ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994
Seria: AUTOMATYKA z.115 Nr kol. 1251

Jarosław NABRZYSKI, Jan WĘGLARZ

Politechnika Poznańska

KONCEPCJA SZKIELETOWEGO SYSTEMU EKSPERCKIEGO DLA PROBLEMÓW ROZDZIAŁU ZASOBÓW

Streszczenie: W pracy przedstawiono koncepcję szkieletowego systemu eksperckiego dla obszernej klasy deterministyczych problemów sterowania rozdziałem zasobów. Klasa ta jest scharakteryzowana przez następujące główne cechy: wiele rodzajów zasobów wszystkich kategorii, wiele sposobów wykonywania każdego zadania i wiele kryteriów optymalności. Opisano m.in. architekturę (tablicową) systemu, metajęzyk jej opisu oraz procedurę dialogową sterowania procesem konstrukcji uszeregowań kompromisowych za pomocą preferencji decydenta.

ON SHELL EXPERT SYSTEM FOR RESOURCE-CONSTRAINED PROJECT SCHEDULING

Summary: This paper describes an idea of a shell expert system for a broad class of project scheduling problems. This class is characterized by the following main features: many types of resources of all categories, many performing modes of each activity and many schedule performance measures. Special attention has been paid to the (array) system architecture, the meta-language for its description, and the dialog preference-driven procedure of finding compromise schedules.

CONCEPTION D'UN SYSTEME DE DEVELOPPEMENT DES SYSTEMES EXPERTS POUR L'ORDONNANCEMENT DE PROJETS

Résumé: On présente une conception d'un systeme de développement des systemes experts pour une large classe de problémes d'ordonnancement deterministe. La classe est caractérisée par: plusieurs types de moyens, plusieurs modes d'exécution de chaque tâche et plusieurs criteres d'optimisation. On présente une architecture (type tableau) du systeme, un meta-language de sa déscription et une procédure de contrôle par des préférences d'un décideur.

1. Sformułowanie problemu

Projektowany szkieletowy system ekspercki ma wspomagać podejmowanie decyzji dla obszernej klasy problemów rozdziału zasobów (PRZ), scharakteryzowanej przez następujące, główne cechy:

- wiele rodzajów zasobów dla każdej z następujących kategorii: odnawialne, nieodnawialne i podwójnie ograniczone oraz dyskretne i ciągłe,
- wiele sposobów wykonywania każdego zadania w wypadku dyskretnych żądań zasobowych (w wypadku żądań ciągłych nie będziemy mówili o sposobach, których jest nieprzeliczalnie wiele, ale o przedziałach, z których można dowolnie przydzielać zasoby),
- wiele, w ogólności konfliktowych, kryteriów oceny uszeregowań.

Chcemy ponadto, by system wspomagał konstrukcję zarówno uszeregowań statycznych (*predictive schedules*), jak i dynamicznych (*reactive* lub *real time schedules*). Oczywiście, w miarę pełna ocena wielokryterialna będzie dotyczyła uszeregowań statycznych. Chcemy jednak wykorzystać wiedzę dotyczącą konstrukcji i oceny tych uszeregowań przy podejmowaniu decyzji odnośnie do uszeregowań dynamicznych.

Zadowalające rozwiązanie tak postawionego zadania wymaga spełnienia przez projektowany system szeregu postulatów, z których najważniejsze to:

- możliwość korzystania z wiedzy wielu ekspertów,
- możliwość stosowania różnych metod reprezentacji wiedzy i różnych mechanizmów wnioskowania,
- możliwość strukturalizacji i hierarchizacji opisu i procesu rozwiązywania problemów,
- możliwość złożonego sterowania procesem rozwiązywania problemów.

Skomentujemy krótko powyższe postulaty. Dwa pierwsze wynikają z faktu, że w złożonych sytuacjach praktycznych zwykle trudno o jednego eksperta posiadającego wystarczającą wiedzę o wszystkich aspektach rozwiązywanego problemu. Mówiąc o ekspertach mamy tu zatem na myśli ekspertów dysponujących wiedzą cząstkową, dotyczącą np. zasobów, operacji i uszeregowań. Z tymi ekspertami będą związane osobne źródła wiedzy o różnej, w ogólności, reprezentacji wiedzy i/lub mechanizmach wnioskowania. Postulat trzeci wynika z dużego stopnia strukturalizacji (czasowej, logicznej) zarówno opisu rozwiązywanych problemów, jak i samego procesu ich rozwiązywania. I wreszcie postulat czwarty, bodaj najważniejszy, dotyczy możliwości głębokiej ingerencji decydenta w proces konstruowania rozwiązań, a w szczególności możliwości sterowania tym procesem w zależności od dynamicznie wyrażanych

preferencji decydenta. Nie zakładamy bowiem preegzystencji globalnego modelu preferencji decydenta na początku procesu decyzyjnego (podejście deskryptywne), a budowę tego modelu w iteracyjnej procedurze dialogowej (podejście konstruktywne).

W kolejnych rozdziałach omówimy główne elementy koncepcji szkieletowego systemu eksperckiego realizującego wyżej opisane postulaty. Skoncentrujemy się na przedstawieniu architektury systemu (rozdział 2), metajęzyka opisu tej architektury (rozdział 3) oraz dialogowej procedury sterowania procesu konstrukcji rozwiązań (uszeregowań) za pomocą preferencji decydenta (rozdział 4).

2. Architektura systemu eksperckiego

Biorąc pod uwagę postulaty sformułowane w p.1, wybrano architekturę tablicową, jako dającą potencjalnie największe możliwości ich realizacji. Koncepcja tablicy w systemach eksperckich pojawiła się w systemie rozpoznawania mowy HEARSAY-II [2]. System o takiej architekturze składa się ze zbioru niezależnych modułów, zwanych źródłami wiedzy, w skład których wchodzą m.in bazy wiedzy i mechanizmy wnioskowania (zostaną one opisane w następnym rozdziale). Źródła wiedzy współdzielą pamięć roboczą, zwaną tablicą, przechowującą aktualny stan procesu rozwiązywania problemu. Każde źródło wiedzy śledzi zmiany zachodzące w tablicy i uaktywnia się, jeśli spełnione są skojarzone z nim warunki aktywacji. Może wówczas dokonać odpowiedniej modyfikacji tablicy. Pracą systemu kierują moduły sterujące, które określają tzw. element koncentracji uwagi, czyli następny element architektury systemu, który weźmie udział w przetwarzaniu. Elementem tym, w ogólności, może być źródło wiedzy, obiekt tablicy lub ich kombinacja. W naszym wypadku jest to obiekt tablicy, czyli wybierane są źródła wiedzy, które będą przetwarzać ten obiekt (*event driven scheduling*). W ten sposób rozwiązanie (tu: uszeregowanie) budowane jest krok po kroku, w dynamicznym, nie dającym się z góry przewidzieć ciągu wywołań źródeł wiedzy. Proces kończy się, gdy zajdzie co najmniej jeden z poniższych dwóch warunków:

- brak źródeł wiedzy, które chciałyby kontynuować proces,
- znaleziono pożądane rozwiązanie.

W systemie eksperckim kluczową rolę odgrywają dane, bez których wnioskowanie jest niemożliwe. Znajdują się one w tablicy i są reprezentowane w postaci ram. Pojedyncza rama jest pewną złożoną strukturą, której składowe (*slots*) mają różne charakterystyki. Składowe te służą do opisu określonych cech obiektu reprezentowanego przez ramę. Wyróżniamy dwa rodzaje składowych: atrybuty i metody. Pierwszy rodzaj składowych jest wykorzystywany do

reprezentowania wiedzy opisowej (deklaratywnej), drugi jest używany do reprezentowania akcji, które można na danym obiekcie wykonać.

Ramy mogą tworzyć hierarchię dziedziczenia, która jest budowana w oparciu o związki między klasami i podklasami, czyli obiektami i ich pochodnymi. Mechanizm dziedziczenia, to główna zaleta reprezentacji ramowej.

Rama opisująca klasę obiektów zawiera składowe charakteryzujące wszystkich instancji tej klasy - jest pierwowzorem każdego obiektu tej klasy. Chcąc zatem odczytać lub zapisać informacje dotyczące konkretnego obiektu, posługujemy się atrybutami i metodami zdefiniowanymi dla klasy, której obiekt ten jest reprezentantem. Rama reprezentująca klasę obiektów może być połączona związkiem typu klasa-podklasa z wieloma klasami. Tak zdefiniowane podklasy dziedziczą wszystkie składowe od klas umieszczonych wyżej w hierarchii dziedziczenia. Mogą,oczywiście,zawierać inne składowe, charakterystyczne tylko dla nich.

3. Metajęzyk opisu architektury tablicowej

Wszystkie elementy systemu (źródła wiedzy, mechanizmy wnioskowania, bazy wiedzy itp.) wymagają szczegółowej definicji. W tym celu zdefiniowano specjalizowany metajęzyk, którego podstawową cechą jest możliwość definiowania obiektów (razem z ich metodami), operowania na nich, zapis reguł i faktów, z uwzględnieniem specyfiki problemów szeregowania zadań. Zdefiniowany metajęzyk umożliwia:

- opis systemu: deklaracje stałych i zmiennych, deklaracje poziomów tablicy, klas, obiektów i ich atrybutów, deklaracje procedur i funkcji, inicjalizacja tablicy,
- definiowanie źródeł wiedzy,
- implementację standardowych funkcji i metod.

Metajęzyk, o którym mowa, w dalszej części pracy będzie nazywany EXLAN. (EXpert system LANguage). EXLAN jest oryginalnym językiem programowania obejmującym zarówno konwencjonalne instrukcje sterujące i struktury danych występujące w takich językach, jak Pascal czy C, jak i struktury danych i instrukcje specyficzne dla architektury tablicowej. Przy definiowaniu tego języka wykorzystano niektóre elementy języka opisu architektury tablicowej przedstawionego w [1]. Za pomocą EXLAN-u zdefiniowane będą pewne obiekty predefiniowane, charakterystyczne dla problemów rozdziału zasobów.

3.1. Standardowe elementy języka

Poniżej przedstawiono standardowe elementy metajęzyka. Wszystkie słowa kluczowe i literały przedstawiono drukiem wytłuszczonym.

- Litery: **A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s u v w _**
- Cyfry: **0 1 2 3 4 5 6 7 8 9**
- Znaki specjalne: + - * = < > ( ) [ ] { } . , : ; # @. Niektóre operatory i inne elementy języka zapisuje się za pomocą dwóch następujących po sobie znaków specjalnych:
  - operator przypisania :=,
  - operator relacji: <> <= >=,
  - zakres: .. .
- Słowa kluczowe: **AND, FUNCTION, ON, CONST, COMMENT IN, DO, DIV, INTERRUPT, FOR, FALSE, NONRENEWABLE, WHEN, FILE, OBJECT, FORWARD, IF, PROCEDURE, BACKWARD, IN, RENEWABLE, MIXED, THEN, SET, PARENT, OR, START, OWN, MOD, TRUE, SLOT, NOT, XOR, METHOD, GOTO, CLASS, TYPE, VAR ...**

Słowa kluczowe mogą być pisane tylko wielkimi literami. Są one integralną częścią języka i są zastrzeżone, tzn. nie mogą być zdefiniowane przez projektanta bazy wiedzy jako identyfikatory.

- Identyfikatory: Identyfikatorem (nazwą) nazywamy ciąg liter i cyfr zaczynających się od litery. Należy pamiętać, że literą może być również znak podkreślenia (_). Identyfikatory służą do oznaczenia nazw obiektów, ich atrybutów i nazw (pól i metod). Wiele identyfikatorów posiada predefiniowane znaczenie. Służą one do oznaczenia funkcji i procedur standardowych oraz predefiniowanych klas obiektów. Predefiniowane obiekty będą omówione w następnych rozdziałach.
- Funkcje standardowe: Funkcje standardowe są reprezentowane przez ich identyfikatory. Wszystkie funkcje i metody dotyczące bazy wiedzy i obiektów będą napisane w zdefiniowanym wyżej metajęzyku.. W tym celu zdefiniowano różnorodne instrukcje sterujące, podobne do tych, które występują w języku Pascal. Można tu wymienić instrukcje:
- **FOR... DO...,**
- **WHILE... DO...,**

- REPEAT... UNTIL...,
- CASE... OF...,
- IF... THEN... (pomimo podobieństwa do składni reguł instrukcja ta jest odróżniana od reguł. Nie jest ona umieszczana w bazie wiedzy. Wszystkie metody i funkcje są kompilowane w osobnym module, odseparowanym od bazy wiedzy).

3.2. Opis tablicy i obiektów

Tablica podzielona jest na dwa obszary: statyczny i dynamiczny. Pierwszy obszar jest przeznaczony na informacje, które stanowią bazę danych początkowych dla systemu. Obszar ten tworzony jest w części inicjującej pracę systemu.

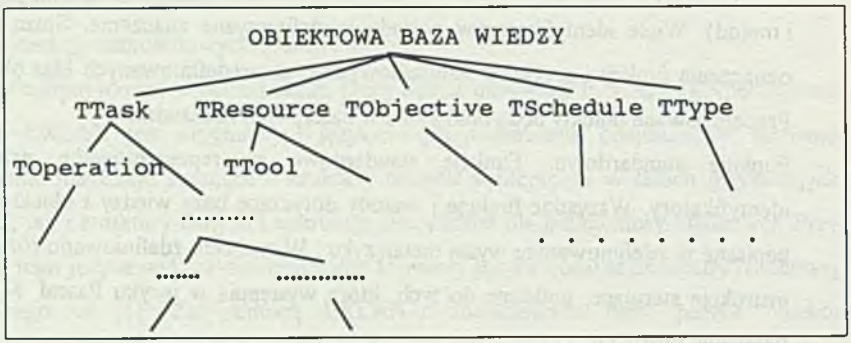
Drugi obszar stanowi właściwą tablicę systemu. Pełni rolę pamięci roboczej, a informacje w nim zawarte zmieniają się dynamicznie w trakcie pracy systemu. Deklarację obszarów w języku EXLAN przedstawiono poniżej (notacja EBNF). Deklaracja ta przyporządkowuje obszarom tablicy określone klasy obiektów.

<deklaracja_źródła> ::= STATIC[<deklaracja_obiektów>]
DYNAMIC[<deklaracja_obiektów>]

<deklaracja_obiektów> ::= <obiekt>{<obiekt>}

Położenie klasy obiektów decyduje o tym, gdzie będą tworzone ich instancje. Obiekty w obydwu obszarach są reprezentowane w niemal identycznej formie. Jedyna różnica polega na deklaracji obszaru statycznego i dynamicznego dla danego obiektu.

Za pomocą języka EXLAN w szkieletowym systemie eksperckim zdefiniowano kilka klas obiektów predefiniowanych, charakterystycznych dla problemów szeregowania zadań. Struktura tych klas obiektów została przedstawiona na rysunku 1.



Rys. 1. Struktura obiektów w systemie

Fig. 1. Object structure in the expert system

Jak widać, w systemie zdefiniowano pięć podstawowych klas obiektów nadrzędnych. Należą do nich: *TTASK, TRESOURCE, TOBJECTIVE, TSCHEDULE* i *TTYPE*. Klasy TOperation i TTool to przykładowe klasy pochodne odpowiednio klas TTask i TResource. Jako przykład klasy obiektów przedstawimy klasę TTASK.

Klasa TTASK

NazwaKlasy:TTASK;

AtrybutyInstancji:

READY_TIME: **NUMERIC**;

DUE_DATE: **NUMERIC**;

PREDECESSORS: ***SYMBOL**;

PROCESSING_MODE: ***SYMBOL**;

STATUS: **SYMBOL**;

PRIORITY:**SYMBOL**;

MetodyInstancji:

INIT(...);

SCHEDULE_TASK(...)

RESCHEDULE_TASK(...)

PREDECESSORS jest listą wszystkich poprzedników danego zadania. Jej składnia jest następująca:

<PREDECESSORS>::=(<TaskName>,...,<TaskName>)

<TaskName>::=<identyfikator>

Nazwa zadania (identyfikator) jest dowolnym łańcuchem znaków, w szczególności znakiem liczbowym.

Pole PROCESSING_MODE ma następującą składnię:

<PROCESSING_MODE>::=<Lista_sposobów_wykonywania>

<Lista_sposobów_wykonywania>::=(([<ResourceRequired>,<Amount>],
<Duration>),...,([<ResourceRequired>,<Amount>],
<Duration>))

Poszczególne sposoby wykonywania zadania są numerowane w kolejności zapisu od 0 do n-1, gdzie n jest liczbą sposobów wykonywania danego zadania.

Pole STATUS służy do zapisu stanu, w którym dane zadanie się znajduje. Podczas generowania uszeregowania zadanie może znajdować się w jednym z sześciu stanów:

- 0, zadanie nie jest szeregowalne,
- 1, zadanie jest szeregowalne,
- 2, zadanie jest właśnie wykonywane,
- 3, zadanie zostało wykonane,
- 4, zadanie spełnia tylko pierwszy z dwóch poniżej wymienionych warunków szeregowalności,
- 5, zadanie spełnia warunek szeregowalności ze względu na ograniczenia twarde, a nie spełnia tego warunku ze względu na ograniczenia miękkie.

Mówimy, że zadanie Z_i jest szeregowalne w chwili t wtedy i tylko wtedy, gdy zachodzą dwa warunki:

- wszystkie zadania poprzedzające zadanie Z_i zostały wykonane do chwili t,
- wszystkie zasoby potrzebne do wykonania zadania A_i są dostępne w chwili t.

Pole PRIORITY służy do nadawania priorytetów poszczególnym zadaniom i jest opcjonalne. Możliwe są następujące priorytety: LOW, MEDIUM, HIGH.

Metody **SCHEDULE_TASK** i **RESCHEDULE_TASK** służą odpowiednio do przydzielenia zasobów do zadania i wykonania tego zadania oraz do reszeregowania zadania, czyli powrotu do stanu, w którym system znajdował się przed przydziałem zasobów do danego zadania. Jest to konieczne, gdy uszeregowanie chwilowe lub całkowite jest niedopuszczalne. Zachodzi wtedy potrzeba wycofania się z danego ruchu. Każdy obiekt zawiera metodę **INIT**. Służy ona do nadania wartości początkowych atrybutom instancji danej klasy.

Poza wyżej wymienionymi obiektami ekspert ma możliwość definiowania własnych obiektów, zależnych od specyfiki danego problemu. Wyżej wymienione obiekty są jednak wystarczające do zdefiniowania standardowego problemu rozdziału zasobów.

3.3. Opis źródeł wiedzy

Obok tablicy źródła są drugim, najważniejszym elementem architektury tablicowej. Każde źródło wiedzy jest pewnym złożonym modułem posiadającym warunki aktywacji, własny mechanizm wnioskowania oraz wiedzę wyrażoną w postaci reguł. Definicję źródła wiedzy przedstawiono poniżej.

<deklaracja_źródła> ::=

KNOWLEDGE_SOURCE<identyfikator_źródła>

PRECONDITION<lista_warunków_aktywacji>

METHOD<typ_metody_wnioskowania>

KNOWLEDGE_BASE<nazwa_zbioru_z_bazą_wiedzy>

....

Przez warunek aktywacji rozumiemy zdarzenie, które miało miejsce w tablicy. Wystąpienie określonego zdarzenia w trakcie poszukiwania rozwiązania powoduje wywołanie wszystkich źródeł wiedzy, które zawierają dane zdarzenie w warunkach aktywacji. Do każdego źródła przydzielono pewien priorytet, określający kolejność uaktywniania źródeł w wypadku, gdy w danym momencie warunki aktywacji są spełnione dla kilku źródeł.

Źródła wiedzy mogą wykorzystywać różne metody wnioskowania, to jest wnioskowanie w przód, w tył lub też mieszane. Metody te można w odpowiedni sposób dynamicznie przełączać i krzyżować, uzyskując różne warianty postępowania [4].

W dalszym ciągu przedstawimy opis baz wiedzy i reguł.

Wiedza wykorzystywana w procesie wnioskowania źródeł wiedzy wyrażona jest w postaci reguł i przechowywana w oddzielnych plikach, zwanych bazami wiedzy. Deklarację bazy wiedzy w języku EXLAN podano poniżej.

<baza_wiedzy>::=<ciąg_reguł>

<ciąg_reguł>::= <reguła> {<reguła>}

<reguła>::= RULE <identyfikator_reguły>

IF <warunek>

THEN <akcja>

...

Odseparowanie wiedzy od mechanizmów wnioskowania jest konieczne ze względu na potrzebę uruchamiania systemu eksperckiego z różną wiedzą. Pozwala to również na wprowadzenie innej niż regułowa reprezentacji wiedzy, co ma duże znaczenie przy szkieletowych systemach eksperckich. Poszczególne bazy wiedzy mogą zawierać wiedzę pochodzącą od różnych ekspertów, zgodnie z ich udziałem w rozwiązywaniu problemu.

Jak wspomniano wcześniej, źródła wiedzy w trakcie swego działania dokonują zmian w tablicy. Jest to możliwe dzięki akcjom reguł, które modyfikują atrybuty określonych obiektów.

Reguły są głównym elementem bazy wiedzy. Są one skonstruowane jako akcje typu IF...THEN... Warunek reguły jest normalnym wyrażeniem logicznym lub kilkoma takimi wyrażeniami połączonymi słowami kluczowymi AND/OR. Warunki w naszym systemie dotyczą zazwyczaj takich aspektów, jak:

- dostępność zasobów,

- stany poszczególnych elementów systemu,
- stany wszystkich obiektów.

Akcja reguły jest listą instrukcji oddzielonych słowem kluczowym AND. W systemie istnieje kilka standardowych akcji. Są to:

- akcje odpowiedzialne za wywoływanie odpowiednich operacji w celu znalezienia rozwiązania,
- akcje określające cechy obiektów (np. zmiana sposobu wykonywania zadania),
- akcje odpowiedzialne za wykrywanie i usuwanie błędów.

Reguły są podzielone na kilka klas, w zależności od problemu, który dane reguły rozwiązują. Poniżej opiszemy krótko kilka z tych klas.

Klasa 1: Reguły, które pozwalają na wprowadzenie dodatkowych ograniczeń i przetransformowanie ich do ograniczeń standardowych. Dodatkowe ograniczenia mogą wynikać ze specyfiki danego problemu. Będą to takie ograniczenia, których nie można przedstawić za pomocą tylko i wyłącznie obiektów i ich atrybutów. Poniżej przedstawiono przykładowe ograniczenia, które można przedstawić za pomocą reguł z klasy 1:

- grupowanie zadań według różnych cech wspólnych (wymagane zasoby, sposoby wykonywania, momenty gotowości itp.),
- nadawanie priorytetów poszczególnym zadaniom,
- ograniczenia dla utworzonych klas zadań (np. zadania z klasy A muszą być wykonane przed zadaniami z klasy B),
- rezerwacja zasobów dla pewnych zadań o wysokim priorytecie,
- określenie minimalnego (maksymalnego) zadanego odstępu czasowego pomiędzy wykonaniem dwóch (lub więcej) zadań,
- dodatkowe ograniczenia kolejnościowe (wynikające np. z przybywania zadań o wyższym priorytecie).

Klasa 2: Reguły tej klasy służą do wyboru najodpowiedniejszego algorytmu do rozwiązania rozważanego problemu. Zakładamy, że istnieje baza algorytmów przystosowanych do różnych klas PRZ. Reguły z tej klasy w celu wybrania odpowiedniego algorytmu sprawdzają parametry problemu. Mogą to być ograniczenia kolejnościowe, charakterystyki zadań itp.

Klasa 3: Reguły odpowiedzialne za sterowanie algorytmem wybranym przez reguły z klasy 2. Jak wspomniano wcześniej, algorytmy będą odwoływały się do baz wiedzy (poprzez uruchomienie modułu wnioskowania) w celu sprawdzania, czy nie zawierają one akcji dotyczących danego momentu generowania rozwiązania. Takie bieżące sprawdzanie procesu

szeregowania pozwala na podejmowanie odpowiednich kroków przez algorytm już w trakcie generowania rozwiązania. Odpowiednio sformułowane reguły pozwolą na definiowanie w bazach wiedzy warunków, które musi spełniać uszeregowanie, aby było dopuszczalne. Pewne kombinacje lub kroki mogą być zabronione. Na przykład lista tabu w algorytmie tabu może być zbiorem wyrażeń logicznych, określających zabronione konfiguracje systemu. Można w ten sposób oddzielić listę tabu od algorytmu, co uczyni ten algorytm bardziej uniwersalnym. Jest to, oczywiście, jedno z wielu możliwych rozwiązań.

Klasa 4: W regułach z tej klasy ekspert może umieścić sposoby relaksacji ograniczeń miękkich. Tylko poprzez zastosowanie wiedzy eksperta pozwalającej na wprowadzenie ograniczeń nienumerycznych możemy wprowadzić takie rozróżnienie pomiędzy ograniczeniami miękkimi i twardymi. Mówiąc o ograniczeniach twardych (ograniczenia kolejnościowe, dostępność zasobów - zależnie od sytuacji), mamy na myśli ograniczenia, które reprezentują warunki konieczne dopuszczalności uszeregowań. Natomiast ograniczenia miękkie to takie, które mogą zostać przekroczone (np. terminy zakończenia wykonywania zadań).

Klasa 5: Sprawdzenie jakości uszeregowań dopuszczalnych wygenerowanych przez algorytm jest zadaniem reguł z klasy 5. Sprawdzając konkretne wartości funkcji kryterialnych można ocenić poszczególne uszeregowania i poklasyfikować je na odpowiednie grupy. Przez grupy rozwiązań należy rozumieć uszeregowania określane np. jako bardzo dobre, dobre, złe. W ten sposób ograniczymy zbiór uszeregowań, który system przedstawi decydentowi do oceny.

Klasa 6: służy do reprezentacji wiedzy wykorzystywanej w trybie on-line. Reguły te określają reakcje systemu na takie zdarzenia, jak awarie wszelkiego rodzaju zasobów (awarie maszyn, choroby pracowników) lub pojawienie się zadań priorytetowych, które muszą być wykonane natychmiast po zgłoszeniu. Zmiany takie powinny być uwzględnione w uszeregowaniu uzyskanym przez moduł off-line.

4. Sterowanie za pomocą preferencji decydenta

W ogólności, na drodze zastosowania szeregu algorytmów przybliżonych (różnych dla różnych kryteriów), zostaje wygenerowany zbiór uszeregowań dopuszczalnych. Spośród tych uszeregowań do dalszych rozważań bierzemy zbiór uszeregowań niezdominowanych N, dla którego stosujemy interaktywną procedurę złożoną z trzech kroków.

Krok początkowy

Dla rozwiązań niezdominowanych tworzy się macierz wypłat Z (*pay-off table*) o

wymiarach kxk, której elementy z_{ij} oznaczają wartość kryterium i dla najlepszego rozwiązania z punktu widzenia kryterium j (i,j=1,...k, gdzie wartość k jest liczbą rozpatrywanych kryteriów). Przekątna tej macierzy definiuje zatem rozwiązanie idealne, w ogólności nierealizowalne w ramach przyjętych ograniczeń.

• *Krok obliczeniowy*

Znajdujemy uszeregowanie najbliższe idealnemu w sensie wybranej funkcji skalaryzującej, którą jest rozszerzona, ważona norma Czebyszewa.

• *Krok decyzyjny*

Uszeregowanie znalezione w kroku obliczeniowym przedstawiane jest decydentowi do oceny i jeśli ten uzna je za satysfakcjonujące, to procedura się zatrzymuje. W przeciwnym razie decydent określa kryterium, na którym decyduje się stracić o określoną wartość Δz_i, żeby zyskać na innych kryteriach. Wartość Δz_i jest przeliczana następnie na wartość fukcji kary, dodawana do funkcji skalaryzującej i procedura wraca do kroku obliczeniowego.

Funkcja skalaryzująca wykorzystywana w kroku obliczeniowym ma postać:

$$s(z,\pi) = \max\{\pi_i(z_i - u_i)\} + \varepsilon \sum \pi_i(z_i - u_i$$

gdzie $\pi_i = \Phi / \sum \Phi_i,\ \Phi_i = (U_i - u_i)/U_i, i = 1, 2, ..., k,$

$$\varepsilon = 0.001, u_i = \min\{z_i\}, U_i = \max\{z_i\}, i = 1, 2, ..., k.$$

Wartość Δz_i określona w kroku decyzyjnym jest przeliczana na wartość funkcji kary

$$b_j(\Delta z, \pi) = \tfrac{1}{2} \sum (\max\{0, \pi_i(z_i - z_i^{j-1} - \Delta z_i)\})^2,$$

gdzie j jest indeksem iteracji, a $z^{j-1} \in N$ uszeregowaniem znalezionym w iteracji j-1. Współczynnik π_i przymuje przy tym wartość 0 w iteracji j.

Zauważmy, że powyższa procedura jest w istocie taka sama,jak w systemie MPS [3]. Istotna różnica polega na tym, ze algorytmy generujące uszeregowania dopouszczalne odwołują się do baz wiedzy (por. reguły z klasy 3).

LITERATURA

[1] Michalski, A.: Język opisu architektury tablicowej systemu eksperckiego, Materiały II Krajowej Konferencji Naukowej "Inżynieria Wiedzy i Systemy Ekspertowe", Wrocław 1993.

[2] Rich, E.: Artificial Intelligence, McGraw-Hill Company, 1983.

[3] Słowiński, R, Soniewicki, B., Węglarz, J.: MPS - Komputerowy system wspomagania decyzji dla wielokryterialnego rozdziału zasobów, Wydawnictwa Politechniki Poznańskiej, Archiwum Robotyki i Automatyki, t.XXXVI, z.1, Poznań 1991.

[4] Węglarz, J., Nabrzyski, J.: On a Knowledge-based system for multiobjective project scheduling (przygotowano do publikacji).

Recenzent: Prof. dr inż. Henryk Kowalowski

Wpłynęło do Redakcji do 30.04.1994 r.

Abstract

We present a knowledge-based multiobjective project scheduling system which handles quite a general class of nonpreemptive scheduling problems. This class is characterized by the following facts:

- renewable, nonrenewable and doubly-constrained resources of many types,
- multiple performing modes of activities,
- multiple project performance measures.

The problem is how to schedule precedence and resource-constrained activities of a project in order to find the best compromise solution or a set of "good" compromise solutions. Resource constraints can concern: the amount available at every moment, and the consumption over a given time period. The set of project performance measures is composed of time objectives (e.g. project duration, mean weighted flow time, mean weighted lateness) and cost objectives (e.g. weighted resource consumption, net present value).

Knowledge concerning different aspects of technology, resources and schedules is stored in several separated knowledge sources that communicate through an array. The array is a kind of working memory of global data structure in which the current state of the solution process of the scheduling problem under consideration is stored. Each knowledge source represents an expert who observes all changes in the array and simulates the system to find a better solution. Each expert defines his knowledge using the specialized object-oriented meta-language.

The knowledge is represented by production rules and frames. Rules operate on objects which instances are stored in the array. In project scheduling problems objects correspond to the activities, resources and schedules. Some objects are predefined, e.g. TTask, TResource, TSchedule. Rules are divided into some classes depending on the problem, which is handled by the particular rule.

The scheduling problem is thought as an interactive search process driven by the decision-maker's preferences.