

Na prawach rękopisu

Instytut Organizacji i Zarządzania
Politechniki Wrocławskiej
Raport serii PRE nr 14

**PROBLEMY SZEREGOWANIA PRAC
NA POJEDYNCZEJ MASZYNIE
Z NIEPRECYZYJNIE OKREŚLONYMI
PARAMETRAMI**
(rozprawa doktorska)

Adam Kasperski

Praca doktorska
napisana pod kierunkiem
prof. nadzw. dr hab. Stefana Chanasa

Słowa kluczowe:
planowanie, szeregowanie prac,
liczba rozmyta, rozkład możliwości

Wrocław 2001

Spis rzeczy

Wstęp	4
1 Podstawowe pojęcia z zakresu teorii zbiorów rozmytych i liczb rozmytych	8
1.1 Pojęcie zbioru rozmytego i liczby rozmytej	8
1.2 Szczególne przypadki liczb rozmytych	10
1.3 Operacje na liczbach rozmytych	13
1.4 Interpretacja liczby rozmytej	15
1.5 Indeksy Dubois i Prade	18
1.6 Wartość średnia liczby rozmytej	30
2 Rozmyte problemy szeregowania na pojedynczej maszynie	34
2.1 Przegląd klasycznych problemów szeregowania prac na pojedynczej maszynie	34
2.2 Definicja rozmytego problemu szeregowania na pojedynczej maszynie	38
2.3 Charakterystyka funkcji kosztu harmonogramu w rozmytym problemie szeregowania	41
2.4 Konstrukcja różnych funkcji kosztu harmonogramu w rozmytym problemie szeregowania	49
2.5 Problem $1 prec, FN F_{max}$ z f -monotonicznymi funkcjami kosztów zakończenia prac	52
2.5.1 Konstrukcja uogólnionego algorytmu Lawlera	53
2.5.2 Problem $1 prec, L - R max\{Poss(Ness)(\tilde{C}_i > \tilde{d}_i)\}$	56
2.5.3 Problem $1 prec, L - R max\{E(w_i\tilde{L}_i)\}$	57

2.5.4	Problem $1 prec, PL - PR max\{E(w_i\tilde{T}_i)\}$	59
2.6	Problem $1 TR max\{Ness(\tilde{C}_i \neq \tilde{d}_i)\}$	62
2.7	Problem $1 FN E(\tilde{max}\{\tilde{T}_i\})$	66
2.8	Problem $1 FN E(\tilde{max}\{\tilde{L}_i\})$	78
2.9	Problem $1 prec, FN Poss(\tilde{F} > \tilde{G})$	85
2.9.1	Problem $1 prec, L - R Poss(\tilde{max}\{w_i\tilde{L}_i\} > \tilde{G})$	89
2.9.2	Problem $1 L - R Poss(\sum w_i\tilde{C}_i > \tilde{G})$	92
2.10	Problem $1 FN \sum Poss(Ness)(\tilde{C}_i > \tilde{d}_i)$	95
2.11	Problemy $1 prec, L - R \sum E(w_i\tilde{C}_i)$ i $1 TG \sum E(w_i\tilde{T}_i)$	97
3	Ocena optymalności harmonogramu w problemach szeregowania prac na pojedynczej maszynie z rozmytymi parametrami	102
3.1	Koncepcja oceny stopnia optymalności harmonogramu	103
3.2	Przedziałowy problem $i - 1 prec f_{max}$	106
3.2.1	Warunki konieczne i dostateczne optymalności harmono- gramu w problemie $1 prec f_{max}$	106
3.2.2	Problem $i - 1 prec max\{w_iL_i\}$	112
3.2.3	Problem $i - 1 prec max\{w_iT_i\}$	120
4	Metody heurystyczne dla NP-trudnych rozmytych problemów szeregowania	124
4.1	Algorytm ewolucyjny dla rozmytego problemu szeregowania . . .	125
4.2	Inne heurystyki dla rozmytego problemu szeregowania	128
4.3	Przykłady numeryczne dla $1 TR \sum E(w_i\tilde{T}_i)$	129
	Zakończenie	142
	Literatura	147

Spis algorytmów

1	Obliczanie indeksu $Poss(\tilde{A} \geq \tilde{B})$ z dokładnością ε	29
2	Obliczanie indeksu $Poss(\tilde{A} > \tilde{B})$ z dokładnością ε	29
3	Algorytm Lawlera dla problemu $1 prec f_{max}$	53
4	Uogólniony algorytm Lawlera dla problemu $1 prec, FN F_{max}$. . .	54
5	Algorytm dla problemu $1 prec, L - R max\{Poss(Ness)(\tilde{C}_i > \tilde{d}_i)\}$	56
6	Algorytm dla problemu $1 prec, L - R max\{E(w_i\tilde{L}_i)\}$	58
7	Algorytm dla problemu $1 prec, PL - PR max\{E(w_i\tilde{T}_i)\}$	61
8	T-transformacja $1 MT(\sigma, k, n)$ do $1 MT(\sigma, k - 1, n)$	75
9	T-transformacja $1 MT(\sigma, 1, n)$ do $1 ML$	80
10	Algorytm dla problemu $1 prec, FN Poss(\tilde{F} > \tilde{G})$	87
11	Funkcja $EXISTS(\lambda)$ dla $1 prec, L - R Poss(max\{w_i\tilde{L}_i\} > \tilde{G})$. .	90
12	Funkcja $EXISTS(\lambda)$ dla $1 L - R Poss(\sum w_i\tilde{C}_i > \tilde{G})$	93
13	Obliczenie stopnia optymalności harmonogramu p w problemie S .	105
14	Algorytm dla problemu $i - 1 prec max\{w_iL_i\}$	114
15	Algorytm dla problemu $i - 1 prec max\{w_iT_i\}$	123
16	Ogólny schemat algorytmu ewolucyjnego	126

Wstęp

Szeregowanie prac jest jednym z najważniejszych i najbardziej obszernych działów badań operacyjnych. Uzyskane w tym zakresie wyniki mają duże znaczenie w zastosowaniach praktycznych. Klasa wszystkich problemów szeregowania jest bardzo zróżnicowana. Obejmuje ona zagadnienia różniące się liczbą maszyn, charakterystyką prac, funkcją celu oraz zbiorem zadanych parametrów. Szczególne znaczenie w szeregowaniu prac mają wyniki dotyczące złożoności obliczeniowej pewnych klas problemów. Niestety, okazuje się, że tylko dla nielicznych z nich można skonstruować efektywne obliczeniowo algorytmy wielomianowe. Stwarza to potrzebę konstruowania oraz badania metod przybliżonych, w tym różnego rodzaju heurystyk.

Literatura poświęcona problemom szeregowania jest bardzo rozległa i nie można jej zwięźle i jednolicie usystematyzować. Na szczególną uwagę zasługuje książka Bruckera [3], w której są podane algorytmy dla większości najważniejszych problemów szeregowania. Zaprezentowany jest tam również aktualny stan wiedzy na temat złożoności obliczeniowej wszystkich najważniejszych klas tych problemów.

W klasycznych, rozpatrywanych do tej pory, problemach szeregowania zakłada się, że wszystkie parametry są zdeterminowane. Założenie to może jednak okazać się krępujące w praktyce. Często, zwłaszcza w zastosowaniach przemysłowych, ma miejsce sytuacja, w której jest niemożliwe precyzyjne ustalenie wszystkich parametrów. Mogą one być obarczone niepewnością wynikającą z przyczyn niezależnych od decydenta. Naturalnym postępowaniem w tego typu sytuacjach jest zastosowanie teorii rachunku prawdopodobieństwa i określenie parametrów w postaci zmiennych losowych o zadanych rozkładach prawdopodobieństwa. Z takim podejściem można spotkać się w literaturze. Uzyskane wyniki pokazują jednak, że nie jest ono zbyt praktyczne, ponieważ często prowadzi do problemów bardzo trudnych z obliczeniowego punktu widzenia. Z drugiej strony, określenie adekwatnych rozkładów prawdopodobieństwa dla los-

wych parametrów może być również trudne i wiązać się ze znacznymi kosztami eksperymentów statystycznych.

W pracy skoncentrowano się na podejściu, które przypomina pod pewnymi względami podejście probabilistyczne i nie prowadzi do tak dużych trudności obliczeniowych. Istotą tego podejścia jest zastosowanie do modelowania nieprecyzyjnych parametrów zbiorów rozmytych. Dlatego będzie ono nazywane w pracy, przez analogię do podejścia probabilistycznego, podejściem rozmytym. Pojęcie zbioru rozmytego zostało wprowadzone przez Zadeha w 1965 roku [36]. Od tego czasu teoria zbiorów rozmytych, w tym liczb rozmytych, została w znacznym stopniu rozwinięta i znalazła liczne zastosowania w badaniach operacyjnych. W ostatniej dekadzie pojawiła się pewna liczba publikacji, w których zbiory rozmyte zostały zastosowane w teorii szeregowania. W szczególności ukazały się prace poświęcone podejściu rozmytemu do szeregowania prac na pojedynczej maszynie. Na przykład, w pracy [33] rozważany jest problem, w którym pożądane terminy zakończenia prac są rozmyte a za optymalne rozwiązanie przyjmuje się harmonogram, dla którego stopień satysfakcji decydenta z terminowości wykonania prac jest największy. W pracy [15] jest badany problem, w którym pożądane terminy zakończenia prac są rozmyte a dodatkowymi zmiennymi decyzyjnymi są szybkości maszyn. W pracy [16] jest rozpatrywany z kolei problem szeregowania na pojedynczej maszynie, w którym relacja poprzedzania między pracami jest rozmyta. Ukazały się również publikacje poświęcone rozmytym problemom szeregowania na więcej niż jednej maszynie (np: [17],[18],[34]). Szeroki przegląd różnych podejść do rozmytych problemów szeregowania można znaleźć w książce [32].

Podejście rozmyte w szeregowaniu prac budzi obecnie na tyle duże zainteresowanie, że są organizowane specjalne konferencje, poświęcone wyłącznie temu zagadnieniu. Na przykład, w 2000 roku odbyła się w Mons międzynarodowa konferencja *EUROFUSE Workshop on Scheduling and Planning*, poświęcona wyłącznie rozmytym problemom szeregowania. Prezentowane były na niej między innymi rozwiązania, które zostały już zastosowane w przemyśle, co jest dowodem na to, że opracowane metody mogą w istotny sposób usprawnić funkcjonowanie różnych obszarów działalności przedsiębiorstw.

Ponieważ podejście rozmyte do szeregowania prac jest stosunkowo nowe, więc trudno jest usystematyzować wszystkie wyniki, które do tej pory uzyskano. Można natomiast zauważyć, że zdecydowanie dominują problemy, w których liczby rozmyte są traktowane jako rozkłady preferencji, określające stopień zadowolenia lub niezadowolenia decydenta z przyjęcia określonych wartości przez parametry

problemu. Rozmytymi parametrami są najczęściej pożądane terminy zakończenia prac, które najłatwiej jest w ten sposób zinterpretować. Badania, które zostały do tej pory przeprowadzone, są raczej wycinkowe. Brakuje systematycznego podejścia, w którym byłaby zdefiniowana ogólna postać rozmytego problemu szeregowania i podane jego podstawowe własności. Brakuje również ogólniejszych wyników dotyczących złożoności obliczeniowej, które mają kluczowe znaczenie w zastosowaniach praktycznych.

Niniejsza praca stanowi kontynuację oraz w pewnym stopniu rozwinięcie podejścia rozmytego do teorii szeregowania. Rozważane są problemy, w których parametry są zadane nieprecyzyjnie w postaci liczb rozmytych. Liczby rozmyte są interpretowane jako rozkłady możliwości a nie jako rozkłady preferencji, co sprawia, że podejście prezentowane w pracy jest podobne do podejścia probabilistycznego. Rozważania ograniczono wyłącznie do klasy problemów na pojedynczej maszynie. Jest to obszerna klasa, zawierająca wiele ważnych problemów, z których znaczna część ma wielomianową złożoność obliczeniową.

Głównym celem pracy jest sformułowanie problemu szeregowania prac na pojedynczej maszynie z nieprecyzyjnie zadanymi parametrami, podanie jego ogólnych własności oraz analiza złożoności obliczeniowej różnych szczególnych przypadków takiego problemu. Dla każdego przypadku, który ma złożoność wielomianową zostanie podany odpowiedni algorytm. Dodatkowym celem pracy jest pokazanie, że również problemy trudne obliczeniowo można próbować rozwiązywać za pomocą metod heurystycznych.

Rozdział pierwszy jest poświęcony podstawowym pojęciom dotyczącym liczb rozmytych. Większość z tych pojęć jest znana ale zostały tam również podane oraz udowodnione nowe fakty, które są potrzebne w dalszej części pracy. Omówiono również interpretację liczby rozmytej, która jest przyjęta w analizowanych w pracy problemach szeregowania.

W rozdziale drugim jest zdefiniowane pojęcie rozmytego problemu szeregowania na pojedynczej maszynie. Problem ten polega na wyznaczeniu harmonogramu, dla którego wartość funkcji kosztu jest najmniejsza, przy czym funkcja ta zależy od rozmytych czasów zakończenia prac. Rozmytość czasów zakończenia prac wynika z rozmytości parametrów problemu (tj. czasów trwania i pożądanych terminów zakończenia prac). Podano kilka ważnych własności rozmytego problemu szeregowania, w szczególności zdefiniowano klasę funkcji kosztu, dla których problem ten ma zawsze optymalne rozwiązanie o prostej strukturze. Największa część rozdziału drugiego jest poświęcona konkretnym przypadkom

rozmytego problemu szeregowania. Bada się tam złożoność obliczeniową kilkunastu nowych problemów. Dla każdego problemu o złożoności wielomianowej opracowano odpowiedni algorytm, a dla każdego problemu NP-trudnego zamieszczono dowód NP-trudności. Większość badanych problemów została zilustrowana odpowiednimi przykładami numerycznymi.

W rozdziale trzecim zaprezentowano odmienne podejście do problemów szeregowania z rozmytymi parametrami. Jeżeli parametry te nie są dokładnie znane, to zakłada się, że nie można jednoznacznie stwierdzić, które rozwiązanie będzie optymalne. Można natomiast mówić o stopniu optymalności danego rozwiązania, który określa możliwość, że rozwiązanie to będzie optymalne. W rozdziale przedstawiono koncepcję stopnia optymalności rozwiązania w problemie szeregowania z rozmytymi parametrami oraz podano efektywne algorytmy służące do obliczania tego stopnia dla rozwiązań w dwóch konkretnych problemach.

W rozdziale czwartym pokazano jeden ze sposobów radzenia sobie z problemami NP-trudnymi. Zaprezentowano w nim algorytm ewolucyjny, za pomocą którego można w sposób przybliżony rozwiązywać większość NP-trudnych problemów, zdefiniowanych w rozdziale drugim. Przedstawiono argumenty za tym, że algorytm ewolucyjny daje w rozsądnym czasie rozwiązanie bliskie optymalnemu. Przeprowadzono eksperymenty numeryczne na stosunkowo dużych problemach, w których liczba prac wynosi 50 i 100.

Ostatni rozdział stanowi podsumowanie, które zawiera między innymi zestawienie wyników uzyskanych w pracy.

Rozdział 1

Podstawowe pojęcia z zakresu teorii zbiorów rozmytych i liczb rozmytych

W rozdziale tym zostaną przedstawione podstawowe definicje i fakty związane ze zbiorami rozmytymi oraz liczbami rozmytymi, które są szczególnym przypadkiem zbiorów rozmytych. Zostanie również podana interpretacja liczby rozmytej, która jest przyjęta w dalszej części pracy, dotyczącej problemów szeregowania. Rozdział został opracowany na podstawie prac [7], [10],[13], [11], [20], [27] i [37]. W rozdziale podane są również dowody nowych twierdzeń i własności liczb rozmytych, które są potrzebne w dalszej części pracy.

1.1 Pojęcie zbioru rozmytego i liczby rozmytej

Definicja 1.1. *Zbiorem rozmytym $\tilde{A}$ w przestrzeni X nazywamy zbiór par uporządkowanych:*

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) : x \in X\},$$

gdzie $\mu_{\tilde{A}} : X \rightarrow [0, 1]$ jest funkcją przynależności zbioru rozmytego $\tilde{A}$. Wartość tej funkcji dla każdego elementu $x \in X$ określa stopień przynależności tego elementu do zbioru $\tilde{A}$.

Jeżeli funkcja przynależności zbioru $\tilde{A}$ przyjmuje wartości dyskretne ze zbioru $\{0, 1\}$, to wówczas $\tilde{A}$ jest klasycznym zbiorem A określonym w X , a jego funkcja przynależności $\mu_{\tilde{A}}(x)$ jest równoważna funkcji charakterystycznej $\chi_A(x)$ zbioru A . Zbiór rozmyty może być więc traktowany jako uogólnienie klasycznego pojęcia

zbioru. Charakterystyczną cechą zbioru rozmytego $\tilde{A}$ jest fakt, że mogą istnieć pewne elementy należące do przestrzeni X , które tylko częściowo należą do $\tilde{A}$. W przypadku klasycznego zbioru każdy element z przestrzeni X albo należy albo nie należy do zbioru.

Relacje równości i zawierania dla zbiorów rozmytych są określone następująco:

Definicja 1.2. *Jeżeli $\tilde{A}, \tilde{B}, \tilde{C}$ są zbiorami rozmytymi w przestrzeni X , to:*

- 1) $\tilde{A} = \tilde{B} \Leftrightarrow \forall_{x \in X} \mu_{\tilde{A}}(x) = \mu_{\tilde{B}}(x)$ *równość,*
- 2) $\tilde{A} \subseteq \tilde{B} \Leftrightarrow \forall_{x \in X} \mu_{\tilde{A}}(x) \leq \mu_{\tilde{B}}(x)$ *inkluzja.*

Operacje dopełnienia, sumy oraz iloczynu dla zbiorów rozmytych są określone następująco:

Definicja 1.3. *Jeżeli $\tilde{A}, \tilde{B}, \tilde{C}$ są zbiorami rozmytymi w przestrzeni X , to:*

- 1) $\tilde{A} = \tilde{B}' \Leftrightarrow \forall_{x \in X} \mu_{\tilde{A}}(x) = 1 - \mu_{\tilde{B}}(x)$ *dopełnienie,*
- 2) $\tilde{C} = \tilde{A} \cup \tilde{B} \Leftrightarrow \forall_{x \in X} \mu_{\tilde{C}}(x) = \max\{\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)\}$ *suma,*
- 3) $\tilde{C} = \tilde{A} \cap \tilde{B} \Leftrightarrow \forall_{x \in X} \mu_{\tilde{C}}(x) = \min\{\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)\}$ *iloczyn.*

W praktycznych zastosowaniach przydaje się pojęcie λ -poziomu zbioru rozmytego, które jest określone następująco:

Definicja 1.4. λ -poziomem zbioru rozmytego $\tilde{A}$, gdzie $\lambda \in (0, 1]$, nazywamy zbiór:

$$\tilde{A}^\lambda = \{x \in X : \mu_{\tilde{A}}(x) \geq \lambda\}.$$

λ -poziom zbioru rozmytego $\tilde{A}$ jest więc klasycznym podzbiorem przestrzeni X , do którego należą te elementy, których stopień przynależności do $\tilde{A}$ jest nie mniejszy niż λ .

Szczególnym przypadkiem zbioru rozmytego jest liczba rozmyta, która jest określona w następujący sposób:

Definicja 1.5. *Liczbą rozmytą $\tilde{A}$ nazywamy zbiór rozmyty określony w przestrzeni liczb rzeczywistych $\mathbb{R}$, którego funkcja przynależności spełnia następujące warunki:*

- 1) *Normalność:* $\exists_{x \in \mathbb{R}} \mu_{\tilde{A}}(x) = 1,$
- 2) *Półciągłość z góry:* $\forall_c \{x : \mu_{\tilde{A}}(x) \geq c\}$ *jest zbiorem domkniętym,*
- 3) *Wypukłość:* $\forall_{x, y \in \mathbb{R}, \lambda \in [0, 1]} \mu_{\tilde{A}}(\lambda x + (1 - \lambda)y) \geq \min\{\mu_{\tilde{A}}(x), \mu_{\tilde{A}}(y)\}.$

Zbiór wszystkich liczb rozmytych będziemy oznaczali symbolem $FN(\mathfrak{R})$.

Z definicji 1.5 wynika, że funkcja przynależności liczby rozmytej $\tilde{A}$ nie musi być funkcją ciągłą (warunek półciągłości z góry jest słabszy od warunku ciągłości). Ponadto musi istnieć co najmniej jedna liczba rzeczywista, dla której stopień przynależności do $\tilde{A}$ wynosi 1.

Liczby rozmyte posiadają następującą własność, która jest ważna w ich praktycznych zastosowaniach:

Własność 1.1. *Jeżeli $\tilde{A}$ jest liczbą rozmytą, to dla każdego $\lambda \in (0, 1]$ $\tilde{A}^\lambda$ jest domkniętym przedziałem $[\underline{a}(\lambda), \bar{a}(\lambda)]$ lub lewostronnie domkniętym przedziałem $[\underline{a}(\lambda), \infty)$ lub prawostronnie domkniętym przedziałem $(-\infty, \bar{a}(\lambda)]$ lub całym zbiorem liczb rzeczywistych $\mathfrak{R}$.*

W kolejnych rozdziałach pracy będziemy używali oznaczenia $\tilde{A}^\lambda = [\underline{a}(\lambda), \bar{a}(\lambda)]$ pamiętając, że możliwy jest przypadek, w którym $\underline{a}(\lambda) = -\infty$ lub $\bar{a}(\lambda) = \infty$. Korzystając z własności funkcji przynależności liczby rozmytej można pokazać, że funkcje $\underline{a}(\lambda)$ i $\bar{a}(\lambda)$ dla $\lambda \in (0, 1]$ są lewostronnie ciągłe.

Szczególne znaczenie w pracy będą miały dodatnie oraz nieujemne liczby rozmyte, które są określone następująco:

Definicja 1.6. *Liczba rozmyta $\tilde{A} \in FN(\mathfrak{R})$ jest nieujemna jeżeli $\mu_{\tilde{A}}(x) = 0$ dla każdego $x < 0$.*

Definicja 1.7. *Liczba rozmyta $\tilde{A} \in FN(\mathfrak{R})$ jest dodatnia jeżeli $\mu_{\tilde{A}}(x) = 0$ dla każdego $x \leq 0$.*

1.2 Szczególne przypadki liczb rozmytych

Definicja 1.5 jest najbardziej ogólną definicją liczby rozmytej. W praktycznych zastosowaniach najczęściej wykorzystuje się pewne, szczególne klasy tych liczb.

Definicja 1.8. *Liczbą przedziałową nazywamy liczbę rozmytą $\tilde{A}$, której funkcja przynależności jest określona następująco:*

$$\mu_{\tilde{A}}(x) = \begin{cases} 1 & \text{dla } x \in [\underline{a}, \bar{a}], \\ 0 & \text{dla } x \in (-\infty, \underline{a}) \cup (\bar{a}, \infty), \end{cases} \quad (1.1)$$

gdzie $-\infty \leq \underline{a} \leq \bar{a} \leq \infty$. Liczbę przedziałową oznaczamy $\tilde{A} = [\underline{a}, \bar{a}]$.

Jeżeli $\tilde{A}$ jest liczbą przedziałową, to dla każdego $\lambda \in (0, 1]$ $\tilde{A}^\lambda = [\underline{a}, \bar{a}]$. Warto zauważyć, że każda liczba rzeczywista może być traktowana jako szczególny przypadek liczby przedziałowej, dla której $\underline{a} = \bar{a}$. Klasa liczb przedziałowych zawiera wszystkie domknięte przedziały zbioru liczb rzeczywistych, łącznie z przedziałami postaci $(-\infty, a]$, $[a, \infty)$ i $(-\infty, \infty)$. Z historycznego punktu widzenia liczby przedziałowe były wykorzystywane zanim określono ogólniejsze pojęcie liczby rozmytej.

W praktycznych zastosowaniach najczęściej wykorzystuje się liczby rozmyte, których funkcja przynależności jest ciągła. Niech $L(y)$ i $R(y)$ będą ciągłymi, nierosnącymi funkcjami, określonymi na $[0, +\infty)$, ściśle malejącymi do 0 w tych podprzedziałach przedziału $[0, +\infty)$, w których są dodatnie i spełniającymi warunek $L(0) = R(0) = 1$.

Definicja 1.9. *Liczbą rozmytą typu L-R nazywamy liczbę rozmytą $\tilde{A}$, której funkcja przynależności $\mu_{\tilde{A}}$ jest określona następująco:*

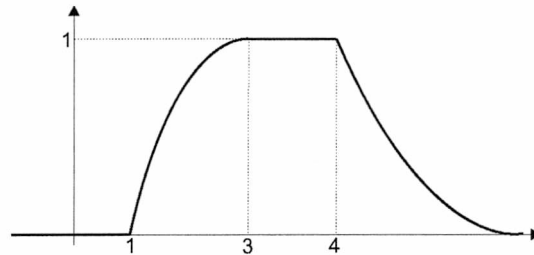
$$\mu_{\tilde{A}}(x) = \begin{cases} 1 & \text{dla } x \in [\underline{a}, \bar{a}], \\ L\left(\frac{\underline{a}-x}{\alpha_A}\right) & \text{dla } x \leq \underline{a}, \\ R\left(\frac{x-\bar{a}}{\beta_A}\right) & \text{dla } x \geq \bar{a}, \end{cases} \quad (1.2)$$

gdzie $\alpha_A, \beta_A > 0$. Liczbę rozmytą typu L-R oznaczamy $\tilde{A} = (\underline{a}, \bar{a}, \alpha_A, \beta_A)_{L-R}$.

Funkcje L i R są nazywane funkcjami kształtu. Przykładami takich funkcji są funkcje:

Liniowa:	$LIN(y) = \max\{0, 1 - y\},$
Potęgowa:	$POW_p(y) = \max\{0, 1 - y^p\}, p \geq 1,$
Wykładnicza:	$EXP_p(y) = e^{-py}, p \geq 1,$
Wymierna:	$RAT_p(y) = \frac{1}{1+y^p}, p \geq 1.$

Przykład liczby rozmytej typu $L - R$ jest przedstawiony na rysunku 1.1.



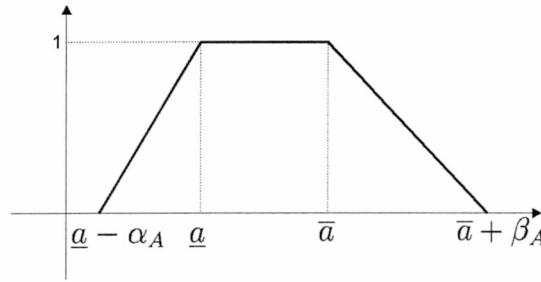
Rys. 1.1: Liczba rozmyta $\tilde{A} = (3, 4, 2, 1)_{POW_2-EXP_1}$

Liczbę rozmytą typu $L-R$, dla której L i R są funkcjami liniowymi nazywamy **liczbą trapezową** (patrz rys. 1.2). Liczby trapezowe będziemy oznaczali

$$\tilde{A} = (\underline{a}, \bar{a}, \alpha_A, \beta_A).$$

Szczególnym przypadkiem liczby trapezowej jest **liczba trójkątna**, dla której $\underline{a} = \bar{a}$. Liczby trójkątne będziemy oznaczali

$$\tilde{A} = (a, \alpha_A, \beta_A).$$



Rys. 1.2: Trapezowa liczba rozmyta $\tilde{A} = (\underline{a}, \bar{a}, \alpha_A, \beta_A)$

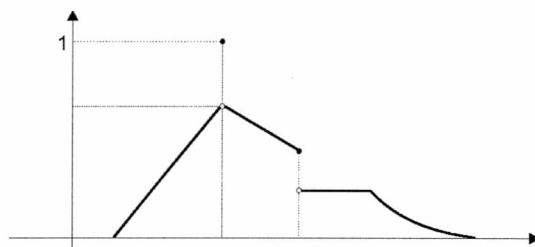
Liczby rozmyte typu $L-R$ posiadają następującą własność:

Własność 1.2. *Jeżeli $\tilde{A} = (\underline{a}, \bar{a}, \alpha_A, \beta_A)_{L-R}$ jest liczbą rozmytą typu $L-R$, to dla każdego $\lambda \in (0, 1]$:*

$$\tilde{A}^\lambda = [\underline{a}(\lambda), \bar{a}(\lambda)] = [\underline{a} - \alpha_A L^{-1}(\lambda), \bar{a} + \beta_A R^{-1}(\lambda)], \quad (1.3)$$

gdzie L^{-1} i R^{-1} są funkcjami odwrotnymi do L i R w tej części dziedziny, w której są dodatnie. Ponadto funkcja $\underline{a}(\lambda)$ jest ciągła i ściśle rosnąca dla $\lambda \in (0, 1]$, a funkcja $\bar{a}(\lambda)$ jest ciągła i ściśle malejąca dla $\lambda \in (0, 1]$.

Liczby przedziałowe oraz liczby rozmyte typu $L-R$ mają korzystne własności z obliczeniowego punktu widzenia. Mogą one być łatwo zakodowane w postaci kilku rzeczywistych parametrów. W kolejnym rozdziale pokażemy, że w pewnych przypadkach, podstawowe operacje na tych liczbach sprowadzają się do prostych operacji na liczbach rzeczywistych. W praktyce można rozpatrywać także liczby rozmyte, których funkcja przynależności jest bardziej złożona. Przykład takiej liczby jest przedstawiony na rysunku 1.3.



Rys. 1.3: Przykład liczby rozmytej z bardziej złożoną funkcją przynależności

Praktyczne zastosowanie liczb rozmytych z dowolnymi funkcjami przynależności prowadzi jednak do znacznych komplikacji obliczeniowych. W większości problemów badanych w pracy będą stosowane liczby przedziałowe oraz liczby rozmyte typu $L - R$.

1.3 Operacje na liczbach rozmytych

Operacje na liczbach rozmytych są określone za pomocą następującej zasady rozszerzania Zadeha:

Definicja 1.10. Niech $\varphi(\mathbf{x}) = \varphi(x_1, \dots, x_n)$ oznacza n -argumentową funkcję rzeczywistą, $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}$. Funkcja przynależności zbioru rozmytego $\tilde{A} = \varphi(\tilde{A}_1, \dots, \tilde{A}_n)$, gdzie $\tilde{A}_i \in FN(\mathbb{R})$, $i = 1, \dots, n$, jest określona następująco:

$$\mu_{\tilde{A}}(z) = \begin{cases} \sup_{x \in \mathbb{R}^n; \varphi(x)=z} \min\{\mu_{\tilde{A}_1}(x_1), \dots, \mu_{\tilde{A}_n}(x_n)\} & \text{jeżeli } \varphi^{-1}(z) \neq \emptyset, \\ 0 & \text{jeżeli } \varphi^{-1}(z) = \emptyset. \end{cases} \quad (1.4)$$

Korzystając z definicji 1.10 można określić funkcje przynależności liczb rozmytych, które są wynikiem rozszerzonych operacji dodawania, odejmowania, mnożenia i maksimum na liczbach rozmytych. Operacje te będziemy oznaczali odpowiednio: $\tilde{+}$, $\tilde{-}$, $\tilde{*}$, $\tilde{\max}$.

$$\mu_{\tilde{A} \tilde{+} \tilde{B}}(z) = \sup_{x_1, x_2: x_1 + x_2 = z} \min\{\mu_{\tilde{A}}(x_1), \mu_{\tilde{B}}(x_2)\}, \quad (1.5)$$

$$\mu_{\tilde{A} \tilde{-} \tilde{B}}(z) = \sup_{x_1, x_2: x_1 - x_2 = z} \min\{\mu_{\tilde{A}}(x_1), \mu_{\tilde{B}}(x_2)\}, \quad (1.6)$$

$$\mu_{\tilde{A} \tilde{*} \tilde{B}}(z) = \sup_{x_1, x_2: x_1 * x_2 = z} \min\{\mu_{\tilde{A}}(x_1), \mu_{\tilde{B}}(x_2)\}, \quad (1.7)$$

$$\mu_{\tilde{\max}\{\tilde{A} \tilde{B}\}}(z) = \sup_{x_1, x_2: \max\{x_1, x_2\} = z} \min\{\mu_{\tilde{A}}(x_1), \mu_{\tilde{B}}(x_2)\}. \quad (1.8)$$

Szczególnym przypadkiem mnożenia liczb rozmytych jest mnożenie liczby rozmytej przez liczbę rzeczywistą. W poprzednim punkcie stwierdziliśmy, że liczba rzeczywista $z \in \mathfrak{R}$ może być traktowana jako szczególny przypadek liczby rozmytej o funkcji przynależności:

$$\mu_z(x) = \begin{cases} 1 & \text{dla } x = z, \\ 0 & \text{dla } x \neq z. \end{cases} \quad (1.9)$$

Oznaczmy $z \tilde{*} \tilde{A} = z \tilde{A}$. Z zasady rozszerzania Zadeha otrzymujemy:

$$\mu_{z \tilde{A}}(y) = \sup_{x_2: z * x_2 = y} \min\{1, \mu_{\tilde{A}}(x_2)\} = \mu_{\tilde{A}}\left(\frac{y}{z}\right) \text{ dla } z \neq 0 \quad (1.10)$$

i $z \tilde{A} = 0$ dla $z = 0$.

W praktycznych zastosowaniach wyznaczenie funkcji przynależności (1.5) - (1.8) wprost z definicji może być trudne. Zachodzą następujące, ważne fakty:

Własność 1.3. Jeżeli $\tilde{A} = [\underline{a}, \bar{a}]$, $\tilde{B} = [\underline{b}, \bar{b}]$ są liczbami przedziałowymi, to:

$$\tilde{A} \tilde{+} \tilde{B} = [\underline{a} + \underline{b}, \bar{a} + \bar{b}], \quad (1.11)$$

$$\tilde{A} \tilde{-} \tilde{B} = [\underline{a} - \bar{b}, \bar{a} - \underline{b}], \quad (1.12)$$

$$\text{m}\ddot{\text{a}}\text{x}\{\tilde{A}, \tilde{B}\} = [\max\{\underline{a}, \underline{b}\}, \max\{\bar{a}, \bar{b}\}], \quad (1.13)$$

$$z \tilde{A} = [z \underline{a}, z \bar{a}], \quad z \geq 0. \quad (1.14)$$

Twierdzenie 1.1. Jeżeli funkcja $\varphi : \mathfrak{R} \times \mathfrak{R} \rightarrow \mathfrak{R}$ jest ciągła, to dla dowolnych liczb rozmytych $\tilde{A}, \tilde{B} \in FN(\mathfrak{R})$ i dla dowolnego $\lambda \in (0, 1]$ zachodzi:

$$(\varphi(\tilde{A}, \tilde{B}))^\lambda = \varphi(\tilde{A}^\lambda, \tilde{B}^\lambda)$$

gdzie $\varphi(\tilde{A}, \tilde{B})$, $\tilde{A}, \tilde{B} \in FN(\mathfrak{R})$, jest zbiorem rozmytym określonym za pomocą zasady rozszerzania Zadeha (patrz definicja 1.10).

Z twierdzenia 1.1 oraz ciągłości operacji dodawania, odejmowania, mnożenia i maksimum w zbiorze liczb rzeczywistych wynika następująca własność:

Własność 1.4. Jeżeli $\tilde{A}$ i $\tilde{B}$ są liczbami rozmytymi, to dla każdego $\lambda \in (0, 1]$ zachodzą następujące warunki:

$$(\tilde{A} \tilde{+} \tilde{B})^\lambda = \tilde{A}^\lambda \tilde{+} \tilde{B}^\lambda = [\underline{a}(\lambda) + \underline{b}(\lambda), \bar{a}(\lambda) + \bar{b}(\lambda)], \quad (1.15)$$

$$(\tilde{A} \tilde{-} \tilde{B})^\lambda = \tilde{A}^\lambda \tilde{-} \tilde{B}^\lambda = [\underline{a}(\lambda) - \bar{b}(\lambda), \bar{a}(\lambda) - \underline{b}(\lambda)], \quad (1.16)$$

$$(\text{m}\ddot{\text{a}}\text{x}\{\tilde{A}, \tilde{B}\})^\lambda = \text{m}\ddot{\text{a}}\text{x}\{\tilde{A}^\lambda, \tilde{B}^\lambda\} = [\max\{\underline{a}(\lambda), \underline{b}(\lambda)\}, \max\{\bar{a}(\lambda), \bar{b}(\lambda)\}], \quad (1.17)$$

$$(z \tilde{A})^\lambda = z \tilde{A}^\lambda = [z \underline{a}(\lambda), z \bar{a}(\lambda)], \quad z \geq 0. \quad (1.18)$$

Własność 1.4 ma kluczowe znaczenie w zastosowaniach. Wynika z niej, że podstawowe operacje mogą być łatwo wykonywane na λ -poziomach liczb rozmytych.

Operacje dodawania, odejmowania i mnożenia przez dodatnią liczbę rzeczywistą mogą być wykonywane szczególnie prosto w klasie liczb rozmytych typu $L - R$. Zachodzi następująca własność:

Własność 1.5.

$$(\underline{a}, \bar{a}, \alpha_A, \beta_A)_{L-R} \tilde{+} (\underline{b}, \bar{b}, \alpha_B, \beta_B)_{L-R} = (\underline{a} + \underline{b}, \bar{a} + \bar{b}, \alpha_A + \alpha_B, \beta_A + \beta_B)_{L-R}, \quad (1.19)$$

$$(\underline{a}, \bar{a}, \alpha_A, \beta_A)_{L-R} \tilde{-} (\underline{b}, \bar{b}, \alpha_B, \beta_B)_{R-L} = (\underline{a} - \underline{b}, \bar{a} - \bar{b}, \alpha_A + \beta_A, \beta_A + \alpha_B)_{L-R}, \quad (1.20)$$

$$z(\underline{a}, \bar{a}, \alpha_A, \beta_A)_{L-R} = (z\underline{a}, z\bar{a}, z\alpha_A, z\beta_A)_{L-R}, \quad z > 0. \quad (1.21)$$

Niestety dla rozszerzonej operacji maksimum nie można podać równie prostej formuły. Zauważmy, że dodawanie i odejmowanie liczb rozmytych typu $L - R$ może być łatwo wykonane tylko wtedy, gdy odpowiednie funkcje kształtu są takie same. Własność 1.5, w przypadku dodawania i odejmowania, nie zachodzi więc dla dowolnych dwóch liczb rozmytych typu $L - R$. Warto również zauważyć, że w zbiorze liczb rozmytych odejmowanie nie jest operacją odwrotną do dodawania (tzn. istnieją liczby rozmyte $\tilde{A}$, $\tilde{B}$, $\tilde{C}$ takie, że $\tilde{A} \tilde{+} \tilde{B} = \tilde{C}$ ale $\tilde{C} \tilde{-} \tilde{B} \neq \tilde{A}$). Jest to istotna różnica w porównaniu z analogicznymi operacjami określonymi w zbiorze liczb rzeczywistych.

1.4 Interpretacja liczby rozmytej

W praktycznych zastosowaniach liczb rozmytych możliwych jest kilka różnych sposobów ich interpretacji. W pracy zakładamy, że liczba rozmyta $\tilde{A}$ wyznacza rozkład możliwości pewnej zmiennej rzeczywistej V , której dokładna wartość nie jest znana. Zmienną V nazywamy w takim przypadku zmienną rozmytą (przez analogię do zmiennej losowej). Bardziej formalnie wyrażenie " V jest $\tilde{A}$ ", gdzie $\tilde{A}$ jest liczbą rozmytą, generuje rozkład możliwości zmiennej V w następujący sposób:

$$Poss(V = x) = \mu_{\tilde{A}}(x). \quad (1.22)$$

Wyrażenie (1.22) oznacza, że możliwość przyjęcia przez zmienną V wartości x jest określona przez wartość funkcji przynależności $\mu_{\tilde{A}}(x)$.

Interpretacja rozkładu możliwości zmiennej rozmytej przypomina interpretację rozkładu prawdopodobieństwa zmiennej losowej. Wartość funkcji przynależności $\mu_{\tilde{A}}(x)$ oznacza stopień pewności co do możliwości zajścia zdarzenia $V = x$, wyznaczony na podstawie informacji posiadanej o zmiennej V . Istnieje jednak istotna różnica między pojęciami rozkładu prawdopodobieństwa i rozkładu możliwości. Za pomocą rozkładu możliwości określamy potencjalne, często oceniane subiektywnie możliwości zajścia określonych zdarzeń, natomiast rozkład prawdopodobieństwa jest obiektywną (daną przez naturę) własnością modelowanej zmiennej. Rozkład możliwości nie musi więc pokrywać się z rozkładem prawdopodobieństwa. Nie musi też spełniać warunku niezbędnego dla rozkładu prawdopodobieństwa (tzn. jest dopuszczalne $\int_{-\infty}^{\infty} \mu_{\tilde{A}}(x) \neq 1$). Jedynym dającym się stwierdzić związkiem jest implikacja: zdarzenie w małym stopniu możliwe jest także w małym stopniu prawdopodobne. Duża (potencjalna) możliwość zajścia zdarzenia nie musi pociągać za sobą dużego prawdopodobieństwa realizacji tego zdarzenia. Jest to tak zwana zasada zgodności między rozkładami możliwości i prawdopodobieństwa (patrz [36]).

Mając rozkłady możliwości zmiennych rozmytych U i V można określić możliwość zajścia bardziej złożonych zdarzeń. Jeżeli $[a, b] \subseteq \mathbb{R}$, to:

$$Poss(V \in [a, b]) = \sup_{x \in [a, b]} \mu_{\tilde{A}}(x). \quad (1.23)$$

Jeżeli $\odot$ jest pewną operacją określoną w zbiorze liczb rzeczywistych to:

$$Poss(U \odot V = x) = \mu_{\tilde{A} \tilde{\odot} \tilde{B}}(x), \quad (1.24)$$

gdzie $\tilde{\odot}$ jest operacją na liczbach rozmytych określoną za pomocą zasady rozszerzania Zadeha. Funkcja przynależności liczby rozmytej $\tilde{A} \tilde{\odot} \tilde{B}$ wyznacza więc rozkład możliwości zmiennej rozmytej $U \odot V$.

W ogólnym przypadku można określać możliwość zajścia dowolnej relacji między zmiennymi rozmytymi. Szczególne znaczenie w pracy będzie miało określenie możliwości zajścia określonej relacji większości między U i V (na przykład możliwości zajścia zdarzenia $U \geq V$). W punkcie 1.5 podamy sposób określania możliwości zajścia tego typu zdarzeń korzystając z indeksów zaproponowanych przez Dubois i Prade.

W literaturze poświęconej zbiorom rozmytym można spotkać również pojęcie konieczności zajścia danego zdarzenia. Jeżeli A jest pewnym zdarzeniem a $\bar{A}$ jest zdarzeniem przeciwnym do A to:

$$Ness(A) = 1 - Poss(\bar{A}). \quad (1.25)$$

Można pokazać, że jeżeli $[a, b] \subseteq \mathfrak{R}$, to:

$$\text{Ness}(V \in [a, b]) = 1 - \inf\{\lambda \mid \tilde{A}^\lambda \subseteq [a, b]\}, \quad [a, b] \in I(\mathbb{R}), \quad (1.26)$$

gdzie $I(\mathbb{R})$ jest zbiorem wszystkich domkniętych przedziałów. Wyrażenie (1.26) określa konieczność przyjęcia przez zmienną V wartości z zadanego przedziału $[a, b]$.

Z praktycznego punktu widzenia interesujące jest zagadnienie wyznaczenia rozkładu prawdopodobieństwa zmiennej rozmytej V na podstawie wyrażenia " V jest $\tilde{A}$ " (tzn. przekształcenia zmiennej rozmytej w zmienną losową). Oczywiście nie może być tu mowy o jednoznacznej i jednocześnie w pełni uzasadnionej procedurze wyznaczania takiego rozkładu. Przedstawimy teraz jedno z podejść do tego problemu przedstawione w pracy [5].

Definicja 1.11. Niech S będzie zmienną losową o rozkładzie jednostajnym na odcinku $[0, 1]$ a T zmienną losową o rozkładzie jednostajnym na odcinku $(0, 1]$. Zmienną losową $Z_{\tilde{A}}$, związaną z liczbą rozmytą $\tilde{A}$ nazywamy zmienną losową określoną następująco:

$$Z_{\tilde{A}} = (1 - S)\bar{a}(T) + S\underline{a}(T). \quad (1.27)$$

W [5] zmienna $Z_{\tilde{A}}$ nazywana jest generatorem pojedynczych wartości zmiennej rozmytej V . Generator ten jest równoznaczny z procedurą losowego wyboru liczby z rozmytego przedziału $\tilde{A}$. Warto zauważyć, że w przypadku gdy $\tilde{A}$ jest liczbą przedziałową, $\tilde{A} = [\underline{a}, \bar{a}]$ (patrz definicja 1.8), to $Z_{\tilde{A}}$ jest zmienną losową o rozkładzie jednostajnym na $[\underline{a}, \bar{a}]$. Także w innych, szczególnych przypadkach liczb rozmytych rozkład zmiennej losowej $Z_{\tilde{A}}$ może zostać bezpośrednio wyznaczony. Na przykład jeżeli $\tilde{A} = (a, \alpha_A, \beta_A)$ jest liczbą trójkątną, to dystrybuenta $F(x)$ rozkładu prawdopodobieństwa zmiennej losowej $Z_{\tilde{A}}$ ma następującą postać:

$$F(x) = \begin{cases} 0 & dla \quad x \leq a - \alpha_A, \\ \frac{1}{\alpha_A + \beta_A} \left[(a - x) \ln \left(\frac{a - x}{e\alpha_A} \right) \right] + \alpha_A & dla \quad a - \alpha_A < x < a, \\ \frac{\alpha_A}{\alpha_A + \beta_A} & dla \quad x = a, \\ \frac{1}{\alpha_A + \beta_A} \left[(x - a) \ln \left(\frac{e\beta_A}{x - a} \right) \right] + \alpha_A & dla \quad a < x < a + \beta_A, \\ 1 & dla \quad x \geq a + \beta_A. \end{cases}$$

Szczególne znaczenie w niniejszej pracy będzie miała wartość oczekiwana zmiennej losowej $Z_{\tilde{A}}$. Własności tego parametru zostaną szczegółowo przedstawione w punkcie 1.6.

Na zakończenie tego rozdziału podamy kilka uwag dotyczących interpretacji pojęć stosowanych w dalszej części pracy. Mówiąc o liczbie rozmytej $\tilde{A}$ będziemy mieli na myśli zmienną rozmytą, której rozkład możliwości wyznaczony jest przez funkcję przynależności liczby $\tilde{A}$. Operacje na liczbach rozmytych w rzeczywistości oznaczać będą operacje na zmiennych rozmytych. Możliwość zajścia określonej relacji $R(U, V)$ między zmiennymi rozmytymi U i V będziemy utożsamiać z możliwością zajścia tej relacji między liczbami rozmytymi $\tilde{A}$ i $\tilde{B}$ tzn:

$$Poss(R(U, V)) = Poss(R(\tilde{A}, \tilde{B})).$$

W dalszej części nie będziemy już posługiwać się terminem zmiennej rozmytej, zastępując go na mocy powyższych uwag pojęciem liczby rozmytej.

1.5 Indeksy Dubois i Prade

W punkcie tym określimy możliwość oraz konieczność zajścia zadanych relacji większości między dwiema liczbami rozmytymi¹. Do tego celu wykorzystamy indeksy zaproponowane przez Dubois i Prade [12], które są określone w sposób następujący:

Definicja 1.12. *Jeżeli $\tilde{A}$ i $\tilde{B}$ są liczbami rozmytymi, to:*

$$Poss(\tilde{A} \geq \tilde{B}) = \sup_{x \geq y} \min\{\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(y)\}, \quad (1.28)$$

$$Poss(\tilde{A} > \tilde{B}) = \sup_x \inf_{y \geq x} \min\{\mu_{\tilde{A}}(x), 1 - \mu_{\tilde{B}}(y)\}. \quad (1.29)$$

Indeks $Poss(\tilde{A} \geq \tilde{B})$ określa możliwość, że liczba rozmyta $\tilde{A}$ jest nie mniejsza niż liczba rozmyta $\tilde{B}$. Analogicznie indeks $Poss(\tilde{A} > \tilde{B})$ określa możliwość, że liczba rozmyta $\tilde{A}$ jest większa od liczby rozmytej $\tilde{B}$. Z definicji 1.12 oraz z określenia funkcji przynależności liczby rozmytej wynika, że indeksy Dubois i Prade przyjmują wartości z przedziału $[0, 1]$.

Korzystając z określenia indeksu $Poss(\tilde{A} \geq \tilde{B})$, zdefiniujemy indeks określający możliwość zajścia relacji równości między liczbami rozmytymi $\tilde{A}$ i $\tilde{B}$ [31].

Definicja 1.13. *Jeżeli $\tilde{A}$ i $\tilde{B}$ są liczbami rozmytymi, to:*

$$Poss(\tilde{A} = \tilde{B}) = \min\{Poss(\tilde{A} \geq \tilde{B}), Poss(\tilde{B} \geq \tilde{A})\}. \quad (1.30)$$

¹Analogicznym pojęciem w teorii prawdopodobieństwa jest prawdopodobieństwo zdarzenia, że wartość zmiennej losowej X jest większa od wartości zmiennej losowej Y .

Z określenia konieczności zajścia zdarzenia (patrz formuła (1.25)) możemy określić konieczność, że liczba $\tilde{A}$ jest większa od $\tilde{B}$ oraz konieczność, że liczba $\tilde{A}$ jest różna od $\tilde{B}$:

$$Ness(\tilde{A} > \tilde{B}) = 1 - Poss(\tilde{B} \geq \tilde{A}), \quad (1.31)$$

$$Ness(\tilde{A} \neq \tilde{B}) = 1 - Poss(\tilde{A} = \tilde{B}). \quad (1.32)$$

Prawdziwa jest następująca zależność między zdefiniowanymi indeksami:

Własność 1.6. *Dla każdych dwóch liczb rozmytych $\tilde{A}$, $\tilde{B}$:*

$$Poss(\tilde{A} \geq \tilde{B}) \geq Poss(\tilde{A} > \tilde{B}) \geq Ness(\tilde{A} > \tilde{B}). \quad (1.33)$$

Własność 1.6 jest naturalna. Możliwość zajścia nieostrej relacji większości między dwiema liczbami rozmytymi jest niemniejsza od możliwości zajścia ostrej relacji większości między tymi liczbami. Z kolei konieczność zajścia ostrej relacji większości między dwiema liczbami jest niewiększa od możliwości zajścia tej relacji.

Podamy teraz kilka lematów za pomocą których można obliczyć wartości określonych indeksów dla dwóch zadanych liczb rozmytych. Zdefiniujemy:

$$\underline{a}(0) = \inf\{x : \mu_{\tilde{A}}(x) > 0\}, \quad (1.34)$$

$$\overline{a}(0) = \sup\{x : \mu_{\tilde{A}}(x) > 0\}. \quad (1.35)$$

Przedział $[\underline{a}(0), \overline{a}(0)]$ określany jest w literaturze terminem podstawa (support) liczby rozmytej $\tilde{A}$. Zauważmy, że możliwy jest przypadek, w którym $\underline{a}(0) = -\infty$ lub $\overline{a}(0) = \infty$. Łatwo pokazać, że jeżeli $\underline{a}(0)$ (odpowiednio $\overline{a}(0)$) jest skończone, to $\mu_{\tilde{A}}(\underline{a}(0)) = 0$ (odpowiednio $\mu_{\tilde{A}}(\overline{a}(0)) = 0$). Ponadto dla każdego $x \in (\underline{a}(0), \overline{a}(0))$ zachodzi $\mu_{\tilde{A}}(x) > 0$.

Lemat 1.1. *Dla dowolnych liczb rozmytych $\tilde{A}, \tilde{B} \in FN(\mathfrak{R})$ zachodzi następujący warunek:*

$$Poss(\tilde{A} > \tilde{B}) = \sup_{x \in [\overline{b}(1), \infty)} \min\{\mu_{\tilde{A}}(x), 1 - \mu_{\tilde{B}}(x)\}. \quad (1.36)$$

Dowód Zdefiniujmy funkcję $\varphi : \mathfrak{R} \rightarrow [0, 1]$ w następujący sposób:

$$\varphi(x) = \inf_{y \geq x} \min\{\mu_{\tilde{A}}(x), 1 - \mu_{\tilde{B}}(y)\}. \quad (1.37)$$

Funkcja $\varphi(x)$ przyjmuje wartości równe 0 dla $x \leq \bar{b}(1)$, czyli jest spełniony warunek:

$$\forall_{x < \bar{b}(1)} \varphi(x) = \varphi(\bar{b}(1)) = 0. \quad (1.38)$$

Wynika on bezpośrednio z definicji funkcji $\varphi(x)$ (patrz (1.37)) oraz z faktu, że $1 - \mu_{\bar{B}}(\bar{b}(1)) = 0$. Z definicji indeksu $Poss(\tilde{A} > \tilde{B})$ (patrz definicja 1.12) i z warunku (1.38) wynika następujący warunek:

$$Poss(\tilde{A} > \tilde{B}) = \sup_x \varphi(x) = \sup_{x \in [\bar{b}(1), \infty)} \varphi(x). \quad (1.39)$$

Weźmy dowolne $x \geq \bar{b}(1)$. Zdefiniujmy następujące zbiory:

$$S^-(x) = \{y : y \geq x \text{ i } \mu_{\tilde{A}}(x) < 1 - \mu_{\tilde{B}}(y)\},$$

$$S^+(x) = \{y : y \geq x \text{ i } \mu_{\tilde{A}}(x) \geq 1 - \mu_{\tilde{B}}(y)\}.$$

Z określenia zbiorów $S^-(x)$ i $S^+(x)$ wynika, że:

$$S^-(x) \cap S^+(x) = \emptyset,$$

$$S^-(x) \cup S^+(x) = \{y | y \geq x\}.$$

Założmy najpierw, że $S^+(x) \neq \emptyset$. Z definicji zbioru $S^+(x)$ otrzymujemy, że

$$\inf_{y \in S^+(x)} \min\{\mu_{\tilde{A}}(x), 1 - \mu_{\tilde{B}}(y)\} = \inf_{y \in S^+(x)} (1 - \mu_{\tilde{B}}(y)).$$

Zauważmy, że funkcja $\mu_{\tilde{B}}(y)$ jest nierosnąca dla $y \geq \bar{b}(1)$. Stąd oraz z tego, że zbiór $S^+(x)$ jest niepusty wynika, że $y = x \in S^+(x)$. Zatem funkcja $\mu_{\tilde{B}}(y)$ osiąga maksimum w zbiorze $S^+(x)$ dla $y = x$. Stąd:

$$\varphi^+(x) = \inf_{y \in S^+(x)} (1 - \mu_{\tilde{B}}(y)) = 1 - \mu_{\tilde{B}}(x). \quad (1.40)$$

Założmy teraz, że $S^-(x) \neq \emptyset$. Z określenia zbioru $S^-(x)$ wynika, że:

$$\varphi^-(x) = \inf_{y \in S^-(x)} \min\{\mu_{\tilde{A}}(x), 1 - \mu_{\tilde{B}}(y)\} = \mu_{\tilde{A}}(x). \quad (1.41)$$

Rozpatrzmy następujące przypadki:

1. $S^+(x) \neq \emptyset$ i $S^-(x) \neq \emptyset$. Wówczas z warunków (1.40) i (1.41) wynika, że:

$$\varphi(x) = \min\{\varphi^+(x), \varphi^-(x)\} = \min\{\mu_{\tilde{A}}(x), 1 - \mu_{\tilde{B}}(x)\}.$$

2. $S^+(x) = \emptyset$ i $S^-(x) = \{y|y \geq x\}$. Wówczas z faktu, że $x \in S^-(x)$ i definicji zbioru $S^-(x)$ otrzymujemy, że $\mu_{\tilde{A}}(x) < 1 - \mu_{\tilde{A}}(x)$. Stąd oraz z warunku (1.41) wynika, że:

$$\varphi(x) = \varphi^-(x) = \mu_{\tilde{A}}(x) = \min\{\mu_{\tilde{A}}(x), 1 - \mu_{\tilde{B}}(x)\}.$$

3. $S^-(x) = \emptyset$ i $S^+(x) = \{y|y \geq x\}$. Wówczas z faktu, że $x \in S^+(x)$ i definicji zbioru $S^+(x)$ otrzymujemy, że $\mu_{\tilde{A}}(x) \geq 1 - \mu_{\tilde{A}}(x)$. Stąd oraz z warunku (1.40) wynika, że:

$$\varphi(x) = \varphi^+(x) = 1 - \mu_{\tilde{B}}(x) = \min\{\mu_{\tilde{A}}(x), 1 - \mu_{\tilde{B}}(x)\}.$$

Z analizy przypadków 1-3 wynika, że:

$$\varphi(x) = \min\{\mu_{\tilde{A}}(x), 1 - \mu_{\tilde{B}}(x)\}, \quad x \geq \bar{b}(1). \quad (1.42)$$

Z warunków (1.39) i (1.42) wynika teza lematu. $\square$

Lemat 1.2. *Dla dowolnych liczb rozmytych $\tilde{A}, \tilde{B} \in FN(\mathfrak{R})$ zachodzi następujący warunek:*

$$Poss(\tilde{A} \geq \tilde{B}) = 0 \Leftrightarrow \bar{a}(0) \leq \underline{b}(0) \quad (1.43)$$

Dowód Z definicji indeksu $Poss(\tilde{A} \geq \tilde{B})$ (definicja 1.12) oraz z nieujemności funkcji przynależności liczby rozmytej wynika, że warunek $Poss(\tilde{A} \geq \tilde{B}) = 0$ jest spełniony wtedy i tylko wtedy gdy:

$$\forall_{x,y;x \geq y} \min\{\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(y)\} = 0. \quad (1.44)$$

Z kolei z równości (1.34) i (1.35) wynika, że warunek (1.44) jest równoważny następującemu warunkowi:

$$\forall_{x,y;x \geq y} (x \notin (\underline{a}(0), \bar{a}(0)) \text{ lub } y \notin (\underline{b}(0), \bar{b}(0))). \quad (1.45)$$

Warunek (1.45) zachodzi wtedy i tylko wtedy, gdy $\bar{a}(0) \leq \underline{b}(0)$, co kończy dowód. $\square$

Lemat 1.3. *Dla dowolnych liczb rozmytych $\tilde{A}, \tilde{B} \in FN(\mathfrak{R})$ i dowolnego $r \in (0, 1]$ jest spełniony następujący warunek:*

$$Poss(\tilde{A} \geq \tilde{B}) \geq r \Leftrightarrow \bar{a}(r) \geq \underline{b}(r). \quad (1.46)$$

Dowód

$\Leftarrow$ Niech r będzie dowolną liczbą z przedziału $(0, 1]$. Załóżmy, że $\bar{a}(r) \geq \underline{b}(r)$. Stąd oraz z definicji i własności λ -poziomu liczby rozmytej wynika, że istnieją $x, y \in \mathfrak{R}$ takie, że $x \geq y$ i $\mu_{\tilde{A}}(x) \geq r$ i $\mu_{\tilde{B}}(y) \geq r$ (warunki te są spełnione choćby przez $x = \bar{a}(r)$ i $y = \underline{b}(r)$). Z nierówności $\mu_{\tilde{A}}(x) \geq r$ i $\mu_{\tilde{B}}(y) \geq r$ wynika, że $\min\{\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(y)\} \geq r$. Stąd i z definicji supremum otrzymujemy

$$Poss(\tilde{A} \geq \tilde{B}) = \sup_{x \geq y} \min\{\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(y)\} \geq r.$$

$\Rightarrow$ Niech r będzie dowolną liczbą z przedziału $(0, 1]$. Załóżmy, że:

$$Poss(\tilde{A} \geq \tilde{B}) = r_0 \geq r.$$

Stąd oraz z definicji supremum wynika, że:

$$\forall \varepsilon > 0 \exists x \geq y \min\{\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(y)\} \geq r_0 - \varepsilon. \quad (1.47)$$

Z warunku (1.47), korzystając z własności operacji minimum ($\min\{a, b\} \geq p \Leftrightarrow a \geq p \text{ i } b \geq p$) oraz z definicji i własności λ -poziomu liczby rozmytej, otrzymujemy następujący warunek:

$$\forall \varepsilon > 0 \exists x \geq y (x \in [\underline{a}(r_0 - \varepsilon), \bar{a}(r_0 - \varepsilon)] \text{ i } y \in [\underline{b}(r_0 - \varepsilon), \bar{b}(r_0 - \varepsilon)]). \quad (1.48)$$

Z warunku (1.48) wynika następujący warunek:

$$\forall \varepsilon > 0 \bar{a}(r_0 - \varepsilon) \geq \underline{b}(r_0 - \varepsilon). \quad (1.49)$$

Z warunku (1.49) oraz lewostronnej ciągłości funkcji $\bar{a}(t)$ i $\underline{b}(t)$ dla $t \in (0, 1]$, otrzymujemy:

$$\bar{a}(r_0) \geq \underline{b}(r_0). \quad (1.50)$$

Z warunku (1.50), nierówności $r_0 \geq r$ oraz z faktu, że $\bar{a}(t)$, $t \in (0, 1]$ jest funkcją nierosnącą a $\underline{b}(t)$, $t \in (0, 1]$ jest funkcją niemalejącą otrzymujemy ostatecznie

$$\bar{a}(r) \geq \underline{b}(r),$$

co kończy dowód. □

Lemat 1.4. Dla każdych liczb rozmytych $\tilde{A}$, $\tilde{B}$ zachodzi następujący warunek:

$$Poss(\tilde{A} = \tilde{B}) = 1 \Leftrightarrow [\underline{a}(1), \bar{a}(1)] \cap [\underline{b}(1), \bar{b}(1)] \neq \emptyset$$

Dowód Z określenia indeksu $Poss(\tilde{A} = \tilde{B})$ (patrz definicja 1.13) wynika, że $Poss(\tilde{A} = \tilde{B}) = 1$ wtedy i tylko wtedy gdy:

$$Poss(\tilde{A} \geq \tilde{B}) = 1 \text{ i } Poss(\tilde{B} \geq \tilde{A}) = 1. \quad (1.51)$$

Z lematu 1.3 otrzymujemy, że warunek (1.51) zachodzi wtedy i tylko wtedy gdy:

$$\bar{a}(1) \geq \underline{b}(1) \text{ i } \bar{b}(1) \geq \underline{a}(1),$$

który zachodzi wtedy i tylko wtedy gdy $[\underline{a}(1), \bar{a}(1)] \cap [\underline{b}(1), \bar{b}(1)] \neq \emptyset$. $\square$

Lemat 1.5. Dla dowolnych liczb rozmytych $\tilde{A}, \tilde{B} \in FN(\mathfrak{R})$ zachodzi następujący warunek:

$$Poss(\tilde{A} > \tilde{B}) = 0 \Leftrightarrow \bar{a}(0) \leq \bar{b}(1) \quad (1.52)$$

Dowód Z lematu 1.1 oraz z faktu, że funkcja przynależności liczby rozmytej przyjmuje wartości z przedziału $[0, 1]$ wynika, że warunek $Poss(\tilde{A} > \tilde{B}) = 0$ zachodzi wtedy i tylko wtedy gdy:

$$\forall_{x \in [\bar{b}(1), \infty]} \min\{\mu_{\tilde{A}}(x), 1 - \mu_{\tilde{B}}(x)\} = 0. \quad (1.53)$$

Z kolei z równości (1.34) i (1.35) wynika, że warunek (1.53) jest równoważny warunkowi:

$$\forall_{x \in [\bar{b}(1), \infty]} (x \notin (\underline{a}(0), \bar{a}(0)) \text{ lub } x \in [\underline{b}(1), \bar{b}(1)]). \quad (1.54)$$

Warunek (1.54) zachodzi wtedy i tylko wtedy, gdy $\bar{a}(0) \leq \underline{b}(1)$, co kończy dowód. $\square$

Lemat 1.6. Dla dowolnych liczb rozmytych $\tilde{A}, \tilde{B} \in FN(\mathfrak{R})$ zachodzi następujący warunek:

$$Poss(\tilde{A} > \tilde{B}) = 1 \Leftrightarrow \bar{a}(1) \geq \bar{b}(0) \quad (1.55)$$

Dowód Z lematu 1.1 oraz z definicji supremum wynika, że warunek $Poss(\tilde{A} > \tilde{B}) = 1$ jest równoważny warunkowi:

$$\forall_{\varepsilon > 0} \exists_{x \in [\bar{b}(1), \infty]} \min\{\mu_{\tilde{A}}(x), 1 - \mu_{\tilde{B}}(x)\} \geq 1 - \varepsilon. \quad (1.56)$$

Z własności operacji minimum wynika, że warunek (1.56) jest spełniony wtedy i tylko wtedy gdy jest spełniony warunek:

$$\forall_{\varepsilon>0} \exists_{x \in [\bar{b}(1), \infty]} (\mu_{\tilde{A}}(x) \geq 1 - \varepsilon \text{ i } 1 - \mu_{\tilde{B}}(x) \geq 1 - \varepsilon). \quad (1.57)$$

Z definicji i własności λ -poziomu liczby rozmytej otrzymujemy, że warunek (1.57) jest równoważny warunkowi:

$$\forall_{\varepsilon>0} \exists_{x \in [\bar{b}(1), \infty]} (x \in [\underline{a}(1 - \varepsilon), \bar{a}(1 - \varepsilon)] \text{ i } x \in [\bar{b}(\varepsilon), \infty)), \quad (1.58)$$

który z kolei zachodzi wtedy i tylko wtedy gdy:

$$\forall_{\varepsilon>0} (\bar{a}(1 - \varepsilon) \geq \bar{b}(\varepsilon)). \quad (1.59)$$

Z lewostronnej ciągłości funkcji $\bar{a}(t)$ oraz z określenia $\bar{b}(0)$ wynika, że warunek (1.59) jest równoważny następującemu warunkowi:

$$\bar{a}(1) \geq \bar{b}(0),$$

co kończy dowód. □

Lemat 1.7. *Dla każdych liczb rozmytych $\tilde{A}, \tilde{B} \in FN(\mathfrak{R})$ i dowolnego $r \in (0, 1)$ zachodzi następujący warunek:*

$$Poss(\tilde{A} > \tilde{B}) \geq r \Leftrightarrow \bar{a}(r) \geq \bar{b}(1 - r). \quad (1.60)$$

Dowód

$\Leftarrow$ Niech r będzie dowolną liczbą z przedziału $(0, 1)$. Załóżmy, że jest spełniony warunek $\bar{a}(r) \geq \bar{b}(1 - r)$. Stąd oraz z nierówności $\underline{a}(r) \leq \bar{a}(r)$ otrzymujemy warunek:

$$[\underline{a}(r), \bar{a}(r)] \cap [\bar{b}(1 - r), \infty) \neq \emptyset, \quad (1.61)$$

który jest równoważny warunkowi:

$$\exists_x \underline{a}(r) \leq x \leq \bar{a}(r) \text{ i } x \geq \bar{b}(1 - r). \quad (1.62)$$

Korzystając z definicji λ -poziomu liczby rozmytej oraz implikacji $x \geq \bar{b}(1 - r) \Rightarrow x \geq \bar{b}(1)$ ($\bar{b}(\lambda)$ jest funkcją nierosnącą) z warunku (1.62) otrzymujemy warunek:

$$\exists_{x \geq \bar{b}(1)} \mu_{\tilde{A}}(x) \geq r \text{ i } 1 - \mu_{\tilde{B}}(x) \geq r. \quad (1.63)$$

Z lematu 1.1, warunku (1.63) oraz z definicji supremum otrzymujemy warunek:

$$Poss(\tilde{A} > \tilde{B}) \geq r.$$

$\Rightarrow$ Niech r będzie dowolną liczbą z przedziału $(0, 1)$. Załóżmy, że:

$$Poss(\tilde{A} > \tilde{B}) = r_0 \geq r.$$

Z lematu 1.1 oraz z definicji supremum wynika, że:

$$\forall \varepsilon > 0 \exists x \in [\bar{b}(1), \infty) \min\{\mu_{\tilde{A}}(x), 1 - \mu_{\tilde{B}}(x)\} \geq r_0 - \varepsilon. \quad (1.64)$$

Z warunku (1.64), własności operacji minimum oraz z definicji λ -poziomu liczby rozmytej wynika następujący warunek:

$$\forall \varepsilon > 0 \exists x \in [\bar{b}(1), \infty) (x \leq \bar{a}(r_0 - \varepsilon) \text{ i } x \geq \bar{b}(1 - r_0 + \varepsilon)). \quad (1.65)$$

Z warunku (1.65) wynika następujący warunek:

$$\forall \varepsilon > 0 \bar{a}(r_0 - \varepsilon) \geq \bar{b}(1 - r_0 + \varepsilon). \quad (1.66)$$

Z warunku (1.66) oraz z lewostronnej ciągłości funkcji $\bar{a}(t)$ dla $t \in (0, 1]$ i prawostronnej ciągłości funkcji $\bar{b}(1 - t)$ dla $t \in [0, 1)$, otrzymujemy:

$$\bar{a}(r_0) \geq \bar{b}(1 - r_0). \quad (1.67)$$

Ponieważ $r_0 \geq r$, $\bar{a}(t)$ jest funkcją nierosnącą dla $t \in (0, 1]$ i $\bar{b}(1 - t)$ jest funkcją niemalejącą dla $t \in [0, 1)$ więc $\bar{a}(r) \geq \bar{b}(1 - r)$, co kończy dowód. $\square$

Lemat 1.8. *Jeżeli funkcja przynależności liczby $\tilde{A}$ jest ciągła i ściśle malejąca w przedziale $[\bar{a}(1), \bar{a}(0)]$ a funkcja przynależności liczby $\tilde{B}$ jest ciągła i ściśle rosnąca w przedziale $[\underline{b}(0), \underline{b}(1)]$ to:*

$$Poss(\tilde{A} \geq \tilde{B}) > \lambda \Leftrightarrow \bar{a}(\lambda) > \underline{b}(\lambda), \lambda \in [0, 1), \quad (1.68)$$

$$Poss(\tilde{A} \geq \tilde{B}) = \lambda \Leftrightarrow \bar{a}(\lambda) = \underline{b}(\lambda), \lambda \in (0, 1). \quad (1.69)$$

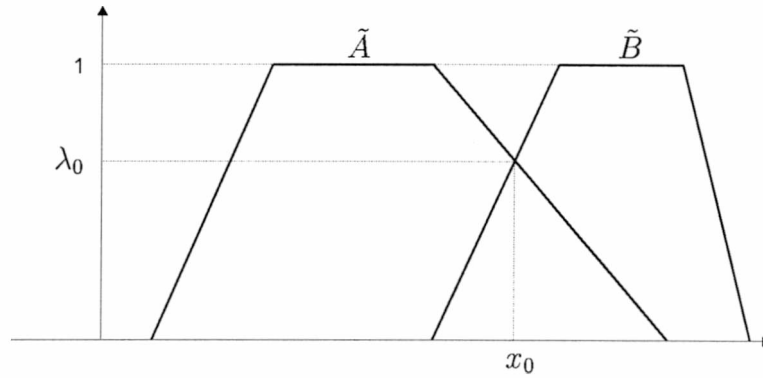
Dowód Oba warunki są konsekwencją lematu 1.3 oraz faktu, że funkcja $\bar{a}(t)$ jest ciągła i ściśle malejąca dla $t \in (0, 1)$ a funkcja $\underline{b}(t)$ jest ciągła i ściśle rosnąca dla $t \in (0, 1)$, przy przyjętych założeniach o funkcjach przynależności liczb rozmytych $\tilde{A}$ i $\tilde{B}$ (patrz rysunek 1.4). $\square$

Lemat 1.9. *Jeżeli funkcja przynależności liczby $\tilde{A}$ jest ciągła i ściśle malejąca w przedziale $[\bar{a}(1), \bar{a}(0)]$ a funkcja przynależności liczby $\tilde{B}$ jest ciągła i ściśle malejąca w przedziale $[\bar{b}(1), \bar{b}(0)]$ to*

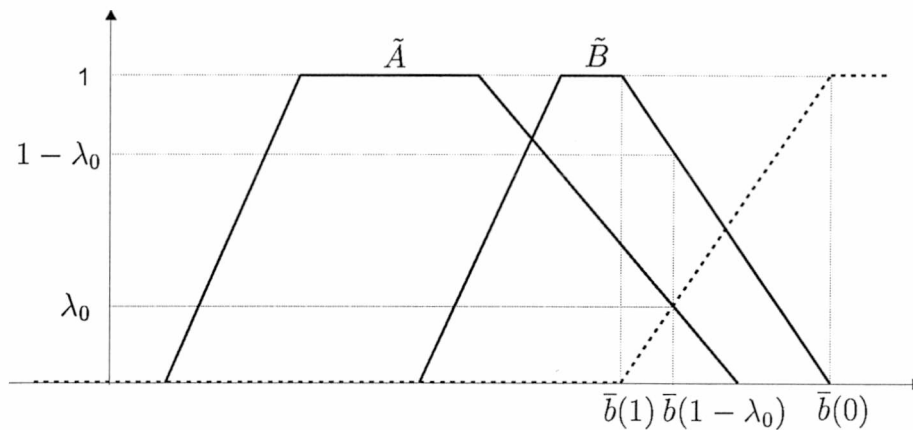
$$Poss(\tilde{A} > \tilde{B}) > \lambda \Leftrightarrow \bar{a}(\lambda) > \bar{b}(1 - \lambda), \lambda \in [0, 1), \quad (1.70)$$

$$Poss(\tilde{A} > \tilde{B}) = \lambda \Leftrightarrow \bar{a}(\lambda) = \bar{b}(1 - \lambda), \lambda \in (0, 1). \quad (1.71)$$

Dowód Oba warunki są konsekwencją lematu 1.7 oraz faktu, że funkcja $\bar{a}(t)$ jest ciągła i ściśle malejąca dla $t \in (0, 1)$ a funkcja $\bar{b}(1 - t)$ jest ciągła i ściśle rosnąca dla $t \in (0, 1)$, przy przyjętych założeniach o funkcjach przynależności liczb rozmytych $\tilde{A}$ i $\tilde{B}$ (patrz rysunek 1.5) $\square$



Rys. 1.4: Indeks $Poss(\tilde{A} \geq \tilde{B}) = \lambda_0 \in (0, 1)$



Rys. 1.5: Indeks $Poss(\tilde{A} > \tilde{B}) = \lambda_0 \in (0, 1)$

Podamy teraz metody łatwego obliczenia rozpatrywanych w tym punkcie indeksów dla szczególnych przypadków liczb rozmytych. Udowodnimy następujący lemat:

Lemat 1.10. *Jeżeli $\tilde{A} = (\underline{a}, \bar{a}, \alpha_A, \beta_A)_{L-R}$ i $\tilde{B} = (\underline{b}, \bar{b}, \alpha_B, \beta_B)_{R-S}$ są liczbami rozmytymi typu odpowiednio $L-R$ i $R-S$, to:*

$$Poss(\tilde{A} \geq \tilde{B}) = \begin{cases} 1 & \text{dla } \bar{a} \geq \underline{b}, \\ R(\frac{\underline{b} - \bar{a}}{\alpha_B + \beta_A}) & \text{dla } \bar{a} < \underline{b}. \end{cases} \quad (1.72)$$

Dowód Z lematu 1.3 wynika, że jeżeli $\bar{a} \geq \underline{b}$, to $Poss(\tilde{A} \geq \tilde{B}) = 1$.

Założmy teraz, że $\bar{a} < \underline{b}$. Z określenia funkcji kształtu $R(y)$ wynika, że wówczas $R(\frac{\underline{b} - \bar{a}}{\alpha_B + \beta_A}) \in [0, 1)$. Założmy, że $R(\frac{\underline{b} - \bar{a}}{\alpha_B + \beta_A}) = \lambda \in (0, 1)$. Ponieważ funkcja $R(y)$ jest ściśle monotoniczna w tej części dziedziny, w której jest dodatnia, więc istnieje funkcja odwrotna do R w tej części dziedziny i zachodzi następujący warunek:

$$R^{-1}(\lambda) = \frac{\underline{b} - \bar{a}}{\alpha_B + \beta_A},$$

czyli:

$$\bar{a} + \beta_A R^{-1}(\lambda) = \underline{b} - \alpha_B R^{-1}(\lambda). \quad (1.73)$$

Korzystając z określenia liczb rozmytych $\tilde{A}, \tilde{B}$ oraz z własności 1.2 otrzymujemy, że warunek (1.73) zachodzi wtedy i tylko wtedy, gdy $\bar{a}(\lambda) = \underline{b}(\lambda)$. Stąd oraz z lematu 1.8 otrzymujemy, że:

$$Poss(\tilde{A} \geq \tilde{B}) = \lambda = R(\frac{\underline{b} - \bar{a}}{\alpha_B + \beta_A}).$$

Założmy teraz, że $R(\frac{\underline{b} - \bar{a}}{\alpha_B + \beta_A}) = 0$. Z określenia liczb rozmytych $\tilde{A}, \tilde{B}$ otrzymujemy, że:

$$\bar{a}(0) \leq \frac{\underline{b} - \bar{a}}{\alpha_B + \beta_A} \leq \underline{b}(0).$$

Stąd oraz z lematu 1.2 otrzymujemy, że:

$$Poss(\tilde{A} \geq \tilde{B}) = 0 = R(\frac{\underline{b} - \bar{a}}{\alpha_B + \beta_A}),$$

co kończy dowód. □

Wszystkie indeksy mogą być obliczone za pomocą prostych formuł dla liczb trapezowych. Zachodzi następujący lemat:

Lemat 1.11. *Jeżeli $\tilde{A} = (\underline{a}, \bar{a}, \alpha_A, \beta_A)$ i $\tilde{B} = (\underline{b}, \bar{b}, \alpha_B, \beta_B)$ są liczbami trapezowymi, to prawdziwe są następujące wzory:*

$$Poss(\tilde{A} \geq \tilde{B}) = \begin{cases} 1 & \text{dla } \underline{b} - \bar{a} \leq 0, \\ 1 - \frac{\underline{b} - \bar{a}}{\alpha_B + \beta_A} & \text{dla } \underline{b} - \bar{a} \in (0, \alpha_B + \beta_A), \\ 0 & \text{dla } \underline{b} - \bar{a} \geq \alpha_B + \beta_A. \end{cases} \quad (1.74)$$

$$Poss(\tilde{A} > \tilde{B}) = \begin{cases} 1 & \text{dla } \bar{b} - \bar{a} \leq -\beta_B, \\ \frac{\beta_A + \bar{a} - \bar{b}}{\beta_B + \beta_A} & \text{dla } \bar{b} - \bar{a} \in (-\beta_B, \beta_A), \\ 0 & \text{dla } \bar{b} - \bar{a} \geq \beta_A. \end{cases} \quad (1.75)$$

$$Ness(\tilde{A} > \tilde{B}) = \begin{cases} 1 & \text{dla } \bar{b} - \underline{a} \leq -(\alpha_A + \beta_B), \\ \frac{\underline{a} - \bar{b}}{\alpha_A + \beta_B} & \text{dla } \bar{b} - \underline{a} \in (-(\alpha_A + \beta_B), 0), \\ 0 & \text{dla } \bar{b} - \underline{a} \geq 0. \end{cases} \quad (1.76)$$

Dowód Wzór (1.74) wynika bezpośrednio z lematu 1.10. Wzór (1.76) wynika z lematu 1.10 i równości (1.31). Przeprowadzimy dowód formuły (1.75).

Dla liczb trapezowych $\tilde{A}, \tilde{B}$ zachodzi $\bar{a}(0) = \bar{a} + \beta_A$, $\bar{b}(0) = \bar{b} + \beta_B$. Z lematów 1.5 i 1.6 otrzymujemy:

$$Poss(\tilde{A} > \tilde{B}) = 1 \Leftrightarrow \bar{a}(1) \geq \bar{b}(0) \Leftrightarrow \bar{a} \geq \bar{b} + \beta_B \Leftrightarrow \bar{b} - \bar{a} \leq -\beta_B,$$

$$Poss(\tilde{A} > \tilde{B}) = 0 \Leftrightarrow \bar{a}(0) \leq \bar{b}(1) \Leftrightarrow \bar{a} + \beta_A \leq \bar{b} \Leftrightarrow \bar{b} - \bar{a} \geq \beta_A,$$

a stąd warunek:

$$Poss(\tilde{A} > \tilde{B}) \in (0, 1) \Leftrightarrow \bar{b} - \bar{a} \in (-\beta_B, \beta_A).$$

Z lematu 1.9 otrzymujemy:

$$Poss(\tilde{A} > \tilde{B}) = \lambda \in (0, 1) \Leftrightarrow \bar{a}(\lambda) = \bar{b}(1 - \lambda) \Leftrightarrow \bar{a} + \beta_A(1 - \lambda) = \bar{b} + \beta_B\lambda.$$

Stąd, jeżeli $\bar{b} - \bar{a} \in (-\beta_B, \beta_A)$, to

$$Poss(\tilde{A} > \tilde{B}) = \frac{\beta_A + \bar{a} - \bar{b}}{\beta_A + \beta_B},$$

co kończy dowód. □

Jeżeli $\tilde{A}$ i $\tilde{B}$ są liczbami rozmytymi, których funkcje przynależności są ciągłe i ściśle monotoniczne w przedziałach $[\underline{a}(0), \underline{a}(1)]$, $[\bar{a}(1), \bar{a}(0)]$, $[\underline{b}(0), \underline{b}(1)]$, $[\bar{b}(1), \bar{b}(0)]$ (warunek ten spełniają na przykład liczby rozmyte typu $L - R$), to wszystkie indeksy mogą być obliczone z dowolną dokładnością ε za pomocą odpowiedniej procedury numerycznej. Procedury te są przedstawione w postaci algorytmów 1 i 2.

Algorytm 1 Obliczanie indeksu $Poss(\tilde{A} \geq \tilde{B})$ z dokładnością ε

Require: $\tilde{A}, \tilde{B}, \varepsilon \in (0, 1]$

Ensure: $Poss(\tilde{A} \geq \tilde{B})$

- 1: **if** $\bar{a}(1) \geq \underline{b}(1)$ **then return** 1
 - 2: $\lambda_1 \leftarrow 0.5, \lambda_2 \leftarrow 0, k \leftarrow 2$
 - 3: **while** $|\lambda_1 - \lambda_2| \geq \varepsilon$ **do**
 - 4: $\lambda_2 \leftarrow \lambda_1$
 - 5: **if** $\bar{a}(\lambda_1) \geq \underline{b}(\lambda_1)$ **then** $\lambda_1 \leftarrow \lambda_1 + \frac{1}{2^k}$ **else** $\lambda_1 \leftarrow \lambda_1 - \frac{1}{2^k}$
 - 6: $k \leftarrow k + 1$
 - 7: **end while**
 - 8: **return** λ_1
-

Algorytm 2 Obliczanie indeksu $Poss(\tilde{A} > \tilde{B})$ z dokładnością ε

Require: $\tilde{A}, \tilde{B}, \varepsilon \in (0, 1]$

Ensure: $Poss(\tilde{A} > \tilde{B})$

- 1: **if** $\bar{a}(1) \geq \bar{b}(\varepsilon)$ **then return** 1
 - 2: $\lambda_1 \leftarrow 0.5, \lambda_2 \leftarrow 0, k \leftarrow 2$
 - 3: **while** $|\lambda_1 - \lambda_2| \geq \varepsilon$ **do**
 - 4: $\lambda_2 \leftarrow \lambda_1$
 - 5: **if** $\bar{a}(\lambda_1) \geq \bar{b}(1 - \lambda_1)$ **then** $\lambda_1 \leftarrow \lambda_1 + \frac{1}{2^k}$ **else** $\lambda_1 \leftarrow \lambda_1 - \frac{1}{2^k}$
 - 6: $k \leftarrow k + 1$
 - 7: **end while**
 - 8: **return** λ_1
-

Poprawność algorytmów 1 i 2 wynika bezpośrednio z lematów 1.8 i 1.9. Jeżeli potrafimy obliczyć odpowiednie λ -poziomy liczb rozmytych $\tilde{A}$ i $\tilde{B}$ w czasie stałym (niezależnym od λ), to złożoność algorytmów 1 i 2 zależy wyłącznie od przyjętej dokładności ε i jest rzędu $O(|\log \varepsilon|)$. Zauważmy, że jeżeli $\tilde{A}$ i $\tilde{B}$ są liczbami rozmytymi typu $L - R$, to λ -poziom może być obliczony za pomocą formuły (1.3). Aby obliczyć indeks $Ness(\tilde{A} > \tilde{B})$ należy skorzystać z algorytmu 1 i równości (1.31). Aby obliczyć indeksy $Poss(\tilde{A} = \tilde{B})$ i $Ness(\tilde{A} \neq \tilde{B})$ należy skorzystać z algorytmu 1, definicji 1.13 oraz z warunku (1.32).

1.6 Wartość średnia liczby rozmytej

W punkcie 1.4 została określona zmienna losowa $Z_{\tilde{A}}$ związaną z liczbą rozmytą $\tilde{A}$ (patrz definicja 1.11). Korzystając z tego pojęcia zdefiniujemy teraz wartość średnią liczby rozmytej [5].

Definicja 1.14. *Wartością średnią liczby rozmytej $\tilde{A}$ nazywamy liczbę rzeczywistą $E(\tilde{A})$ określoną następująco:*

$$E(\tilde{A}) = E[Z_{\tilde{A}}],$$

gdzie $E[Z_{\tilde{A}}]$ jest wartością oczekiwaną zmiennej losowej $Z_{\tilde{A}}$.

Wartość średnią liczby rozmytej $\tilde{A}$ można obliczyć korzystając z następującego twierdzenia:

Twierdzenie 1.2. *Dla każdej liczby rozmytej $\tilde{A}$*

$$E(\tilde{A}) = \frac{1}{2} \int_0^1 (\underline{a}(t) + \bar{a}(t)) dt. \quad (1.77)$$

Z twierdzenia 1.2 wynika, że wartość średnia liczby rozmytej jest równoważna indeksowi wprowadzonemu przez Yagera, który w [35] określił go wprost za pomocą formuły (1.77). Wartość średnia liczby rozmytej istnieje wtedy i tylko wtedy gdy istnieje odpowiednia całka (1.77). Zachodzi następujący lemat:

Lemat 1.12. *Jeżeli $\tilde{A} = (\underline{a}, \bar{a}, \alpha_A, \beta_A)_{L-R}$ jest liczbą rozmytą typu $L-R$, a całki $\int_0^1 R^{-1}(t)dt$, $\int_0^1 L^{-1}(t)dt$ istnieją, to:*

$$E(\tilde{A}) = \frac{1}{2}(\underline{a} + \bar{a}) - \frac{1}{2}(\alpha_A \chi - \beta_A \psi), \quad (1.78)$$

gdzie $\chi = \int_0^1 L^{-1}(t)dt$, $\psi = \int_0^1 R^{-1}(t)dt$.

Dowód Z twierdzenia 1.2 i z formuły (1.3) otrzymujemy:

$$E(\tilde{A}) = \frac{1}{2} \int_0^1 (\underline{a} - L^{-1}(t)\alpha_A + \bar{a} + R^{-1}(t)\beta_A) dt.$$

Z założenia istnienia odpowiednich całek otrzymujemy:

$$E(\tilde{A}) = \frac{1}{2}(\underline{a} + \bar{a}) - \frac{1}{2} \left(\alpha_A \int_0^1 L^{-1}(t)dt - \beta_B \int_0^1 R^{-1}(t)dt \right),$$

czyli wzór (1.78) zachodzi. $\square$

Współczynniki χ oraz ψ mogą być łatwo obliczone dla potęgowych i wykładniczych funkcji kształtu. Dla funkcji POW_p wynoszą one $\frac{p}{p+1}$ a dla funkcji EXP_p są one równe $\frac{1}{p}$. Przykładem liczby rozmytej typu $L - R$, dla której wartość średnia nie istnieje, jest liczba z wymiernymi funkcjami kształtu RAT_p dla $p = 1$.

Z lematu 1.12 wynika, że wartość średnia istnieje dla każdych liczb rozmytych typu $L - R$ z potęgowymi i wykładniczymi funkcjami kształtu.

Własność 1.7. *Jeżeli $\tilde{A}$ i $\tilde{B}$ mają wartości średnie, to:*

$$E(\tilde{A} + \tilde{B}) = E(\tilde{A}) + E(\tilde{B}), \quad (1.79)$$

$$E(\tilde{A} - \tilde{B}) = E(\tilde{A}) - E(\tilde{B}), \quad (1.80)$$

$$E(c\tilde{A}) = cE(\tilde{A}), c \in \mathbb{R}. \quad (1.81)$$

Własność 1.8. *Jeżeli $\tilde{A}$ i $\tilde{B}$ mają wartości średnie to:*

$$E(\max\{\tilde{A}, \tilde{B}\}) \geq \max\{E(\tilde{A}), E(\tilde{B})\}. \quad (1.82)$$

Wartość średnia liczby rozmytej nie zachowuje operacji maksimum ponieważ można łatwo podać przykład liczb rozmytych $\tilde{A}, \tilde{B}$, dla których w formule (1.82) zachodzi ostra relacja większości. Jest to ważna własność, która będzie miała duże konsekwencje w dalszych rozważaniach.

Wyznaczenie wartości średniej z maksimum dwóch liczb rozmytych nie jest łatwe. Dla dowolnych liczb rozmytych typu $L - R$ nie da się wyprowadzić prostej i ogólnej formuły. Można natomiast udowodnić następujący lemat:

Lemat 1.13. *Jeżeli $\tilde{A} = (\underline{a}, \bar{a}, \alpha_A, \beta_A)_{L-R}$ jest liczbą rozmytą typu $L-R$, $L(y) = \max\{0, 1 - y^{p_1}\}$ i $R(y) = \max\{0, 1 - y^{p_2}\}$ (L i R są potęgowymi funkcjami kształtu), to:*

$$E(\max\{0, \tilde{A}\}) = \begin{cases} 0 & \text{dla } P > 1, \\ \frac{1}{2} \left((1 - P^{p_2})\bar{a} + \frac{p_2}{p_2+1}(1 - P^{p_2+1})\beta_A \right) & \text{dla } P \in [0, 1], \\ \frac{1}{2} \left(\bar{a} + \frac{p_2}{p_2+1}\beta_A \right) & \text{dla } P, Q < 0, \\ \frac{1}{2} \left(\bar{a} + Q^{p_1}\underline{a} + \frac{p_2}{p_2+1}\beta_A - \frac{p_1}{p_1+1}Q^{p_1+1}\alpha_A \right) & \text{dla } Q \in [0, 1], \\ \frac{1}{2} \left(\bar{a} + \underline{a} + \frac{p_2}{p_2+1}\beta_A - \frac{p_1}{p_1+1}\alpha_A \right) & \text{dla } Q > 1, \end{cases}$$

gdzie $P = -\frac{\bar{a}}{\beta_A}$ i $Q = \frac{\underline{a}}{\alpha_A}$.

Dowód Należy rozpatrzyć pięć przypadków pokazanych na rysunku 1.6. Każdy z nich przedstawia inne położenie funkcji przynależności liczby $\tilde{A}$ względem osi OY .

Oznaczmy $\tilde{M} = \max\{0, \tilde{A}\}$ i rozpatrzmy następujące przypadki:

$$1. \bar{a} < 0 \text{ i } 1 - \left(\frac{-\bar{a}}{\beta_A}\right)^{p_2} < 0 \Leftrightarrow \frac{-\bar{a}}{\beta_A} > 1 \Leftrightarrow P > 1 :$$

$$E(\tilde{M}) = 0.$$

$$2. \bar{a} \leq 0 \text{ i } 1 - \left(\frac{-\bar{a}}{\beta_A}\right)^{p_2} \geq 0 \Leftrightarrow \frac{-\bar{a}}{\beta_A} \in [0, 1] \Leftrightarrow P \in [0, 1] :$$

$$\begin{aligned} E(\tilde{M}) &= \frac{1}{2} \int_0^{1 - \left(\frac{-\bar{a}}{\beta_A}\right)^{p_2}} \bar{a}(t) dt = \frac{1}{2} \int_0^{1 - P^{p_2}} (\bar{a} + \beta_A(1 - t)^{\frac{1}{p_2}}) dt = \\ &= \frac{1}{2} \left((1 - P^{p_2})\bar{a} + \frac{p_2}{p_2+1}(1 - P^{p_2+1})\beta_A \right). \end{aligned}$$

$$3. \bar{a} > 0 \text{ i } \underline{a} < 0 \Leftrightarrow P < 0 \text{ i } Q < 0 :$$

$$E(\tilde{M}) = \frac{1}{2} \int_0^1 \bar{a}(t) dt = \frac{1}{2} \int_0^1 (\bar{a} + \beta_A(1 - t)^{\frac{1}{p_2}}) dt = \frac{1}{2} \left(\bar{a} + \frac{p_2}{p_2+1}\beta_A \right).$$

$$4. \underline{a} \geq 0 \text{ i } 1 - \left(\frac{\underline{a}}{\alpha_A}\right)^{p_1} \in [0, 1] \Leftrightarrow Q \in [0, 1] :$$

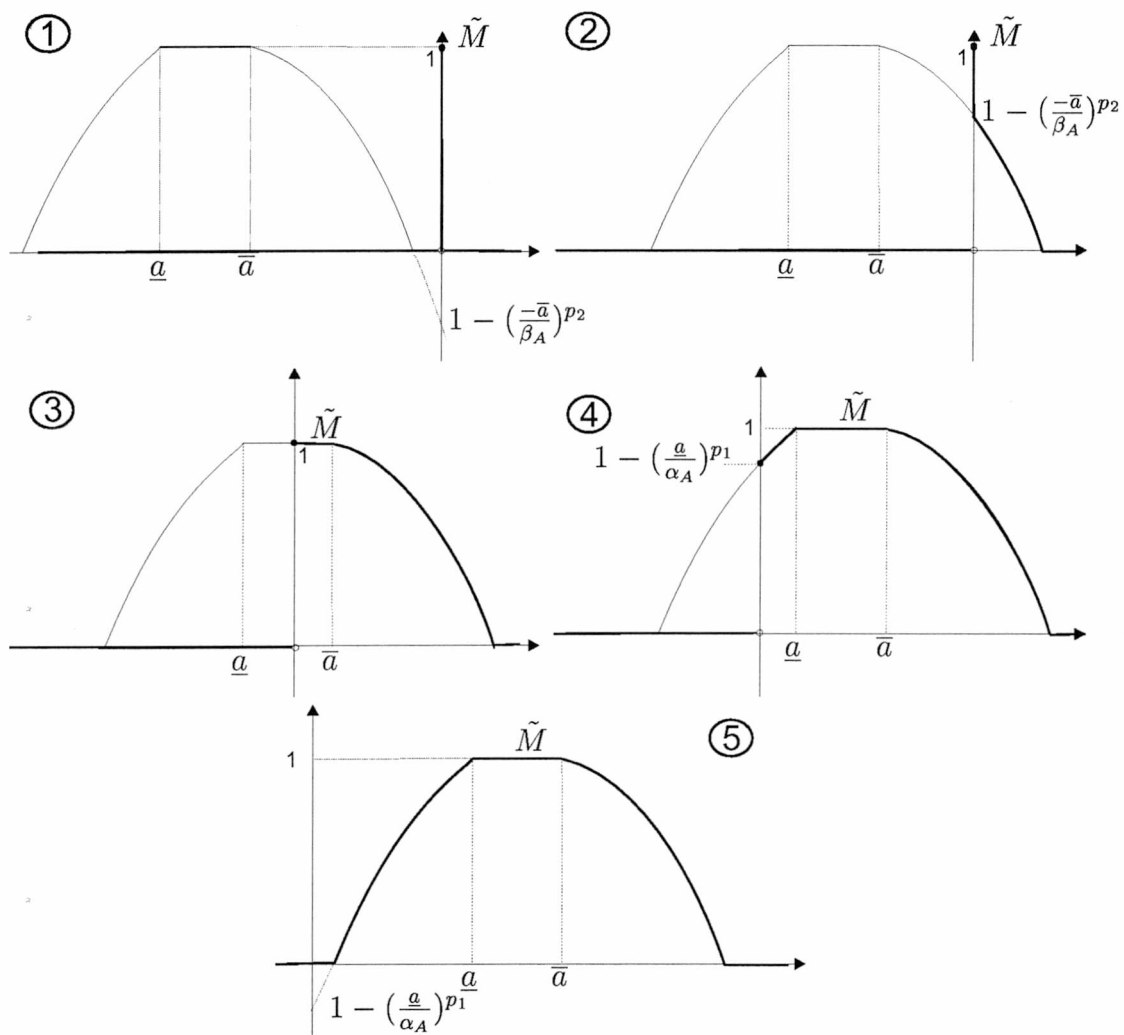
$$\begin{aligned} E(\tilde{M}) &= \frac{1}{2} \int_{1 - \left(\frac{\underline{a}}{\alpha_A}\right)^{p_1}}^1 \underline{a}(t) dt + \frac{1}{2} \int_0^1 \bar{a}(t) dt = \\ &= \frac{1}{2} \int_{1 - Q^{p_1}}^1 (\underline{a} - \alpha_A(1 - t)^{\frac{1}{p_1}}) dt + \frac{1}{2} \int_0^1 (\bar{a} + \beta_A(1 - t)^{\frac{1}{p_2}}) dt = \\ &= \frac{1}{2} \left(\bar{a} + Q^{p_1}\underline{a} + \frac{p_2}{p_2+1}\beta_A - \frac{p_1}{p_1+1}Q^{p_1+1}\alpha_A \right). \end{aligned}$$

$$5. \underline{a} \geq 0 \text{ i } 1 - \left(\frac{\underline{a}}{\alpha_A}\right)^{p_1} < 0 \Leftrightarrow Q > 1 :$$

$$E(\tilde{M}) = \frac{1}{2} \int_0^1 \underline{a}(t) dt + \frac{1}{2} \int_0^1 \bar{a}(t) dt = \frac{1}{2} \left(\bar{a} + \underline{a} + \frac{p_2}{p_2+1}\beta_A - \frac{p_1}{p_1+1}\alpha_A \right).$$

□

Lemat 1.13 zachodzi także dla liczb trapezowych i trójkątnych ponieważ funkcja liniowa jest szczególnym przypadkiem funkcji potęgowej.



Rys. 1.6: Przypadki analizowane w dowodzie lematu 1.13

Rozdział 2

Rozmyte problemy szeregowania na pojedynczej maszynie

W rozdziale tym teoria liczb rozmytych zostanie zastosowana do uogólnienia klasycznych problemów szeregowania na pojedynczej maszynie na przypadek rozmyty. Parametrami nieprecyzyjnie określonymi będą czasy trwania i pożądane terminy zakończenia prac. Będą one zadane w postaci liczb rozmytych, których funkcje przynależności są interpretowane jako rozkłady możliwości. Do konstrukcji różnych funkcji kosztu harmonogramu zostaną zastosowane indeksy Dubois i Prade oraz wartość średnia liczby rozmytej.

Struktura rozdziału jest następująca: w punkcie 2.1 zostanie przedstawiony przegląd znanych problemów szeregowania prac na pojedynczej maszynie, w których parametry są zdeterminowane. W punkcie 2.2 zostanie zdefiniowany rozmyty problem szeregowania prac na pojedynczej maszynie i podany szereg jego ogólnych własności. W kolejnych punktach zostanie zbadany zbiór konkretnych rozmytych problemów szeregowania. Szczególny nacisk będzie położony na analizę złożoności obliczeniowej, która ma kluczowe znaczenie w praktycznych zastosowaniach.

2.1 Przegląd klasycznych problemów szeregowania prac na pojedynczej maszynie

Szczególnie ważne miejsce w klasie wszystkich problemów szeregowania ze zdeterminowanymi parametrami zajmują te problemy, w których prace są wykonywane na jednej maszynie. Najbardziej ogólnie problem szeregowania na pojedynczej maszynie może być sformułowany następująco [3]: dany jest zbiór $J = \{1, \dots, n\}$

prac, które mają być wykonywane na maszynie M . Zakładamy, że maszyna M może wykonywać w danej chwili tylko jedną pracę. Z każdą pracą ze zbioru J związany jest szereg parametrów, z których najczęściej stosowanymi są:

1. t_i - czas trwania pracy $i \in J$,
2. d_i - pożądaný termin zakończenia pracy $i \in J$,
3. r_i - czas, w którym praca $i \in J$ staje się dostępna,
4. w_i - waga określająca ważność pracy $i \in J$.

W klasycznym przypadku wszystkie te parametry są zdeterminowane (najczęściej zakłada się, że są one liczbami całkowitymi).

Często narzucany jest pewien częściowy porządek wykonywania prac. Oznacza to, że dla każdej pracy $i \in J$ można podać zbiór prac, które nie mogą rozpocząć się przed zakończeniem wykonywania pracy i . Porządek ten zadany jest przez acykliczną relację $prec \subset J \times J$. Jeżeli $(i, j) \in prec$, to wykonywanie pracy j nie może rozpocząć się przed zakończeniem wykonywania pracy i .

Prace ze zbioru J mogą być podzielne albo niepodzielne. Jeżeli prace są podzielne to wykonywanie danej pracy na maszynie M może zostać przerwane i dokończone w późniejszym terminie tak aby łączny czas wykonywania pracy $i \in J$ wynosił t_i . Jeżeli prace są niepodzielne, to muszą być one wykonywane w całości w sposób ciągły.

Harmonogramem p nazywamy przyporządkowanie każdej jednostce czasowej na maszynie M pewnej pracy ze zbioru J w ten sposób, że łączny czas wykonywania każdej pracy $i \in J$ jest równy t_i . Harmonogram jest dopuszczalny jeżeli jest w nim spełniona relacja poprzedzania $prec$ i nie są naruszone inne dodatkowe ograniczenia (np. prace są rozpoczynane nie wcześniej niż zadane terminy dostępności r_i dla $i \in J$.)

Dla każdego dopuszczalnego harmonogramu p można wyznaczyć czas zakończenia pracy $i \in J$ w tym harmonogramie. Oznaczmy ten czas przez $C_i(p)$. Dla każdej pracy definiuje się ponadto funkcję $f_i : \mathbb{R} \rightarrow \mathbb{R}$, której wartość $f_i(t)$ określa koszt zakończenia pracy $i \in J$ w czasie t . Najczęściej określone są następujące funkcje kosztu:

$$\begin{aligned} L_i(p) &= C_i(p) - d_i && - \text{nieterminowość pracy } i \in J, \\ T_i(p) &= \max\{0, C_i(p) - d_i\} && - \text{opóźnienie pracy } i \in J, \\ U_i(p) &= \begin{cases} 0 & \text{jeżeli } C_i(p) \leq d_i \\ 1 & \text{jeżeli } C_i(p) > d_i \end{cases} && - \text{kara jednostkowa dla pracy } i \in J. \end{aligned}$$

Dla każdego problemu szeregowania definiuje się również funkcję $f : \mathbb{R}^n \rightarrow \mathbb{R}$, której wartość $f(C_1(p), \dots, C_n(p))$ określa koszt zakończenia wszystkich prac w harmonogramie p . Najczęściej funkcja ta przyjmuje jedną z dwóch postaci:

$$f(C_1(p), \dots, C_n(p)) = \max_{i \in J} \{w_i f_i(C_i(p))\},$$

$$f(C_1(p), \dots, C_n(p)) = \sum_{i \in J} w_i f_i(C_i(p)).$$

Celem problemu szeregowania jest znalezienie takiego harmonogramu p^* , dla którego funkcja kosztu jest najmniejsza.

Dla problemów szeregowania stworzono specjalną notację zwaną notacją Grahama ([3],[14]). Każdy problem szeregowania na pojedynczej maszynie można oznaczyć poprzez ciąg symboli postaci $1|\beta_1\beta_2\beta_3\beta_4|\gamma$ gdzie:

1. β_1 określa czy dozwolony jest podział prac. Jeżeli tak, to $\beta_1 = pmnt$ w przeciwnym wypadku parametr ten nie występuje.
2. β_2 określa typ relacji poprzedzania $prec$. Relację tą można przedstawić w postaci acyklicznego grafu skierowanego $G = (V, A)$, gdzie $V \subseteq J$ i $(i, j) \in A \Leftrightarrow (i, j) \in prec$. W teorii szeregowania wyróżnia się następujące szczególne przypadki relacji $prec$:
 - (a) **chains** - jeżeli każdy wierzchołek grafu G ma co najwyżej jeden poprzednik i co najwyżej jeden następnik.
 - (b) **in – tree** - jeżeli graf G jest drzewem, którego każdy wierzchołek ma co najwyżej jeden następnik.
 - (c) **out – tree** - jeżeli graf G jest drzewem, którego każdy wierzchołek ma co najwyżej jeden poprzednik.
 - (d) **sp – graph** - jeżeli graf G jest grafem szeregowo-równoległym.
3. $\beta_3 = r_i$ jeżeli dla każdej pracy $i \in J$ są określone terminy dostępności. Jeżeli wszystkie prace są dostępne od chwili 0, to parametr ten nie występuje.
4. β_4 określa ograniczenia na czasy trwania prac. Na przykład jeżeli parametr ten ma postać $t_i = 1$, to oznacza to, że wszystkie czasy trwania prac są jednostkowe. Jeżeli czasy te mogą być dowolnymi liczbami to parametr β_4 nie występuje.

5. γ opisuje funkcję celu, czyli koszt zakończenia wszystkich prac w zadanym harmonogramie. Przyjmuje się oznaczenie $\gamma = f_{max}$ jeżeli funkcja kosztu jest postaci $\max_{i \in J} \{f_i(C_i(p))\}$ i wszystkie funkcje f_i są jednakowe. Na przykład $\gamma = T_{max}$ oznacza, że funkcja celu jest postaci $\max_{i \in J} \{T_i(p)\}$.

Na przykład problem $1|pmnt; chains; r_i| \sum w_i C_i$ oznacza problem szeregowania na pojedynczej maszynie, w którym relacja poprzedzania jest typu *chains*, dopuszczalny jest podział prac i są zadane czasy dostępności r_i dla każdej pracy. Wartość funkcji celu dla harmonogramu p jest równa $\sum_{i \in J} w_i C_i(p)$.

Kluczowe znaczenie w praktycznych zastosowaniach problemów szeregowania ma ich złożoność obliczeniowa. Najważniejsze rezultaty dotyczące złożoności obliczeniowej problemów szeregowania na pojedynczej maszynie są przedstawione w tablicy 2.1 [3].

Problemy wielomianowe		Problemy NP-trudne
$1 prec; r_i C_{max}$	$O(n^2)$	$*1 r_i L_{max}$
$1 prec; r_i; t_i = t L_{max}$	$O(n^3 \log n)$	$*1 r_i \sum C_i$
$1 prec f_{max}$	$O(n^2)$	$*1 prec \sum C_i$
$1 prec; r_i; t_i = 1 f_{max}$	$O(n^2)$	$*1 pmnt; chains; r_j \sum C_i$
$1 pmnt; prec; r_i f_{max}$	$O(n^2)$	$*1 prec; t_i = 1 \sum w_i C_i$
$1 pmnt; r_i \sum C_i$	$O(n \log n)$	$*1 chains; r_i; t_i = 1 \sum w_i C_i$
$1 prec; r_i; t_i = t \sum C_i$	$O(n^2)$	$*1 pmnt; r_i \sum w_i C_i$
$1 sp - graph \sum w_i C_i$	$O(n \log n)$	$*1 chains; t_i = 1 \sum U_i$
$1 \sum U_i$	$O(n \log n)$	$1 \sum w_i U_i$
$1 pmnt; r_i \sum U_i$	$O(n^5)$	$1 \sum T_i$
$1 r_i; t_i = 1 \sum f_i$	$O(n^3)$	$*1 \sum w_i T_i$

Tab. 2.1: Złożoność obliczeniowa najważniejszych problemów szeregowania na pojedynczej maszynie. Problemy silnie NP-trudne oznaczone są symbolem *. Uwaga: w problemach dla których funkcja celu jest postaci f_{max} zakłada się, że wszystkie funkcje kosztu f_i są niemalejące (źródło: [3]).

2.2 Definicja rozmytego problemu szeregowania na pojedynczej maszynie

Podamy teraz ogólną definicję rozmytego problemu szeregowania na pojedynczej maszynie, który jest głównym przedmiotem badań w tym rozdziale. Dany jest zbiór $J = \{1, 2, \dots, n\}$ prac, które mają być wykonywane na pojedynczej maszynie. Każdej pracy $i \in J$ przypisujemy następujące rozmyte parametry:

- $\tilde{t}_i$ - czas trwania pracy $i \in J$,
- $\tilde{d}_i$ - pożądany termin zakończenia pracy $i \in J$,
- w_i - waga charakteryzująca ważność pracy $i \in J$.

Czasy trwania prac $\tilde{t}_i$ są dodatnimi liczbami rozmytymi, pożądane terminy zakończenia prac $\tilde{d}_i$ są dowolnymi liczbami rozmytymi, natomiast wagi w_i są zdeteminowane. Zakładamy ponadto, że wszystkie prace należące do zbioru J są dostępne od chwili 0.

Definicja 2.1. *Harmonogramem nazywamy listę $p = ((i_1, \tilde{s}_1), \dots, (i_l, \tilde{s}_l))$, która spełnia następujące warunki:*

1. $\tilde{s}_i$ dla $i = 1, \dots, l$ są dodatnimi liczbami rozmytymi,
2. $i_l \in J$, $i_j \in J \cup \{\circ\}$, $i_j \neq i_{j+1}$ dla $j = 1, \dots, l-1$,
3. $\sum_{\{k: i_k=j\}} \tilde{s}_k = \tilde{t}_j$ dla $j \in J$.

Listę p interpretujemy następująco: najpierw na maszynie wykonujemy pracę i_1 przez czas $\tilde{s}_1$, następnie pracę i_2 przez czas $\tilde{s}_2$ itd. Jeżeli $i_k = \circ$, to oznacza, że maszyna przez czas $\tilde{s}_k$ nie wykonuje żadnej pracy. Definicja 2.1 jest uogólnieniem pojęcia harmonogramu, który występuje w klasycznych problemach szeregowania. Warunek 2. gwarantuje, że dwa sąsiednie elementy harmonogramu dotyczą różnych prac (w przeciwnym wypadku można byłoby otrzymać identyczny harmonogram łącząc oba elementy). Warunek ten zapewnia również aby ostatni element harmonogramu nie był przestojem maszyny. Z warunku 3. wynika, że łączny czas wykonywania pracy $i \in J$ jest równy $\tilde{t}_i$.

Dla zadanego harmonogramu $p = ((i_1, \tilde{s}_1), \dots, (i_l, \tilde{s}_l))$ zdefiniujemy:

$$\underline{i}(p) = \max\{k : \forall_{1 \leq j < k} i_j \neq i\}, \quad (2.1)$$

$$\bar{i}(p) = \min\{k : \forall_{l \geq j > k} i_j \neq i\}, \quad (2.2)$$

$$p^i = \{j \in J : \bar{j}(p) \leq \bar{i}(p)\}. \quad (2.3)$$

Liczby naturalne $\underline{i}(p)$, $\bar{i}(p)$ oznaczają odpowiednio, pierwszą i ostatnią pozycję pracy i w harmonogramie p . Do zbioru p^i należy praca i oraz wszystkie prace, których ostatnie pozycje w harmonogramie p występują przed ostatnią pozycją pracy i .

Zdefiniujmy czas rozpoczęcia $\tilde{S}_i(p)$ i czas zakończenia $\tilde{C}_i(p)$ pracy $i \in J$ w harmonogramie p następująco:

$$\tilde{S}_i(p) = \begin{cases} \sum_{1 \leq k < \underline{i}(p)} \tilde{s}_k & \text{dla } \underline{i}(p) > 1, \\ 0 & \text{dla } \underline{i}(p) = 1, \end{cases} \quad (2.4)$$

$$\tilde{C}_i(p) = \sum_{1 \leq k \leq \bar{i}(p)} \tilde{s}_k. \quad (2.5)$$

Definicja 2.2. Mówimy, że w harmonogramie p występuje **podział prac** jeżeli istnieje praca $i \in J$, która znajduje się w więcej niż jednym elemencie listy p .

Podział prac oznacza, że praca i może zostać wykonana częściowo w jednym przedziale czasowym i dokończona w innym. Jeżeli w harmonogramie nie występuje podział prac, to każda praca i jest wykonywana tylko w jednym przedziale czasowym i $\tilde{C}_i(p) = \tilde{S}_i(p) + \tilde{t}_i$.

Definicja 2.3. Mówimy, że w harmonogramie p występują **przestoje maszyny** jeżeli na liście p istnieje co najmniej jeden element postaci $(\circ, \tilde{s}_k)$.

Przykład: Dany jest zbiór prac $J = \{1, 2, 3, 4\}$, dla których czasy trwania są zadane w postaci następujących trójkątnych liczb rozmytych:

$$\tilde{t}_1 = (5, 2, 3); \tilde{t}_2 = (4, 3, 2); \tilde{t}_3 = (6, 3, 3); \tilde{t}_4 = (2, 2, 3).$$

Rozpatrzmy następujący harmonogram:

$$p = ((\mathbf{2}, (2, 1, 1)), (\mathbf{1}, (5, 2, 3)), (\circ, (1, 1, 2)), (\mathbf{4}, (1, 1, 2)), (\mathbf{2}, (2, 2, 1)), (\mathbf{3}, (6, 3, 3)), (\mathbf{4}, (1, 1, 1))).$$

Rozmyte czasy rozpoczęcia i zakończenia wszystkich prac w tym harmonogramie są przedstawione w tablicy 2.2

Zauważmy, że w harmonogramie p występuje podział prac ponieważ prace 2 i 4 powtarzają się na liście dwa razy. W harmonogramie występują również przestoje maszyny (zawiera on element $(\circ, (1, 1, 2))$). $\square$

$i \in J$	$\underline{i}(p)$	$\bar{i}(p)$	p^i	$\tilde{S}_i(p)$	$\tilde{C}_i(p)$
1	2	2	$\{1\}$	$(2, 1, 1)$	$(7, 3, 4)$
2	1	5	$\{1, 2\}$	0	$(11, 7, 9)$
3	6	6	$\{1, 2, 3\}$	$(11, 7, 9)$	$(17, 10, 12)$
4	4	7	$\{1, 2, 3, 4\}$	$(8, 4, 6)$	$(18, 11, 13)$

Tab. 2.2: Czasy rozpoczęcia i zakończenia wszystkich prac w przykładowym harmonogramie.

Definicja 2.4. Niech $prec \subset J \times J$ będzie acykliczną i przechodnią¹ relacją, którą nazywamy relacją poprzedzania. Mówimy, że harmonogram p spełnia relację poprzedzania $prec$ jeżeli spełniony jest następujący warunek:

$$(i, j) \in prec \Rightarrow \underline{j}(p) > \bar{i}(p).$$

W przypadku gdy parametry są zdeterminowane, warunek $\underline{j}(p) > \bar{i}(p)$ jest równoważny warunkowi $S_j(p) \geq C_i(p)$ (czas rozpoczęcia pracy j jest większy bądź równy czasowi zakończenia pracy i). W przypadku rozmytym wyrażenie $\tilde{S}_j(p) \geq \tilde{C}_i(p)$ nie jest dobrze określone ponieważ zbioru liczb rozmytych nie można w naturalny sposób uporządkować.

Harmonogram p nazywamy **dopuszczalnym** jeżeli spełnia on narzuconą relację poprzedzania. Z acykliczności relacji $prec$ wynika, że zawsze istnieje co najmniej jeden dopuszczalny harmonogram. Jeżeli natomiast $prec = \emptyset$, to każdy harmonogram jest dopuszczalny.

Założmy, że dana jest funkcja, określona na zbiorze $FN(\mathfrak{R})^n$ i przyjmująca wartości ze zbioru liczb rzeczywistych, $F : FN(\mathfrak{R})^n \rightarrow \mathfrak{R}$. Wartość funkcji $F(\tilde{C}_1(p), \dots, \tilde{C}_n(p))$ określa **koszt harmonogramu** p . Koszt harmonogramu p zależy więc w ogólnym przypadku od rozmytych czasów zakończenia wszystkich prac należących do zbioru J . **Rozmytym problemem szeregowania na pojedynczej maszynie** nazywamy zagadnienie wyznaczenia dopuszczalnego harmonogramu p dla którego:

$$F(\tilde{C}_1(p), \dots, \tilde{C}_n(p)) \rightarrow \min.$$

Funkcję kosztu harmonogramu będziemy oznaczać również skróconym oznaczeniem $F(p)$.

¹ $(i, j) \in prec$ i $(j, k) \in prec \Rightarrow (i, k) \in prec$.

W punkcie 2.1 zaprezentowana została notacja Grahama, służąca do oznaczania klasycznych problemów szeregowania na pojedynczej maszynie. Zmodyfikujemy teraz tą notację tak, aby można było za jej pomocą oznaczać problemy rozmyte. Należy podkreślić, że nie jest to notacja uniwersalna a zdefiniowana jedynie dla potrzeb niniejszej pracy.

Każdy, konkretny rozmyty problem szeregowania będzie oznaczany ciągiem symboli $1||\beta_1, \beta_2, \beta_3|\gamma$, gdzie:

1. β_1 - typ relacji poprzedzania; $\beta_1 \in \{prec, chains, in - tree, out - tree, sp - graph\}$. Jeżeli $\beta_1 = prec$, to relacja poprzedzania jest dowolna. Jeżeli w problemie nie narzuca się żadnej relacji poprzedzania, to parametr β_1 nie występuje.
2. $\beta_2 = pmtn$ jeżeli w problemie jest dopuszczalny podział prac i przestoje maszyny. W przeciwnym wypadku parametr β_2 nie występuje.
3. β_3 -typ rozmytych parametrów $\tilde{t}_i, \tilde{d}_i$; $\beta_3 \in \{FN, L - R, PL - PR, TR, TG\}$:
 - (a) FN - dowolne liczby rozmyte,
 - (b) $L - R$ - liczby rozmyte typu $L - R$,
 - (c) $PL - PR$ - liczby rozmyte typu $L - R$ z potęgowymi funkcjami kształtu,
 - (d) TR - liczby trapezowe,
 - (e) TG - liczby trójkątne.

W niektórych problemach przyjmowane będą pewne dodatkowe założenia dotyczące typu parametrów. Założenia te będą podawane w tych punktach, w których dany problem będzie szczegółowo analizowany.

4. γ - Opis funkcji kosztu harmonogramu.

Na przykład, $1|in - tree, TR|E(\sum w_i \tilde{C}_i)$ oznacza problem, w którym nie jest dopuszczalny podział prac, relacja poprzedzania jest typu $in - tree$, parametry $\tilde{t}_i$ oraz $\tilde{d}_i$ są trapezowymi liczbami rozmytymi a koszt harmonogramu p jest równy $E(\sum_{i=1}^n w_i \tilde{C}_i(p))$.

2.3 Charakterystyka funkcji kosztu harmonogramu w rozmytym problemie szeregowania

Zdefiniowanie odpowiedniej funkcji kosztu harmonogramu jest najważniejszym i jednocześnie najtrudniejszym zadaniem podczas formułowania rozmytego prob-

lemu szeregowania. W punkcie tym podamy kilka definicji i twierdzeń charakteryzujących w ogólny sposób funkcję kosztu harmonogramu.

Definicja 2.5. Niech F będzie dowolną funkcją określoną w zbiorze $FN(\mathbb{R})^n$ i przyjmującą wartości rzeczywiste. Funkcję tą nazywamy f -monotoniczną względem i -tego argumentu wtedy i tylko wtedy, gdy dla każdego wektora $(\tilde{x}_1, \dots, \tilde{x}_n) \in FN(\mathbb{R})^n$ i dowolnej, nieujemnej liczby rozmytej $\tilde{y} \in FN(\mathbb{R})$ zachodzi warunek:

$$F(\tilde{x}_1, \dots, \tilde{x}_i + \tilde{y}, \dots, \tilde{x}_n) \geq F(\tilde{x}_1, \dots, \tilde{x}_i, \dots, \tilde{x}_n).$$

Pojęcie f -monotoniczności jest uogólnieniem pojęcia monotoniczności. Jeżeli ograniczymy dziedzinę f -monotonicznej funkcji F do liczb rzeczywistych, to jest ona niemalejąca w klasycznym sensie względem zadanego argumentu.

Lemat 2.1. Jeżeli funkcje $F_1 : FN(\mathbb{R}) \rightarrow \mathbb{R}$ i $F_2 : FN(\mathbb{R}) \rightarrow \mathbb{R}$ są f -monotoniczne względem swoich argumentów, to funkcja $F : FN(\mathbb{R})^2 \rightarrow \mathbb{R}$ o wartościach

$$F(\tilde{x}, \tilde{y}) = F_1(\tilde{x}) + F_2(\tilde{y})$$

jest f -monotoniczna względem każdego ze swoich argumentów.

Dowód Jeżeli $\tilde{A}$ jest nieujemną liczbą rozmytą, to dla każdego $\tilde{x}, \tilde{y}$ jest spełniony warunek:

$$F(\tilde{x} + \tilde{A}, \tilde{y}) = F_1(\tilde{x} + \tilde{A}) + F_2(\tilde{y}) \geq F_1(\tilde{x}) + F_2(\tilde{y}) = F(\tilde{x}, \tilde{y}),$$

czyli F jest f -monotoniczną funkcją względem $\tilde{x}$. Dowód f -monotoniczności funkcji F względem argumentu $\tilde{y}$ jest analogiczny. $\square$

Lemat 2.2. Jeżeli funkcje $F_1 : FN(\mathbb{R}) \rightarrow \mathbb{R}$ i $F_2 : FN(\mathbb{R}) \rightarrow \mathbb{R}$ są f -monotoniczne względem swoich argumentów, to funkcja $F : FN(\mathbb{R})^2 \rightarrow \mathbb{R}$ o wartościach

$$F(\tilde{x}, \tilde{y}) = \max\{F_1(\tilde{x}), F_2(\tilde{y})\}$$

jest f -monotoniczna względem każdego ze swoich argumentów.

Dowód Jeżeli $\tilde{A}$ jest nieujemną liczbą rozmytą, to dla każdego $\tilde{x}, \tilde{y}$ zachodzi warunek:

$$F(\tilde{x} + \tilde{A}, \tilde{y}) = \max\{F_1(\tilde{x} + \tilde{A}), F_2(\tilde{y})\} \geq \max\{F_1(\tilde{x}), F_2(\tilde{y})\} = F(\tilde{x}, \tilde{y}),$$

czyli F jest f -monotoniczną funkcją względem $\tilde{x}$. Dowód f -monotoniczności funkcji F względem argumentu $\tilde{y}$ jest analogiczny. $\square$

Lematy 2.1 i 2.2 mogą być łatwo uogólnione na sumę i maksimum więcej niż dwóch funkcji.

Definicja 2.6. Funkcję kosztu harmonogramu $F(p)$ nazywamy f -regularną jeżeli jest ona f -monotoniczna względem każdego ze swoich argumentów, tj. dla każdej pracy $i \in J$ jest ona f -monotoniczną funkcją względem czasu zakończenia tej pracy (czasu $\tilde{C}_i(p)$).

Pojęcie f -regularności jest uogólnieniem pojęcia regularności funkcji kosztu w przypadku klasycznych i zdeterminowanych problemów szeregowania (patrz [3]). Jeżeli ograniczymy dziedzinę wszystkich rozmytych parametrów problemu do liczb rzeczywistych, to funkcja f -regularna jest funkcją regularną w klasycznym sensie.

Dla danego problemu koszt harmonogramu powinien być funkcją f -regularną jeżeli decydentowi zależy na możliwie szybkim zakończeniu każdej z prac. Pojęcie f -regularności ma ponadto duże znaczenie teoretyczne.

Podamy teraz ważne własności rozmytego problemu szeregowania, wynikające z f -regularności funkcji kosztu harmonogramu.

Lemat 2.3. Jeżeli dla danego rozmytego problemu szeregowania funkcja kosztu harmonogramu jest f -regularna, to dla każdego dopuszczalnego harmonogramu p można skonstruować dopuszczalny harmonogram bez przestojów maszyny, którego koszt jest niewiekszy niż koszt p .

Dowód Załóżmy, że p jest dopuszczalnym harmonogramem, w którym na k -tym miejscu występuje przestój maszyny, tj. element postaci $(\circ, \tilde{s}_k)$:

$$p = ((i_1, \tilde{s}_1), \dots, (\circ, \tilde{s}_k), \dots, (i_l, \tilde{s}_l)),$$

gdzie $1 \leq k < l$.

Utwórzmy harmonogram p_1 , usuwając z p element $(\circ, \tilde{s}_k)$. Łatwo zauważyć, że harmonogram p_1 jest również dopuszczalny. Otrzymujemy:

$$\tilde{C}_i(p) = \begin{cases} \tilde{C}_i(p_1) & \text{jeżeli } \bar{i}(p) \leq k, \\ \tilde{C}_i(p_1) + \tilde{s}_k & \text{jeżeli } \bar{i}(p) > k. \end{cases} \quad (2.6)$$

Widać, że rozmyty czas zakończenia każdej pracy i w harmonogramie p jest albo równy czasowi zakończenia pracy i w harmonogramie p_1 albo jest zwiększony o nieujemną liczbę rozmytą $\tilde{s}_k$. Ponieważ, z założenia, koszt harmonogramu jest funkcją f -regularną więc koszt harmonogramu p_1 jest niewiekszy niż koszt harmonogramu p . W analogiczny sposób można usunąć wszystkie elementy postaci $(\circ, \tilde{s}_i)$ z harmonogramu p_1 bez zwiększania wartości funkcji kosztu, otrzymując nie gorszy i dopuszczalny harmonogram bez przestojów maszyny. $\square$

Założenie f -regularności funkcji kosztu w lemacie 2.3 jest istotne. W przypadku, gdy funkcja kosztu harmonogramu nie jest f -regularna, może się okazać, że wprowadzenie przestoju maszyny spowoduje zmniejszenie kosztu.

Lemat 2.4. *Jeżeli dla danego rozmytego problemu szeregowania funkcja kosztu harmonogramu jest f -regularna, to dla każdego dopuszczalnego harmonogramu p można skonstruować dopuszczalny harmonogram bez podziału prac, którego koszt jest nie większy niż koszt p .*

Dowód Załóżmy, że p jest dopuszczalnym harmonogramem, w którym praca j występuje na pozycji u i v , gdzie $v > u + 1$. Załóżmy ponadto, że w harmonogramie p nie występują przestoje maszyny (w przeciwnym wypadku z lematu 2.3 wynika, że można usunąć z p wszystkie przestoje bez zwiększania wartości funkcji kosztu). Harmonogram p ma następującą postać:

$$p = ((i_1, \tilde{s}_1), \dots, (i_{u-1}, \tilde{s}_{u-1}), (j, \tilde{s}_u), (i_{u+1}, \tilde{s}_{u+1}), \dots, (j, \tilde{s}_v), \dots, (i_l, \tilde{s}_l)), \quad (2.7)$$

gdzie $i_u = i_v = j$, $u \in \{1, \dots, l-2\}$, $v \in \{u+2, \dots, l\}$. Utwórzmy harmonogram p_1 , którego postać jest następująca:

$$p_1 = ((i_1, \tilde{s}_1), \dots, (i_{u-1}, \tilde{s}_{u-1}), (i_{u+1}, \tilde{s}_{u+1}), \dots, (j, \tilde{s}_v \tilde{+} \tilde{s}_u), \dots, (i_l, \tilde{s}_l)).$$

Harmonogram p_1 jest również dopuszczalny ponieważ dla każdych prac $i, k \in J$ zachodzi:

$$(i, k) \in prec \Rightarrow \underline{k}(p) > \bar{i}(p) \Rightarrow \underline{k}(p_1) > \bar{i}(p_1).$$

Otrzymujemy następującą zależność między harmonogramami p i p_1 :

$$\tilde{C}_i(p) = \begin{cases} \tilde{C}_i(p_1) & \text{jeżeli } \bar{i}(p) < u \text{ lub } \bar{i}(p) \geq v, \\ \tilde{C}_i(p_1) \tilde{+} \tilde{s}_u & \text{jeżeli } u \leq \bar{i}(p) < v. \end{cases} \quad (2.8)$$

Oznacza to, że rozmyty czas zakończenia każdej pracy i w harmonogramie p jest albo równy czasowi zakończenia pracy i w harmonogramie p_1 albo jest zwiększony o nieujemną liczbę rozmytą $\tilde{s}_u$. Ponieważ, z założenia, koszt harmonogramu jest funkcją f -regularną więc koszt harmonogramu p_1 jest nie większy niż koszt harmonogramu p . Zauważmy, że liczba elementów na liście p_1 jest o jeden mniejsza niż liczba elementów na liście p . Jeżeli liczba elementów na liście p_1 jest równa liczbie prac, to otrzymaliśmy harmonogram bez podziału prac nie gorszy niż p . W przeciwnym wypadku w harmonogramie p_1 występuje nadal podział prac i całe rozumowanie możemy powtórzyć. W ten sposób otrzymujemy algorytm konstruujący harmonogram bez podziału prac, którego koszt jest nie gorszy niż koszt zadanego harmonogramu p . $\square$

W przypadku, gdy funkcja kosztu harmonogramu nie jest f -regularna, może się okazać, że dopuszczenie podziału prac spowoduje zmniejszenie kosztu. Lematy 2.3 i 2.4 są oczywiście prawdziwe również wtedy, gdy wszystkie parametry problemu są zdeterminowane. W takim przypadku otrzymujemy własności podane w [3] dla klasycznych problemów szeregowania na pojedynczej maszynie.

Udowodnimy teraz następujące, ważne twierdzenie:

Twierdzenie 2.1. *Jeżeli dla danego rozmytego problemu szeregowania funkcja kosztu harmonogramu jest f -regularna, to problem ten ma rozwiązanie optymalne² bez podziału prac i bez przestojów maszyny.*

Dowód Załóżmy nie wprost, że problem nie ma rozwiązania optymalnego. Oznacza to, że dla każdego dopuszczalnego harmonogramu p istnieje dopuszczalny harmonogram p_1 , którego koszt jest mniejszy niż koszt p . Oznaczmy przez Π zbiór wszystkich dopuszczalnych harmonogramów problemu, w których nie występują podział prac i przestoje maszyny. Zauważmy, że zbiór Π jest skończony. Oznaczmy przez p^* harmonogram należący do Π , którego koszt jest najmniejszy. Z założenia nie wprost wynika, że istnieje harmonogram p_1 , którego koszt jest mniejszy niż p^* . Z f -regularności funkcji kosztu otrzymujemy (patrz lematy 2.3 i 2.4), że istnieje harmonogram p_2 bez podziału prac i przestojów maszyny o koszcie niewiększym niż koszt p_1 . Harmonogram p_2 również należy do Π , co jest sprzeczne z faktem, że harmonogram p^* ma najmniejszy koszt w zbiorze Π .

Pokazaliśmy, że problem z f -regularną funkcją kosztu harmonogramu ma optymalne rozwiązanie. Z lematów 2.3 i 2.4 wynika, że rozwiązanie to musi należeć do zbioru Π , z czego otrzymujemy, że optymalnym rozwiązaniem problemu jest $p^* \in \Pi$. $\square$

Z udowodnionych twierdzeń wynika, że szukając optymalnego rozwiązania dla rozmytego problemu szeregowania z f -regularną funkcją kosztu, wystarczy ograniczyć zbiór rozwiązań dopuszczalnych tylko do tych harmonogramów, w których nie występują podział prac i przestoje maszyny. Zauważmy, że w takim przypadku każdy harmonogram p jest jednoznacznie określony przez pewną permutację σ zbioru J , tj.:

$$p = ((i_1, \tilde{t}_{i_1}), \dots, (i_n, \tilde{t}_{i_n})) \equiv (i_1, \dots, i_n), \quad (2.9)$$

²W ogólnym przypadku problem może nie mieć rozwiązania optymalnego. Na przykład, jeżeli funkcja kosztu jest równa $-\sum_{i \in J} E(\tilde{C}_i(p))$, to dla każdego harmonogramu p można utworzyć harmonogram lepszy dodając w dowolnym miejscu przestój maszyny.

gdzie $i_j \in J$ dla $j = 1, \dots, n$. W dalszej części pracy w problemach tego typu będziemy używali zamiast harmonogramu określeń **permutacja prac** lub **sekwencja prac**.

Udowodnimy teraz kilka dodatkowych, użytecznych własności wartości średniej liczby rozmytej oraz indeksów Dubois i Prade, które wykorzystamy w dalszym ciągu pracy do konstrukcji różnych funkcji kosztu harmonogramu.

Niech $\tilde{F} : FN(\mathfrak{R})^n \rightarrow FN(\mathfrak{R})$ będzie funkcją określoną w zbiorze $FN(\mathfrak{R})^n$ i przyjmującą wartości rozmyte. Oznaczmy:

$$(\tilde{F}(\tilde{x}_1, \dots, \tilde{x}_n))^\lambda = [\underline{f}(\tilde{x}_1, \dots, \tilde{x}_n, \lambda), \overline{f}(\tilde{x}_1, \dots, \tilde{x}_n, \lambda)], \lambda \in (0, 1].$$

Lemat 2.5. *Jeżeli funkcje $\underline{f}(\tilde{x}_1, \dots, \tilde{x}_n, \lambda)$, $\overline{f}(\tilde{x}_1, \dots, \tilde{x}_n, \lambda)$ są dla każdego $\lambda \in (0, 1]$ f -monotoniczne względem $\tilde{x}_i$ i dla każdego wektora $(\tilde{x}_1, \dots, \tilde{x}_n) \in FN(\mathfrak{R})^n$ istnieje wartość średnia liczby rozmytej $\tilde{F}(\tilde{x}_1, \dots, \tilde{x}_n)$, to $E(\tilde{F}(\tilde{x}_1, \dots, \tilde{x}_n))$ jest f -monotoniczną funkcją względem $\tilde{x}_i$.*

Dowód Z założenia f -monotoniczności funkcji $\underline{f}$ i $\overline{f}$ wynika, że dla każdej nieujemnej liczby rozmytej $\tilde{A}$, dla każdego wektora $(\tilde{x}_1, \dots, \tilde{x}_i, \dots, \tilde{x}_n) \in FN(\mathfrak{R})^n$ i dla każdego $\lambda \in (0, 1]$ są spełnione warunki:

$$\underline{f}(\tilde{x}_1, \dots, \tilde{x}_i \tilde{+} \tilde{A}, \dots, \tilde{x}_n, \lambda) \geq \underline{f}(\tilde{x}_1, \dots, \tilde{x}_i, \dots, \tilde{x}_n, \lambda),$$

$$\overline{f}(\tilde{x}_1, \dots, \tilde{x}_i \tilde{+} \tilde{A}, \dots, \tilde{x}_n, \lambda) \geq \overline{f}(\tilde{x}_1, \dots, \tilde{x}_i, \dots, \tilde{x}_n, \lambda).$$

Ze wzoru (1.77) otrzymujemy równości

$$E(\tilde{F}(\tilde{x}_1, \dots, \tilde{x}_i \tilde{+} \tilde{A}, \dots, \tilde{x}_n)) = \int_0^1 \left(\underline{f}(\tilde{x}_1, \dots, \tilde{x}_i \tilde{+} \tilde{A}, \dots, \tilde{x}_n, \lambda) + \overline{f}(\tilde{x}_1, \dots, \tilde{x}_i \tilde{+} \tilde{A}, \dots, \tilde{x}_n, \lambda) \right) d\lambda,$$

$$E(\tilde{F}(\tilde{x}_1, \dots, \tilde{x}_i, \dots, \tilde{x}_n)) = \int_0^1 \left(\underline{f}(\tilde{x}_1, \dots, \tilde{x}_i, \dots, \tilde{x}_n, \lambda) + \overline{f}(\tilde{x}_1, \dots, \tilde{x}_i, \dots, \tilde{x}_n, \lambda) \right) d\lambda,$$

z których wynika, że:

$$E(\tilde{F}(\tilde{x}_1, \dots, \tilde{x}_i \tilde{+} \tilde{A}, \dots, \tilde{x}_n)) \geq E(\tilde{F}(\tilde{x}_1, \dots, \tilde{x}_i, \dots, \tilde{x}_n)).$$

Zatem funkcja $E(\tilde{F}(\tilde{x}_1, \dots, \tilde{x}_n))$ jest f -monotoniczna względem argumentu $\tilde{x}_i$. $\square$

Lemat 2.6. *Jeżeli funkcja $\overline{f}(\tilde{x}_1, \dots, \tilde{x}_n, \lambda)$ jest dla każdego $\lambda \in (0, 1]$ f -monotoniczna względem argumentu $\tilde{x}_i$, to indeks $Poss(\tilde{F}(\tilde{x}_1, \dots, \tilde{x}_n) > \tilde{d})$ jest f -monotoniczną funkcją względem argumentu $\tilde{x}_i$.*

Dowód Z definicji f -monotoniczności wynika, że aby udowodnić lemat należy pokazać, że dla każdego wektora $(\tilde{x}_1, \dots, \tilde{x}_n) \in FN(\mathfrak{R})^n$ i dla każdej nieujemnej liczby rozmytej $\tilde{A}$ jest spełniony warunek:

$$Poss(\tilde{F}(\tilde{x}_1, \dots, \tilde{x}_i + \tilde{A}, \dots, \tilde{x}_n) > \tilde{d}) \geq Poss(\tilde{F}(\tilde{x}_1, \dots, \tilde{x}_n) > \tilde{d}). \quad (2.10)$$

Jeżeli $Poss(\tilde{F}(\tilde{x}_1, \dots, \tilde{x}_n) > \tilde{d}) = 0$, to warunek (2.10) jest oczywisty ponieważ indeks $Poss(\tilde{A} > \tilde{B})$ przyjmuje wartości nieujemne dla każdych liczb rozmytych $\tilde{A}, \tilde{B}$. Załóżmy teraz, że:

$$Poss(\tilde{F}(\tilde{x}_1, \dots, \tilde{x}_n) > \tilde{d}) = \lambda_0 \in (0, 1]. \quad (2.11)$$

Z warunku (2.11) oraz lematów 1.6 i 1.7 otrzymujemy:

$$\bar{f}(\tilde{x}_1, \dots, \tilde{x}_n, \lambda_0) \geq \bar{d}(1 - \lambda_0). \quad (2.12)$$

Stąd, korzystając z założenia, że $\bar{f}(\tilde{x}_1, \dots, \tilde{x}_n, \lambda)$ jest dla każdego λ f -monotoniczną funkcją względem argumentu $\tilde{x}_i$, otrzymujemy następujący warunek:

$$\bar{f}(\tilde{x}_1, \dots, \tilde{x}_i + \tilde{A}, \dots, \tilde{x}_n, \lambda_0) \geq \bar{d}(1 - \lambda_0). \quad (2.13)$$

Stosując ponownie lematy 1.6 i 1.7 z warunku (2.13) otrzymujemy nierówność:

$$Poss(\tilde{F}(\tilde{x}_1, \dots, \tilde{x}_i + \tilde{A}, \dots, \tilde{x}_n) > \tilde{d}) \geq \lambda_0. \quad (2.14)$$

Z warunków (2.11) i (2.14) wynika ostatecznie warunek (2.10), co kończy dowód. $\square$

Lemat 2.7. *Jeżeli funkcja $\bar{f}(\tilde{x}_1, \dots, \tilde{x}_n, \lambda)$ jest dla każdego $\lambda \in (0, 1]$ f -monotoniczną względem argumentu $\tilde{x}_i$, to indeks $Ness(\tilde{F}(\tilde{x}_1, \dots, \tilde{x}_n) > \tilde{d})$ jest f -monotoniczną funkcją względem argumentu $\tilde{x}_i$.*

Dowód Niech $\tilde{A}$ będzie nieujemną liczbą rozmytą. Z zależności (1.31) oraz definicji f -monotoniczności wynika, że aby udowodnić lemat wystarczy pokazać, że dla każdego wektora $(\tilde{x}_1, \dots, \tilde{x}_n) \in FN(\mathfrak{R})^n$:

$$Poss(\tilde{d} \geq \tilde{F}(\tilde{x}_1, \dots, \tilde{x}_i + \tilde{A}, \dots, \tilde{x}_n)) \leq Poss(\tilde{d} \geq \tilde{F}(\tilde{x}_1, \dots, \tilde{x}_n)). \quad (2.15)$$

Dowód warunku (2.15) jest analogiczny do dowodu warunku (2.10) w lemacie poprzednim. W dowodzie należy skorzystać z lematu 1.3. $\square$

Rozpatrzmy następujące, szczególne przypadki funkcji $\tilde{F}(\tilde{x}_1, \dots, \tilde{x}_n)$:

$$\begin{aligned} \tilde{F}_1(\tilde{x}_1, \dots, \tilde{x}_n) &= \max_{i=1, \dots, n} \{w_i(\tilde{x}_i - \tilde{d}_i)\}, \\ \tilde{F}_2(\tilde{x}_1, \dots, \tilde{x}_n) &= \max_{i=1, \dots, n} \{\max\{0, w_i(\tilde{x}_i - \tilde{d}_i)\}\}, \\ \tilde{F}_3(\tilde{x}_1, \dots, \tilde{x}_n) &= \sum_{i=1}^n w_i \tilde{x}_i, \end{aligned}$$

gdzie $\tilde{d}_i \in FN(\mathfrak{R})$ oraz $w_i \in \mathfrak{R}^+$ są stałymi dla $i = 1, \dots, n$.

Lemat 2.8. Funkcje $f_k(\tilde{x}_1, \dots, \tilde{x}_n, \lambda)$ i $\bar{f}_k(\tilde{x}_1, \dots, \tilde{x}_n, \lambda)$, $k = 1, 2, 3$, są dla każdego $\lambda \in (0, 1]$ f -monotoniczne względem każdego ze swoich argumentów.

Dowód Pokażemy, że funkcja $f_1(\tilde{x}_1, \dots, \tilde{x}_n, \lambda)$ jest f -monotoniczna względem argumentu $\tilde{x}_1$. Dla pozostałych argumentów dowód jest analogiczny.

Niech $\tilde{A}$ będzie nieujemną liczbą rozmytą a λ dowolną liczbą należącą do przedziału $(0, 1]$. Dla każdego $(\tilde{x}_1, \dots, \tilde{x}_n) \in FN(\mathfrak{R})^n$, z własności 1.4, mamy:

$$f_1(\tilde{x}_1 \tilde{+} \tilde{A}, \dots, \tilde{x}_n, \lambda) = \max\{w_1(\underline{x}_1(\lambda) + \underline{a}(\lambda) - \bar{d}_1(\lambda)), \max_{i=2, \dots, n} \{w_i(\underline{x}_i(\lambda) - \bar{d}_i(\lambda))\}\},$$

$$f_1(\tilde{x}_1, \dots, \tilde{x}_n, \lambda) = \max\{w_1(\underline{x}_1(\lambda) - \bar{d}_1(\lambda)), \max_{i=2, \dots, n} \{w_i(\underline{x}_i(\lambda) - \bar{d}_i(\lambda))\}\}.$$

Ponieważ $\underline{a}(\lambda) \geq 0$ więc $f_1(\tilde{x}_1 \tilde{+} \tilde{A}, \dots, \tilde{x}_n, \lambda) \geq f_1(\tilde{x}_1, \dots, \tilde{x}_n, \lambda)$. Zatem funkcja f_1 jest f -monotoniczna względem argumentu $\tilde{x}_1$. Dla pozostałych funkcji dowód jest analogiczny. $\square$

Wniosek 2.1. Następujące funkcje $F : FN(\mathfrak{R})^n \rightarrow \mathfrak{R}$ są f -monotonicznymi funkcjami względem każdego z argumentów $\tilde{x}_1, \dots, \tilde{x}_n$:

1. $\sum_{i=1}^n w_i E(\tilde{x}_i)$,
2. $\max_{i=1, \dots, n} \{w_i E(\tilde{x}_i)\}$,
3. $E(\max_{i=1, \dots, n} \{w_i(\tilde{x}_i \tilde{-} \tilde{d}_i)\})$,
4. $E(\max_{i=1, \dots, n} \{\max\{0, w_i(\tilde{x}_i \tilde{-} \tilde{d}_i)\}\})$,
5. $\sum_{i=1}^n U_i(\tilde{x}_i > \tilde{d}_i)$, $U_i \in \{Poss, Ness\}$,
6. $\max_{i=1, \dots, n} \{U_i(\tilde{x}_i > \tilde{d}_i)\}$, $U_i \in \{Poss, Ness\}$,
7. $Poss(\max_{i=1, \dots, n} \{w_i(\tilde{x}_i \tilde{-} \tilde{d}_i)\} > \tilde{G})$
8. $Poss(\sum_{i=1}^n w_i \tilde{x}_i > \tilde{G})$,

gdzie $\tilde{d}_i \in FN(\mathfrak{R})$, $w_i \in \mathfrak{R}^+$ oraz $\tilde{G}$ są stałymi dla $i = 1, \dots, n$.

Dowód Własność f -monotoniczności funkcji 1-8 względem każdego z argumentów $\tilde{x}_1, \dots, \tilde{x}_n$ wynika bezpośrednio z lematów 2.1, 2.2, 2.5, 2.6, 2.7 i 2.8. $\square$

Funkcje 1-8, określone we wniosku 2.1, zostaną wykorzystane w kolejnym rozdziale do skonstruowania różnych, f -regularnych funkcji kosztu harmonogramu.

2.4 Konstrukcja różnych funkcji kosztu harmonogramu w rozmytym problemie szeregowania

W klasycznych problemach szeregowania definiuje się nieterminowość oraz opóźnienie prac $i \in J$ w danym harmonogramie p (patrz punkt 2.1). Obecnie uogólnimy te pojęcia na przypadek rozmyty.

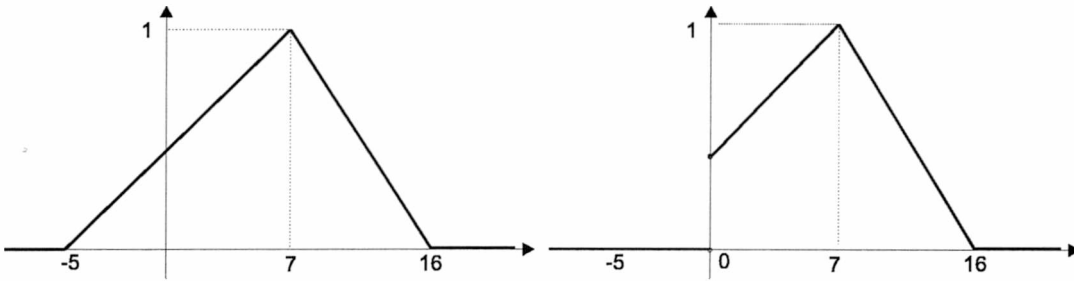
Definicja 2.7. *Rozmytą nieterminowością pracy $i \in J$ w harmonogramie p nazywamy liczbę rozmytą $\tilde{L}_i(p)$ określoną następująco:*

$$\tilde{L}_i(p) = \tilde{C}_i(p) \tilde{-} \tilde{d}_i. \quad (2.16)$$

Definicja 2.8. *Rozmytym opóźnieniem pracy $i \in J$ w harmonogramie p nazywamy liczbę rozmytą $\tilde{T}_i(p)$ określoną następująco:*

$$\tilde{T}_i(p) = \tilde{max}\{0, \tilde{C}_i(p) \tilde{-} \tilde{d}_i\}. \quad (2.17)$$

Funkcje przynależności rozmytej nieterminowości i rozmytego opóźnienia określają rozkłady możliwości dla nieterminowości lub opóźnienia pracy i w harmonogramie p . Załóżmy, na przykład, że w pewnym harmonogramie p rozmyty czas zakończenia i rozmyty pożądaný termin zakończenia pracy i są określone za pomocą trójkątnych liczb rozmytych, odpowiednio $\tilde{C}_i(p) = (15, 6, 7)$ i $\tilde{d}_i = (8, 2, 6)$. Funkcje przynależności $\tilde{L}_i(p)$, $\tilde{T}_i(p)$ są przedstawione na rysunku 2.1.



Rys. 2.1: Funkcje przynależności $\tilde{L}_i(p)$, $\tilde{T}_i(p)$

Korzystając z pojęcia wartości średniej liczby rozmytej można skonstruować następujące funkcje kosztu harmonogramu:

$$F(p) = \max_{i=1, \dots, n} \{w_i E(\tilde{L}_i(p))\}, \quad (2.18)$$

$$F(p) = \max_{i=1, \dots, n} \{w_i E(\tilde{T}_i(p))\}, \quad (2.19)$$

$$F(p) = E \left(\max_{i=1, \dots, n} \{w_i \tilde{L}_i(p)\} \right), \quad (2.20)$$

$$F(p) = E \left(\max_{i=1, \dots, n} \{w_i \tilde{T}_i(p)\} \right), \quad (2.21)$$

$$F(p) = E \left(\sum_{i=1}^n w_i \tilde{C}_i(p) \right) = \sum_{i=1}^n w_i E(\tilde{C}_i(p)), \quad (2.22)$$

$$F(p) = E \left(\sum_{i=1}^n w_i \tilde{L}_i(p) \right) = \sum_{i=1}^n w_i E(\tilde{L}_i(p)), \quad (2.23)$$

$$F(p) = E \left(\sum_{i=1}^n w_i \tilde{T}_i(p) \right) = \sum_{i=1}^n w_i E(\tilde{T}_i(p)). \quad (2.24)$$

W podejściu probabilistycznym (w którym parametry zadane są jako zmienne losowe) analogiczne funkcje kosztu otrzymalibyśmy, zastępując wartość średnią liczby rozmytej wartością oczekiwaną zmiennej losowej.

Wybór konkretnej funkcji kosztu zależy od decydenta. Jeżeli zależy mu na minimalizacji największego opóźnienia (nieterminowości), to powinien wybrać jedną z funkcji (2.18)-(2.21). Zauważmy, że funkcje (2.18) i (2.20), a także (2.19) i (2.21) nie są równoważne. Wynika to z faktu, że wartość średnia nie zachowuje operacji maksimum. Funkcje (2.20) i (2.21) wydają się być lepsze od funkcji (2.18) i (2.19) z tego powodu, że w pierwszym przypadku wartość średnia jest obliczana tylko jeden raz, natomiast w drugim przypadku wartość średnia obliczana jest wielokrotnie. Każde obliczenie wartości średniej powoduje utratę pewnej informacji o rozkładzie możliwości parametrów problemu.

Jeżeli decydentowi zależy na minimalizacji wszystkich czasów zakończenia, opóźnień lub nieterminowości, to powinien wybrać funkcję odpowiednio (2.22), (2.23) lub (2.24).

Zauważmy, że z wniosku 2.1 oraz z definicji rozmytego opóźnienia i rozmytej nieterminowości wynika, że wszystkie funkcje (2.18)-(2.24) są f -regularne. Z twierdzenia 2.1 wynika, że w rozmytych problemach szeregowania z funkcjami kosztu postaci (2.18)-(2.24) wystarczy rozpatrywać wyłącznie harmonogramy bez podziału prac i przestojów maszyny (czyli permutacje zbioru J).

Do konstrukcji funkcji kosztu harmonogramu można wykorzystać również in-

deksy Dubois i Prade, zdefiniowane w punkcie 1.5. Załóżmy, że decydentowi zależy na tym, żeby możliwość (lub konieczność) przekroczenia pożądanego terminu zakończenia przez każdą z prac była jak najmniejsza. Wówczas funkcja kosztu harmonogramu może mieć jedną z następujących postaci:

$$F(p) = \max_{i=1, \dots, n} \{Poss(Ness)(\tilde{C}_i(p) > \tilde{d}_i)\}, \quad (2.25)$$

$$F(p) = \sum_{i=1}^n Poss(Ness)(\tilde{C}_i(p) > \tilde{d}_i). \quad (2.26)$$

W podejściu probabilistycznym odpowiednie funkcje kosztu otrzymalibyśmy zastępując indeksy $Poss$ i $Ness$ prawdopodobieństwem zdarzenia, że czas zakończenia pracy i przekroczy pożądaną termin zakończenia tej pracy.

Wybór konkretnego indeksu ($Poss$ lub $Ness$) zależy od tego jak silnie decydentowi zależy na szybkim zakończeniu danej pracy. Jeżeli mu na tym bardzo zależy, to powinien wybrać indeks $Poss(\tilde{A} > \tilde{B})$ ponieważ

$$Poss(\tilde{A} > \tilde{B}) \geq Ness(\tilde{A} > \tilde{B}).$$

Korzystając z indeksów Dubois i Prade można konstruować funkcje kosztu, w których minimalizuje się możliwość zajścia bardziej złożonych zdarzeń. Rozpatrzmy następujące funkcje kosztu:

$$F(p) = Poss\left(\sum_{i=1}^n w_i \tilde{C}_i(p) > \tilde{G}\right), \quad (2.27)$$

$$F(p) = Poss\left(\max_{i=1, \dots, n} \{w_i \tilde{L}_i\} > \tilde{G}\right). \quad (2.28)$$

Liczba rozmyta $\tilde{G}$ jest dodatkowym parametrem podawanym przez decydenta, który będziemy nazywać rozmytym celem. W przypadku funkcji (2.27) decydent chce, żeby ważona suma czasów zakończenia prac nie przekraczała pewnej liczby $\tilde{G}$. Jeżeli decydent potrafi dokładnie określić swój cel, to liczba $\tilde{G}$ jest zdeterminowana. Możliwa jest również sytuacja, w której decydent nie potrafi dokładnie sprecyzować wartości celu (np.: decydent chce, żeby ważona suma czasów zakończenia prac była niewiększa niż około 10). W takim przypadku liczba $\tilde{G}$ będzie rozmyta.

Z wniosku 2.1 wynika, że wszystkie funkcje (2.25)-(2.28) są f -regularne. Zatem z twierdzenia 2.1 wynika, że w rozmytych problemach szeregowania z funkcjami kosztu postaci (2.25) - (2.28) wystarczy rozpatrywać wyłącznie harmonogramy bez podziału prac i przestojów maszyny (czyli permutacje zbioru J).

We wszystkich podanych do tej pory funkcjach kosztu zakładamy, że decydentowi zależy na możliwie szybkim zakończeniu każdej pracy. W praktycznych zastosowaniach może się jednak zdarzyć, że zbyt szybkie zakończenie prac również nie jest pożądane. Ma to miejsce, na przykład, w systemach produkcji *Just In Time*, w których zadania powinny kończyć się w dokładnie wyznaczonym terminie. W takim przypadku można wykorzystać indeks $Ness(\tilde{A} \neq \tilde{B})$. Skonstruujmy następującą funkcję kosztu harmonogramu:

$$F(p) = \max_{i=1, \dots, n} \{Ness(\tilde{C}_i(p) \neq \tilde{d}_i)\}. \quad (2.29)$$

Bardzo istotną cechą funkcji (2.29) jest to, że nie jest ona f -regularna. Wynika to z faktu, że funkcja $Ness(\tilde{C}_i(p) \neq \tilde{d}_i)$ nie jest f -monotoniczną funkcją $\tilde{C}_i(p)$, zatem w przypadku funkcji (2.29) dopuszczenie podziału prac i przestojów maszyny może prowadzić do uzyskania lepszych harmonogramów.

W kolejnych punktach tego rozdziału zbadamy rozmaite problemy szeregowania z funkcjami kosztu postaci (2.18) - (2.29), kładąc szczególny nacisk na analizę ich złożoności obliczeniowej. Dla każdego problemu, który ma złożoność wielomianową podamy odpowiedni algorytm a dla każdego problemu NP-trudnego przedstawimy dowód NP-trudności.

2.5 Problem $1|prec, FN|F_{max}$ z f -monotonicznymi funkcjami kosztów zakończenia prac

W punkcie tym najpierw przeanalizujemy najbardziej ogólną postać problemu $1|prec, FN|F_{max}$, w której relacja poprzedzania $prec$ jest dowolna i funkcja kosztu harmonogramu jest postaci:

$$F_{max}(p) = \max_{i=1, \dots, n} \{f_i(\tilde{C}_i(p))\}. \quad (2.30)$$

Pokażemy, że przy założeniu f -monotoniczności wszystkich funkcji f_i , problem ten można rozwiązać za pomocą uogólnionej wersji znanego algorytmu Lawlera. Następnie pokażemy w jaki sposób można za pomocą tego algorytmu efektywnie rozwiązać problemy z funkcjami kosztu postaci (2.18), (2.19) i (2.25).

2.5.1 Konstrukcja uogólnionego algorytmu Lawlera

Rozważmy klasyczny problem szeregowania na pojedynczej maszynie $1|prec|f_{max}$, w którym czasy trwania prac t_i dla $i \in J$ są zdeterminowane a funkcja kosztu harmonogramu jest postaci:

$$f_{max}(p) = \max_{i=1, \dots, n} \{f_i(C_i(p))\}.$$

Jeżeli założymy, że wszystkie funkcje $f_i(t)$ dla $i \in J$ są niemalejące, to optymalny harmonogram może zostać skonstruowany za pomocą algorytmu Lawlera ([3],[23]) przedstawionego w postaci algorytmu 3.

Algorytm 3 Algorytm Lawlera dla problemu $1|prec|f_{max}$

Require: $n, (t_i)_{i=1}^n, (f_i)_{i=1}^n, prec$

Ensure: p /* optymalny harmonogram (permutacja zbioru J) */

```

1:  $S \leftarrow \{1, \dots, n\}$  /* pomocniczy zbiór */
2:  $p \leftarrow ()$ 
3: while  $S \neq \emptyset$  do
4:    $T \leftarrow \sum_{i \in S} t_i$ 
5:    $S' \leftarrow S \setminus \{i \in S \mid \exists k \in S (i, k) \in prec\}$ 
6:   Znajdź  $i \in S'$  takie, że  $f_i(T)$  jest najmniejsze
7:    $p \leftarrow i \circ p$  /* dołącz  $i$  na początek  $p$  */
8:    $S \leftarrow S \setminus \{i\}$ 
9: end while
10: return  $p$ 
```

Założmy teraz, że czasy trwania prac są liczbami rozmytymi a funkcja kosztu harmonogramu jest postaci:

$$F_{max}(p) = \max_{i=1, \dots, n} \{f_i(\tilde{C}_i(p))\},$$

gdzie $f_i : FN(\mathbb{R}) \rightarrow \mathbb{R}$, $i = 1, \dots, n$. Zakładamy, że dla każdego $i \in J$ funkcja f_i jest f -monotoniczna względem czasu zakończenia pracy $i \in J$. Zauważmy, że przy takim założeniu funkcja F_{max} jest f -regularna (patrz lemat 2.2). Z twierdzenia 2.1 wynika więc, że wystarczy rozpatrywać wyłącznie harmonogramy bez podziału prac i przestojów maszyny. Zatem analizowany problem sprowadza się do wyznaczenia takiej dopuszczalnej permutacji $p = (p(1), \dots, p(n))$ zbioru J dla której funkcja $F_{max}(p)$ osiąga minimum.

Algorytm 4, przeznaczony do rozwiązywania problemu $1|prec, FN|F_{max}$ jest uogólnioną wersją algorytmu Lawlera. Jedyną, zasadniczą różnicą między klasycznym algorytmem Lawlera a algorytmem 4 występuje w wierszu 4, w którym

obliczamy sumę liczb rozmytych, zdefiniowaną za pomocą zasady rozszerzania Zadeha (patrz definicja 1.10).

Algorytm 4 Uogólniony algorytm Lawlera dla problemu $1|prec, FN|F_{max}$

Require: $n, (\tilde{t}_i)_{i=1}^n, (f_i)_{i=1}^n, prec$

Ensure: p /* optymalny harmonogram (permutacja zbioru J) */

```

1:  $S \leftarrow \{1, \dots, n\}$  /* pomocniczy zbiór */
2:  $p \leftarrow ()$ 
3: while  $S \neq \emptyset$  do
4:    $\tilde{T} \leftarrow \sum_{i \in S} \tilde{t}_i$ 
5:    $S' \leftarrow S \setminus \{i \in S \mid \exists k \in S (i, k) \in prec\}$ 
6:   Znajdź  $i \in S'$  takie, że  $f_i(\tilde{T})$  jest najmniejsze
7:    $p \leftarrow i \circ p$  /* dołącz  $i$  na początek  $p$  */
8:    $S \leftarrow S \setminus \{i\}$ 
9: end while
10: return  $p$ 
```

Udowodnimy teraz twierdzenie, które stanowi teoretyczne uzasadnienie poprawności algorytmu 4.

Twierdzenie 2.2. *Jeżeli dla każdej pracy $i \in J$ funkcja $f_i(\tilde{C}_i(p))$ jest f -monotoniczna względem $\tilde{C}_i(p)$, to algorytm 4 konstruuje optymalną permutację prac.*

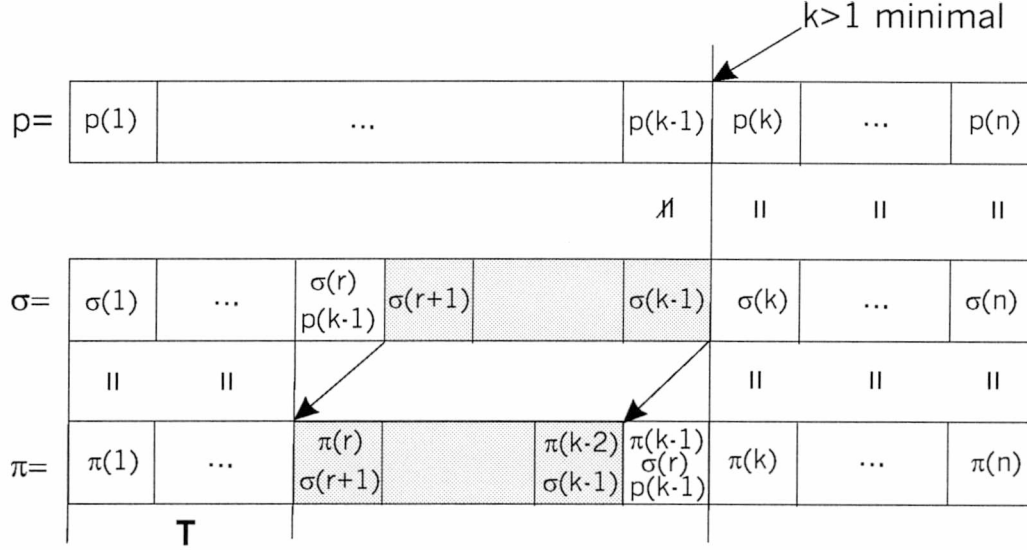
Dowód Oznaczmy przez $p = (p(1), \dots, p(n))$ permutację skonstruowaną przez algorytm. Załóżmy nie wprost, że permutacja p nie jest optymalna. Wybierzmy spośród wszystkich rozwiązań optymalnych permutację $\sigma = (\sigma(1), \dots, \sigma(n))$ taką, że $\sigma(i) = p(i)$ dla $i = n, n-1, \dots, k$ oraz $\sigma(k-1) \neq p(k-1)$, przy czym k jest minimalne. Z założenia nie wprost wynika, że $k > 1$ (w przeciwnym wypadku $\sigma = p$ i p byłoby optymalną permutacją). Przypuśćmy, że $\sigma(r) = p(k-1)$, gdzie $1 \leq r < k-1$. Utwórzmy nową permutację π przez przesunięcie pracy $\sigma(r)$ bezpośrednio przed pracą $\sigma(k)$ w permutacji σ . Relacje między permutacjami p , σ i π są zaprezentowane na rysunku 2.2.

Zauważmy, że permutacja π jest dopuszczalna ponieważ $\sigma(r) = p(k-1)$ nie ma następników w zbiorze prac $\{\sigma(1), \dots, \sigma(k-1)\}$ (tj. $(\sigma(r), \sigma(i)) \notin prec$ dla $i = 1, \dots, k-1$). Pokażemy, że permutacja π jest również optymalna. Jest oczywiste, że dla $i = 1, \dots, r-1$ oraz dla $i = k, \dots, n$:

$$f_{\sigma(i)}(\tilde{C}_{\sigma(i)}(\sigma)) = f_{\pi(i)}(\tilde{C}_{\pi(i)}(\pi)). \quad (2.31)$$

Mamy również:

$$f_{\sigma(k-1)}(\tilde{C}_{\sigma(k-1)}(\sigma)) \geq f_{\pi(k-1)}(\tilde{C}_{\pi(k-1)}(\pi)), \quad (2.32)$$



Rys. 2.2: Relacje między permutacjami p, σ, π

ponieważ $\pi(k-1) = p(k-1)$ i permutacja p jest skonstruowana przez algorytm (patrz krok 6 algorytmu). Dla każdego $i = r, \dots, k-2$, z f -monotoniczności funkcji f_i oraz z faktu, że $\tilde{t}_{\sigma(r)}$ jest nieujemną liczbą rozmytą wynika, że:

$$\begin{aligned} f_{\sigma(i+1)}(\tilde{C}_{\sigma(i+1)}(\sigma)) &= f_{\sigma(i+1)}(\tilde{T} + \tilde{t}_{\sigma(r)} + \sum_{j=r}^i \tilde{t}_{\pi(j)}) \geq \\ &\geq f_{\sigma(i+1)}(\tilde{T} + \sum_{j=r}^i \tilde{t}_{\pi(j)}) = f_{\pi(i)}(\tilde{C}_{\pi(i)}(\pi)), \end{aligned} \quad (2.33)$$

gdzie $\tilde{T} = \sum_{i=1}^{r-1} \tilde{t}_i$.

Z równości (2.31), (2.32) i (2.33) wynika, że koszt permutacji π jest nie większy niż koszt permutacji σ . Oznacza to, że permutacja π jest również optymalna co przeczy warunkowi minimalności k . $\square$

Złożoność obliczeniowa algorytmu 4 zależy od tego, czy potrafimy efektywnie obliczyć sumę rozmytych liczb w kroku 4 i wartość funkcji kosztu f_i dla każdego $i \in J$ w kroku 6. Jeżeli wymienione kroki dadzą się zrealizować w czasie zależnym wielomianowo od liczby prac, to złożoność całego algorytmu będzie wielomianowa. W kolejnych punktach zbadamy szczególne przypadki problemu $1|prec, FN|F_{max}$, które mogą być rozwiązane efektywnie.

2.5.2 Problem $1|prec, L - R|max\{Poss(Ness)(\tilde{C}_i > \tilde{d}_i)\}$

Założmy, że czasy trwania i pożądane terminy zakończenia prac są liczbami rozmytymi typu $L - R$, określonymi następująco:

$$\tilde{t}_i = (\underline{t}_i, \bar{t}_i, \alpha_i, \beta_i)_{L-R}, \quad i = 1, \dots, n,$$

$$\tilde{d}_i = (\underline{d}_i, \bar{d}_i, \phi_i, \psi_i)_{S_i-T_i}, \quad i = 1, \dots, n.$$

Rozpatrzmy funkcję kosztu harmonogramu, która ma następującą postać:

$$F_{max}(p) = \max_{i=1, \dots, n} \{D_i(\tilde{C}_i(p) > \tilde{d}_i)\},$$

gdzie $D_i \in \{Poss, Ness\}$ dla $i \in J$.

Z wniosku 2.1 wynika, że funkcje $Poss(\tilde{C}_i(p) > \tilde{d}_i)$ i $Ness(\tilde{C}_i(p) > \tilde{d}_i)$ są f -monotoniczne względem $\tilde{C}_i(p)$. Do rozwiązania problemu z tak zdefiniowaną funkcją kosztu możemy więc wykorzystać uogólniony algorytm Lawlera. Odpowiednio modyfikując algorytm 4 otrzymujemy algorytm 5, który konstruuje optymalną permutację prac przy założeniu, że indeksy D_i , $i = 1, \dots, n$, są liczone zadaną dokładnością $\varepsilon > 0$.

Algorytm 5 Algorytm dla problemu $1|prec, L - R|max\{Poss(Ness)(\tilde{C}_i > \tilde{d}_i)\}$

Require: $n, (\tilde{t}_i)_{i=1}^n, (\tilde{d}_i)_{i=1}^n, prec, (D_i)_{i=1}^n, \varepsilon \in (0, 1]$

Ensure: p /* optymalny harmonogram (permutacja zbioru J) */

```

1:  $S \leftarrow \{1, \dots, n\}$  /* pomocniczy zbiór */
2:  $p \leftarrow ()$ 
3: while  $S \neq \emptyset$  do
4:    $\tilde{T} \leftarrow (\sum_{i \in S} \underline{t}_i, \sum_{i \in S} \bar{t}_i, \sum_{i \in S} \alpha_i, \sum_{i \in S} \beta_i)_{L-R}$ 
5:    $S' \leftarrow S \setminus \{i \in S \mid \exists k \in S(i, k) \in prec\}$ 
6:   for  $i \in S'$  do
7:     Oblicz  $D_i(\tilde{T} > \tilde{d}_i)$  z dokładnością  $\varepsilon$  korzystając z algorytmów 1 lub 2
8:   end for
9:   Znajdź  $i \in S'$  takie, że  $D_i(\tilde{T} > \tilde{d}_i)$  jest najmniejsze
10:   $p \leftarrow i \circ p$  /* dołącz  $i$  na początek  $p$  */
11:   $S \leftarrow S \setminus \{i\}$ 
12: end while
13: return  $p$ 
```

W kroku 4 korzystamy z własności dodawania liczb rozmytych typu $L - R$ (patrz własność 1.5) a w kroku 7 korzystamy z algorytmów obliczających indeksy Dubois i Prade zadaną dokładnością ε . Łatwo pokazać, że złożoność podanego algorytmu jest rzędu $O(n^2 |\log \varepsilon|)$.

Algorytm 5 upraszcza się, jeżeli wszystkie rozmyte parametry są liczbami trapezowymi. Wówczas w kroku 7 można bezpośrednio obliczyć odpowiednie indeksy Dubois i Prade, korzystając z lematu 1.11. W takim przypadku złożoność algorytmu 5 jest rzędu $O(n^2)$.

Przykład: Niech $J = \{1, 2, 3, 4, 5, 6\}$, $prec = \{(1, 3), (2, 5)\}$ a $\tilde{t}_i, \tilde{d}_i$ będą trapezowymi liczbami rozmytymi. Parametry oraz funkcje kosztu dla każdej pracy $i \in J$ przedstawione są w tablicy 2.3.

$i \in J$	$\tilde{t}_i$	$\tilde{d}_i$	$f_i(\tilde{C}_i(p))$
1	(1,2,1,1)	(1,6,1,3)	$Ness(\tilde{C}_1(p) > \tilde{d}_1)$
2	(2,4,1,2)	(1,10,1,5)	$Poss(\tilde{C}_2(p) > \tilde{d}_2)$
3	(2,2,1,3)	(1,9,1,4)	$Ness(\tilde{C}_3(p) > \tilde{d}_3)$
4	(3,4,1,3)	(1,6,1,6)	$Ness(\tilde{C}_4(p) > \tilde{d}_4)$
5	(1,3,1,2)	(1,13,1,3)	$Poss(\tilde{C}_5(p) > \tilde{d}_5)$
6	(1,1,1,1)	(1,7,1,6)	$Poss(\tilde{C}_6(p) > \tilde{d}_6)$

Tab. 2.3: Parametry przykładowego problemu

Poszczególne kroki algorytmu są przedstawione w tablicy 2.4. W każdym kroku symbolem $\times$ są oznaczone te prace, które nie należą do zbioru S' . Prace o najmniejszym koszcie są podkreślone.

krok	$\tilde{T}$	$f_1(\tilde{T})$	$f_2(\tilde{T})$	$f_3(\tilde{T})$	$f_4(\tilde{T})$	$f_5(\tilde{T})$	$f_6(\tilde{T})$	p^*
1	(10,16,6,12)	$\times$	$\times$	<u>0.1</u>	0.33	1	1	(3)
2	(8,14,5,9)	0.25	$\times$	$\times$	<u>0.18</u>	0.83	1	(4,3)
3	(5,10,4,6)	<u>0</u>	$\times$	$\times$	$\times$	0.33	0.75	(1,4,3)
4	(4,8,3,5)	$\times$	$\times$	$\times$	$\times$	<u>0</u>	0.54	(5,1,4,3)
5	(3,5,2,3)	$\times$	<u>0</u>	$\times$	$\times$	$\times$	0.11	(2,5,1,4,3)
6	(1,1,1,1)	$\times$	$\times$	$\times$	$\times$	$\times$	<u>0</u>	(6,2,5,1,4,3)

Tab. 2.4: Poszczególne kroki algorytmu

Optymalnym rozwiązaniem jest permutacja $p^* = (6, 2, 5, 1, 4, 3)$ o koszcie 0.18. Zauważmy, że pracą o największym koszcie (równym 0.18) jest praca 4. Oznacza to, że możliwość (lub konieczność) przekroczenia pożądanego terminu zakończenia przez każdą pracę w harmonogramie p jest nie większa niż 0.18. $\square$

2.5.3 Problem $1|prec, L - R| \max\{E(w_i \tilde{L}_i)\}$

Załóżmy, że $\tilde{t}_i, \tilde{d}_i$, $i = 1, \dots, n$, są dowolnymi liczbami rozmytymi typu $L - R$, dla których istnieje wartość średnia (patrz lemat 1.12). Rozpatrzmy następującą

funkcję kosztu harmonogramu:

$$F_{max}(p) = \max_{i=1, \dots, n} \{E(w_i \tilde{L}_i(p))\}.$$

Z wniosku 2.1 wynika, że funkcja $E(w_i \tilde{L}_i(p))$ jest f -monotoniczna względem $\tilde{C}_i(p)$ dla każdego $i \in J$. Możemy więc zastosować do badanego problemu uogólniony algorytm Lawlera. Zauważmy ponadto, że dla każdej pracy $i \in J$ zachodzi równość (patrz własność 1.7):

$$E(w_i \tilde{L}_i(p)) = E(w_i(\tilde{C}_i(p) - \tilde{d}_i)) = w_i(E(\tilde{C}_i(p)) - E(\tilde{d}_i)) = w_i\left(\sum_{j \in p^i} E(\tilde{t}_j) - E(\tilde{d}_i)\right),$$

która pozwala dodatkowo uprościć algorytm 4. W efekcie otrzymujemy algorytm 6, który jest w istocie klasycznym algorytmem Lawlera dla problemu, w którym czasy trwania i pożądane terminy zakończenia prac są zastąpione ich wartościami średnimi.

Algorytm 6 Algorytm dla problemu $1|prec, L - R|max\{E(w_i \tilde{L}_i)\}$

Require: $n, (\tilde{t}_i)_{i=1}^n, (\tilde{d}_i)_{i=1}^n, (w_i)_{i=1}^n, prec$

Ensure: p /* optymalny harmonogram (permutacja zbioru J) */

```

1:  $S \leftarrow \{1, \dots, n\}$  /* pomocniczy zbiór */
2:  $p \leftarrow ()$ 
3: for all  $i \in \{1, \dots, n\}$  do
4:    $t_i \leftarrow E(\tilde{t}_i)$  /* korzystając z lematu 1.12 */
5:    $d_i \leftarrow E(\tilde{d}_i)$  /* korzystając z lematu 1.12 */
6: end for
7: while  $S \neq \emptyset$  do
8:    $T \leftarrow \sum_{i \in S} t_i$ 
9:    $S' \leftarrow S \setminus \{i \in S \mid \exists k \in S(i, k) \in prec\}$ 
10:  Znajdź  $i \in S'$  takie, że  $w_i(T - d_i)$  jest najmniejsze
11:   $p \leftarrow i \circ p$  /* dołącz  $i$  na początek  $p$  */
12:   $S \leftarrow S \setminus \{i\}$ 
13: end while
14: return  $p$ 
```

Zauważmy, że odpowiednie wartości średnie w krokach 4 i 5 algorytmu można efektywnie obliczyć za pomocą formuły z lematu 1.12. W kolejnych krokach algorytmu są wykonywane wyłącznie proste operacje na liczbach rzeczywistych.

Złożoność algorytmu 6 jest taka sama jak złożoność klasycznego algorytmu Lawlera i jest rzędu $O(n^2)$.

Rozpatrzmy teraz problem $1|L-R|\max\{E(\tilde{L}_i)\}$, który jest szczególnym przypadkiem problemu $1|prec, L-R|E(w_i\tilde{L}_i)$ z wagami $w_i = 1$ dla każdej pracy $i \in J$ oraz $prec = \emptyset$.

Twierdzenie 2.3. *Jeżeli $\sigma = (\sigma(1), \dots, \sigma(n))$ jest permutacją taką, że*

$$E(\tilde{d}_{\sigma(1)}(\sigma)) \leq E(\tilde{d}_{\sigma(2)}(\sigma)) \leq \dots \leq E(\tilde{d}_{\sigma(n)}(\sigma)), \quad (2.34)$$

to σ jest optymalną permutacją dla problemu $1|L-R|\max\{E(\tilde{L}_i)\}$.

Dowód Dowód wynika bezpośrednio z algorytmu 6. Jeżeli $prec = \emptyset$ i $w_i = 1$ dla każdej pracy i , to w kroku 9 mamy zawsze $S = S'$ i kroki 8-10 algorytmu 6 są równoważne następującej instrukcji:

8: Znajdź $i \in S$ takie, że $d_i = E(\tilde{d}_i)$ jest największe.

Stąd otrzymujemy regułę (2.34). □

Złożoność algorytmu dla problemu $1|L-R|\max\{E(\tilde{L}_i)\}$ jest taka sama jak złożoność algorytmu sortowania i jest rzędu $O(n \log n)$.

2.5.4 Problem $1|prec, PL-PR|\max\{E(w_i\tilde{T}_i)\}$

W punkcie tym zbadamy problem z funkcją kosztu harmonogramu określoną następująco:

$$F_{max}(p) = \max_{i=1, \dots, n} \{E(w_i\tilde{T}_i(p))\}.$$

Jeżeli wszystkie parametry są zdeterminowane, to optymalne rozwiązanie problemu $1|prec, FN|\max\{E(w_i\tilde{L}_i)\}$, rozpatrywanego w poprzednim punkcie, jest jednocześnie optymalnym rozwiązaniem problemu $1|prec, FN|\max\{E(w_i\tilde{T}_i)\}$, wystarczy więc jedynie skonstruować algorytm dla pierwszego z tych problemów³.

Powyższy fakt nie jest prawdziwy w przypadku gdy parametry mogą być rozmyte. Pokażemy to na następującym prostym przykładzie, w którym parametry są nieujemnymi, trójkątnymi liczbami rozmytymi.

Przykład: Załóżmy, że $J = \{1, 2\}$, a parametry problemu są równe:

$$\tilde{t}_1 = (2, 1, 1), \tilde{t}_2 = (2, 1, 1),$$

³W takim przypadku problem $1|prec, FN|\max\{E(w_i\tilde{L}_i)\}$ jest równoważny klasycznemu problemowi $1|prec|\max\{w_iL_i\}$ a problem $1|prec, FN|\max\{E(w_i\tilde{T}_i)\}$ jest równoważny klasycznemu problemowi $1|prec|\max\{w_iT_i\}$. Łatwo pokazać, korzystając z określenia funkcji $T_i(p)$ oraz $L_i(p)$, że jeżeli p jest optymalnym rozwiązaniem problemu $1|prec|\max\{w_iL_i\}$, to p jest również optymalnym rozwiązaniem problemu $1|prec|\max\{w_iT_i\}$ z tymi samymi parametrami. Implikacja odwrotna jednak nie zachodzi.

$$\tilde{d}_1 = (4, 1, 1), \tilde{d}_2 = (3, 1, 7),$$

$$w_1 = w_2 = 1.$$

Zbiór rozwiązań dopuszczalnych tego problemu składa się z dwóch permutacji: $p_1 = (1, 2)$ i $p_2 = (2, 1)$. Wartości średnie rozmytych opóźnień i nieterminowości dla obu permutacji i obu prac są podane w tablicy 2.5.

p_i	$\tilde{L}_1(p_i)$	$\tilde{L}_2(p_i)$	$E(\tilde{L}_1(p_i))$	$E(\tilde{L}_2(p_i))$	$E(\tilde{T}_1(p_i))$	$E(\tilde{T}_2(p_i))$
p_1	(-2,2,2)	(1,9,3)	-2	-0.5	0	1.333
p_2	(0,3,3)	(-1,8,2)	0	-2.5	0.75	0.125

Tab. 2.5: Wartości średnie rozmytych nieterminowości i rozmytych opóźnień prac w przykładowym problemie

Z danych przedstawionych w tablicy 2.5 otrzymujemy, że:

$$\max\{E(\tilde{L}_1(p_1), E(\tilde{L}_2(p_1))\} < \max\{E(\tilde{L}_1(p_2), E(\tilde{L}_2(p_2))\}, \quad (2.35)$$

$$\max\{E(\tilde{T}_1(p_1), E(\tilde{T}_2(p_1))\} > \max\{E(\tilde{T}_1(p_2), E(\tilde{T}_2(p_2))\}. \quad (2.36)$$

Z warunków (2.35) i (2.36) wynika, że optymalna permutacja dla problemu $1|prec, FN| \max\{E(w_i \tilde{L}_i)\}$ nie musi być jednocześnie optymalną permutacją dla problemu $1|prec, FN| \max\{E(w_i \tilde{T}_i)\}$ z takimi samymi parametrami. Problemy te są więc istotnie różne. $\square$

Zauważmy, że obliczenie $E(w_i \tilde{T}_i(p))$ jest znacznie trudniejsze niż obliczenie $E(w_i \tilde{L}_i(p))$. Załóżmy, że wszystkie parametry problemu są liczbami rozmytymi typu $L - R$ z potęgowymi funkcjami kształtu, określonymi następująco:

$$\tilde{t}_i = (\underline{t}_i, \bar{t}_i, \alpha_i, \beta_i)_{L-R}, \quad i = 1, \dots, n,$$

$$\tilde{d}_i = (\underline{d}_i, \bar{d}_i, \phi_i, \psi_i)_{R-L}, \quad i = 1, \dots, n,$$

gdzie:

$$L(x) = \max\{0, 1 - x^{p_1}\}, R(x) = \max\{0, 1 - x^{p_2}\}.$$

Z wniosku 2.1 wynika, że dla każdego $i \in J$ funkcja $E(w_i \tilde{T}_i(p))$ jest f -monotoniczną funkcją $\tilde{C}_i(p)$. Dlatego do rozwiązania problemu możemy zastosować uogólniony algorytm Lawlera (patrz algorytm 4), który w tym przypadku sprowadza się do algorytmu 7.

W kroku 8 Algorytmu 7 korzystamy z formuły podanej w lemacie 1.13, która pozwala na bezpośrednie obliczenie odpowiedniej wartości średniej. Złożoność obliczeniowa algorytmu 7 jest rzędu $O(n^2)$.

Algorytm 7 Algorytm dla problemu $1|prec, PL - PR|max\{E(w_i\tilde{T}_i)\}$

Require: $n, (\tilde{t}_i)_{i=1}^n, (\tilde{d}_i)_{i=1}^n, (w_i)_{i=1}^n, prec$

Ensure: p /* optymalny harmonogram (permutacja zbioru J) */

```

1:  $S \leftarrow \{1, \dots, n\}$  /* pomocniczy zbiór */
2:  $p \leftarrow ()$ 
3: while  $S \neq \emptyset$  do
4:    $\tilde{T} \leftarrow (\sum_{i \in S} \underline{t}_i, \sum_{i \in S} \bar{t}_i, \sum_{i \in S} \alpha_i, \sum_{i \in S} \beta_i)_{L-R}$ 
5:    $S' \leftarrow S \setminus \{i \in S \mid \exists_{k \in S} (i, k) \in prec\}$ 
6:   for all  $i \in S'$  do
7:      $\tilde{D}_i \leftarrow (\underline{T} - \underline{d}_i, \bar{T} - \underline{d}_i, \alpha_T + \psi_i, \beta_T + \phi_i)_{L-R}$ 
8:      $T_i \leftarrow w_i E(max\{0, \tilde{D}_i\})$  /* Korzystając z lematu 1.13 */
9:   end for
10:  Znajdź  $i \in S'$  takie, że  $T_i$  jest najmniejsze
11:   $p \leftarrow i \circ p$  /* dołącz  $i$  na początek  $p$  */
12:   $S \leftarrow S \setminus \{i\}$ 
13: end while
14: return  $p$ 

```

Przykład: Niech $J = \{1, 2, 3, 4, 5, 6\}$, $prec = \{(2, 5), (3, 6)\}$ a $\tilde{t}_i, \tilde{d}_i$ będą trapezowymi liczbami rozmytymi. Wszystkie parametry problemu są podane w tablicy 2.6.

$i \in J$	$\tilde{t}_i$	$\tilde{d}_i$	w_i
1	(1,2,1,1)	(12,12,3,3)	1
2	(2,4,1,2)	(10,11,3,5)	3
3	(2,2,1,3)	(8,9,2,4)	2
4	(3,4,1,3)	(12,15,6,6)	2
5	(1,3,1,2)	(8,9,3,3)	3
6	(1,1,1,1)	(10,13,5,6)	2

Tab. 2.6: Parametry przykładowego problemu

Poszczególne kroki algorytmu są prezentowane w tablicy 2.7. W każdym kroku symbolem $\times$ są oznaczone te prace, które nie należą do zbioru S' . Prace o najmniejszym koszcie są podkreślone. Optymalnym rozwiązaniem jest permutacja $p^* = (2, 5, 3, 6, 4, 1)$, której koszt wynosi 10.5.

krok	$\tilde{T}$	T_1	T_2	T_3	T_4	T_5	T_6	p^*
1	(10,16,6,12)	<u>5.75</u>	×	×	13	23.31	14.5	(1)
2	(9,14,5,11)	×	×	×	<u>10.5</u>	19.5	12	(4,1)
3	(6,10,4,8)	×	×	×	×	11.25	<u>6.5</u>	(6,4,1)
4	(5,9,3,7)	×	×	<u>5.5</u>	×	9	×	(3,6,4,1)
5	(3,7,2,4)	×	×	×	×	<u>3.85</u>	×	(5,3,6,4,1)
6	(2,4,1,2)	×	<u>0</u>	×	×	×	×	(2,5,3,6,4,1)

Tab. 2.7: Poszczególne kroki algorytmu

2.6 Problem $1|TR|max\{Ness(\tilde{C}_i \neq \tilde{d}_i)\}$

W punkcie tym zbadamy szczególny przypadek problemu $1|prec, FN|F_{max}$, w którym funkcja kosztu harmonogramu nie jest f -regularna.

W praktycznych zastosowaniach decydentowi może zależeć na tym, żeby czasy zakończenia prac były możliwie bliskie pożądanym terminom zakończenia. W takim przypadku dla każdej pracy jest niekorzystne zarówno przekroczenie zadanego terminu jak i zbyt wczesne jej zakończenie.

W modelu problemu uwzględnimy to, wykorzystując zdefiniowany w punkcie 1.5 indeks $Ness(\tilde{A} \neq \tilde{B})$. Będziemy szukać takiego harmonogramu p dla którego funkcja:

$$F(p) = \max_{i=1,\dots,n} \{Ness(\tilde{C}_i(p) \neq \tilde{d}_i)\}$$

osiąga minimum. Zauważmy najpierw, że funkcja ta nie jest f -regularna ponieważ funkcja $Ness(\tilde{C} \neq \tilde{d})$ nie jest f -monotoniczna względem argumentu $\tilde{C}$. Pokażemy ten fakt na prostym przykładzie. Niech $\tilde{C} = (1, 1, 1)$, $\tilde{d} = (3, 1, 1)$, $\tilde{A} = (2, 1, 1)$. Zauważmy, że $\tilde{A}$ jest nieujemną liczbą rozmytą. Otrzymujemy:

$$Ness(\tilde{C} \neq \tilde{d}) = 1 > Ness(\tilde{C} \tilde{+} \tilde{A} \neq \tilde{d}) = 0,$$

czyli nie jest spełniony warunek f -monotoniczności.

W celu znalezienia optymalnego harmonogramu nie możemy więc skorzystać z uogólnionego algorytmu Lawlera. Ponadto problem z tą samą funkcją kosztu harmonogramu, w którym dodatkowo dopuszczalny jest podział prac i przestoje maszyny, nie musi być równoważny (patrz twierdzenie 2.1) rozważanemu problemowi.

Ograniczymy się do prostego przypadku, w którym wszystkie liczby rozmyte są liczbami trapezowymi, nie ma narzuconej relacji poprzedzania a podział prac i przestoje maszyny nie są dopuszczalne. Pokażemy teraz, że tak zdefiniowany

problem jest NP-trudny. W dowodzie wykorzystamy następujący NP-zupełny problem podziału [3]:

Definicja 2.9 (Problem podziału). *Danych jest n dodatnich liczb całkowitych $s_1, \dots, s_n$. Należy rozstrzygnąć czy istnieje podzbiór $I^* \subseteq I = \{1, \dots, n\}$, taki że:*

$$\sum_{i \in I^*} s_i = \sum_{i \in I \setminus I^*} s_i. \quad (2.37)$$

Określimy teraz wersję decyzyjną problemu $1|TR|\max\{Ness(\tilde{C}_i \neq \tilde{d}_i)\}$.

Definicja 2.10 (Wersja decyzyjna problemu $1|TR|\max\{Ness(\tilde{C}_i \neq \tilde{d}_i)\}$). *Dla danych liczb całkowitych $N > 1$ i K , liczb trapezowych $\tilde{t}_i = (\underline{t}_i, \bar{t}_i, \alpha_{t_i}, \beta_{t_i})$, $\tilde{d}_i = (\underline{d}_i, \bar{d}_i, \alpha_{d_i}, \beta_{d_i})$, gdzie $\underline{t}_i, \bar{t}_i, \alpha_{t_i}, \beta_{t_i}, \underline{d}_i, \bar{d}_i, \alpha_{d_i}, \beta_{d_i}$, $i = 1, \dots, N$, są dodatnimi liczbami całkowitymi, rozstrzygnąć, czy istnieje permutacja p zbioru $\{1, \dots, N\}$ taka, że:*

$$\max_{i=1, \dots, N} \{Ness(\tilde{C}_i(p) \neq \tilde{d}_i)\} \leq K. \quad (2.38)$$

Twierdzenie 2.4. *Wersja decyzyjna problemu $1|TG|\max\{Ness(\tilde{C}_i \neq \tilde{d}_i)\}$ jest NP-zupełna.*

Dowód Pokażemy najpierw, że badany problem należy do klasy NP. Danymi wejściowymi problemu są: liczba całkowita N , liczba całkowita K oraz wektory trapezowych liczb rozmytych $\tilde{t}_i, \tilde{d}_i$ o długości N . Każdy element tych wektorów jest zakodowany w postaci czterech liczb całkowitych. Łącznie zakodowanie każdego problemu wymaga więc zapamiętania $4N + 2$ liczb całkowitych. Świadectwem (patrz [6]) jest permutacja σ zbioru $\{1, \dots, N\}$, do zapamiętania której wystarczy N liczb całkowitych. Wynika z tego, że rozmiar świadectwa jest ograniczony przez pewien wielomian zależny od rozmiaru problemu. Dla konkretnych danych wejściowych i permutacji σ można łatwo skonstruować wielomianowy algorytm sprawdzający czy koszt σ jest nie większy od zadanej liczby całkowitej K . W tym celu należy najpierw obliczyć rozmyte czasy zakończenia prac $\tilde{C}_i(\sigma)$ korzystając z własności operacji dodawania liczb trapezowych (patrz własność 1.5). Następnie można sprawdzić warunek (2.38) dla permutacji σ , korzystając z definicji indeksu $Ness(\tilde{A} \neq \tilde{B})$ (patrz formuła (1.32) i definicja 1.13) oraz wzorów dla liczb trapezowych, podanych w lemacie 1.11. Wynika z tego, że problem należy do klasy NP.

Pokażemy teraz, że problem podziału jest wielomianowo redukowalny do wersji decyzyjnej problemu $1|TR|\max\{Ness(\tilde{C}_i \neq \tilde{d}_i)\}$. Niech $(n; s_1, \dots, s_n)$ będzie

zadaną instancją problemu podziału. Oznaczmy $S = \frac{1}{2} \sum_{i=1}^n s_i$. Skonstruujmy odpowiednią instancję problemu szeregowania $(N, K, (\tilde{t}_i)_{i=1}^N, (\tilde{d}_i)_{i=1}^N)$ w następujący sposób:

1. $N = n + 1$,
2. $\tilde{t}_i = (s_i, s_i, 1, 1)$ dla $i = 1, \dots, n$,
3. $\tilde{t}_{n+1} = (s_{n+1}, s_{n+1}, 1, 1) = (1, 1, 1, 1)$,
4. $\tilde{d}_i = (1, 2S + 1, 1, 1)$ dla $i = 1, \dots, n$,
5. $\tilde{d}_{n+1} = (S + 1, S + 1, 1, 1)$,
6. $K = 0$.

Zauważmy, że wszystkie czasy trwania prac $\tilde{t}_i$, $i = 1, \dots, n + 1$ są dodatnimi liczbami rozmytymi. Wynika to z faktu, że s_i , $i = 1, \dots, n$, są dodatnimi liczbami całkowitymi.

Korzystając z własności operacji dodawania liczb trapezowych (własność 1.5), otrzymujemy dla każdego harmonogramu p i dla każdej pracy $i \in \{1, \dots, n + 1\}$:

$$\tilde{C}_i(p) = \left(\sum_{j \in p^i} s_j, \sum_{j \in p^i} s_j, |p^i|, |p^i| \right),$$

gdzie $p^i \subseteq \{1, \dots, n + 1\}$ jest zbiorem do którego należy praca i oraz wszystkie prace, które występują w permutacji p przed pracą i (patrz równość (2.3)). Korzystając z określenia liczby trapezowej (patrz rysunek 1.2) otrzymujemy:

1. $[\underline{d}_i(1), \bar{d}_i(1)] = [1, 2S + 1]$ dla $i = 1, \dots, n$,
2. $[\underline{d}_{n+1}(1), \bar{d}_{n+1}(1)] = [S + 1, S + 1]$,
3. $[\underline{c}_i(p, 1), \bar{c}_i(p, 1)] = [\sum_{j \in p^i} s_j, \sum_{j \in p^i} s_j]$ dla $i = 1, \dots, n + 1$.

Z warunków tych wynika, że dla każdej pracy $i = 1, \dots, n$ i dla każdej permutacji p zachodzi warunek:

$$[\underline{c}_i(p, 1), \bar{c}_i(p, 1)] \cap [\underline{d}_i(1), \bar{d}_i(1)] \neq \emptyset. \quad (2.39)$$

Z warunku (2.39) oraz z lematu 1.4 wynika, że dla każdej pracy $i = 1, \dots, n$ i dla każdej permutacji p zachodzi:

$$Poss(\tilde{C}_i(p) = \tilde{d}_i) = 1. \quad (2.40)$$

Stąd dla każdej pracy $i = 1, \dots, n$ i dla każdej permutacji p :

$$Ness(\tilde{C}_i(p) \neq \tilde{d}_i) = 1 - Poss(\tilde{C}_i(p) = \tilde{d}_i) = 0. \quad (2.41)$$

Dla pracy $n+1$ i dla każdej permutacji p otrzymujemy z lematu 1.4 następujący warunek:

$$Poss(\tilde{C}_{n+1}(p) = \tilde{d}_{n+1}) = 1 \Leftrightarrow \left[\sum_{j \in p^{n+1}} s_j, \sum_{j \in p^{n+1}} s_j \right] \cap [S+1, S+1] \neq \emptyset. \quad (2.42)$$

Ponieważ $s_{n+1} = 1$, to warunek (2.42) jest równoważny następującemu warunkowi:

$$Poss(\tilde{C}_{n+1}(p) = \tilde{d}_{n+1}) = 1 \Leftrightarrow \sum_{j \in p^{n+1} \setminus \{n+1\}} s_j = S,$$

który z kolei jest równoważny warunkowi:

$$Ness(\tilde{C}_{n+1}(p) \neq \tilde{d}_{n+1}) = 0 \Leftrightarrow \sum_{j \in p^{n+1} \setminus \{n+1\}} s_j = S. \quad (2.43)$$

Pokażemy teraz, że:

$$\exists_{I^*} \sum_{i \in I^*} s_i = \sum_{i \in I \setminus I^*} s_i \Leftrightarrow \exists_p \max_{i=1, \dots, n+1} \{Ness(\tilde{C}_i(p) \neq \tilde{d}_i)\} \leq 0. \quad (2.44)$$

$\Rightarrow$

Założmy, że jest spełniony warunek:

$$\exists_{I^*} \sum_{i \in I^*} s_i = \sum_{i \in I \setminus I^*} s_i.$$

Oznaczmy przez σ dowolną permutację zbioru I^* a przez ρ dowolną permutację zbioru $I \setminus I^*$. Utwórzmy harmonogram $p = (\sigma, n+1, \rho)$. Z założenia mamy $\sum s_{\sigma(i)} = S$. Korzystając z warunków (2.41) i (2.43) otrzymujemy dla każdego $i = 1, \dots, n+1$ $Ness(\tilde{C}_i(p) \neq \tilde{d}_i) \leq 0$, z czego wynika prawa strona równoważności (2.44).

$\Leftarrow$

Założmy, że jest spełniony warunek:

$$\exists_p \max_{i=1, \dots, n+1} \{Ness(\tilde{C}_i(p) \neq \tilde{d}_i)\} \leq 0.$$

Wynika z tego, że dla każdej pracy $i = 1, \dots, n+1$ jest spełniona nierówność $Ness(\tilde{C}_i(p) \neq \tilde{d}_i) \leq 0$. W szczególności zachodzi ona dla pracy $n+1$. Wobec

tego, korzystając z warunku (2.43) otrzymujemy $\sum_{j \in p^{n+1} \setminus \{n+1\}} s_j = S$, stąd lewa strona równoważności jest spełniona dla $I^* = p^{n+1} \setminus \{n+1\}$, co kończy dowód warunku (2.44).

Jest oczywiste, że czas konstrukcji instancji problemu szeregowania jest ograniczony z góry przez wielomian zależny od rozmiaru problemu podziału. Stąd oraz z prawdziwości warunku (2.44) wynika, że dla zdefiniowanego przekształcenia są spełnione warunki wielomianowej redukcji. Oznacza to, że wersja decyzyjna problemu $1|TR|\max\{Ness(\tilde{C}_i \neq \tilde{d}_i)\}$ jest NP-zupełna. $\square$

Twierdzenie 2.5. *Problem $1|TR|\max\{Ness(\tilde{C}_i \neq \tilde{d}_i)\}$ jest NP-trudny.*

Dowód Rozważmy szczególny przypadek problemu $1|TR|\max\{Ness(\tilde{C}_i \neq \tilde{d}_i)\}$, w którym wszystkie parametry definiujące liczby trapezowe są dodatnimi liczbami całkowitymi. Wówczas z twierdzenia 2.4 otrzymujemy, że wersja decyzyjna takiego problemu jest NP-zupełna z czego wynika, że odpowiedni problem optymalizacyjny jest NP-trudny. Jeżeli szczególny przypadek problemu optymalizacyjnego jest NP-trudny to problem ogólniejszy jest również NP-trudny. $\square$

2.7 Problem $1|FN|E(\max\{\tilde{T}_i\})$

W rozdziale tym zbadamy rozmyty problem szeregowania, dla którego nie jest określona relacja poprzedzania i funkcja kosztu harmonogramu jest zdefiniowana następująco:

$$F(p) = E \left(\max_{i=1, \dots, n} \{\tilde{T}_i(p)\} \right).$$

Z wniosku 2.1 wynika, że funkcja ta jest f -regularna. Dlatego aby rozwiązać problem wystarczy skonstruować permutację $p = (p(1), \dots, p(n))$ zbioru J , dla której wartość funkcji kosztu $F(p)$ jest minimalna.

Jeżeli wszystkie parametry problemu są zdeterminowane, to $\tilde{T}_i(p) = T_i(p)$ dla każdego $i \in J$ i zachodzi następujący warunek:

$$\max_{i=1, \dots, n} \{E(\tilde{T}_i(p))\} = E \left(\max_{i=1, \dots, n} \{\tilde{T}_i(p)\} \right) = \max_{i=1, \dots, n} \{T_i(p)\}.$$

W takim przypadku badany problem jest równoważny problemowi $1||T_{max}$, dla którego optymalny harmonogram można łatwo otrzymać, korzystając z klasycznego algorytmu Lawlera (patrz algorytm 3).

W ogólniejszym przypadku, gdy parametry są rozmyte, zachodzi jedynie następujący warunek (patrz lemat 1.8):

$$\max_{i=1,\dots,n} \{E(\tilde{T}_i(p))\} \leq E \left(\max_{i=1,\dots,n} \{\tilde{T}_i(p)\} \right),$$

zatem problemy $1|FN|\max\{E(\tilde{T}_i)\}$ i $1|FN|E(\max\{\tilde{T}_i\})$ mogą mieć różne rozwiązania optymalne. Pokażemy, że problem $1|FN|E(\max\{\tilde{T}_i\})$ jest NP-trudny nawet w przypadku, gdy tylko pożądane terminy zakończenia prac $\tilde{d}_i$, $i = 1, \dots, n$, są liczbami rozmytymi. Dowód będzie przebiegał w dwóch etapach. W pierwszym udowodnimy NP-trudność pewnego problemu ze zdeterminowanymi parametrami, który jest modyfikacją klasycznego problemu szeregowania $1||\sum T_i$, a w drugim pokażemy, że problem ten jest równoważny szczególnemu przypadkowi badanego problemu $1|FN|E(\max\{\tilde{T}_i\})$.

Wprowadźmy następujące oznaczenie:

$$\tilde{T}(p) = \max_{i=1,\dots,n} \{\tilde{T}_i(p)\}.$$

Z własności 1.4 otrzymujemy dla $\lambda \in (0, 1]$:

$$\tilde{T}(p)^\lambda = [\underline{T}(p, \lambda), \overline{T}(p, \lambda)] = \left[\max_{i=1,\dots,n} \{\underline{T}_i(p, \lambda)\}, \max_{i=1,\dots,n} \{\overline{T}_i(p, \lambda)\} \right], \quad (2.45)$$

gdzie:

$$\underline{T}_i(p, \lambda) = \max\{0, \underline{c}_i(p, \lambda) - \underline{d}_i(\lambda)\} = \max\{0, \sum_{j \in p^i} \underline{t}_j(\lambda) - \underline{d}_i(\lambda)\}, \quad i = 1, \dots, n,$$

$$\overline{T}_i(p, \lambda) = \max\{0, \overline{c}_i(p, \lambda) - \underline{d}_i(\lambda)\} = \max\{0, \sum_{j \in p^i} \overline{t}_j(\lambda) - \underline{d}_i(\lambda)\}, \quad i = 1, \dots, n.$$

Zdefiniujemy następującą funkcję $E^N : FN(\mathfrak{R}) \rightarrow \mathfrak{R}$:

Definicja 2.11. Niech $\tilde{A}$ będzie liczbą rozmytą a N liczbą naturalną. Wówczas:

$$E^N(\tilde{A}) = \frac{1}{2N} \sum_{i=1}^N \left(\underline{a}(\frac{i}{N}) + \overline{a}(\frac{i}{N}) \right). \quad (2.46)$$

Zauważmy, że $E(\tilde{A}) = \lim_{N \rightarrow \infty} E^N(\tilde{A})$. Funkcja $E^N(\tilde{A})$ może być więc traktowana jako przybliżenie wartości średniej liczby rozmytej, tym dokładniejsze im większa jest wartość N .

Określimy teraz ogólną klasę liczb rozmytych, która będzie potrzebna w dalszych rozważaniach. Niech (L_N) będzie dowolnym podciągiem ciągu $(1, \dots, N-1)$

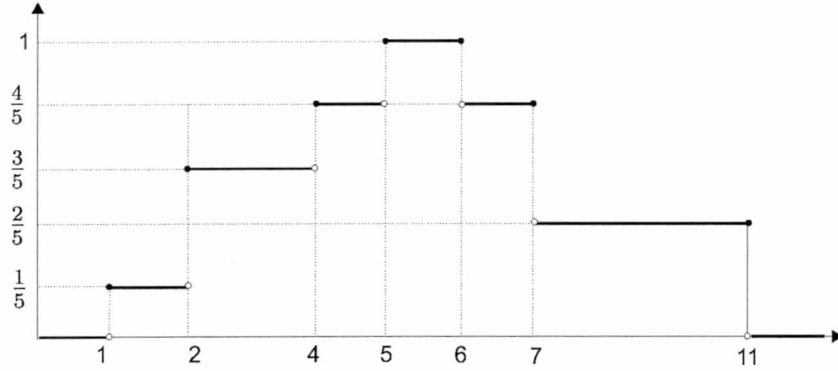
a (R_N) dowolnym podciągiem ciągu $(N-1, \dots, 1)$. Dopuszczamy również sytuację, w której (L_N) lub (R_N) są pustymi podciągami. Oznaczmy przez $|L_N|, |R_N| \in \{0, \dots, N-1\}$ liczbę elementów znajdujących się w tych ciągach. Niech będą dane liczby rzeczywiste:

$$\underline{a}_1 < \underline{a}_2 < \dots < \underline{a}_{|L_N|+1} \leq \bar{a}_{|R_N|+1} < \bar{a}_{|R_N|} < \dots < \bar{a}_1.$$

Zdefiniujmy liczbę rozmytą $\tilde{A}$, której funkcja przynależności ma następującą postać:

$$\mu_{\tilde{A}}(x) = \begin{cases} 0 & \text{dla } x \in (-\infty, \underline{a}_1), \\ \frac{L_N(i)}{N} & \text{dla } x \in [\underline{a}_i, \underline{a}_{i+1}), i = 1, \dots, |L_N|, \\ 1 & \text{dla } x \in [\underline{a}_{|L_N|+1}, \bar{a}_{|R_N|+1}], \\ \frac{R_N(i)}{N} & \text{dla } x \in (\bar{a}_{i+1}, \bar{a}_i], i = |R_N|, \dots, 1, \\ 0 & \text{dla } x \in (\bar{a}_1, \infty). \end{cases} \quad (2.47)$$

Liczbę rozmytą, której funkcja przynależności jest postaci (2.47) będziemy nazywać liczbą rozmytą typu **N -schodkowego**, gdzie N jest wartością parametru występującego we wzorze (2.47). Przykład liczby typu 5-schodkowego pokazany jest na rysunku 2.3.



Rys. 2.3: Przykład liczby rozmytej typu 5-schodkowego

Zauważmy, że każda liczba przedziałowa (w tym liczba rzeczywista) jest liczbą rozmytą typu N -schodkowego dla każdego N . Wynika to z faktu, że jeżeli przyjmiemy za (L_N) i (R_N) podciągi puste, to formuła (2.47) jest równoważna definicji liczby przedziałowej (patrz definicja 1.8). Natomiast liczba rozmyta typu $L - R$ nie jest typu N -schodkowego dla żadnego N .

Bezpośrednio z określenia liczby rozmytej typu N -schodkowego (patrz formuła (2.47)) wynika następująca własność:

Własność 2.1. Jeżeli $\tilde{A}$ jest typu N -schodkowego i $\tilde{A}^\lambda = [\underline{a}(\lambda), \bar{a}(\lambda)]$, $\lambda \in (0, 1]$, to funkcje $\underline{a}(\lambda)$ i $\bar{a}(\lambda)$ są stałe w każdym z przedziałów $(\frac{k-1}{N}, \frac{k}{N}]$, dla $k = 1, \dots, N$.

Lemat 2.9. Jeżeli dla zadanej liczby naturalnej $N > 0$ czasy trwania prac $\tilde{t}_i$ oraz pożądane terminy zakończenia prac $\tilde{d}_i$, $i = 1, \dots, n$, są liczbami rozmytymi typu N -schodkowego, to dla każdej permutacji p jest spełniona równość:

$$E(\tilde{T}(p)) = E^N(\tilde{T}(p)).$$

Dowód Jeżeli $\tilde{t}_i$ oraz $\tilde{d}_i$, $i = 1, \dots, n$, są liczbami typu N -schodkowego, to z własności 2.1 otrzymujemy, że funkcje $\underline{t}_i(\lambda)$, $\bar{t}_i(\lambda)$, $\underline{d}_i(\lambda)$, $\bar{d}_i(\lambda)$, $i = 1, \dots, n$, są stałe w każdym z przedziałów $(\frac{k-1}{N}, \frac{k}{N}]$ dla $k = 1, \dots, N$. Ponadto są one nieciągłe tylko w skończonym zbiorze punktów $\{\frac{1}{N}, \frac{2}{N}, \dots, \frac{N-1}{N}\}$. Z formuły (2.45) wynika, że dokładnie tą samą własność mają funkcje $\underline{T}(p, \lambda)$ i $\bar{T}(p, \lambda)$ dla każdej permutacji p , czyli są one również stałe w każdym z przedziałów $(\frac{k-1}{N}, \frac{k}{N}]$ dla $k = 1, \dots, N$ i nieciągłe tylko w skończonym zbiorze punktów $\{\frac{1}{N}, \frac{2}{N}, \dots, \frac{N-1}{N}\}$. Funkcje $\underline{T}(p, \lambda)$ i $\bar{T}(p, \lambda)$ są więc całkowalne w przedziale $(0, 1]$ a ponadto jest prawdziwa następująca zależność:

$$E(\tilde{T}(p)) = \frac{1}{2} \int_0^1 (\underline{T}(p, t) + \bar{T}(p, t)) dt = \frac{1}{2} \sum_{k=1}^N \int_{\frac{k-1}{N}}^{\frac{k}{N}} (\underline{T}(p, t) + \bar{T}(p, t)) dt. \quad (2.48)$$

Ponieważ:

$$\int_{\frac{k-1}{N}}^{\frac{k}{N}} (\underline{T}(p, t) + \bar{T}(p, t)) dt = \frac{1}{N} \left(\underline{T}(p, \frac{k}{N}) + \bar{T}(p, \frac{k}{N}) \right), \quad k = 1, \dots, N,$$

więc stąd oraz z (2.48) i określenia E^N otrzymujemy, że:

$$E(\tilde{T}(p)) = \frac{1}{2N} \sum_{k=1}^N \left(\underline{T}(p, \frac{k}{N}) + \bar{T}(p, \frac{k}{N}) \right) = E^N(\tilde{T}(p)),$$

co kończy dowód. □

Zdefiniujemy teraz rozmyty problem szeregowania, w którym dla zadanej liczby naturalnej $N > 0$ czasy trwania prac $\tilde{t}_i$ oraz pożądane terminy zakończenia prac $\tilde{d}_i$, $i = 1, \dots, n$, są liczbami rozmytymi typu N -schodkowego, a koszt harmonogramu jest określony następująco:

$$F(p) = E^N \left(\max_{i=1, \dots, n} \{\tilde{T}_i(p)\} \right).$$

Zakładamy, że podział prac oraz przestoje maszyn nie są dopuszczalne i relacja poprzedzania $prec = \emptyset$. Zakładamy ponadto, że $N \leq n$. Wówczas rozmiar problemu ograniczony jest przez pewien wielomian zależny od liczby prac n . Wynika to z faktu, że każda liczba N -schodkowa może zostać jednoznacznie zakodowana w postaci co najwyżej $4N + 1 \leq 4n + 1$ liczb całkowitych (patrz formuła (2.47)). Również obliczenie funkcji $E^N(\tilde{T}(p))$ dla zadanej permutacji p może być wykonane w czasie ograniczonym przez wielomian zależny tylko od liczby prac n . Tak określony problem oznaczamy symbolem $1|FN|E^N(\max\{\tilde{T}_i(p)\})$ pamiętając, że wszystkie parametry są zadane w postaci liczb rozmytych typu N -schodkowego, gdzie $N \leq n$. Prawdziwe jest następujące twierdzenie:

Twierdzenie 2.6. *Problem $1|FN|E^N(\max\{\tilde{T}_i\})$ jest szczególnym przypadkiem problemu $1|FN|E(\max\{\tilde{T}_i\})$.*

Dowód Bezpośrednia konsekwencja lematu 2.9. □

Pokażemy, że problem $1|FN|E^N(\max\{\tilde{T}_i\})$ jest NP-trudny. Z twierdzenia 2.6 wynika, że z NP-trudności tego problemu wynika bezpośrednio NP-trudność ogólniejszego problemu $1|FN|E(\max\{\tilde{T}_i\})$.

Zanim pokażemy, że problem $1|FN|E^N(\max\{\tilde{T}_i\})$ jest NP-trudny, rozważymy pewien problem szeregowania ze zdefiniowanymi parametrami, będący uogólnieniem znanego, NP-trudnego problemu $1||\sum T_i$.

Założmy, że t_i oraz d_i są liczbami całkowitymi dla $i \in J$. W takim przypadku wartości $C_i(p)$ oraz $T_i(p)$ są również liczbami całkowitymi. Zakładamy ponadto, że podział prac oraz przestoje maszyny nie są dopuszczalne oraz nie jest określona relacja poprzedzania (tzn. $prec = \emptyset$). Niech σ będzie ustaloną permutacją zbioru J a k liczbą naturalną taką, że $1 \leq k \leq n$, gdzie n jest liczbą prac w zbiorze J . Zdefiniujmy następującą funkcję kosztu harmonogramu (permutacji) p , w której σ, k, n są ustalonymi parametrami:

$$MT(p; \sigma, k, n) = \sum_{i=1}^k T_{\sigma(i)}(p) + \sum_{i=k+1}^n \max_{j=k, \dots, i} \{T_{\sigma(j)}(p)\}. \quad (2.49)$$

Przykład: Niech $n = 5$, $k = 3$, $\sigma = (2, 3, 1, 4, 5)$. Wówczas:

$$MT(p; (2, 3, 1, 4, 5), 3, 5) = T_2(p) + T_3(p) + T_1(p) + \max\{T_1(p), T_4(p)\} + \max\{T_1(p), T_4(p), T_5(p)\} \quad \square$$

Szukamy takiej permutacji p dla której funkcja $MT(p; \sigma, k, n)$ osiąga minimum. Tak zdefiniowany problem szeregowania oznaczamy symbolem $1||MT(\sigma, k, n)$.

Własność 2.2. *Problem $1||MT(\sigma, n, n)$ jest NP-trudny.*

Dowód Jeżeli $k = n$, to:

$$MT(p; \sigma, k, n) = \sum_{i=1}^n T_{\sigma(i)}(p) = \sum_{i=1}^n T_i(p),$$

co oznacza, że problem $1||MT(\sigma, n, n)$ jest równoważny znanemu problemowi $1||\sum T_i$, który jest NP-trudny (zobacz [3]). $\square$

Pokażemy, że problem $1||MT(\sigma, 1, n)$ będący szczególnym przypadkiem problemu $1||MT(\sigma, k, n)$, w którym $k = 1$ jest również NP-trudny.

Niech $Mv_{u,v}(\sigma)$, gdzie $u, v \in \{1, \dots, n\}$, oznacza permutację, która powstała z permutacji σ poprzez przesunięcie elementu $\sigma(u)$ na pozycję v , tzn.:

$$Mv_{u,v}(\sigma) = (\sigma(1), \dots, \sigma(u-1), \sigma(u+1), \dots, \sigma(v-1), \sigma(u), \sigma(v), \dots, \sigma(n)).$$

Przykład: $Mv_{2,5}((2, \underline{4}, 5, 6, 1, 3)) = (2, 5, 6, 1, \underline{4}, 3)$. $\square$

Udowodnimy następujący lemat:

Lemat 2.10. *Dla każdej permutacji p i dla każdego $1 < k \leq n$ zachodzą następujące warunki:*

$$\forall_{v=k-1, \dots, n} MT(p; \sigma, k, n) \leq MT(p, Mv_{k-1,v}(\sigma), k-1, n), \quad (2.50)$$

$$\exists_{v=k-1, \dots, n} MT(p; \sigma, k, n) = MT(p, Mv_{k-1,v}(\sigma), k-1, n). \quad (2.51)$$

Dowód Oznaczmy $\sigma_v = Mv_{k-1,v}(\sigma)$ dla $v = k-1, \dots, n$. Wprowadźmy również następujące oznaczenia:

$$MT(p; \sigma, k, n) = \sum_{i=1}^n \max A_i(p, \sigma, k), \quad (2.52)$$

$$MT(p; \sigma_v, k-1, n) = \sum_{i=1}^n \max A_i(p, \sigma_v, k-1), \quad (2.53)$$

gdzie zbiory $A_i(p, \sigma, k)$ i $A_i(p, \sigma_v, k-1)$, $i = 1, \dots, n$, są określone w następujący sposób:

$$A_i(p, \sigma, k) = \begin{cases} \{T_{\sigma(i)}(p)\} & \text{dla } i = 1, \dots, k, \\ \{T_{\sigma(k)}(p), \dots, T_{\sigma(i)}(p)\} & \text{dla } i = k+1, \dots, n, \end{cases} \quad (2.54)$$

$$A_i(p, \sigma_v, k-1) = \begin{cases} \{T_{\sigma(i)}(p)\} & \text{dla } i = 1, \dots, k-2, \\ \{T_{\sigma(k)}(p), \dots, T_{\sigma(i+1)}(p)\} & \text{dla } i = k-1, \dots, v-1, \\ \{T_{\sigma(k-1)}(p), \dots, T_{\sigma(v)}(p)\} & \text{dla } i = v, \\ \{T_{\sigma(k-1)}(p), \dots, T_{\sigma(i)}(p)\} & \text{dla } i = v+1, \dots, n. \end{cases} \quad (2.55)$$

Niech $\rho = (1, 2, \dots, n)$ i $\rho_v = Mv_{k-1,v}(\rho)$, $v = k-1, \dots, n$. Jest oczywiste, że:

$$MT(p; \sigma, k, n) = \sum_{i=1}^n \max A_i(p, \sigma, k) = \sum_{i=1}^n \max A_{\rho_v(i)}(p, \sigma, k), \quad (2.56)$$

gdzie zbiory $A_{\rho_v(i)}(p, \sigma, k)$, $i = 1, \dots, n$, są określone następująco:

$$A_{\rho_v(i)}(p, \sigma, k) = \begin{cases} \{T_{\sigma(i)}(p)\} & \text{dla } i = 1, \dots, k-2, \\ \{T_{\sigma(k)}(p), \dots, T_{\sigma(i+1)}(p)\} & \text{dla } i = k-1, \dots, v-1, \\ \{T_{\sigma(k-1)}(p)\} & \text{dla } i = v, \\ \{T_{\sigma(k)}(p), \dots, T_{\sigma(i)}(p)\} & \text{dla } i = v+1, \dots, n. \end{cases} \quad (2.57)$$

Z równości (2.55) i (2.57) otrzymujemy dla każdego $v = k-1, \dots, n$:

$$A_{\rho_v(i)}(p, \sigma, k) = A_i(p, \sigma_v, k-1) \quad \text{dla } i = 1, \dots, v-1, \quad (2.58)$$

$$A_{\rho_v(i)}(p, \sigma, k) \subseteq A_i(p, \sigma_v, k-1) \quad \text{dla } i = v, \dots, n. \quad (2.59)$$

Wynika z tego, że dla każdego $v = k-1, \dots, n$ i dla każdego $i = 1, \dots, n$ zachodzi:

$$\max A_{\rho_v(i)}(p, \sigma, k) \leq \max A_i(p, \sigma_v, k-1),$$

z czego wynika, że dla każdego $v = k-1, \dots, n$:

$$\sum_{i=1}^n \max A_{\rho_v(i)}(p, \sigma, k) \leq \sum_{i=1}^n \max A_i(p, \sigma_v, k-1). \quad (2.60)$$

Z warunków (2.53), (2.56) i (2.60) wynika pierwsza część tezy lematu (warunek (2.50)). Pokażemy teraz, że istnieje $v^* \in \{k-1, \dots, n\}$, takie, że:

$$\forall_{i=1, \dots, n} \max A_{\rho_{v^*}(i)}(p, \sigma, k) = \max A_i(p, \sigma_{v^*}, k-1). \quad (2.61)$$

Zauważmy, że wobec warunku (2.58) wystarczy pokazać, że warunek (2.61) zachodzi dla każdego $i = v^*, \dots, n$. Zdefiniujmy v^* w następujący sposób:

$$v^* = \begin{cases} k-1 & \text{jeżeli } T_{\sigma(k-1)}(p) \leq T_{\sigma(k)}(p), \\ \max\{i \mid k \leq i \leq n, \forall_{j=k, \dots, i} T_{\sigma(k-1)}(p) > T_{\sigma(j)}(p)\} & \text{jeżeli } T_{\sigma(k-1)}(p) > T_{\sigma(k)}(p). \end{cases} \quad (2.62)$$

Pokażemy najpierw, że warunek (2.61) zachodzi dla $i = v^*$. W tym celu należy pokazać (patrz równości (2.55) i (2.57)), że:

$$\max\{T_{\sigma(k-1)}(p), \dots, T_{\sigma(v^*)}(p)\} = T_{\sigma(k-1)}(p). \quad (2.63)$$

Jeżeli $v^* = k - 1$, to warunek (2.63) jest oczywisty. Jeżeli natomiast $v^* > k - 1$, to z określenia (2.62) wynika, że $T_{\sigma(k-1)} > T_{\sigma(k)} > \dots > T_{\sigma(v^*)}$ i warunek (2.63) jest również prawdziwy.

Pokażemy teraz, że warunek (2.61) zachodzi również dla $i \in \{v^* + 1, \dots, n\}$. W tym celu należy pokazać (patrz równości (2.55) i (2.57)), że:

$$\max\{T_{\sigma(k-1)}(p), \dots, T_{\sigma(i)}(p)\} = \max\{T_{\sigma(k)}(p), \dots, T_{\sigma(i)}(p)\}. \quad (2.64)$$

Prawdziwość warunku (2.64) wynika z tego, że $i \geq v^* + 1 > k - 1$ oraz $T_{\sigma(v^*+1)} \geq T_{\sigma(k-1)}$ (patrz określenie (2.62)).

Z warunków (2.63) i (2.64) wynika, że warunek (2.61) zachodzi dla v^* . Stąd otrzymujemy:

$$\sum_{i=1}^n \max A_{\rho_{v^*}(i)}(p, \sigma, k) = \sum_{i=1}^n \max A_i(p, \sigma, k) = \sum_{i=1}^n \max A_i(p, \sigma_{v^*}, k - 1),$$

czyli, wobec określeń (2.52) i (2.53), druga część tezy lematu (warunek (2.51)) jest również prawdziwa. $\square$

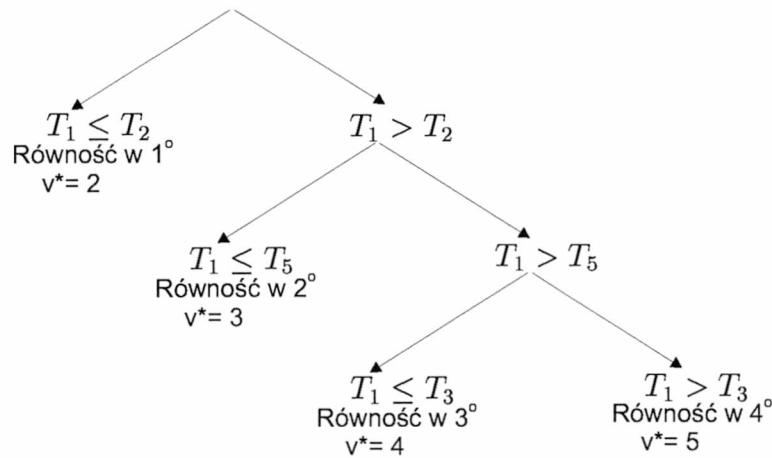
Zilustrujemy powyższy dowód na następującym przykładzie:

Przykład: Niech $n=5$, $k=3$ i $\sigma = (4, 1, 2, 5, 3)$. W celu uproszczenia oznaczeń założmy, że $\max\{A, B\} \equiv \{A, B\}$ i $T_i(p) \equiv T_i$. Przesuwany element T_1 jest podkreślony. Dla każdej permutacji p otrzymujemy:

$$\begin{aligned} 1^\circ \quad MT(p; Mv_{2,2}(\sigma), 2, 5) &= \{T_4\} + \{\underline{T_1}\} + \{\underline{T_1}, T_2\} + \{\underline{T_1}, T_2, T_5\} + \{\underline{T_1}, T_2, T_5, T_3\} \\ MT(p; \sigma, 3, 5) &= \{T_4\} + \{\underline{T_1}\} + \{T_2\} + \{T_2, T_5\} + \{T_2, T_5, T_3\} \\ 2^\circ \quad MT(p; Mv_{2,3}(\sigma), 2, 5) &= \{T_4\} + \{T_2\} + \{T_2, \underline{T_1}\} + \{T_2, \underline{T_1}, T_5\} + \{T_2, \underline{T_1}, T_5, T_3\} \\ MT(p; \sigma, 3, 5) &= \{T_4\} + \{T_2\} + \{\underline{T_1}\} + \{T_2, T_5\} + \{T_2, T_5, T_3\} \\ 3^\circ \quad MT(p; Mv_{2,4}(\sigma), 2, 5) &= \{T_4\} + \{T_2\} + \{T_2, T_5\} + \{T_2, T_5, \underline{T_1}\} + \{T_2, T_5, \underline{T_1}, T_3\} \\ MT(p; \sigma, 3, 5) &= \{T_4\} + \{T_2\} + \{T_2, T_5\} + \{\underline{T_1}\} + \{T_2, T_5, T_3\} \\ 4^\circ \quad MT(p; Mv_{2,5}(\sigma), 2, 5) &= \{T_4\} + \{T_2\} + \{T_2, T_5\} + \{T_2, T_5, T_3\} + \{T_2, T_5, T_3, \underline{T_1}\} \\ MT(p; \sigma, 3, 5) &= \{T_4\} + \{T_2\} + \{T_2, T_5\} + \{T_2, T_5, T_3\} + \{\underline{T_1}\} \end{aligned}$$

Łatwo zaobserwować, porównując kolejne składniki sum w każdym z punktów od 1° do 4° , że dla każdego $v = 2, \dots, 5$ jest spełniona nierówność $MT(p; \sigma, 3, 5) \leq MT(p; Mv_{2,v}(\sigma), 2, 5)$.

Pokażemy teraz, że w przynajmniej jednym punkcie od 1° do 4° zachodzi równość $MT(p; \sigma, 3, 5) = MT(p; Mv_{2,v}(\sigma), 2, 5)$. Rozpatrzmy drzewo pokazane na rysunku 2.4.



Rys. 2.4: Możliwe przypadki relacji między T_1 i T_2, T_5, T_3

Przedstawia ono możliwe przypadki relacji między przesuwany elementem T_1 i elementami T_2, T_5, T_3 . Drzewo to jest graficzną reprezentacją formuły (2.62). Korzystając z tej reprezentacji otrzymujemy $v^* \in \{2, 3, 4, 5\}$, dla którego spełniona jest równość $MT(p; \sigma, 3, 5) = MT(p; Mv_{2,v^*}(\sigma), 2, 5)$. Na przykład, jeżeli $T_1 > T_2$ i $T_1 \leq T_5$, to w 2° mamy $MT(p; \sigma, 3, 5) = MT(p; Mv_{2,3}(\sigma), 2, 5)$. $\square$

Wniosek 2.2. Dla każdej permutacji p zachodzi warunek:

$$MT(p; \sigma, k, n) = \min_{v=k-1, \dots, n} \{MT(p; Mv_{k-1,v}(\sigma), k-1, n)\}.$$

Dowód Bezpośrednia konsekwencja lematu 2.10. $\square$

Lemat 2.11. Niech σ i ρ będą dwiema permutacjami zbioru $J = \{1, \dots, n\}$. Wówczas problem $1 \parallel MT(\rho, k, n)$ jest T -transformowalny⁴ do problemu $1 \parallel MT(\sigma, k, n)$.

⁴T-transformacja oznacza wielomianową transformację Turinga problemu optymalizacyjnego π_1 do problemu optymalizacyjnego π_2 (patrz [2]). Pojęciem tym określamy algorytm A rozwiązujący problem π_1 z zastosowaniem Algorytmu B rozwiązującego problem π_2 , przy czym złożoność algorytmu A jest wielomianowa jeżeli złożoność algorytmu B jest wielomianowa. Problem optymalizacyjny P_1 jest NP-trudny jeżeli istnieje problem optymalizacyjny P_2 , który jest T -transformowalny do P_1 .

Dowód Niech $(n, k, \rho, (t_i)_{i=1}^n, (d_i)_{i=1}^n)$ będzie instancją problemu $1||MT(\rho, k, n)$, a $(n, k, \sigma, (t_i^*)_{i=1}^n, (d_i^*)_{i=1}^n)$ odpowiednią instancją problemu $1||MT(\sigma, k, n)$ gdzie:

$$t_{\sigma(i)}^* = t_{\rho(i)}, \quad i = 1, \dots, n,$$

$$d_{\sigma(i)}^* = d_{\rho(i)}, \quad i = 1, \dots, n.$$

Wówczas dla każdej permutacji p zachodzi $T_{\rho(i)}(p) = T_{\sigma(i)}(p)$, z czego wynika, że:

$$MT(p; \rho, k, n) = MT(p; \sigma, k, n).$$

Można więc skonstruować wielomianowy algorytm dla problemu $1||MT(\rho, k, n)$ przy założeniu, że istnieje wielomianowy algorytm dla problemu $1||MT(\sigma, k, n)$. W tym celu wystarczy jedynie w odpowiedni sposób przenieść parametry t_i oraz d_i , $i = 1, \dots, n$. $\square$

Z lematu tego wynika, że wybór permutacji σ w problemie $1||MT(\sigma, k, n)$ nie ma wpływu na jego złożoność obliczeniową.

Lemat 2.12. *Problem $1||MT(\sigma, k, n)$ jest T-transformowalny do problemu $1||MT(\sigma, k-1, n)$.*

Dowód Z wniosku 2.2 wynika, że aby rozwiązać problem $1||MT(\sigma, k, n)$ wystarczy rozwiązać wielomianową liczbę (równą $n-k+2$) problemów $1||MT(\rho_j, k-1, n)$ dla tego samego zbioru parametrów t_i , d_i , ze zmniejszoną o jeden wartośćią parametru k i lekko zmodyfikowaną permutacją σ . Fakty te zostały wykorzystane do konstrukcji algorytmu 8.

Algorytm 8 T-transformacja $1||MT(\sigma, k, n)$ do $1||MT(\sigma, k-1, n)$

Require: $n, k, \sigma, (t_i)_{i=1}^n, (d_i)_{i=1}^n$

Ensure: p^*

```

1:  $p^* \leftarrow ()$  /* Optymalne rozwiązanie problemu  $1||MT(\sigma, k, n)$  */
2:  $BestEst \leftarrow \infty$  /* Ocena  $p^*$  */
3: for all  $j \in \{k-1, \dots, n\}$  do
4:    $\rho \leftarrow Mv_{k-1,j}(\sigma)$ 
5:   Rozwiąż problem  $1||MT(\rho, k-1, n)$ . Zapamiętaj jego rozwiązanie jako  $p$ .
6:   if  $MT(p; \rho, k-1, n) < BestEst$  then
7:      $BestEst \leftarrow MT(p; \rho, k-1, n)$ 
8:      $p^* \leftarrow p$ 
9:   end if
10: end for
11: return  $p^*$ 
```

Algorytm 8 jest wielomianowy jeżeli krok 5 może być wykonany w czasie ograniczonym przez wielomian zależny od rozmiaru problemu $1||MT(\rho_j, k-1, n)$, czyli wtedy i tylko wtedy, gdy istnieje wielomianowy algorytm rozwiązujący każdy z problemów $1||MT(\rho_j, k-1, n)$, gdzie $\rho_j = Mv_{k-1,j}(\sigma)$.

Z lematu 2.11 wynika, że istnieją wielomianowe algorytmy dla wszystkich problemów $1||MT(\rho_j, k-1, n)$ wtedy i tylko wtedy gdy istnieje wielomianowy algorytm dla problemu $1||MT(\sigma, k-1, n)$. Wynika z tego, że problem $1||MT(\sigma, k, n)$ jest T-transformowalny do problemu $1||MT(\sigma, k-1, n)$. $\square$

Twierdzenie 2.7. *Problem $1||MT(\sigma, 1, n)$ jest NP-trudny.*

Dowód Z własności 2.2 wiemy, że problem $1||MT(\sigma, n, n)$ jest NP-trudny. Z lematu 2.12 wynika, że problem ten jest T-transformowalny do problemu

$$1||MT(\sigma, n-1, n),$$

czyli problem $1||MT(\sigma, n-1, n)$ jest również NP-trudny. Stosując indukcję względem parametru k dowodzimy, że problem $1||MT(\sigma, 1, n)$ jest NP-trudny. $\square$

Pokażemy teraz, że problem $1|FN|E^N(\text{m}\ddot{\text{a}}\text{x}\{\tilde{T}_i\})$ jest NP-trudny. W dowodzie wykorzystamy problem $1||MT(\sigma, 1, n)$, którego NP-trudność udowodniliśmy w twierdzeniu 2.7.

Twierdzenie 2.8. *Problem $1|FN|E^N(\text{m}\ddot{\text{a}}\text{x}\{\tilde{T}_i\})$ jest NP-trudny nawet jeżeli spośród wszystkich parametrów tylko pożądane terminy zakończenia prac są rozmyte.*

Dowód Pokażemy, że NP-trudny problem $1||MT(\sigma, 1, n)$ jest T-transformowalny do problemu $1|FN|E^N(\text{m}\ddot{\text{a}}\text{x}\{\tilde{T}_i\})$. Niech $(\sigma, n, (t_i)_{i=1}^n, (d_i)_{i=1}^n)$ będzie daną instancją problemu $1||MT(\sigma, 1, n)$. Zdefiniujemy $c = \sum_{i=1}^n t_i$. Możemy przyjąć, że $d_i \leq c$ dla każdego $i = 1, \dots, n$. Jeżeli nie, to optymalne rozwiązanie problemu nie ulegnie zmianie jeżeli podstawimy $d_i = c$. Ustalmy $\sigma = (n, n-1, \dots, 1)$ (patrz lemat 2.11). Przez $C_i(p)$ i $T_i(p)$ będziemy oznaczać odpowiednio czas zakończenia i opóźnienie pracy i w permutacji p dla problemu $1||MT(\sigma, 1, n)$.

Zdefiniujemy teraz odpowiednią instancję $(n, N, (\tilde{t}_i)_{i=1}^n, (\tilde{d}_i)_{i=1}^n)$ rozmytego problemu $1|E^N(\text{m}\ddot{\text{a}}\text{x}\{\tilde{T}_i\})$ w następujący sposób:

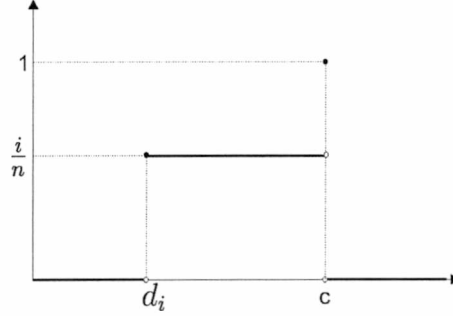
1. $N = n$,
2. $\tilde{t}_i, i = 1, \dots, n$, są liczbami rozmytymi, takimi że:

$$\mu_{\tilde{t}_i}(x) = \begin{cases} 1 & x = t_i, \\ 0 & x \neq t_i, \end{cases}$$

stąd $\tilde{t}_i^\lambda = [t_i, t_i]$ dla $\lambda \in (0, 1]$ ($\tilde{t}_i$ są więc zdeterminowane).

3. $\tilde{d}_i$, $i = 1, \dots, n$, są liczbami rozmytymi takimi, że (patrz rysunek 2.5):

$$\mu_{\tilde{d}_i}(x) = \begin{cases} 1 & \text{dla } x = c, \\ \frac{i}{n} & \text{dla } x \in [d_i, c), \\ 0 & \text{dla } x \in (-\infty, d_i) \cup (c, \infty), \end{cases}$$



Rys. 2.5: Funkcja przynależności $\mu_{\tilde{d}_i}(x)$

stąd:

$$\tilde{d}_i^\lambda = \begin{cases} [d_i, c] & \text{dla } \lambda \leq \frac{i}{n}, \\ [c, c] & \text{dla } \lambda > \frac{i}{n}. \end{cases} \quad (2.65)$$

Zauważmy, że $\tilde{t}_i$ oraz $\tilde{d}_i$, $i = 1, \dots, n$, są liczbami rozmytymi typu N -schodkowego. Stąd oraz z faktu, że $N = n$ wynika, że $(N, n, (\tilde{t}_i)_{i=1}^n, (\tilde{d}_i)_{i=1}^n)$ jest instancją problemu $1|FN|E^N(\max\{\tilde{T}_i\})$. Z definicji funkcji $E^N(\tilde{A})$ i formuły (2.45) otrzymujemy:

$$E^N(\tilde{T}(p)) = \frac{1}{2N} \sum_{j=1}^N \left(\max_{i=1, \dots, N} \{\underline{T}_i(p, \frac{j}{N})\} + \max_{i=1, \dots, N} \{\overline{T}_i(p, \frac{j}{N})\} \right).$$

Dla każdego $i = 1, \dots, N$ i dla każdego $j = 1, \dots, N$ zachodzi:

$$\underline{T}_i(p, \frac{j}{N}) = \max\{0, \underline{c}_i(p, \frac{j}{N}) - \underline{d}_i(\frac{j}{N})\} = \max\{0, C_i(p) - c\} = 0, \quad (2.66)$$

$$\overline{T}_i(p, \frac{j}{N}) = \max\{0, \overline{c}_i(p, \frac{j}{N}) - \underline{d}_i(\frac{j}{N})\} = \max\{0, C_i(p) - \underline{d}_i(\frac{j}{N})\}. \quad (2.67)$$

Z (2.65) otrzymujemy dla każdego $i = 1, \dots, N$:

$$\underline{d}_i(\lambda) = \begin{cases} d_i & \text{dla } \lambda \leq \frac{i}{N}, \\ c & \text{dla } \lambda > \frac{i}{N}. \end{cases} \quad (2.68)$$

Podstawiając $\lambda = \frac{j}{N}$, $j = 1, \dots, N$, do formuły (2.68) otrzymujemy:

$$\underline{d}_i\left(\frac{j}{N}\right) = \begin{cases} d_i & \text{dla } 1 \leq j \leq i \leq N, \\ c & \text{dla } N \geq j > i \geq 1. \end{cases} \quad (2.69)$$

Z (2.67) i (2.69) wynika, że:

$$\bar{T}_i\left(p, \frac{j}{N}\right) = \begin{cases} \max\{0, C_i(p) - d_i\} = T_i(p) & \text{dla } 1 \leq j \leq i \leq N, \\ \max\{0, C_i(p) - c\} = 0 & \text{dla } N \geq j > i \geq 1. \end{cases} \quad (2.70)$$

Z (2.67) i (2.70) otrzymujemy:

$$E^N(\tilde{T}(p)) = \frac{1}{2N} \sum_{j=1}^N \max_{i=1, \dots, N} \{\bar{T}_i(p, \frac{j}{N})\} = \frac{1}{2N} \sum_{j=1}^N \max_{i=j, \dots, N} \{T_i(p)\}.$$

Stąd, oraz z określenia funkcji $MT(p; \sigma, 1, n)$ i faktu, że $N = n$ otrzymujemy ostatecznie:

$$E^N(\tilde{T}(p)) = \frac{1}{2} MT(p; \sigma, 1, n). \quad (2.71)$$

Z równości (2.71) wynika, że problem $1||MT(\sigma, 1, n)$ i odpowiadający mu problem $1|FN|E^N(\max\{\tilde{T}_i\})$ mają takie same rozwiązania optymalne. Jeżeli istnieje wielomianowy algorytm dla problemu $1|FN|E^N(\max\{\tilde{T}_i\})$, to można łatwo skonstruować wielomianowy algorytm dla problemu $1|MT(\sigma, 1, n)$. Wynika z tego, że problem $1|FN|E^N(\max\{\tilde{T}_i\})$ jest NP-trudny. Z określenia transformacji wynika, że problem ten jest NP-trudny nawet jeżeli spośród wszystkich parametrów tylko pożądane terminy zakończenia prac są rozmyte (patrz kroki transformacji 1 - 3). $\square$

Twierdzenie 2.9. *Problem $1|FN|E(\max\{\tilde{T}_i\})$ jest NP-trudny nawet jeżeli spośród wszystkich parametrów tylko pożądane terminy zakończenia prac są rozmyte.*

Dowód Ponieważ szczególny przypadek problemu $1|FN|E(\max\{\tilde{T}_i\})$, w którym czasy są zdeterminowane jest NP-trudny (patrz twierdzenie 2.6 i twierdzenie 2.8), to ogólniejszy problem jest również NP-trudny. $\square$

2.8 Problem $1|FN|E(\max\{\tilde{L}_i\})$

W punkcie tym zbadamy rozmyty problem szeregowania, dla którego nie jest określona relacja poprzedzania a funkcja kosztu harmonogramu jest określona następująco:

$$F(p) = E\left(\max_{i=1, \dots, n} \{\tilde{L}_i(p)\}\right).$$

Problem ten jest więc podobny do problemu $1|FN|E(\text{m}\ddot{\text{a}}\text{x}\{\tilde{T}_i\})$, który był rozpatrywany w poprzednim punkcie (zamiast rozmytego opóźnienia w funkcji kosztu stosujemy rozmytą nieterminowość). Wprowadźmy następujące oznaczenie:

$$\tilde{L}(p) = \text{m}\ddot{\text{a}}\text{x}_{i=1,\dots,n} \{\tilde{L}_i(p)\}.$$

Z własności 1.4 otrzymujemy dla każdego $\lambda \in (0, 1]$:

$$\tilde{L}(p)^\lambda = [\underline{L}(p, \lambda), \overline{L}(p, \lambda)] = \left[\max_{i=1,\dots,n} \{\underline{L}_i(p, \lambda)\}, \max_{i=1,\dots,n} \{\overline{L}_i(p, \lambda)\} \right], \quad (2.72)$$

gdzie:

$$\begin{aligned} \underline{L}_i(p, \lambda) &= \underline{c}_i(p, \lambda) - \bar{d}_i(\lambda) = \sum_{j \in p^i} \underline{t}_j(\lambda) - \bar{d}_i(\lambda), \\ \overline{L}_i(p, \lambda) &= \bar{c}_i(p, \lambda) - \underline{d}_i(\lambda) = \sum_{j \in p^i} \bar{t}_j(\lambda) - \underline{d}_i(\lambda). \end{aligned}$$

Pokażemy, że problem $1|FN|E(\text{m}\ddot{\text{a}}\text{x}\{\tilde{L}_i\})$ jest również NP-trudny. Do dowodu wykorzystamy wyniki uzyskane w punkcie 2.7. Rozważmy najpierw pewien problem szeregowania ze zdeterminowanymi parametrami. Załóżmy, że $t_i, d_i, i = 1, \dots, n$, są liczbami całkowitymi, nie jest określona relacja poprzedzania ($prec = \emptyset$) a podział prac i przestoje maszyny nie są dozwolone. Wówczas czasy zakończenia prac $C_i(p)$ oraz nieterminowości $L_i(p), i = 1, \dots, n$, są również całkowite. Szukamy takiej permutacji p , dla której funkcja:

$$ML(p) = \sum_{i=1}^n \max\{L_i(p), \dots, L_n(p)\} \quad (2.73)$$

osiąga minimum. Problem ten będziemy oznaczać symbolem $1||ML$.

Zdefiniujmy $\sigma_n = (n, n-1, \dots, 1)$. Udowodnimy następujący lemat:

Lemat 2.13. *Niech $(t_i)_{i=1}^n, (d_i)_{i=1}^n$ będą zadanymi wektorami liczb całkowitych, gdzie dodatkowo $t_n = d_n = 0$. Wówczas dla każdej permutacji $\pi = (n, p)$, gdzie p jest dowolną permutacją zbioru $\{1, \dots, n-1\}$, są spełnione równości:*

$$MT(\pi; \sigma_n, 1, n) = ML(\pi), \quad (2.74)$$

$$MT(\pi; \sigma_n, 1, n) = MT(p; \sigma_{n-1}, 1, n-1). \quad (2.75)$$

Dowód Z określenia funkcji $ML(\pi)$ (patrz równość (2.73)) otrzymujemy:

$$ML(\pi) = L_n(\pi) + \max\{L_n(\pi), L_{n-1}(\pi)\} + \dots + \max\{L_n(\pi), L_{n-1}(\pi), \dots, L_1(\pi)\}.$$

Ponieważ z założenia $t_n = d_n = 0$ i $\pi = (n, p)$, więc $L_n(\pi) = T_n(\pi) = 0$, z czego wynika następująca zależność:

$$\begin{aligned} ML(\pi) &= 0 + \max\{0, L_{n-1}(\pi)\} + \dots + \max\{0, L_{n-1}(\pi), \dots, L_1(\pi)\} = \\ &= T_n(\pi) + \max\{T_n(\pi), T_{n-1}(\pi)\} + \dots + \max\{T_n(\pi), T_{n-1}(\pi), \dots, T_1(\pi)\} = \\ &= MT(\pi; \sigma_n, 1, n), \end{aligned}$$

czyli równość (2.74) jest prawdziwa.

Z faktu, że $C_n(\pi) = 0$ i $\pi = (n, p)$ wynika, że $C_i(\pi) = C_i(p)$ i $T_i(\pi) = T_i(p)$ dla każdego $i = 1, \dots, n-1$. Wynika z tego następująca zależność:

$$\begin{aligned} MT(\pi; \sigma_n, 1, n) &= 0 + \max\{0, T_{n-1}(\pi)\} + \dots + \max\{0, T_{n-1}(\pi), \dots, T_1(\pi)\} = \\ &= \max\{T_{n-1}(p)\} + \dots + \max\{T_{n-1}(p), \dots, T_1(p)\} = \\ &= MT(p; \sigma_{n-1}, 1, n-1), \end{aligned}$$

czyli równość (2.75) jest również prawdziwa. □

Twierdzenie 2.10. *Problem $1||ML$ jest NP-trudny.*

Dowód Pokażemy, że problem $1||MT(\sigma, 1, n)$, którego NP-trudność pokazaliśmy w punkcie 2.7 jest T-transformowalny do problemu $1||ML$. Weźmy dowolną instancję $((t_i)_{i=1}^n, (d_i)_{i=1}^n, \sigma, n)$ problemu $1||MT(\sigma, 1, n)$. Ponieważ problem ten jest NP-trudny dla każdego σ , więc ustalmy $\sigma = (n, n-1, \dots, 1) = \sigma_n$. Odpowiedni T-transformacją jest algorytm 9.

Algorytm 9 T-transformacja $1||MT(\sigma, 1, n)$ do $1||ML$

Require: $n, (t_i)_{i=1}^n, (d_i)_{i=1}^n$ /* Parametry problemu $1||MT(\sigma_n, 1, n)$ */

Ensure: p /* Optymalne rozwiązanie problemu $1||MT(\sigma_n, 1, n)$ */

- 1: $t_{n+1} \leftarrow 0$
 - 2: $d_{n+1} \leftarrow 0$
 - 3: Rozwiąż problem $1||ML$ dla parametrów $n+1, (t_i)_{i=1}^{n+1}, (d_i)_{i=1}^{n+1}$. Podstaw pod π optymalną permutację dla tego problemu.
 - 4: Przesuń pracę $n+1$ w π na pozycję 1
 - 5: $p \leftarrow (\pi(2), \dots, \pi(n))$
 - 6: **return** p
-

Zauważmy, że algorytm 9 jest wielomianowy jeżeli krok 3 jest realizowany w czasie ograniczonym przez wielomian zależny od rozmiaru problemu $1||ML$, czyli jeżeli problem $1||ML$ ma złożoność wielomianową.

Pokażemy teraz, że permutacja p skonstruowana przez algorytm 9 jest rzeczywiście optymalnym rozwiązaniem problemu $1||MT(\sigma_n, 1, n)$. Zauważmy, że permutacja π , po przekształceniu w kroku 4 algorytmu, jest również optymalnym rozwiązaniem problemu $1||ML$ dla parametrów $n+1, (t_i)_{i=1}^{n+1}, (d_i)_{i=1}^{n+1}$. Wynika to z faktu, że przesunięcie pracy $n+1$ (której czas trwania wynosi 0) na początek permutacji π nie spowoduje zwiększenia funkcji kosztu $ML(\pi)$. Zauważmy również (patrz krok 5 algorytmu), że $\pi = (n+1, p)$. Ponieważ $t_{n+1} = d_{n+1} = 0$, więc korzystając z lematu 2.13 otrzymujemy następującą zależność:

$$MT(\pi; \sigma_{n+1}, 1, n+1) = MT(p; \sigma_n, 1, n) = ML(\pi). \quad (2.76)$$

Założmy, że p nie jest optymalnym rozwiązaniem problemu $1||MT(\sigma_n, 1, n)$. Istnieje wówczas permutacja p^* zbioru $(1, \dots, n)$ taka, że:

$$MT(p^*; \sigma_n, 1, n) < MT(p; \sigma_n, 1, n). \quad (2.77)$$

Niech $\pi^* = (n+1, p^*)$. Ponieważ $t_{n+1} = d_{n+1} = 0$, więc ponownie korzystając z lematu 2.13 otrzymujemy:

$$MT(\pi^*; \sigma_{n+1}, 1, n+1) = MT(p^*; \sigma_n, 1, n) = ML(\pi^*),$$

stąd oraz z warunków (2.76) i (2.77) otrzymujemy, że:

$$ML(\pi^*) < ML(\pi),$$

co daje sprzeczność z założeniem optymalności permutacji π dla problemu $1||ML$. Oznacza to, że permutacja p skonstruowana przez algorytm 9 jest optymalnym rozwiązaniem problemu $1||MT(\sigma_n, 1, n)$.

Udowodniliśmy, że gdyby istniał wielomianowy algorytm dla problemu $1||ML$, to dałoby się skonstruować wielomianowy algorytm dla NP-trudnego problemu $1||MT(\sigma, 1, n)$. Oznacza to, że problem $1||ML$ jest NP-trudny. $\square$

Pokażemy teraz, że problem $1||E(\text{m}\ddot{\text{a}}\text{x}\{\tilde{L}_i\})$ jest NP-trudny. Metoda postępowania będzie podobna do zastosowanej w punkcie 2.7. Najpierw definiujemy rozmyty problem szeregowania $1||E^N(\text{m}\ddot{\text{a}}\text{x}\{\tilde{L}_i\})$, w którym czasy trwania i pożądane terminy zakończenia prac są liczbami rozmytymi typu N -schodkowego, a funkcja kosztu harmonogramu jest określona następująco:

$$F(p) = E^N \left(\text{m}\ddot{\text{a}}\text{x}_{i=1, \dots, n} \{ \tilde{L}_i(p) \} \right).$$

Zakładamy również, że $N \leq n$ (wówczas rozmiar problemu jest ograniczony przez pewien wielomian zależny od n).

Można pokazać (w sposób analogiczny jak w twierdzeniu 2.6), że problem $1||E^N(\text{m}\ddot{\text{a}}\text{x}\{\tilde{L}_i\})$ jest szczególnym przypadkiem problemu $1||E(\text{m}\ddot{\text{a}}\text{x}\{\tilde{L}_i\})$. Udowodnimy teraz, że problem $1||E^N(\text{m}\ddot{\text{a}}\text{x}\{\tilde{L}_i\})$ jest NP-trudny. Dowód będzie podobny do dowodu NP-trudności problemu $1||E^N(\text{m}\ddot{\text{a}}\text{x}\{\tilde{T}_i\})$.

Twierdzenie 2.11. *Problem $1||E^N(\text{m}\ddot{\text{a}}\text{x}\{\tilde{L}_i\})$ jest NP-trudny, nawet jeżeli spośród wszystkich parametrów tylko pożądane terminy zakończenia prac są rozmyte.*

Dowód Pokażemy, że NP-trudny problem $1||ML$ jest T-transformowalny do problemu $1||E^N(\text{m}\ddot{\text{a}}\text{x}\{\tilde{L}_i\})$. Niech $((t_i)_{i=1}^n, (d_i)_{i=1}^n, n)$ będzie zadaną instancją problemu $1||ML$. Zdefiniujmy $c = \sum_{i=1}^n t_i$ i $d = \max_{i=1, \dots, n} \{d_i\} + c$. Przez $C_i(p)$, $L_i(p)$ będziemy oznaczać odpowiednio czas zakończenia oraz nieterminowość pracy i w permutacji p dla problemu $1||ML$.

Zdefiniujemy teraz odpowiednią instancję $(N, n, (\tilde{t}_i)_{i=1}^n, (\tilde{d}_i)_{i=1}^n)$ rozmytego problemu $1||E^N(\text{m}\ddot{\text{a}}\text{x}\{\tilde{L}_i\})$ w następujący sposób:

1. $N = n$,
2. $\tilde{t}_i$, $i = 1, \dots, n$, są liczbami rozmytymi, takimi że:

$$\mu_{\tilde{t}_i}(x) = \begin{cases} 1 & \text{dla } x = t_i, \\ 0 & \text{dla } x \neq t_i. \end{cases}$$

Stąd $\tilde{t}_i^\lambda = [t_i, t_i]$ dla $\lambda \in (0, 1]$ ($\tilde{t}_i$ są więc zdeterminowane).

3. $\tilde{d}_i$, $i = 1, \dots, n$, są liczbami rozmytymi, takimi że:

$$\mu_{\tilde{d}_i}(x) = \begin{cases} 1 & \text{dla } x = d, \\ \frac{i}{n} & \text{dla } x \in [d_i, d), \\ 0 & \text{dla } x \in (-\infty, d_i) \cup (d, \infty), \end{cases}$$

stąd:

$$\tilde{d}_i^\lambda = \begin{cases} [d_i, d] & \text{dla } \lambda \leq \frac{i}{n}, \\ [d, d] & \text{dla } \lambda > \frac{i}{n}. \end{cases} \quad (2.78)$$

Zauważmy, że $\tilde{t}_i$ oraz $\tilde{d}_i$, $i = 1, \dots, n$, są liczbami rozmytymi typu N -schodkowego. Stąd oraz z faktu, że $N = n$ wynika, że $(N, n, (\tilde{t}_i)_{i=1}^n, (\tilde{d}_i)_{i=1}^n)$ jest instancją problemu $1||FN|E^N(\text{m}\ddot{\text{a}}\text{x}\{\tilde{L}_i\})$.

Z definicji funkcji $E^N(\tilde{A})$ i formuły (2.72) otrzymujemy:

$$E^N(\tilde{L}(p)) = \frac{1}{2N} \sum_{j=1}^N \left(\max_{i=1, \dots, N} \{L_i(p, \frac{j}{N})\} + \max_{i=1, \dots, N} \{\bar{L}_i(p, \frac{j}{N})\} \right). \quad (2.79)$$

Dla każdego $i = 1, \dots, N$ i dla każdego $j = 1, \dots, N$ są spełnione równości:

$$L_i(p, \frac{j}{N}) = c_i(p, \frac{j}{N}) - \bar{d}_i(\frac{j}{N}) = C_i(p) - d, \quad (2.80)$$

$$\bar{L}_i(p, \frac{j}{N}) = \bar{c}_i(p, \frac{j}{N}) - \underline{d}_i(\frac{j}{N}) = C_i(p) - \underline{d}_i(\frac{j}{N}). \quad (2.81)$$

Z (2.78) otrzymujemy (analogicznie jak w dowodzie twierdzenia 2.8):

$$\underline{d}_i(\frac{j}{N}) = \begin{cases} d_i & \text{dla } 1 \leq j \leq i \leq N, \\ d & \text{dla } N \geq j > i \geq 1. \end{cases} \quad (2.82)$$

Z (2.81) i (2.82) wynika, że:

$$\bar{L}_i(p, \frac{j}{N}) = \begin{cases} C_i(p) - d_i = L_i(p) & \text{dla } 1 \leq j \leq i \leq N, \\ C_i(p) - d & \text{dla } N \geq j > i \geq 1. \end{cases} \quad (2.83)$$

Z (2.83) otrzymujemy:

$$\sum_{j=1}^N \max_{i=1, \dots, N} \{\bar{L}_i(p, \frac{j}{N})\} = \sum_{j=1}^N \max\{C_1(p) - d, \dots, C_{j-1}(p) - d, L_j(p), \dots, L_N(p)\}. \quad (2.84)$$

Z określenia wartości liczby d wynika, że dla każdego $i = 1, \dots, N$ zachodzi:

$$C_i(p) - d = C_i(p) - \sum_{j=1}^N t_j - \max_{j=1, \dots, N} \{d_j\} \leq C_N(p) - d_N = L_N(p). \quad (2.85)$$

Stąd oraz z równości (2.84) wynika, że:

$$\sum_{j=1}^N \max_{i=1, \dots, N} \{\bar{L}_i(p, \frac{j}{N})\} = \sum_{j=1}^N \max\{L_j(p), \dots, L_N(p)\} = ML(p). \quad (2.86)$$

Z warunku (2.80) otrzymujemy:

$$\sum_{j=1}^N \max_{i=1, \dots, N} \{L_i(p, \frac{j}{N})\} = \sum_{j=1}^N \max_{i=1, \dots, N} \{C_i(p) - d\} = N \cdot \max_{i=1, \dots, N} \{C_i(p) - d\}. \quad (2.87)$$

Ponieważ:

$$\max_{i=1,\dots,N} \{C_i(p) - d\} = \max_{i=1,\dots,N} \{C_i(p)\} - d = \sum_{j=1}^N t_j - d = c - d,$$

więc:

$$\sum_{j=1}^N \max_{i=1,\dots,N} \{\underline{L}_i(p, \frac{j}{N})\} = N(c - d). \quad (2.88)$$

Z warunków (2.79), (2.86) i (2.88) wynika ostatecznie, że:

$$E^N(\tilde{L}(p)) = \frac{1}{2N}(ML(p) + N(c - d)).$$

Ponieważ N , c i d są stałymi, to problem $1||ML$ i odpowiadający mu problem $1|FN|E^N(\max\{\tilde{L}_i\})$ mają takie same rozwiązania optymalne. Jeżeli istnieje wielomianowy algorytm dla problemu $1|FN|E^N(\max\{\tilde{L}_i\})$ to można łatwo skonstruować wielomianowy algorytm dla problemu $1||ML$. Wynika z tego, że problem $1|FN|E^N(\max\{\tilde{L}_i\})$ jest NP-trudny. Z określenia transformacji wynika, że problem ten jest NP-trudny nawet jeżeli spośród wszystkich parametrów tylko pożądane terminy zakończenia prac są rozmyte. $\square$

Twierdzenie 2.12. *Problem $1||E(\max\{\tilde{L}_i\})$ jest NP-trudny, nawet jeżeli spośród wszystkich parametrów jedynie pożądane terminy zakończenia prac są rozmyte.*

Dowód Ponieważ szczególny przypadek problemu $1||E(\max\{\tilde{L}_i\})$ jest NP-trudny (patrz tw. 2.11), to ogólniejszy problem jest również NP-trudny. $\square$

Na zakończenie tego punktu podamy kilka uwag dotyczących złożoności obliczeniowej problemów $1|FN|E(\max\{\tilde{T}_i\})$ i $1|FN|E(\max\{\tilde{L}_i\})$. Oba problemy są NP-trudne w przypadku gdy czasy trwania prac są zdeterminowane a pożądane terminy zakończenia prac są liczbami rozmytymi typu N -schodkowego. W dowodach NP-trudności wykorzystaliśmy więc liczby rozmyte, których funkcje przynależności są nieciągłe. Nie wiadomo jaka jest złożoność tych problemów w przypadku, gdy pożądane terminy zakończenia prac są liczbami rozmytymi typu $L - R$ (w szczególności liczbami trójkątnymi). Prawdopodobnie są one również NP-trudne a dowód tego faktu powinien być przedmiotem dalszych badań. Nie wiadomo również jaka jest złożoność obu problemów gdy pożądane terminy zakończenia prac są zdeterminowane a czasy trwania prac są liczbami rozmytymi.

Pewne komplikacje obliczeniowe wynikają również z faktu, że trudno jest obliczyć wartości $E(\tilde{T}(p))$ i $E(\tilde{L}(p))$ dla zadanej permutacji p . Najprostszym

rozwiązaniem jest obliczenie wartości przybliżonych $E^N(\tilde{T}(p))$ i $E^N(\tilde{L}(p))$, tym dokładniejszych im większa jest wartość N . Wyniki uzyskane w tym oraz poprzednim punkcie pokazują jednak, że zastosowanie takiej aproksymacji w przypadku, gdy pożądane terminy zakończenia prac są dowolnymi liczbami rozmytymi, nie zmniejsza złożoności obliczeniowej obu problemów (są one nadal NP-trudne).

Na rysunku 2.6 pokazane jest zestawienie problemów szeregowania, których NP-trudność została wykazana w tym punkcie oraz w punkcie poprzednim.

$$\begin{array}{ccccccc}
 1||\sum T_i & \rightarrow & 1||MT(\sigma, 1, n) & \rightarrow & 1|FN|E^N(\text{m}\ddot{\text{a}}\text{x}\{\tilde{T}_i\}) & \rightarrow & 1|FN|E(\text{m}\ddot{\text{a}}\text{x}\{\tilde{T}_i\}) \\
 & & \downarrow & & & & \\
 & & 1||ML & \rightarrow & 1|FN|E^N(\text{m}\ddot{\text{a}}\text{x}\{\tilde{L}_i\}) & \rightarrow & 1|FN|E(\text{m}\ddot{\text{a}}\text{x}\{\tilde{L}_i\})
 \end{array}$$

Rys. 2.6: Zestawienie NP-trudnych problemów szeregowania analizowanych w punktach 2.7 i 2.8. $P1 \rightarrow P2$ oznacza, że problem $P1$ jest T-transformowalny do problemu $P2$ ($P1$ może być również szczególnym przypadkiem $P2$)

2.9 Problem $1|prec, FN|Poss(\tilde{F} > \tilde{G})$

W punkcie 2.1 przedstawiony został przegląd klasycznych, zdeterminowanych problemów szeregowania na pojedynczej maszynie. Dla każdego takiego problemu definiuje się pewną funkcję kosztu harmonogramu $F(p)$, która zależy od czasów zakończenia prac i przyjmuje wartości rzeczywiste. Jeżeli parametry problemu są rozmyte, to niektóre funkcje kosztu harmonogramu mogą być bezpośrednio uogólnione na funkcje przyjmujące wartości rozmyte $\tilde{F}(p)$. Na przykład funkcja $F(p) = \sum_{i=1}^n w_i T_i(p)$ może zostać uogólniona na funkcję $\tilde{F}(p) = \sum_{i=1}^n w_i \tilde{T}_i(p)$. Jeżeli koszt harmonogramu jest rozmyty to pojęcie optymalności traci sens. Wynika to z faktu, że w zbiorze liczb rozmytych nie jest określona relacja porządku. Jednym z możliwych rozwiązań tego problemu jest minimalizacja wartości średniej rozmytego kosztu, czyli wartości funkcji $E(\tilde{F}(p))$. Podejście to zastosowane było w dwóch poprzednich punktach. Inne rozwiązanie polega na zastosowaniu podejścia, które nazywamy podejściem celowym. Decydent podaje dodatkową informację, która określa poziom kosztu, który nie powinien być przekroczony. Poziom ten jest zadany w postaci pewnej liczby rozmytej $\tilde{G}$, którą nazywamy celem decydenta. Harmonogram p jest tym lepszy im mniejsza jest możliwość, że koszt $\tilde{F}(p)$ przekroczy cel $\tilde{G}$. Jeżeli decydent potrafi dokładnie sformułować swój cel (np.: maksymalne opóźnienie prac powinno być niewiększe niż 10), to

liczba $\tilde{G}$ będzie zdeterminowana. Może się jednak zdarzyć, że cel jest określony nieprecyzyjnie (np.: maksymalne opóźnienie prac nie powinno być duże, przy czym przez duże rozumiemy wartości około 10). W takim przypadku cel może być zadany w postaci liczby rozmytej.

W punkcie tym zbadamy rozmyty problem szeregowania, dla którego funkcja kosztu harmonogramu jest postaci:

$$F(p) = Poss(\tilde{F}(p) > \tilde{G}). \quad (2.89)$$

Zakładamy dodatkowo, że podział prac oraz przestoje maszyny nie są dopuszczalne, co oznacza, że optymalnego harmonogramu wystarczy poszukiwać wśród permutacji zbioru prac. Wprowadźmy następujące oznaczenia:

$$(\tilde{F}(p))^\lambda = [\underline{f}(p, \lambda), \overline{f}(p, \lambda)],$$

$$\tilde{G}^\lambda = [\underline{g}(\lambda), \overline{g}(\lambda)].$$

Lemat 2.14. *Założmy, że dla każdej permutacji p funkcje $\overline{f}(p, \lambda)$ i $\overline{g}(\lambda)$ są ciągłymi i ściśle malejącymi funkcjami λ dla $\lambda \in (0, 1)$. Wówczas dla każdego p i dla każdego $\lambda \in (0, 1)$ zachodzą następujące warunki:*

$$Poss(\tilde{F}(p) > \tilde{G}) = \lambda \Leftrightarrow \overline{f}(p, \lambda) = \overline{g}(1 - \lambda),$$

$$Poss(\tilde{F}(p) > \tilde{G}) < \lambda \Leftrightarrow \overline{f}(p, \lambda) < \overline{g}(1 - \lambda).$$

Dowód Wprost z lematu 1.9. □

Lemat 2.15. *Jeżeli dla pewnego $\lambda_0 \in (0, 1)$ warunek $\overline{f}(p, \lambda) \leq \overline{g}(1 - \lambda)$ jest sprzeczny (tzn. nie istnieje p dla którego $\overline{f}(p, \lambda_0) \leq \overline{g}(1 - \lambda_0)$), to warunek ten jest również sprzeczny dla każdego $\lambda \leq \lambda_0$.*

Dowód Z definicji λ -poziomu liczby rozmytej otrzymujemy dla każdego p :

$$\lambda \leq \lambda_0 \Rightarrow (\tilde{F}(p))^{\lambda_0} \subseteq (\tilde{F}(p))^\lambda \Rightarrow \overline{f}(p, \lambda_0) \leq \overline{f}(p, \lambda),$$

$$\lambda \leq \lambda_0 \Rightarrow 1 - \lambda \geq 1 - \lambda_0 \Rightarrow \tilde{G}^{1-\lambda} \subseteq \tilde{G}^{1-\lambda_0} \Rightarrow \overline{g}(1 - \lambda_0) \geq \overline{g}(1 - \lambda).$$

Stąd wynika, że dla każdego p jest prawdziwa następująca implikacja:

$$\overline{f}(p, \lambda_0) > \overline{g}(1 - \lambda_0) \Rightarrow \forall_{\lambda \leq \lambda_0} \overline{f}(p, \lambda) > \overline{g}(1 - \lambda),$$

czyli sprzeczność warunku $\bar{f}(p, \lambda) \leq \bar{g}(1 - \lambda)$ dla λ_0 pociąga za sobą sprzeczność tego warunku dla każdego $\lambda \leq \lambda_0$. $\square$

Zdefiniujemy teraz funkcję $EXISTS(\lambda)$, $\lambda \in (0, 1)$, określoną następująco:

$$EXISTS(\lambda) = \begin{cases} p^* & \text{jeżeli } \bar{f}(p^*, \lambda) \leq \bar{g}(1 - \lambda), \\ () & \text{jeżeli } \forall_p \bar{f}(p, \lambda) > \bar{g}(1 - \lambda). \end{cases}$$

Funkcja $EXISTS(\lambda)$ konstruuje dla zadanego $\lambda \in (0, 1)$ permutację p^* spełniającą warunek $\bar{f}(p^*, \lambda) \leq \bar{g}(1 - \lambda)$ lub zwraca pustą permutację jeżeli żadna permutacja nie spełnia tego warunku. Jeżeli funkcje $\bar{f}$ i $\bar{g}$ są ciągłymi i ściśle malejącymi funkcjami λ , to z lematów 2.14 i 2.15 wynika, że funkcja $EXISTS(\lambda)$ konstruuje permutację o koszcie niewiększym niż λ lub stwierdza, że permutacja o koszcie niewiększym niż λ nie istnieje.

Problem $1|prec, FN|Poss(\tilde{F} > \tilde{G})$ można rozwiązać za pomocą algorytmu 10, w którym korzystamy z funkcji $EXISTS(\lambda)$.

Algorytm 10 Algorytm dla problemu $1|prec, FN|Poss(\tilde{F} > \tilde{G})$

Require: $n, (\tilde{t}_i)_{i=1}^n, (\tilde{d}_i)_{i=1}^n, (w_i)_{i=1}^n, \tilde{G}, \varepsilon \in (0, 1]$

Ensure: p^* /* optymalne rozwiązanie problemu */

```

1:  $p^* \leftarrow EXISTS(\varepsilon)$ 
2: if  $p^* \neq ()$  then
3:   return /*  $p^*$  jest optymalnym rozwiązaniem o koszcie  $\varepsilon$  */
4: end if
5:  $p^* \leftarrow EXISTS(1 - \varepsilon)$ 
6: if  $p^* = ()$  then
7:    $p^* \leftarrow (1, \dots, n)$  /* lub dowolną inną permutację */
8:   return  $p^*$  /* koszt każdej permutacji jest większy niż  $1 - \varepsilon$  */
9: end if
10:  $\lambda_1 \leftarrow 0.5; \lambda_2 \leftarrow 0; k \leftarrow 1$ 
11: while  $|\lambda_2 - \lambda_1| \geq \varepsilon$  do
12:    $\lambda_2 \leftarrow \lambda_1$ 
13:   if  $EXISTS(\lambda_1) = ()$  then
14:      $\lambda_1 \leftarrow \lambda_1 + \frac{1}{2^{k+1}}$ 
15:   else
16:      $p^* \leftarrow EXISTS(\lambda_1)$ 
17:      $\lambda_1 \leftarrow \lambda_1 - \frac{1}{2^{k+1}}$ 
18:   end if
19:    $k \leftarrow k + 1$ 
20: end while
21: return  $p^*$  /*  $p^*$  jest optymalnym rozwiązaniem o koszcie  $\lambda_1$  */
```

Poprawność algorytmu 10 jest konsekwencją następującego twierdzenia:

Twierdzenie 2.13. *Jeżeli dla każdego p funkcje $\bar{f}(p, \lambda)$ i $\bar{g}(\lambda)$ są ciągłymi i ściśle malejącymi funkcjami λ dla $\lambda \in (0, 1)$, to algorytm 10 konstruuje permutację p^* , której koszt różni się od kosztu optymalnej permutacji co najwyżej o ε .*

Dowód Z własności indeksu $Poss(\tilde{A} > \tilde{B})$ wynika, że koszt każdej permutacji należy do przedziału $[0, 1]$. Rozpatrzmy trzy możliwe przypadki, które mogą zajść podczas działania algorytmu 10:

1. Jeżeli istnieje permutacja p taka, że $Poss(\tilde{F}(p) > \tilde{G}) \in [0, \varepsilon]$, to algorytm skonstruuje tą permutację w kroku 1 i zakończy pracę. Z definicji funkcji $EXISTS(\lambda)$ zachodzi bowiem warunek $\bar{f}(p, \varepsilon) \leq \bar{g}(1 - \varepsilon)$ i wobec lematu 2.14 koszt p jest nie większy niż ε . Jest oczywiste, że koszt tej permutacji różni się od kosztu optymalnej permutacji co najwyżej o ε .
2. Jeżeli dla każdej permutacji p zachodzi $Poss(\tilde{F}(p) > \tilde{G}) \in (1 - \varepsilon, 1]$, to algorytm stwierdzi ten fakt w kroku 5 i zakończy pracę. Jeżeli bowiem $EXISTS(1 - \varepsilon) = ()$, to z określenia funkcji $EXISTS(\lambda)$ i z lematu 2.15 wynika, że dla każdej permutacji p zachodzi $Poss(\tilde{F}(p) > \tilde{G}) > 1 - \varepsilon$. Wynika z tego, że koszt dowolnej permutacji różni się od kosztu optymalnego rozwiązania o nie więcej niż ε . W takim przypadku algorytm przyjmie za rozwiązanie permutację $p = (1, \dots, n)$ i zakończy pracę.
3. Jeżeli algorytm nie zakończył pracy w krokach od 1 do 9, to oznacza, że koszt optymalnej permutacji należy do przedziału $(\varepsilon, 1 - \varepsilon]$. Wtedy, po każdej iteracji pętli 11-20 błąd bezwzględny w oszacowaniu kosztu optymalnej permutacji jest mniejszy niż $\frac{1}{2^k}$. Wynika to z faktu, że korzystając z lematów 2.14 i 2.15, zawężamy w każdym kroku przedział, w którym znajduje się optymalna wartość kosztu o połowę. Po ostatniej iteracji $k \geq \lceil -\log_2 \varepsilon \rceil + 1$ i różnica między kosztem optymalnej permutacji i kosztem permutacji p^* jest nie większa niż ε . $\square$

Złożoność obliczeniowa algorytmu 10 jest rzędu $O(|\log \varepsilon| f(n))$, gdzie $f(n)$ jest złożonością obliczeniową funkcji $EXISTS(\lambda)$, przy założeniu, że zależy ona wyłącznie od liczby prac n . Zauważmy, że obliczenie funkcji $EXISTS(\lambda)$ wymaga rozwiązania pewnego problemu decyzyjnego, który może być trudny (jego złożoność zależy od wyboru konkretnej funkcji $\tilde{F}(p)$).

W kolejnych punktach przeanalizujemy dwa szczególne przypadki problemu $1|prec, FN|Poss(\tilde{F} > \tilde{G})$, które są łatwe z punktu widzenia złożoności obliczeniowej.

2.9.1 Problem $1|prec, L - R|Poss(\max\{w_i\tilde{L}_i\} > \tilde{G})$

Założmy, że czasy trwania prac i pożądane terminy zakończenia prac oraz cel $\tilde{G}$ są liczbami rozmytymi typu $L - R$, gdzie:

$$\tilde{t}_i = (\underline{t}_i, \bar{t}_i, \alpha_i, \beta_i)_{L_i - R_i}, \quad i = 1, \dots, n,$$

$$\tilde{d}_i = (\underline{d}_i, \bar{d}_i, \phi_i, \psi_i)_{S_i - T_i}, \quad i = 1, \dots, n.$$

Funkcja kosztu harmonogramu jest określona w następujący sposób:

$$F(p) = Poss(\max_{i=1, \dots, n} \{w_i \tilde{L}_i(p)\} > \tilde{G}).$$

Szukamy więc takiego harmonogramu p , dla którego możliwość, że maksymalna ważona nieterminowość przekroczy zadany cel $\tilde{G}$ jest najmniejsza. Zauważmy najpierw, że z wniosku 2.1 wynika, że funkcja $F(p)$ jest f -regularna czyli problem, w którym dodatkowo jest dopuszczalny podział prac i przestoje maszyny ma rozwiązanie optymalne o takim samym koszcie.

Funkcja $\tilde{F}(p)$ w formule (2.89) przyjmuje w analizowanym przypadku następującą postać:

$$\tilde{F}(p) = \max_{i=1, \dots, n} \{w_i \tilde{L}_i(p)\}.$$

Korzystając z własności λ -poziomu (patrz własność 1.5), możemy wyznaczyć funkcję $\bar{f}(p, \lambda)$:

$$\bar{f}(p, \lambda) = \max_{i=1, \dots, n} \{w_i(\bar{c}_i(p, \lambda) - \underline{d}_i(\lambda))\} = \max_{i=1, \dots, n} \{w_i(\sum_{j \in p^i} \bar{t}_j(\lambda) - \underline{d}_i(\lambda))\}. \quad (2.90)$$

Lemat 2.16. *Jeżeli parametry $\tilde{t}_i, \tilde{d}_i, i = 1, \dots, n$ są liczbami rozmytymi typu $L - R$, to funkcja $\bar{f}(p, \lambda)$ jest dla każdego p ciągłą i ściśle malejącą funkcją λ dla $\lambda \in (0, 1)$.*

Dowód Ponieważ wszystkie parametry $\tilde{t}_i, \tilde{d}_i, i = 1, \dots, n$ są liczbami rozmytymi typu $L - R$, więc z własności 1.2 otrzymujemy, że funkcje $\bar{t}_i(\lambda)$ i $\underline{d}_i(\lambda)$ są ciągłe dla $\lambda \in (0, 1)$. Z elementarnej analizy wiadomo, że suma, różnica oraz maksimum funkcji ciągłych jest funkcją ciągłą. Z określenia funkcji $\bar{f}(p, \lambda)$ (patrz formuła (2.90)) wynika więc, że funkcja ta jest dla każdej permutacji p ciągłą funkcją λ .

Z własności 1.2 otrzymujemy również, że funkcje $\underline{d}_i(\lambda)$ są ściśle rosnące a funkcje $\bar{t}_i(\lambda)$ są ściśle malejące dla $\lambda \in (0, 1)$. Stąd jeżeli $\lambda_1 > \lambda_2$, to dla każdego $i \in J$ zachodzi:

$$w_i(\bar{c}_i(p, \lambda_1) - \underline{d}_i(\lambda_1)) < w_i(\bar{c}_i(p, \lambda_2) - \underline{d}_i(\lambda_2)),$$

czyli:

$$\max_{i=1,\dots,n} \{w_i(\bar{c}_i(p, \lambda_1) - \underline{d}_i(\lambda_1))\} < \max_{i=1,\dots,n} \{w_i(\bar{c}_i(p, \lambda_2) - \underline{d}_i(\lambda_2))\}.$$

Stąd $\bar{f}(p, \lambda_1) < \bar{f}(p, \lambda_2)$ i funkcja $\bar{f}(p, \lambda)$ jest ściśle malejącą funkcją λ . $\square$

Z lematu 2.16 wynika, że do rozwiązania badanego w tym punkcie problemu możemy zastosować algorytm 10. Podamy teraz sposób obliczenia funkcji $EXISTS(\lambda)$ dla tego problemu. Dla zadanego $\lambda \in (0, 1)$ zdefiniujmy klasyczny problem $1|prec|\max\{w_i L_i\}$ z parametrami: $t_i = \bar{t}_i(\lambda)$, $d_i = \underline{d}_i(\lambda)$, $prec$, w_i , $i = 1, \dots, n$. Mamy wówczas:

$$\max_{i=1,\dots,n} \{w_i L_i(p)\} = \max_{i=1,\dots,n} \{w_i(\sum_{j \in p^i} \bar{t}_j(\lambda) - \underline{d}_i(\lambda))\} = \bar{f}(p, \lambda).$$

Aby stwierdzić, czy dla zadanego λ istnieje p takie, że $\bar{f}(p, \lambda) \leq \bar{g}(1 - \lambda)$ należy wyznaczyć optymalne rozwiązanie problemu $1|prec|\max\{w_i L_i\}$ dla zadanego λ i sprawdzić, czy koszt tego rozwiązania jest nie większy niż $\bar{g}(1 - \lambda)$. Problem $1|prec|\max\{w_i L_i\}$ można rozwiązać za pomocą klasycznego algorytmu Lawlera (patrz algorytm 3). Funkcja $EXISTS(\lambda)$ jest zapisana w postaci algorytmu 11.

Algorytm 11 Funkcja $EXISTS(\lambda)$ dla $1|prec, L - R|Poss(\max\{w_i \tilde{L}_i\} > \tilde{G})$

Require: $n, prec, (\tilde{t}_i)_{i=1}^n, (\tilde{d}_i)_{i=1}^n, (w_i)_{i=1}^n, \tilde{G}, \lambda$

Ensure: p^*

```

1: for all  $i \in J$  do
2:    $t_i \leftarrow \bar{t}_i + \beta_i R_i^{-1}(\lambda)$ 
3:    $d_i \leftarrow \underline{d}_i - \phi_i S_i^{-1}(\lambda)$ 
4: end for
5: Rozwiąż klasyczny problem  $1|prec|\max\{w_i L_i\}$  z parametrami
    $n, prec, (t_i)_{i=1}^n, (d_i)_{i=1}^n, (w_i)_{i=1}^n$ , stosując algorytm Lawlera (patrz algorytm 3)
   i podstaw pod  $p^*$  optymalną permutację.
6: if  $\max_{i=1,\dots,n} \{w_i(C_i(p^*) - d_i)\} \leq \bar{g}(1 - \lambda)$  then
7:   return  $p^* / * \bar{f}(p^*, \lambda) \leq \bar{g}(1 - \lambda) * /$ 
8: else
9:   return  $() / * \forall_p \bar{f}(p, \lambda) > \bar{g}(1 - \lambda) * /$ 
10: end if
```

Złożoność algorytmu 11 jest taka sama jak złożoność klasycznego algorytmu Lawlera i jest rzędu $O(n^2)$. Dysponując algorytmami 10 i 11 możemy efektywnie rozwiązać problem $1|prec, L - R|Poss(\max\{w_i \tilde{L}_i\} > \tilde{G})$. Złożoność pełnego algorytmu dla tego problemu jest rzędu $O(n^2 |\log \varepsilon|)$, gdzie ε jest zadaną dokładnością w wyznaczeniu kosztu optymalnej permutacji.

Przykład: Niech $J = \{1, \dots, 10\}$, $prec = \{(1, 2), (3, 4), (2, 5), (3, 8)\}$ a $\tilde{t}_i$ oraz $\tilde{d}_i$, $i = 1, \dots, 10$, będą liczbami trapezowymi. Wszystkie parametry problemu są podane w tablicy 2.8. Załóżmy, że cel jest również trapezową liczbą rozmytą, $\tilde{G} = (10, 12, 5, 5)$, a $\varepsilon = 0.001$.

$i \in J$	$\tilde{t}_i$	$\tilde{d}_i$	w_i
1	(1 1 1 3)	(10 22 3 8)	0.7
2	(2 5 1 3)	(10 12 3 5)	0.6
3	(3 6 2 3)	(11 13 3 9)	0.2
4	(3 5 2 2)	(10 11 5 9)	0.2
5	(2 2 1 2)	(10 11 5 5)	0.4
6	(2 3 1 2)	(12 12 6 10)	0.6
7	(4 5 2 2)	(8 9 7 10)	0.5
8	(2 2 1 2)	(10 11 3 5)	0.8
9	(3 4 3 2)	(8 9 8 8)	0.1
10	(2 2 1 1)	(11 15 6 8)	0.5

Tab. 2.8: Parametry przykładowego problemu

Przebieg algorytmu jest przedstawiony w tablicy 2.9.

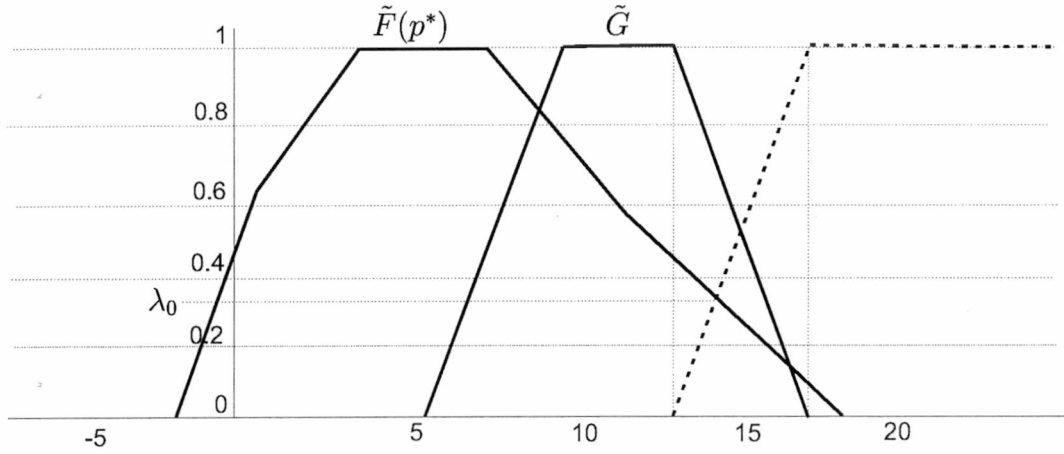
krok	λ	$\bar{g}(1 - \lambda)$	$\bar{f}(p^*, \lambda)$	$EXISTS(\lambda)$
1	0.001	12.005	17.389	()
2	0.999	16.995	6.510	(7,3,8,1,2,6,10,5,4,9)
3	0.5	14.5	12	(3,8,1,7,2,6,10,5,4,9)
4	0.25	13.25	14.85	()
5	0.375	13.875	13.5	(3,8,1,7,2,6,10,5,4,9)
6	0.312	13.562	14.212	()
7	0.343	13.718	13.875	()
8	0.359	13.796	13.687	(3,8,1,7,2,6,10,5,4,9)
9	0.351	13.757	13.781	()
11	0.355	13.777	13.734	(3,8,1,7,2,6,10,5,4,9)
12	0.353	13.767	13.757	(3,8,1,7,2,6,10,5,4,9)

Tab. 2.9: Poszczególne kroki algorytmu dla przykładowego problemu

Optymalnym rozwiązaniem problemu jest permutacja

$$p^* = (3, 8, 1, 7, 2, 6, 10, 5, 4, 9),$$

o koszcie $F(p) = 0.353$. Funkcje przynależności liczby rozmytej $\tilde{F}(p^*)$ i rozmytego celu $\tilde{G}$ są przedstawione na rysunku 2.7.



Rys. 2.7: $Poss(\tilde{F}(p^*) > \tilde{G}) = Poss(\max_{i=1, \dots, n} \{w_i \tilde{L}_i(p^*)\} > \tilde{G}) = \lambda_0$

2.9.2 Problem $1|L - R|Poss(\sum w_i \tilde{C}_i > \tilde{G})$

Założmy, że $prec = \emptyset$ a czasy trwania prac oraz cel $\tilde{G}$ są liczbami rozmytymi typu $L - R$, gdzie:

$$\tilde{t}_i = (\underline{t}_i, \bar{t}_i, \alpha_i, \beta_i)_{L_i - R_i}, \quad i = 1, \dots, n.$$

Funkcja kosztu harmonogramu jest określona następująco:

$$F(p) = Poss(\sum_{i=1}^n w_i \tilde{C}_i(p) > \tilde{G}).$$

Z wniosku 2.1 wynika, że funkcja ta jest f -regularna, czyli problem w którym dodatkowo dopuszczalny jest podział prac i przestoje maszyny ma rozwiązanie optymalne o takim samym koszcie. Funkcja $\tilde{F}(p)$ w formule (2.89) przyjmuje w analizowanym przypadku następującą postać:

$$\tilde{F}(p) = \sum_{i=1}^n w_i \tilde{C}_i(p).$$

Korzystając z własności λ -poziomu, możemy wyznaczyć funkcję $\bar{f}(p, \lambda)$:

$$\bar{f}(p, \lambda) = \sum_{i=1}^n w_i \bar{c}_i(p, \lambda) = \sum_{i=1}^n (w_i \sum_{j \in p^i} \bar{t}_j(\lambda)). \quad (2.91)$$

Lemat 2.17. *Jeżeli wszystkie parametry $\tilde{t}_i, \tilde{d}_i, i = 1, \dots, n$ są liczbami rozmytymi typu $L-R$, to funkcja $\bar{f}(p, \lambda)$ jest dla każdego p ciągłą i ściśle malejącą funkcją λ dla $\lambda \in (0, 1)$.*

Dowód Analogiczny do dowodu lematu 2.16. $\square$

Podamy teraz sposób obliczenia funkcji $EXISTS(\lambda)$ dla tego problemu. Dla zadanego $\lambda \in (0, 1)$ utwórzmy klasyczny problem szeregowania $1||\sum w_i C_i$ z parametrami: $t_i = \bar{t}_i(\lambda)$, w_i , $i = 1, \dots, n$. Mamy wówczas:

$$\sum_{i=1}^n w_i C_i(p) = \sum_{i=1}^n (w_i \sum_{j \in p^i} \bar{t}_j(\lambda)) = \bar{f}(p, \lambda). \quad (2.92)$$

Aby stwierdzić, czy dla zadanego λ istnieje p takie, że $\bar{f}(p, \lambda) \leq \bar{g}(1 - \lambda)$ należy znaleźć optymalne rozwiązanie problemu $1||\sum w_i C_i$ dla zadanego λ i sprawdzić, czy koszt tego rozwiązania jest nie większy $\bar{g}(1 - \lambda)$. Optymalne rozwiązanie problemu $1||\sum w_i C_i$ otrzymujemy stosując tzw. regułę Smitha (patrz [3]), czyli wykonując prace zgodnie z nie malejącymi wartościami $\frac{t_i}{w_i}$.

Korzystając z reguły Smitha skonstruujemy algorytm 12 obliczający funkcję $EXISTS(\lambda)$ dla badanego problemu.

Algorytm 12 Funkcja $EXISTS(\lambda)$ dla $1|L - R|Poss(\sum w_i \tilde{C}_i > \tilde{G})$

Require: $n, (\tilde{t}_i)_{i=1}^n, (w_i)_{i=1}^n, \tilde{G}, \lambda$

Ensure: p^*

- 1: **for all** $i \in J$ **do**
 - 2: $t_i \leftarrow \tilde{t}_i + \beta_i R_i^{-1}(\lambda)$
 - 3: **end for**
 - 4: Utwórz permutację p^* wykonując prace w kolejności niemalejących wartości t_i/w_i (reguła Smitha)
 - 5: **if** $\sum_{i=1}^n w_i C_i(p^*) \leq \bar{g}(1 - \lambda)$ **then**
 - 6: **return** p^* / * $\bar{f}(p^*, \lambda) \leq \bar{g}(1 - \lambda)$ * /
 - 7: **else**
 - 8: **return** () / * $\forall_p \bar{f}(p, \lambda) > \bar{g}(1 - \lambda)$ * /
 - 9: **end if**
-

Złożoność algorytmu 12 jest rzędu $O(n \log n)$. Dysponując algorytmami 10 i 12 możemy efektywnie rozwiązać problem $1|prec, L - R|Poss(\sum w_i \tilde{C}_i > \tilde{G})$. Złożoność pełnego algorytmu dla tego problemu jest rzędu $O(n \log n |\log \varepsilon|)$, gdzie ε jest zadaną dokładnością wyznaczenia kosztu optymalnej permutacji.

Przykład: Niech $J = \{1, \dots, 10\}$ a $\tilde{t}_i$ oraz $\tilde{d}_i$, $i = 1, \dots, n$, będą liczbami trapezowymi. Wszystkie parametry problemu podane są w tablicy 2.10. Założmy, że cel jest również trapezową liczbą rozmytą $\tilde{G} = (60, 70, 20, 20)$ i $\varepsilon = 0.001$.

Przebieg algorytmu przedstawiony jest w tablicy 2.11.

$i \in J$	$\tilde{t}_i$	w_i
1	(1 1 1 3)	0.7
2	(2 5 1 3)	0.6
3	(3 6 2 3)	0.2
4	(3 5 2 2)	0.2
5	(2 2 1 2)	0.4
6	(2 3 1 2)	0.6
7	(4 5 2 2)	0.5
8	(2 2 1 2)	0.8
9	(3 4 3 2)	0.1
10	(2 2 1 1)	0.5

Tab. 2.10: Parametry przykładowego problemu

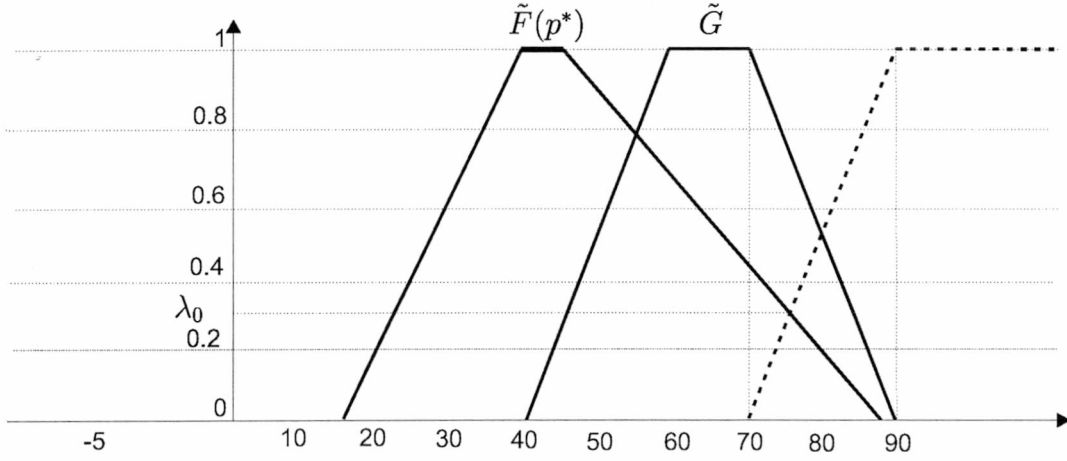
krok	λ	$\bar{g}(1 - \lambda)$	$\bar{f}(p^*, \lambda)$	$EXISTS(\lambda)$
1	0.001	70.02	88.859	()
2	0.999	89.98	48.141	(1,8,10,6,5,2,7,4,3,9)
3	0.5	80	68.7	(1,8,10,6,5,2,7,4,3,9)
4	0.25	75	78.85	()
5	0.375	77.5	73.825	(8,1,10,6,5,2,7,4,3,9)
6	0.312	76.25	76.337	()
7	0.343	76.875	75.081	(8,1,10,6,5,2,7,4,3,9)
8	0.328	76.562	75.709	(8,1,10,6,5,2,7,4,3,9)
9	0.320	76.406	76.023	(8,1,10,6,5,2,7,4,3,9)
11	0.316	76.328	76.180	(8,1,10,6,5,2,7,4,3,9)
12	0.314	76.289	76.259	(8,1,10,6,5,2,7,4,3,9)

Tab. 2.11: Poszczególne kroki algorytmu dla przykładowego problemu

Optymalnym rozwiązaniem jest permutacja $p^* = (8, 1, 10, 6, 5, 2, 7, 4, 3, 9)$ o koszcie $F(p^*) = 0.314$. Dla tej permutacji:

$$\tilde{F}(p^*) = \sum_{i=1}^n w_i \tilde{C}_i(p^*) = (38.3, 48.7, 21.6, 40.2).$$

Funkcje przynależności liczb rozmytych $\sum_{i=1}^n w_i \tilde{C}_i(p^*)$ i $\tilde{G}$ są przedstawione na rysunku 2.8.



Rys. 2.8: $Poss(\tilde{F}(p^*) > \tilde{G}) = Poss(\sum_{i=1}^n w_i \tilde{C}_i(p^*) > \tilde{G}) = \lambda_0$

2.10 Problem $1|FN| \sum Poss(Ness)(\tilde{C}_i > \tilde{d}_i)$

W punkcie tym zbadamy rozmyty problem szeregowania, w którym minimalizowana jest suma możliwości (lub konieczności) przekroczenia pożądanego terminu zakończenia przez prac należących do J .

Założmy, że nie jest określona relacja poprzedzania ($prec = \emptyset$) a funkcja kosztu harmonogramu jest zdefiniowana następująco:

$$F(p) = \sum_{i=1}^n D_i(\tilde{C}_i(p) > \tilde{d}_i),$$

gdzie $D_i \in \{Poss, Ness\}$ dla $i = 1, \dots, n$.

Z wniosku 2.1 wynika, że funkcja $F(p)$ jest f -regularna, czyli dopuszczenie podziału prac i przestojów maszyny w tym problemie nie ma wpływu na koszt uzyskanego optymalnego rozwiązania.

Zauważmy, że jeżeli parametry $\tilde{t}_i, \tilde{d}_i, i = 1, \dots, n$, są zdeterminowane, to:

$$Poss(Ness)(C_i(p) > d_i) = U_i(p) = \begin{cases} 1 & \text{jeżeli } C_i(p) > d_i, \\ 0 & \text{jeżeli } C_i(p) \leq d_i. \end{cases}$$

W takim przypadku problem $1|FN| \sum Poss(Ness)(\tilde{C}_i > \tilde{d}_i)$ jest równoważny klasycznemu problemowi $1|| \sum U_i$ (patrz punkt 2.1) i może być efektywnie rozwiązany przez algorytm wielomianowy (patrz [3]). Problem staje się trudniejszy w ogólniejszym przypadku, gdy parametry są liczbami rozmytymi. Udowodnimy następujące twierdzenie:

Twierdzenie 2.14. *Problem $1|FN|\sum Poss(\tilde{C}_i > \tilde{d}_i)$ jest NP-trudny nawet jeżeli spośród wszystkich parametrów jedynie pożądane terminy zakończenia prac $\tilde{d}_i$, $i = 1, \dots, n$, są rozmyte.*

Dowód Pokażemy, że NP-trudny problem $1||\sum T_i$ jest T-transformowalny do $1|FN|\sum Poss(\tilde{C}_i > \tilde{d}_i)$. Niech $(n, (t_i)_{i=1}^n, (d_i)_{i=1}^n)$ będzie zadaną instancją problemu $1||\sum T_i$. Przez $C_i(p)$, $T_i(p)$ będziemy oznaczać odpowiednio czas zakończenia i opóźnienie pracy i w permutacji p dla problemu $1||\sum T_i$. Zdefiniujmy $D = \sum_{i=1}^n t_i + 1$. Skonstruujmy odpowiednią instancję $(n, (\tilde{t}_i)_{i=1}^n, (\tilde{d}_i)_{i=1}^n)$ rozmytego problemu $1|FN|\sum Poss(\tilde{C}_i > \tilde{d}_i)$ w następujący sposób:

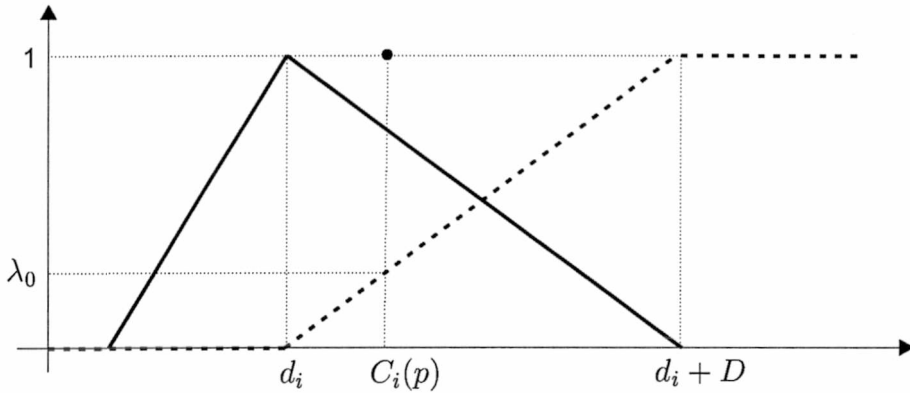
1. $\tilde{t}_i = t_i$, $i = 1, \dots, n$, ($\tilde{t}_i$ są więc zdeterminowane),
2. $\tilde{d}_i = (d_i, D, D)$, $i = 1, \dots, n$.

Ponieważ czasy trwania prac są zdeterminowane, więc czasy zakończenia prac $i \in J$ w harmonogramie p są również zdeterminowane, czyli:

$$\tilde{C}_i(p) = C_i(p).$$

Zachodzi ponadto następująca zależność (patrz lemat 1.1 i rysunek 2.9):

$$Poss(C_i(p) > \tilde{d}_i) = \begin{cases} 1 & C_i(p) - d_i \geq D, \\ \frac{C_i(p) - d_i}{D} & C_i(p) - d_i \in (0, D), \\ 0 & C_i(p) - d_i \leq 0. \end{cases} \quad (2.93)$$



Rys. 2.9: $Poss(C_i(p) > \tilde{d}_i) = \lambda_0$

Ponieważ dla każdego $i \in J$ zachodzi $C_i(p) - d_i < D$, więc wzór (2.93) można zapisać w jeszcze prostszej postaci:

$$Poss(C_i(p) > \tilde{d}_i) = \begin{cases} \frac{C_i(p) - d_i}{D} & C_i(p) - d_i > 0, \\ 0 & C_i(p) - d_i \leq 0. \end{cases} \quad (2.94)$$

Stąd:

$$\begin{aligned} Poss(C_i(p) > \tilde{d}_i) &= \frac{\max\{0, C_i(p) - d_i\}}{D} = \frac{T_i(p)}{D}, \\ \sum_{i=1}^n Poss(C_i(p) > \tilde{d}_i) &= \frac{1}{D} \sum_{i=1}^n T_i(p), \\ \sum_{i=1}^n T_i(p) &= D \sum_{i=1}^n Poss(C_i(p) > \tilde{d}_i). \end{aligned}$$

Z ostatniej równości oraz z faktu, że $D > 0$ jest stałą wynika, że problem $1|| \sum T_i$ oraz odpowiedni problem $1|FN| \sum Poss(\tilde{C}_i > \tilde{d}_i)$ mają takie same rozwiązania optymalne.

Przedstawiona transformacja może być łatwo wykonana przez algorytm wielomianowy. Gdyby istniał wielomianowy algorytm dla $1|FN| \sum Poss(\tilde{C}_i > \tilde{d}_i)$, to można byłoby rozwiązać w czasie wielomianowym NP-trudny problem $1|| \sum T_i$. Wynika z tego, że problem $1|FN| \sum Poss(\tilde{C}_i > \tilde{d}_i)$ jest NP-trudny. Ponadto z przeprowadzonej transformacji wynika, że problem ten jest NP-trudny jeżeli spośród wszystkich parametrów jedynie pożądane terminy zakończenia prac są rozmyte. $\square$

Twierdzenie 2.15. *Problem $1|FN| \sum Ness(\tilde{C}_i > \tilde{d}_i)$ jest NP-trudny nawet jeżeli spośród wszystkich parametrów jedynie pożądane terminy zakończenia prac $\tilde{d}_i$, $i = 1, \dots, n$, są rozmyte.*

Dowód Analogiczny do dowodu twierdzenia 2.14. $\square$

2.11 Problemy $1|prec, L - R| \sum E(w_i \tilde{C}_i)$ i $1|TG| \sum E(w_i \tilde{T}_i)$

W punkcie tym zbadamy rozmyte problemy szeregowania, w których funkcja kosztu harmonogramu jest szczególnym przypadkiem funkcji postaci:

$$F(p) = \sum_{i=1}^n E(\tilde{f}_i(\tilde{C}_i(p))),$$

gdzie $\tilde{f}_i : FN(\mathbb{R}) \rightarrow FN(\mathbb{R})$. Zakładamy, że dla każdego harmonogramu p istnieje wartość średnia liczby rozmytej $\tilde{f}_i(\tilde{C}_i(p))$, $i = 1, \dots, n$.

Rozważmy najpierw problem $1|prec, L - R| \sum E(w_i \tilde{C}_i)$, w którym funkcja kosztu harmonogramu jest określona następująco:

$$F(p) = \sum_{i=1}^n E(w_i \tilde{C}_i(p)).$$

Założmy, że czasy trwania prac są liczbami rozmytymi typu $L - R$, dla których istnieje wartość średnia. Z własności wartości średniej liczby rozmytej (patrz własność 1.7) otrzymujemy następującą zależność:

$$\sum_{i=1}^n E(w_i \tilde{C}_i(p)) = \sum_{i=1}^n w_i E(\tilde{C}_i(p)) = \sum_{i=1}^n w_i \sum_{j \in p^i} E(\tilde{t}_j).$$

Z zależności tej wynika, że analizowany problem jest równoważny klasycznemu problemowi $1|prec| \sum w_i C_i$, w którym rozmyte czasy trwania prac są zastąpione ich wartościami średnimi. Odpowiednie wartości średnie rozmytych czasów można łatwo wyznaczyć za pomocą formuły podanej w lemacie 1.12. Złożoność problemu $1|prec, L - R| \sum E(w_i \tilde{C}_i)$ jest więc taka sama jak złożoność analogicznego problemu $1|prec| \sum w_i C_i$. Jeżeli w problemie $1|prec, L - R| \sum E(w_i \tilde{C}_i)$ relacja poprzedzania $prec = \emptyset$, to optymalny harmonogram (permutację prac) otrzymujemy wykonując prace zgodnie z regułą Smitha, czyli zgodnie z niemalejącymi wartościami $\frac{E(\tilde{t}_i)}{w_i}$ (patrz [3]). Problem jest również wielomianowy jeżeli relacja $prec$ jest typu $sp - graph$, natomiast jeżeli relacja $prec$ jest dowolna, to problem jest silnie NP-trudny (patrz [3]).

Rozważmy teraz rozmyty problem szeregowania $1|FN| \sum E(w_i \tilde{T}_i)$, w którym nie jest określona relacja poprzedzania a funkcja kosztu harmonogramu jest równa:

$$F(p) = \sum_{i=1}^n E(w_i \tilde{T}_i(p)).$$

Jeżeli czasy trwania i pożądane terminy zakończenia prac są zdeterminowane, to łatwo pokazać, korzystając z własności wartości średniej liczby rozmytej, że:

$$\sum_{i=1}^n E(w_i \tilde{T}_i(p)) = \sum_{i=1}^n w_i T_i(p).$$

Oznacza to, że w tym szczególnym przypadku, problem $1|FN| \sum E(w_i \tilde{T}_i)$ jest równoważny klasycznemu problemowi $1|| \sum w_i T_i$, który jest silnie NP-trudny (patrz [3]). Wynika z tego, że w ogólnym przypadku problem $1|FN| \sum E(w_i \tilde{T}_i)$ jest silnie NP-trudny.

Rozważmy teraz problem $1|TG|\sum E(w_i\tilde{T}_i)$, który jest szczególnym przypadkiem problemu $1|FN|\sum E(w_i\tilde{T}_i)$, w którym czasy trwania i pożądane terminy zakończenia prac są rozmytymi liczbami trójkątnymi. Ponieważ liczby zdeterminowanej nie można traktować jako szczególnego przypadku liczby trójkątnej, więc z NP-trudności problemu $1||\sum w_iT_i$ nie wynika bezpośrednio NP-trudność problemu $1|TG|\sum E(w_i\tilde{T}_i)$. Zauważmy, że w przypadku tego problemu nie można również w prosty sposób zastąpić czasów trwania prac ich wartościami średnimi i szukać dalej rozwiązania problemu $1||\sum w_iT_i$. Wynika to z faktu, że wartość średnia nie zachowuje, w przypadku liczb rozmytych, operacji maksimum. Prawdziwa jest jedynie następująca nierówność:

$$E(\tilde{T}_i(p)) = E(\max\{0, \tilde{C}_i(p) - \tilde{d}_i\}) \geq \max\{0, E(\tilde{C}_i(p)) - E(\tilde{d}_i)\}$$

Pokażemy, że problem $1|TG|\sum E(w_i\tilde{T}_i)$ jest również silnie NP-trudny. Dowód tego faktu ma znaczenie w pracy z tego względu, że w kolejnym rozdziale na problemie $1|TG|\sum E(w_i\tilde{T}_i)$ będzie testowana heurystyka, za pomocą której można w sposób przybliżony rozwiązywać problemy NP-trudne.

Twierdzenie 2.16. *Problem $1|TG|\sum E(w_i\tilde{T}_i)$ jest silnie NP-trudny.*

Dowód Pokażemy, że silnie NP-trudny problem $1||\sum w_iT_i$, w którym wszystkie parametry są liczbami całkowitymi (patrz [3]), jest T-transformowalny do problemu $1|TG|\sum E(w_i\tilde{T}_i)$. Niech $(n, (t_i)_{i=1}^n, (d_i)_{i=1}^n, (w_i)_{i=1}^n)$ będzie zadaną instancją problemu $1||\sum w_iT_i$. Przez $C_i(p)$ i $T_i(p)$ będziemy oznaczać odpowiednio czas zakończenia i opóźnienie pracy i w permutacji p dla problemu $1||\sum w_iT_i$.

Niech $(n, (\tilde{t}_i)_{i=1}^n, (\tilde{d}_i)_{i=1}^n, (w_i)_{i=1}^n)$ będzie odpowiednią instancją rozmytego problemu $1|TG|\sum E(w_i\tilde{T}_i)$, gdzie:

$$\tilde{t}_i = (2t_i, \varepsilon, \varepsilon), \quad i = 1, \dots, n,$$

$$\tilde{d}_i = (2d_i, \varepsilon, \varepsilon), \quad i = 1, \dots, n,$$

$$\varepsilon = \frac{1}{2n \sum_{i=1}^n w_i}.$$

Z własności operacji na liczbach trójkątnych otrzymujemy dla każdej pracy $i \in J$:

$$\tilde{C}_i(p) = \left(\sum_{j \in p^i} 2t_j, |p^i|\varepsilon, |p^i|\varepsilon \right) = (2C_i(p), |p^i|\varepsilon, |p^i|\varepsilon).$$

Obliczymy teraz dla każdej pracy i wartość wyrażenia:

$$E(w_i\tilde{T}_i(p)) = w_i E(\max\{0, \tilde{C}_i(p) - \tilde{d}_i\}).$$

Dla każdej pracy $i \in J$ prawdziwa jest następująca równość:

$$w_i E(\tilde{T}_i(p)) = w_i E(\max\{0, \tilde{D}_i(p)\}), \quad (2.95)$$

gdzie:

$$\tilde{D}_i(p) = \tilde{C}_i(p) - \tilde{d}_i = (2C_i(p) - 2d_i, 2|p^i|\varepsilon, 2|p^i|\varepsilon).$$

Do obliczenia wyrażenia (2.95) wykorzystamy wzór z lematu 1.13. Obliczymy najpierw współczynniki P i Q występujące w tym wzorze. Otrzymujemy:

$$P = -\frac{2C_i(p) - 2d_i}{2|p^i|\varepsilon} = \frac{2d_i - 2C_i(p)}{2|p^i|\varepsilon} = -Q.$$

Ponieważ $C_i(p)$ i d_i przyjmują wartości całkowite a $2|p^i|\varepsilon \in (0, 1)$ więc możliwe są jedynie następujące trzy przypadki: $P = 0$, $P > 1$ lub $P < -1$ ($Q > 1$). Stosując wzór z lematu 1.13 otrzymujemy więc następującą formułę:

$$w_i E(\tilde{T}_i(p)) = \begin{cases} 0 & \text{jeżeli } P > 1 \ (d_i > C_i(p)), \\ w_i \frac{|p^i|\varepsilon}{2} & \text{jeżeli } P = 0 \ (d_i = C_i(p)), \\ w_i (C_i(p) - d_i) & \text{jeżeli } Q > 1 \ (d_i < C_i(p)). \end{cases} \quad (2.96)$$

Ponieważ $T_i(p) = \max\{0, C_i(p) - d_i\}$, więc ze wzoru (2.96) otrzymujemy dla każdej pracy $i \in J$ i dla każdej permutacji p następujące oszacowanie:

$$w_i T_i(p) \leq w_i E(\tilde{T}_i(p)) \leq w_i T_i(p) + w_i \frac{|p^i|\varepsilon}{2}.$$

Stąd:

$$\sum_{i=1}^n w_i T_i(p) \leq \sum_{i=1}^n w_i E(\tilde{T}_i(p)) \leq \sum_{i=1}^n w_i T_i(p) + \varepsilon \sum_{i=1}^n w_i \frac{|p^i|}{2}. \quad (2.97)$$

Z określenia ε i z faktu, że $|p^i| \leq n$ otrzymujemy:

$$\varepsilon \sum_{i=1}^n w_i \frac{|p^i|}{2} = \frac{\sum_{i=1}^n |p^i| w_i}{4n \sum_{i=1}^n w_i} \leq \frac{n \sum_{i=1}^n w_i}{4n \sum_{i=1}^n w_i} = \frac{1}{4} < 1. \quad (2.98)$$

Z warunków (2.97) i (2.98) otrzymujemy:

$$\sum_{i=1}^n w_i T_i(p) \leq \sum_{i=1}^n w_i E(\tilde{T}_i(p)) < \sum_{i=1}^n w_i T_i(p) + 1. \quad (2.99)$$

Pokażemy teraz, że jeżeli p jest optymalnym rozwiązaniem rozmytego problemu $1|TG| \sum E(w_i \tilde{T}_i)$, to p jest również optymalnym rozwiązaniem problemu

$1|| \sum w_i T_i$. W przeciwnym wypadku, ponieważ $w_i T_i(p)$ zawsze przyjmuje wartości całkowite, musiałyby istnieć rozwiązanie p^* takie, że:

$$\sum_{i=1}^n w_i T_i(p^*) + 1 \leq \sum_{i=1}^n w_i T_i(p),$$

ale wówczas, stosując warunek (2.99) otrzymujemy:

$$\sum_{i=1}^n w_i E(\tilde{T}_i(p^*)) < \sum_{i=1}^n w_i T_i(p^*) + 1 \leq \sum_{i=1}^n T_i(p) \leq \sum_{i=1}^n w_i E(\tilde{T}_i(p)),$$

co byłoby sprzeczne z założeniem optymalności permutacji p dla $1|TG| \sum E(w_i \tilde{T}_i)$.

Aby znaleźć optymalne rozwiązanie problemu $1|| \sum w_i T_i$ wystarczy zatem znaleźć optymalne rozwiązanie odpowiedniej instancji problemu $1|TG| \sum E(w_i \tilde{T}_i)$. Przekształcenie jednej instancji w drugą może zostać łatwo wykonane w czasie wielomianowym. Wynika z tego, że silnie NP-trudny problem $1|| \sum w_i T_i$ jest T-transformowalny do problemu $1|TG| \sum E(w_i \tilde{T}_i)$ co oznacza, że rozmyty problem szeregowania $1|TG| \sum E(w_i \tilde{T}_i)$ jest również silnie NP-trudny. $\square$

Rozdział 3

Ocena optymalności harmonogramu w problemach szeregowania prac na pojedynczej maszynie z rozmytymi parametrami

W każdym z rozważanych w rozdziale drugim rozmytych problemów szeregowania pojęcie rozwiązania optymalnego było precyzyjne (jednoznacznie określone). Rozwiązaniem optymalnym był taki harmonogram, dla którego pewna funkcja kosztu, przyjmująca wartości rzeczywiste, osiągała wartość najmniejszą. Zatem, podobnie jak w przypadku klasycznego problemu szeregowania, każdy harmonogram albo jest albo nie jest rozwiązaniem optymalnym problemu. Podejście takie można jednak uważać za pewne uproszczenie. Obliczając funkcję kosztu harmonogramu traci się bowiem pewną część informacji, która jest zawarta w rozmytych parametrach. W rzeczywistości, jeżeli dokładna wartość parametrów nie jest znana, to nie można stwierdzić, czy zadany harmonogram będzie optymalny. Można natomiast mówić o ocenie możliwości, że zadany harmonogram będzie optymalny. W rozdziale tym zostanie przedstawiona definicja stopnia optymalności zadanego harmonogramu w problemie z rozmytymi parametrami oraz zostanie podany ogólny schemat algorytmu służącego do obliczania tego stopnia. Szczegółowe wyniki będą przedstawione dla problemu $1|prec|f_{max}$.

3.1 Koncepcja oceny stopnia optymalności harmonogramu

Rozpatrzmy pewien problem szeregowania S zbioru prac $J = \{1, \dots, n\}$ na pojedynczej maszynie z zadaną funkcją kosztu harmonogramu $f(p)$ (patrz [3]). Wartość funkcji $f(p)$ zależy od harmonogramu p oraz od pewnego zbioru

$$Z = \{z_1, \dots, z_l\}$$

ustalonych parametrów, które mogą być liczbami rzeczywistymi¹. Parametrami tymi są czasy trwania prac, pożądane terminy zakończenia, wagi itd. Dla zadanego harmonogramu p , funkcję $f(p)$ można traktować również jako funkcję parametrów $z_1, \dots, z_l$, tzn:

$$f(p) = f(p; z_1, \dots, z_l).$$

Założmy teraz, że parametry problemu zadane są nieprecyzyjnie w postaci liczb rozmytych $\tilde{z}_i$ z funkcjami przynależności $\mu_{\tilde{z}_i}(x)$, $i = 1, \dots, l$. Dopuszczamy również sytuację, w której niektóre parametry są zdeterminowane (są więc szczególnymi przypadkami liczb rozmytych). Podamy teraz metodę określania jaka jest możliwość, że zadany harmonogram p jest optymalny. Jeżeli parametry byłyby zadane w postaci zmiennych losowych to analogicznie szukalibyśmy odpowiedzi na pytanie jakie jest prawdopodobieństwo, że zadany harmonogram jest optymalny.

Niech $\Gamma^S(p) \subseteq \mathbb{R}^l$ będzie zbiorem układów wartości parametrów $(z_1, \dots, z_l)$, dla których zadany harmonogram p jest optymalny w problemie szeregowania S , tzn:

$$\forall_{(z_1, \dots, z_l) \in \Gamma^S(p)} \forall_{\pi} f(p; z_1, \dots, z_l) \leq f(\pi; z_1, \dots, z_l).$$

Definicja 3.1. *Stopień możliwości, że zadany harmonogram p jest optymalnym rozwiązaniem problemu S jest określony następująco:*

$$opt_S(p) = \begin{cases} 0 & \Gamma^S(p) = \emptyset, \\ \sup_{(z_1, \dots, z_l) \in \Gamma^S(p)} \{\min_{i=1, \dots, l} \mu_{\tilde{z}_i}(z_i)\} & \Gamma^S(p) \neq \emptyset. \end{cases} \quad (3.1)$$

Z definicji 3.1 wynika, że jeżeli harmonogram p jest niedopuszczalny, to $\Gamma^S(p) = \emptyset$ i $opt_S(p) = 0$.

¹W klasycznych problemach szeregowania często zakłada się, że parametry są liczbami całkowitymi. Będziemy rozważać tylko te problemy, w których założenie to nie ma istotnego znaczenia.

W dalszych rozważaniach ważny będzie przypadek, w którym parametry zadane są w postaci liczb przedziałowych (tzn. $\tilde{z}_i = [z_i, \bar{z}_i]$, $i = 1, \dots, l$). Zauważmy, że wówczas wskaźnik optymalności $opt_S(p)$ przyjmuje wartości 0 lub 1, przy czym $opt_S(p) = 1$ wtedy i tylko wtedy, gdy istnieje układ parametrów $(z_1, \dots, z_l) \in \Gamma^S(p)$ taki, że $z_i \in [z_i, \bar{z}_i]$, $i = 1, \dots, l$. Problem wyznaczenia stopnia optymalności $opt_S(p)$ zadanego harmonogramu p , jeżeli wszystkie parametry są zadane w postaci liczb przedziałowych, nazywamy **przedziałowym problemem szeregowania**. Przedziałowy problem szeregowania S będziemy oznaczać symbolem $i - S$. Ważność przedziałowego problemu szeregowania wynika z następującego twierdzenia:

Twierdzenie 3.1. *Stopień optymalności harmonogramu p jest nie mniejszy od ustalonej wartości $\lambda \in (0, 1]$, tj. $opt_S(p) \geq \lambda$, wtedy i tylko wtedy gdy istnieje układ parametrów $(z_1, \dots, z_l) \in \Gamma^S(p)$ taki, że $z_i \in \tilde{z}_i^\lambda$ dla $i = 1, \dots, l$.*

Dowód

$\Leftarrow$ Załóżmy, że istnieje układ parametrów $(z_1, \dots, z_l) \in \Gamma^S(p)$ taki, że $z_i \in \tilde{z}_i^\lambda$ dla $i = 1, \dots, l$, gdzie $\lambda \in (0, 1]$. Wówczas z definicji λ -poziomu liczby rozmytej otrzymujemy, że $\mu_{\tilde{z}_i}(z_i) \geq \lambda$ dla każdego $i = 1, \dots, l$. Stąd:

$$\min_{i=1, \dots, l} \{\mu_{\tilde{z}_i}(z_i)\} \geq \lambda. \quad (3.2)$$

Z warunku (3.2) oraz z definicji stopnia optymalności (patrz definicja 3.1) wynika, że $opt_S(p) \geq \lambda$.

$\Rightarrow$ Załóżmy, że $opt_S(p) = \lambda_0 \in (0, 1]$. Wówczas z definicji stopnia optymalności 3.1 i określenia supremum wynika, że:

$$\forall \varepsilon > 0 \exists (z_1, \dots, z_l) \in \Gamma^S(p) \min_{i=1, \dots, l} \{\mu_{\tilde{z}_i}(z_i)\} \geq \lambda_0 - \varepsilon.$$

Stąd oraz z własności operacji minimum wynika, że:

$$\forall \varepsilon > 0 \exists (z_1, \dots, z_l) \in \Gamma^S(p) \forall i=1, \dots, l \mu_{\tilde{z}_i}(z_i) \geq \lambda_0 - \varepsilon. \quad (3.3)$$

Z warunku (3.3) oraz z definicji λ -poziomu liczby rozmytej wynika następujący warunek:

$$\forall \varepsilon > 0 \exists (z_1, \dots, z_l) \in \Gamma^S(p) \forall i=1, \dots, l z_i \in [z_i(\lambda_0 - \varepsilon), \bar{z}_i(\lambda_0 - \varepsilon)], \quad (3.4)$$

Stąd oraz z lewostronnej ciągłości funkcji $\underline{z}_i(t), \bar{z}_i(t)$ dla $t \in (0, 1]$ otrzymujemy:

$$\exists (z_1, \dots, z_l) \in \Gamma^S(p) \forall i=1, \dots, l z_i \in [\underline{z}_i(\lambda_0), \bar{z}_i(\lambda_0)], \quad (3.5)$$

Jeżeli $opt_S(p) = \lambda_0 \geq \lambda$, to z warunku (3.5) i z faktu, że $[\underline{z}_i(\lambda_0), \bar{z}_i(\lambda_0)] \subseteq [\underline{z}_i(\lambda), \bar{z}_i(\lambda)]$ otrzymujemy, że:

$$\exists_{(z_1, \dots, z_l) \in \Gamma^S(p)} \forall_{i=1, \dots, l} z_i \in [\underline{z}_i(\lambda), \bar{z}_i(\lambda)],$$

co kończy dowód. □

Z twierdzenia 3.1 wynika, że aby stwierdzić, czy stopień optymalności zadane-
go harmonogramu p w problemie S jest nie mniejszy niż λ , wystarczy rozwiązać
problem przedziałowy $i - S$, w którym parametry są zadane w postaci odpowied-
nich λ -poziomów liczb rozmytych $\tilde{z}_i$, $i = 1, \dots, l$.

Korzystając z twierdzenia 3.1 podamy algorytm 13 wyznaczający stopień op-
tymalności zadanego harmonogramu z zadaną dokładnością ε .

Algorytm 13 Obliczenie stopnia optymalności harmonogramu p w problemie S

Require: Parametry problemu S , p , $\varepsilon \in (0, 1]$

Ensure: $opt_S(p)$

```

1: if  $p$  jest niedopuszczalny then
2:   return 0 / *  $opt_S(p) = 0$  */
3: end if
4: if  $opt_S(p) = 1$  w problemie  $i - S$  dla parametrów  $\tilde{z}_i^1$ ,  $i = 1, \dots, l$  then
5:   return 1 / *  $opt_S(p) = 1$  */
6: end if
7:  $\lambda_1 \leftarrow 0.5, \lambda_2 \leftarrow 0, k \leftarrow 1$ 
8: while  $|\lambda_1 - \lambda_2| \geq \varepsilon$  do
9:    $\lambda_2 \leftarrow \lambda_1$ 
10:  if  $opt_S(p) = 1$  w problemie  $i - S$  dla parametrów  $\tilde{z}_i^{\lambda_1}$ ,  $i = 1, \dots, l$  then
11:     $\lambda_1 \leftarrow \lambda_1 + \frac{1}{2^{k+1}}$ 
12:  else
13:     $\lambda_1 \leftarrow \lambda_1 - \frac{1}{2^{k+1}}$ 
14:  end if
15:   $k \leftarrow k + 1$ 
16: end while
17: return  $\lambda_1$  / *  $opt_S(p) = \lambda_1$  */
```

Poprawność algorytmu 13 jest bezpośrednią konsekwencją twierdzenia 3.1. Najpierw w wierszu 1 sprawdzamy, czy harmonogram p jest optymalny w stopniu 1. Jeżeli tak to algorytm kończy pracę. W przeciwnym wypadku algorytm szuka wartości $opt_S(p)$ w przedziale $(0, 1)$, zawężając po każdej iteracji pętli 5 - 13 dopuszczalny przedział dla $opt_S(p)$ o połowę. Łatwo sprawdzić, że otrzymana ocena optymalności harmonogramu p różni się od wartości dokładnej co najwyżej o ε .

Złożoność obliczeniowa algorytmu 13 jest rzędu $O(|\log \varepsilon| f(n))$, gdzie $f(n)$ jest złożonością obliczeniową problemu przedziałowego $i - S$, przy założeniu, że zależy ona wyłącznie od liczby prac n . Jeżeli potrafimy efektywnie rozwiązać problem przedziałowy $i - S$, to równie łatwo można ocenić optymalność zadanego harmonogramu z zadaną dokładnością ε w przypadku, gdy parametry są dowolnymi liczbami rozmytymi.

3.2 Przedziałowy problem $i - 1|prec|f_{max}$

Zanim zajmiemy się dwoma konkretnymi przedziałowymi problemami szeregowania typu $i - 1|prec|f_{max}$ (punkty 3.2.2 i 3.2.3), sformułujemy i udowodnimy warunki konieczne i dostateczne optymalności harmonogramu (permutacji) w zdeterminowanym problemie szeregowania $1|prec|f_{max}$. Warunki te są ciekawe, niezależnie od prowadzonych w tej pracy rozważań. Tutaj zostaną one wykorzystane do konstrukcji algorytmów dla przedziałowych problemów szeregowania.

3.2.1 Warunki konieczne i dostateczne optymalności harmonogramu w problemie $1|prec|f_{max}$

Rozpatrzmy klasyczny problem szeregowania $1|prec|f_{max}$. Przypomnijmy, że w problemie tym szukamy harmonogramu (permutacji zbioru prac J) p , spełniającego relację poprzedzania $prec$, dla którego funkcja:

$$f_{max}(p) = \max_{i=1, \dots, n} \{f_i(C_i(p))\}$$

osiąga minimum. Wartość funkcji $f_i(C_i(p))$ określa koszt zakończenia pracy $i \in J$ w harmonogramie (permutacji) p . Zakładamy, że dla każdej pracy $i \in J$ funkcja f_i jest niemalejącą funkcją czasu zakończenia pracy i , tzn.:

$$\forall p, T \geq 0 \quad f_i(C_i(p) + T) \geq f_i(C_i(p)).$$

Klasyczny problem $1|prec|f_{max}$, w którym wszystkie funkcje f_i są niemalejące jest najbardziej ogólnym problemem szeregowania na pojedynczej maszynie dla którego znany jest algorytm wielomianowy. Można go rozwiązać za pomocą algorytmu Lawlera (patrz algorytm 3). Analizę przedziałowego problemu $i - 1|prec|f_{max}$ zaczniemy od podania pewnych dalszych własności klasycznego problemu $1|prec|f_{max}$. Naszym celem będzie sformułowanie warunków koniecznych i dostatecznych na to, aby permutacja p była optymalnym rozwiązaniem problemu $1|prec|f_{max}$.

Wprowadzimy teraz pojęcie pracy krytycznej w permutacji p .

Definicja 3.2. Pracę $k \in J$ w permutacji p nazywamy krytyczną jeżeli:

$$f_k(C_k(p)) = \max_{i=1, \dots, n} \{f_i(C_i(p))\}.$$

Pracą krytyczną w permutacji p jest więc praca $k \in J$, dla której koszt zakończenia $f_k(C_k(p))$ jest największy. Zauważmy, że w danej permutacji p może wystąpić więcej niż jedna praca krytyczna. Udowodnimy następujące twierdzenie:

Twierdzenie 3.2. Jeżeli permutacja p jest dopuszczalna i istnieje w niej praca krytyczna k taka, że dla każdej pracy $j \in J$ takiej, że $j \in p^k$ i $(j, k) \notin \text{prec}$ zachodzi:

$$f_k(C_k(p)) \leq f_j(C_j(p)), \quad (3.6)$$

to p jest optymalnym rozwiązaniem problemu $1|\text{prec}|f_{\max}$.

Dowód Załóżmy, że w permutacji p istnieje praca krytyczna k dla której zachodzi warunek (3.6). Oznaczmy:

$$T = C_k(p) = \sum_{j \in p^k} t_j. \quad (3.7)$$

Założmy nie wprost, że permutacja p nie jest optymalna. Oznacza to, że istnieje dopuszczalna permutacja π dla której:

$$\max_{i=1, \dots, n} \{f_j(C_j(\pi))\} < \max_{j=1, \dots, n} \{f_j(C_j(p))\} = f_k(T). \quad (3.8)$$

Z warunku 3.8 wynika, że:

$$\forall_{j \in p^k} f_j(C_j(\pi)) < f_k(T). \quad (3.9)$$

Z założenia otrzymujemy:

$$\forall_{j \in p^k, (j, k) \notin \text{prec}} f_j(T) \geq f_k(T). \quad (3.10)$$

Ponieważ funkcje $f_i, i = 1, \dots, n$, są niemalejącymi funkcjami czasów zakończenia prac, to z warunków (3.9) i (3.10) wynika, że:

$$\forall_{j \in p^k, (j, k) \notin \text{prec}} C_j(\pi) < T,$$

czyli wszystkie prace $j \in p^k$ takie, że $(j, k) \notin prec$ muszą kończyć się w permutacji π wcześniej niż T (w szczególności praca k musi zakończyć się w permutacji π wcześniej niż T). Zauważmy, że prace $j \in p^k$ takie, że $(j, k) \in prec$ również muszą zakończyć się w permutacji π wcześniej niż T bowiem w przeciwnym wypadku π byłoby niedopuszczalne. Wynika z tego, że wszystkie prace $j \in p^k$ muszą kończyć się w π wcześniej niż T czyli:

$$\sum_{j \in p^k} t_j < T,$$

co daje sprzeczność z określeniem T (patrz formuła (3.7)). $\square$

Twierdzenie 3.2 określa warunek dostateczny na to aby permutacja p była optymalnym rozwiązaniem problemu $1|prec|f_{max}$. Niestety warunek ten nie jest jednocześnie warunkiem koniecznym. Pokażemy to na prostym przykładzie. Niech $J = \{1, 2, 3\}$, $t_1 = t_2 = t_3 = 1$, $f_1(t) = f_2(t) = 0$, $f_3(t) = 1$, $prec = \emptyset$. Łatwo zauważyć, że koszt każdej permutacji prac należących do J wynosi 1, czyli każda permutacja jest optymalna. Weźmy więc optymalną permutację $p = (1, 2, 3)$. W permutacji tej istnieje tylko jedna praca krytyczna (praca 3) ale:

$$1 = f_3(C_3(p)) > f_1(C_3(p)) = f_2(C_3(p)) = 0,$$

czyli warunek (3.6) z twierdzenia 3.2 nie jest spełniony.

Sformułujemy teraz warunek konieczny na to aby permutacja p była optymalnym rozwiązaniem problemu $1|prec|f_{max}$.

Twierdzenie 3.3. *Jeżeli permutacja p jest optymalnym rozwiązaniem problemu $1|prec|f_{max}$, to istnieje w niej praca krytyczna k taka, że dla każdej pracy $j \in J$ takiej, że $j \in p^k$ i $(j, k) \notin prec$ zachodzi:*

$$f_k(C_k(p)) \leq f_j(C_k(p)) \text{ lub } f_k(C_k(p)) = f_k(C_k(p) - t_j). \quad (3.11)$$

Dowód Załóżmy nie wprost, że permutacja p jest optymalna i warunek (3.11) nie jest spełniony, czyli dla każdej pracy krytycznej k istnieje praca $j \in J$ taka, że $j \in p^k$, $(j, k) \notin prec$ i jest prawdziwy warunek:

$$f_k(C_k(p)) > f_j(C_k(p)) \text{ i } f_k(C_k(p)) > f_k(C_k(p) - t_j). \quad (3.12)$$

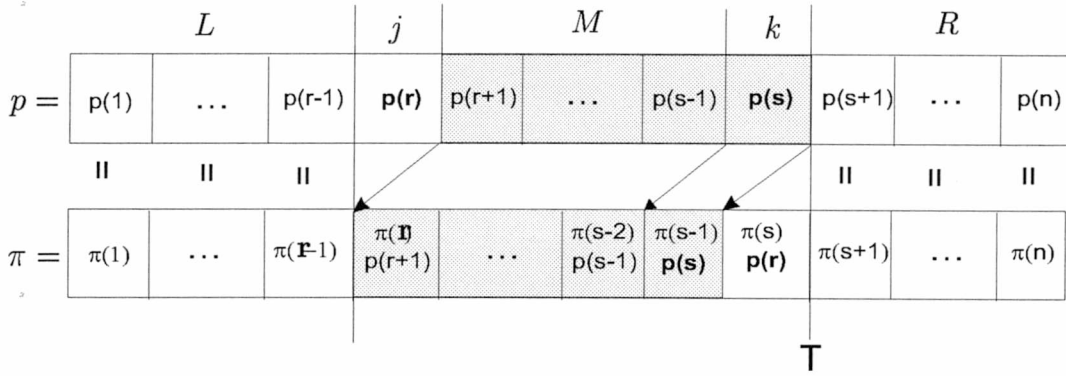
Warunek (3.12) jest zaprzeczeniem warunku (3.11). Zauważmy, że zaprzeczeniem warunku $f_k(C_k(p)) = f_k(C_k(p) - t_j)$ jest warunek $f_k(C_k(p)) > f_k(C_k(p) - t_j)$ ponieważ funkcja f_k jest z założenia niemalejąca.

Niech k będzie taką pracą krytyczną, która kończy się najwcześniej w permutacji p (czyli żadna praca poprzedzająca pracę k w permutacji p nie jest krytyczna). Oznaczmy $T = C_k(p)$. Niech $j \in p^k, (j, k) \notin prec$ będzie pracą, dla której zachodzi warunek (3.12), czyli:

$$f_k(T) > f_j(T) \text{ i } f_k(T) > f_k(T - t_j). \quad (3.13)$$

Utwórzmy permutację π , przesuując pracę j bezpośrednio za pracę k w permutacji p (patrz rys. 3.1). Permutacja π jest również dopuszczalna ponieważ z założenia $(j, k) \notin prec$. Wprowadźmy następujące oznaczenia (patrz rys. 3.1):

$$\begin{aligned} L &= \{p(1), \dots, p(r-1)\}, \\ j &= p(r), \\ M &= \{p(r+1), \dots, p(s-1)\}, \\ k &= p(s), \\ R &= \{p(s+1), \dots, p(n)\}. \end{aligned}$$



Rys. 3.1: Relacja między permutacjami p i π . Zakładamy, że żadna z prac $p(1), \dots, p(s-1)$ nie jest krytyczna w p

Z konstrukcji permutacji π wynika, że:

$$\forall_{i \in L} f_i(C_i(\pi)) = f_i(C_i(p)) < f_k(T), \quad (3.14)$$

$$\forall_{i \in R} f_i(C_i(\pi)) = f_i(C_i(p)) \leq f_k(T). \quad (3.15)$$

Ponieważ $C_i(\pi) < C_i(p)$ dla $i \in M$, więc z faktu, że wszystkie funkcje f_i są niemalejące wynika, że:

$$\forall_{i \in M} f_i(C_i(\pi)) \leq f_i(C_i(p)) < f_k(T). \quad (3.16)$$

Dla prac j i k z warunku (3.13) otrzymujemy:

$$f_j(C_j(\pi)) = f_j(T) < f_k(T), \quad (3.17)$$

$$f_k(C_k(\pi)) = f_k(T - t_j) < f_k(T). \quad (3.18)$$

Z warunków (3.14)-(3.18) wynika, że:

$$\forall_{i \in L \cup \{j, k\} \cup M} f_i(C_i(\pi)) < f_k(T), \quad (3.19)$$

$$\forall_{i \in R} f_i(C_i(\pi)) = f_i(C_i(p)) \leq f_k(T), \quad (3.20)$$

czyli koszt permutacji π jest niewiekszy niż koszt permutacji p .

Założmy teraz, że k jest jedyną pracą krytyczną występującą w p . Wówczas z warunków (3.19) i (3.20) oraz z faktu, że żadna praca należąca do R nie jest krytyczna w p wynika, że dla każdej pracy $i \in \{1, \dots, n\}$ zachodzi:

$$f_i(C_i(\pi)) < f_k(T),$$

czyli koszt permutacji π jest mniejszy niż p co przeczy założeniu, że permutacja p jest optymalna.

Założmy teraz, że w permutacji p występuje więcej niż jedna praca krytyczna. Wówczas musi to być jedna z prac należących do zbioru R ponieważ założyliśmy, że k jest tą pracą krytyczną, która kończy się najwcześniej w p .

Zauważmy, że z warunków (3.19) i (3.20) wynika, że permutacja π ma dokładnie o jedną pracę krytyczną mniej niż permutacja p . Niech $l \in R$ będzie następną po k pracą krytyczną występującą w permutacji p . Z warunków (3.19) i (3.20) wynika, że koszt permutacji π jest równy kosztowi permutacji p i wynosi $f_l(C_l(\pi))$, czyli l jest również pracą krytyczną w permutacji π .

Z założenia nie wprost wynika, że w permutacji p musi również istnieć praca $j \in J$ taka, że $j \in p^l$, $(j, l) \notin prec$ i jest spełniony warunek:

$$f_l(C_l(p)) > f_j(C_l(p)) \text{ i } f_l(C_l(p)) > f_l(C_l(p) - t_j). \quad (3.21)$$

Ponieważ $C_l(\pi) = C_l(p)$ i zbiór p^l jest równy zbiorowi π^l , więc z warunku (3.21) wynika, że $j \in \pi^l$ i zachodzi:

$$f_l(C_l(\pi)) > f_j(C_l(\pi)) \text{ i } f_l(C_l(\pi)) > f_l(C_l(\pi) - t_j).$$

Możemy więc ponownie utworzyć nową permutację przesuując pracę j bezpośrednio za pracę l . W ten sposób można utworzyć ciąg permutacji o tym samym koszcie co permutacja p , w którym każda kolejna permutacja ma coraz mniejszą

liczbę prac krytycznych. W pewnym momencie w ciągu tym wystąpi permutacja o tym samym koszcie co permutacja p , w której występuje tylko jedna praca krytyczna. Wówczas koszt kolejnej permutacji będzie już mniejszy niż koszt p co przeczy optymalności permutacji p . $\square$

Warunek konieczny sformułowany w twierdzeniu 3.3 różni się od warunku dostatecznego dodatkową, alternatywną częścią $f_k(C_k(p)) = f_k(C_k(p) - t_j)$. Zauważmy, że warunek ten oznacza, że funkcja f_k jest stała w pewnym, lewostronnym otoczeniu $C_k(p)$. Udowodnimy teraz następujące twierdzenie:

Twierdzenie 3.4. *Założmy, że dla każdej pracy krytycznej k w dopuszczalnej permutacji p funkcja $f_k(t)$ jest ściśle rosnąca². Wówczas permutacja p jest optymalna wtedy i tylko wtedy gdy istnieje w niej praca krytyczna k taka, że dla każdej pracy $j \in J$ takiej, że $j \in p^k$ i $(j, k) \notin prec$ zachodzi:*

$$f_k(C_k(p)) \leq f_j(C_k(p)). \quad (3.22)$$

Dowód

$\Leftarrow$ Wprost z twierdzenia 3.2.

$\Rightarrow$ Założmy, że permutacja p jest optymalna. Z założenia, że funkcja $f_k(t)$ jest ściśle rosnąca wynika, że dla każdego $j \in p^k$ zachodzi:

$$f_k(C_k(p) - t_j) < f_k(C_k(p)). \quad (3.23)$$

Z warunku (3.23) i twierdzenia 3.3 otrzymujemy, że w permutacji p musi wówczas istnieć praca krytyczna k taka, że dla każdej pracy $j \in J$ takiej, że $j \in p^k$ i $(j, k) \notin prec$ zachodzi:

$$f_k(C_k(p)) \leq f_j(C_k(p)),$$

co kończy dowód. $\square$

W twierdzeniu 3.4 podaliśmy, przy pewnych dodatkowych założeniach, warunek konieczny i dostateczny na to aby dopuszczalna permutacja p była optymalna. Twierdzenie to wykorzystamy w kolejnych punktach do rozwiązania szczególnych przypadków problemu $i - 1|prec|f_{max}$.

²Warunek ten można osłabić. Wystarczy, że funkcja f_k dla każdej pracy krytycznej k występującej w p jest ściśle rosnąca w pewnym przedziale $(C_k(p) - \varepsilon, C_k(p)]$, gdzie $\varepsilon > 0$. Dla potrzeb niniejszej pracy wystarczy silniejszy warunek podany w twierdzeniu. Słabszy warunek może być przydatny w innych zastosowaniach.

3.2.2 Problem $i - 1|prec|\max\{w_i L_i\}$

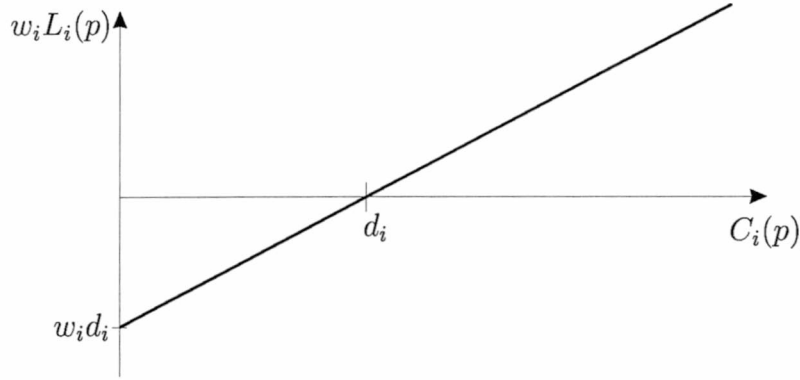
Rozpatrzmy klasyczny problem $1|prec|\max\{w_i L_i\}$, w którym funkcja kosztu harmonogramu ma następującą postać:

$$f(p) = \max_{i=1,\dots,n} \{w_i L_i(p)\} = \max_{i=1,\dots,n} \{w_i (C_i(p) - d_i)\}.$$

W problemie tym parametrami są więc czasy trwania prac t_i , pożądane terminy zakończenia d_i oraz wagi w_i zadane dla każdej pracy $i \in J$. Dla każdej pracy $i \in J$:

$$f_i(C_i(p)) = w_i L_i(p) = w_i (C_i(p) - d_i).$$

Zauważmy, że funkcja f_i jest dla dowolnych $w_i > 0$ oraz d_i ściśle rosnącą funkcją czasu zakończenia pracy $i \in J$ (patrz rys. 3.2).



Rys. 3.2: Funkcja kosztu $w_i L_i(p)$

Rozważmy teraz odpowiedni problem przedziałowy $i - 1|prec|\max\{w_i L_i\}$, w którym dla każdej pracy $i \in J$ czasy trwania $\tilde{t}_i$ i pożądane terminy zakończenia $\tilde{d}_i$ są liczbami przedziałowymi, tj. $\tilde{t}_i = [t_i, \bar{t}_i]$, $\tilde{d}_i = [\underline{d}_i, \bar{d}_i]$, natomiast wagi $w_i > 0$ są zdeterminowane³.

Będziemy szukać odpowiedzi na pytanie czy $opt_S(p) = 1$ dla zadanej permutacji p , czyli czy istnieją $t_i \in \tilde{t}_i$ oraz $d_i \in \tilde{d}_i$, $i = 1, \dots, n$, dla których permutacja p jest optymalnym rozwiązaniem problemu $1|prec|\max w_i L_i$. Aby uprościć dalsze oznaczenia będziemy konstruować odpowiedni algorytm dla ustalonej permutacji $p = (1, \dots, n)$. Dla każdej innej permutacji wystarczy jedynie w odpowiedni

³Założenie to upraszcza w znaczący sposób złożoność problemu, ponadto nieprecyzyjnie zadane wagi nie mają, w przeciwieństwie do pozostałych parametrów, prostej interpretacji.

sposób przenieumerować parametry problemu⁴. Wybór permutacji $p = (1, \dots, n)$ nie powoduje więc utraty ogólności rozważań. Wprowadźmy następujące oznaczenie:

$$T_i = C_i(p) = \sum_{j=1}^i t_j.$$

T_i jest więc czasem zakończenia pracy i w permutacji $p = (1, \dots, n)$ dla ustalonych czasów trwania prac t_j , $j = 1, \dots, n$. W dalszym ciągu będziemy zakładać, że permutacja p jest dopuszczalna (tzn. spełnia zadaną relację poprzedzania $prec$). Zachodzi następujące twierdzenie:

Twierdzenie 3.5. *$opt_S(p) = 1$ wtedy i tylko wtedy, gdy istnieje układ parametrów $t_i \in [\underline{t}_i, \bar{t}_i]$, $d_i \in [\underline{d}_i, \bar{d}_i]$, $i = 1, \dots, n$, dla których w p istnieje praca krytyczna $k \in J$ taka, że dla każdego $j = 1, \dots, k$ takiego, że $(j, k) \notin prec$ zachodzi warunek:*

$$w_k(T_k - d_k) \leq w_j(T_k - d_j). \quad (3.24)$$

Dowód Ponieważ dla każdego układu parametrów $t_i \in [\underline{t}_i, \bar{t}_i]$ i $d_i \in [\underline{d}_i, \bar{d}_i]$, $i = 1, \dots, n$, funkcja $f_k(T_k) = w_k(T_k - d_k)$ jest ściśle rosnąca, więc teza twierdzenia wynika bezpośrednio z twierdzenia 3.4. $\square$

Pokażemy teraz, że stopień optymalności permutacji $p = (1, \dots, n)$ można wyznaczyć, badając niesprzeczność pewnego układu nierówności liniowych. Prawdziwe jest bowiem następujące twierdzenie:

Twierdzenie 3.6. *$opt_S(p) = 1$ wtedy i tylko wtedy gdy istnieje $k \in J$, dla którego następujący układ U^k jest niesprzeczny:*

$$U^k = \begin{cases} 1: & \underline{t}_i \leq t_i \leq \bar{t}_i, & i = 1, \dots, n, \\ 2: & \underline{d}_i \leq d_i \leq \bar{d}_i, & i = 1, \dots, n, \\ 3: & T_i = t_1 + \dots + t_i, & i = 1, \dots, n, \\ 4: & w_k(T_k - d_k) \geq w_i(T_i - d_i), & i = 1, \dots, n, \\ 5: & w_k(T_k - d_k) \leq w_i(T_k - d_i), & i = 1, \dots, k, (i, k) \notin prec. \end{cases}$$

⁴Jeżeli $\pi = (\pi(1), \dots, \pi(n))$ jest dowolną permutacją, to parametry $\tilde{t}_{\pi(i)}, \tilde{d}_{\pi(i)}, w_{\pi(i)}$ zamieniamy na odpowiednio $\tilde{t}_i, \tilde{d}_i, w_i$ dla $i = 1, \dots, n$, natomiast relację poprzedzania $prec$ zamieniamy na relację $prec'$ taką, że:

$$(\pi(i), \pi(j)) \in prec \Rightarrow (i, j) \in prec'.$$

Jest oczywiste, $opt_S(\pi) = 1$ wtedy i tylko wtedy gdy po dokonanej transformacji parametrów $opt_S((1, \dots, n)) = 1$. Odpowiednie przekształcenie może zostać łatwo wykonane w czasie wielomianowo zależnym od liczby prac n .

Dowód Twierdzenie jest bezpośrednią konsekwencją twierdzenia 3.5. Ograniczenia 1: i 2: gwarantują, że wartości parametrów należą do zadanych przedziałów. Warunek 3: wiąże czasy zakończenia pracy w permutacji $p = (1, \dots, n)$ z czasami t_i . Warunek 4: zapewnia krytyczność zadanej pracy k a warunek 5: jest równoważny warunkowi dostatecznemu i koniecznemu optymalności permutacji p . Jeżeli dla pewnego k układ U^k jest niesprzeczny, to dowolne jego rozwiązanie stanowi układ wartości parametrów, dla którego są spełnione warunki optymalności podane w twierdzeniu 3.5, czyli dla którego permutacja p jest optymalna. Również na odwrót, jeżeli istnieje układ parametrów dla którego permutacja p jest optymalna to na mocy twierdzenia 3.5 co najmniej jeden z układów U^k dla $k \in J$ musi mieć rozwiązanie. $\square$

Zauważmy, że dla każdego k układ U^k jest układem równań i nierówności liniowych. Jego niesprzeczność można więc łatwo sprawdzić w czasie wielomianowym. Problem $i - 1|prec|\max\{w_i L_i\}$ można rozwiązać za pomocą algorytmu 14.

Algorytm 14 Algorytm dla problemu $i - 1|prec|\max\{w_i L_i\}$

Require: $n, (w_i)_{i=1}^n, ([\underline{d}_i, \bar{d}_i])_{i=1}^n, ([\underline{t}_i, \bar{t}_i])_{i=1}^n, prec, p$

Ensure: $opt_S(p)$

```

1: Jeżeli  $p \neq (1, \dots, n)$  to przetransformuj odpowiednio parametry problemu
2: if  $p$  jest niedopuszczalna then
3:   return 0
4: end if
5: for  $k = 1$  to  $n$  do
6:   if Układ  $U^k$  jest niesprzeczny then
7:     return 1
8:   end if
9: end for
10: return 0 /* Wszystkie układy  $U^k$  dla  $k \in J$  są sprzeczne */
```

Algorytm 14 w połączeniu z algorytmem 13 pozwala na efektywną ocenę stopnia optymalności dowolnej permutacji w problemie $1|prec|\max\{w_i L_i\}$, w którym czasy trwania prac i pożądane terminy zakończenia prac są dowolnymi liczbami rozmytymi.

Zbadamy teraz szczególny przypadek problemu $i - 1|prec|\max\{w_i L_i\}$, w którym $w_i = 1$ dla każdej pracy $i \in J$. Pokażemy, że wówczas układ U^k jest niesprzeczny wtedy i tylko wtedy gdy są spełnione pewne warunki narzucone na końce przedziałów $[\underline{t}_i, \bar{t}_i]$, $[\underline{d}_i, \bar{d}_i]$, $i \in J$. Zauważmy najpierw, że jeżeli wszystkie

wagi są równe jedności, to układ U^k można przedstawić w następującej postaci:

$$U^k = \begin{cases} 1: & \underline{t}_i \leq t_i \leq \bar{t}_i, & i = 1, \dots, n, \\ 2: & \underline{d}_i \leq d_i \leq \bar{d}_i, & i = 1, \dots, n, \\ 3: & T_i = t_1 + \dots + t_i, & i = 1, \dots, n, \\ 4: & T_k - T_i \geq d_k - d_i, & i = 1, \dots, n, \\ 5: & d_k - d_i \geq 0, & i = 1, \dots, k, (i, k) \notin prec. \end{cases}$$

Pokażemy, że zachodzi następujący lemat:

Lemat 3.1. *Układ U^k jest niesprzeczny wtedy i tylko wtedy gdy jest niesprzeczny następujący układ U_1^k :*

$$U_1^k = \begin{cases} 1: & t_i = \bar{t}_i, & i = 1, \dots, k, \\ 2: & t_i = \underline{t}_i, & i = k+1, \dots, n, \\ 3: & d_i = \bar{d}_i, & i = 1, \dots, k, (i, k) \in prec, \\ 4: & d_i = \bar{d}_i, & i = k+1, \dots, n, \\ 5: & \underline{d}_i \leq d_i \leq \bar{d}_i, & i = 1, \dots, k, (i, k) \notin prec, \\ 6: & T_i = t_1 + \dots + t_i, & i = 1, \dots, n, \\ 7: & T_k - T_i \geq d_k - d_i, & i = 1, \dots, n, \\ 8: & d_k - d_i \geq 0 & i = 1, \dots, k, (i, k) \notin prec. \end{cases}$$

Dowód Zauważmy najpierw, że układ U_1^k , otrzymujemy ustalając w układzie U^k wszystkie parametry z wyjątkiem pożądaných terminów zakończenia prac d_i dla $i = 1, \dots, k$ takich, że $(i, k) \notin prec$.

Załóżmy, że układ U_1^k jest niesprzeczny. Wówczas rozwiązanie tego układu musi spełniać również układ U^k czyli układ U^k też jest niesprzeczny.

Załóżmy teraz, że układ U^k jest niesprzeczny. Niech $X^* = (t_1^*, \dots, t_n^*, d_1^*, \dots, d_n^*)$ będzie rozwiązaniem układu U^k . Pokażemy, że zastępując wartości części składowych wektora X^* odpowiednimi końcami przedziałów (zgodnie z warunkami 1-4: w definicji układu U_1^k) otrzymamy rozwiązanie spełniające układ U_1^k . Wystarczy jedynie pokazać, że po takim podstawieniu będzie spełniony warunek 7: w definicji układu U_1^k , ponieważ pozostałe warunki będą spełnione automatycznie.

Niech $T_i^* = t_1^* + \dots + t_i^*$ dla $i = 1, \dots, n$. Z określenia T_i^* wynika więc, że:

$$T_k^* - T_i^* = t_1^* + \dots + t_k^* - t_1^* - \dots - t_i^*.$$

Stąd:

$$T_k^* - T_i^* = t_{i+1}^* + \dots + t_k^* \leq \bar{t}_{i+1} + \dots + \bar{t}_k, \quad i = 1, \dots, k-1, \quad (3.25)$$

$$T_k^* - T_i^* = -t_{k+1}^* - \dots - t_i^* \leq -\underline{t}_{k+1} - \dots - \underline{t}_i, \quad i = k+1, \dots, n. \quad (3.26)$$

Zachodzi również następująca zależność:

$$d_k^* - d_i^* \geq d_k^* - \bar{d}_i, \quad i = k+1, \dots, n; \quad i = 1, \dots, k, \quad (i, k) \in prec. \quad (3.27)$$

Z założenia, że X^* spełnia układ U^k wynika, że:

$$T_k^* - T_i^* \geq d_k^* - d_i^*, \quad i = 1, \dots, n.$$

Stąd oraz z warunków (3.25), (3.26) i (3.27) wynika, że warunek 7: w definicji układu U_k^1 , po zastąpieniu wartości części składowych wektora X^* odpowiednimi końcami przedziałów (zgodnie z warunkami 1: - 4: w definicji układu U_1^k) jest również spełniony dla każdego $i = 1, \dots, n$. $\square$

Zauważmy, że aby rozstrzygnąć niesprzeczność układu U_k^1 należy tylko sprawdzić, czy istnieje układ pożądaných terminów zakończenia $d_i \in [\underline{d}_i, \bar{d}_i]$ dla $i = 1, \dots, k$ takich, że $(i, k) \notin prec$, które spełniają układ U_1^k . Wprowadźmy następujące oznaczenia:

$$\Delta_{ki} = \begin{cases} \bar{t}_{i+1} + \dots + \bar{t}_k & \text{dla } i = 1, \dots, k-1, \\ 0 & \text{dla } i = k, \\ -\underline{t}_{k+1} - \dots - \underline{t}_i & \text{dla } i = k+1, \dots, n. \end{cases}$$

$$J_k^L = \{i | i \leq k; (i, k) \notin prec\},$$

$$J_k^R = \{i | i \leq k; (i, k) \in prec\},$$

$$\Upsilon_k = \min_{i \in J_k^R \cup \{k, \dots, n\}} \{\Delta_{ki} + \bar{d}_i\}$$

Korzystając z powyższych oznaczeń łatwo pokazać, że warunki 1:, 2:, 3:, 4:, 6: i 7: w układzie U_1^k można zastąpić następującymi dwoma warunkami:

$$\begin{aligned} d_k &\leq \Delta_{ki} + \bar{d}_i & i \in J_k^R \cup \{k, \dots, n\}, \\ d_k - d_i &\leq \Delta_{ki} & i \in J_k^L, \end{aligned}$$

które z kolei są równoważne warunkom:

$$\begin{aligned} d_k &\leq \Upsilon_k \\ d_k - d_i &\leq \Delta_{ki} & i \in J_k^L, \end{aligned}$$

Stąd układ U_1^k może być zapisany w następującej równoważnej postaci:

$$U_1^k = \begin{cases} 1: & \underline{d}_i \leq d_i \leq \bar{d}_i & i \in J_k^L, \\ 2: & d_k \leq \Upsilon_k, \\ 3: & 0 \leq d_k - d_i \leq \Delta_{ki} & i \in J_k^L. \end{cases} \quad (3.28)$$

Zauważmy, że Υ_k oraz Δ_{ki} są ustalone w układzie U_1^k . Udowodnimy następujące twierdzenie:

Twierdzenie 3.7. Układ U_1^k jest niesprzeczny wtedy i tylko wtedy gdy są spełnione następujące warunki:

$$\Phi_k = \begin{cases} 1 : \gamma = \max_{i \in J_k^L} \{\underline{d}_i\}, \\ 2 : \delta_i = \min\{\gamma, \bar{d}_i\} \quad \text{dla } i \in J_k^L, \\ 3 : \delta_k \leq \Upsilon_k, \\ 4 : 0 \leq \delta_k - \delta_i \leq \Delta_{ki} \quad \text{dla } i \in J_k^L. \end{cases}$$

Dowód Pokażemy najpierw, że jeżeli warunki Φ_k są spełnione, to układ U_1^k jest niesprzeczny. Załóżmy, że warunki Φ_k są spełnione. Wynika z tego, że $\delta_i, i \in J_k^L$, spełniają warunki 2: i 3: w układzie U_k^1 (patrz(3.28)). Wystarczy więc jeszcze pokazać, że $\delta_i \in [\underline{d}_i, \bar{d}_i]$ dla każdego $i \in J_k^L$. Z określenia δ_i wynika od razu, że $\delta_i \leq \bar{d}_i$. Z faktów, że $\gamma \geq \underline{d}_i$ oraz $\bar{d}_i \geq \underline{d}_i$ wynika również, że $\delta_i \geq \underline{d}_i$.

Pokażemy teraz, w drugą stronę, że niespełnienie warunków Φ_k pociąga za sobą sprzeczność układu U_1^k . Należy więc pokazać, że niespełnienie warunków 3: lub 4: w Φ_k pociąga za sobą sprzeczność układu U_1^k .

Założmy, że dla pewnego $i \in J_k^L$ zachodzi $\delta_k - \delta_i < 0$. Wówczas z określenia δ_i otrzymujemy następujący warunek:

$$\min\{\gamma, \bar{d}_k\} < \min\{\gamma, \bar{d}_i\}.$$

Z warunku tego wynika, że $\bar{d}_k < \gamma = \max_{i \in J_k^L} \{\underline{d}_i\}$, czyli istnieje $i^* \in J_k^L$ takie, że $\bar{d}_k < \underline{d}_{i^*}$. Wynika z tego, że warunek 3: w układzie U_1^k (patrz (3.28)) nie może być spełniony dla i^* bowiem dla każdych $d_k \in [\underline{d}_k, \bar{d}_k]$, $d_{i^*} \in [\underline{d}_{i^*}, \bar{d}_{i^*}]$ zachodzi $d_k - d_{i^*} < 0$.

Założmy, że dla pewnego $i^* \in J_k^L$ zachodzi $\delta_k - \delta_{i^*} > \Delta_{ki^*}$. Z określenia Δ_{ki} wynika, że $\Delta_{ki} \geq 0$ dla $i \in J_k^L$. Stąd oraz z określenia δ_i wynika, że:

$$\min\{\gamma, \bar{d}_k\} - \min\{\gamma, \bar{d}_{i^*}\} > \Delta_{ki^*} \geq 0.$$

Stąd oraz z własności operacji minimum wynika, że:

$$\gamma - \min\{\gamma, \bar{d}_{i^*}\} > \Delta_{ki^*} \geq 0. \quad (3.29)$$

Z warunku (3.29) wynika, że $\gamma > \bar{d}_{i^*}$. Stąd oraz z określenia γ otrzymujemy następujący warunek:

$$\max_{i \in J_k^L} \{\underline{d}_i\} - \bar{d}_{i^*} > \Delta_{ki^*}. \quad (3.30)$$

Pokażemy teraz, że nie istnieje d_k , które spełniałoby warunek 3: w układzie U_1^k (patrz (3.28)) dla wszystkich $i \in J_k^L$. Rzeczywiście jeżeli $d_k \geq \max_{i \in J_k^L} \{\underline{d}_i\}$, to

bezpośrednio z (3.30) wynika, że warunek 3: w U_1^k nie może być spełniony dla i^* . Jeżeli natomiast $d_k < \max_{i \in J_k^L} \{d_i\}$, to $d_k < d_i$ dla każdego $i \in J_k^L$ i $d_k - d_i < 0$ dla każdego $d_i \in [d_i, \bar{d}_i]$, czyli warunek 3: w U_1^k nie jest spełniony dla każdego $i \in J_k^L$.

Założmy, że $\delta_k > \Upsilon_k$. Z określenia δ_k otrzymujemy, że:

$$\min\{\gamma, \bar{d}_k\} > \Upsilon_k$$

Stąd oraz z własności operacji minimum wynika, że $\gamma > \Upsilon_k$ i z określenia γ otrzymujemy:

$$\max_{i \in J_k^L} \{d_i\} > \Upsilon_k. \quad (3.31)$$

Pokażemy teraz, że nie istnieje d_k , które spełniałoby warunek 2: w układzie U_1^k (patrz (3.28)). Rzeczywiście jeżeli $d_k \geq \max_{i \in J_k^L} \{d_i\}$, to bezpośrednio z (3.31) wynika, że warunek 2: w U_1^k nie może być spełniony. Jeżeli natomiast $d_k < \max_{i \in J_k^L} \{d_i\}$, to rozumowanie jest analogiczne jak w przypadku formuły (3.30).

Pokazaliśmy, że jeżeli warunki 3: i 4: w Φ_k nie są spełnione to układ U_1^k jest sprzeczny. $\square$

Algorytm dla problemu $i - 1|prec|L_{max}$ jest analogiczny do algorytmu 14. W kroku 6: Zamiast sprawdzać niesprzeczność układu U_k wystarczy sprawdzić czy zachodzą warunki Φ_k . Algorytm dla problemu $i - 1|prec|L_{max}$ ma złożoność rzędu $O(n^3)$. Wynika to z faktu, że sprawdzenie każdego z warunków Φ_k ma złożoność rzędu $O(n^2)$. Ponieważ w najgorszym przypadku należy sprawdzić n warunków, to złożoność całego algorytmu jest rzędu $O(n^3)$.

Pokażemy teraz ocenę stopnia optymalności permutacji dla przykładowego problemu $1|prec|L_{max}$, w którym czasy trwania prac i pożądane terminy zakończenia prac są liczbami rozmytymi.

Przykład: Niech $J = \{1, \dots, 6\}$ i $prec = \{(2, 5), (3, 4), (3, 6)\}$. Wszystkie parametry problemu są przedstawione w tabeli 3.1.

$i \in J$	$\tilde{t}_i$	$\tilde{d}_i$
1	$(1, 1, 1, 3)_{POW_3-POW_2}$	$(8, 12, 3, 8)_{POW_2-POW_2}$
2	$(2, 5, 1, 2)_{LIN-POW_2}$	$(14, 15, 5, 2)_{LIN-POW_2}$
3	$(3, 6, 2, 3)_{POW_2-LIN}$	$(11, 13, 3, 2)_{POW_2-POW_3}$
4	$(4, 5, 2, 2)_{LIN-LIN}$	$(6, 7, 3, 3)_{LIN-LIN}$
5	$(2, 2, 1, 2)_{POW_2-LIN}$	$(5, 9, 2, 2)_{POW_3-LIN}$
6	$(3, 3, 1, 1)_{LIN-POW_2}$	$(9, 10, 3, 4)_{LIN-LIN}$

Tab. 3.1: Parametry przykładowego problemu

Wyznamy stopień optymalności $opt_S(p)$ permutacji $p = (1, 2, 3, 4, 5, 6)$ korzystając z algorytmów 13 i 14. Przyjmujemy dokładność obliczeń $\varepsilon = 0.01$.

KROK 1 Sprawdź warunki $\Phi_1, \dots, \Phi_6$ dla parametrów:

$$\begin{aligned}(\tilde{t}_i^1)_{i=1}^6 &= ([1, 1], [2, 5], [3, 6], [4, 5], [2, 2], [3, 3]), \\ (\tilde{d}_i^1)_{i=1}^6 &= ([8, 12], [14, 15], [11, 13], [6, 7], [5, 9], [9, 10]).\end{aligned}$$

Żaden z warunków $\Phi_1, \dots, \Phi_6$ nie jest spełniony czyli $opt_S(p) < 1$.

KROK 2 Sprawdź warunki $\Phi_1, \dots, \Phi_6$ dla parametrów:

$$\begin{aligned}(\tilde{t}_i^{0.5})_{i=1}^6 &= ([0.2, 3.12], [1.5, 6.41], [1.58, 7.5], [3, 6], [1.29, 3], [2.5, 3.70]), \\ (\tilde{d}_i^{0.5})_{i=1}^6 &= ([5.87, 17.65], [11.5, 16.58], [8.87, 14.58], [4.5, 8.5], [3.41, 10], [7.5, 12]).\end{aligned}$$

Warunek Φ_5 jest spełniony, czyli $opt_S(p) \in [0.5, 1)$. Permutacja p jest optymalna dla realizacji:

$$\begin{aligned}(t_i)_{i=1}^7 &= (3.12, 6.41, 7.5, 6, \underline{1.29}, 2.5), \\ (d_i)_{i=1}^7 &= (8.87, 16.58, 8.87, 8.5, \underline{8.87}, 12).\end{aligned}$$

KROK 3 Sprawdź warunki $\Phi_1, \dots, \Phi_6$ dla parametrów:

$$\begin{aligned}(\tilde{t}_i^{0.75})_{i=1}^6 &= ([0.37, 2.5], [1.75, 6], [2, 6.75], [3.5, 5.5], [1.5, 2.5], [2.75, 3.5]), \\ (\tilde{d}_i^{0.75})_{i=1}^6 &= ([6.5, 16], [12.7, 16.25], [9.5, 14.25], [5.2, 7.75], [3.7, 9.5], [8.25, 11]).\end{aligned}$$

Żaden z warunków $\Phi_1, \dots, \Phi_6$ nie jest spełniony czyli $opt_S(p) \in [0.5, 0.75)$.

KROK 4 Sprawdź warunki $\Phi_1, \dots, \Phi_6$ dla parametrów:

$$\begin{aligned}(\tilde{t}_i^{0.62})_{i=1}^6 &= ([0.27, 2.83], [1.62, 6.22], [1.77, 7.12], [3.25, 5.75], [1.38, 2.75], [2.62, 3.61]), \\ (\tilde{d}_i^{0.62})_{i=1}^6 &= ([6.16, 16.8], [12.12, 16.44], [9.16, 14.44], [4.87, 8.12], [3.55, 9.75], [7.87, 11.5]).\end{aligned}$$

Żaden z warunków $\Phi_1, \dots, \Phi_6$ nie jest spełniony czyli $opt_S(p) \in [0.5, 0.62)$.

KROK 5 Sprawdź warunki $\Phi_1, \dots, \Phi_6$ dla parametrów:

$$\begin{aligned}(\tilde{t}_i^{0.56})_{i=1}^6 &= ([0.24, 2.98], [1.56, 6.32], [1.67, 7.31], [3.12, 5.87], [1.33, 2.87], [2.56, 3.66]), \\ (\tilde{d}_i^{0.56})_{i=1}^6 &= ([6.01, 17.29], [11.81, 16.51], [9.01, 14.51], [4.68, 8.31], [3.48, 9.87], [7.68, 11.75]).\end{aligned}$$

Warunek Φ_5 jest spełniony czyli $opt_S(p) \in [0.56, 0.62]$. Permutacja p jest optymalna dla realizacji:

$$\begin{aligned}(t_i)_{i=1}^6 &= (2.98, 6.32, 7.31, 5.87, \underline{2.87}, 2.56), \\ (d_i)_{i=1}^6 &= (9.01, 16.51, 9.01, 8.31, \underline{9.01}, 11.75).\end{aligned}$$

KROK 6 Sprawdź warunki $\Phi_1, \dots, \Phi_6$ dla parametrów:

$$\begin{aligned}(\tilde{t}_i^{0.59})_{i=1}^6 &= ([0.25, 2.91], [1.59, 6.27], [1.72, 7.21], [3.18, 5.81], [1.36, 2.81], [2.59, 3.63]), \\ (\tilde{d}_i^{0.59})_{i=1}^6 &= ([6.08, 17.09], [11.96, 16.48], [9.08, 14.48], [4.78, 8.21], [3.51, 9.81], [7.78, 11.62]).\end{aligned}$$

Żaden z warunków $\Phi_1, \dots, \Phi_6$ nie jest spełniony czyli $opt_S(p) \in [0.56, 0.59]$.

KROK 7 Sprawdź warunki $\Phi_1, \dots, \Phi_6$ dla parametrów:

$$\begin{aligned}(\tilde{t}_i^{0.57})_{i=1}^6 &= ([0.25, 2.94], [1.57, 6.29], [1.7, 7.26], [3.15, 5.84], [1.35, 2.84], [2.57, 3.64]), \\ (\tilde{d}_i^{0.57})_{i=1}^6 &= ([6.05, 17.19], [11.89, 16.5], [9.05, 14.5], [4.73, 8.26], [3.5, 9.84], [7.73, 11.68]).\end{aligned}$$

Warunek Φ_5 jest spełniony, czyli $opt_S(p) \in [0.57, 0.59]$. Permutacja p jest optymalna dla realizacji:

$$\begin{aligned}(t_i)_{i=1}^6 &= (2.94, 6.29, 7.26, 5.84, \underline{2.84}, 2.57), \\ (d_i)_{i=1}^6 &= (9.05, 16.5, 9.05, 8.26, \underline{9.05}, 11.68).\end{aligned}$$

KONIEC $opt_S(p) = 0.58 \pm 0.01$.

3.2.3 Problem $i - 1|prec|max\{w_i T_i\}$

Pokażemy teraz w jaki sposób zmodyfikować algorytm 14, żeby można było za jego pomocą rozwiązać również problem $i - 1|prec|max\{w_i T_i\}$.

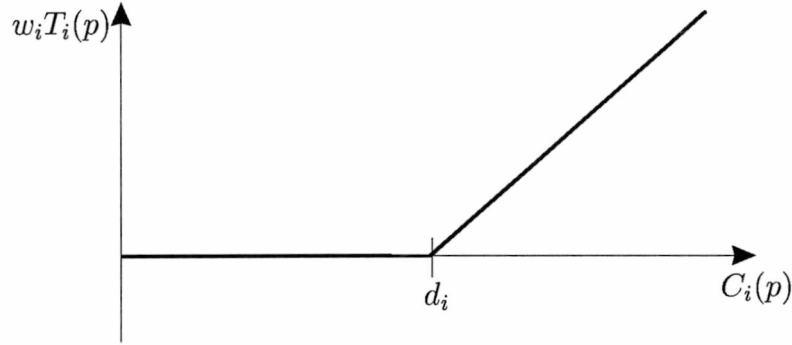
W klasycznym problemie $1|prec|max\{w_i T_i\}$ funkcja kosztu harmonogramu jest określona następująco:

$$f(p) = \max_{i=1, \dots, n} \{w_i T_i(p)\} = \max_{i=1, \dots, n} \{\max\{0, w_i(C_i(p) - d_i)\}\}. \quad (3.32)$$

Dla każdej pracy $i \in J$ koszt zakończenia jest więc równy:

$$f_i(C_i(p)) = T_i(p) = \max\{0, w_i(C_i(p) - d_i)\}.$$

Zauważmy, że funkcja ta nie jest ściśle rosnącą funkcją czasu zakończenia pracy i (patrz rys. 3.3).



Rys. 3.3: Funkcja kosztu $w_i T_i(p)$

Pokażemy teraz kilka dalszych własności problemu $1|prec|\max\{w_i T_i\}$.

Lemat 3.2. *Jeżeli koszt permutacji p w problemie $1|prec|\max\{w_i T_i\}$ jest równy 0, to permutacja p jest optymalna.*

Dowód Z określenia funkcji kosztu $f(p)$ (patrz warunek (3.32)) wynika, że 0 jest najmniejszą wartością przyjmowaną przez tą funkcję dla dowolnych parametrów. Oznacza to, że jeżeli $f(p) = 0$, to nie istnieje permutacja o mniejszym koszcie, czyli p jest optymalna. $\square$

Lemat 3.3. *Jeżeli koszt każdej permutacji w problemie $1|prec|\max\{w_i T_i\}$ jest większy od 0, to problem ten jest równoważny problemowi $1|prec|\max\{w_i L_i\}$ z tymi samymi parametrami.*

Dowód Z założenia, dla każdej permutacji p zachodzi:

$$\max_{i=1,\dots,n} \{w_i T_i(p)\} = T_k(p) > 0,$$

dla pewnego $k = 1, \dots, n$. Ponieważ $T_k(p) > 0$, więc:

$$T_k(p) = \max\{0, C_k(p) - d_k\} = C_k(p) - d_k = L_k(p),$$

stąd:

$$\max_{i=1,\dots,n} \{w_i T_i(p)\} = \max_{i=1,\dots,n} \{w_i L_i(p)\}. \quad (3.33)$$

Warunek (3.33) zachodzi dla każdej permutacji p , z czego wynika, że problemy $1|prec|\max\{w_i T_i\}$ oraz $1|prec|\max\{w_i L_i\}$ z tymi samymi parametrami mają takie same rozwiązania optymalne, czyli są równoważne. $\square$

Rozpatrzmy teraz przedziałowy problem $i - 1|prec| \max\{w_i T_i\}$. W problemie tym czasy trwania prac i pożądane terminy zakończenia prac są zadane w postaci liczb przedziałowych odpowiednio: $\tilde{t}_i = [\underline{t}_i, \bar{t}_i]$, $\tilde{d}_i = [\underline{d}_i, \bar{d}_i]$, $i = 1, \dots, n$, natomiast wszystkie wagi $w_i > 0$ są zdeterminowane. Będziemy szukać odpowiedzi na pytanie czy $opt_S(p) = 1$ dla zadanej permutacji p , czyli czy istnieją $t_i \in \tilde{t}_i$ oraz $d_i \in \tilde{d}_i$, $i = 1, \dots, n$, dla których permutacja p jest optymalnym rozwiązaniem problemu $1|prec| \max\{w_i T_i\}$.

Udowodnimy następujący lemat:

Lemat 3.4. *Dla dowolnej permutacji p istnieje układ parametrów $t_i \in [\underline{t}_i, \bar{t}_i]$, $d_i \in [\underline{d}_i, \bar{d}_i]$, $i = 1, \dots, n$, dla których $f(p; t_1, \dots, t_n, d_1, \dots, d_n) = 0$ wtedy i tylko wtedy gdy $f(p; \underline{t}_1, \dots, \underline{t}_1, \bar{d}_1, \dots, \bar{d}_n) = 0$.*

Dowód

$\Leftarrow$ Oczywiście.

$\Rightarrow$ Załóżmy, że $f(p; t_1, \dots, t_n, d_1, \dots, d_n) = 0$ dla pewnego układu parametrów $t_i \in [\underline{t}_i, \bar{t}_i]$, $d_i \in [\underline{d}_i, \bar{d}_i]$. Wówczas dla tego układu spełniony jest warunek:

$$T_i(p) = \max\{0, C_i(p) - d_i\} = 0, i \in J. \quad (3.34)$$

Oznaczmy $\underline{C}_i(p) = \sum_{j \in p^i} \underline{t}_j$. Ponieważ $\underline{C}_i(p) \leq C_i(p)$ i $\bar{d}_i \geq d_i$, więc dla każdego $i \in J$:

$$T_i(p) = \max\{0, C_i(p) - d_i\} \geq \max\{0, \underline{C}_i(p) - \bar{d}_i\} \geq 0.$$

Stąd oraz z warunku (3.34) wynika, że $f(p; \underline{t}_1, \dots, \underline{t}_1, \bar{d}_1, \dots, \bar{d}_n) = 0$. $\square$

Twierdzenie 3.8. *Jeżeli w problemie $i - 1|prec| \max\{w_i T_i\}$ jest spełniony warunek:*

$$f(p; \underline{t}_1, \dots, \underline{t}_1, \bar{d}_1, \dots, \bar{d}_n) > 0,$$

to problem ten jest równoważny problemowi $i - 1|prec| \max\{w_i L_i\}$ z tymi samymi parametrami.

Dowód Z lematu 3.4 wynika, że jeżeli $f(p; \underline{t}_1, \dots, \underline{t}_1, \bar{d}_1, \dots, \bar{d}_n) > 0$, to dla każdych $t_i \in [\underline{t}_i, \bar{t}_i]$, $d_i \in [\underline{d}_i, \bar{d}_i]$ jest spełniona nierówność $f(p; t_1, \dots, t_1, d_1, \dots, d_n) > 0$, tzn. że dla każdego układu wartości parametrów koszt permutacji p jest większy od 0. Z lematu 3.3 wynika, że permutacja p jest wówczas optymalna dla pewnego układu parametrów w problemie $1|prec| \max\{w_i T_i\}$ wtedy i tylko wtedy gdy jest optymalna dla tego samego układu parametrów w problemie $1|prec| \max\{w_i L_i\}$. Wynika z tego, że odpowiednie problemy przedziałowe są sobie równoważne. $\square$

Algorytm 15 Algorytm dla problemu $i - 1|prec|\max\{w_i T_i\}$

Require: $n, (w_i)_{i=1}^n, ([\underline{d}_i, \bar{d}_i])_{i=1}^n, ([\underline{t}_i, \bar{t}_i])_{i=1}^n, prec, p$

Ensure: $opt_S(p)$

```
1: Jeżeli  $p \neq (1, \dots, n)$  to przetransformuj odpowiednio parametry problemu
2: if  $p$  jest niedopuszczalna then
3:   return 0
4: end if
5: if  $f(p; \underline{t}_1, \dots, \underline{t}_1, \bar{d}_1, \dots, \bar{d}_n) = 0$  then
6:   return 1
7: end if
8: for  $k = 1$  to  $n$  do
9:   if Układ  $U^k$  jest niesprzeczny then
10:    return 1
11:  end if
12: end for
13: return 0 /* Wszystkie układy  $U^k$  są sprzeczne
```

Korzystając z twierdzenia 3.8 można podać algorytm 15 dla problemu $i - 1|prec|\max\{w_i T_i\}$.

Algorytm 15 różni się od algorytmu 14 jedynie dodatkowymi wierszami 5-7, w których sprawdzamy założenie twierdzenia 3.8. Jeżeli $f(p; \underline{t}_1, \dots, \underline{t}_1, \bar{d}_1, \dots, \bar{d}_n) = 0$, to na mocy lematu 3.2 permutacja p jest optymalna dla układu parametrów $\underline{t}_i, \bar{d}_i, i = 1, \dots, n$ i oczywiście $opt_S(p) = 1$. W przeciwnym wypadku rozwiązujemy już w dalszym ciągu problem $i - 1|prec|\max\{w_i L_i\}$. Jeżeli $w_i = 1$ dla każdego $i \in J$, to zamiast badać niesprzeczność układu U_k w wierszu 9 możemy sprawdzić czy zachodzą odpowiednie warunki Φ_k .

Rozdział 4

Metody heurystyczne dla NP-trudnych rozmytych problemów szeregowania

W rozdziale drugim została udowodniona NP-trudność kilku rozmytych problemów szeregowania. Oznacza to, że w praktycznych zastosowaniach, gdy liczba prac jest duża, nie da się dla tych problemów szybko i efektywnie skonstruować optymalnego harmonogramu. Jeżeli zbiór dopuszczalnych harmonogramów pokrywa się ze zbiorem wszystkich permutacji zbioru prac, to liczba dopuszczalnych harmonogramów wynosi $n!$ (gdzie n jest liczbą prac). W takim przypadku prosty algorytm przeglądający wszystkie harmonogramy staje się nieefektywny już dla kilkunastu prac.

W ostatnich latach do rozwiązywania problemów NP-trudnych zastosowano kilka interesujących heurystyk. Heurystyka jest to metoda, za pomocą której można znaleźć dobre, przybliżone rozwiązanie danego problemu. Stosując heurystyki dla problemów NP-trudnych napotyka się na dwie zasadnicze trudności. Pierwsza polega na tym, że często heurystyki te wymagają wcześniejszego ustalenia szeregu parametrów. Wartości tych parametrów mają silny wpływ na jakość uzyskiwanych rozwiązań a ich najlepsze wartości zależą od rodzaju problemu, jego rozmiaru i danych wejściowych. Najczęściej istnieje niewiele przesłanek teoretycznych, które pozwalają na odpowiedni dobór parametrów heurystyki. Dlatego ustala się je subiektywnie lub eksperymentalnie. Drugą istotną trudnością jest fakt, że aby ocenić efektywność heurystyki należy oszacować o ile uzyskane za jej pomocą rozwiązanie różni się od rozwiązania optymalnego. Jest to bardzo trudne, ponieważ w ogólnym przypadku optymalne rozwiązanie nie jest znane.

Jedną z najciekawszych i jednocześnie najbardziej uniwersalnych heurystyk

jest idea algorytmów ewolucyjnych, które z dużym powodzeniem zostały zastosowane do rozwiązywania wielu problemów NP-trudnych. W rozdziale tym skonstruujemy algorytm ewolucyjny dla rozmytego problemu szeregowania, który spełnia następujące dodatkowe założenia:

1. $prec = \emptyset$,
2. nie jest dopuszczalny podział prac i przestoje maszyny.

Powyższe warunki oznaczają, że zbiór dopuszczalnych harmonogramów pokrywa się ze zbiorem wszystkich permutacji zbioru prac. Zauważmy, że prawie wszystkie NP-trudne problemy szeregowania, sformułowane w rozdziale drugim, spełniają te założenia. W rozdziale tym zaprezentujemy również wyniki eksperymentów przeprowadzonych dla jednego, wybranego problemu. Dla tego problemu porównamy efektywność algorytmu ewolucyjnego z przeszukiwaniem losowym i innymi prostymi heurystykami.

4.1 Algorytm ewolucyjny dla rozmytego problemu szeregowania

Idea algorytmu ewolucyjnego polega na generowaniu w każdej iteracji populacji rozwiązań problemu¹ $P(t) = \{p_1^t, \dots, p_N^t\}$. Każde rozwiązanie jest zakodowane w postaci pewnej struktury danych. Każde rozwiązanie p_i^t , $i = 1, \dots, N$, należące do populacji $P(t)$ jest oceniane za pomocą pewnej funkcji, która jest miarą jego dopasowania w populacji. Nową populację w iteracji $t + 1$ tworzy się przez wybór z bieżącej populacji rozwiązań najlepiej dopasowanych. Niektóre rozwiązania w nowo utworzonej populacji podlegają dalszej transformacji za pomocą operatorów genetycznych. Mogą to być operatory jednoargumentowe (mutacje), które wprowadzają niewielkie zmiany do rozwiązań i dwuargumentowe (krzyżowanie), w których nowe rozwiązania powstają poprzez pewną kombinację dwóch starych rozwiązań. Powyższy schemat przypomina w uproszczeniu ewolucję jaka zachodzi w świecie organizmów żywych. Po wykonaniu pewnej liczby iteracji możemy oczekiwać, że najlepsze uzyskane rozwiązanie będzie bliskie rozwiązaniu optymalnemu.

Ogólny schemat algorytmu ewolucyjnego przedstawia algorytm 16.

¹Populacja nie może być dokładnie utożsamiana ze zbiorem ponieważ niektóre rozwiązania mogą się w niej powtarzać

Algorytm 16 Ogólny schemat algorytmu ewolucyjnego

Require: Parametry problemu, parametry algorytmu ewolucyjnego

Ensure: p^* /* Przybliżone rozwiązanie problemu /*

```
1:  $t \leftarrow 0$ 
2: Wygeneruj losowo początkową populację  $P(t)$ 
3: Oceń każde rozwiązanie  $p_i^t \in P(t)$ 
4:  $p^* \leftarrow$  najlepsze rozwiązanie z  $P(t)$ 
5: while not warunek zakończenia do
6:    $t \leftarrow t + 1$ 
7:   Utwórz  $P(t)$ , wykonując selekcję rozwiązań z  $P(t - 1)$ 
8:   Przekształć  $P(t)$ , wykonując operacje krzyżowania
9:   Przekształć  $P(t)$ , wykonując operacje mutacji
10:  Oceń każde rozwiązanie  $p_i^t \in P(t)$ 
11:  if Ocena najlepszego rozwiązania z  $P(t)$  lepsza niż ocena  $p^*$  then
12:     $p^* \leftarrow$  najlepsze rozwiązanie z  $P(t)$ 
13:  end if
14: end while
15: return  $p^*$ 
```

W przypadku rozmytych problemów szeregowania populacja $P(t)$ składa się z N permutacji zbioru prac J , gdzie N jest parametrem algorytmu ewolucyjnego. Populację początkową $P(0)$ uzyskuje się generując N permutacji w sposób losowy.

Ocena każdego rozwiązania $p_i^t \in P(t)$ polega na obliczeniu dla niego wartości funkcji kosztu harmonogramu $F(p_i^t)$. Dopasowanie rozwiązania jest tym lepsze im mniejsza jest dla niego wartość tej funkcji.

W kroku 7 algorytmu tworzona jest nowa populacja poprzez selekcję rozwiązań z bieżącej populacji. W literaturze można spotkać wiele różnych sposobów dokonywania selekcji. Wszystkie jednak posiadają tę własność, że prawdopodobieństwo wyboru danego rozwiązania do nowej populacji jest tym większe im lepsze jest jego dopasowanie. W przypadku rozmytego problemu szeregowania do nowej populacji powinny najczęściej trafiać te permutacje, których koszt jest relatywnie najmniejszy. Zastosujemy metodę selekcji, która w literaturze ([30]) nosi nazwę **selekcji rankingowej**. Najpierw porządkujemy populację $P(t - 1)$ w ten sposób, że:

$$F(p_1^{t-1}) \geq \dots \geq F(p_N^{t-1})$$

Następnie każdej permutacji $p_i^{t-1} \in P(t - 1)$ przypisujemy następujące prawdopodobieństwo wyboru do nowej populacji:

$$Prob(p_i^{t-1}) = \frac{2i}{N(N+1)}.$$

Nową populację $P(t)$ tworzymy stosując tzw. metodę ruletki ([26]). W selekcji rankingowej każde rozwiązanie z populacji $P(t-1)$ ma szansę znaleźć się w populacji $P(t)$. Cechą charakterystyczną tej metody jest stosunkowo niewielka presja selekcyjna. Oznacza, że jedno rozwiązanie o bardzo dobrym dopasowaniu nie zdominuje szybko całej populacji.

W pracy ([26]) są podane pewne wyniki eksperymentalne dotyczące zastosowania algorytmów ewolucyjnych do klasycznych zadań szeregowania i do problemu komiwojażera (który ma podobną strukturę). Podane są między innymi operatory krzyżowania i mutacji, dla których uzyskano dobre wyniki eksperymentalne. W przypadku krzyżowania jest to operator krzyżowania z uporządkowaniem, zaproponowany w [8] i określony następująco:

Definicja 4.1. (Krzyżowanie z uporządkowaniem)

Dla dwóch permutacji p_1, p_2 i zadanych parametrów k, l , gdzie $1 \leq k < l \leq n$ zostaw bez zmian podsekwencje $(p_1(k), \dots, p_1(l))$ oraz $(p_2(k), \dots, p_2(l))$ i zmień uporządkowanie pozostałych prac w permutacji p_1 tak aby odpowiadało ono uporządkowaniu tych prac w permutacji p_2 i analogicznie wykonaj to samo dla permutacji p_2 .

Przykład: Niech $p_1 = (1, 2, 3, \underline{4, 5, 6, 7, 8, 9})$, $p_2 = (4, 5, 2, \underline{1, 8, 7, 6, 9, 3})$, $k = 4$, $l = 7$. W wyniku skrzyżowania p_1 i p_2 otrzymujemy: $o_1 = (2, 1, 8, \underline{4, 5, 6, 7, 9, 3})$ i $o_2 = (3, 4, 5, \underline{1, 8, 7, 6, 9, 2})$. $\square$

W kroku 8 algorytmu 16 jest wykonywana operacja krzyżowania. Najpierw z populacji $P(t)$ wybieramy rozwiązania, które będą podlegać krzyżowaniu. Wyboru dokonujemy generując dla każdego rozwiązania liczbę losową z przedziału $[0, 1]$. Jeżeli jest ona mniejsza od zadanego parametru $PCross$ (prawdopodobieństwo krzyżowania) to dane rozwiązanie będzie podlegać krzyżowaniu. W przeciwnym wypadku przechodzi ono bez zmian do następnego etapu (mutacji). Wybrane rozwiązania są losowo ustawiane w pary i krzyżowane z zastosowaniem krzyżowania z uporządkowaniem przy czym parametry k i l są generowane losowo.

Zdefiniujmy teraz (za [26]) dwa operatory mutacji, które zostały z powodzeniem zastosowane do klasycznych zadań szeregowania:

Definicja 4.2. (Wymiana prac) *Dla permutacji p i zadanych parametrów k i l , gdzie $1 \leq k, l \leq n$, $k \neq l$ zamień miejscami prace $p(k)$ i $p(l)$.*

Definicja 4.3. (Mieszanie podlist) *Dla permutacji p i zadanych parametrów k i l , gdzie $1 \leq k < l \leq n$ zmień w sposób losowy uporządkowanie prac w podsekwencji $(p(k), \dots, p(l))$.*

Wymiana prac jest mutacją, która powoduje niewielką zmianę w permutacji p . Mutacja polegająca na mieszaniu podlist może zmodyfikować tę permutację w znacznie większym stopniu (jeżeli $k = 1$ i $l = n$ to otrzymamy całkiem nową, losową permutację). W algorytmie ewolucyjnym zastosujemy oba operatory mutacji przy czym wyboru rodzaju mutacji będziemy dokonywać w sposób losowy. W kroku 9 algorytmu 16 dla każdego rozwiązania w populacji $P(t)$ generujemy liczbę losową z przedziału $[0, 1]$. Jeżeli jest ona mniejsza od zadanego parametru $PMut$ (prawdopodobieństwa mutacji), to rozwiązanie to zostaje poddane mutacji polegającej na mieszaniu podlist lub wymianie prac. Wyboru rodzaju mutacji oraz parametrów k i l za każdym razem dokonujemy losowo.

Algorytm ewolucyjny powinien zakończyć działanie jeżeli zostało znalezione rozwiązanie bliskie optymalnemu lub populacja składa się z takich samych (lub bardzo podobnych) rozwiązań. Algorytm ewolucyjny dla rozmytego problemu szeregowania kończy działanie jeżeli przez T następujących po sobie iteracji nie zostanie znalezione lepsze rozwiązanie. Liczba T jest parametrem algorytmu. Im większa jest wartość T tym większa jest liczba populacji i jednocześnie dłuższy czas obliczeń.

Algorytm ewolucyjny wymaga wprowadzenia następujących parametrów:

1. N - liczba rozwiązań w populacji,
2. $PCross$ - prawdopodobieństwo krzyżowania,
3. $PMut$ - prawdopodobieństwo mutacji,
4. T - liczba następujących po sobie iteracji w których nie zostało znalezione lepsze rozwiązanie.

Wszystkie parametry mają duże znaczenie dla działania algorytmu i ich wartości powinny być ustalone eksperymentalnie.

4.2 Inne heurystyki dla rozmytego problemu szeregowania

Algorytm ewolucyjny nie jest jedyną heurystyką, którą można zastosować do przybliżonego rozwiązywania problemów NP-trudnych. Zawsze warto badać jednocześnie kilka metod i porównywać otrzymane rozwiązania. W punkcie tym podamy kilka bardzo prostych heurystyk, które zostaną wykorzystane do porównania z algorytmem ewolucyjnym. Jedną z najprostszych heurystyk jakie można

skonstruować jest przeszukiwanie losowe. Polega ona na tym, że generuje się ciąg losowych permutacji o ustalonej długości i wybiera tę, dla której funkcja kosztu jest najmniejsza. Heurystyka ta z oczywistych powodów nie nadaje się do zastosowań praktycznych ponieważ prawdopodobieństwo ślepego trafienia na dobre rozwiązanie jest bardzo małe. Nadaje się ona jednak bardzo dobrze do wstępnej oceny jakości innych, bardziej wyrafinowanych metod. Dobra heurystyka musi dawać rozwiązania co najmniej lepsze od przeszukiwania losowego.

Dla rozmytych problemów szeregowania można podać kilka innych, bardzo prostych heurystyk. W szczególności jeżeli funkcja kosztu jest f -regularna i zależy od $\tilde{d}_i$, $i = 1, \dots, n$, to można próbować badać następujące heurystyki:

Definicja 4.4. HED

Utwórz permutację $p = (p(1), \dots, p(n))$ taką, że:

$$\frac{w_{p(i)}}{E(\tilde{d}_{p(i)}(p))} \geq \dots \geq \frac{w_{p(n)}}{E(\tilde{d}_{p(n)}(p))}$$

Definicja 4.5. HMD

Utwórz permutację $p = (p(1), \dots, p(n))$ taką, że:

$$\frac{w_{p(i)}}{\underline{d}_{p(i)} + \bar{d}_{p(i)}} \geq \dots \geq \frac{w_{p(n)}}{\underline{d}_{p(n)} + \bar{d}_{p(n)}}$$

Uzasadnienie dla zastosowania heurystyk HED i HMD dla problemów z f -regularną funkcją kosztu jest oczywiste. Najpierw powinny być wykonywane prace ważniejsze (dla których waga w_i jest większa) oraz te prace, dla których pożądany termin zakończenia jest najbliższy. Obie heurystyki mogą być wykorzystane do porównywania z innymi metodami.

4.3 Przykłady numeryczne dla $1|TR|\sum E(w_i\tilde{T}_i)$

W rozdziale drugim (punkt 2.11) pokazaliśmy, że rozmyty problem szeregowania $1|TG|\sum E(w_i\tilde{T}_i)$ jest silnie NP-trudny. Wynika z tego, że ogólniejszy problem $1|TR|\sum E(w_i\tilde{T}_i)$, w którym parametry są liczbami trapezowymi jest również silnie NP-trudny. W tym punkcie przetestujemy algorytm ewolucyjny oraz proste heurystyki podane w punkcie 4.2 na problemie $1|TR|\sum E(w_i\tilde{T}_i)$.

Jeżeli $\tilde{t}_i, \tilde{d}_i$ dla każdej pracy $i \in J$ są liczbami całkowitymi, to szczególnym przypadkiem problemu $1|FN|\sum E(w_i\tilde{T}_i)$ jest silnie NP-trudny problem $1||\sum w_iT_i$. Wynika to z faktu, że w takim przypadku $\tilde{C}_i(p), \tilde{T}_i(p)$, $i = 1, \dots, n$, są również całkowite i:

$$E(w_i\tilde{T}_i(p)) = w_i\tilde{T}_i = w_iT_i.$$

Oznacza to, że problemy $1||\sum w_i T_i$ oraz $1|TR|E(\sum w_i \tilde{T}_i)$ są szczególnymi przypadkami tego samego problemu $1|FN|E(\sum w_i T_i)$. Efektywność algorytmu ewolucyjnego dla problemu $1||\sum w_i T_i$ może być oczywiście inna niż dla problemu $1|TR|\sum E(w_i \tilde{T}_i)$, rozsądne jednak wydaje się przyjęcie założenia, że różnica ta nie będzie zbyt duża ponieważ funkcja kosztu jest dla obu problemów taka sama (różnią się one typem danych).

Dla klasycznego problemu szeregowania $1||\sum w_i T_i$ istnieje zbiór ponad stu przykładowych modeli dla 50 i 100 prac, dla których znane są rozwiązania optymalne lub bliskie optymalnemu. Wszystkie te przykładowe modele są dostępne w Internecie pod adresem mscmga.ms.ic.ac.uk/pub/ i można wykorzystać je do przetestowania algorytmu ewolucyjnego zdefiniowanego w punkcie 4.1. W dalszym ciągu będziemy nazywać te modele modelami testowymi.

Do przetestowania algorytmu ewolucyjnego wykorzystamy losowo wybrany podzbiór 20 ze 125 modeli testowych (dla 50 i 100 prac), w których $\tilde{t}_i, \tilde{d}_i$ są liczbami całkowitymi oraz 5 losowo wygenerowanych modeli, w których $\tilde{t}_i, \tilde{d}_i$ są rozmytymi liczbami trapezowymi. Każdy model, w którym parametry są rozmyte został wygenerowany za pomocą następującej procedury:

1. $\tilde{t}_i = (TD, TU, T\alpha, TBETA), i = 1, \dots, n$, gdzie²:
 $TD = rand([1, 99]),$
 $TU = rand([TD, \min\{TD + 4, 99\}]),$
 $T\alpha = rand([1, TD]),$
 $TBETA = rand([1, 100 - TU]),$
2. $\tilde{d}_i = (DD, DU, D\alpha, DBETA), i = 1, \dots, n$, gdzie:
 $DD = rand([1, 1.3P - 1]),$ gdzie $P = \frac{1}{2} \sum_{i=1}^n (\underline{t}_i + \bar{t}_i),$
 $DU = rand([DD, DD + 4]),$
 $D\alpha = rand([1, DD]),$
 $DBETA = rand([1, 1.3P - DU]),$
3. $w_i = rand([1, 10]), i = 1, \dots, n.$

Modele testowe będziemy oznaczać symbolem K_i , gdzie indeks $i \in \{1, \dots, 125\}$ oznacza numer modelu w pliku testowym. Modele z parametrami rozmytymi będziemy oznaczać symbolem F_i gdzie $i = 1, \dots, 5$. Dla każdego modelu uruchamiamy algorytm ewolucyjny dziesięć razy za każdym razem startując z innej, losowo wygenerowanej populacji. Dla każdego modelu wyznaczymy najlepszy,

² $rand([a, b])$ -całkowita liczba losowa z przedziału $[a, b]$

najgorszy oraz średni koszt znaleziony w tych dziesięciu przebiegach. Dla modeli testowych obliczamy ile razy algorytm ewolucyjny znalazł optymalne rozwiązanie (w 10 przebiegach). Wyznamy również średni czas obliczeń dla każdego modelu.

Ustalamy następujące parametry algorytmu ewolucyjnego:

1. $N = 100$ dla 50 prac, $N = 150$ dla 100 prac
2. $PCross = 0.4$, $PMut = 0.4$
3. $T=300$

Wyniki uzyskane za pomocą algorytmu ewolucyjnego porównamy z wynikami uzyskanymi po zastosowaniu heurystyk *HED* i *HMD* oraz najlepszym rozwiązaniem znalezionym podczas przeszukiwania losowego. Czas przeszukiwania losowego ustalamy dla każdego modelu jako dziesięciokrotność średniego czasu obliczeń wykonywanych przez algorytm ewolucyjny. Uzyskane wyniki są przedstawione w tabelach 4.1-4.4. Wszystkie obliczenia zostały wykonane na komputerze z procesorem PIII i 128MB pamięci RAM.

Nr	OPT	ROPT	BST	AVG	WST	TM(s.)
K_1	2134	8	2134(0.00%)	2160.2(1.22%)	2346(9.93%)	0.719
K_2	1996	0	1998(0.10%)	2011.6(0.78%)	2062(3.30%)	0.692
K_5	1518	4	1518(0.00%)	1569(3.39%)	1604(5.66%)	0.725
K_6	26276	0	26281(0.02%)	26389.60(0.43%)	26509(0.88%)	0.993
K_{13}	45383	3	45383(0.00%)	45519.8.20(0.30%)	45967(1.28%)	1.13
K_{22}	150800	10	150800(0.00%)	150800(0.00%)	150800(0.00%)	1.43
K_{25}	240179	10	240179(0.00%)	240179(0.00%)	240179(0.00%)	1.477
K_{30}	22	8	22(0.00%)	36.6(66.36%)	95(331.82%)	0.623
K_{40}	26423	10	26423(0.00%)	26423(0.00%)	26423(0.00%)	1.21
K_{47}	133289	5	133289(0.00%)	133307(0.01%)	133325(0.02%)	1.46
K_{55}	0	10	0	0	0	0.432
K_{59}	3770	1	3770(0.00%)	4178.7(10.84%)	4465(18.43%)	1.36
K_{68}	92756	3	92756(0.00%)	93164.10(0.43%)	93690(1.00%)	1.75
K_{74}	135394	2	135394(0.00%)	135468.60(0.05%)	135535(0.10%)	1.54
K_{82}	4879	2	4879(0.00%)	4933.5(1.11%)	5010(2.68%)	1.56
K_{84}	508	6	508(0.00%)	566.3(11.47%)	982(93.30%)	1.16
K_{89}	28846	0	28849(0.01%)	28905(0.20%)	29170(1.123%)	1.77
K_{96}	177909	4	177909(0.00%)	177949(0.02%)	177975(0.03%)	1.46
K_{113}	35106	5	35106(0.00%)	35428.2(0.91%)	36014(2.58%)	1.87
K_{119}	106043	2	106043(0.00%)	106096(0.04%)	106160(0.11%)	1.85
F_1	?	-	35750.7	35750.7	35750.7	10.40
F_2	?	-	35877.9	35878.2	35881.2	12.37
F_3	?	-	26381.9	26382.3	26385.9	12.69
F_4	?	-	52311	52315.3	52324	14.30
F_5	?	-	30538.9	30540.6	30555.6	11.86

Tab. 4.1: Wyniki uzyskane za pomocą algorytmu ewolucyjnego dla 50 prac. W kolumnie Nr znajduje się oznaczenie modelu. W kolumnie OPT znajduje się optymalne rozwiązanie problemu dla modeli testowych. W kolumnie ROPT podane jest ile razy algorytm ewolucyjny znalazł optymalne rozwiązanie w przypadku modeli testowych (w których parametry są liczbami całkowitymi). W kolumnach BST, AVG, WST znajdują się odpowiednio najlepszy, średni i najgorszy uzyskany koszt w 10 przebiegach algorytmu oraz odchylenie procentowe tego kosztu od optymalnego (w nawiasie). W kolumnie TM znajduje się średni czas obliczeń.

Nr	EWOL	HED	HMD	RAND
K_1	2134	5050(136.64%)	5050(136.64%)	3184(49.20%)
K_2	1998	3407(70.52%)	3407(70.52%)	2845(42.39%)
K_5	1518	5821(283.41%)	5821(283.41%)	3570(135.17%)
K_6	26281	41362(57.38%)	41362(57.38%)	41582(58.22%)
K_{13}	45383	67459(48.64%)	67459(48.64%)	67875(49.56%)
K_{22}	150800	240749(59.65%)	240749(59.65%)	187540(24.36%)
K_{25}	240179	357740(48.94%)	357740(48.94%)	292843(%)
K_{30}	22	2744(12372.73%)	2744(12372.73%)	598(2618.18%)
K_{40}	26423	38424(45.42%)	38424(45.42%)	64715(144.91%)
K_{47}	133289	240493(80.43%)	240493(80.43%)	197726(48.34%)
K_{55}	0	6781	6781	2309
K_{59}	3770	12710(237.1%)	12710(237.1%)	16337(333.3%)
K_{68}	92756	145433(36.22%)	145433(36.22%)	151844(38.91%)
K_{74}	135394	262735(48.47%)	262735(48.47%)	192640(29.72%)
K_{82}	4879	21708(344.93%)	21708(344.93%)	32414((564.33%)
K_{84}	508	16717(3190.75%)	16717(3190.75%)	20997(4033.26%)
K_{89}	28849	52173(44.71%)	52173(44.71%)	83746(65.55%)
K_{96}	177909	258596(45.35%)	258596(45.35%)	230207(29.39%)
K_{113}	35106	56378(60.59%)	56378(60.59%)	88271(151.44%)
K_{119}	106120	192129(81.05%)	192129(81.05%)	150881(42.17%)
F_1	35775.4	45329.4(26.70%)	44466.8(24.29%)	68403.8(47.70%)
F_2	35887.6	47863(33.36%)	47240(31.63%)	65085(44.86%)
F_3	26382.7	37640.3(42.67%)	37017(40.31%)	53063.4(50.28%)
F_4	52316.6	62290.7(19.06%)	62407.7(19.28%)	78320(33.20%)
F_5	30540.4	37938.8(24.22%)	39446.3(29.16%)	54123(43.57%)

Tab. 4.2: Porównanie algorytmu ewolucyjnego z heurystykami HED, HMD oraz przeszukiwaniem losowym dla 50 prac. W kolumnie Nr znajduje się oznaczenie modelu. W kolumnie EWOL znajduje się koszt najlepszego rozwiązania uzyskanego za pomocą algorytmu ewolucyjnego(w 10 przebiegach). W kolumnach HED i HMD znajdują się koszty rozwiązań uzyskane za pomocą tych heurystyk. W kolumnie RAND znajduje się koszt najlepszego rozwiązania znalezione w wyniku przeszukiwania losowego. W nawiasach podane są odchylenia procentowe od najlepszego kosztu znalezione przez algorytm ewolucyjny.

Nr	OPT	ROPT	BST	AVG	WST	TM(s.)
K_1	5988	2	5988(0.00%)	6074.5(1.44%)	6243(4.25%)	3.0
K_4	5011	3	5011(0.00%)	5244.1(4.65%)	5624(12.23%)	3.64
K_6	58258	1	58258(0.00%)	58413(0.266%)	58988(1.25%)	3.85
K_{11}	181649	0	181839(0.10%)	182775.1(0.61%)	183172(0.83%)	4.61
K_{14}	157476	1	157476(0.00%)	157682(0.13%)	157878(0.25%)	5.24
K_{21}	898925	2	898925(0.00%)	899002(0.00%)	899105(0.01%)	6.23
K_{29}	480	0	554(15.42%)	585.7(22.02%)	847(76.46%)	7.50
K_{33}	32964	1	32964(0.00%)	33102.8(0.42%)	33267(0.92%)	2.43
K_{44}	360193	0	360220(0.00%)	360676.3(0.13%)	362033(0.51%)	8.9
K_{48}	748988	1	748988(0.00%)	749000(0.00%)	749006(0.00%)	8.79
K_{54}	0	10	0	0	0	1.2
K_{56}	9046	0	9510(5.12%)	9875.2(9.16%)	10969(21.25)	3.27
K_{59}	7965	1	7965(0.00%)	8054.1(1.11%)	8238(3.43)	3.20
K_{66}	243872	0	243929(0.02%)	244124(0.1%)	244389(0.21%)	10.49
K_{73}	623429	2	623429(0.00%)	623427(0.00%)	623445(0.00%)	8.09
K_{75}	575274	1	575274(0.00%)	575299(0.00%)	575326(0.00%)	8.85
K_{81}	1400	0	1679(19.92%)	2310.4(65.02%)	2890(106.42%)	4.07
K_{84}	136	10	136(0.00%)	136(0.00%)	136(0.00%)	2.43
K_{88}	55544	0	56063(0.93%)	56123.2(1.04%)	56208(1.19%)	9.49
K_{94}	273993	0	274083(0.03%)	274461(0.17%)	275255(0.46%)	9.57
F_1	?	?	118690	118703	118717	79.79
F_2	?	?	200505	200521	200551	84.85
F_3	?	?	195800	195828	195872	77.01
F_4	?	?	97370	97371.3	97376.3	63.61
F_5	?	?	172011	172021	172036	91.67

Tab. 4.3: Wyniki uzyskane za pomocą algorytmu ewolucyjnego dla 100 prac. W kolumnie Nr znajduje się oznaczenie modelu. W kolumnie OPT znajduje się najlepsze znane do tej pory rozwiązanie problemu dla modeli testowych. W kolumnie ROPT podane jest ile razy algorytm ewolucyjny znalazł najlepsze znane rozwiązanie w przypadku modeli testowych. W kolumnach BST, AVG, WST znajdują się odpowiednio najlepszy, średni i najgorszy uzyskany koszt w 10 przebiegach algorytmu oraz odchylenie procentowe tego kosztu od najlepszego znanego (w nawiasie). W kolumnie TM znajduje się średni czas obliczeń.

Nr	EWOL	HED	HMD	RAND
K_1	5988	14853(148.05%)	14853(148.05%)	19330(222.8%)
K_4	5011	20375(306.61%)	20375(306.61%)	15668(212.67%)
K_6	58258	109646(88.21%)	109646(88.21%)	135467(132.53%)
K_{11}	181839	284573(56.50%)	284573(56.50%)	323556(77.94%)
K_{14}	157476	181193(15.06%)	181193(15.06%)	295898(87.90%)
K_{21}	898925	1400000(55.74%)	1400000(55.74%)	1267350(40.99%)
K_{29}	554	20095(3527%)	20095(3527%)	124339(22343%)
K_{33}	32964	93956(185.03%)	93956(185.03%)	126449(283.6\$)
K_{44}	360220	582228(61.63%)	582228(61.63%)	632340(75.54%)
K_{48}	748988	1192100(59.16%)	1192100(59.16%)	1081590(44.41%)
K_{54}	0	14807	14807	33783
K_{56}	9510	54125(469.1%)	54125(469.1%)	124167(1205%)
K_{59}	7965	65612(723.75%)	65612(723.75%)	97744(1127%)
K_{66}	243929	461599(89.23%)	461599(89.23%)	494470(102.7%)
K_{73}	623429	855786(37.27%)	855786(37.27%)	914892(46.75%)
K_{75}	575274	874785(52.06%)	874785(52.06%)	852076(48.12%)
K_{81}	1679	68226(3963%)	68226(3963%)	165114(9734%)
K_{84}	136	54743(40152.21%)	54743(40152.21%)	115293
K_{88}	56063	125692(124.2%)	125692(124.2%)	308335(450%)
K_{94}	274083	544391(98.68%)	544391(98.68%)	591796(115.9%)
F_1	118830	156406(31.62%)	155512(30.87%)	250592(110.8%)
F_2	200630	236330(17.79%)	230717(15.00%)	416030(107.36%)
F_3	195940	235543(20.21%)	246488(25.80%)	358810(83.12%)
F_4	97526	136931(40.40%)	138750(42.27%)	276373(183.38%)
F_5	172221	224147(30.15%)	230402(33.78%)	328087(90.50%)

Tab. 4.4: Porównanie algorytmu ewolucyjnego z heurystykami HED, HMD oraz przeszukiwaniem losowym dla 100 prac. W kolumnie Nr znajduje się oznaczenie modelu. W kolumnie EWOL znajduje się koszt najlepszego rozwiązania uzyskanego za pomocą algorytmu ewolucyjnego(w 10 przebiegach). W kolumnach HED i HMD znajdują się koszty rozwiązań uzyskane za pomocą tych heurystyk. W kolumnie RAND znajduje się koszt najlepszego rozwiązania znalezione w wyniku przeszukiwania losowego. W nawiasach podane są odchylenia procentowe od najlepszego kosztu znalezione przez algorytm ewolucyjny.

Oceniając uzyskane wyniki można powiedzieć, że algorytm ewolucyjny działa bardzo dobrze dla modeli testowych w przypadku gdy zbiór J składa się z 50 prac. Tylko dla trzech z dwudziestu modeli testowych algorytm nie znalazł optymalnego rozwiązania (zostały znalezione rozwiązania bardzo bliskie optymalnemu). Dla pozostałych modeli optymalne rozwiązanie zostało uzyskane przynajmniej raz (w 10 przebiegach). Na ogólną liczbę 200 przebiegów optymalne rozwiązanie zostało znalezione 87 razy co daje średnio 43.5% pewności, że algorytm w pojedynczym przebiegu znajdzie optymalne rozwiązanie.

W przypadku gdy zbiór J składa się ze 100 prac, efektywność algorytmu dla modeli testowych jest trochę gorsza. Dla ośmiu z dwudziestu modeli testowych algorytm nie znalazł optymalnego rozwiązania. Na ogólną liczbę 200 przebiegów optymalne rozwiązanie zostało znalezione 35 razy co daje średnio 17.5% pewności, że algorytm w pojedynczym przebiegu znajdzie optymalne rozwiązanie. Zauważmy, że tylko dla trzech modeli testowych najlepsze znalezione rozwiązanie przez algorytm ewolucyjny jest gorsze o więcej niż 1% od rozwiązania optymalnego.

Algorytm ewolucyjny jest zdecydowanie lepszy niż przeszukiwanie losowe. Interesujący jest fakt, że heurystyki HED(HMD) dla modeli testowych działają bardzo słabo. W wielu przypadkach są gorsze niż przeszukiwanie losowe. W przypadku modeli rozmytych sytuacja jest odwrotna. Heurystyki HED i HMD dają rozwiązania średnio 40% gorsze niż algorytm ewolucyjny co jest wynikiem znacznie lepszym niż przeszukiwanie losowe. Można postawić hipotezę, że im bardziej rozmyte są parametry modelu tym lepsze rozwiązania dają obie te heurystyki, przy czym nie można jednoznacznie powiedzieć, która z nich jest lepsza.

Zakładając, że efektywność algorytmu ewolucyjnego jest podobna dla modeli rozmytych, można z powodzeniem stosować ten algorytm w praktyce dla problemu $1||\sum E(w_i\tilde{T}_i)$. Czas działania algorytmu jest na tyle mały, że nie stanowi poważniejszego ograniczenia. Przeprowadzone eksperymenty pokazały, że należy zawsze uruchomić algorytm kilka (lub nawet kilkanaście) razy i wybrać najlepsze uzyskane rozwiązanie. Istnieje wówczas duża pewność, że koszt uzyskanego rozwiązania będzie tylko nieznacznie różnić się od kosztu rozwiązania optymalnego.

Algorytm ewolucyjny wymaga podania czterech parametrów, które mają znaczący wpływ na jego efektywność. Parametry te powinny zostać ustalone eksperymentalnie. Przedstawimy teraz analizę wpływu parametrów na uzyskane wyniki dla modeli F_1 w przypadku 50 i 100 prac. Uzyskane wyniki pozwolą na sfor-

mułowanie pewnych ogólnych zaleceń dotyczących ustalania parametrów algorytmu ewolucyjnego dla problemu $1||\sum E(w_i\tilde{T}_i)$.

Przeprowadzimy dwa eksperymenty:

1. Dla ustalonych N oraz T będziemy zmieniać prawdopodobieństwa krzyżowania i mutacji od 0.1 do 1 z krokiem 0.1. Dla każdej kombinacji tych prawdopodobieństw algorytm zostanie uruchomiony 10 razy i zostanie obliczona wartość średnia uzyskanych kosztów.
2. Dla ustalonych prawdopodobieństw krzyżowania i mutacji będziemy zmieniać parametry N oraz T od 20 do 200 z krokiem 20. Dla każdej kombinacji wartości N, T algorytm zostanie uruchomiony 10 razy i zostanie obliczona wartość średnia uzyskanych kosztów oraz średni czas obliczeń.

Jako miarę efektywności algorytmu dla ustalonych parametrów przyjmiemy procentowe odchylenie uzyskanego średniego kosztu od kosztu najlepszego rozwiązania jakie udało się w ogóle uzyskać w przypadku modeli F_1 . Dla 50 prac najlepszy znaleziony koszt wynosi 35750.7 a dla 100 prac 118690.

P. Krzyżowania	1.0	0.01	<u>0.00</u>	0.01	0.07	0.18	1.14	2.75	6.54	9.45	12.05
	0.9	0.04	0.01	<u>0.00</u>	0.02	0.18	0.59	1.65	2.79	6.63	5.50
	0.8	0.04	0.01	0.02	0.01	0.14	0.45	0.99	1.31	2.19	1.77
	0.7	0.01	0.01	<u>0.00</u>	0.01	0.07	0.37	0.75	0.90	1.31	1.73
	0.6	0.03	0.02	0.02	<u>0.00</u>	0.05	0.21	0.54	0.69	1.00	0.72
	0.5	0.01	0.02	0.02	0.01	0.04	0.20	0.35	0.66	0.82	0.63
	0.4	0.02	0.01	<u>0.00</u>	<u>0.00</u>	0.03	0.15	0.34	0.32	0.38	0.47
	0.3	0.05	0.01	0.01	<u>0.00</u>	0.07	0.09	0.16	0.39	0.27	0.35
	0.2	0.01	0.02	0.02	<u>0.00</u>	0.02	0.09	0.17	0.16	0.32	0.29
	0.1	0.03	<u>0.00</u>	0.01	0.01	0.03	0.11	0.16	0.17	0.13	0.27
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
		P. Mutacji									

Rys. 4.1: Wpływ prawdopodobieństw krzyżowania i mutacji na efektywność algorytmu ewolucyjnego w przypadku 50 prac. Wartości parametrów N i T są ustalone i wynoszą odpowiednio 100 i 200.

Eksperyment pokazał, że prawdopodobieństwo mutacji ma duży wpływ na efektywność algorytmu. Prawdopodobieństwo to nie może być zbyt duże. Jego wartość ustalona na powyżej 0.5 znacznie pogarsza efektywność algorytmu. Prawdopodobieństwo krzyżowania ma wyraźnie mniejsze znaczenie - jednak wydaje się, że duża wartość tego parametru również pogarsza efektywność algorytmu. Z powyższego rysunku wynika, że prawdopodobieństwo mutacji powinno należeć do przedziału (0.1, 0.4) a prawdopodobieństwo krzyżowania do przedziału (0.1, 0.6).

P. Krzyżowania	1.0	0.03	0.03	0.01	0.04	0.56	3.05	7.51	12.5	18.3	22.0
	0.9	0.03	0.02	0.01	0.02	0.32	1.53	3.58	5.68	9.25	15.4
	0.8	0.04	0.01	0.01	0.01	0.35	1.59	2.28	3.99	3.69	4.69
	0.7	0.02	0.01	0.01	0.02	0.17	0.82	0.84	1.71	2.31	3.03
	0.6	0.09	0.02	0.03	0.01	0.13	0.37	1.26	0.82	1.94	1.32
	0.5	0.03	0.02	0.01	0.01	0.05	0.44	0.40	0.74	0.66	0.69
	0.4	0.03	0.02	0.01	0.01	0.06	0.23	0.44	0.60	0.59	0.51
	0.3	0.02	0.02	0.01	0.01	0.05	0.17	0.24	0.48	0.79	0.56
	0.2	0.03	0.05	0.02	0.02	0.04	0.12	0.11	0.32	0.66	0.42
	0.1	0.04	0.02	0.02	0.01	0.03	0.08	0.14	0.15	0.45	0.37
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
		P. Mutacji									

Rys. 4.2: Wpływ prawdopodobieństw krzyżowania i mutacji na efektywność algorytmu ewolucyjnego w przypadku 100 prac. Wartości parametrów N i T są ustalone i wynoszą odpowiednio 150 i 200.

W przypadku modelu dla 100 prac uzyskane wyniki są takie same jak dla 50 prac. Prawdopodobieństwo mutacji powinno należeć do przedziału $(0.1, 0.4)$ a prawdopodobieństwo krzyżowania do przedziału $(0.1, 0.6)$.

200	0.15	0.07	0.03 10.9	0.03	0.01	0.01	0.02	0.00	0.01	0.00
180	0.09	0.04 9.12	0.03	0.02	0.02	0.02	0.00	0.01	0.01	0.01
160	0.28	0.01 7.9	0.01	0.05	0.03	0.02	0.00	0.01	0.00	0.00
140	0.17	0.07	0.02 8.4	0.03	0.00	0.00	0.01	0.02	0.01	0.01
120	0.15	0.06	0.04 7.3	0.02	0.05	0.03	0.01	0.00	0.01	0.01
100	0.25	0.06	0.07	0.03 6.7	0.00	0.04	0.02	0.02	0.00	0.02
80	0.23	0.09	0.06	0.05	0.01 6.4	0.03	0.01	0.05	0.01	0.00
60	0.37	0.11	0.06	0.05	0.07	0.06	0.03 6.3	0.04	0.03	0.03
40	0.6	0.12	0.12	0.1	0.07	0.07	0.03 4.8	0.03 4.8	0.05 5.0	0.02 5.7
20	1.57	0.56	0.20	0.16	0.12	0.1	0.07	0.1	0.06	0.06
	20	40	60	80	100	120	140	160	180	200

T

Rys. 4.3: Wpływ liczebności populacji N i parametru T na efektywność algorytmu ewolucyjnego w przypadku 50 prac. Prawdopodobieństwa krzyżowania i mutacji są ustalone i wynoszą odpowiednio 0.4 i 0.4. W niektórych kratkach, poniżej odchyłeń procentowych, podany jest również średni czas obliczeń.

Eksperyment pokazał, że parametry N oraz T są ze sobą ściśle powiązane. Dla danego rozmiaru populacji istnieje najlepszy zakres wartości parametru T . Na przykład jeżeli rozmiar populacji wynosi 100 to wartość parametru T powinna wynosić 80. Zauważmy, że większa wartość T nie powoduje istotnego zwiększenia efektywności algorytmu a jedynie wydłuża czas obliczeń. Biorąc pod uwagę właśnie czas obliczeń można powiedzieć, że korzystniejsze jest ustalanie niewielkiej liczebności populacji i większej wartości parametru T . Da $N = 40$ i $T = 140$ algorytm ma podobną efektywność jak dla $N = 200$ i $T = 60$ ale działa ponad dwa razy szybciej.

200	0.16	0.03 62.4	0.01	0.02	0.01	0.01	0.02	0.01	0.01	0.01
180	0.14	0.02 51.6	0.02	0.02	0.02	0.02	0.01	0.01	0.02	0.01
160	0.16	0.03 46.0	0.02	0.02	0.03	0.02	0.01	0.01	0.02	0.01
140	0.19	0.04 39.6	0.03	0.03	0.02	0.02	0.01	0.02	0.02	0.01
120	0.20	0.05 35.9	0.03	0.02	0.02	0.02	0.02	0.02	0.02	0.01
100	0.30	0.07 33.6	0.05	0.03	0.03	0.02	0.02	0.02	0.02	0.02
80	0.52	0.08 30.5	0.05	0.04 30.5	0.03 32.0	0.03	0.03	0.02	0.02	0.02
60	0.58	0.15	0.08	0.06	0.05	0.03 28.9	0.02	0.03	0.03	0.02
40	1.14	0.27	0.14	0.08	0.09	0.04 26.5	0.04 26.6	0.03 27.5	0.03 27.7	0.03 30.0
20	3.91	0.84	0.33	0.24	0.18	0.16	0.10	0.08	0.07	0.06
	20	40	60	80	100	120	140	160	180	200

Rys. 4.4: Wpływ liczebności populacji N i parametru T na efektywność algorytmu ewolucyjnego w przypadku 100 prac. Prawdopodobieństwa krzyżowania i mutacji są ustalone i wynoszą odpowiednio 0.4 i 0.4. W niektórych kratkach, poniżej odchyłeń procentowych, podany jest również średni czas obliczeń.

W przypadku 100 prac wnioski są analogiczne jak dla 50 prac. Najlepszą efektywność uzyskuje się, przyjmując $N = 40$ i $T \in [120, 140]$.

Zakończenie

Celem pracy była analiza różnych problemów szeregowania prac na pojedynczej maszynie, w których parametry są zadane w sposób nieprecyzyjny.

W rozdziale drugim przedstawiono nowe podejście do tego typu problemów, w którym parametry są liczbami rozmytymi, interpretowanymi jako rozkłady możliwości. Podano definicję funkcji kosztu harmonogramu, która zależy od rozmytych czasów zakończenia prac i przyjmuje wartości rzeczywiste. Określono ważną klasę funkcji f -regularnych i udowodniono fakt, że problem szeregowania, w którym koszt harmonogramu jest funkcją f -regularną ma zawsze optymalne rozwiązanie, które ponadto jest pewną permutacją zbioru prac.

Największa część rozdziału drugiego została poświęcona szczególnym przypadkom rozmytych problemów szeregowania. Do konstrukcji funkcji kosztu harmonogramu dla tych problemów wykorzystano indeksy Dubois i Prade oraz wartość średnią liczby rozmytej. Pojęcia te są dobrze znane i powszechnie stosowane w teorii i zastosowaniach zbiorów rozmytych. Wszystkie problemy, zdefiniowane w tym rozdziale są nowe, dlatego duże znaczenie miała ich klasyfikacja ze względu na złożoność obliczeniową. Dla każdego z tych problemów został skonstruowany wielomianowy algorytm albo przedstawiono dowód NP-trudności.

Jednym z najważniejszych rezultatów uzyskanych w rozdziale drugim jest uogólnienie klasycznego algorytmu Lawlera na przypadek rozmyty. Za pomocą tego algorytmu można efektywnie rozwiązywać problemy należące do szerokiej klasy problemów szeregowania z rozmytymi parametrami.

Duże znaczenie teoretyczne i praktyczne mają również dowody NP-trudności problemów, które są charakterystyczne dla podejścia rozmytego. Problemy te są łatwe w przypadku gdy parametry są zdeterminowane, natomiast stają się trudne jeżeli parametry mogą być rozmyte. Należy podkreślić, że przy okazji dowodzenia NP-trudności tych problemów, została pokazana również NP-trudność kilku nowych, interesujących problemów, w których parametry są zdeterminowane.

W rozdziale trzecim przedstawiono metodę oceny optymalności harmono-

gramu w problemie szeregowania na pojedynczej maszynie z rozmytymi parametrami. Skonstruowano efektywne algorytmy dla kilku szczególnych przypadków takiego problemu. Ważnym rezultatem teoretycznym jest sformułowanie warunków koniecznych i wystarczających optymalności zadanej permutacji w problemie $1|prec|f_{max}$. Warunki te zostały wykorzystane w pracy do konstrukcji wspomnianych algorytmów. Mają one jednak znaczenie szersze i mogą być wykorzystane również do innych celów, na przykład do analizy wrażliwości rozwiązania na zmiany parametrów w problemie $1|prec|f_{max}$.

W tabeli 4.5 podano wszystkie zbadane w pracy problemy, które są łatwe z obliczeniowego punktu widzenia.

Problem	Złożoność
$1 prec, L - R \max\{Poss(Ness)(\tilde{C}_i > \tilde{d}_i)\}$	$O(n^2 \log \varepsilon)$
$1 prec, TR \max\{Poss(Ness)(\tilde{C}_i > \tilde{d}_i)\}$	$O(n^2)$
$1 prec, L - R \max\{E(w_i \tilde{L}_i)\}$	$O(n^2)$
$1 prec, PL - PR \max\{E(w_i \tilde{T}_i)\}$	$O(n^2)$
$1 L - R \max\{E(\tilde{L}_i)\}$	$O(n \log n)$
$1 L - R Poss(\max\{w_i \tilde{L}_i\} > \tilde{G})$	$O(n^2 \log \varepsilon)$
$1 L - R Poss(\sum w_i \tilde{C}_i > \tilde{G})$	$O(n \log n \log \varepsilon)$
$1 sp - graph, L - R \sum E(w_i \tilde{C}_i)$	$O(n \log n)$
$i - 1 prec \max\{w_i L_i\}$	Programowanie liniowe
$i - 1 prec \max\{w_i T_i\}$	Programowanie liniowe
$i - 1 prec L_{max}$	$O(n^3)$
$i - 1 prec T_{max}$	$O(n^3)$

Tab. 4.5: Zestawienie problemów zbadanych w pracy, które mają złożoność wielomianową. Parametr n oznacza ilość prac, natomiast parametr ε oznacza bezwzględny błąd w ocenie kosztu optymalnego rozwiązania

Dla każdego z nich został skonstruowany odpowiedni algorytm. Dla niektórych z tych problemów złożoność algorytmu zależy również od przyjętej dokładności w obliczeniu wartości funkcji kosztu. Problemy $i - 1|prec| \max\{w_i L_i\}$ i $i - 1|prec| \max\{w_i T_i\}$ zostały sprowadzone do rozwiązania pewnego zadania programowania liniowego. Zatem złożoność obliczeń zależy od metody, która zostanie zastosowana do rozwiązania takiego problemu. Ponieważ wykazano, że zadanie programowania liniowego należy do wielomianowej klasy złożoności ([22],[21]), więc złożoność obu problemów szeregowania jest również wielomianowa. Wszystkie problemy przedstawione w tabeli 4.5 mogą więc być efektywnie rozwiązane nawet dla dużej liczby prac.

W tabeli 4.6 są przedstawione problemy, których NP-trudność została wykazana w pracy. Interesujący jest fakt, że część tych problemów staje się łatwa, jeżeli parametry są zdeterminowane.

Problem	Uwagi
$1 TG \max\{Ness(\tilde{C}_i \neq \tilde{d}_i)\}$	Funkcja kosztu nie jest f -regularna
$1 FN E(\max\{\tilde{T}_i\})$	NP-trudny jeżeli $\tilde{t}_i$ zdeterminowane a $\tilde{d}_i$ rozmyte.
$1 FN E^N(\max\{\tilde{T}_i\})$	Równoważny $1 T_{max}$ jeżeli $\tilde{t}_i$ oraz $\tilde{d}_i$ zdeterminowane.
$1 FN E^N(\max\{\tilde{L}_i\})$	NP-trudny jeżeli $\tilde{t}_i$ zdeterminowane, $\tilde{d}_i$ rozmyte a $N \leq n$.
$1 FN E(\max\{\tilde{L}_i\})$	Równoważny $1 T_{max}$ jeżeli $\tilde{t}_i$ oraz $\tilde{d}_i$ zdeterminowane.
$1 FN E^N(\max\{\tilde{L}_i\})$	NP-trudny jeżeli $\tilde{t}_i$ zdeterminowane a $\tilde{d}_i$ rozmyte.
$1 FN \sum Poss(Ness)(\tilde{C}_i > \tilde{d}_i)$	Równoważny $1 L_{max}$ jeżeli $\tilde{t}_i$ oraz $\tilde{d}_i$ zdeterminowane.
$1 prec, FN \sum E(w_i \tilde{C}_i)$	NP-trudny jeżeli $\tilde{t}_i$ zdeterminowane, $\tilde{d}_i$ rozmyte a $N \leq n$.
$1 TG \sum E(w_i \tilde{T}_i)$	Równoważny $1 \sum U_i$ jeżeli $\tilde{t}_i$ oraz $\tilde{d}_i$ zdeterminowane.
$1 MT(\sigma, k, n)$	Silnie NP-trudny
$1 ML$	Silnie NP-trudny. NP-trudny jeżeli $w_i = 1$
	Parametry całkowite.
	NP-trudny dla każdego ustalonego $k = 1, \dots, n$
	Parametry całkowite

Tab. 4.6: Zestawienie problemów zbadanych w pracy, dla których została pokazana NP-trudność. W dwóch ostatnich problemach parametry są zdeterminowane.

Dla żadnego z przedstawionych w tablicy 4.6 NP-trudnych problemów prawdopodobnie nie istnieje algorytm wielomianowy³, co oznacza, że dla dużej liczby prac nie można w rozsądnym czasie znaleźć optymalnego rozwiązania. W związku z tym konieczne jest konstruowanie metod heurystycznych, które w rozsądnym czasie dają rozwiązania bliskie optymalnemu.

W rozdziale trzecim zaproponowano algorytm ewolucyjny, który został przetestowany na silnie NP-trudnym problemie $1|TR|\sum E(w_i \tilde{T}_i)$. Algorytm ten daje bardzo dobre rozwiązania w szczególnym przypadku tego problemu, w którym parametry są zdeterminowane (tj. w problemie $1||\sum w_i T_i$). Można się spodziewać, że algorytm jest równie efektywny dla ogólniejszego problemu, w którym parametry są rozmyte.

³Algorytm wielomianowy dla problemu NP-trudnego nie istnieje jeżeli prawdziwa jest hipoteza $P \neq NP$. Do chwili obecnej jest ona jednak wciąż nierozstrzygnięta chociaż powszechnie zakłada się, że jest ona prawdziwa.

Ponieważ głównym celem pracy nie była dokładna analiza metod heurystycznych, więc algorytm ewolucyjny został zbadany tylko dla jednego problemu. Również zakres przeprowadzonych eksperymentów był ograniczony a ich celem było jedynie wykazanie, że odpowiedni dobór parametrów algorytmu ma istotny wpływ na uzyskane rozwiązanie oraz na czas obliczeń. Inne heurystyki, takie jak tabu-search i symulowane wyżarzanie, również były badane jednak dawały one wyniki znacznie gorsze niż algorytmu ewolucyjny. Z tego powodu nie zostały zamieszczone w pracy.

Wyniki uzyskane w pracy można uważać za początek szerszych badań nad problemami szeregowania prac z nieprecyzyjnie określonymi parametrami. W pracy zajęto się wyłącznie problemami szeregowania na pojedynczej maszynie, naturalne jest więc rozszerzenie zakresu badań na większą liczbę maszyn. Czasy trwania prac i pożądane terminy zakończenia nie są jedynymi parametrami, które można zadawać w sposób nieprecyzyjny. Można na przykład badać problemy, w których terminy dostępności prac są nieprecyzyjne. Również interesujące są problemy, w których relacja poprzedzania nie jest zadana w sposób precyzyjny.

W konstrukcji funkcji kosztu harmonogramu zastosowano indeksy Dubois i Prade oraz wartość średnią liczby rozmytej. Oczywiście nie są to jedyne pojęcia, które można do tego celu wykorzystać. W literaturze poświęconej zbiorom rozmytym można spotkać bardzo wiele różnych indeksów oraz funkcji rangujących, które można wykorzystać do konstrukcji funkcji kosztu harmonogramu. Należy jednak podkreślić, że indeksy Dubois i Prade oraz wartość średnia liczby rozmytej są jednymi z najprostszych pojęć z punktu widzenia własności obliczeniowych. Z tego względu wyniki dotyczące NP-trudności pewnych problemów są szczególnie cenne, ponieważ mogą one implikować NP-trudność problemów z innymi, bardziej złożonymi funkcjami kosztu.

Również wyniki uzyskane w rozdziale trzecim są tylko początkiem szerszych badań. Istnieje jeszcze wiele innych problemów na pojedynczej maszynie (oraz ogromna liczba problemów, w których liczba maszyn jest większa niż jeden), dla których można szukać algorytmów służących do oceny optymalności zadanego harmonogramu. Tutaj zaprezentowana została ogólna metoda konstruowania tego typu algorytmów a konkretne wyniki zostały przedstawione dla dwóch problemów.

Bardzo wiele publikacji poświęcono w literaturze konstrukcji różnego rodzaju metod heurystycznych, służących do efektywnego rozwiązywania problemów NP-trudnych. Naturalny jest więc dalszy kierunek badań, polegający na adaptacji

tych heurystyk do problemów szeregowania z nieprecyzyjnie określonymi parametrami, które są NP-trudne. W pracy został zaproponowany wyłącznie algorytm ewolucyjny, który jest tylko jedną z wielu znanych metod. Interesująca byłaby głębsza analiza porównawcza algorytmu ewolucyjnego z takimi metodami jak tabu-search i symulowane wyżarzanie.

Literatura

- [1] Błażewicz J., Cellary W., Słowiński R., Węglarz J., Badania operacyjne dla informatyków, WNT, Warszawa 1983.
- [2] Błażewicz J., Złożoność obliczeniowa problemów kombinatorycznych, WNT, Warszawa 1988.
- [3] Brucker P., Scheduling Algorithms, Second, Revised and Enlarged Edition, Springer 1997.
- [4] Chanas S., Wybrane problemy badań operacyjnych z rozmytymi parametrami, Wydawnictwo Politechniki Wrocławskiej, Monografie 1988.
- [5] Chanas S., Nowakowski M., Single value simulation of fuzzy variable, *Fuzzy Sets and Systems*, 25(1988), 43-57.
- [6] Cormen T., Leiserson C., Rivest R., Wprowadzenie do algorytmów, Wydawnictwa Naukowo-Techniczne, Warszawa 1994.
- [7] Czogała E., Pedrycz W., Elementy i metody teorii zbiorów rozmytych, PWN, Warszawa 1985.
- [8] Davis L., Applying Adaptive Algorithms to Epistatic Domains, *Proceedings of the International Joint Conference on Artificial Intelligence*(1985), 162-164.
- [9] Dubois D., Prade H., The use of fuzzy numbers in decision analysis, *Fuzzy Information and Decision Processes* (1982).
- [10] Dubois D., Prade H., Fuzzy Sets and Systems: Theory and Applications, Academic Press, Inc., New York 1980.
- [11] Dubois D., Prade H., The mean value of fuzzy number, *Fuzzy Sets and Systems* 24, (1987) 279-300.

- [12] Dubois D., Prade H., Ranking fuzzy numbers in the setting of possibility theory, *Information Science*, 30(1983), 183-224.
- [13] Dubois D., Prade H., Operations on fuzzy numbers, *Int., J. Systems Sci.*, 30(1978), 613-626.
- [14] Graham R. E., Lawler E. L., Lenstra J. K., Rinnoy Kan, Optimization and approximation in deterministic sequencing and scheduling, a survey, *Ann. Discrete Math.*, 4(1979), 287-326.
- [15] Han S., Ishii H., Fuji S., One machine scheduling problem with fuzzy due dates, *European Journal of Operational Research*, 79(1994), 1-12.
- [16] Ishii H., Tada M., Single machine scheduling problem with fuzzy precedence relation, *European Journal of Operational Research* 87 (1995), 284-288.
- [17] Ishibuchi H., Murata T., Kyu Hung Lee, Formulation of fuzzy flowshop scheduling problems with fuzzy processing times, Proceedings of the Fifth IEEE International Conference on Fuzzy Systems. FUZZ-IEEE'96, vol. 1, 199-205.
- [18] Ishibuchi H., Yamamoto N., Misaki S., Tanaka H., Local search algorithms for flow shop scheduling with fuzzy due dates, *International Journal of Production Economics*, 33(1994), 53-66.
- [19] Janiak A., Wybrane problemy i algorytmy szeregowania zadań i rozdziału zasobów, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1999.
- [20] Kacprzyk J., Zbiory rozmyte w analizie systemowej, PWN, Warszawa 1986.
- [21] Karmarkar N., A new polynomial time algorithm for linear programming, *Combinatorica*, 4(1984), 373-395.
- [22] Khachian L. G., A polynomial algorithm for linear programming, *Doklady Akad. Nauk USSR*, 244(1979) 1093-1096.
- [23] Lawler E. L., Optimal sequencing of a single machine subject to precedence constraints, *Management Science*, 19(1973), 544-546.
- [24] Lin C.K., Haley K.B., Sparks C., A comparative study of both standard and adaptive versions of threshold accepting and simulated annealing algorithms in three scheduling problems, *European Journal of Operational Research* 83(1995), 330-346.

- [25] McCahon C. S., Lee E. S., Fuzzy job sequencing for a flow shop , *European Journal of Operational Research*, 62(1992), 294-301.
- [26] Michalewicz Z., Algorytmy genetyczne + struktury danych = programy ewolucyjne, WNT, Warszawa 1996
- [27] Nguyen T. Hung, A Note on the Extension Principle for Fuzzy Sets, *Journal of Mathematical Analysis and Applications* 64(1978),369-380.
- [28] Ozelkan E., Duckstein L., Optimal fuzzy counterparts of scheduling rules, *European Journal of Operational Research* , 113(1997), 593-609.
- [29] Peng Tian, Jian Ma, Dong-Mo Zhang, Application of the simulated annealing algorithm to the combinatorial optimisation problem with permutation property: An investigation of generation mechanizm *European Journal of Operational Research* 118,(1999), 81-94
- [30] Reeves F.,A genetic algorithm for flow shop sequencing problem, Department of Statistic and Operational Research, Coventry University 1994
- [31] Sakawa M., Yano H., Interactive decision making for multiobjective linear fractional programming problems with fuzzy parameters based on solutions concepts incorporating fuzzy goals, *Japanese Journal of Fuzzy Theory and Systems*, 3(1991), 45-62.
- [32] Słowiński R., Hapke M. (Edts.), Scheduling Under Fuziness, Physica-Verlag 2000.
- [33] Tanaka K., Vlach M., Single machine scheduling problem with fuzzy due dates, *Seventh IFSA World Congress*, Praga 1997, 195-199.
- [34] Tsujimura Y., Seung H. P., In S. C., Gen M., An effective method for solving flow shop scheduling problems with fuzzy processing times, *Coputers And Industrial Engineering*, 1993(25), 239-242.
- [35] Yager R. R., A procedure for ordering fuzzy subset of the unit interval, *Information Science*, 24(1981), 143-161.
- [36] Zadeh L.A., Fuzzy Sets, *Information and Control*, 1965(8), 338-353.

- [37] Zadeh L.A., The Concept of Linguistic Variable and its Application to Approximate Reasoning, American Elsevier Publishing Company, New York 1973.