ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1990
Seria: AUTOMATYKA z.102 Nr kol.1084

Eugeniusz Nowicki
Czesław Smutnicki

Politechnika Wrocławska
Instytut Cybernetyki Technicznej

# SYSTEM WSPOMAGANIA DECYZJI DLA POTRZEB HARMONOGRAMOWANIA PRODUKCJI

Streszczenie. W pracy rozważa się problem szeregowania zadań przy ograniczeniach zasobowych i czasowych z nieregularnym wskaźnikiem jakości uszeregowania. Wskaźnik ten jest liniową kombinacją wskaźników: maksymalna kara za przyśpieszenie, maksymalna kara za spóźnienie, sumaryczna kara za przyśpieszenie oraz sumaryczna kara za spóźnienie terminów wykonania zadań. Omawiany problem jest tematem międzynarodowego programu badawczego "International Comparative Study in DSS Developement" koordynowanego przez IIASA. W pracy przedstawiono algorytmy aproksymacyjne oraz omówiono podstawowe elementy realizowanego systemu wspomagania decyzji.

## 1. Wstęp

Problemy szeregowania i rozdziału środków zasobowych występujące w procesach produkcyjnych charakteryzują się zwykle dużą złożonością wynikającą z różnorodnych ograniczeń (czasowych, zasobowych) oraz z rozmiaru zagadnienia. W związku z tym stosowanie algorytmów wyznaczających rozwiązanie optymalne wymaga bardzo dużych nakładów obliczeniowych, zwykle nie do przyjęcia w praktyce. Co więcej, stosowanie algorytmów przybliżonych, w sytuacjach gdy problem wyznaczenia jakiegokolwiek rozwiązania dopuszczalnego jest nietrywialny, napotyka na podobne trudności natury obliczeniowej.

W związku z powyższym, w pracy [2], została przedstawiona propozycja powołania międzynarodowego projektu badawczego, którego celem byłoby zbudowawanie prototypowego systemu wspomagania decyzji w zakresie szeregowania zadań przy ograniczeniach czasowych i zasobowych. W systemie tym decydent w oparciu o swoją wiedzę, doświadczenie i odpowiednie wyniki obliczeń komputerowych podejmuje decyzje o ostatecznym kształcie harmonogramu. Jako punkt wyjścia do rozważań przyjęto modelowe zagadnienie o takim stopniu ogólności, który jest do zaakceptowania w bardzo wielu praktycznych sytuacjach. Projekt badawczy pod nazwą "International Comparative Study in DSS Developement" został zainicjowany w r.1987 i jest koordynowany przez IIASA.

W niniejszej pracy, będącej kontynuacją prac [8],[10] w ramach tego projektu, prezentujemy system opracowany przez autorów dla komputerów klasy IBM PC. System ten wspiera decydenta w zakresie następujących działań:

sformułowanie problemu oraz przygotowanie danych, kontrola i wykrywanie niezgodności danych, automatyczne generowanie rozwiązania, "ręczne" tworzenie i poprawianie rozwiązania, sprawdzanie dopuszczalności rozwiązania i usuwanie ewentualnych niedopuszczalności.

Praca była finansowana przez RP.I.02 "Teoria sterowania i optymalizacji ciągłych układów dynamicznych i procesów dyskretnych".

## 2. Model matematyczny

Modelowe zagadnienie rozważane we wspomnianym Projekcie IIASA zostało opisane w pracy [2] i posiada następujące sformułowanie matematyczne.

Dany jest zbiór n niepodzielnych zadań $J = \langle 1,2,\ldots,n\rangle$ oraz zbiór m zasobów $M = \langle 1,2,\ldots,m\rangle$. Zasoby są niepodzielne, odnawialne, ilość każdego zasobu jest równa jednej jednostce. Dla każdego zadania $j\in J$ określono:
(i) termin gotowości $r_j$, pożądany termin zakończenia $d_j$ oraz najpóźniejszy dopuszczalny termin zakończenia $D_j$, $0\le r_j\le d_j\le D_j$,
(ii) zbiór alternatywnych sposobów wykonywania $\mathcal{M}_j\subseteq 2^M$; każdy sposób $M_j\in\mathcal{M}_j$ jest zdefiniowany przez zasoby zaangażowane w realizację tego zadania,
(iii) czas wykonywania $p_j(M_j)\ge 0$ sposobem $M_j$, $M_j\in\mathcal{M}_j$,
(iv) wagę przyśpieszenia $v_j\ge 0$ oraz wagę spóźnienia $w_j\ge 0$.
Dodatkowo dana jest relacja $R\subset J\times J$ taka, że graf $(J,R)$ jest acykliczny. Dla każdej pary $(i,j)\in R$ określono odpowiednio dolne $\alpha_{ij}$ i górne $\beta_{ij}$ ograniczenie czasu oczekiwania pomiędzy terminem zakończenia zadania i a terminem rozpoczęcia zadania j, $\alpha_{ij}\le\beta_{ij}$; dopuszcza się ujemne wartości $\alpha_{ij}$, $\beta_{ij}$.

Uszeregowanie zadań określa zestaw par $(M_j,S_j)$, $j\in J$, gdzie $M_j\in\mathcal{M}_j$ – sposób wykonywania, $S_j$ – termin rozpoczęcia wykonywania zadania j. Uszeregowanie $(M_j,S_j)$, $j\in J$ jest dopuszczalne, jeżeli spełnia warunki:

$$r_j \le S_j, \qquad C_j \le D_j, \quad j\in J, \tag{1}$$

$$\alpha_{ij} \le S_j - C_i \le \beta_{ij}, \quad (i,j)\in R, \tag{2}$$

$$(M_i \cap M_j \ne \emptyset) \Rightarrow ((C_i \le S_j) \vee (C_j \le S_i)), \quad i,j\in J, \tag{3}$$

gdzie

$$C_j = S_j + p_j(M_j). \tag{4}$$

Zadanie j w uszeregowaniu $(M_j,S_j)$ ma przyśpieszenie $E_j=\max(0,d_j-C_j)$ oraz spóźnienie $T_j=\max(0,C_j-d_j)$ względem żądanego terminu zakończenia $d_j$, $j\in J$.

Poszukiwane jest uszeregowanie dopuszczalne $(M_j^*,S_j^*)$, $j\in J$ minimalizujące

$$K = v_{max}\max_{j\in J} v_jE_j + w_{max}\max_{j\in J} w_jT_j + v_{sum}\sum_{j\in J} v_jE_j + w_{sum}\sum_{j\in J} w_jT_j \tag{5}$$

gdzie $v_{max}$, $w_{max}$, $v_{sum}$, $w_{sum}$ zadane wagi.

Powyższy model umożliwia opisanie obszernej klasy praktycznych sytuacji produkcyjnych, w tym m.in. okresowa dostępność zasobów, patrz np. [6].

W rozważanym problemie samo wyznaczenie jakiegokolwiek rozwiązania dopuszczalnego jest już problemem silnie NP-zupełnym; jedynie gdy $D_j=\infty$, $j\in J$, $\beta_{ij}=\infty$, $(i,j)\in R$, można łatwo wygenerować takie rozwiązanie. Główną trudnością występująca jednak w trakcie rozwiązywania problemu jest przede wszystkim nieregularność funkcji celu (funkcja celu jest regularna, jeżeli jest niemalejąca ze względu na każdą zmienną przy ustalonych pozostałych). Uwzględniając definicje $E_j$, $T_j$, kryterium (5) można przedstawić następująco

$$K = v_{max} \max_{j\in J} \{v_j \max(0, d_j - C_j)\} + w_{max} \max_{j\in J} \{w_j \max(0, C_j - d_j)\} +$$

$$v_{sum} \sum_{j\in J} v_j \max(0, d_j - C_j) + w_{sum} \sum_{j\in J} w_j \max(0, C_j - d_j). \qquad (5')$$

Bezpośrednio z (5') wynika, że kryterium K, potraktowane jako funkcja tylko jednego wybranego $C_k$ (pozostałe $C_j$ oraz wszystkie $M_j$ są ustalone) jest funkcją nierosnącą dla $C_k \le d_k$ oraz niemalejącą dla $C_k \ge d_k$, zatem kryterium (5) jest nieregularne. W przypadku regularnych funkcji celu uszeregowanie optymalne należy do klasy tzw. uszeregowań "zwartych", tzn. zadania są umieszczone maksymalnie w lewo na osi czasu. Dodatkowo, uszeregowanie to jest stosunkowo łatwo generować i dlatego główna idea algorytmów heurystycznych (dla kryteriów regularnych) polega na konstrukcji pewnego uszeregowania zwartego wykorzystując odpowiednie reguły priorytetowe, patrz np. [5], [8], [9], [11]. W przypadku nieregularnych funkcji celu, uszeregowanie optymalne zwykle nie należy do klasy uszeregowań zwartych. Odnośne wyniki literaturowe są szczątkowe i ograniczają się do stosunkowo prostych zagadnień szeregowania (np. zagadnienia jednomaszynowe, przepływowe bez dodatkowych zasobów [1], [4], [7]). Biorąc powyższe pod uwagę, wydaje się, iż jedynym sensownym podejściem do rozwiązania omawianego problemu jest połączenie różnych algorytmów heurystycznych z wiedzą i doświadczeniem użytkownika w dialogowym systemie podejmowania decyzji.

## 3. Algorytmy heurystyczne

W pracy [6] zaproponowano ogólny schemat algorytmu heurystycznego dla problemu (5). Schemat ten zawiera dwie fazy: (1) wyznaczenie uszeregowania zwartego przy zastosowaniu pewnej reguły priorytetowej w połączeniu z dynamiczną analizą dostępności zasobów; w danej chwili czasowej analizuje się kolejno zasoby 1,2,...,m pod kątem możliwości ich wykorzystania, (2) poprawa uszeregowania poprzez rozwiązanie odpowiedniego zadania PL dostarczającego uszeregowania niekoniecznie zwartego. W dalszym ciągu prac badano inny wariant fazy pierwszej. Polega on na tym, że w danej chwili czasowej analizuje się zadania pod kątem możliwości ich realizacji. W wyniku badań

porównawczych stwierdzono, że wariant ten jest korzystniejszy. Odpowiedni algorytm opisano poniżej.

Algorytm, dla ustalonej chwili czasowej t, znajduje zbiór X zadań i sposobów, którymi zadania te mogą być wykonywane, poczynając od chwili t. Wszystkie zadania będące poprzednikami zadań ze zbioru X muszą już być wykonane. Następnie stosując pewną regułę priorytetową wybieramy, spośród elementów zbioru X, zadanie oraz sposób jego realizacji. Dopuszcza się taką modyfikację reguł priorytetowych, która w danej chwili nie wybiera żadnego zadania z X. Wybrane zadanie jest szeregowane (wybranym sposobem), zbiór X jest odpowiednio modyfikowany (pewne sposoby wykonywania pozostałych zadań z X mogą nie być teraz realizowalne ze względu na niedostępność zasobów) oraz proces szeregowania jest kontynuowany. Jeżeli zbiór X jest pusty lub reguła priorytetowa nie wybrała żadnego zadania, to przechodzimy do najbliższej chwili czasowej, w której kończy się jedno z realizowanych zadań lub pewne zadanie staje się gotowe do wykonywania ze względu np. na jego termin gotowości. Szczegółowy schemat algorytmu przedstawiono poniżej. W schemacie tym przyjęto następujące oznaczenia:

- $t$ – bieżąca chwila czasowa,
- $t_i$ – chwila czasowa, poczynając od której zasób $i$ jest nie wykorzystywany, $i \in M$,
- $c_j$ – liczba poprzedników zadania $j$, które nie zostały uszeregowane, $j \in J$,
- $U$ – zbiór zadań uszeregowanych,
- $Z$ – zbiór zadań nieuszeregowanych, których wszystkie poprzedniki zostały uszeregowane; $Z \subseteq J-U$,
- $S_j$ – najwcześniejszy możliwy termin rozpoczęcia zadania $j$, $j \in J$
- $X$ – zbiór zadań i sposobów, którymi zadania te mogą być wykonywane, poczynając od bieżącej chwili czasowej t,
- $k$ – aktualnie szeregowane zadanie,
- $A'$ – wybrany sposób wykonywania zadania $k$,
- $(M'_j, S'_j), j \in J$ – uszeregowanie otrzymane algorytmem.

W algorytmie zakłada się, że dla każdej pary $(i,j) \in R$ zachodzi naturalny warunek $\min_{A \in M_i} p_i(A) > -a_{ij}$.

Algorytm fazy 1.

Krok 0. (* inicjalizacja *).
Podstaw $t_i := 0$, $i \in M$, $c_j := \mathrm{card}\{i: (i,j) \in R\}$, $S_j := r_j$, $j \in J$. Wyznacz $Z := \{j \in J: c_j = 0\}$ oraz podstaw $t := \min\{S_j: j \in Z\}$, $U := \emptyset$.

Krok 1. (* wyznaczenie zbioru X *)
Wyznacz $X := \{(j,A): j \in Z, S_j \leq t, A \in M_j, t_i \leq t, i \in A\}$.

Krok 2. (* uszeregowanie zadania k sposobem $M'_k$ w chwili $S'_k = t$ *)
Jeżeli $X = \emptyset$, to przejdź do kroku 3.

Korzystając z pewnej reguły priorytetowej wybierz $(k,A')\in X$.
Jeżeli para nie została wybrana, to przejdź do kroku 3.
Podstaw $S'_k := t$, $M'_k := A'$, $Z := Z-(k)$, $U := U\cup(k)$;
$S_j := \max(S_j, t+p_k(M'_k)+\alpha_{kj})$, $(k,j)\in R$; $t_i := t+p_k(M'_k)$, $i\in M'_k$;
$c_j := c_j-1$, $(k,j)\in R$; $Z := Z\cup(j\in J: (k,j)\in R,\ c_j=0)$;
$X := X-((j,A)\in X: j=k)$, $X := X-((j,A)\in X: M'_k\cap A\neq\emptyset)$;
Przejdź do kroku 2.

Krok 3. (* zmiana chwili analizy zdarzeń *)
Jeżeli $Z=\emptyset$, to stop.
Podstaw $t := \max(\min(t_i: i\in M, t_i>t), \min(S_j: j\in Z))$ i przejdź do kroku 1.

W celu poprawności działania powyższego algorytmu (tzn. uszeregowania wszystkich zadań) reguła priorytetowa powinna spełniać np. następujący warunek: jeżeli poczynając od aktualnej chwili czasowej wszystkie zasoby są dostępne, to zadanie ze zbioru X musi być wybrane. Zauważmy też, że algorytm może generować uszeregowania niedopuszczalne ze względu na ograniczenia z (1)-(2) postaci $C_j \le D_j$, $j\in J$, $S_j - C_i \le \beta_{ij}$, $(i,j)\in R$. Wszystkie pozostałe ograniczenia są spełnione. W pracach [2] i [6] podano sposób modelowania okresowej dostępności zasobów poprzez wprowadzenie sztucznych zadań. Jednakże w sformułowanym algorytmie istnieje prostszy sposób uwzględnienia tego ograniczenia. W tym celu należy zestaw warunków ograniczających w definicji zbioru X w kroku 1 rozszerzyć o warunek "wszystkie żądane zasoby sposobu A dla zadania j są dostępne w przedziale $[t,t+p_j(A)]$" oraz zmiana chwil analizy zdarzeń w kroku 3 powinna uwzględniać momenty czasowe dostępności zasobów. W wyniku modyfikacji może się zdarzyć, że pewne zadania nie zostaną uszeregowane, ponieważ wymagane zasoby są niedostępne lub zbyt krótko dostępne. W konsekwencji warunek stopu w kroku 3 nigdy nie zatrzyma algorytmu. Zatem, oprócz wspomnianych już modyfikacji, należy przewidzieć odpowiednią modyfikację warunku stopu. W przypadku gdy pewne zadania nie zostaną uszeregowane, proponujemy naruszyć arbitralnie pewne warunki ograniczające (np. przedłużyć ostatni przedział dostępności wszystkich zasobów do nieskończoności) i ponownie zastosować algorytm. Oczywiście tak otrzymane rozwiązanie jest niedopuszczalne i nie może być poddawane działaniom fazy (2) ogólnego algorytmu heurystycznego.

W kroku 2 algorytmu może być zastosowana dowolna reguła priorytetowa spełniająca wspomniany wcześniej warunek. W pracy [6] zaproponowano szereg reguł priorytetowych nazwanych odpowiednio ML, MG, SP, LF, DF oraz ASP. Wszystkie wymienione reguły mogą być zastosowane również w przypadku omawianego algorytmu. W dalszym ciągu proponujemy jeszcze jedną regułę (ASP') opartą na najkrótszych czasach wykonywania zadań; jest to pewna modyfikacja reguły ASP. Przyjmijmy następujące oznaczenia: $I=(j\in J: (j,A)\in X)$, $p_j^* = p_j(M_j^*) = \min(p_j(A): (j,A)\in X)$, $p_j^{**} = p_j(M_j^{**}) = \min(p_j(A): A\in M_j)$, $j\in I$. Niech $t_j^o$ będzie naj-

wcześniejszym terminem udostępnienia wszystkich zasobów ze zbioru $M_j^{**}$: $t_j^o \geq t$. Zbiór I określa zadania, które mogą być wykonywane w bieżącej chwili t, a $p_j^*$ i $p_j^{**}$ określają odpowiednio najkrótszy czas wykonywania zadania j sposobami, które mogą być realizowane od chwili t oraz najkrótszy czas wykonywania zadania j sposobami ze zbioru $M_j$. Oczywiście, zachodzi $p_j^{**} \leq p_j^*$.

ASP' (zadania z najkrótszym sposobem wykonywania najpierw):
Jeżeli dla każdego $j \in I$ zachodzi $t+p_j^* > t_j^o+p_j^{**}$, to nic nie wybieraj.
W przeciwnym przypadku wybierz zadanie $k \in I$, spełniające warunek $p_k^*-p_k^{**} = \min\{p_j^*-p_j^{**}: j \in I\}$ oraz sposób jego wykonywania $A' = M_k^*$.

Prowadzono także badania związane z realizacją fazy drugiej ogólnego algorytmu heurystycznego opisanego w [6]. Zaproponowane tam zadanie PL ma rozmiar $(3n+2) \times (5n+|R|+|R^o|)$, gdzie $R^o = \{(i,j): M_i' \cap M_j' \neq \emptyset, S_i' < S_j', i,j \in J\}$. Mimo zastosowania pewnych metod specjalizowanych, zadanie to okazało się zbyt czaso- i pamięcio-chłonne dla realizowanego systemu, już dla $n \geq 50$. W konsekwencji dla dużych problemów zaproponowano uproszczoną realizację fazy drugiej algorytmu. Polega ona na tym, że wszystkie zadania $j, j \in J$ przesuwane są na osi czasu o jednakową wielkość, tak by zachować dopuszczalność oraz minimalizować kryterium (5). Poniżej przedstawimy dokładny opis techniki postępowania, zakładając okresową dostępność zasobów.

Niech $[a_{il}, b_{il}]$, $l = 1, \ldots, l_i$, oznacza przedziały dostępności zasobu i, gdzie $l_i$ jest liczbą tych przedziałów, $i \in M$. Dalej, niech $e_{ij}$ oznacza numer przedziału dostępności zasobu $i \in M_j'$, w którym wykonywane jest zadanie $j, j \in J$. Zachodzi $1 \leq e_{ij} \leq l_i$ oraz

$$a_{ie_{ij}} \leq S_j', \quad C_j' \leq b_{ie_{ij}}, \quad i \in M_j' \tag{6}$$

gdzie

$$C_j' = S_j' + p_j(M_j'), \quad j \in J.$$

Zakładając, że x określa szukane przesunięcie na osi czasu, z (6) oraz (1) dostaniemy

$$a_{ie_{ij}} \leq S_j' + x, \quad C_j' + x \leq b_{ie_{ij}}, \quad i \in M_j', \quad j \in J$$

oraz

$$r_j \leq S_j' + x, \quad C_j' + x \leq D_j, \quad j \in J.$$

Stąd wynika, że x może być wybierane z przedziału $[a_m, b_m]$, gdzie

$$a_m = \max_{j \in J} [\max(r_j, \max_{i \in M_j'} (a_{ie_{ij}})) - S_j'],$$

$$b_m = \min_{j \in J} [\min(D_j, \min_{i \in M_j'} (b_{ie_{ij}})) - C_j'].$$

Zauważmy, że jeżeli uszeregowanie $(M_j', S_j'), j \in J$ nie spełnia ograniczeń $C_j' \leq D_j$,

to możliwe jest $a_m > b_m$. Zatem dalszy ciąg tej fazy realizuje się tylko wtedy, gdy $a_m < b_m$. Niech

$$K(x) = v_{max} \max_{j \in J} \{v_j \max(0, d_j - C'_j - x)\} + w_{max} \max_{j \in J} \{w_j \max(0, C'_j + x - d_j)\} +$$

$$v_{sum} \sum_{j \in J} v_j \max(0, d_j - C'_j - x) + w_{sum} \sum_{j \in J} w_j \max(0, C'_j + x - d_j).$$

Można sprawdzić, że funkcja $K(x)$, $x \in [a_m, b_m]$ jest odcinkami liniowa i wypukła. Ostatecznie uproszczona realizacja fazy drugiej polega na wyznaczeniu $a_m$, $b_m$ oraz $x^* \in [a_m, b_m]$ takiego, że $K(x^*) = \min(K(x): x \in [a_m, b_m])$. W wyniku otrzymujemy nowe uszeregowanie $(M'_j, S''_j)$, $j \in J$, gdzie $S''_j = S'_j + x^*$. Warto zauważyć, że otrzymane uszeregowanie spełnia wszystkie te ograniczenia, które spełniało uszeregowanie poprzednie. Co więcej, jeżeli ograniczenia $C'_j \leq D_j$, $j \in J$ nie były spełnione oraz $a_m < b_m$, to zachodzi $C''_j = S''_j + p_j(M'_j) \leq D_j$, $j \in J$. Dla wyznaczenia $x^*$ można zastosować np. metodę złotego podziału. Oczywiście proponowany przebieg fazy drugiej może być realizowany, zakładając, że przesunięciu na osi czasu podlega tylko pewien podzbiór zadań zbioru J; wtedy wielkości $a_m$, $b_m$ należy wyznaczać w odpowiednio inny sposób. W ogólnym przypadku możliwe jest wielokrotne przesuwanie różnych podzbiorów zadań.

Zaproponowany ogólny algorytm heurystyczny był poddany analizie eksperymentalnej w ograniczonym zakresie. Między innymi badano skuteczność różnych reguł priorytetowych na losowo generowanych przykładach. W wyniku badań wybrano zestaw reguł opisanych w pracy [6] oraz powyżej. Ograniczony zakres badań wynikał z następujących powodów:

- nie są znane w literaturze algorytmy rozwiązywania problemów o takim stopniu ogólności, z nieregularną funkcją celu,
- wszelkie możliwe dolne ograniczenia wartości funkcji celu (wyliczane w rozsądnym czasie) są bardzo niedokładne, co powoduje, że nie można porównać otrzymanego rozwiązania z rozwiązaniem optymalnym.

Ostateczna ocena zaproponowanego algorytmu będzie możliwa dopiero po zakończeniu wspomnianego programu badawczego IIASA, na drodze porównania z wynikami innych zespołów badawczych.

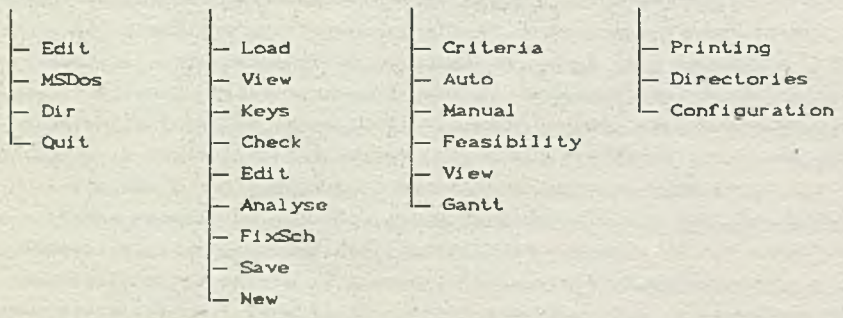
Problem korygowania niedopuszczalności uszeregowań otrzymywanych w wyniku fazy pierwszej (oraz nie usuniętych w fazie drugiej) ogólnego algorytmu heurystycznego został pozostawiony użytkownikowi. Jednocześnie w systemie zaprojektowano funkcje umożliwiające mu szybkie wykrywanie niedopuszczalności oraz "ręczną" modyfikację uszeregowania przy wydatnej pomocy systemu.

## 4. System wspomagania decyzji

System został zaprojektowany do wykorzystania na komputerach klasy IBM PC. Funkcje systemu zrealizowano zgodnie z logicznym ciągiem działań zmie-

rzających do sformułowania i rozwiązania problemu. Zgodnie z tą zasadą wszystkie funkcje podzielono na 4 grupy, rys.1:

- funkcje związane z działaniami na plikach dyskowych i komunikacją z systemem operacyjnym (File),
- funkcje związane ze sformułowaniem problemu, przygotowaniem danych wejściowych oraz ze wstępną analizą problemu (Data),
- funkcje związane z procesem rozwiązywania problemu (Solver),
- funkcje dodatkowe związane z trybami pracy systemu, konfiguracją sprzętowo-programową, itp. (Options).




Rys. 1. Poziom pierwszy menu systemu.
Fig. 1. First level of the system menu

Grupa (File) realizuje następujące funkcje:

- (Edit) edycja plików tekstowych niezbędna przy tworzeniu pliku tekstowego zawierającego dane wejściowe w wymaganej standardowej postaci; system automatycznie przechodzi do tego trybu pracy, jeżeli próba czytania standardowego pliku danych zakończyła się niepowodzeniem,
- (MSDos) rozszerzone działania w zakresie funkcji systemu operacyjnego,
- (Dir) udostępnienie danych zawartych w katalogach dysku,
- (Quit) wyjście z systemu.

Grupa (Data) jest podstawową w zakresie operacji na danych wejściowych i realizuje następujące funkcje:

- (Load) wczytanie danych wejściowych problemu oraz zestawu uszeregowań (jeżeli nimi dysponujemy),
- (View) przeglądanie danych wejściowych; dane te są umieszczone w 11 standardowych tablicach: projekty, zadania, poprzedzania, zasoby, dostępności zasobów, zbiory zasobowe, sposoby wykonywania zadań, kryteria oceny zadań, kryteria oceny projektów, kryteria globalne, uszeregowania,
- (Keys) wybór pola kluczowego do uporządkowania danych zawartych w posz-

czególnych tablicach,
- (Check) sprawdzenie zgodności wczytanych danych; system po wykryciu niezgodności sygnalizuje rodzaj i miejsce wystąpienia błędu oraz automatycznie przechodzi do funkcji (Edit) w grupie funkcji (Data),
- (Edit) edycja danych zawartych w poszczególnych tablicach, wykorzystywana do modyfikacji danych w procesie rozwiązywania problemu,
- (Analyse) wstępna analiza danych,
- (FixSch) wybór aktualnego uszeregowania spośród dostępnych z tablicy uszeregowań,
- (Save) zapis na dysku danych problemu oraz uszeregowania (w tym także danych zmodyfikowanych w trakcie rozwiązywania problemu),
- (New) kasowanie niepotrzebnych danych lub uszeregowań.

Grupa (Solver) realizuje funkcje:
- (Criteria) wybór aktualnego kryterium do oceny uszeregowania, spośród zdefiniowanych w tablicy kryterów globalnych,
- (Auto) automatyczne generowanie uszeregowania wybranym algorytmem heurystycznym,
- (Manual) "ręczne" konstruowanie lub modyfikowanie uszeregowania przy aktywnym wsparciu (podpowiedziach) systemu,
- (Feasibility) sprawdzanie dopuszczalności uszeregowania wraz z pełną informacją o liczbie, rodzajach i miejscach wystąpienia naruszenia ograniczeń; dodatkowo podawana jest informacja o wartości kryterium,
- (View) tekstowa prezentacja uszeregowania w kontekście danych problemu,
- (Gantt) graficzna prezentacja uszeregowania z szerokim zestawem funkcji pomocniczych.

Grupa (Options) realizuje następujące funkcje:
- (Printing) drukowanie danych i wyników na drukarce,
- (Directories) ustawianie osobnych kartotek dla plików z danymi wejściowymi, plików z rozwiązaniami, plików roboczych, plików własnych systemu oraz plików dla biblioteki algorytmów,
- (Configuration) zestawienie konfiguracji programowo-sprzętowej systemu.

W założeniach proces rozwiązywania problemu jest podzielony na etapy realizowane przy aktywnym wsparciu systemu. Etap początkowy obejmuje sprecyzowanie danych wejściowych problemu (poprzez wczytanie ich z dysku lub bezpośrednie wprowadzenie przez użytkownika). Dane te podlegają kontroli poprawności i zgodności, a następnie wstępnej analizie. Właściwy proces rozwiązywania rozpoczyna się od sformułowania kryterium optymalizacji (tzn. podania $v_{max}$, $v_{sum}$, $w_{max}$, $w_{sum}$) i prowadzi do otrzymania uszeregowania. Uszeregowanie może zostać wygenerowane przez system (przy użyciu odpowiedniego algorytmu optymalizacyjnego) lub przez użytkownika. Użytkownik generuje rozwiązanie przez modyfikację istniejących uszeregowań lub tworzenie nowych. W dalszym ciągu system ocenia uszeregowanie (w sensie kryterium) oraz sprawdza jego dopuszczalność szczegółowo informując użytkownika o ewentualnych

naruszeniach ograniczeń. Wykorzystując otrzymane informacje użytkownik kontynuuje proces rozwiązywania modyfikując dane wejściowe, kryterium optymalizacji lub generując kolejne uszeregowania. W etapie końcowym użytkownik wybiera rozwiązanie najbardziej go satysfakcjonujące spośród wykreowanych. Na każdym etapie procesu rozwiązywania system kontroluje logiczny ciąg działań użytkownika i podpowiada ich właściwą kolejność. Niezależnie od tego dostępny jest stale system pomocnika ekranowego (help).

Aktualnie prowadzone są badania w kierunku poszukiwania nowych algorytmów rozwiązywania, w tym również dla problemu (1)-(4) sformułowanego wielokryterialnie (możliwe jest zastosowanie tutaj podejścia z pracy [12]).

LITERATURA

[1] Achuthan N.R., Grabowski J., Sidney J.B.: Optimal Flow Shop Scheduling with Earliness and Tardiness Penalties. OPSEARCH 1981, t.4.

[2] Anthonisse J.M., van Hee K.M., Lenstra J.K.: Resource-constrained project scheduling: an international exercise in DSS developement. IIASA Report 1987. Laxenburg, Austria.

[3] Błażewicz J., Lenstra J.K., Rinnooy Kan A.H.G.: Scheduling Subject to Resource Constraints: Classification and Complexity. Discrete Applied Mathematics 1983, t.5, s. 11-24.

[4] Grabowski J., Smutnicki C.: Problemy szeregowania z minimaksową funkcją kary. Archiwum Automatyki i Telemechaniki 1986, t.1-2, s. 21-37.

[5] Kelley J.E.jr: The Cricital-Path Method: Resource Planning and Scheduling, in: Industrial Scheduling, eds. Muth J.F. and Thompson G.L., Prentice Hall, Engl. Cliffs, New Jersey, 1963.

[6] Nowicki E., Smutnicki C.: System wspomagania decyzji w harmonogramowaniu zadań, Zeszyty Naukowe AGH, ser: Automatyka 49, 1989, 237-245.

[7] Sidney J.B.: Optimal Single-machine Scheduling with Earliness and Tardiness Penalties. Operations Research 1977, t.25, s. 62-69.

[8] Słowiński R.: Multiobjective Network Scheduling with Efficient Use of Renewable and Nonrenewable Resource. European Journal of Operational Research 1981, t.7, s. 265-273.

[9] Słowiński R., Soniewicki B.: Algorytm wielokryterialnego rozdziału zasobów w sieciowym planowaniu przedsięwzięć oraz jego implementacja mikrokomputerowa, Zeszyty Naukowe POlitechniki Śląskiej, ser: Automatyka 94, 1988, 303-316.

[10] Smutnicki C.: DSS for Project Scheduling. A Review of Problems, in Methodology and Software for Interacive Decision Support, eds. Lewandowski A., Stanchev I., Lecture Notes in Economics and Mathematical Systems 337, Springer-Verlag, 1987, 211-216.

[11] Talbot B.F.: Resource-Constrained Project Scheduling with Time-Resource Tradeoff. The Nonpreemptive Case. Management Science 1986, t.28, s.1197-1210.

[12] Wierzbicki A.P.: On the Completness and Constructiveness of Parametric Characterizations to Vector Optimization Problems. OR Spectrum 1986, t.8, s.73-87.

Recenzent: Doc.dr hinż. E.Toczyłowski

Wpłynęło do Redakcji do 1990-04-30.

DECISION SUPPORT FOR PRODUCTION SCHEDULING PURPOSES

S u m m a r y

The paper deals with resource- and time-constrained scheduling problem with non-regular goal function. The problem is the subject of the international research program "International Comparative Study in DSS Developement". Some approximation algorithms are presented and essential parts of DSS software are described.

ОПЕРАЦИОННАЯ СИСТЕМА ПРИНЯТИЯ РЕШЕНИЙ ДЛЯ СОСТАВЛЕНИЯ ГРАФИКА ПРОИЗВОДСТВА

Р е з ю м е

В статье рассмотрена проблема упрядочения задач при ресурсных и временных ограничениях с нерегулярным показателем качества упорядочения. Проблема является темой международной исследовательской программы "International Comparative Study in DSS Developement" координированной IIASA -ом. Представлены аппроксимационные алгоритмы, а также описаны существенные элементы построенной операционной системы принятия решений.