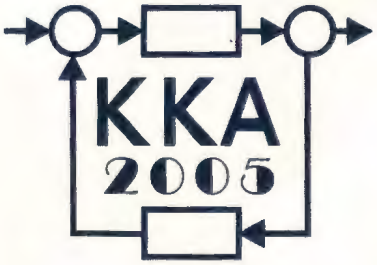
XV Krajowa Konferencja Automatyki

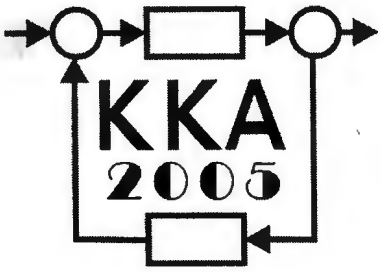
Tom II



Redaktorzy:
Zdzisław Bubnicki
Roman Kulikowski
Janusz Kacprzyk

XV Krajowa Konferencja Automatyki

Tom II



Redaktorzy:
Zdzisław BUBNICKI
Roman KULIKOWSKI
Janusz KACPRZYK

ORGANIZATOR
Komitet Automatyki i Robotyki Polskiej Akademii Nauk
Instytut Badań Systemowych Polskiej Akademii Nauk
WSPÓŁORGANIZATORZY
Politechnika Warszawska
Przemysłowy Instytut Automatyki i Pomiarów
Polskie Stowarzyszenie Pomiarów, Automatyki i Robotyki

ORGANIZATOR

Komitet Automatyki i Robotyki Polskiej Akademii Nauk
Instytut Badań Systemowych Polskiej Akademii Nauk

WSPÓŁORGANIZATORZY

Politechnika Warszawska
Przemysłowy Instytut Automatyki i Pomiarów
Polskie Stowarzyszenie Pomiarów, Automatyki i Robotyki

KOMITET PROGRAMOWY

Przewodniczący	Zdzisław BUBNICKI
Zastępca Przewodniczącego	Roman KULIKOWSKI

CZŁONKOWIE

Stanisław BAŃKA	Michał BIAŁKO
Mikołaj BUSŁOWICZ	Władysław FINDEISEN
Ryszard GESSING	Henryk GÓRECKI
Jakub GUTENBAUM	Jerzy JÓZEFCZYK
Stanisław KACZANOWSKI	Tadeusz KACZOREK
Janusz KACPRZYK	Jerzy KLAMKA
Józef KORBICZ	Zbigniew KOWALSKI
Krzysztof KOZŁOWSKI	Juliusz L. KULIKOWSKI
Krzysztof KUŹMIŃSKI	Kazimierz MALANOWSKI
Krzysztof MALINOWSKI	Wojciech MITKOWSKI
Antoni NIEDERLIŃSKI	Władysław PEŁCZEWSKI
Tadeusz PUCHAŁKA	Leszek RUTKOWSKI
Stanisław SKOCZOWSKI	Roman SŁOWIŃSKI
Jerzy ŚWIĄTEK	Andrzej ŚWIERNIAK
Ryszard TADEUSIEWICZ	Piotr TATJEWSKI
Krzysztof TCHOŃ	Leszek TRYBUS
Jan WĘGLARZ	Andrzej P. WIERZBICKI

KOMITET ORGANIZACYJNY

Przewodniczący	Roman KULIKOWSKI
Zastępcy Przewodniczącego	Janusz KACPRZYK
	Stanisław KACZANOWSKI
	Tadeusz KACZOREK
	Krzysztof MALINOWSKI
Członkowie	Roman OSTROWSKI
	Tadeusz PUCHAŁKA
	Dariusz WAGNER
Sekretarze naukowi	Jan STUDZIŃSKI
	Jan W. OWSIŃSKI

ISBN 83-89475-01-4

Druk: ARGRAF, Warszawa

STEROWANIE KOMPLEKSAMI OPERACJI

ALGORYTM HARMONOGRAMOWANIA ZADAŃ PODZIELNYCH NA MASZYNACH RÓWNOLEGŁYCH PRZY UWZGLĘDNIENIU PRZEZBROJEŃ I OGRANICZEŃ ZASOBOWYCH †

Tomasz ŚLIWIŃSKI*, Eugeniusz TOCZYŁOWSKI*

*Politechnika Warszawska, Instytut Automatyki i Informatyki Stosowanej,
ul. Nowowiejska 15/19, 00-665 Warszawa
e-mail:T.Sliwinski@elka.pw.edu.pl, E.Toczylowski@ia.pw.edu.pl

Streszczenie: W pracy jest rozważany dwufazowy algorytm harmonogramowania zadań podzielnych na maszynach równoległych przy ograniczeniach zasobowych i przezbrojeniach. W pierwszej fazie jest rozwiązywane zadanie planowania nadrzędnego za pomocą techniki generacji kolumn. Dla fazy drugiej badane są trzy warianty szeregowania planów za pomocą algorytmu genetycznego z uwzględnieniem uproszczonych kryteriów obliczania długości harmonogramu szczegółowego.

Słowa kluczowe: procesory równoległe, zadania podzielne, przezbrojenia, ograniczenia zasobowe

1. OPIS PROBLEMU

Zagadnienie harmonogramowania zadań podzielnych na maszynach równoległych przy ograniczeniach zasobowych i przezbrojeniach występuje w licznych gałęziach przemysłu, np. w przemyśle chemicznym, farmaceutycznym czy spożywczym. Równolegle pracujące maszyny lub linie produkcyjne tworzą gniazdo produkcyjne, w którym jest możliwe równoległe prowadzenie jednoczesnej produkcji wieloasortymentowej poprzez przydzielenie różnych zadań do maszyn tworzących gniazdo. Zbiór zadań $\mathcal{K}$ ma zostać obsłużony przez zespół równolegle pracujących, niejednorodnych maszyn $\mathcal{L}$. Zakłada się, że zadania mogą być podzielne w czasie i przestrzeni, tzn. każde zadanie może być rozdzielone i realizowane na wielu maszynach, może zostać w dowolnym momencie przerwane a następnie kontynuowane w innym momencie na tej samej lub innych maszynach.

Każda maszyna może obsługiwać nie więcej niż jedno zadanie, przy czym każda z maszyn może charakteryzować się inną prędkością przetwarzania zadań oraz różnym zużyciem zasobów wspólnych. Swoboda produkcji jest ograniczona przez dodatkowe ograniczenia zasobowe. W pracy rozważa się zagadnienie z ograniczonym zasobem odnawialnym współdzielonym pomiędzy wszystkie maszyny. Zasobem takim mogą być pracownicy lub wymagane narzędzia składowane we wspólnym dla wszystkich maszyn magazynie o ograniczonej pojemności. Ilość potrzebnego zasobu zależy od zadania i maszyny, na której jest wykonywane. Zmiana obsługiwanego przez maszynę zadania powoduje konieczność przezbrojenia, przy czym jego czas zależy zarówno od zadania poprzedniego i następnego, jak i od maszyny, na której jest wykonywane. Celem jest znalezienie harmonogramu czasooptymalnego.

2. PODEJŚCIE DWUFAZOWE

Złożoność problemu wynika z konieczności jednoczesnego uwzględnienia ograniczeń zasobowych i przezbrojeń. Już problem szeregowania zadań na pojedynczej maszynie z uwzględnieniem przezbrojeń, który odpowiada zadaniu komiwojażera, jest NP-trudny. Zwiększenie liczby maszyn oraz dodanie ograniczeń zasobowych jeszcze bardziej zwiększa jego złożoność. Pewne heurystyki dla tych problemów były prezentowane w pracach [1, 4, 5, 2, 3].

Nowoczesne systemy produkcyjne charakteryzują się stosunkowo krótkimi czasami przezbrojeń, które zazwyczaj zajmują nie więcej niż kilka procent całego czasu produkcji. Uzasadnia to zastosowanie podejścia dwufazowego, w którym w modelu planowania nadrzędnego można zaniedbać przezbrojenia i uwzględnić je dopiero podczas tworzenia harmonogramu szczegółowego. Niniejsza praca rozwija dwufazowe podejście strukturalne z prac [6, 7] polegające na rozwiązywaniu nadrzędnych zadań planowania za pomocą techniki generacji kolumn a następnie wyznaczaniu harmonogramu szczegółowego.

Planowanie nadrzędne. W zastosowanym podejściu plan nadrzędny jest przedstawiany jako asynchroniczny, wieloetapowy proces decyzyjny, w którym horyzont decyzyjny jest podzielony na wiele elementarnych etapów. Każdy z nich odpowiada pod-

†Praca finansowana z grantu KBN 3T11C 03028.

okresowi o zróżnicowanej długości, w którym stan procesu, czyli rodzaj i sposób wykonywania zadań oraz alokacja zasobów, jest stały. Pojedynczy etap jest nazywany *planem elementarnym*. Struktura problemu planowania nadrzędnego pozwala na jego dekompozycję na podproblemy odpowiadające planom elementarnym oraz umożliwia zastosowanie techniki generacji kolumn, co znacząco upraszcza i przyspiesza proces optymalizacji. Pomijany jest natomiast problem przejść pomiędzy kolejnymi planami. Uproszczenie polega na pominięciu problemu przezbrojeń i skoncentrowaniu się na ograniczeniach technicznych i zasobowych.

Szeregowanie i harmonogramowanie. Po wyznaczeniu planu nadrzędnego tworzony jest harmonogram szczegółowy uwzględniający zarówno przezbrojenia jak i wszystkie ograniczenia techniczne oraz zasobowe. Występują tu dwa zagadnienia: szeregowanie planów elementarnych oraz wyznaczenie harmonogramu szczegółowego. Obydwa zagadnienia są ze sobą ściśle związane, jednak połączenie szeregowania i harmonogramowania szczegółowego w jednym algorytmie znacząco zwiększa złożoność obliczeniową problemu. W niniejszej pracy zastosowane zostało podejście kompromisowe, w którym rozdzielono szeregowanie od harmonogramowania szczegółowego przez wprowadzenie uproszczonych miar jakości jako funkcji celu algorytmu szeregującego.

3. PLANOWANIE NADRZĘDNE

W problemie planowania dokonywany jest podział całego horyzontu planowania na zbiór podokresów z określonym przydziałem zadań do maszyn (planów elementarnych), gwarantującym nienaruszalność chwilowych ograniczeń zasobowych. Podstawowe indeksy i parametry modelu:

- k zadanie, $k \in \mathcal{K}$;
- l maszyna, $l \in \mathcal{L}$;
- β indeks planu elementarnego, $\beta \in \mathcal{B}$;
- p_{lk} czas obsługi *całego* zadania k na procesorze l;
- α_{lk} ilość odnawialnego zasobu rezerwowanego na czas obsługi zadania k na maszynie l;
- W całkowita dostępna ilość zasobu odnawialnego.

Wprowadzamy następujące zmienne decyzyjne:

- y_β czas trwania planu elementarnego β;
- v_{lk}^β zmienna binarna; równa 1 wtedy i tylko wtedy, gdy w planie elementarnym β zadanie k jest realizowane na maszynie l.

Zbiór odpowiednich planów elementarnych minimalizujący długość harmonogramu bez przezbrojeń może zostać wyznaczony przez rozwiązanie następującego zadania optymalizacji

Zminimalizować

$$\sum_{\beta \in B} y_\beta \tag{1}$$

przy ograniczeniach

$$\sum_{\beta \in B} \Big(\sum_{l \in L} \frac{1}{p_{lk}} v_{lk}^\beta \Big) y_\beta = 1, \quad k \in K \tag{2}$$

$$\sum_{k \in K} v_{lk}^\beta \leq 1, \quad l \in L, \beta \in \mathcal{B} \tag{3}$$

$$\sum_{l \in L} \sum_{k \in K} \alpha_{lk} v_{lk}^\beta \leq W, \quad \beta \in \mathcal{B} \tag{4}$$

$$v_{lk}^\beta \in \{0,1\}, \quad l \in L, k \in K, \beta \in \mathcal{B} \tag{5}$$

$$y_\beta \geq 0 \quad \beta \in \mathcal{B} \tag{6}$$

Funkcja celu (1) definiuje całkowitą długość harmonogramu. Równość (2) gwarantuje, że każde zadanie zostanie całkowicie wykonane. Ograniczenia (3) i (4) dotyczą pojedynczego planu elementarnego. Pierwsze zabezpiecza przed sytuacją, w której na pojedynczej maszynie jest jednocześnie przetwarzane więcej niż jedno zadanie. Drugie zapobiega przekroczeniu dostępnej ilości zasobu odnawialnego w trakcie realizacji planów elementarnych.

Problem (1)- (6) jest trudnym zadaniem programowania kwadratowego, w którym liczba zmiennych rośnie wykładniczo z jego rozmiarem. Struktura zadania pozwala jednak na jego efektywnie rozwiązanie, przez wykorzystanie dekompozycji Danziga-Wolfe'a i opartej na niej techniki generacji kolumn. Dekompozycja problemu na zadanie *nadrzędne* i *podrzędne* jest możliwa dzięki istnieniu dwóch grup ograniczeń. Pierwsza grupa (2), wiąże wszystkie zmienne y_β występujące w funkcji celu. Druga grupa (3) (4), to ograniczenia dotyczące wyłącznie jednej kolumny macierzy ograniczeń (2) a przez to jednej zmiennej y_β.

Zadanie nadrzędne. Zadanie nadrzędne techniki generacji kolumn jest oparte na funkcji celu (1) oraz na ograniczeniach wspólnych (2). Rozważamy następujący model zadania nadrzędnego:

$$\min \sum_{\beta \in B} y_\beta \tag{7}$$

$$\text{p.o.} \sum_{\beta \in B} \Big(\sum_{l \in L} \frac{1}{p_{lk}} v_{lk}^\beta \Big) y_\beta = 1, \quad k \in K \tag{8}$$

$$y_\beta \geq 0, \quad \beta \in B \tag{9}$$

Podobnie jak w zadaniu wyjściowym funkcją celu jest całkowita długość harmonogramu, a ograniczenia zapewniają pełne wykonanie wszystkich zadań należących do danego zlecenia produkcyjnego. W przeciwieństwie jednak do zadania wyjściowego, v_{lk}^β nie są zmiennymi lecz stałymi parametrami liniowych ograniczeń (8). Oczywiście macierz ograniczeń musi zawierać kolumny odpowiadające wszystkim dopuszczalnym kombinacjom tych parametrów. Otrzymane zadanie jest zadaniem programowania liniowego z dużą liczbą zmiennych, co sprawia, że przechowywanie macierzy ograniczeń w całości jest niemożliwe. Rozwiązaniem jest zastosowanie techniki generacji kolumn. W metodzie tej w każdym kroku algorytmu sympleksowego nowe kolumny wprowadzane do

bazy są generowane przez rozwiązanie pewnego pomocniczego zadania optymalizacji, tzw. zadania podrzędnego.

Zadanie podrzędne. Nowa kolumna wprowadzana do macierzy bazowej powinna charakteryzować się jak największą wartością odpowiadającej jej ceny zredukowanej (dla zadania minimalizacji). Dla rozpatrywanego problemu (7)–(9) cena zredukowana wyraża się następującym wzorem:

$$y_{0\beta} = \sum_{k\in K}\Big(\sum_{l\in L}\frac{1}{p_{lk}}v_{lk}^{\beta}\Big)\pi_k - 1 = \sum_{k\in K}\sum_{l\in L}\Big(\frac{\pi_k}{p_{lk}}\Big)v_{lk}^{\beta} - 1,$$

gdzie π jest wektorem zmiennych dualnych bieżącego rozwiązania. Zmienne te odpowiadają ograniczeniom (8) zadania nadrzędnego. Każda kolumna generowana w zadaniu podrzędnym określa przydział zadań do procesorów, a tym samym definiuje jeden plan elementarny. Dopuszczalność przydziału jest zapewniona przez dodatkowe ograniczenia zadania podrzędnego:

$$\sum_{k\in K} v_{lk}^{\beta} \le 1, \quad l \in L \tag{10}$$

$$\sum_{l\in L}\sum_{k\in K} \alpha_{lk} v_{lk}^{\beta} \le W \tag{11}$$

$$v_{lk}^{\beta} \in \{0,1\}, \quad l \in L, k \in K \tag{12}$$

Ograniczenia te odpowiadają bezpośrednio ograniczeniom (3)–(5) z zadania wyjściowego. W zależności od potrzeb, zadanie podrzędne może zostać rozszerzone o dowolne warunki nakładane na pojedynczy plan elementarny, np. warunek na maksymalną liczbę maszyn ($L_{\max}$), jakie mogą jednocześnie obsługiwać dane zadanie: $\sum_{l\in L} v_{lk}^{\beta} \le L_{\max}$, $k \in K$.

Przykład. Rozważany jest system produkcyjny składający się z 3 różnych równoległych linii produkcyjnych, na których wytwarzanych jest 10 typów części. W systemie tym pojedyncze zadanie jest podzielne i polega na wyprodukowaniu pewnej liczby części każdego typu. Znane są czasy wykonania poszczególnych zadań na każdej linii produkcyjnej

$$[p_{lk}] = \begin{bmatrix} 11 & 20 & 19 & 17 & 19 & 18 & 22 & 29 & 16 & 27 \\ 29 & 27 & 22 & 26 & 21 & 10 & 13 & 10 & 23 & 12 \\ 10 & 13 & 19 & 25 & 16 & 27 & 21 & 19 & 19 & 24 \end{bmatrix}$$

Podczas wykonywania zadań wymagana jest pewna ilość dzielonego zasobu odnawialnego, różna dla poszczególnych zadań

$$[\alpha_{lk}] = \begin{bmatrix} 8 & 10 & 6 & 6 & 8 & 6 & 7 & 8 & 8 & 6 \\ 7 & 7 & 6 & 9 & 6 & 9 & 8 & 9 & 6 & 6 \\ 8 & 7 & 6 & 6 & 10 & 6 & 8 & 8 & 10 & 7 \end{bmatrix}$$

Przyjmujemy $W = 18$. Zadanie może być jednocześnie wykonywane tylko na jednej linii produkcyjnej. Ponieważ problem przezbrojeń został pominięty, wynikiem planowania jest zbiór planów elementarnych oraz ich czasy wykonania (Tabela 1). Wyznaczony plan produkcji definiuje przydział zadań do linii produkcyjnych w każdej chwili harmonogramu. Kolejność planów elementarnych nie jest ustalona.

4. HARMONOGRAMOWANIE

Po wyznaczeniu planów elementarnych dokonywane jest ich szeregowanie oraz jest wyznaczany harmonogram szczegółowy. W niniejszej pracy rozdzielono szeregowanie od harmonogramowania szczegółowego przez wykorzystywanie uproszczonej miary jakości jako funkcji celu algorytmu szeregującego. W idealnym przypadku uszeregowanie optymalne w sensie stosowanej miary jakości powinno dawać optymalny harmonogram szczegółowy. Niestety nie ma prostej recepty na skonstruowanie takiej miary jakości, która jednocześnie byłaby efektywna obliczeniowo. Dlatego zastosujemy przybliżone miary jakości pozwalających generować zadowalające uszeregowania planów elementarnych. W tym punkcie zakładamy, że mamy ustalone uszeregowanie planów elementarnych.

Miara jakości (długość harmonogramu) może być wyznaczona jedną z trzech metod różniących nakładem obliczeń: (i) minimalizacja całkowitego czasu przezbrojeń, (ii) metoda ścieżki krytycznej [8], oraz (iii) metoda ścieżki krytycznej z realokacją zadań [9]. Miary (ii) i (iii) pozwalają na wyznaczenie stosunkowo dokładnych harmonogramów szczegółowych.

Całkowity czas przezbrojeń

Niech *operacją* będzie wykonanie części zadania na danej maszynie w określonym planie elementarnym. Rozważmy system z jedną maszyną produkcyjną. W klasycznym podejściu skrócenie harmonogramu uzyskuje się przez takie uszeregowanie operacji, aby suma czasów wszystkich przezbrojeń pomiędzy kolejnymi operacjami była jak najmniejsza. Wynika to z faktu, że czasy operacji, wstępnie ustalone na etapie planowania produkcji, nie są zmieniane w trakcie harmonogramowania, a o długości harmonogramu decydują wyłącznie przezbrojenia. Zastosowanie tej miary jakości dla problemu z jedną maszyną pozwala więc na wyznaczanie optymalnego harmonogramu szczegółowego.

Rozważmy teraz problem z maszynami równoległymi. W ogólnym przypadku najkrótszy całkowity czas przezbrojeń nie gwarantuje uzyskania optymalnego harmonogramu szczegółowego. Przyczyną jest sztywna synchronizacja harmonogramów kolejnych planów elementarnych na różnych maszynach, wynikająca z konieczności zapewnienia dostępu do wspólnych zasobów w każdym momencie produkcji. Jednak przeprowadzone eksperymenty pokazały, że całkowity czas przezbrojeń można traktować jako bardzo dobrą przybliżoną miarę jakości ostatecznego harmonogramu. Ważną jej zaletą jest bardzo mała złożoność obliczeniowa.

Metoda ścieżki krytycznej

W podejściu opracowanym przez Hindiego i Toczyłowskiego w [8] dla danego uszeregowania planów elementarnych harmonogram szczegółowy uwzględniający wszystkie operacje i przezbrojenia może zo-

linie produkcyjne	plany elementarne β									
	1	2	3	4	5	6	7	8	9	10
1	-	4	-	-	-	4	-	6	9	9
2	7	10	6	8	8	5	7	5	7	-
3	5	3	2	1	2	3	1	3	-	5
czas trwania y_β	1.78	12.00	8.89	5.89	4.11	5.00	4.11	2.00	7.11	8.89

Tabela 1. Plan nadrzędny wyznaczony dla opisywanego przykładu – przydział zadań do linii produkcyjnych w kolejnych planach elementarnych oraz czasy trwania planów elementarnych.

stać wyznaczony metodą ścieżki krytycznej. Pomysł polega na zbudowaniu sieci działań uwzględniającej zależności czasowe i kolejnościowe operacji oraz konieczność występowania i czasy trwania przezbrojeń. Sieć działań jest tworzona sposób uniemożliwiający nakładanie się na siebie sąsiednich planów elementarnych. Zapewnia się to poprzez określenie relacji poprzedzania dla poszczególnych operacji zgodnie z kolejnością planów elementarnych, do których przynależą. Najprostszą metodą wyznaczenia relacji poprzedzania jest określenie zbioru następników danej operacji, jako wszystkich operacji które są wykonywane po niej na każdej z maszyn.

Na rys. 1 przedstawiono przykładowe relacje poprzedzania operacji dla danego uszeregowania planów elementarnych. Po przypisaniu czasów wykonywania operacji do wierzchołków grafu i czasów przezbrojeń do łuków poziomych otrzymuje się sieć działań, na której przy zastosowaniu metody ścieżki krytycznej możliwe jest wyznaczenie minimalnego czasu trwania całego harmonogramu oraz chwil rozpoczęcia poszczególnych operacji.

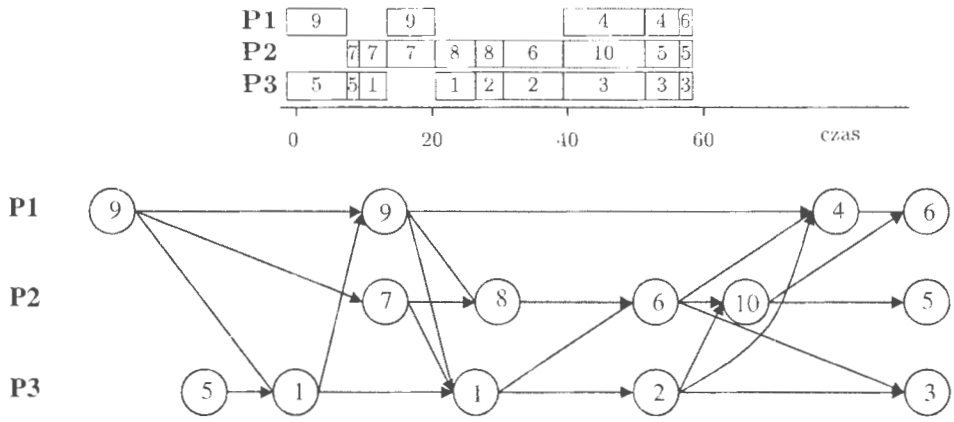
Metodę ścieżki krytycznej można wykorzystać jako przybliżoną miarę jakości, wykorzystywaną na etapie szeregowania, po którym następuje wyznaczenie ostatecznego harmonogramu szczegółowego. Metoda ścieżki krytycznej jest dokładniejsza od metody minimalizującej czas przezbrojeń a ponadto istnieją dla niej bardzo efektywne algorytmy obliczeniowe. Jednak harmonogramy wynikowe stworzone tą metodą mogą cechować się mniejszą od optymalnej zwartością (niepotrzebnymi przestojami maszyn). Dlatego w pracy Toczyłowskiego [9] wprowadzono istotne rozwinięcie tej metody, polegające na możliwości ingerowania w poszczególne plany elementarne. Podejście to, nazwane *metodą ścieżki krytycznej z realokacją zadań* umożliwia generowanie znacznie lepszych harmonogramów, kosztem pewnego zwiększenia złożoności obliczeniowej samego algorytmu.

Metoda ścieżki krytycznej z realokacją zadań

W metodzie ścieżki krytycznej z realokacją zadań możliwe są zmiany długości operacji przez przesuwanie małych porcji zadania pomiędzy planami elementarnymi, do których to zadanie zostało przydzielone (zobacz [9]). Pomysł polega na stworzeniu dla danej sieci działań zadania programowania liniowego, w którym zmiennymi są nie tylko czasy rozpoczęcia poszczególnych operacji ale także czasy ich trwania. Możliwe staje się więc zlikwidowanie niektórych przestojów maszyn przez wydłużenie jednych operacji i skrócenie innych. Otrzymany w wyniku rozwiązania takiego zadania harmonogram cechuje się znacznie lepszą zwartością w stosunku do harmonogramów wyznaczanych metodą ścieżki krytycznej. Operacje, których czas trwania został skrócony do zera, mogą zostać przy tym całkowicie usunięte z harmonogramu wraz ze związanymi z nimi przezbrojeniami. Niestety złożoność zadania, które uwzględniałoby usuwanie operacji i przezbrojeń, przewyższyłaby zalety opisywanego podejścia. Dlatego można zastosować algorytm przybliżony, w którym operacje o zerowej długości są usuwane w następnej iteracji. Tworzona jest wtedy nowa sieć działań nie zawierająca w ogóle operacji o zerowej długości i dla niej rozwiązywane jest kolejne zadanie programowania liniowego. Procedura może być powtarzana wielokrotnie, jeśli tylko prowadzi do poprawy aktualnego rozwiązania. Wprowadźmy następujące oznaczenia:

- O – zbiór wszystkich operacji w grafie poprzedzania; $n \in O$.
- O^k – zbiór wszystkich operacji składających się na zadanie k,
- l_n procesor na którym jest wykonywana operacja n,
- k_n zadanie realizowane podczas wykonywania operacji n,
- p_n – czas wykonywania całego zadania k_n procesorze l_n; $p_n = p_{l_n k_n}$
- x_n – zmienna ciągła; porcja zadania k_n wykonywana podczas trwania operacji n,
- s_n – zmienna ciągła; chwila rozpoczęcia operacji n.
- c_n – zmienna ciągła; chwila zakończenia operacji n,
- F^n zbiór następników n w grafie poprzedzania.
- f^n – operacja będąca bezpośrednim następnikiem operacji n na tym samym procesorze,
- B – zbiór operacji bez następników,
- c_{nm} – czas przezbrojenia między dwiema kolejnymi operacjami n i m na jednym procesorze.

Możemy teraz sformułować następujący model zadania programowania liniowego wyznaczania ścieżki



Rys. 1. Uszeregowanie planów elementarnych oraz odpowiadający mu graf poprzedzania.

krytycznej z realokacją porcji zadań.

$$\min T \tag{13}$$

$$e_n = s_n + x_n p_n, \quad n \in O \tag{14}$$

$$e_n \leq s_m, \quad n \in O,\ m \in F^n \tag{15}$$

$$e_n \leq c_{nf^n} + s_{f^n}, \quad n \notin B \tag{16}$$

$$e_n \leq T, \quad n \in B \tag{17}$$

$$0 \leq x_n \leq 1, \quad n \in O \tag{18}$$

$$\sum_{n \in O^k} x_n = 1, \quad k \in \mathcal{K} \tag{19}$$

W funkcji celu (13) minimalizowana jest długość harmonogramu. Ograniczenia (14) i (15) zapewniają prawidłowe relacje poprzedzania pomiędzy operacjami należącym do sąsiednich planów elementarnych. Nierówność (16) wymusza uwzględnienie w harmonogramie przezbrojeń pomiędzy kolejnymi operacjami na tym samym procesorze.

5. SZEREGOWANIE PLANÓW

Zagadnienie szeregowania planów elementarnych jest problemem permutacyjnym. Dla każdego uszeregowania można wyznaczyć wartość pewnej miary jego jakości, a znalezienie uszeregowania optymalnego jest równoważne znalezieniu minimalizującej jej permutacji. W pracy zdecydowano się na zastosowanie algorytmu genetycznego jako podstawowej metody służącej do szeregowania planów elementarnych. Główną wadą tego podejścia oraz innych metod ewolucyjnych jest konieczność wykonania dużej liczby obliczeń funkcji celu w procesie optymalizacji. Skutkuje to tym, że algorytmy genetyczne najlepiej nadają się do zastosowań, w których funkcja celu może zostać relatywnie łatwo wyznaczona. Zaletą algorytmów genetycznych jest natomiast to, że nie wykorzystują one bezpośrednio wiedzy o problemie w trakcie wyznaczania rozwiązań. Jest to szczególnie pożądana cecha, gdy wiedza o sposobie szukania rozwiązań jest niedostępna, tak jak to jest w przypadku rozpatrywanego zagadnienia. Tylko bowiem w sytuacji gdy minimalizowaną miarą jakości jest całkowity czas przezbrojeń, zagadnienie to z pewnym przybliżeniem można modelować przez problem komiwojażera. Gdy natomiast miarą jakości jest długość ścieżki krytycznej lub tym bardziej długość ostatecznego harmonogramu szczegółowego, złożoność zagadnienia zaczyna daleko wykraczać poza problem komiwojażera.

Algorytmy genetyczne są algorytmami przeszukiwania losowego symulującymi naturalną ewolucję organizmów. Operują one na zbiorze potencjalnych rozwiązań i iteracyjnie, wykonując serię transformacji, zmierzają do suboptymalnych lub optymalnych rozwiązań. Używane transformacje naśladują naturalne procesy biologiczne takie jak krzyżowania i mutacje. Podstawowy schemat algorytmu genetycznego tutaj zastosowany jest następujący. Najpierw generowany jest zbiór losowych rozwiązań dopuszczalnych (uszeregowań). Następnie rozwiązania te są oceniane przez zastosowanie przybliżonej bądź dokładnej miary jakości. Bazując na wyznaczonej wartości miary jakości, wybierana jest pewna część rozwiązań, które zostaną poddane mutacji i krzyżowaniu. Zmienione lub całkowicie nowe rozwiązania są następnie ponownie oceniane. Cały proces jest powtarzany wielokrotnie. W algorytmie zastosowano metody krzyżowania, mutacji oraz selekcji.

Przykład Kontynuujemy poprzedni przykład. W drugiej fazie opisywanego algorytmu jest wyznaczane optymalne uszeregowanie planów elementarnych. Rozważane są trzy różne funkcje celu dla algorytmu szeregowania: całkowity czas przezbrojeń, metoda ścieżki krytycznej oraz metoda ścieżki krytycznej z realokacją zadań. Ponieważ w każdym z tych przypadków ostateczny harmonogram dla danego uszeregowania będzie wyznaczony z użyciem metody ścieżki krytycznej z realokacją zadań, więc dwie pierwsze funkcje celu możemy potraktować jako przybliżone miary jakości uszeregowania. Na rys. 2a przedstawione zostało uszeregowanie planów minimalizujące całkowity czas przezbrojeń. Uszeregowanie z rys. 3a minimalizuje ścieżkę krytyczną w odpowiadającej mu sieci działań, zaś uszeregowanie z rys. 4a minimalizuje ścieżkę krytyczną z realokacją

zadań. Dla wszystkich trzech uszeregowań, wyznaczone zostały harmonogramy szczegółowe z użyciem metody ścieżki krytycznej bez- oraz z realokacją zadań. Wynikowe harmonogramy zostały przedstawione w podpunktach b) oraz c) na rysunkach 2-4.

6. EKSPERYMENTY OBLICZENIOWE

Przeprowadzone zostały eksperymenty obliczeniowe dla pierwszej oraz drugiej fazy badanego algorytmu. We wszystkich badaniach zastosowano ten sam schemat generacji zadań testowych. Najpierw, dla każdego zadania k i maszyny l wygenerowane zostały czasy wykonania p_{lk} oraz ilości wymaganego zasobu a_{lk} jako liczby losowe o rozkładzie jednostajnym z przedziałów, odpowiednio, $[10, 30]$ oraz $[6, 10]$. Następnie określony został dostępny zasób wspólny. Jego poziom został tak ustalany, aby w danym momencie przeciętnie tylko 75% maszyn mogło pracować, podczas gdy pozostałe maszyny pozostają nieaktywne. Dla tych problemów, w których liczba zadań jest mniejsza niż liczba maszyn, ograniczenie zasobowe jest wciąż aktywne dzięki ustaleniu go na poziomie, przy którym tylko 75% zadań może być jednocześnie wykonywanych. Na końcu, dla każdej maszyny i dla wszystkich możliwych kombinacji zadań będących poprzednikami i następnikami, wygenerowane zostały czasy przezbrojeń jako losowe liczby ciągłe o rozkładzie jednostajnym z przedziału $[1.2, 2.0]$, co stanowiło w przybliżeniu 6 do 10 procent średniego czasu wykonania p_{lk}. Jeśli poprzednikiem i następnikiem było to samo zadanie, czas przezbrojenia był zerowy. Wszystkie eksperymenty zostały przeprowadzone na komputerze PC klasy Pentium IV, 1.7 GHz z wykorzystaniem pakietu CPEX 6.0.

W celu zbadania efektywności obliczeniowej pierwszej fazy algorytmu przeprowadzona została seria testów dla różnych rozmiarów problemu i dla różnych poziomów dostępnego zasobu wspólnego. Wyniki przedstawione w Tabeli 2 reprezentują średni czas obliczeń dla 10 losowych zadań, dla których zasób wspólny został ustalony na poziomie nie stanowiącym żadnego ogranicznika. W Tabeli 3 przedstawiono natomiast wyniki testów dla zadań z aktywnym ograniczeniem zasobowym. We wszystkich testach ustanowiono 200-stu sekundowy limit czasu obliczeń. Indeks obok liczby oznaczającej średni czas obliczeń, oznacza liczbę zadań (spośród 10), które albo przekroczyły limit czasu albo zakończyły się z powodu błędów numerycznych.

Celem eksperymentów dla drugiej fazy badanego algorytmu było porównanie trzech wersji algorytmów szeregowania używających przybliżonych i dokładnej miary jakości. Wszystkie wyniki są średnią z wykonania 10 losowych problemów z 10 maszynami, 40 zadaniami i aktywnym ograniczeniem zasobowym. Parametry użytego algorytmu genetycznego były następujące. Prawdopodobieństwa mutacji przez zamianę oraz przez proste odwrócenie były takie same i równe 0.03. Prawdopodobieństwo krzyżowania było równe 0.5 zaś rozmiar populacji wynosił 50.

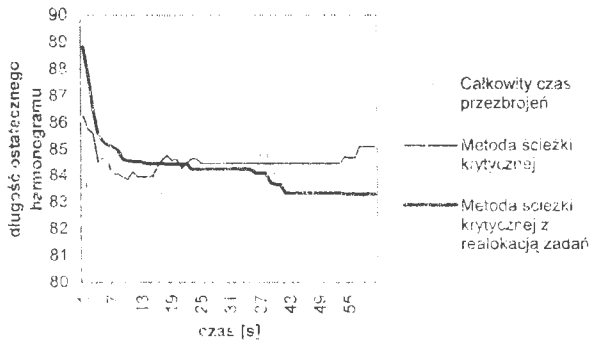
liczba	liczba zadań			
maszyn	10	20	30	40
10	0.0	0.2	0.3	0.6
20	0.0	0.2	0.7	1.3
30	0.0	0.3	0.6	2.0
40	0.1	0.2	0.6	2.5

Tabela 2. Czasy obliczeń fazy pierwszej dla zadań o różnych rozmiarach ze swobodnie dostępnym zasobem wspólnym.

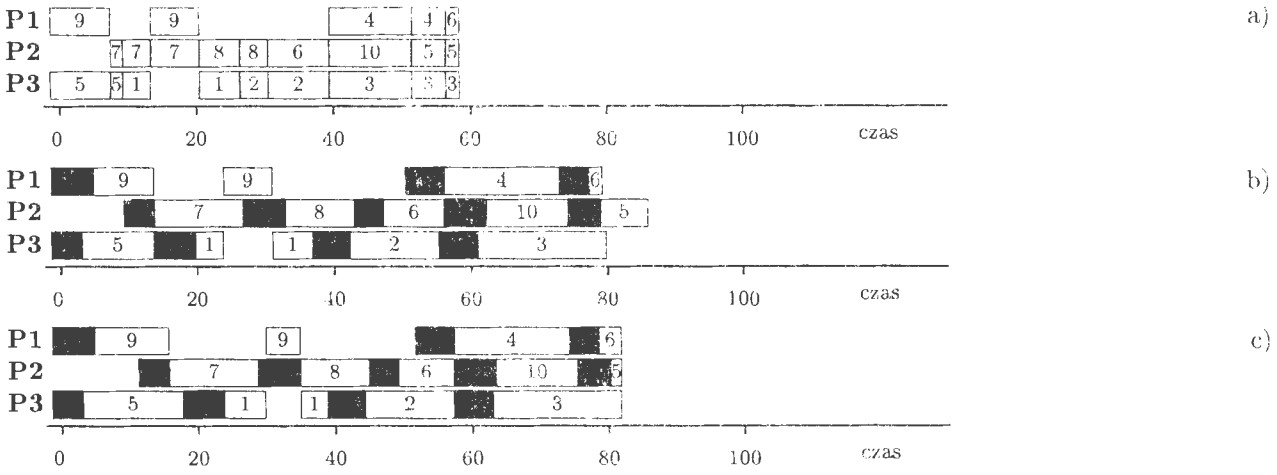
liczba	liczba zadań			
maszyn	10	20	30	40
10	0.2	1.6	2.0	4.7
20	0.6	5.3	8.5	15.7
30	0.9	9.7	19.2	40.4
40	0.8	^{2}90.7	57.6	^{1}71.7

Tabela 3. Czasy obliczeń fazy pierwszej dla zadań o różnych rozmiarach z ograniczonym zasobem wspólnym.

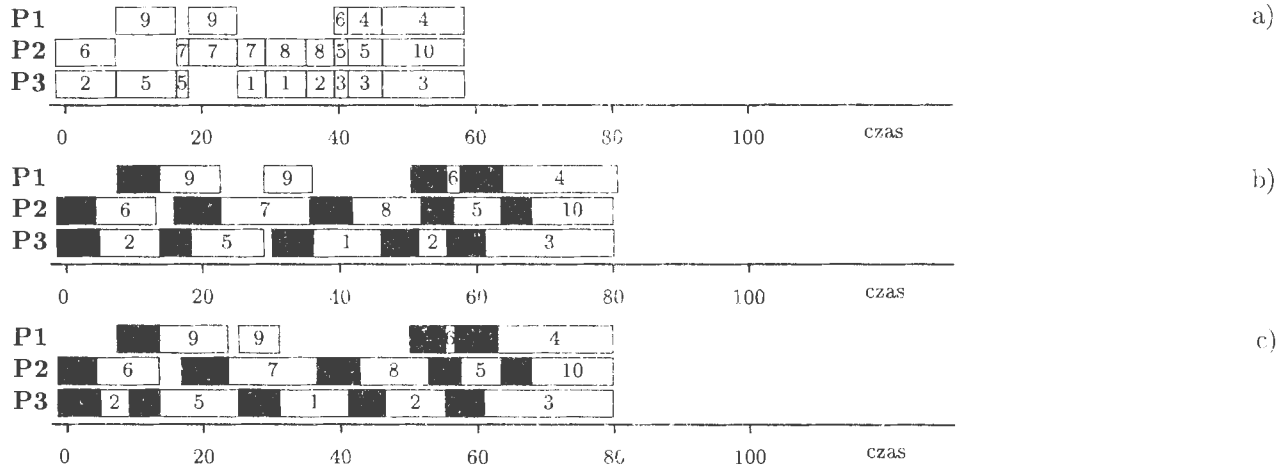
W pierwszym eksperymencie przetestowana została zbieżność algorytmów w czasie. Wyniki przedstawione na Rysunku 5 odpowiadają sytuacji, gdy algorytm szeregowania zatrzymuje się po danej liczbie sekund, a dla najlepszego znalezionego do tej pory uszeregowania jest wyznaczany końcowy harmonogram szczegółowy metodą ścieżki krytycznej z realokacją zadań. Wyjaśnia to, dlaczego wykresy odpowiadające użyciu miar przybliżonych nie są monotoniczne. W eksperymencie tym widać także efektywność wyznaczania poszczególnych miar jakości. Zgodnie z oczekiwaniami, podczas 60 sekund testów, największą liczbę razy został wyznaczony całkowity czas przezbrojeń (1124620 razy), na drugim miejscu był algorytm z metodą ścieżki krytycznej jako miarą jakości (275222 razy). Najbardziej złożone obliczeniowo okazało się być wyznaczanie ścieżki krytycznej z realokacją zadań (127 razy). Na rys. 5 widać, że algorytmy przybliżone były w stanie uzyskać dobre wyniki w ciągu kilku pierwszych sekund działania,



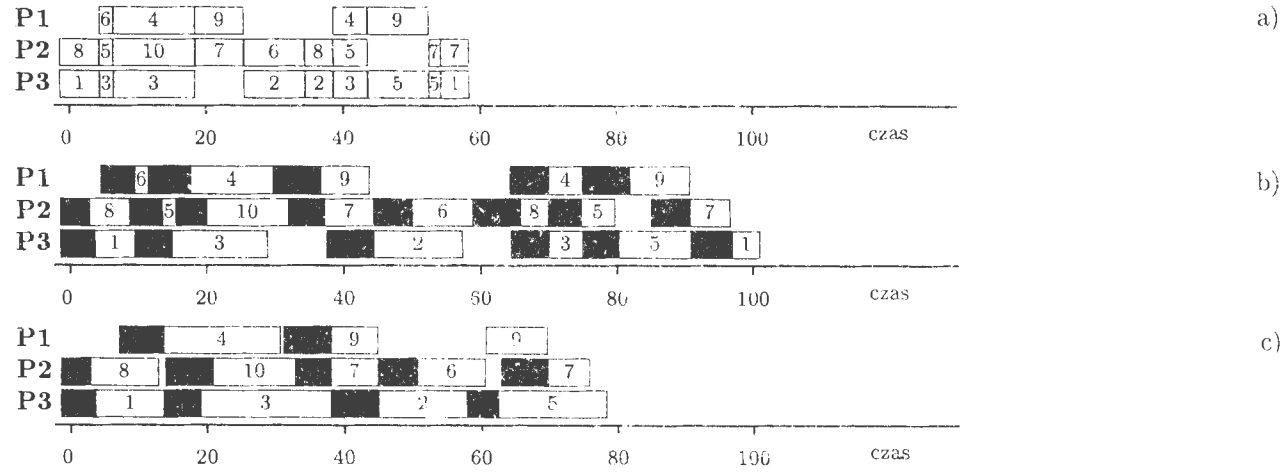
Rys. 5. Zbieżność w czasie algorytmu szeregowania dla różnych miar jakości uszeregowania.



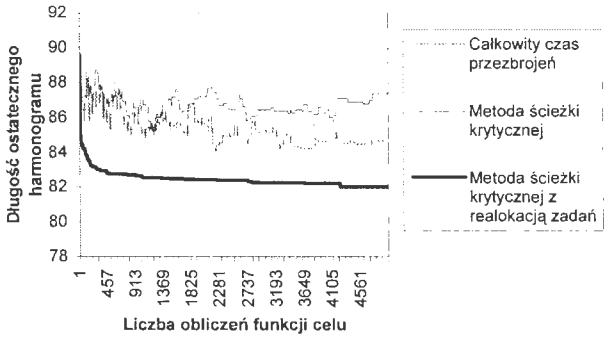
Rys. 2. Harmonogramy wyznaczone dla uszeregowania planów elementarnych minimalizującego całkowity czas przezbrojeń; a) uszeregowanie, b) harmonogram szczegółowy wyznaczony przy pomocy metody ścieżki krytycznej, c) harmonogram szczegółowy wyznaczony przy pomocy metody ścieżki krytycznej z realokacją zadań.



Rys. 3. Harmonogramy wyznaczone dla uszeregowania minimalizującego długość ścieżki krytycznej w harmonogramie szczegółowym *bez realokacji* zadań; a) uszeregowanie, b) harmonogram szczegółowy wyznaczony przy pomocy metody ścieżki krytycznej, c) harmonogram szczegółowy wyznaczony przy pomocy metody ścieżki krytycznej z realokacją zadań.



Rys. 4. Harmonogramy wyznaczone dla uszeregowania minimalizującego długość ścieżki krytycznej w harmonogramie szczegółowym *z realokacją* zadań; a) uszeregowanie, b) harmonogram szczegółowy wyznaczony przy pomocy metody ścieżki krytycznej, c) harmonogram szczegółowy wyznaczony przy pomocy metody ścieżki krytycznej z realokacją zadań.



Rys. 6. Zbieżność algorytmu szeregowania w zależności od liczby obliczeń funkcji celu dla różnych miar jakości uszeregowania.

podczas gdy algorytm dokładny potrzebował na to znacznie dłuższego czasu.

W drugim doświadczeniu zbadano zbieżność algorytmów w funkcji liczby obliczeń funkcji celu. W tym miejscu należy wyjaśnić przyczynę, dla której zdecydowano się na uwzględnienie liczby obliczeń funkcji celu nie zaś np. liczby iteracji, co jest powszechnie przyjmowane w literaturze. Po pierwsze, z obliczeniem funkcji celu związany jest wysiłek obliczeniowy algorytmu, po drugie, zmiana parametrów algorytmu genetycznego, np. prawdopodobieństwa mutacji, wpływa na liczbę obliczeń funkcji celu w jednej iteracji i w ten sposób zmienia czas trwania iteracji. A ponieważ główne zainteresowanie jest kierowane na całkowity czas obliczeń, wygodniej jest operować liczbą obliczeń funkcji celu zamiast liczbą iteracji.

W eksperymencie nie wprowadzono ograniczenia czasowego, ustalono natomiast maksymalną liczbę obliczeń funkcji celu na 5000. Testowane zadania są dokładnie takie same jak w pierwszym eksperymencie. Wyniki przedstawiono na rys. 6. Można zauważyć, że algorytm korzystający z dokładnej miary jakości dał lepsze rezultaty niż algorytmy z miarami przybliżonymi. Stało się to jednak kosztem czasu potrzebnego do wykonania zadania – algorytm z miarą przybliżoną zakończył działanie po kilku sekundach, podczas gdy algorytm z miarą dokładną potrzebował do zakończenia ponad pół godziny.

7. PODSUMOWANIE

Badany w pracy strukturalny algorytm szeregowania zadań podzielnych na maszynach równoległych z przezbrojeniami i ograniczeniami zasobowymi łączy skutecznie różne elementarne techniki optymalizacji, w tym technikę generacji kolumn, algorytmy ewolucyjne, programowanie liniowe i specjalizowane heurystyki. Wyniki eksperymentów obliczeniowych wskazują na wysoką efektywność pierwszej wersji algorytmu szeregowania planów elementarnych w przypadku znaczącego limitowania czasu obliczeń. Zwiększenie limitu obliczeń jest najbardziej skutecznie wykorzystywane w wariancie algorytmu szeregowania planów z wykorzystywaniem metody ścieżki krytycznej z realokacją zadań.

ALGORITHM FOR PREEMPTIVE TASK SCHEDULING ON PARALLEL PROCESSORS WITH SETUP TIMES AND RENEWABLE RESOURCES

Abstract: A two-stage algorithm for scheduling preemptive tasks on parallel machines with minimum makespan criterion and requirements for limited renewable resources and existence of sequence dependent setup times is investigated. In the first stage of the algorithm a set of best elementary feasible plans is obtained through column generation. For the second stage we compare genetic algorithms for sequencing elementary plans, where various approximate criterions for calculation minimum makespan are used.

Literatura

[1] G. Dobson, U. Karmarkar, J. Rummel (1989). Batching to minimize flow times on parallel heterogeneous machines. Mgmt Sci. 35, 607-613.

[2] Ewa Figielska, E. Toczyłowski, Algorytm szeregowania podzielnych operacji na równoległych procesorach przy ograniczeniach zasobów zużywalnych i odnawialnych, *Optymalizacja w zagadnieniach kombinatorycznych*, Wyd. WSM, Gdynia, 1997, 18-31.

[3] Figielska, E.: Preemptive scheduling with changeovers: using column generation technique and genetic algorithm, Computers & Industrial Engineering 37, 1999, 81-84.

[4] M.D. Oliff (1987). Disaggregate planning for parallel processors. IIE Trans 19, 215-219.

[5] K. So (1990). Some heuristics for scheduling jobs on parallel machines with setups. Mgmt Sci. 36, 467-475.

[6] E. Toczyłowski (1989). Niektóre metody strukturalne optymalizacji do sterowania w dyskretnych systemach wytwarzania. WNT, Warszawa 1989.

[7] E. Toczyłowski, 'Algorithms for preemtive scheduling of independent tasks in the presence of general renewable and consumable resources', *Zeszyty Nauk. AGH*, Automatyka 59, 163-172 (1991)

[8] Hindi, K. S., E. Toczyłowski, Detailed Scheduling of Batch Production in a Cell with Parallel Facilities and Common Renewable Resources, *Computers and Industrial Engineering*, Vol. 28, pp. 839-850, Elsevier, 1995

[9] Toczyłowski, E.: Preemptive Scheduling of independent Tasks in the Presence of Setup Times and Renewable Resources, 17th International Symposium on Mathematical Programming ISMP'2000, 7-11 August 2000, Atlanta, USA

Instytut Badań Systemowych
Polskiej Akademii Nauk

ISBN 83-89475-01-4