EFEKTYWNE WYZNACZANIE OPTYMALNEGO ROZDZIAŁU ZADAŃ W SYSTEMIE MASZYN RÓWNOLEGŁYCH

Zbigniew BUCHALSKI

Streszczenie: Celem artykułu jest prezentacja rezultatów badań problemu czasowo-optymalnego szeregowania zadań i rozdziału ograniczonych zasobów w systemie maszyn równoległych. System ten posiada *m* równoległych maszyn, na których należy wykonać *n* zadań. Zakładamy, że wszystkie zadania są niezależne i liczba zadań jest większa od liczby maszyn. Zakładamy ponadto, że występuje stałość przydziału zasobów do maszyn podczas wykonywania całego zbioru zadań. Dla zadanej funkcji czasu realizacji zadań sformułowano model matematyczny zagadnienia. Ponieważ zagadnienie to należy do klasy problemów *NP*-zupełnych zaproponowano pewien algorytm heurystyczny dla rozwiązania postawionego zagadnienia. Zaprezentowano wyniki badań numerycznych wykonanych na bazie podanego w pracy algorytmu heurystycznego.

Słowa kluczowe: systemy maszyn równoległych, szeregowanie zadań, 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. Badania 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, 21].

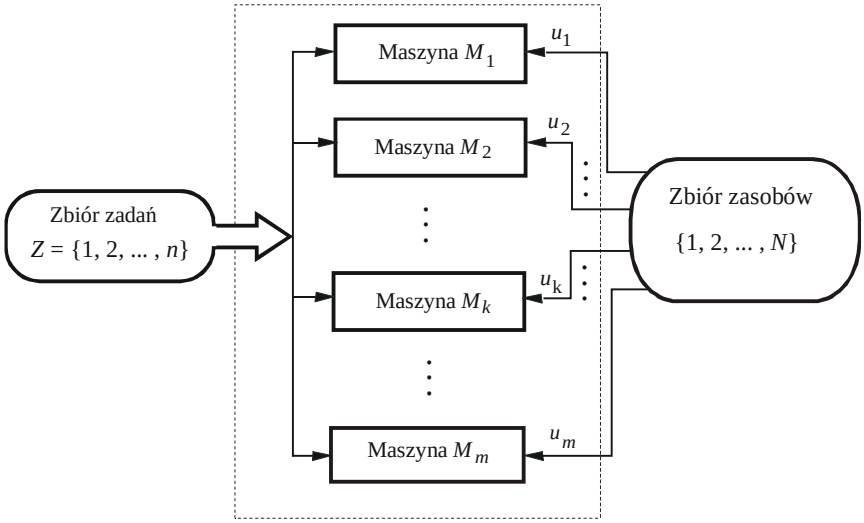
Celem niniejszej pracy jest znalezienie takiego uszeregowania *n* niezależnych niepodzielnych zadań na *m* maszynach pracujących równolegle oraz takiego przydziału zasobu podzielnego w sposób ciągły do tych maszyn, aby minimalizować czas T_{zak} zakończenia wykonywania wszystkich zadań. Sposób wykonania zadań określa niezbędna

do tego ilość zasobu. Tematyka ta poruszana już była we wcześniejszych pracach autora [22, 23, 24, 25, 26].

W niniejszym artykule zaprezentowano pewien algorytm heurystyczny wyznaczający czasowo-optymalne szeregowanie n zadań niezależnych niepodzielnych i ograniczonej liczby jednostek zasobu nieodnawialnego podzielnego w sposób ciągły do m maszyn pracujących równolegle. Przedstawiono wyniki badań numerycznych przeprowadzonych na tym algorytmie dla losowo generowanych danych.

2. Sformułowanie problemu. Model matematyczny zagadnienia

Rozpatrzmy dyskretny system produkcyjny zawierający maszyny połączone równolegle przedstawiono na rys. 1.



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 zadanie 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:

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

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

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

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}.

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

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

Ograniczenia nałożone na rozwiązanie tego problemu są następujące:

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

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

(iii) $u_1, u_2, ..., u_m$ - 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:

$$T_{zak} = \min_{\substack{Z_1, Z_2, ..., Z_m \\ u_1, u_2, ..., u_m}} \max_{1 \le k \le m} \left\{ \sum_{i \in Z_k} T_i'(u_k, k) \right\} \tag{4}$$

przy następujących ograniczeniach:

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

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

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:

$$T_i'(u_k, k) = a_{ik} + \frac{b_{ik}}{u_k}, \quad u_k \in [0, N], \quad 1 \le k \le m, \quad 1 \le i \le n. \tag{5}$$

Do rozwiązania postawionego problemu pomocny będzie następujący lemat:

LEMAT 1

Jeżeli $u_k^*, Z_k^*, k = 1,2,..., m$ są rozwiązaniami zadania (4), to:

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

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

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

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 (4). Wartość tej funkcji jest rozwiązaniem następującego układu równań:

$$\begin{cases} \sum_{i \in Z_k} a_{ik} + \dfrac{\sum_{i \in Z_k} b_{ik}}{u_k} = F(Z_1, Z_2, \dots, Z_m); \quad k : Z_k \ne \varnothing, \quad k = 1, 2, \dots, m \\ \sum_{k=1}^{m} u_k = N; \quad u_k > 0, \quad k : Z_k \ne \varnothing, \quad k = 1, 2, \dots, m. \end{cases} \tag{6}$$

Wykorzystując **LEMAT 1** oraz (6) zadanie minimalizacji (4) można przedstawić w następującej postaci:

$$T_{zak} = \min_{Z_1, Z_2, ..., Z_m} F(Z_1, Z_2, ..., Z_m), \tag{7}$$

przy następujących ograniczeniach:

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

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

Jeżeli $Z_1^*, Z_2^*, .., Z_m^*$ jest rozwiązaniem zadania (7), to $u_k^*, Z_k^*, k = 1,2,..., m,$ gdzie

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

jest rozwiązaniem zadania (4).

3. 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, …, m. Po tym uszeregowaniu następuje zróżnicowanie maszyn pod względem liczby przydzielanych im zasobów i sprawdzenie czy skrócony został czas zakończenia wykonywania wszystkich zadań T_{zak}.

Kolejne kroki algorytmu heurystycznego są następujące:

Krok 1. Oblicz czasy wykonywania zadań na poszczególnych maszynach

$T_i(u_k, k) = a_{ik} + \frac{b_{ik}}{u_k}$, i = 1, 2,…, n, k = 1, 2, … , m dla zadanej wartości $u_k = \frac{N}{m}$ i losowo generowanych parametrów a_{ik}, b_{ik}..

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:

$$T_{śr} = \frac{\sum_{i=1}^{n} T_i(u_k, k)}{m}; \qquad i \in Z, k \in M, u_k = \frac{N}{m}. \tag{10}$$

Krok 4. Uszereguj kolejne zadania z listy L na pierwszej wolnej maszynie aż do momentu, gdy suma czasów wykonywania zadań uszeregowanych na tej maszynie nie przekroczy czasu $T_{śr}$ i usuń te zadania z listy L.

Krok 5. Do maszyny, na której ostatnio było wykonywane szeregowanie zadań, przydziel zadanie o najdłuższym z czasów tworzących listę L, ale takie, aby łączny czas wykonywania wszystkich zadań uszeregowanych na tej maszynie nie przekroczył czasu $T_{śr}$ i usuń ostatnio przydzielone zadanie z listy L.

Krok 6. Wróć do **Kroku 5** i powtarzaj ten krok aż do momentu, gdy na liście L nie znajdzie się żadne zadanie, które spełniałoby warunek postawiony w **Kroku 5.** Jeżeli są jeszcze maszyny, na których nie uszeregowano żadnych zadań, to wróć do **Kroku 4.** W przeciwnym wypadku przejdź do **Kroku 7.**

Krok 7. Pozostałe na liście L zadania uszereguj kolejno na maszynach wg algorytmu *LPT* (Longest Processing Time) i usuń te zadania z listy L. Szeregowanie to wykonuj aż do momentu wyczerpania się listy L.

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

Krok 10. Dla uszeregowania zadań na maszynach utworzonego w **Krokach 4÷7** i dla liczby zasobów u_k przydzielonych maszynom w **Kroku 9** oblicz czas zakończenia wykonywania wszystkich zadań T_{zak}.

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

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

Krok 13. Wyznacz dyskretne ilości zasobów $\hat{u}_k$, $k = 1, 2, \ldots, m$ według zależności:

$$\hat{u}_{\alpha(k)} = \begin{cases} \lfloor u_{\alpha(k)} \rfloor + 1; & k = 1, 2, \ldots, \Delta, \\ \lfloor u_{\alpha(k)} \rfloor \quad ; & k = \Delta + 1,\ \Delta + 2, \ldots, m, \end{cases} \tag{11}$$

gdzie $\Delta = N - \sum_{j=1}^{m} \lfloor u_j \rfloor$ oraz α jest permutacją elementów zbioru $M=\{1,2,\ldots, m\}$ 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 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.

4. Wyniki badań numerycznych

Przeprowadzono badania numeryczne na bazie przedstawionego algorytmu dla ośmiu zwiększających się kolejno wartości współczynnika podziału zasobów β z przedziału [1.5, 3.0, 4.5, ... , 12.0]. Parametry charakteryzujące *i*-te zadanie i *k*-tą maszynę a_{ik},, b_{ik} wylosowane zostały ze zbioru {3.0, 6.0, ... , 90.0} przez generator o jednostajnym rozkładzie prawdopodobieństwa. Dla każdej kombinacji *n* i *m* wygenerowano 30 instancji. Rezultaty analizy porównawczej algorytmu heurystycznego skonstruowanego dla potrzeb niniejszej pracy i znanego z literatury algorytm *LPT* przedstawione zostały w tab.1.
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*.

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
25/4	15	1	14	2,4	2,6	2,1
25/8	16	1	13	2,8	2,8	2,3
25/12	15	2	13	3,5	4,3	3,6
25/16	16	2	12	3,7	4,8	4,3
25/20	17	1	12	3,9	5,9	5,4
50/4	15	0	15	1,9	2,8	2,4
50/8	16	2	12	2,8	3,9	3,4
50/12	17	1	12	3,9	4,8	3,7
50/16	17	2	11	4,9	6,7	5,4
50/20	18	1	11	5,3	7,9	5,7
75/4	14	2	14	2,9	3,8	3,4
75/8	16	0	14	3,1	6,5	5,8
75/12	17	2	11	3,8	7,4	6,4
75/16	18	1	11	4,9	8,8	7,6
75/20	19	0	11	5,4	9,8	8,5
100/4	15	2	13	2,8	5,7	5,1
100/8	16	1	13	3,5	6,8	6,1
100/12	18	2	10	3,9	8,8	7,6
100/16	19	2	9	5,3	9,7	8,8
100/20	21	1	8	7,4	11,8	10,7

5. Podsumowanie

Przedstawione w poprzednim rozdziale 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 znane-go 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. 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, pp. 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.: 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.
7. Brah S.A., Loo 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., Hariri A., Potts C.: 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, pp. 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, pp. 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, pp. 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, s. 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, s. 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, pp. 226-253.
20. Santos D.L., J.L., Deal D.E.: An evaluation of sequencing heuristics in flow shops with multiple processors, Computers and Industrial Engineering, Vol. 30, No. 4, 1996, 681-691.
21. Węglarz J.: Multiprocessor scheduling with memory allocation - a deterministic approach. IEEE Trans. Comput., C-29, 1980, 703-710.
22. Buchalski Z.: Szeregowanie zadań na różnych maszynach równoległych z rozdziałem ograniczonych zasobów. Zeszyty Naukowe Politechniki Śląskiej, Automatyka, No. 1389, Gliwice, 1998, 77-84.
23. 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 PWr., Wrocław, 2005, pp. 20-26.
24. 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.
25. 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.
26. 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.: (71) 320 32 92
e-mail: zbigniew.buchalski@pwr.wroc.pl