ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1996
Seria AUTOMATYKA z. 117 Nr kol. 1337

Maciej HAPKE
Politechnika Poznańska

PROGRAMOWANIE SIECIOWE W WARUNKACH NIEPEWNOŚCI

Streszczenie. Rozważany jest problem programowania sieciowego, w którym zadania mogą być wykonywane na różne sposoby i wymagają do realizacji zasobów różnych kategorii. Zaproponowano uogólnienie procedury szeregowania i rozdziału zasobów na przypadek rozmyty. Procedura ta wykorzystuje oryginalne rozwiązanie problemu następstwa charakterystycznych chwil czasowych wyrażonych w kategoriach liczb rozmytych. Rozważany jest również problem porównywania rozmytych rezultatów funkcji kryterialnych. Ostatnia część zawiera numeryczny przykład działania uogólnionej procedury szeregującej.

PROJECT SCHEDULING UNDER UNCERTAINTY

Summary. The multi-mode project scheduling problem under multi-category resource constraints with fuzzy time parameters of activities is considered. A generalization of a parallel scheduling procedure to fuzzy time parameters has been proposed. The procedure solves the problem of succession of particular time moments expressed by means of fuzzy numbers. The problem of comparison of fuzzy optimization results is also considered. Last part of this paper presents a numerical example.

1. Wstęp

Współczesny menedżer często korzysta z dostępnych technik i produktów informatycznych wykorzystujących znane metody zarządzania. Metody te, budowane na pewnych modelach badań operacyjnych, nie zawsze wystarczają do uwzględnienia wszystkich okoliczności towarzyszących realizacji projektu. Bardzo częsta niezgodność planu proponowanego przez istniejące systemy z rzeczywistością wymusza konstrukcję nowych modeli i metod. Ważnym elementem, który ostatnio coraz częściej uwzględnia się w pracach naukowych, jest niepewność danych, którą modeluje się za pomocą rozkładu prawdopodobieństwa lub za pomocą liczb rozmytych. To drugie podejście do planowania przyszłości wydaje się być najwłaściwsze wtedy, gdy brak jest danych z przeszłości.

Problemy z informacją niepewną można rozwiązywać stosując transformację problemu niedeterministycznego do zbioru równoważnych problemów deterministycznych [16]. Do rozwiązywania otrzymanych w wyniku takiej transformacji problemów wykorzystuje się

następnie znane metody deterministyczne. W wyniku takiego podejścia możliwe jest uzyskanie kilku zdeterminowanych uszeregowań dopuszczalnych. Możliwy rozrzut spodziewanego rezultatu funkcji kryterialnej przedstawia się za pomocą globalnego rezultatu rozmytego powstałego w wyniku agregacji rozwiązań zdeterminowanych. Innym podejściem jest wykorzystanie takich procedur szeregujących, które uwzględniają rozmyte parametry czasowe traktując je w sposób całościowy. Podejście to wydaje się być bardziej naturalne. Ponadto efektem działania takiej procedury szeregującej jest jedno uszeregowanie, dopuszczalne dla dowolnej realizacji czasów trwania poszczególnych zadań.

Problemy szeregowania zadań z rozmytymi parametrami czasowymi zostały wcześniej poruszone w wielu publikacjach. Najważniejsze z nich dotyczyły głównie modelu PERT [13, 2], który nie uwzględnia ograniczeń zasobowych, oraz ogólnego systemu obsługi [5, 6, 11, 12]. Studia nad bardzo praktycznym problemem szeregowania zadań w warunkach niepewności - problemem programowania sieciowego - zostały zapoczątkowane przez autora i jego współpracowników w pracach [8, 9]. Niniejsza praca stanowi kontynuację tych studiów. W pracy proponuje się zastosowanie uogólnionej procedury szeregującej do rozwiązywania problemów programowania sieciowego z rozmytymi parametrami czasowymi, przy ograniczonych zasobach.

W następnym rozdziale sformułowany jest problem programowania sieciowego z rozmytymi parametrami czasowymi. Kolejny rozdział opisuje sposób modelowania informacji niepewnej, podstawowe operacje na liczbach rozmytych oraz operacje porównania wyników (liczb) rozmytych. Rozdział czwarty poświęcony jest propozycji rozmytej procedury szeregowania zadań i rozdziału zasobów dla rozważanego problemu. W rozdziale następnym przedstawiony jest numeryczny przykład działania uogólnionej procedury szeregującej. Ostatni rozdział podsumowuje pracę.

2. Sformułowanie problemu

Problem programowania sieciowego z rozmytymi parametrami typu czasowego może być charakteryzowany przez cztery komponenty [15,17]: zbiór zasobów R, zbiór zadań Z, ograniczenia kolejnościowe w zbiorze Z, oraz zbiór kryteriów oceny projektu C. Zbiór R składa się z:

- p typów zasobów odnawialnych $R_1^r,...,R_p^r$, których wykorzystanie w każdym momencie ograniczone jest do $N_1^r,...,N_p^r$ jednostek,
- v typów zasobów nieodnawialnych $R_1^n,...,R_v^n$, których zużycie w każdym momencie ograniczone jest do $B_1^n,...,B_v^n$ jednostek,
- u typów zasobów podwójnie ograniczonych $R_1^d,...,R_u^d$, których zużycie w każdym momencie ograniczone jest do $B_1^d,...,B_v^d$ jednostek, a wykorzystanie do $N_1^d,...,N_p^d$ jednostek.

Zbiór Z składa się z n zadań, które posiadają dyskretne żądania zasobowe. Każde zadanie $Z_j \in Z$ może być wykonane w jednym z w różnych trybów. Tryb wykonania m zadania Z_j zdefiniowany jest przez wektory r^r_{mkj}, r^n_{mlj}, r^d_{mhj} (k=1,...,p; l=1,...,v; h=1,...,u), których elementy określają wykorzystanie i zużycie zasobów różnych kategorii dla każdego trybu. Dla każdego trybu określony jest czas trwania $\bar{p}_{mj}$ zadania Z_j. Ponadto dla każdego zadania określony jest czas gotowości $\bar{a}_j$ oraz oczekiwany termin zakończenia $\bar{d}_j$. Zakłada się, że wszystkie czasowe parametry zadań są, w ogólności, niepewne i modelowane za pomocą liczb rozmytych (ang. fuzzy numbers).

Ograniczenia kolejnościowe reprezentowane są za pomocą acyklicznego grafu skierowanego, gdzie węzły przedstawiają zadania, a strzałki reprezentują ograniczenia kolejnościowe pomiędzy zadaniami. Zbiór C składa się z następujących czasowych i czasowo-kosztowych kryteriów oceny projektu: maksymalnego czasu wykonania projektu, maksymalnego wygładzenia zużycia zasobów, maksymalnego opóźnienia, średniego czasu przepływu, całkowitego kosztu projektu. Zauważmy, że z powodu niepewności parametrów czasowych zadań, wartości kryteriów czasowych i czasowo-kosztowych są również niepewne i wyrażone będą w kategoriach liczb rozmytych.

W ogólności, szeregowanie zadań projektu polega na takim przydziale zasobów ze zbioru R do poszczególnych zadań ze zbioru Z, by zachowując wszystkie ograniczenia osiągnąć jak *najlepszy kompromis* pomiędzy kryteriami ze zbioru C.

3. Modelowanie niepewności

W pracy tej do modelowania niepewności proponuje się zastosowanie płaskiej liczby rozmytej L-R reprezentowanej przez pięcioodcinkową krzywą łamaną. Liczbę taką będziemy definiować symbolicznie w następujący sposób: $M = (\underline{m}^\varepsilon, \underline{m}^\lambda, \underline{m}, \overline{m}, \overline{m}^\lambda, \overline{m}^\varepsilon)$. Poszczególne komponenty liczby rozmytej oznaczają zakresy możliwych rozrzutów niepewnych parametrów, które ekspert potrafi wyrazić na trzech różnych poziomach przynależności: ε - $[\underline{m}^\varepsilon, \overline{m}^\varepsilon]$, λ - $[\underline{m}^\lambda, \overline{m}^\lambda]$ oraz na poziomie 1 - $[\underline{m}, \overline{m}]$.

Jeśli $\tilde{A}$ i $\tilde{B}$ są liczbami rozmytymi w reprezentacji pięcioodcinkowej, to mogą być zdefiniowane następujące operacje:

$$\tilde{A} + \tilde{B} = \left(\underline{a}^\varepsilon + \underline{b}^\varepsilon,\ \underline{a}^\lambda + \underline{b}^\lambda,\ \underline{a} + \underline{b},\ \overline{a} + \overline{b},\ \overline{a}^\lambda + \overline{b}^\lambda,\ \overline{a}^\varepsilon + \overline{b}^\varepsilon\right),$$

$$\tilde{A} - \tilde{B} = \left(\underline{a}^\varepsilon - \underline{b}^\varepsilon,\ \underline{a}^\lambda - \underline{b}^\lambda,\ \underline{a} - \underline{b},\ \overline{a} - \overline{b},\ \overline{a}^\lambda - \overline{b}^\lambda,\ \overline{a}^\varepsilon - \overline{b}^\varepsilon\right),$$

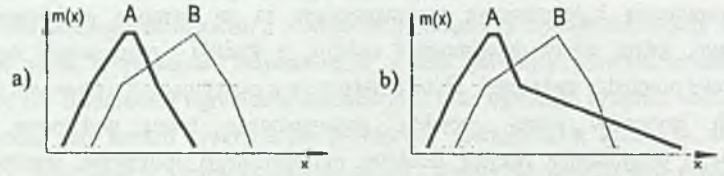
$$max(\tilde{A}, \tilde{B}) = (max(\underline{a}^\varepsilon, \underline{b}^\varepsilon), max(\underline{a}^\lambda, \underline{b}^\lambda), max(\underline{a}, \underline{b}), max(\overline{a}, \overline{b}), max(\overline{a}^\lambda, \overline{b}^\lambda), max(\overline{a}^\varepsilon, \overline{b}^\varepsilon)),$$

$$min(\tilde{A}, \tilde{B}) = (min(\underline{a}^\varepsilon, \underline{b}^\varepsilon), min(\underline{a}^\lambda, \underline{b}^\lambda), min(\underline{a}, \underline{b}), min(\overline{a}, \overline{b}), min(\overline{a}^\lambda, \overline{b}^\lambda), min(\overline{a}^\varepsilon, \overline{b}^\varepsilon)).$$

Podczas rozwiązywania problemów z danymi rozmytymi często zdarzają się sytuacje, w których należy porównać dwa lub więcej parametrów rozmytych. W tej pracy porównywanie parametrów rozmytych dokonywane jest w dwóch różnych sytuacjach:

- w algorytmie szeregowania, podczas wyznaczania następnych (rozmytych) chwil czasowych, w których rozpatrywany jest przydział zasobów do zadań,
- podczas porównywania rozmytych rezultatów optymalizacji.

Wynik porównania jest oczywisty, jeśli dwie różne liczby rozmyte nie nakładają się na siebie. Rysunek 1 przedstawia możliwe sposoby nakładania się liczb rozmytych.



Rys. 1a,b. Przykłady nakładania się liczb rozmytych
Fig. 1a, b. Examples of overlapping of two fuzzy numbers

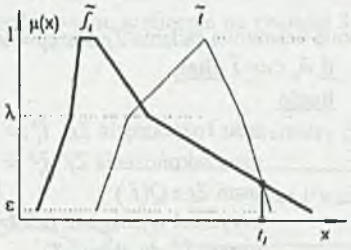
Na rysunku 1a można zauważyć, że odpowiednie wartości zarówno dolnych, jak i górnych granic jakiegokolwiek α-cięcia liczby $\bar{B}$ są większe od tych dla liczby $\bar{A}$. W tej sytuacji zachodzi relacja $max(\bar{A}, \bar{B}) = \bar{B}$. Będziemy mówić wtedy, że $\bar{B}$ jest **silnie większe lub równe** $\bar{A}$, a relacja ta będzie nazywana relacją **silnej nierówności** i będzie oznaczana jako $\bar{B} >>= \bar{A}$ [10]. Rysunek 1b przedstawia inny przypadek nakładania się liczb rozmytych $\bar{A}$ i $\bar{B}$, gdzie relacja silnej nierówności nie zachodzi. Chociaż odpowiednio $\underline{b}^{\epsilon}, \underline{b}^{\lambda}, \underline{b}, \overline{b}, \overline{b}^{\lambda} > \underline{a}^{\epsilon}, \underline{a}^{\lambda}, \underline{a}, \overline{a}, \overline{a}^{\lambda}$, to $\overline{b}^{\epsilon} < \overline{a}^{\epsilon}$. W takich sytuacjach, gdy nie ma konieczności stosowania porównania silnego, a należy rozstrzygnąć, która wartość rozmyta jest większa, stosujemy inne podejście oparte na metodzie kompensacji pól ograniczonych funkcją przynależności liczb rozmytych [14]. Relacja oparta na tej metodzie nazywana będzie relacją **słabej nierówności**. Zastosowanie obu relacji zostanie wyjaśnione w następnych rozdziałach.

4. Rozmyta procedura szeregująca

Procedury szeregujące wykorzystywane przez metody heurystyczne do rozwiązywania problemów programowania sieciowego można ogólnie podzielić na szeregowe i równoległe. Nazwa procedury szeregowej pochodzi od sekwencyjnego sposobu rozpatrywania zadań. Procedura równoległa rozważa w jednym momencie grupę zadań, które mogą rozpocząć się ze względu na spełnienie ograniczeń kolejnościowych. Główna różnica pomiędzy nimi polega na sposobie rozwiązywania konfliktów zasobowych. Jeśli zadanie nie może się rozpocząć w chwili $\bar{t}$ z powodu braku wolnych zasobów, procedura szeregowa poszukuje najwcześniejszego momentu, w którym wymagane zasoby są dostępne. W tej samej sytuacji procedura

równoległa sprawdza, czy inne zadanie (o najwyższym priorytecie) może się rozpocząć w rozpatrywanym momencie, i jeśli tak, szereguje je w chwili $\tilde{t}$.

Aby uogólnić obie procedury do wersji rozmytych, należy rozwiązać problem następstwa rozmytych chwil czasowych, w których rozpatrywany ma być przydział zasobów. Zaznaczyć tu należy, że naszym celem jest utworzenie takiej procedury, aby otrzymane uszeregowanie było dopuszczalne dla jakiejkolwiek realizacji rozmytych parametrów czasowych.



Rys. 2. Przykład uszeregowania niedopuszczalnego
Fig. 2. Example of an unfeasible fuzzy schedule

Załóżmy, że procedura rozważa przydział zasobu R do zadań w chwili $\tilde{t}$. Aby spełnić ograniczenia kolejnościowe, w chwili $\tilde{t}$ tylko te zadania mogą ubiegać się o zasób R, których wszystkie poprzedniki zostały zakończone. Załóżmy, że zadanie Z_1 jest poprzednikiem zadania Z_2, f_t jest czasem zakończenia zadania Z_1. Procedura sprawdza, czy wszystkie poprzedniki zadania Z_2 zostały zakończone przed $\tilde{t}$. W sytuacji przedstawionej na rysunku 2 relacja słabej nierówności byłaby prawdziwa, ponieważ $\tilde{t} >= f_t$.

Zauważmy jednak, że dla bardzo pesymistycznej realizacji czasu trwania zadania Z_1 (większej od t_1) otrzymane uszeregowanie byłoby niedopuszczalne, ponieważ zadanie Z_2 rozpoczęłoby się, zanim zostałoby ukończone zadanie Z_1. Przykład ten pokazuje, że wykorzystanie w procedurze szeregującej relacji słabej nierówności może prowadzić do uszeregowań niedopuszczalnych. Aby tego uniknąć, relacja ta musi zostać zastąpiona relacją silnej nierówności. Uogólnienie procedury szeregowania na przypadek rozmyty polega zatem na rozwiązaniu problemu następstwa charakterystycznych rozmytych chwil czasowych przez zastosowanie wprowadzonej w rozdziale 3 relacji silnej nierówności.

Wielu badaczy problemów programowania sieciowego zauważyło, że procedura równoległa daje z reguły lepsze rozwiązania od szeregowej. Skoncentrujemy się zatem na tej pierwszej. Poniżej przedstawiony zostanie algorytm procedury napisany w kodzie pseudo-Pascal.

procedure Rozmyta_procedura_równoległa;
begin
$\tilde{t} := \tilde{t}_0$
repeat
Utwórz zbiór $Q(\tilde{t})$ składający się z tych zadań, które nie zostały jeszcze uszeregowane i których wszystkie bezpośrednie poprzedniki zostały zakończone przed $\tilde{t}$;
for każdego Z_i z $Q(\tilde{t})$, w kolejności ich priorytetów **do**
begin
if żądania zasobowe zadania Z_i dostępności zasobów **then**
if $\tilde{a}_i <<= \tilde{t}$ **then**
begin
czas rozpoczęcia Z_i: $\tilde{t}_i^s := \tilde{t}$
czas zakończenia Z_i: $\tilde{t}_i^f := \tilde{t}_i^s + \tilde{p}_i$
usuń Z_i z $Q(\tilde{t})$
przydziel wymagane zasoby do Z_i
wstaw $\tilde{t}_i^f$ do zbioru T
end
else
wstaw $\tilde{a}_i$ do zbioru T
end
$\tilde{t} := max(\tilde{t}, \tilde{l})$
if $\tilde{l} = \tilde{t}^f$ (jest czasem zakończenia dowolnego zadania) **then**
uaktualnij dostępność zasobów
usuń $\tilde{l}$ ze zbioru T
until wszystkie zadania zostaną uszeregowane
end

gdzie:
$\tilde{l}$ - najmniejsza (w słabym sensie) wartość ze zbioru T,
zbiór $T = \{\tilde{t}_i^f : \forall\, Z_i \in S(\tilde{t})\} \cup \{\tilde{a}_i : \forall\, Z_i \in A(\tilde{t})\}$,
$S(\tilde{t})$ - zbiór zadań uszeregowanych przed $\tilde{t}$, $\tilde{t}_i^s <<= \tilde{t}$,
$A(\tilde{t})$ - podzbiór zbioru $Q(\tilde{t})$ tych zadań, które nie są gotowe w chwili $\tilde{t}$, $\tilde{a}_i \geq \tilde{t}$.

Aby zilustrować działanie uogólnionej procedury równoległej, w rozdziale 6 zamieszczony zostanie prosty przykład numeryczny.

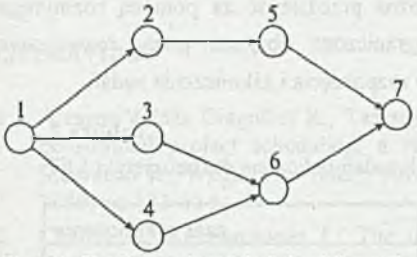
5. Heurystyczne sposoby rozwiązywania problemu programowania sieciowego

Najbardziej popularnymi metodami rozwiązywania problemu programowania sieciowego są heurystyki priorytetowe [1,3,4]. Charakteryzują się one dużą efektywnością i dają z reguły dobre rozwiązania. Lepsze rozwiązania uzyskiwane są jedynie przy użyciu procedur metaheurystycznych, jednak przy znacznie dłuższym czasie obliczeń. Adaptacja trzech najbardziej popularnych metaheurystyk: symulowanej relaksacji, przeszukiwania "tabu" oraz algorytmów genetycznych do problemu programowania sieciowego w warunkach

niepewności oraz wyniki testów obliczeniowych są treścią innej pracy przedłożonej do publikacji [7].

6. Przykład numeryczny

Rozważmy problem składający się z siedmiu zadań, pomiędzy którymi istnieją zależności kolejnościowe przedstawione graficznie na rysunku 3. Tablica 1 zawiera parametry poszczególnych zadań.



Rys. 3. Graf ograniczeń kolejnościowych przykładowego problemu
Fig. 3. The precedence graph of an example problem

Tablica 1

Parametry poszczególnych zadań

nr	czas trwania	żądanie zasobowe
1	42, 45, 50, 54, 54, 61	8
2	36, 44, 46, 52, 61, 66	17
3	49, 51, 54, 62, 71, 79	12
4	34, 40, 45, 46, 53, 59	3
5	16, 22, 30, 32, 33, 42	13
6	43, 49, 51, 57, 57, 57	7
7	52, 52, 58, 62, 65, 69	16

Dla ułatwienia prezentacji, czasy gotowości poszczególnych zadań są równe zeru. Każde zadanie wymaga do swojego wykonania pewnej ilości zasobu odnawialnego, którego dostępność jest ograniczona w każdym momencie projektu do 30 jednostek. Zaprezentowane zostanie tu działanie uogólnionej procedury równoległej na przykładzie rozmytej heurystyki priorytetowej LFT (Late Finish Time).

W pierwszym etapie zadania zostały uporządkowane na liście priorytetowej wg rosnących (w sensie słabej nierówności) wartości najpóźniejszych rozmytych czasów zakończenia zadania, pochodzących z metody ścieżki krytycznej. W takiej właśnie kolejności zadania rozpatrywane są przez procedurę szeregującą. Kolejność rozpatrywania przydziału zasobów do zadań jest zatem następująca: 1, 3, 4, 2, 6, 5, 7. Poniżej przedstawione zostanie kilka pierwszych iteracji procedury.

Charakterystyczną cechą procedury równoległej jest rozpatrywanie przydziału zasobów do zadań w następujących po sobie momentach czasowych $\tilde{t}$ realizacji projektu. W chwili $\tilde{t} = (0, 0, 0, 0, 0, 0)$ zbiór $Q(\tilde{t})$ składa się z zadania Z_1, które zostaje uszeregowane,ponieważ wymagane zasoby są dostępne. Dostępność zasobów zostaje zaktualizowana do R=30-8=22, a czas zakończenia zadania Z_1, $\tilde{t}_1^f = (42, 45, 50, 54, 54, 61)$, zostaje wstawiony do zbioru T

jako możliwy termin rozpoczęcia jego następników. Zbiór $Q(\tilde{t})$ jest pusty, ponieważ Z_1 zostało uszeregowane. Następny $\tilde{t} = max(\tilde{t}\,;\, \tilde{t}_1^f) = \tilde{t}_1^f$. Czas $\tilde{t}_1^f$ zostaje usunięty ze zbioru T. W nowym $\tilde{t}$ zadanie Z_1 zwalnia zasób, stąd dostępność R=30. Tym razem zbiór $Q(\tilde{t})$ składa się z zadań: Z_3, Z_4, Z_2 (kolejność z listy priorytetowej). W aktualnej chwili $\tilde{t}$ uszeregowane mogą być tylko zadania Z_3 i Z_4, dla Z_2 nie wystarczy zasobu, bo R=15, a Z_2 wymaga 17 jednostek. Czasy zakończenia uszeregowanych zadań umieszczane są w zbiorze T, $\tilde{t}_3^f$ = (91, 96, 104, 116, 125, 140), $\tilde{t}_4^f$ = (76, 85, 95, 100, 107, 120). Ze zbioru T zostaje wybrana i usunięta najmniejsza wartość tj. $\tilde{t}_4^f$, po czym zostaje obliczona nowa wartość $\tilde{t} = max(\tilde{t}\,,\, \tilde{t}_4^f) = \tilde{t}_4^f$. Następnie zostaje uaktualniona dostępność zasobów na R=18 i zadanie Z_2 może zostać uszeregowane. Procedura ta powtarza się, aż wszystkie zadania zostaną uszeregowane.

Uzyskane w ten sposób uszeregowanie można przedstawić za pomocą rozmytego wykresu Gantta [8], jednak ze względu na ograniczoną objętość pracy rozwiązanie przedstawiono jedynie w tablicy 2 zawierającej czasy rozpoczęcia i zakończenia zadań.

Tablica 2

Czasy rozpoczęcia i zakończenia poszczególnych zadań obliczone dla heurystyki LFT

nr	czas rozpoczęcia	czas zakończenia
1	0, 0, 0, 0, 0, 0	42, 45, 50, 54, 54, 61
2	76, 85, 95, 100, 107, 120	112, 129, 141, 152, 168, 186
3	42, 45, 50, 54, 54, 61	91, 96, 104, 116, 125, 140
4	42, 45, 50, 54, 54, 61	76, 85, 95, 100, 107, 120
5	112, 129, 141, 152, 168, 186	128, 151, 171, 184, 201, 228
6	91, 96, 104, 116, 125, 140	134, 145, 155, 173, 182, 197
7	134, 151, 171, 184, 201, 228	186, 203, 229, 246, 266, 297

7. Wnioski

W pracy zaprezentowano uogólnienie na przypadek rozmyty procedur szeregowania zadań i rozdziału ograniczonych zasobów w problemie programowania sieciowego, gdzie zadania mogą być wykonywane na różne sposoby i wymagają do realizacji zasobów różnych kategorii. Uogólnienie to polega na oryginalnym rozwiązaniu problemu następstwa charakterystycznych chwil czasowych wyrażonych w kategoriach liczb rozmytych oraz problemu porównywania wartości rozmytych. Ostatnia część zawiera numeryczny przykład działania uogólnionej procedury szeregującej.

Przedstawiony w niniejszej pracy model wydaje się mieć szerokie zastosowania praktyczne. Może to być np. zarządzanie realizacją projektów rolniczych [9], gdzie czasów gotowości niektórych zadań zależących np. od warunków atmosferycznych nie można określić

w sposób precyzyjny, lub zarządzanie realizacją projektów programistycznych [8], w których menedżer projektu nie potrafi dokładnie oszacować czasu trwania niektórych zadań.

Podziękowania

Autor pragnie podziękować Fundacji na Rzecz Nauki Polskiej za stypendium, które otrzymuje w roku 1996, Poznańskiemu Centrum Superkomputerowo-Sieciowemu, na którego zasobach wykonane zostały testy obliczeniowe, oraz swojemu opiekunowi naukowemu prof. Romanowi Słowińskiemu, którego cenne uwagi wywarły wpływ na postać tej pracy. Praca została zrealizowana w ramach projektu badawczego KBN nr 8 T11F 010 08p02.

LITERATURA

1. Alvares-Valdés Olaguíbel R., Tamarit Goerlich J.M.: Heuristic algorithms for resource-constrained project scheduling: a review and an empirical analysis. Section I.5 in: Słowiński R., Weglarz J. (eds.), Advances in Project Scheduling, Elsevier, Amsterdam, 1989, pp.113-134.
2. Chanas S., Kamburowski J.: The use of fuzzy variables in PERT. Fuzzy Sets and Systems 5, pp.11-19, 1981.
3. Cooper D.F.: Heuristics for scheduling resource-constrained projects: an experimental investigation. Management Science 22, 1976, pp.1186-1194.
4. Davis E.W., Patterson J.H.: A comparison of heuristic and optimal solution in resource-constrained project scheduling. Management Science 21 (8), 1975, pp.944-955.
5. Fortemps P.: Jobshop scheduling with imprecise durations: a fuzzy approach. Tech. rep. Faculte Politechnique de Mons - Belgium, submitted, 1995.
6. Grabot B., Geneste L.: Dispathing rules in scheduling: a fuzzy approach. Int. J. Prod. Res., vol. 32 no. 4, pp.903-915, 1994.
7. Hapke M., Kominek P., Słowiński R.: Metaheurystyczne procedury rozwiązywania problemu programowania sieciowego w warunkach niepewności. Praca przedłożona do publikacji w Zeszytach Naukowych Politechniki Śląskiej z.117, prezentowana na X KKADPP, Zakopane, 18-21 września 1996.
8. Hapke M., Jaszkiewicz A., Słowiński R.: Fuzzy Project Scheduling System for Software Development. Fuzzy Sets and Systems 21, 1994, pp.101-117.
9. Hapke M., Słowiński R.: A DSS for resource-constrained project scheduling under uncertainty. Journal of Decision Systems 2, 1993, pp.111-128.
10. Hapke M., Słowiński R.: Fuzzy Priority Heuristics for Project Scheduling. Fuzzy Sets and Systems, 1996 (w druku).
11. Ishii H.: Fuzzy combinatorial optimization. Japanese Journal of Fuzzy Theory and Systems, vol. 4, no. 1, 1992.
12. Ishii H.,Tada M., Masuda T.: Two scheduling problems with fuzzy due dates. Fuzzy Sets and Systems, 46, pp.339-347, 1992.
13. Prade H.: Using fuzzy set theory in a scheduling problem. Fuzzy Sets and Systems 2, pp.153-165, 1979.

14. Roubens M.: Inequality constraints between fuzzy numbers and their use in mathematical programming. Section 7 in: R. Słowiński, J. Teghem Eds., Stochastic Versus Fuzzy Approaches to Multiobjective Mathematical Programming under Uncertainty (Kluwer Academic Publishers, Dordrecht, 1990, pp.321-330.
15. Słowiński R.: Multiobjective network scheduling with efficient use of renewable and non renewable resources. Europ. J. Opl. Res 7, pp.265-273, 1981.
16. Słowiński R., Teghem T. (eds.): Stochastic Versus Fuzzy Approaches to Multiobjective Mathematical Programming under Uncertainty. Kluwer Academic Publishers, Dordrecht, 1990.
17. Węglarz J.: On certain models of resource allocation problems. Kybernetes 9, 1, pp.61-66, 1980.

Recenzent:Dr inż. Jerzy Mościński

Wpłynęło do Redakcji do 30.06.1996 r.

Abstract

The paper considers quite a general class of nonpreemptive project scheduling problems with renewable, nonrenewable and doubly-constrained resources, multiple performing modes of activities, precedence constraints in the form of an activity network. Activity time parameters and thus time-cost criteria are uncertain and modelled by means of fuzzy numbers.

The author presents a generalized fuzzy parallel scheduling procedure which, instead of transforming the fuzzy problem to a set of its deterministic associates, handles the fuzzy data directly, in particular steps of the procedure. While solving problems with fuzzy data there often appear situations in which two or more fuzzy numbers are to be compared. In the problem considered in this paper, the comparison of fuzzy numbers takes place in two situations: while solving precedence problems of fuzzy events and while comparing fuzzy results of optimization. In this paper, different comparison methods are used in two situations. The precedence problem of fuzzy events is proposed to be solved by strong inequality relation and the problem of comparison of fuzzy optimization results is proposed to be solved by weak inequality relation.