonych na tym algorytmie dla losowo generowanych danych.

2. Sformułowanie problemu

Rozpatrzmy dyskretny system produkcyjny zawierający maszyny połączone równolegle przedstawiony na poniższym rysunku:



Rys. 1. System maszyn równoległych

Na system maszyn równoległych nakładamy następujące założenia:

(i) posiada m różnych maszyn $M = \{1, 2,..., k,... , m\}$, na których należy wykonać n niezależnych zadań $Z = \{1, 2,..., i,..., n\}$,

(ii) zadanie może być wykonywane na dowolnej maszynie i w trakcie jego wykonywania nie może być przerywane,

(iii) liczba zadań do wykonania jest większa od liczby maszyn $n > m$,

(iv) realizacja każdego z zadań na maszynach musi następować niezwłocznie po zakończeniu wykonywania poprzedniego zadania lub nastąpić w chwili zerowej, gdy za-danie realizowane jest jako pierwsze na jednej z maszyn.

Niech N oznacza globalną ilość zasobów nieodnawialnych, a przez u_k oznaczmy tą część zasobów, które zostaną przydzielone k-tej maszynie w trakcie wykonywania zadań uszeregowanych na tej maszynie. Ograniczenie dotyczące zasobów jest następujące:

$$\sum_{k=1}^{m} u_k \le N, \quad u_k \ge 0, \quad 1 \le k \le m.$$

Czas wykonywania i-tego zadania na k-tej maszynie określony jest przez następującą funkcję $T_i(u_k, k)$:

$$T_i(u_k, k) = a_{ik} + \frac{b_{ik}}{u_k}, \quad u_k \in \{1,2,\ldots, N\}, \quad 1 \le k \le m, \quad 1 \le i \le n. \tag{1}$$

Parametry $a_{ik} > 0$ i $b_{ik} > 0$ charakteryzują *i*-te zadanie i *k*-tą maszynę.

Należy znaleźć takie uszeregowanie zadań na maszynach i taki przydział ograniczonych zasobów do maszyn równoległych, aby minimalizować czas zakończenia wykonania całego zbioru zadań T_{zak}.

3. Model formalny zagadnienia

Jeżeli oznaczymy przez $Z_k \subset Z$ zbiór zadań uszeregowanych na *k*-tej maszynie, to T_{zak} znajdziemy rozwiązując następujący problem minimalizacyjny:

$$T_{zak} = \min_{\substack{Z_1, Z_2, \ldots, Z_m \\ u_1, u_2, \ldots, u_m}} \max_{1 \le k \le m} \left\{ \sum_{i \in Z_k} T_i(u_k, k) \right\}. \tag{2}$$

Ograniczenia nałożone na rozwiązanie tego problemu są następujące:

(i) $Z_r \cap Z_s = \phi; \quad r, s = 1,2,\ldots,m, \quad r \ne s, \quad \bigcup_{k=1}^{m} Z_k = Z,$

(ii) $\sum_{k=1}^{m} u_k \le N,$

(iii) $u_1, u_2, \ldots, u_m$ - całkowite dodatnie.

Dla uproszczenia problemu przyjmiemy najpierw, że zasoby nieodnawialne u_1, u_2, ..., u_m są typu ciągłego. Przy tym założeniu wyznaczymy rozwiązanie optymalne, a następnie zaokrąglimy otrzymane wartości zasobów do najbliższych liczb naturalnych.
Czas T_{zak} znajdziemy rozwiązując następujący problem minimalizacji dyskretno-ciągłej:

$$T_{zak} = \min_{\substack{Z_1, Z_2, \ldots, Z_m \\ u_1, u_2, \ldots, u_m}} \max_{1 \le k \le m} \left\{ \sum_{i \in Z_k} T_i'(u_k, k) \right\} \tag{3}$$

przy następujących ograniczeniach:

(i) $Z_r \cap Z_s = \phi; \quad r, s = 1,2,\ldots,m, \quad r \ne s, \quad \bigcup_{k=1}^{m} Z_k = Z,$

(ii) $\sum_{k=1}^{m} u_k \le N; \quad u_k \ge 0, \quad k = 1,2,\ldots,m,$

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

$$T_i'(u_k,k)=a_{ik}+\frac{b_{ik}}{u_k},\quad u_k\in[0,N],\quad 1\le k\le m,\quad 1\le i\le n. \tag{4}$$

Do rozwiązania postawionego problemu pomocny będzie następujący lemat:

LEMAT 1

Jeżeli $u_k^*, Z_k^*, k=1,2,\ldots,m$ są rozwiązaniami optymalnymi zadania (3), to:

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

$$u_k^*=0,\quad k: Z_k^*=\phi,\quad k=1,2,...,m;$$

(ii) $$\sum_{i\in Z_k^*} T_i'(u_k^*,k)=const;\quad k: Z_k^*\neq\phi,\quad k=1,2,...,m.$$

Warunek (i) w **LEMACIE 1** oznacza, że w przydziale czasowo-optymalnym zasobów i zadań do maszyn wykorzystuje się wszystkie jednostki zasobów, a warunek (ii), że czasy pracy tych maszyn, na których wykonywane są jakieś zadania, są identyczne.

Zdefiniujmy funkcję $F(Z_1, Z_2, ..., Z_m)$ określoną dla m zbiorów $Z_1, Z_2, ..., Z_m$, dla których zachodzi ograniczenie (i) dla wzoru (3). Wartość tej funkcji jest rozwiązaniem następującego układu równań:

$$\begin{cases} \sum\limits_{i\in Z_k} a_{ik} + \dfrac{\sum\limits_{i\in Z_k} b_{ik}}{u_k} = F(Z_1, Z_2, \ldots, Z_m);\quad k: Z_k\neq\varnothing,\quad k=1, 2, \ldots, m \\ \sum\limits_{k=1}^{m} u_k = N;\quad u_k>0,\quad k: Z_k\neq\varnothing,\quad k=1, 2, \ldots, m. \end{cases} \tag{5}$$

Wykorzystując **LEMAT 1** oraz (5) zadanie minimalizacji (3) można przedstawić w następującej postaci:

$$T_{zak}=\min_{Z_1,Z_2,\ldots,Z_m} F(Z_1, Z_2,..., Z_m), \tag{6}$$

przy następujących ograniczeniach:

(i) $Z_r\cap Z_s=\phi,\quad r,s=1,2,...,m,\quad r\neq s,$

(ii) $\bigcup\limits_{k=1}^{m} Z_k = Z.$

Jeżeli $Z_1^*, Z_2^*,..,Z_m^*$ jest rozwiązaniem zadania (6), to $u_k^*, Z_k^*, k = 1,2,\ldots,m$, gdzie:

$$u_k^* = \begin{cases} \dfrac{\sum_{i \in Z_k^*} b_{ik}}{F\left(Z_1^*, Z_2^*, \ldots, Z_m^*\right) - \sum_{i \in Z_k^*} a_{ik}}; & k: Z_k^* \neq \phi, \quad 1 \le k \le m, \\ 0 \qquad\qquad\qquad\qquad\quad ; & k: Z_k^* = \phi, \quad 1 \le k \le m \end{cases} \tag{7}$$

jest rozwiązaniem zadania (3).

4. Algorytm heurystyczny

Maszyny wchodzące w skład systemu maszyn równoległych różnią się pod względem szybkości wykonywanych zadań. Na szybkość tą wpływ ma ilość zasobów przydzielonych poszczególnym maszynom. Im więcej zasobów zostanie przydzielonych *k*-tej maszynie, tym będzie ona szybsza.
Zasoby przydzielone zostają do maszyn w następujący sposób:

- miarą szybkości realizacji *i*-tego zadania przez *k*-tą maszynę jest tzw. współczynnik podziału zasobów β; $\beta > 1$,
- zakładamy, że maszyną najszybszą jest maszyna pierwsza, a maszyną najwolniejszą jest maszyna *m*-ta,
- maszynie *m*-tej przydzielamy u_m zasobów wg następującej zależności:

$$u_m = \frac{N}{1 + \sum_{k=1}^{m-1} [(m-k) \cdot \beta]} \tag{8}$$

- pozostałym maszynom przydzielamy zasoby wg następującej zależności:

$$u_k = (m-k) \cdot \beta \cdot u_m; \qquad k = 1, 2, \ldots, m-1 \tag{9}$$

Przedstawiony powyżej sposób przydziału zasobów do maszyn wykorzystany zostanie w zaproponowanym heurystycznym algorytmie szeregowania zadań na równoległych maszynach. Algorytm ten skonstruowany został w taki sposób, że najpierw szereguje on zadania na jednakowych maszynach, tj. takich, do których przydzielona została jednakowa liczba dostępnych zasobów, czyli $u_k = \frac{N}{m}, k = 1,2,\ldots,m$ (**Kroki 4÷11**). Po tym uszeregowaniu następuje zróżnicowanie maszyn pod względem liczby przydzielanych im zasobów i sprawdzenie czy skrócony został czas zakończenia wykonywania wszystkich zadań T_{zak} (**Kroki 12÷15**).

Algorytm heurystyczny wyznaczający czasowo-optymalne szeregowanie zadań niezależnych na maszynach równoległych ma następującą postać:

Krok 1. Oblicz czasy wykonywania zadań na poszczególnych maszynach $T_i(u_k, k) = a_{ik} + \frac{b_{ik}}{u_k}$, $i = 1, 2, \ldots, n$, $k = 1, 2, .., m$ dla losowo generowanych parametrów a_{ik}, b_{ik} i zadanej wartości $u_k = \frac{N}{m}$.

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

Krok 3. Oblicz średni czas $T_{śr}$ wykonywania zadań przez każdą z maszyn wg wzoru:

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

Krok 4. Przydzielaj kolejno najdłuższe i najkrótsze zadania z listy L do pierwszej wolnej maszyny aż do momentu, gdy suma czasów wykonywania zadań przydzielonych tej maszynie nie przekroczy czasu $T_{śr}$. Przydzielone zadania usuń z listy L.

Krok 5. Jeżeli są jeszcze maszyny na których nie uszeregowano żadnych zadań to wróć do **Kroku 4**. W przeciwnym wypadku przejdź do **Kroku 6**.

Krok 6. Przydzielaj kolejno najkrótsze zadania z listy L do kolejnych maszyn od pierwszej poczynając a na m-tej kończąc aż do momentu, gdy suma czasów realizacji zadań przez kolejne maszyny nie przekroczy czasu $T_{śr}$. Przydzielone zadania usuń z listy L.

Krok 7. Jeżeli lista L nie została jeszcze wyczerpana to pozostałe zadania przydziel do maszyn wg algorytmu *LPT*.. W przeciwnym wypadku przejdź do **Kroku 8.**

Krok 8. Oblicz czas zakończenia wykonywania wszystkich zadań T_{zak} dla uszeregowania zadań na maszynach utworzonego w **Krokach 4÷7** i dla $u_k = \frac{N}{m}$.

Krok 9. Oblicz sumaryczne czasy wykonywania zadań uszeregowanych na poszczególnych maszynach.

Krok 10. Usuń najkrótsze zadanie z maszyny o najdłuższym sumarycznym czasie wykonywania zadań i przydziel je do maszyny o najkrótszym sumarycznym czasie wykonywania zadań.

Krok 11. Oblicz czas zakończenia wykonywania wszystkich zadań T_{zak} po zamianie zadań w **Kroku 10**. Jeżeli czas ten ulegnie skróceniu, to wróć do **Kroku 9**. W przeciwnym wypadku anuluj ostatnio wykonaną czynność w **Kroku 10** i zakończ szeregowanie zadań na maszynach.

Krok 12. Dla zadanego współczynnika β przydziel zasoby u_k, $k = 1, 2, \ldots, m$ poszczególnym maszynom wyliczone z zależności (8) i (9).

Krok 13. Dla uszeregowania zadań na maszynach utworzonego w **Krokach 4÷11** i dla liczby zasobów u_k przydzielonych maszynom w **Kroku 12** oblicz czas zakończenia wykonywania wszystkich zadań T_{zak}.

Krok 14. Powtórz **Krok 12** i **Krok 13** dla następnych siedmiu zwiększających się kolejno wartości współczynnika β. Po zakończeniu tych prób przejdź do **Kroku 15**.

Krok 15. Porównaj wartości czasów zakończenia wykonywania zadań T_{zak} z kolejnych prób i wybierz najkrótszy z tych czasów.

Krok 16. Wyznacz dyskretne ilości zasobów $\hat{u}_{\alpha(k)}, k = 1, 2, \ldots, m$ według zależności:

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

gdzie $\Delta = N - \sum_{j=1}^{m} \lfloor u_j \rfloor$ oraz α jest taką permutacją elementów zbioru $M = \{1, 2, \ldots, m\}$, 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 maszyny, którym przydzielono zerowe ilości zasobów, to przydziel każdej z tych maszyn po jednej jednostce zasobu pobierając je z kolejnych maszyn poczynając od maszyny, której przydzielono największą ilość zasobów.

5. Wyniki eksperymentów obliczeniowych

Na bazie przedstawionego w poprzednim punkcie pracy algorytmu heurystycznego przeprowadzono eksperymenty obliczeniowe. Algorytm wykonuje za każdym razem osiem prób znalezienia najlepszego, z punktu widzenia czasu realizacji zbioru zadań, rozwiązania. Po wykonaniu serii prób z różnymi wartościami współczynnika podziału zasobów β, porównywane są ze sobą czasy realizacji zbioru zadań i wybierany jest najkrótszy z nich.

Eksperymenty obliczeniowe przeprowadzono dla ośmiu zwiększających się kolejno wartości współczynnika podziału zasobów β ze zbioru {4, 8, … , 32}. Parametry a_{ik}, b_{ik}, charakteryzujące *i*-te zadanie i *k*-tą maszynę wylosowane zostały ze zbioru {5, 10, … , 50} przez generator o jednostajnym rozkładzie prawdopodobieństwa.

Obliczenia przeprowadzono dla zadanej liczby zadań n = 40, 80, 120, 160, 200, liczby maszyn m = 5, 10, 15, 20, 25 oraz dla ograniczonej liczby zasobów N = 10.000. Dla każdej kombinacji n i m wygenerowano 40 instancji. Następnie dokonano analizy porównawczej zaproponowanego w pracy algorytmu heurystycznego ze znanym z literatury algorytmem *LPT* (Longest Processing Time). Rezultaty tej analizy zostały przedstawione w Tabeli 1.

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

n/m	Liczba instancji, dla których:			Δ^H	S^H	S^{LPT}
	$T_{zak}^H < T_{zak}^{LPT}$	$T_{zak}^H = T_{zak}^{LPT}$	$T_{zak}^H > T_{zak}^{LPT}$	%	sek	sek
40/5	20	1	19	2,4	2,7	2,2
40/10	22	0	18	2,7	2,9	2,5
40/15	21	2	17	3,4	4,2	3,8

40/20	23	2	15	3,7	4,9	4,3
40/25	24	3	13	3,9	6,3	5,9
80/5	21	1	18	2,0	2,9	2,3
80/10	22	0	18	2,7	3,8	3,4
80/15	23	0	17	3,8	5,1	4,2
80/20	24	2	14	4,9	6,7	5,8
80/25	25	3	12	5,3	8,4	6,7
120/5	22	1	17	2,9	4,1	3,7
120/10	23	1	16	3,1	6,8	5,9
120/15	24	1	15	3,8	7,9	6,8
120/20	25	2	13	4,9	8,9	7,6
120/25	26	1	13	5,7	9,9	8,6
160/5	22	0	18	2,9	5,9	5,2
160/10	23	1	16	3,2	6,7	6,2
160/15	23	2	15	3,8	9,2	8,4
160/20	25	1	14	4,7	10,2	9,3
160/25	26	1	13	6,3	12,8	10,9
200/5	22	1	17	2,9	6,3	5,7
200/10	23	2	15	3,3	6,9	6,2
200/15	25	2	13	3,9	9,1	8,2
200/20	26	2	12	5,2	10,5	9,6
200/25	27	3	10	6,9	12,6	11,4

W Tab.1 występują następujące wielkości:

n – liczba zadań,

m – liczba maszyn,

T_{zak}^{H} – czas zakończenia wykonywania wszystkich zadań ze zbioru Z przy wykorzystaniu algorytmu heurystycznego,

T_{zak}^{LPT} – czas zakończenia wykonywania wszystkich zadań ze zbioru Z przy wykorzystaniu algorytmu *LPT*,

Δ^H – średnia procentowa poprawa czasu T_{zak}^H w stosunku do T_{zak}^{LPT}:

$$\Delta^H = \frac{T_{zak}^{LPT} - T_{zak}^H}{T_{zak}^H} \cdot 100\,\%,$$

S^H – średni czas obliczeń dla algorytmu heurystycznego,
S^{LPT} – średni czas obliczeń dla algorytmu *LPT*.

6. Uwagi końcowe

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

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

Literatura

1. Błażewicz J., Dell'Olmo P., Drozdowski M., Speranza M. G. : Scheduling multiprocessor tasks on three dedicated processors. Information Processing Letters 41, 1992, pp. 275-280.
2. Janiak A.: Single machine scheduling problem with a common deadline and resource dependent release dates. European Journal of Operational Research, Vol. 53, 1991, pp. 317-325.
3. Janiak A., Kovalyov M.: Single machine scheduling subject to deadlines and resources dependent processing times. European Journal of Operational Research, 1996, Vol. 94, pp. 284-291.
4. Nowicki E., Smutnicki C.: The flow shop with parallel machines. A Tabu search approach. European Journal of Operational Research 106, 1998, pp. 226-253.
5. 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, pp. 26-29.
6. Józefowska J., Węglarz J.: On a methodology for discrete-continuous scheduling. European Journal of Operational Research, Vol. 107, 1998, pp. 338-353.
7. 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, s. 221-229.

8. 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, s. 329-334.
9. 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, pp. 349-361.
10. 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, pp. 20-26.
11. Buchalski Z.: An Heuristic Algorithm for Solving the Scheduling Problem in Multiprocessing Computer System. Polish Journal of Environmental Studies, Vol. 16, No. 4A, 2007, pp. 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