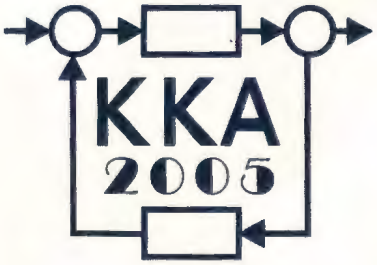
XV Krajowa Konferencja Automatyki

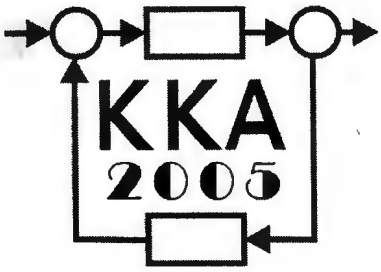
Tom II



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

ZASTOSOWANIE PROGRAMOWANIA Z OGRANICZENIAMI DO WARIANTOWANIA OBSŁUGI ZLECEŃ PRODUKCYJNYCH W MŚP

Paweł SITEK*, Jarosław WIKAREK*, Zbigniew BANASZAK**

*Politechnika Świętokrzyska, Wydział Elektrotechniki, Automatyki i Informatyki i Elektroniki
Al. 1000-lecia Państwa Polskiego 7, 25-314 Kielce, e-mail: sitek@tu.kielce.pl*

*Politechnika Świętokrzyska, Wydział Elektrotechniki, Automatyki i Informatyki i Elektroniki
Al. 1000-lecia Państwa Polskiego 7, 25-314 Kielce, e-mail: j.wikarek@tu.kielce.pl*

**Politechnika Koszalińska, Wydział Elektroniki i Informatyki
ul. Śniadeckich 2, 75-620 Koszalin, e-mail: banaszak@tu.koszalin.pl

Streszczenie: W pracy przedstawiono problem wariantowania obsługi zleceń produkcyjnych dla MŚP w kontekście programowania z ograniczeniami. Zaproponowano uproszczony model harmonogramowania, który w określonych warunkach może służyć do wariantowania obsługi zleceń. Prezentowany model został zaimplementowany w języku programowania z ograniczeniami Oz pakietu MOZART. Przedstawiono wyniki eksperymentów obliczeniowych i zestawiono je z wynikami uzyskanymi przy zastosowaniu algorytmów listowych, które zwykle stosuje się do tego typu problemów.

Słowa kluczowe: CLP, harmonogramowanie, optymalizacja, algorytmy listowe.

1. WPROWADZENIE

Zagadnienie bilansowania możliwości (zdolności produkcyjnych) przedsiębiorstwa z potrzebami (wymaganiami) wynikającymi z realizacji zlecenia należy do klasy problemów silnie NP.-trudnych. To, co stanowi o jego istocie wiąże się z gwarancją tego, aby możliwości i potrzeby nie tylko bilansowały się w planowanym okresie, ale bilansowały się również w każdym momencie tego okresu.

Typowe sformułowanie zagadnienia bilansowania sprowadza się do problemu planowania obsługi zleceń produkcyjnych w warunkach występowania czasowych ograniczeń w dostępie do zasobów systemu wytwórczego. Dane jest przedsiębiorstwo zorientowane na wieloasortymentową, współbieżnie realizowaną produkcję jednostkowa i/lub krótkoseryjną. W szczególności znane są parametry stanowisk wytwórczych, buforów i magazynów, tras jezdnych, środków transportu, itd. Znane są także okresy dostępności poszczególnych zasobów oraz związane z tymi okresami koszty wykorzystania zasobów. Dany jest zbiór potencjalnych zleceń produkcyjnych. Każde ze zleceń scharakteryzowane jest wielkością produkcji, oczekiwanym terminem jej ukończenia, kosztem, sposobem wykonania itd. Poszukiwana jest odpowiedź na pytanie: Czy zlecenia te (ewentualnie, które z nich?) mogą być przyjęte do realizacji? tzn. czy mogą być wykonane po zadanych kosztach i w planowanych terminach?

2. ZAGADNIENIE BILANSOWANIA JAKO PROBLEM SPEŁNIENIA OGRANICZEŃ

Rozważany problem w sposób naturalny można sformułować w kategoriach problemu spelnienia ograniczeń typowego sposobu deklaratywnego przedstawienia programu w systemach programowania z ograniczeniami – CLP (Constraint Logic Programming).

Problem spełnienia ograniczeń ((X,D),C). Dany jest: skończony zbiór zmiennych decyzyjnych $X = \{x_1, x_2, \ldots, x_n\}$, skończona rodzina skończonych dziedzin dyskretnych zmiennych decyzyjnych $D = \{D_i \mid D_i = \{d_{i1}, d_{i2}, \ldots, d_{ij}, \ldots, d_{in}\}, i = 1..n\}$, skończony zbiór ograniczeń limitujących wartości zmiennych decyzyjnych $C = \{C_i \mid i = 1..L\}$, gdzie: $c \in C$ jest pewnym predykatem $P[x_k, x_l, \ldots, x_h]$ zdefiniowanym na podzbiorze zbioru X. Poszukiwane jest rozwiązanie dopuszczalne, tzn. rozwiązanie, w którym wartości wszystkich zmiennych decyzyjnych spełniają wszystkie ograniczenia.

Na zbiorze rozwiązań dopuszczalnych, zbiorze wektorów rozwiązań, można zdefiniować funkcje celu, kryterium i sformułować problemem optymalizacyjny (COP - Constraint Optimisation Problem) (X, D, C, f), gdzie f jest funkcją przypisującą wartość liczbową każdemu wektorowi rozwiązania $X=(x_1,\ldots, x_n)$.

Rozwiązanie problemu COP polega na znalezieniu wektora rozwiązania $X=(x_1,\ldots, x_n)$, który ekstremalizuje funkcję f [1].

Celem pracy jest ukazanie naturalnych zastosowań języków programowania z ograniczeniami w zagadnieniach bilansowania ograniczonych w czasie zdolności produkcyjnych z potrzebami planowanej realizacji zle-

ceń, odniesienie związanych z tym możliwości do komercyjnie dostępnych pakietów programowania matematycznego, a także ukazanie możliwości wykorzystania tych języków w zagadnieniach optymalizacyjnych, np. szeregowania zadań.

3. PROGRAMOWANIE W LOGICE Z OGRANICZENIAMI

Programowanie w logice z ograniczeniami jest zestawem technik i strategii programistycznych umożliwiających modelowanie i rozwiązywanie zagadnień wyznaczania rozwiązań dopuszczalnych i optymalnych dla problemów kombinatorycznych, ciągłych itd. na drodze deklaratywnej. Deklaratywność oznacza, że odpowiedni opis zadania jest programem je rozwiązującym. Głównymi technikami należącymi do programowania w logice z ograniczeniami jest propagacja ograniczeń, backtracking, strategie przeszukiwania i optymalizacji [2].

3.1. Strategie przeszukiwania

Większość metod przeszukiwań w problemach z ograniczeniami oparta jest na mechanizmach „backtrackingu". W językach należących do PLC np. w języku CHIP (Constraint Handling in Prolog), tzw. *standard backtracking* został zastąpiony bardziej efektywnymi strategiami poszukiwan – *Forward Checking* (FC) i *Looking Ahead* (LA) - wspomaganymi efektywną strategią numeracji -- *First Fail* (FF). W algorytmie FC po ukonkretnieniu nowej zmiennej z domen pozostałych zmiennych są usuwane wartości sprzeczne z już ukonkretnionymi zmiennymi.
W algorytmach LA sprawdza się czy domeny zmiennych nie ukonkretnionych maja wartości dopuszczalne. Mechanizmem wyzwalającym powyższe algorytmy backtrackingu nie jest naruszenie ograniczeń jak w *standard backtracking* lecz nieuchronność naruszeń ograniczenia w następnym kroku (FC) oraz nieuchronność naruszenia w dalszych krokach (LA) na co wskazuje obecność niepustych domen ale bez wartości dopuszczalnych.
Wymienione techniki występują w językach CLP w sposób naturalny, co oznacza, że nie wymagają specjalnego ich oprogramowywania. Uzupełnieniem powyższych technik jest strategia numeracji tzn. wyboru kolejnych zmiennych ukonkretnianych w trakcie poszukiwań. Najczęściej stosowaną strategią numeracji jest FF, zgodnie z którą jako pierwsza jest ukonkretniana zmienna mająca najmniejszą domenę.

3.2. Strategie optymalizacji

Mocną stroną języków CLP jest również optymalizacja kombinatoryczna realizowana za pomocą idei znanej z badań operacyjnych – *Branch-and-Bound* (B&B). Idea B&B w CLP polega m.in. na:

- Znalezieniu dolnego ograniczenia wartości wskaźnika jakości (*f*)- (*bound*);
- Odrzuceniu wszystkich tych gałęzi drzewa rozwiązań, które dają wartości gorsze dla *f* od dolnego ograniczenia (*branch*);
- Aktualizacji dolnego ograniczenia w przypadku znalezienia rozwiązania lepszego niż w przypadku stosowania dolnego ograniczenia;

i jest wspomagana strategiami FC oraz LA.
W przypadku zastosowania B&B wraz z FC inicjowanie *backtrackingu* następuje w dwóch przypadkach:

- W wyniku osiągnięcia wskaźnika jakości gorszego od aktualnego ograniczenia (B&B);
- W wyniku uzyskania pustej domeny (FC).

Jeśli zostaną zastosowane metody B&B, FC wraz z LA to inicjowanie *backtrackingu* następuje dodatkowo:

- W wyniku braku wartości dopuszczalnych w niepustych domenach (LA).

4. WARIANTOWANIE OBSŁUGI ZLECEŃ PRODUKCYJNYCH W MŚP

Wariantowanie obsługi zleceń produkcyjnych w MŚP należy do typowego zagadnienia bilansowania możliwości produkcyjnych przedsiębiorstwa z potrzebami wynikającymi z realizacji zleceń, a co za tym idzie z wymaganiami i oczekiwaniami klientów. Powyższy problem rozważany w kontekście spełnienia ograniczeń sprowadza się do wyznaczenia minimalnego czasu wykonania zbioru zleceń na zbiorze posiadanych zasobów w sytuacji gdy znane są zbiory:

- wyrobów $j \in J$ (typów wyrobów),
- zasobów systemu wytwórczego $s \in S$,
- operacji $o \in O$ (operacje dotyczące produktu j należą do zbioru O_j, $O_j \subset O$),
- czasów operacji p_{jos} dla zbioru operacji na zasobie s (czas ten jest równy pojedynczej operacji lub operacji dla partii), $p_{jos} = 0$ gdy dla wyrobu $j \in J$ wykonywanego przy użyciu zasobu $s \in S$ operacja $o \in O_j$ nie jest dostępna (nie istnieje).

Przedstawiony przykład można sklasyfikować jako deterministyczny problem szeregowania zadań. Założenie, że wszystkie parametry problemu znane są a priori odwrotnie niż w problemach probabilistycznych jest uzasadnione w wielu przypadkach praktycznych [3].
Dla MŚP podjęcie decyzji o możliwości realizacji danego zlecenia czy zbioru zleceń bardzo często jest decyzją krytyczną decydującą o istnieniu przedsiebiorstwa [4]. Dopiero w drugiej kolejności może być dokonywana optymalizacja podejmowanej decyzji w oparciu o wybrany wskaźnik jakości. Cechą charakterystyczną MŚP zorientowanego na wieloasortymentową produkcję jest występowanie czasowych ograniczeń dostępu do zasobów. Cecha ta musi być brana pod uwagę przy wielu problemach decyzyjnych, które występują w MŚP, a w szczególności w zagadnieniach obsługi zleceń produkcyjnych.
Jako przykład ilustracyjny wariantowania zleceń produkcyjnych zaproponowano uproszczony model harmonogramowania zleceń w ogólnym systemie obsługi (jobshop) przy dedykowanych zasobach [3]. Przyjęto pewne uproszczenia polegające między innymi na braku alternatywnych marszrut wykonania zleceń. Przy budowie modelu uwzględniono deklaratywność środowiska CLP, już na etapie jego budowy, a jedno z ograniczeń przedstawiono w formie algorytmicznej.

W pierwszym podejściu rozważany przykład można sformułować jako problem spełnienia ograniczeń przy zmiennych decyzyjnych, których wartościami są chwile czasowe rozpoczęcia i zakończenia operacji wykonywanych na posiadanych zasobach (Tab. 1). Operacje są wykonywane dla wszystkich wyrobów. Innymi słowy jest to zadanie znalezienia realizowalnego harmonogramu. Rozwiązanie takiego zadania jest jednoznaczne z pozytywną odpowiedzią na pytanie: czy można zrealizować wymagany zbiór operacji dla wyrobów na posiadanych zasobach. Rozwiązanie takie, choć dopuszczalne (realizowalne) nie posiada żadnego wskaźnika jakości i w praktyce może być wstępem do dalszych obliczeń. Podając kryterium jakości w postaci funkcji celu zadanie staje się problemem optymalizacji długości uszeregowania. Jego rozwiązaniem jest optymalny harmonogram.

Tabela 1. Zmienne decyzyjne problemu harmonogramowania

X_{jos}	zmienna decyzyjna oznaczająca chwilę czasu rozpoczęcia operacji $o \in O_j$, dla wyrobu $j \in J$, na zasobie $s \in S$;
X_{krs}	zmienna decyzyjna oznaczająca chwilę czasu rozpoczęcia operacji $r \in O_k$, dla wyrobu $k \in J$, na zasobie $s \in S$, przy czym $j \neq k$
Y_{jos}	zmienna decyzyjna oznaczająca chwilę czasu zakończenia operacji $o \in O_j$ dla wyrobu $j \in J$ na zasobie $s \in S$;

Zastosowanie metodyki CLP do modelowania harmonogramowania zleceń produkcyjnych pozwoliło na wyrażenie problemu w postaci ograniczeń.

$$X_{jos} + p_{jos} \leq Y_{jos} \tag{1}$$

dla $j \in J$, $o \in O_j$, $s \in S$

$$p_{jo+1s} * X_{jo+1s} \geq p_{jo+1s} * Y_{jos} \tag{2}$$

dla $j \in J$, $o \in O_j$-{ostatnia operacja dla j}, $j \in J$, $s \in S$

(3)

$s \in S$		
	j, $k \in J$, $j \neq k$	
		$o \in O_j$ takich dla których $p_{jos} > 0$ $r \in O_k$ takich dla których $p_{krs} > 0$
		Jeśli $X_{jos} \leq X_{krs}$ to $Y_{jos} \leq X_{krs}$

Ograniczenie (1) określa czas trwania operacji i wiąże ze sobą zmienne decyzyjne X_{jos}, Y_{jos}. Ograniczenie (2) jest ograniczeniem kolejnościowym, które zapewnia, że moment rozpoczęcia kolejnej operacji $o \in O_j$ zlecenia na wyrób $j \in J$ nie rozpocznie się przed zakończeniem operacji ją poprzedzającej. W danej chwili czasu zasób $s \in S$ nie może być przydzielony jednocześnie do dwóch operacji $o \in O_j$, $r \in O_k$, wykonywanych na wyrobach j, $k \in J$, dlatego wprowadzono ograniczenie zasobowe (3), które ogranicza jednoczesny dostęp różnych operacji do współdzielonych przez nie zasobów systemu. Ograniczenie (3) zostało sformułowane w postaci iteracyjnego algorytmu działającego w dwóch pętlach. Pierwsza zewnętrzna pętla jest uruchamiana dla każdego zasobu $s \in S$, druga wewnętrzna dla wszystkich wyrobów j, $k \in J$. Następnie jest sprawdzany warunek logiczny dotyczący czasów rozpoczęcia różnych operacji przydzielonych do zasobu i w zależności od relacji między nimi ustawiane jest powiązanie między końcem operacji zaczynającej się wcześniej a rozpoczęciem następnej operacji na tym zasobie.

Uzupełniając zaproponowany model (1) - (3) o wskaźnik jakości, który w tego typu problemach określany jest najczęściej jako minimalna długości uszeregowania, (4) uzyskano model optymalizacji długości uszeregowania, którego rozwiązaniem jest optymalny harmonogram obsługi zleceń.

$$C_{max} \tag{4}$$

W powyższym modelu nie występują explicite czasowe ograniczenia w dostępie do zasobów. Problem ten jest rozwiązywany na poziomie implementacji, co czyni model prostym i uniwersalnym.

5. IMPLEMENTACJA MODELU

Do implementacji modelu sformułowanego w postaci (1) - (4) wykorzystano środowisko pakietu MOZART i język programowania Oz (Rys. 1). Język Oz jest językiem programowania w logice z ograniczeniami. Implementacja modelu składała się z kilku faz. W pierwszej fazie zaproponowano odpowiednie struktury danych. Aby implementacja posiadała własności skalowalności, co jest ważnym czynnikiem ze względu na elastyczność implementacji oraz łatwość przeprowadzania eksperymentów obliczeniowych, zastosowano struktury w postaci list zarówno dla parametrów jak i zmiennych decyzyjnych. W postaci list zamodelowano również czasową niedostępność zasobów, która może być interpretowana w postaci dodatkowych zleceń pojawiających się w systemie z określonymi (niezmiennymi) chwilami rozpoczęcia poszczególnych operacji.

Następnie oprogramowano ograniczenia modelu, które ze względu na deklaratywność środowiska Oz/MOZART stały się fragmentem programu je rozwiązującym. Na szczególną uwagę zasługuje prostota implementacji ograniczenia zasobowego (3), które w sposób naturalny oprogramowano w języku Oz. Takie podejście w innych środowiskach obliczeniowych np. programowania całkowitoliczbowego w pakiecie LINGO nie jest możliwe. W pakiecie LINGO ograniczenie logiczne typu (3) należałoby rozbić na dodatkowe ograniczenia wprowadzając binarne zmienne przydziału, co niewątpliwie komplikuje model oraz zwiększa złożoność obliczeniową przy poszukiwaniu rozwiązania. Po dokonaniu implementacji w Oz/MOZART ograniczeń modelu z uwzględnieniem listowych struktur parametrów i zmiennych w następnym etapie dokonano wyboru sposobu rozwiązania modelu. W języku Oz dostępne są między innymi następujące możliwości rozwiązania modelu (Rys. 1):

- poszukiwanie pierwszego rozwiązani dopuszczalnego za pomocą predykatu języka Oz

SearchOne {P}, gdzie P jest funkcją opisującą model;

- poszukiwanie wszystkich rozwiązań dopuszczalnych za pomocą predykatu języka Oz *SearchAll {P}*, gdzie P jest funkcją opisującą model;
- poszukiwanie rozwiązania optymalnego za pomocą predykatu języka Oz *SearchBest {P} {W}*, gdzie P jest funkcją opisującą model, natomiast W jest specjalnie definiowaną procedurą określającą sposób zmiany wartości funkcji celu.

W języku Oz istnieją jeszcze inne sposoby poszukiwania rozwiązania modelu między innymi z rysowaniem drzewa przeszukiwań (Rys. 6) (*ExploreBest, ExploreOne).*

Po dokonaniu pełnej implementacji modelu w środowisku Oz/MOZART wykonano eksperymenty obliczeniowe między innymi dla przykładów 1, 2, 3, 4, których parametry przedstawiono w tabeli 2. Każdy przykład zawiera zbiór zleceń do wykonania w systemie.

Tabela 2 Przykłady liczbowe

Przykład 1	Przykład 2
j∈{1,2,3,4,5} o∈{1,2,3} s∈{1,2,3,4}	j∈{1,2,3,4,5} o∈{1,2,3} s∈{1,2,3,4}
j=1 [(1,2), (2,2)]	j=1 [(1,2), (2,2), (4,4)]
j=2 [(2,2), (3,2)]	j=2 [(1,1), (2,2)]
j=3 [(4,2), (3,2)]	j=3 [(1,3), (2,2)]
j=4 [(3,1), (4,1), (2,2)]	j=4 [(2,1), (3,1)]
j=5 [(1,2), (4,1)]	j=5 [(1,2), (2,2), (4,2)]

Przykład 3	Przykład 4
j∈{1,2,3,4,5} o∈{1,2,3} s∈{1,2,3,4}	j∈{1,2,3,4,5,6,7} o∈{1,2,3,4} s∈{1,2,3,4,5,6,7,8,9,10}
j=1 [(1,2), (2,2)]	j=1 [(1,1), (3,1), (4,2)]
j=2 [(2,2), (3,2), (4,1)]	j=2 [(1,2), (2,1)]
j=3 [(4,2), (3,2), (2,2)]	j=3[(2,2), (3,1), (5,2)]
j=4 [(3,2), (4,2), (2,1)]	j=4 [(5,2), (6,1), (2,1)]
j=5 [(1,2), (2,2), (4,2)]	j=5 [(4,2), (5,3), (2,1),(3,1)]
	j=6 [(7,5), (8,1), (9,2),(10,1)]
	j=7 [(8,1), (9,1), (10,4)]

Zlecenie jest określane za pomocą zbioru par (Tab. 2). Pozycja pary w zbiorze określa numer operacji zlecenia. Pierwszy element pary oznacza numer zasobu, na którym wykonywana jest operacja natomiast drugi to czas trwania operacji.

Dla każdego przykładu poszukiwano zarówno rozwiązań dopuszczalnych (dopuszczalne harmonogramy realizacji zleceń) oraz rozwiązania optymalnego ze względu na długość uszeregowania C_{max} (optymalny harmonogram realizacji zleceń). Dodatkowo, dla przykładu 2 wprowadzono czasową niedostępność zasobów. Zasób s=1 jest niedostępny w przedziale <4,5> natomiast s=2 w przedziale <6,7>. Pierwsze uzyskane rezultaty oraz przebieg eksperymentów prowadzą do następujących spostrzeżeń (tab.3):

- Rozwiązanie dopuszczalne znajdowane jest znacznie szybciej niż rozwiązanie optymalne;
- Poszukiwanie rozwiązania wraz z rysowaniem drzewa przeszukiwań wydłuża czas obliczeń i absorbuje znacznie zasoby sprzętu komputerowego (dla większych przykładów, np. 4 brakowało pamięci operacyjnej)
- Kolejność ukonkretniania zmiennych ma decydujący wpływ na szybkość znajdowania rozwiązania.

Tabela 3 Zestawienie wyników

Przykład/ kole. Ukonkretniana.	Rozwiązanie dopuszczalne		Rozwiązanie opty.	
	C_{max}	Czas (ms)	C_{max}	Czas (ms)
1 /{C_{max}, X}	6	359	6	328
1 /{X, C_{max} }	10	9 015	6	23 969
2 /{C_{max}, X}	10	240 484	10	242 906
2 /{C_{max}, X}*	11	424 484	11	468 246
2 /{X, C_{max} }	20	1 083 844	10	4 521 812
3 /{C_{max}, X}	9	626 531	9	614 703
3 /{X, C_{max} }	20	23 965 563	-----	-----
4 /{C_{max}, X}	9	151 922	9	148 469
4/ {X, C_{max} }	-----	-----	-----	-----

* Czasowe ograniczenia w dostępności zasobów

```
declare W
fun {W }
proc{$ R}
   S C X L
in
 S=[
    5       % Liczba wyrobów
    3       % Liczba operacji
    4       % Liczba zbiorów zasobowych
     ]
X={List.make {List.nth S 1} * {List.nth S 2} *
  {List.nth S 3}}
C={List.make {List.nth S 1} * {List.nth S 2} *
  {List.nth S 3}}
L={List.make 1}
   X:::0#100
   L:::0#20
   C=[
      2 0 0 0   0 2 0 0   0 0 0 0
      0 2 0 0   0 0 2 0   0 0 0 1
      0 0 0 2   0 0 2 0   0 2 0 0
      0 0 2 0   0 0 0 2   0 1 0 0
      2 0 0 0   0 2 0 0   0 0 0 2
   R=r(czasy:X koniec:L)
.....
end
end
%{ExploreBest {W}
% proc{$ Old New}% {List.nth Old.koniec 1} >:
% {List.nth New.koniec 1}
% end}
Wynik={SearchBest {W}
  proc{$ Old New}
   {List.nth Old.koniec 1}>:
   {List.nth New.koniec 1}
  end }
{Browse Wynik}
```

Rys. 1. Fragmenty listingu przedstawiającego model harmonogramowania zleceń produkcyjnych w środowisku Oz/MOZART.

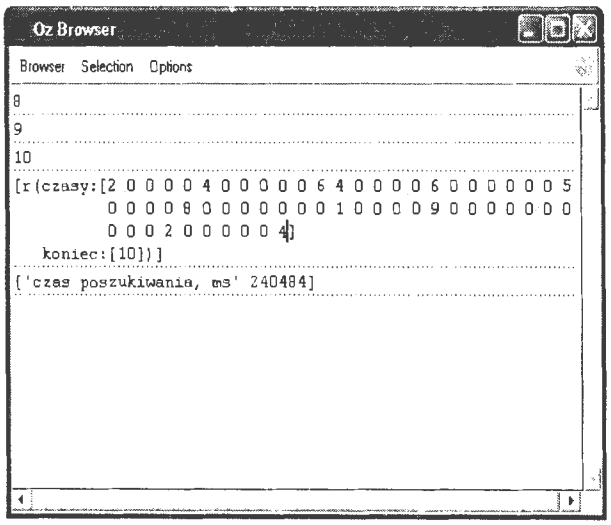
Na rys.2, rys.3, rys.4 przedstawiono zrzuty ekranu Browser`a systemu Oz/MOZART z uzyskanymi rozwiązaniami dla przykładu 2. Wizualizację rozwiązania optymalnego (rys.3) w postaci wykresu Gantt`a pokazano na rys. 5. W celu oszacowania jakości uzyskanego rozwiązania, np. optymalnego harmonogramu, zastoso-

wano do rozwiązania powyższego przykładu algorytmy listowe. Wiadomo z literatury [3], że jest to praktyczny sposób znajdowania rozwiązań tego typu problemów. Algorytmy listowe stosowane są do szeregowania zadań w ogólnych systemach obsługi (job shop) jeśli znany jest zbiór zasobów, na których ma być wykonane zadanie (zlecenie), jak i kolejność wykonywania operacji każdego zadania jest dowolna choć określona (marszruta). Algorytmy listowe należą do klasy algorytmów przybliżonych (heurystycznych, suboptymalnych) i polegają na przydzieleniu zadaniom priorytetów według określonej reguły. Na rys. 5, w postaci wykresów Gantt`a pokazano harmonogramy uzyskane przy pomocy algorytmów listowych LPT, LWR oraz metody CLP – Oz/MOZART. Zestawienie uzyskanych wyników (Tab.4) wskazuje, że długość uszeregowania jest najmniejsza dla zaproponowanego modelu harmonogramowania i jego implementacji w środowisku Oz/MOZART. Bezpośrednie porównanie może dotyczyć jedynie wyników eksperymentów dla przykładów, w których nie występują blokady zasobów.

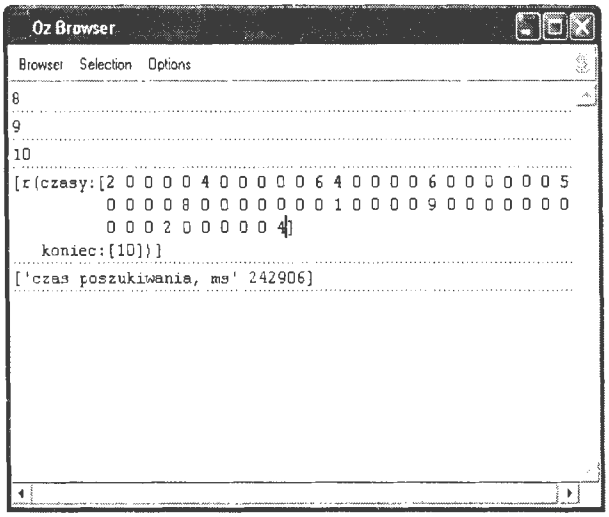
Tabela 4 Długość uszeregowania (przykład 2)

Kryterium	LPT	LWR	CLP - Oz	CLP – Oz*
C_{max}	13	11	10	11

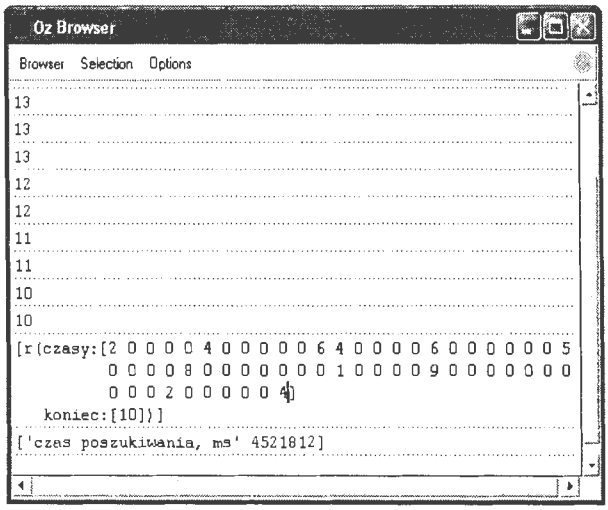
* Czasowa niedostępność zasobów



Rys. 2. Rozwiązanie dopuszczalne (przykład 2) kolejność ukonkretniania zmiennych (C_{max}, X)



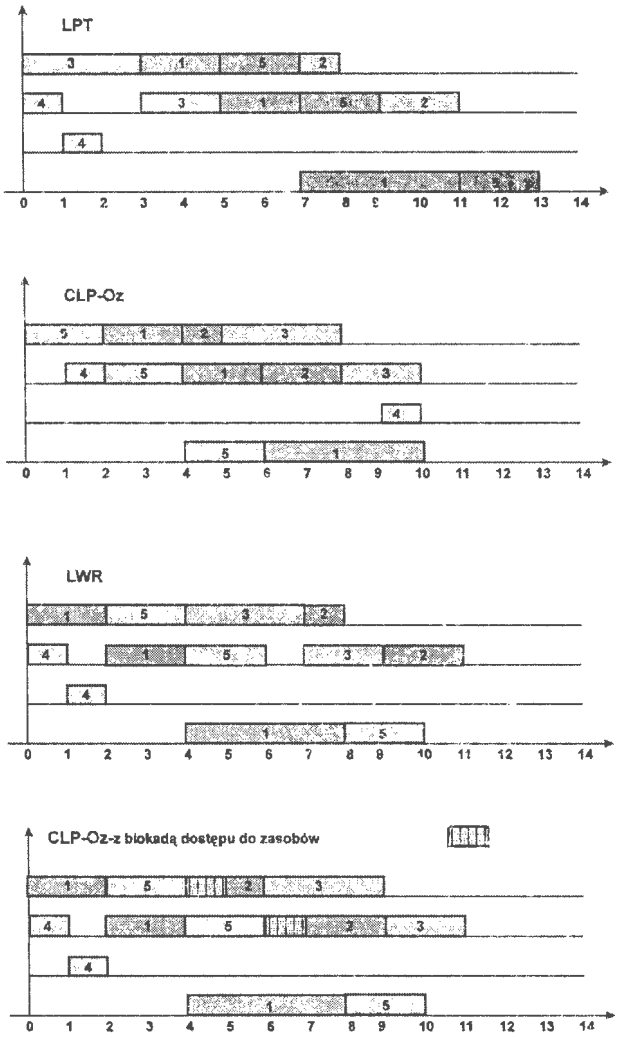
Rys. 3. Rozwiązanie optymalne (przykład 2) kolejność ukonkretniania zmiennych (C_{max}, X)



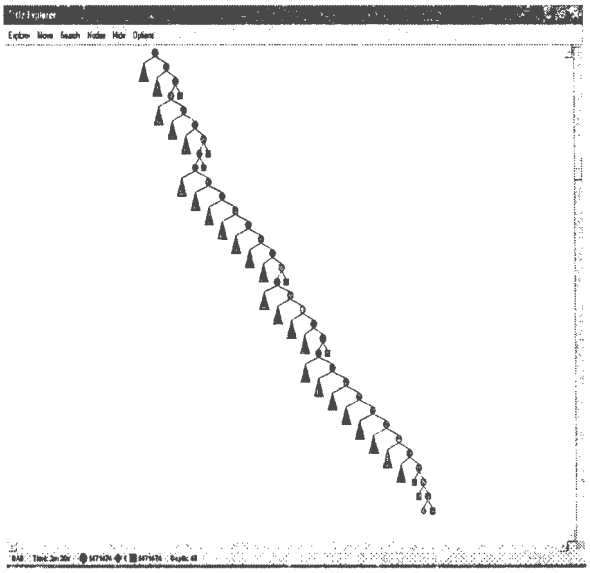
Rys. 4. Rozwiązanie optymalne (przykład 2) kolejność ukonkretniania zmiennych (X, C_{max})

6. WNIOSKI

Zastosowanie metodyki programowania w logice z ograniczeniami jest niezwykle interesujące w odniesieniu do ogólnie rozumianych problemów harmonogramowania zleceń produkcyjnych. Proponowane podejście jest interesujące przynajmniej w dwóch aspektach. Po pierwsze, zaproponowany model wraz z implementacją umożliwia harmonogramowanie zleceń w MŚP przy uwzględnieniu czasowych ograniczeń w dostępie do zasobów. Ograniczenia takie nie są uwzględniane w standardowych algorytmach listowych. Po drugie, proponowane rozwiązanie daje lepsze harmonogramy (mniejsze C_{max}) niż metody heurystyczne w praktyce stosowane do tego typów problemów (np. algorytmy listowe). Dodatkowo, deklaratywność metodyki CLP powoduje łatwość modelowania takich problemów, w których często w sposób logiczny należy wyrazić zależność przyczynowo – skutkową (np. ograniczenie 3). Ponadto środowisko implementacyjne CLP np. Oz/MOZART posiada bardzo silne strategie przeszukiwania i propagacji ograniczeń.



Rys. 5. Harmonogramy w postaci wykresów Gantt`a (przykład 2) metody: LPT, CLP – Oz/MOZART, LWR.



Rys. 6. Drzewo poszukiwań rozwiązania optymalnego (przykład 2) kolejność ukonkretniania zmiennych (C_{max},X) – predykat *ExploreBest*

Propagacja ograniczeń w sposób zasadniczy zmniejsza domeny zmiennych decyzyjnych modelowanych problemów czyniąc wraz z metodami opartymi na backtrackingu metodykę CLP bardzo efektywną w poszukiwaniu rozwiązań. W problemach harmonogramowania zleceń, jeśli za kryterium oceny danego harmonogramu przyjmie się długość uszeregowania C_{max}, można wykorzystać jeszcze jeden aspekt metodyki programowania w logice z ograniczeniami. Tym aspektem jest możliwość wpływania na kolejność ukonkretniania zmiennych, co nie jest możliwe w środowiskach programowania matematycznego np. LINGO [5]. Dla rozpatrywanej klasy problemów harmonogramowania w ogólnym systemie obsługi przy optymalizacji długości uszeregowania wybranie C_{max} jako zmiennej decyzyjnej ukonkretnianej w pierwszej kolejności znacząco wpływa na szybkość znalezienia rozwiązania. Ta właściwość potwierdzona uzyskanymi wynikami (Tab. 3) jest niewątpliwą zaletą stosowania metodyki CLP do tej klasy problemów.

APPLICATION OF CLP TO PROTOTYPING SHOP ORDERS FOR SME

Abstract: The paper presents the problem of prototyping of shop orders support for SME's in terms of constraint programming. A simplified model, which can propose, in specific conditions, be used for that purpose. It was implemented in the constraint language Oz in MOZART package. The results of the calculation experiments were compared with/to the results obtained through priority rule algorithms, which usually used for this kind of problems.

Literatura

[1] Tsang E. (1995) *Foundation of Constraint Satisfations*, Academic Press, London.

[2] Niederliński A. (1999) *Constraint Logic Programming – From Prolog to CHIP*, Proceedings of the Workshop on Constraint Programming for Decision and Control, Gliwice, pp.27-34.

[3] Błażewicz J., Cellary W., Słowiński R., Węglarz J. (1983) *Badania operacyjne dla informatyków*. WNT, Warszawa.

[4] Tomczuk I., Bzdyra K., Banaszak Z. (2004) *Zastosowanie metod programowania z ograniczeniami w MŚP zarządzanych przez projekt*. WNT, Warszawa.

[5] Sitek P., Wikarek J., Zaborowski M. (2000) *Application of CLP to Optimization of Production Variants Configuration in The Flow Production Lines*, Proceedings of the 2-nd Workshop on Constraint Programming for Decision and Control, Gliwice, pp 53 – 59.

Instytut Badań Systemowych
Polskiej Akademii Nauk

ISBN 83-89475-01-4