ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994
Seria: AUTOMATYKA z.114 Nr kol. 1250

Maciej HAPKE, Andrzej JASZKIEWICZ, Roman SŁOWIŃSKI
Politechnika Poznańska, Instytut Informatyki

ROZMYTE PROGRAMOWANIE SIECIOWE W ZASTOSOWANIU DO ZARZĄDZANIA PROJEKTEM SOFTWARE'OWYM

Streszczenie: Referat przedstawia zastosowanie systemu wspomagania decyzji FPS (Fuzzy Project Scheduling) do zarządzania projektem software'owym. Celem systemu FPS jest przydział zasobów (projektantów oprogramowania i programistów) do zadań (projektowanie systemu, projektowanie interfejsu użytkownika, implementacja modułów, integracja modułów i testy) dla jednego z dwóch kryteriów: czasu realizacji projektu lub maksymalnego opóźnienia. Wszystkie parametry typu czasowego (czasy wykonywania oraz pożądane terminy zakończenia) poszczególnych zadań projektu są niepewne. Niepewność tych parametrów modeluje się za pomocą liczb rozmytych typu L-R. Zastosowanie systemu do zarządzania projektem software'owym pokazane jest na realnym przykładzie.

FUZZY PROJECT SCHEDULING IN SOFTWARE PROJECT MANAGEMENT

Summary: This paper presents an FPS (Fuzzy Project Scheduling) decision making support system applied to software project scheduling. The purpose of the FPS system is to allocate resources (software engineers) among dependent activities (system design, user interface design, modules implementation, modules integration and tests), taking into account one of two criteria: project completion time or maximum lateness, under uncertain time parameters of activities. By time parameters we understand: durations and due dates of particular activities. The uncertainty of these parameters is modelled by means of L-R fuzzy numbers. The FPS system is described and a real example of its application to software project management is presented.

ORDONNANCEMENT FLOU D'UN PROJET DE CREATION DU LOGICIEL

Résumé: L'article présente une application du systeme d'aide a l'ordonnancement FPS (Fuzzy Project Scheduling) a la gestion d'un projet de création du logiciel. Le but de FPS et de proposer une allocation de moyens limités (concepteurs et programmeurs) aux tâches (conception du logiciel, conception de l'interface d'utilisateur, programmation par modules, leurs intégration et testes) prenant en compte un de deux criteres: la durée d'ordonnancement ou le retard maximum. Tous les parametres du temps (durées de tâches, dates d'achevement souhaitables) des tâches sont incartains. L'incertitude de cas parametres est modelisée a l'aide des nombres flous de type L-R. Une application du systeme FPS est montrée sur un exemple réel.

1. Wstęp

Programowanie sieciowe ze względu na duże zastosowanie w praktyce jest jednym z bardziej popularnych modeli badań operacyjnych. W dziedzinie tej obserwuje się tendencje tworzenia coraz bardziej realistycznych modeli. Nowe modele zostały wzbogacone o różne sposoby wykonywania zadań [13], nowe kategorie zasobów [24] oraz analizę wielokryterialną [17, 18]. Jednak, jak do tej pory, mało uwagi poświęcono innemu, ważnemu aspektowi rzeczywistości.

Otóż w praktyce często zachodzą sytuacje, w których nie można precyzyjnie określić czasów wykonywania poszczególnych zadań projektu. Dotyczy to przede wszystkim takich projektów, w których stosuje się nowe techniki realizacji. Dobrym przykładem projektu tego typu może być projekt software'owy, w którym przewidywanie czasów wykonywania poszczególnych zadań jest szczególnie trudne. Stosowanie nowych technik i metodologii, jak: obiektowo zorientowane projektowanie i programowanie, wspomagane komputerowo projektowanie oprogramowania, narzędzia projektowania interfejsu użytkownika oraz języki czwartej generacji, powoduje duże trudności w przewidywaniu czasów ukończenia poszczególnych zadań. Nawet gdy nowy projekt podobny jest do projektu już zrealizowanego, to bazowanie na poprzednich doświadczeniach okazuje się pomocne jedynie w niewielkim stopniu. Menadżerowie projektów software'owych, biorąc pod uwagę swe doświadczenie, dobrą wiedzę na temat używanych technik i narzędzi budowy oprogramowania oraz doświadczenie poszczególnych projektantów i programistów, są jednak w stanie w przybliżeniu określić czasy wykonywania poszczególnych zadań. Mogą oni określić zarówno przedziały możliwych czasów wykonywania zadań, jak i najbardziej możliwe wartości z tych przedziałów. Tego typu niepewność może być w naturalny sposób wyrażona za pomocą liczb rozmytych.

W pracy tej przedstawiono zastosowanie systemu wspomagania decyzji FPS (Fuzzy Project Scheduling) (por. [8]) w cyklu tworzenia oprogramowania. Celem systemu FPS jest rozdział zasobów (projektanci oprogramowania i programiści) między zadania niepodzielne (projektowanie systemu, projektowanie interfejsu użytkownika, implementacja modułów, integracja modułów oraz testy), o niepewnych parametrach typu czasowego, przy jednym z dwóch kryteriów: czas ukończenia projektu (C_{max}) lub maksymalne opóźnienie (L_{max}). Parametrami typu czasowego są: czasy wykonywania i pożądane terminy zakończenia poszczególnych zadań, a niepewność modeluje się za pomocą liczb rozmytych typu L-R [5]. Jak do tej pory znanych jest kilka podejść do rozwiązywania problemów programowania sieciowego w warunkach niepewności [4, 5, 6, 14], jednak ograniczają się one jedynie do modelu PERT, który nie uwzględnia ograniczeń zasobowych.

W następnym rozdziale sformułowany jest problem programowania sieciowego z rozmytymi parametrami czasowymi. Kolejny rozdział opisuje sposób transformacji problemu niedeterministycznego do zbioru równoważnych problemów deterministycznych. Opisano w nim również operację porównania wyników (liczb) rozmytych, do czego wykorzystano podejście polegające na porównaniu pól ograniczonych przez funkcje przynależności porównywanych rezultatów rozmytych [16]. System FPS opisano w rozdziale 4. W ostatniej części pracy przedstawiono realny przykład zastosowania systemu w jednej z faz zarządzania projektem software'owym.

2. Matematyczne sformułowanie problemu

System rozdziału zasobów charakteryzowany jest przez cztery komponenty [17, 24]:
$S = \{ R, Z, \zeta, C \}$,

gdzie:
R - skończony zbiór zasobów odnawialnych,
Z - skończony zbiór zadań projektu,
$\prec$ - relacja częściowego porządku w zbiorze Z,
C - skończony zbiór kryteriów oceny projektu.

Horyzont czasowy wykonania projektu T_h podzielony jest na jednostkowe okresy. Wszystkie parametry czasowe projektu są wielokrotnością jednostkowych okresów. Zbiór R składa się z p typów zasobów odnawialnych $R_1,...,R_p$, których zużycie w każdym okresie t ograniczone jest do $N_1,...,N_p$ jednostek. Ograniczenia zasobowe definiowane są przez wektor $N = [N_1,...,N_p]$.

Zbiór Z składa się z n niepodzielnych zadań mających dyskretne żądania zasobowe opisane przez wektor $r = [r_1,...,r_p]$, którego elementy określają liczbę jednostek zasobów odnawialnych $R_1,...,R_p$ żądanych w każdym okresie czasu. Dla każdego zadania Z_j określone są ponadto: czas wykonywania $\tilde{p}_j$, moment gotowości $\tilde{a}_j$ oraz pożądany termin zakończenia $\tilde{d}_j$. Przyjmuje się, że wszystkie parametry czasowe zadań są niepewne. W pracy tej reprezentowane są one przez liczby rozmyte oznaczone znakiem ~.

Zbiór kryterów C składa się z dwóch kryteriów czasowych: długości uszeregowania $\tilde{C}_{max}$ oraz maksymalnego opóźnienia $\tilde{L}_{max}$. Zauważmy, że poszczególne parametry czasowe zadań są niepewne, stąd wartości kryteriów również obarczone są niepewnością.

Procedura szeregowania zadań projektu polega na takim przydziale zasobów ze zbioru R do poszczególnych zadań ze zbioru Z, by zachowując wszystkie ograniczenia osiągnąć jak najlepszą wartość jednego z kryteriów ze zbioru C.

Problem ten można sformułować w postaci zadania zero-jedynkowego rozmytego programowania liniowego [8], a do jego rozwiązania zastosować jedną ze znanych procedur optymalizacyjnych [3].

Ponieważ problemy spotykane w praktyce mają zwykle zbyt duże rozmiary, by sensowne było zastosowanie dokładnej procedury optymalizacyjnej, w rozdziale 4 proponuje się procedurę heurystyczną, która daje w pełni zadowalające rezultaty, biorąc pod uwagę zakładaną niepewność danych.

3. Agregacja informacji niepewnej

3.1. Liczba rozmyta jako model parametru niepewnego

Niech M będzie liczbą rozmytą $M=\{(x,\mu_M(x)) \mid x\in\mathbf{R}\}$, gdzie $\mu_M(x)$ jest funkcją przynależności, o wartościach z przedziału [0, 1], które określają,w jakim stopniu x należy do M. α-cięciem liczby M jest zbiór $I(M,\alpha)=\{x\in\mathbf{R} \mid \mu_M(x)\geq\alpha\}$, gdzie $\alpha\in(0,1]$. Dolne i górne granice α-cięcia $I(M,\alpha)$ są równe odpowiednio $\inf_{x\in\mathbf{R}} I(M,\alpha)$ oraz $\sup_{x\in\mathbf{R}} I(M,\alpha)$. Zakładamy, że obie są skończone.

Wygodną i często stosowaną reprezentacją liczb rozmytych są liczby rozmyte typu L-R [5] zapisywane w postaci:

$$M = (\underline{m},\overline{m},\alpha,\beta)_{LR}$$

gdzie:
$[\underline{m},\overline{m}]$ - rdzeń M,
$\underline{m},\overline{m}$ - dolna i górna wartość M,
α,β - lewe i prawe rozmycie.

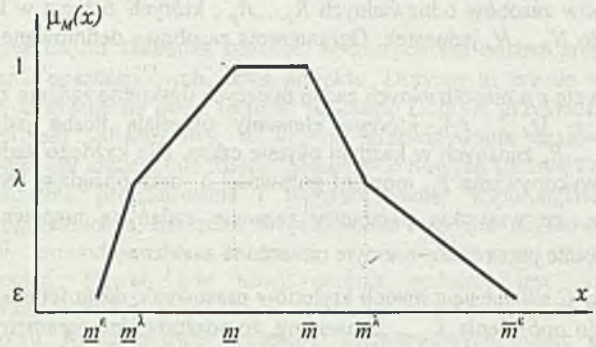
Funkcja przynależności liczby M może być reprezentowana przez dwie funkcje referencyjne L oraz R,takie że $L(0)=R(0)=1$ i $L(1)=R(1)=0$:

$\mu_M(x) = L[(\underline{m} - x)/\alpha]$ jeśli $x < \underline{m}$

$\mu_M(x) = 1$ jeśli $x \in [\underline{m}, \overline{m}]$

$\mu_M(x) = R[(x - \overline{m})/\beta]$ jeśli $x > \overline{m}$, $\alpha, \beta \in R^+$

W pracy tej proponujemy zastosowanie płaskiej liczby rozmytej L-R reprezentowanej przez pięcioodcinkową krzywą łamaną przedstawioną na rysunku 1. Liczbę taką będziemy definiować symbolicznie w następujący sposób: $M = (\underline{m}^{\varepsilon}, \underline{m}^{\lambda}, \underline{m}, \overline{m}, \overline{m}^{\lambda}, \overline{m}^{\varepsilon})$.



Rys.1. Płaska liczba rozmyta
Fig.1. Flat fuzzy number

3.2. Transformacja problemu z informacją niepewną do zbioru równoważnych problemów deterministycznych

W pracy zastosujemy powszechnie stosowane podejście do rozwiązywania problemów z informacją niepewną [20], które można streścić w trzech krokach:

(i)- modelowanie niepewności,

(ii)- transformacja problemu z informacją niepewną do zbioru równoważnych problemów deterministycznych,

(iii)-rozwiązanie poszczególnych problemów deterministycznych oraz ich agregacja do globalnego rezultatu przedstawionego w kategoriach przyjętego modelu niepewności.

Zgodnie z tym podejściem rozwiązanie problemu programowania sieciowego z niepewnymi parametrami czasowymi przebiega tu w sposób następujący:

ad (i)

W fazie modelowania niepewności przez liczby rozmyte pewnym problemem jest uzyskanie od eksperta informacji na temat kształtu funkcji przynależności. Ponieważ w praktyce ekspert jest w stanie określić wartości funkcji przynależności jedynie na kilku poziomach niepewności, dlatego pytamy go o optymistyczne i pesymistyczne, jego zdaniem, wartości dla trzech możliwych wersji niepewności, które odpowiadają konkretnym poziomom odcięcia (1, λ, ε).

W szczególności etap ten polega na oszacowaniu przez eksperta takich przedziałów, do których przynależność oznacza:

- z całą pewnością należy do zbioru możliwych wartości ($\mu(x) = 1$),
- ma dużą szansę, by należeć do zbioru możliwych wartości ($\mu(x) > \lambda$),

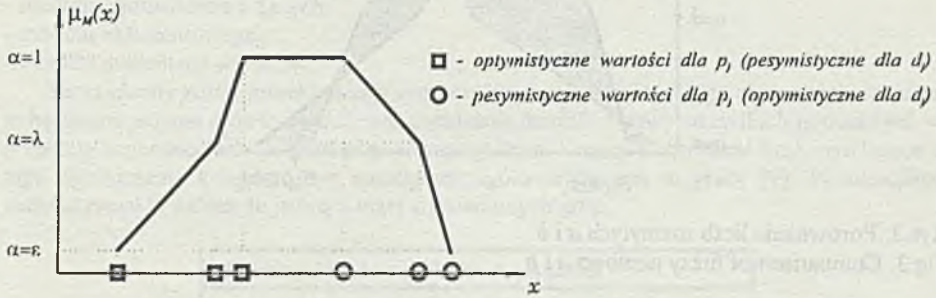
oraz o taki przedział, poza granicami którego leżą wartości, które

- raczej nie należą do zbioru możliwych wartości ($\mu(x) < \varepsilon$).

Sensowne wydaje się przyjęcie λ=0.6 oraz ε=0.1.

ad (ii)

Na zamodelowanej w taki sposób liczbie rozmytej dokonywana jest operacja α-cięcia dla trzech wspomnianych wcześniej poziomów α (1, λ, ε). Każda operacja α-cięcia charakteryzowana jest parą dolnej i górnej granicy, z których jedna zwana jest optymistyczną, a druga pesymistyczną (por. rysunek 2). W praktyce są to te same wartości, które zostały oszacowane przez eksperta w fazie modelowania niepewności. Zatem dla każdej liczby rozmytej uzyskujemy sześć zdeterminowanych wartości, które brane są pod uwagę w procedurze konstrukcji harmonogramu projektu, odpowiednio, w wersji optymistycznej i pesymistycznej.



Rys.2. Trzy α-cięcia liczby rozmytej oraz odpowiadające jej zdeterminowane wartości
Fig. 2. Three α-level cuts of a fuzzy number and six corresponding real values

ad (iii)

Stosujemy tu heurystyki dla dwóch scenariuszy: optymistycznego i pesymistycznego na trzech poziomach α. W rezultacie otrzymujemy sześć różych uszeregowań i tyleż wyników optymalizacji rozpatrywanej funkcji kryterialnej, które agregowane są do rezultatu rozmytego.

W rzeczywistości w prezentowanym systemie pojedynczy problem deterministyczny jest rozwiązywany nie przez jedną, ale przez wiązkę heurystyk. Reprezentują one rodzinę heurystyk równoległych, w których procedura szeregowania przydziela zasoby do poszczególnych zadań w kolejności określanej według odpowiedniej reguły. Skutkiem zastosowania wielu heurystyk jest uzyskanie zbioru rozwiązań dopuszczalnych, z których najlepsze prezentowane jest użytkownikowi systemu. Aby tego dokonać, należy zastosować operację porównywania liczb rozmytych.

3.3. Porównywanie liczb rozmytych

Główna idea operacji porównywania liczb rozmytych oparta jest na porównywaniu pól ograniczonych przez funkcje przynależności liczb rozmytych [1, 11, 16].

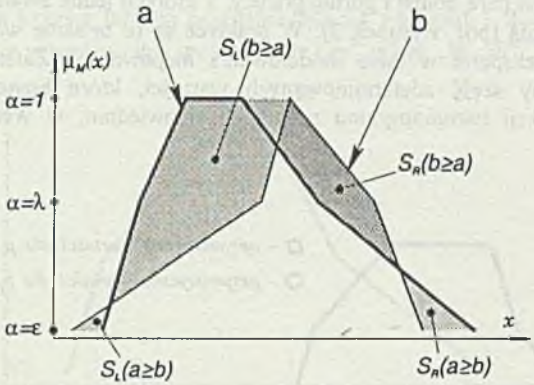
Niech *a*, *b* będą liczbami rozmytymi, a $S_L(a \geq b)$, $S_R(a \geq b)$ polami ograniczonymi przez ich funkcje przynależności (por. rysunek 3):

$$S_L(a \geq b) = \int_{U_1(a,b)} [\inf_{x \in R} I(a,\alpha) - \inf_{x \in R} I(b,\alpha)] d\alpha ,$$

gdzie $U_1(a,b)$ jest podzbiorem [ε,1]: $\{\alpha \mid \inf_{x \in R} I(a,\alpha) \geq \inf_{x \in R} I(b,\alpha)\}$, oraz

$$S_R(a \geq b) = \int_{V_1(a,b)} [\sup_{x \in R} I(a,\alpha) - \sup_{x \in R} I(b,\alpha)] d\alpha ,$$

gdzie $V_I(a,b)$ jest podzbiorem $[\varepsilon,1]$: $\{\alpha \mid \sup_{x\in R} I(a,\alpha) \geq \sup_{x\in R} I(b,\alpha)\}$.
$S_L(a \leq b)$ i $S_R(a \leq b)$ są definiowane analogicznie.



Rys.3. Porównanie liczb rozmytych *a* i *b*
Fig.3. Comparison of fuzzy numbers *a* i *b*

Stopień, w którym $a \geq b$, definiowany jest następująco [16]:
$C(a\geq b) = max\ \{ S_L(a\geq b)+S_R(a\geq b)-S_L(b\geq a)-S_R(b\geq a),\ 0\}$
Przyjmujemy, że $a \geq b$ wtedy, gdy $C(a,b)>0$.

4. Heurystyczne procedury rozwiązywania równoważnego problemu deterministycznego

System FPS (Fuzzy Project Scheduling) zaproponowany przez Hapke i Słowińskiego [8] jest systemem wspomagania decyzji związanych z rozdziałem zasobów pomiędzy zadania projektu przy rozmytych parametrach czasowych. System stosuje trzykrokową procedurę opisaną w podrozdziale 3.2. W systemie każdy pojedynczy problem deterministyczny rozwiązywany jest przez wiązkę dwunastu heurystyk, których zadaniem jest przydział ograniczonych zasobów do zadań według priorytetów wyliczanych za pomocą odpowiedniej reguły heurystycznej [10]. Podczas wyliczania priorytetów zadań bierze się pod uwagę na różne sposoby ich charakterystyki czasowe, zasobowe i/lub kolejnościowe uzyskując w ten sposób dwanaście list priorytetowych. Ponadto, ażeby uzyskać większą różnorodność rozwiązań, dla każdej listy generowanych jest pięć mutacji. Zastosowanie takiego podejścia może dać do sześćdziesięciu różnych rozwiązań. Dla dużych problemów takie podejście okazało się efektywniejsze i mniej czasochłonne niż heurystyka oparta na metodzie symulowanej relaksacji (simulated annealing) [19].

Procedura przydziału zasobów bierze pod uwagę tylko te zadania, gotowe do wykonania, których wszystkie poprzedniki zostały już uszeregowane i przydziela im zasoby według ustalonych wcześniej priorytetów.

Heurystyczne reguły priorytetowe, sposób gengerowania losowych mutacji reguł oraz procedura równoległa zostały szczegółowo opisane w pracy [8].

Uruchomienie opisanej powyżej heurystyki dla sześciu scenariuszy zdeterminowanych parametrów czasowych daje, po agregacji, do sześćdziesięciu różnych wyników rozmytych, z których wybierany jest najlepszy metodą porównywania liczb rozmytych przedstawioną w podrozdziale 3.3.

5. Zastosowanie systemu FPS do zarządzania projektem software'owym

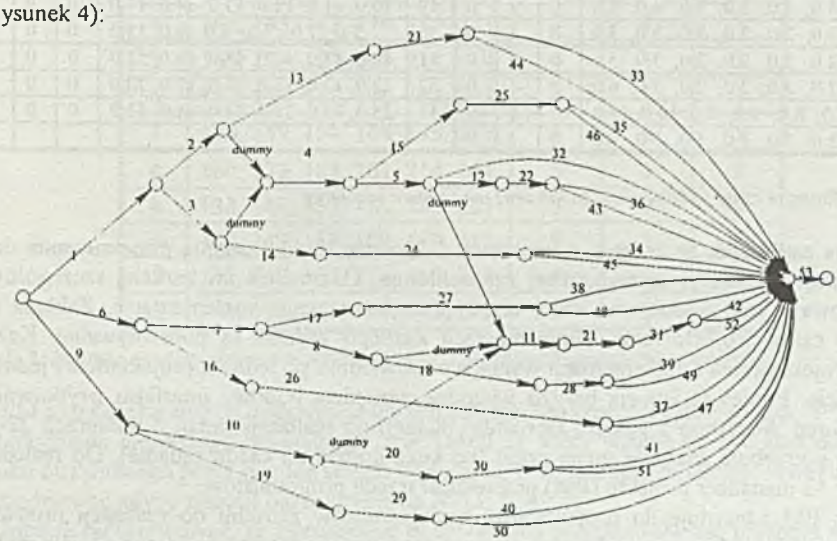
W rozdziale tym pokażemy, jak system FPS może pomóc menadżerowi projektu software'owego w procesie zarządzania, a w szczególności jak może pomóc odpowiedzieć na pytania o koszty związane z czasem wykonania projektu oraz o ryzyko przekroczenia wymaganego czasu ukończenia projektu.

Działanie systemu zostanie pokazane na przykładzie projektu oprogramowania, które będzie przedmiotem rzeczywistej realizacji. Będzie to nowa wersja systemu rozwiązującego problemy programowania sieciowego w warunkach niepewności. System składać się będzie z trzech głównych części:
- modułu wprowadzania danych,
- modułu obliczeniowego,
- modułu prezentacji wyników.

Sporządzony został projekt całego systemu, ale z powodu dużej jego złożoności rozważać tu będziemy jedynie projekt modułu wprowadzania danych. Nazwy wszystkich potrzebnych w projekcie czynności, relacje pomiędzy poszczególnymi klasami i obiektami oraz wynikające z tego ograniczenia kolejnościowe zostały szczegółowo opisane w pracy [9]. Poszczególne zadania projektu należą do jednej z niżej wymienionych grup:

Numery zadań	Nazwa typu zadań
1-11	Szczegółowy projekt klas
12-21	Projekt klas typu EDIT WINDOW
22-31	Projekt interfejsów użytkownika
32-52	Implementacja klas
53	Integracja modułów i testy

W wyniku szczegółowej analizy powstała następująca sieć ograniczeń kolejnościowych (rysunek 4):



Rys.4. Graf ograniczeń kolejnościowych pomiędzy zadaniami projektu
Fig.4. The graph of precedence constraints among project activities

Do wykonania projektu niezbędne są następujące grupy osób:
- projektanci systemu ..... (R_1),
- projektanci interfejsu użytownika ... (R_2),
- programiści ..... (R_3).

Rozmyte czasy wykonywania poszczególnych zadań projektu w reprezentacji zaproponowanej w podrozdziale 3.1 oraz ich żądania zasobowe przedstawione są w Tabeli 1. Czasy wykonywania wyrażone są w godzinach, a żądania zasobowe w jednostkach.

nr	$(\underline{m}^{\varepsilon}, \underline{m}^{\lambda}, \underline{m}, \overline{m}, \overline{m}^{\lambda}, \overline{m}^{\varepsilon})$	r_1	r_2	r_3	nr	$(\underline{m}^{\varepsilon}, \underline{m}^{\lambda}, \underline{m}, \overline{m}, \overline{m}^{\lambda}, \overline{m}^{\varepsilon})$	r_1	r_2	r_3
1	3.0, 3.5, 4.0, 4.0, 5.0, 7.0	1	0	0	28	3.0, 3.0, 3.0, 3.0, 4.0, 6.0	0	1	0
2	9.0, 9.5, 10.0, 10.0, 11.0, 14.0	1	0	0	29	3.0, 3.0, 3.0, 3.0, 4.0, 5.0	0	1	0
3	9.0, 9.5, 10.0, 10.0, 11.0, 15.0	1	0	0	30	7.0, 9.0, 10.0, 10.0, 11.0, 15.0	0	1	0
4	6.0, 7.0, 8.0, 8.0, 9.0, 12.0	1	0	0	31	3.0, 4.0, 5.0, 5.0, 6.0, 9.0	0	1	0
5	13.0, 14.0, 15.0, 15.0, 16.0, 20.0	1	0	0	32	4.0, 4.0, 4.0, 4.0, 5.0, 9.0	0	0	1
6	13.0, 14.0, 15.0, 15.0, 16.0, 20.0	1	0	0	33	13.0, 14.0, 15.0, 17.0, 18.0, 23.0	0	0	1
7	23.0, 24.0, 25.0, 25.0, 27.0, 30.0	1	0	0	34	13.0, 14.0, 15.0, 17.0, 18.0, 23.0	0	0	1
8	14.0, 14.0, 15.0, 15.0, 16.0, 19.0	1	0	0	35	8.0, 9.0, 10.0, 12.0, 13.0, 19.0	0	0	1
9	15.0, 15.0, 15.0, 15.0, 16.0, 20.0	1	0	0	36	23.0, 24.0, 25.0, 27.0, 28.0, 35.0	0	0	1
10	13.0, 14.0, 15.0, 15.0, 16.0, 20.0	1	0	0	37	23.0, 24.0, 25.0, 27.0, 28.0, 33.0	0	0	1
11	23.0, 24.0, 25.0, 25.0, 26.0, 30.0	1	0	0	38	36.0, 37.0, 40.0, 43.0, 45.0, 50.0	0	0	1
12	6.0, 6.0, 6.0, 7.0, 8.0, 9.0	1	0	0	39	17.0, 18.0, 19.0, 20.0, 21.0, 25.0	0	0	1
13	5.0, 5.0, 5.0, 5.0, 6.0, 9.0	1	0	0	40	19.0, 20.0, 21.0, 23.0, 24.0, 30.0	0	0	1
14	5.0, 5.0, 5.0, 5.0, 6.0, 9.0	1	0	0	41	18.0, 19.0, 20.0, 22.0, 23.0, 25.0	0	0	1
15	5.0, 5.0, 5.0, 5.0, 6.0, 9.0	1	0	0	42	30.0, 31.0, 32.0, 33.0, 34.0, 36.0	0	0	1
16	13.0, 14.0, 15.0, 15.0, 17.0, 20.0	1	0	0	43	7.0, 8.0, 9.0, 9.0, 10.0, 13.0	0	0	1
17	4.0, 5.0, 6.0, 6.0, 7.0, 11.0	1	0	0	44	7.0, 8.0, 9.0, 9.0, 10.0, 12.0	0	0	1
18	3.0, 3.5, 4.0, 4.0, 5.0, 9.0	1	0	0	45	7.0, 8.0, 9.0, 9.0, 10.0, 12.0	0	0	1
19	3.0, 3.5, 4.0, 4.0, 5.0, 9.0	1	0	0	46	7.0, 8.0, 9.0, 9.0, 10.0, 12.0	0	0	1
20	18.0, 19.0, 20.0, 23.0, 25.0, 30.0	1	0	0	47	31.0, 32.0, 33.0, 35.0, 36.0, 40.0	0	0	1
21	7.0, 8.0, 9.0, 9.0, 10.0, 15.0	1	0	0	48	7.0, 8.0, 9.0, 9.0, 10.0, 12.0	0	0	1
22	3.0, 3.0, 3.0, 3.0, 4.0, 5.0	0	1	0	49	10.0, 11.0, 12.0, 13.0, 14.0, 17.0	0	0	1
23	2.0, 2.0, 2.0, 2.0, 3.0, 5.0	0	1	0	50	7.0, 7.0, 7.0, 8.0, 9.0, 12.0	0	0	1
24	2.0, 2.0, 2.0, 2.0, 3.0, 5.0	0	1	0	51	40.0, 41.0, 42.0, 45.0, 46.0, 52.0	0	0	1
25	2.0, 2.0, 2.0, 2.0, 3.0, 6.0	0	1	0	52	12.0, 13.0, 15.0, 16.0, 17.0, 20.0	0	0	1
26	7.0, 8.0, 9.0, 9.0, 10.0, 14.0	0	1	0	53	25.0, 30.0, 31.0, 42.0, 44.0, 53.0	0	0	3
27	3.0, 3.0, 4.0, 4.0, 5.0, 9.0	0	1	0					

Tabela 1. *Rozmyte czasy wykonywania zadań oraz ich żądania zasobowe*

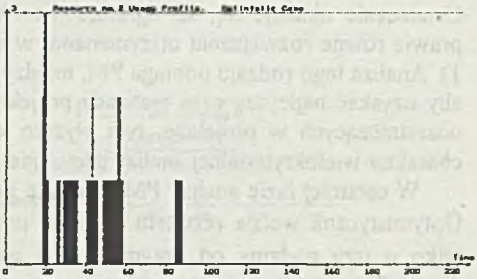
Można zauważyć, że zadania o numerach 1-21 oraz 32-52 (zadania projektowania oraz implementacji modułów) są najbardziej czasochłonne. Oczywiście im bardziej szczegółowo zaprojektowany jest moduł, tym mniej czasu potrzeba na jego implementację. Zakłada się zatem, że czasy projektowania i implementacji każdego zadania są porównywalne. Każde zadanie projektowania i implementacji wymaga odpowiednio po jednym projektancie i jednym programiście. Ponieważ istnieją bardzo wygodne narzędzia budowy interfejsu użytkownika (np. Resource Workshop z pakietu Borlanda), dlatego do realizacji zadań o numerach 22-31 projektant potrzebuje znacznie mniej czasu (po kilka godzin na każde zadanie). Do realizacji zadania nr 53 menadżer projektu (PM) przewidział trzech programistów.

Zanim PM zdecyduje, ilu projektantów i programistów zatrudni do realizacji projektu, chciałby wiedzieć, jakie jest maksymalne zapotrzebowanie projektu na zasoby i jaki byłby rezultat, gdyby nie istniały żadne ograniczenia na zasoby. Uruchamia zatem metodę ścieżki krytycznej (CPM) dla trzech optymistycznych i trzech pesymistycznych scenariuszy na trzech

poziomach niepewności (α równe odpowiednio 1, 0.6, 0.1). W wyniku tego otrzymuje rezultat $\tilde{C}_{max}$=(138, 149, 157, 169, 179, 212).

Niech $N=[N_1, N_2, N_3]$ oznacza wektor ograniczeń kolejnościowych. Gdyby PM zatrudnił maksymalną możliwą liczbę projektantów i programistów N=[7, 3, 8], spodziewałby się rezultatu należącego do $I(\tilde{C}_{max},\lambda)$=[149, 179], podczas gdy bardzo możliwy rezultat należałby do $I(\tilde{C}_{max},1)$=[157, 169].

Jak wynika z uzyskanego profilu wykorzystania zasobów, wszyscy projektanci i programiści byliby jednocześnie wykorzystywani tylko przez bardzo krótki okres trwania projektu. Nie jest zatem konieczne zatrudnianie tylu projektantów i programistów, ilu wskazała metoda ścieżki krytycznej. PM zadecydował, że po to, aby zachować jednolity styl okien dialogowych, zatrudni tylko jednego projektanta interfejsu użytkownika. Rysunek 5 pokazuje zresztą, że do zrealizowania postawionych zadań bez znacznego wpływu na czas realizacji całego projektu jeden projektant interfejsu wystarczy. W dalszej części zostanie przeprowadzona analiza typu "what-if" dla różnych wektorów ograniczeń zasobowych, której wyniki przedstawione są w Tabeli 2.



Rys.5. Profil wykorzystania projektantów interfejsu użytkownika
Fig.5. User interface designer usage profile

nr -	Rezultat rozmyty $\tilde{C}_{max}$ [h]	N_1 [jedn.]	N_2 [jedn.]	N_3 [jedn.]	
1	138, 149, 157, 169, 179, 212	7	3	8	CPM
2	161, 175, 186, 201, 214, 262	3	1	3	
3	143, 156, 162, 180, 193, 238	3	1	4	
4	146, 153, 162, 174, 185, 231	3	1	5	
5	160, 173, 182, 203, 214, 264	4	1	3	
6	141, 153, 157, 170, 184, 223	4	1	4	
7	138, 149, 158, 169, 179, 212	4	1	5	
8	160, 173, 182, 201, 214, 261	5	1	3	
9	138, 149, 157, 169, 183, 221	5	1	4	
10	138, 149, 157, 169, 179, 212	5	1	5	
11	138, 149, 157, 174, 180, 224	6	1	4	
12	145, 157, 165, 179, 190, 233	3	2	4	

Tabela 2. *Rozmyte rezultaty otrzymane przez system FPS dla różnych ograniczeń zasobowych*

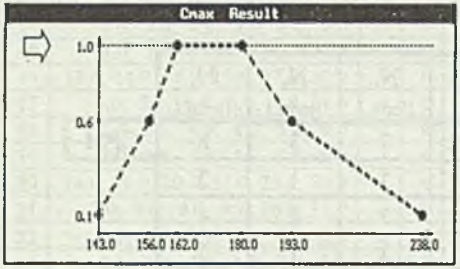
PM rozpoczyna analizę decyzyjną z ograniczeniami wyrażonymi przez wektor N=[3, 1, 3]. Po otrzymaniu pierwszego rezultatu (nr 2 w Tabeli 2), który wydaje się być zbyt odległy od wyniku otrzymanego przez metodę ścieżki krytycznej, PM pyta system, jakiego wyniku mógłby się spodziewać, gdyby zmienił ograniczenia na N=[3, 1, 4], itd.

Można zaobserwować, że zatrudnianie dodatkowych projektantów nie polepsza w sposób zdecydowany rezultatu $\tilde{C}_{max}$ (por. rezultaty 2, 5 oraz 8). Jak już wiadomo, PM zdecydował się zatrudnić w projekcie tylko jednego projektanta interfejsu użytkownika, bazując na profilu

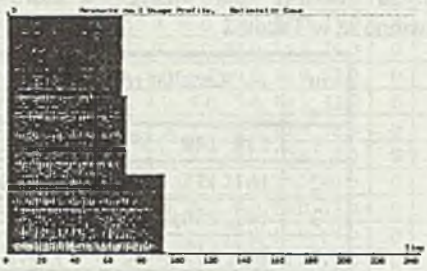
zapotrzebowania. Porównanie rezultatów nr 12 i 3 zdaje się potwierdzać wcześniejsze przypuszczenia, że zwiększenie liczby projektantów interfejsu użytkownika nie wpływa znacząco na skrócenie czasu realizacji całego projektu.

Okazuje się, że największy wpływ na rezultat ma liczba zatrudnionych programistów. Zwiększenie ich liczby z 3 do 4 i z 4 do 5 polepsza rezultat $\bar{C}_{max}$ (por. grupy rezultatów nr 2, 3, 4; 5, 6, 7 oraz 8, 9, 10). System generuje znacząco lepszy rezultat, gdy N_2=1, a N_3=4. Ostatecznie okazuje się, że ograniczenie N=[4, 1, 5] prowadzi do rozwiązania, które jest prawie równe rozwiązaniu otrzymanemu w wyniku uruchomienia CPM (por. rezultaty nr 7 i 1). Analiza tego rodzaju pomaga PM, między innymi, zdecydować, ilu ludzi powinien zatrudnić, aby uzyskać najlepszy czas realizacji projektu. Z drugiej zaś strony, im większa liczba ludzi uczestniczących w projekcie, tym wyższe koszty z nim związane. Analiza taka ma zatem charakter wielokryterialnej analizy decyzyjnej.

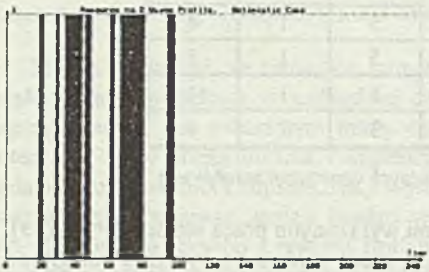
W ostatniej fazie analizy PM decyduje się wybrać jedno z rozwiązań o numerach 3, 4, 6. Optymistyczna wersja rezultatu nr 3 na poziomie α=0.6 ($\inf I(\bar{C}_{max}, 0.6)$=156) jest większa tylko o trzy godziny od optymistycznej wersji rezultatów nr 4 i 6. Choć w najgorszym przypadku wynik ten jest gorszy o osiem i dziewięć godzin od odpowiadających mu pesymistycznych wersji wyników nr 4 i 6, rozwiązanie nr 3 przewyższa pozostałe dwa niższym zużyciem zasobów (N=[3, 1, 4]). Ostatecznie więc PM jest w stanie zaakceptować ten rezultat (rysunek 6). Odpowiadający mu wykres Gantta dla α=0.6 oraz poszczególne profile wykorzystania zasobów przedstawione są na rysunkach 7 -10.



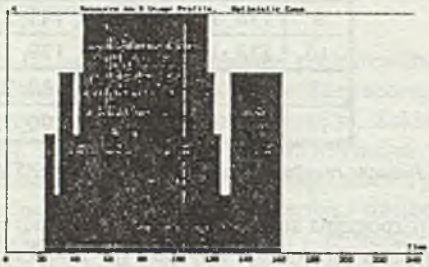
Rys.6. Rezultat C_{max} dla N=[3,1,4]
Fig.6. C_{max} results for N=[3,1,4]



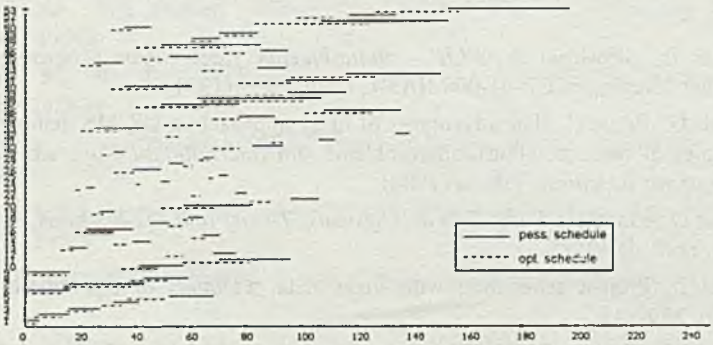
Rys.7. Profil wykorzystania projektantów
Fig.7. System designer usage profile



Rys.8. Profil wykorzystania projektantów interfejsu użytkownika
Fig.8. User interface designer usage profile



Rys.9. Profil wykorzystania programistów
Fig.9. Programmers usage profile



Rys.10. Wykres Gantta dla N=[3,1,4]
Fig.10. The Gantt chart for N=[3,1,4]

Porównanie rezultatów nr 1 i 3 pokazuje, że ostateczny wynik niewiele różni się od wyniku otrzymanego przy użyciu CPM, a wykorzystuje znacznie mniej zasobów. Zarówno końcowy rezultat (rysunek 6), jak i wykres Gantta (rysunek 10) ukazują pewną specyficzną cechę projektów software'owych, jaką jest duże rozmycie końcowego rezultatu. Najbardziej możliwy czas ukończenia projektu należy do przedziału $I(\tilde{C}_{max},1)$=[162, 180], ale może również zawierać się w przedziale $I(\tilde{C}_{max},0.6)$=[156, 193]. Natomiast wartości spoza $I(\tilde{C}_{max},0.1)$=[143, 238] są prawie niemożliwe do uzyskania.

Aby w pełni wykorzystać możliwości systemu FPS, zaleca się stosowanie analizy projektów w czasie rzeczywistym. Analiza taka polega na uruchamianiu systemu dla nowych parametrów, każdorazowo po ukończeniu jakiegokolwiek zadania. Oczywiście im bliżej końca, tym mniejsza niepewność końcowego rezultatu (w projekcie zaczynają przeważać czasy o wartościach ze zbioru liczb rzeczywistych). Po ukończeniu realizacji całego projektu PM może porównać rzeczywisty czas ukończenia projektu z obliczonymi na poszczególnych etapach wynikami rozmytymi, co wzbogaci jego doświadczenie i pozwoli mu lepiej oszacować parametry czasowe przyszłych projektów.

6. Podsumowanie

W pracy tej staraliśmy się pokazać, że system FPS może być bardzo pomocnym narzędziem w cyklu tworzenia oprogramowania. W szczególności może być wykorzystany przy podejmowaniu decyzji dotyczącej liczby zatrudnionych w projekcie osób, pokazując jej wpływ na czas realizacji projektu. Choć rezultaty otrzymywane w wyniku zastosowania systemu charakteryzują się pewnym poziomem niepewności, to dają jednak wystarczająco dużo informacji o rezultacie nie tylko w wersji optymistycznej i pesymistycznej, ale również informują o rozkładzie możliwości uzyskania konkretnej wartości. System może być również wykorzystany do analizy ryzyka przekroczenia dyrektywnego terminu ukończenia projektu.

LITERATURA

[1] Baldwin F.F., Guild N.C.F., Comparison of fuzzy sets on the same decision space, *Fuzzy Sets and Systems* 2 (1979) 213-231.

[2] Coad P., Yourdon E., *Object - Oriented Analysis*, Yourdon Press (Englewood Cliffs, 1990).

[3] Czyżak P., Słowiński R., *FLIP - multiobjective fuzzy linear programming package*, Collaborative Paper, CP-91-008 IIASA, Laxenburg (1991).

[4] Dubois D., Prade H., The advantages of fuzzy approach in OR/MS demonstrated on two examples of resource allocations problems, *5th International Congress on Cybernetics and Systems Research*, Vienna (1980).

[5] Dubois D., Prade H., *Fuzzy Sets and Systems: Theory and Applications*, Academic Press, (New York, 1980).

[6] Geidel J., Project scheduling with fuzzy data, *Methods of Operations Research* 62 (1990), 339-347.

[7] Hapke M., *System PROFAD for fuzzy project scheduling*, Technical Report 11/91, Institute of Computing Science, Technical University of Poznan (1991).

[8] Hapke M., Słowiński R., A DSS for resource-constrained project scheduling under uncertainty, *Journal of Decision Systems* vol. 2 no. 2 (1993) 111-128.

[9] Hapke M., Jaszkiewicz A., Słowiński R.: Fuzzy project scheduling system for software development, artykuł przyjęty do druku w *Fuzzy Sets and Systems*.

[10] Kelley J.E. Jr, The critical-path method: resource planning and scheduling, in: J.F. Muth and G.L. Thompson (eds.), *Industrial Scheduling*, Prentice Hall, Engl. Cliffs, (New Jersey, 1963).

[11] Nakamura K., Preference relation on a set of fuzzy utilities, *Fuzzy Sets and System* 20 (1986) 147-162.

[12] Patterson J.H., Słowiński R., Talbot F.B., Węglarz J., An algorithm for a general class of precedence and resource-constrained scheduling problems. section I.1 in: Słowiński R., Węglarz J. (eds.), *Advances in Project Scheduling* (Elsevier, Amsterdam, 1989) 3-28.

[13] Patterson J.H., Słowiński R., Talbot F.B., Węglarz J., Computational experience with a backtracking algorithm for solving a general class of precedence and resource-constrained scheduling problems, *Europ. J. Opl. Res. 49* (1990) 68-79.

[14] Prade H., Using fuzzy set theory in a scheduling problem: a case study, *Fuzzy Sets and Systems* 2 (1979) 153-165.

[15] Rommelfanger H., FULPAL: An interactive method for solving (multiobjective) fuzzy linear programming problems, section 5 in: Słowiński R., Teghem J. (eds.) *Stochastic Versus Fuzzy Approaches to Multiobjective Mathematical Programming under Uncertainty*, (Kluwer Academic Publishers, Dordrecht 1990) 279-299.

[16] Roubens M., Inequality constraints between fuzzy numbers and their use in mathematical programming, section 7 in: Słowiński R., Teghem J. (eds.) *Stochastic Versus Fuzzy Approaches to Multiobjective Mathematical Programming under Uncertainty*, (Kluwer Academic Publishers, Dordrecht, 1990) 321-330.

[17] Słowiński R., Multiobjective network scheduling with efficient use of renewable and non renewable resources, *Europ. J. Opl. Res*, 7 (1981) 265-273.

[18] Słowiński R., Multiobjective project scheduling under multiple-category resource constraints. section I.7 in Słowiński R., Węglarz J. (eds.): *Advances in Project Scheduling* (Elsevier, Amsterdam, 1989) 151-167.

[19] Słowiński R., Soniewicki B., Węglarz J., *MPS - DSS for multiobjective project scheduling*, Collaborative Paper, CP-91-007 IIASA, Laxenburg (1991).

[20] Słowiński R., Teghem J. (eds.), *Stochastic Versus Fuzzy Approaches to Multiobjective Mathematical Programming under Uncertainty*, (Kluwer Academic Publishers, Dordrecht, 1990).

[21] Słowiński R., Węglarz J. (eds.), *Advances in Project Scheduling*, (Elsevier, Amsterdam,1989).

[22] Sommerville I., *Software Engineering*, Addison-Wesley Publishing Company, (Wokingham, 1992).

[23] Węglarz J. (ed.), Special Issue on Project Management and Scheduling, *Europ. J. Opl. Res.* 49, 1 (1990).

[24] Węglarz J., On certain models of resource allocation problems, *Kybernetes* 9, 1 (1980) 61-66.

Recenzent: Prof. dr hab. inż. Andrzej Grzywak

Wpłynęło do Redakcji do 30.04.1994 r.

Abstract

This paper presents an FPS (Fuzzy Project Scheduling) decision making support system applied to software project scheduling. The purpose of the FPS system is to allocate resources (software engineers) among dependent activities (system design, user interface design, modules implementation, modules integration and tests), taking into account one of two criteria: project completion time or maximum lateness, under uncertain time parameters of activities. By time parameters we understand: durations and due dates of particular activities. The uncertainty of these parameters is modelled by means of L-R fuzzy numbers.

Following usual approach to solving decision problems with uncertain information, a solution procedure is proposed. It can be summarized in three steps. In the first step, the uncertainty is modelled. The second step consists in transformation of the problem with uncertain information to a set of associate deterministic problems. In the last step, the associate deterministic problems are solved and the results are aggregated into a global fuzzy result.

The procedure for transforming the fuzzy scheduling problem into a set of associate deterministic problems is based on the use of α-cuts. Optimistic and pessimistic schedules are generated for three given α-levels, i.e. three optimistic and three pessimistic ones, called scenarios. Each associate deterministic problem is not solved by a single heuristic but by a sixty of them. They represent a family of parallel priority heuristics in which the scheduling of activities proceeds in the order of priorities affected according to a heuristic rule. With a dozen of rules and five random mutations of each of them, up to sixty different results are obtained for each scenario, so, after aggregation of six scenarios, we hare up to sixty different fuzzy results. As only the best one is presented to the user, a method of comparing the fuzzy numbers is proposed. The comparison of obtained fuzzy results is based on the compensation of areas determined by the membership functions. An application of the FPS system to a hypothetical software project scheduling example is presented.