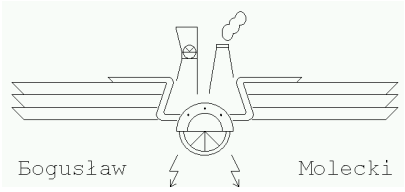


Bogusław Molecki

Algorytmy genetyczne a logika rozmyta

Politechnika Wrocławska 1998

1. Wprowadzenie

Popularne w ostatnich latach algorytmy genetyczne, pozwalają na symulację w komputerowym środowisku procesów ewolucyjnych na organizmach reprezentujących różne próby rozwiązania różnych, często silnie skomplikowanych problemów optymalizacji.

Logika rozmyta natomiast bardzo dobrze nadaje się do wykorzystania przy problemach, gdzie posiadana informacja, charakteryzująca środowisko, jest niepewna i nieprecyzyjna. Jak wyjaśnił twórca logiki rozmytej, L. Zadeh, systemy wykorzystujące ją mają dwa zastosowania:

- wąskie, w którym logika rozmyta jest używana do przybliżonego rozumowania,
- szerokie, w którym jest ona wykorzystywana wraz z teorią zbiorów rozmytych, czyli klasą obiektów, w których przejście między należeniem a nienależeniem jest stopniowe, a nie gwałtowne.

Połączenie tych dwóch technik może być więc bardzo obiecujące. W związku z tym, przeprowadzona została krótka analiza problemu, na podstawie ogólnie dostępnej literatury, zarówno z wykorzystaniem danych ze zbiorów dr Haliny Kwaśnickiej, jak i zbiorów bibliotecznych Politechniki Wrocławskiej oraz artykułów dostępnych w Internecie. Pełen spis podany jest na końcu opracowania.

Podstawowe dziedziny, w których połączenie algorytmów genetycznych i logiki rozmytej może być wykorzystane, zostały wymienione w opracowaniu [Cor ??]. Przytoczone opracowanie stanowi krótki wstęp - opis do bibliografii dotyczącej połączenia logiki rozmytej i algorytmów genetycznych. Bibliografia ta [Cor 96], uwzględniająca podział wymieniony w [Cor ??] jest bardzo cennym opracowaniem, umożliwiającym dalsze poszukiwania w dokładniej już określonej dziedzinie zastosowań. Dodatkową zaletą jest w tym przypadku możliwość łatwego uzyskania jej z Internetu, w postaci plików .html, lub .ps.

W niniejszym opracowaniu natomiast, za cel przyjęto przedstawienie podstawowych zasad połączenia obu dziedzin, wraz z przeglądem występujących możliwości ich zastosowań. Algorytmy genetyczne wraz z logiką rozmytą (której skrócone zasady przedstawione są w rozdziale 2.) znajdują zastosowanie w:

- rozmytych algorytmach genetycznych (opis w rozdziale 3.),
- optymalizacji sterowników rozmytych (opis w rozdziale 4.),
- zadawaniu pytań do baz danych (opis w rozdziale 5.),
- modelowaniu reakcji chemicznych (opis w rozdziale 6.),
- problemach podziału i klasyfikacji (opis w rozdziale 7.),
- podejmowanie decyzji ekonomiczno-finansowych (opis w rozdziale 8.),

oraz innych, nie opisanych tu bezpośrednio dziedzinach zastosowań, takich jak:

- sieci neuronowe,
- systemy ekspertowe.

2. Od logiki tradycyjnej do logiki rozmytej

Na co dzień jesteśmy przyzwyczajeni do tradycyjnej logiki dwuwartościowej, z wartościami równymi prawdzie i fałszowi. Jednym z możliwych zapisów wartości, który będzie używany w tym opracowaniu, polega na oznaczeniu prawdy przez wartość 1, a fałszu przez 0. Dla przypomnienia (i możliwości porównania z następnymi przypadkami) podam tablice wartościujące działania trzech operatorów logicznych (negacji, sumy i koniunkcji):

$\neg$	
0	1
1	1

$\vee$	0	1
0	0	1
1	1	1

$\wedge$	0	1
0	0	0
1	0	1

W pewnych przypadkach jednak sytuacja nie daje opisać się tak restrykcyjnymi pojęciami jak prawda i fałsz, „białe" i „czarne". Postaci jednoznacznie dobre i jednoznacznie złe występują jedynie w bajkach dla dzieci i czechosłowackich westernach („Lemoniadowy Joe"). W większości rzeczywistych przypadków wartość leży gdzieś pośrodku, mówiąc popularnie jest „szara".

Dlatego właśnie powstały logiki wielowartościowe, których teorię opracował polski matematyk, Jan Łukasiewicz (1878÷1956). Mianem tym są określane logiki, które dopuszczają trzy i więcej wartości. W najprostszym przypadku dopuszczona jest dodatkowa w stosunku do logiki tradycyjnej, trzecia wartość, którą liczbowo określa się jako 0,5 (tzw. półprawda). Tablice wartościujące działania logiczne wykorzystują znane reguły „odziedziczone" niejako z logiki tradycyjnej:

$\neg A = 1 - A$ (2.1.);

$A \wedge 0 = 0$ (2.2.); $A \wedge 1 = A$ (2.3.);

$A \vee 0 = A$ (2.4.); $A \vee 1 = 1$ (2.5.).

Niestety, reguły te nie wystarczają do wypełnienia całej tablicy wartościowania dla spójników $\wedge$ i $\vee$. Jak widać, pozostało wolne miejsce wewnątrz tablic, wypełnione są jedynie krawędzie (efekt ten jest jeszcze bardziej widoczny przy logikach więcej-wartościowych).

$\neg$	
0	1
0,5	0,5
1	1

$\vee$	0	0,5	1
0	0	0,5	1
0,5	0,5	**?**	1
1	1	1	1

$\wedge$	0	0,5	1
0	0	0	0
0,5	0	**?**	0,5
1	0	0,5	1

Wartości w centrum tablic nie można wyznaczyć jednoznacznie, wykorzystując zależności z logiki tradycyjnej. Oto przykład:

$B = A$; $A \wedge B = A \wedge A = A$ (2.6.);

$B = \neg A$; $A \wedge B = A \wedge \neg A = 0$ (2.7.);

Powyższe związki (2.6. oraz 2.7.) są prawdziwe dla logiki tradycyjnej. Niestety, uogólnienie ich na logikę wielowartościową musi zakończyć się porażką. Dla przykładu niech A=0,5. Tzw. „zdrowy rozsądek" podpowiada nam, że powyższe zależności powinny być spełnione w każdej logice. W tym przypadku jednak, wartościując równania (2.6. i 2.7.) otrzymamy następujące wyrażenia:

(z 2.6.): $A \wedge B = 0{,}5 \wedge 0{,}5 = 0{,}5$ (2.8.);

(z 2.7.): $A \wedge B = 0{,}5 \wedge 0{,}5 = 0$ (2.9.);

Jak widać, ten sam operator ($\wedge$) dla takich samych wartości dawać powinien dwa różne wyniki (2.8. i 2.9.). Analogiczny przykład można podać dla operatora $\vee$. Rozwiązanie powstało przez stworzenie dodatkowych

operatorów - tzw. „strong-operators” [Kin 98]. Przyjęto następujące ich definicje, rozwiewające wyżej opisane wątpliwości:

$A \wedge B = \min(A,B)$ (2.10.);
$A \otimes B = \max(0,A+B-1)$ (2.11.);
$A \vee B = \max(A,B)$ (2.12.);
$A \oplus B = \min(1,A+B)$ (2.13.);

Na tej podstawie określa się tabele wartościowania operatorów logicznych w logikach wielowartościowych - poniżej podane są, pełne już, tablice wartościowań dla logiki trójwartościowej:

$\neg$	
0	1
0,5	0,5
1	1

$\vee$	0	0,5	1
0	0	0,5	1
0,5	0,5	0,5	1
1	1	1	1

$\wedge$	0	0,5	1
0	0	0	0
0,5	0	0,5	0,5
1	0	0,5	1

Kolejnym krokiem w rozwoju logik niebinarnych było stworzenie przez Lotfiego Zadeha (1921÷) [Woe 95] logiki rozmytej. Można wyróżnić dwa „poziomy” wykorzystywania logiki rozmytej - pierwszy polega na użyciu logiki rozmytej jako naturalnego rozszerzenia logiki wielowartościowej (przejście z wyznaczonych n wartości dyskretnych w logikach n-wartościowych na nieskończoną liczbę wartości z przedziału [0;1] w logice rozmytej).

Często ta część teorii logiki rozmytej wykorzystywana jest nieświadomie, bez wgłębienia się w dalsze jej zależności. Tymczasem, logika rozmyta nawet w tym, wąskim rozumieniu niesie dodatkowe definicje, które porządkują m.in. operatory logiczne.

W logikach wielowartościowych, jak wspomniałem wyżej, definiowane były prócz zwykłych operatorów ($\wedge$ oraz $\vee$) operatory $\otimes$ oraz $\oplus$ (patrz wzory 2.10. ÷ 2.13), które definiowały skrajne wartości możliwe przy próbie rozszerzenia zwykłych operatorów $\wedge$ i $\vee$ na logiki wielowartościowe. Poniżej podane są tablice wartościowań dla logiki trójwartościowej właśnie tych operatorów:

$\oplus$	0	0,5	1
0	0	0,5	1
0,5	0,5	1	1
1	1	1	1

$\otimes$	0	0,5	1
0	0	0	0
0,5	0	0	0,5
1	0	0,5	1

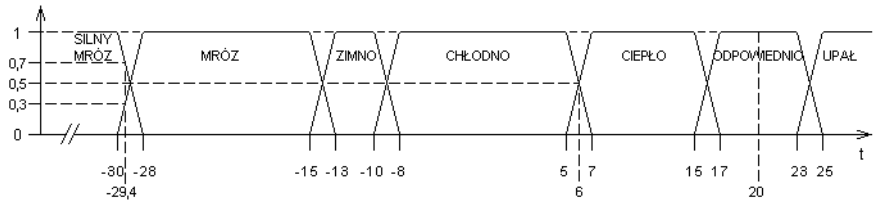
W logice rozmytej wszelkie operatory, których wyniki działania mieszczą się pomiędzy operatorami zwykłymi a dodatkowymi („strong-”) ujęte są w postaci norm. Operatory iloczynu w logice rozmytej określane są przez tzw. T-normy. Operatory sumy określane są jako T-conormy, dualne do T-norm. W polskiej literaturze znaleźć można określenie „S-norma” w stosunku do T-conorm, które jednak nie wydaje się najwłaściwsze.

W zależności od podejścia, przeprowadzane w logice rozmytej obliczenia, mogą wykorzystywać różne operatory sumy i iloczynu. Najbardziej popularne 11 definicji operatorów przytoczonych zostało w [Rut 97]. Poniżej przytoczone zostały pierwsze cztery pozycje zestawienia (operatory nieparametryzowane).

L.p.	T-norma (operatory $\wedge$)	T-conorma (operatory $\vee$)
1	min(A,B)	max(A,B)
2	A*B	A+B-A*B
3	max(A+B-1,0)	max(A+B-1)
4	A jeżeli B=1 B jeżeli A=1 0 jeżeli A,B≠1	A jeżeli B=0 B jeżeli A=0 1 jeżeli A,B≠0

Szersze zastosowanie logiki rozmytej polega na wykorzystaniu funkcji rozmytych przynależności do zbiorów. Określenie rozmytej przynależności do zbiorów pozwala na opisanie wartości zmiennej, bez podawania liczby opisującej tę wartość, a jedynie zbiorów, do których należy, wraz ze stopniem przynależności.

Przykładem może być tu opis temperatury.

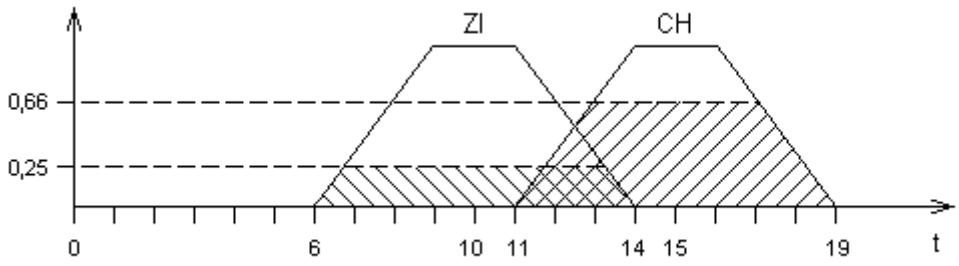


Rys. 2.1. Opis temperatury za pomocą zbiorów rozmytych

Jak widać na powyższym rysunku, temperatura opisana może być za pomocą terminów słownych, określających zbiory rozmyte (podział na rysunku wykonany został według osobistych odczuć autora, stąd być może nie odpowiada on powszechnemu). I tak, temperatura 20°C należy w 100 % do temperatur odpowiednich, zaś np. 6°C w połowie do chłodnych, w połowie do ciepłych. Podobnie, -29,4°C w 70% należy do silnie mroźnych, a w 30% do mroźnych. Rysunek w pełni wyjaśnia proces zamiany wartości liczbowej na wartość opisywaną funkcjami przynależności w logice rozmytej. Proces ten nazywany jest fuzzyfikacją (dokładniejszy opis w [Hei ??] i [Hry 96]).

Figury określające stopień przynależności danego obszaru wartości do określonych zbiorów rozmytych mogą być trapezami, trójkątami, lub innymi figurami geometrycznymi. Ich kształt i usytuowanie powinno jednak zapewniać fakt należenia każdej z wartości zmiennej w sumie w 100% (w innych przypadkach konieczna jest późniejsza normalizacja) do zbiorów rozmytych.

Procesem odwrotnym (przejście z wartości logiki rozmytej na wartości liczbowe) jest defuzzyfikacja. W tym przypadku obszary często definiowane są także jako trapezy odwrócone, lub singletony (pionowe prążki wskazujące środek ciężkości zbioru). Proces zamiany na wartości liczbowe polega na określeniu wartości jako średniej ważonej (aktywnymi polami figur) środków ciężkości tych figur. Aktywne pola figur to pola, obliczane od ich podstawy do wysokości odpowiadającej wartości stopnia przynależności do zbioru (stąd przydatne są w niektórych zastosowaniach trapezy odwrócone). Wyjaśnia to rysunek poniżej :



P_{ZI}=1,81 P_{CH}=4,00 T=(1,81*10+4,00*15)/(1,81+4,00)

Rys. 2.2. Obliczenie wartości na podstawie stopni przynależności do zbiorów rozmytych

Dokładniejszy opis zawierają [Hei ??], [Hry 96]. Ze względu na charakter wykorzystywanych w rzeczywistych zastosowaniach reguł rozmytych, suma stopni przynależności do poszczególnych zbiorów nie musi być równa jeden ! Stąd konieczne jest zapewnienie normalizacji przy średniej ważonej nawet w przypadku użycia singletonów.

Więcej informacji na temat teorii logiki rozmytej zawierają pozycje [Dri 96] i [Rut 97].

3. Użycie logiki rozmytej w algorytmach genetycznych

Oprócz szeroko znanych algorytmów genetycznych, w których do zapisu genotypu używa się kodowania binarnego (chromosomy to ciągi zer i jedynek), można spotkać również systemy oparte na innych alfabetach. Systemy takie nazywane są „real-coded genetic algorithms". Czym wyróżniają się RCGA ? Przede wszystkim budową genotypu. Składa się on bowiem nie z ciągu jedynie zer i jedynek, ale np. liczb rzeczywistych. Oczywiście, ze względu na inną reprezentację, inaczej muszą wyglądać również operatory genetyczne.

Za stosowaniem algorytmów genetycznych z kodowaniem binarnym (tradycyjnych) przemawiają następujące fakty:

- użycie alfabetu binarnego pozwala na maksymalną niezależność tworzonego oprogramowania (dla wszelkich prób zastosowań algorytm posiada jednakowe, lub prawie jednakowe operatory genetyczne);
- żaden inny alfabet nie jest tak elastyczny (przestrzeń zmienności fenu regulowana długością genu w chromosomie), a przy tym tak dostosowany do przetwarzania maszynowego (alfabet binarny jest podstawowym dla komputerów) - co pozwala na łatwe i szybkie stworzenie aplikacji oraz wpływa korzystnie na szybkość jej działania.

Jeśli jednak algorytmy z kodowaniem binarnym są tak dobre, z jakich względów wprowadzono RCGA ? Okazuje się, że za RCGA również przemawiają pewne korzyści:

- możliwość wykorzystania dowolnych dziedzin zmiennych, nawet nie określonych;
- małe zmiany w zmiennych powodują małe zmiany funkcji przystosowania (kłopotliwy przymus używania kodowania Graya w algorytmach z kodowaniem binarnym);
- łatwość używania i rozumienia - brak różnicy w zapisie między fenotypem a genotypem;
- brak konieczności rozkodowywania genu podczas obliczeń funkcji fitness, co wpływa korzystnie na szybkość obliczeń.

Zastosowania logiki rozmytej w operatorach RCGA opisane są w punktach 3.1.÷ 3.4. Więcej na temat samych algorytmów genetycznych wykorzystujących kodowanie rzeczywiste, można znaleźć m.in. w [Her ?2], natomiast tworzenia operatorów rozmytych w [Her 95]. W punkcie 3.5. natomiast opisano krótko schemat możliwości użycia sterownika rozmytego do kontroli algorytmów genetycznych (więcej informacji na temat sterowników rozmytych można znaleźć w rozdziale 4.).

3.1. Podstawowe operatory dla RCGA

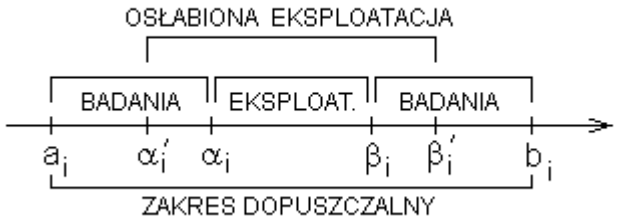
W przypadku RCGA najczęściej wyróżnia się jedynie operator krzyżowania, bez operatora mutacji. Jest to uzasadnione ze względu na zmienioną rolę operatora krzyżowania w przypadku RCGA, w porównianiu z modelem tradycyjnym.

Wartość (allel) i-tego genu w przypadku RCGA przedstawiona jest za pomocą liczby rzeczywistej, z dopuszczalnego przedziału [ai,bi]. Wszelkie operatory odwołujące się do genotypu, rozpatrywane są na poziomie pojedyńczych genów - krzyżowanie dwóch osobników polega na dokonaniu tej operacji osobno na wszystkich kolejnych genach genotypu. Jak łatwo się domyślić, operacja krzyżowania dwóch genów αi oraz βi powinna dać w rezultacie nową wartość genu, która musi mieścić się w przedziale [ai,bi].

Analizując opisaną wyżej sytuację, można wydzielić następujące cztery podprzypadki dotyczące wyników krzyżowania:

- nowa wartość genu mieści się w przedziale [ai,αi],
- nowa wartość genu mieści się w przedziale [βi,bi],
- nowa wartość genu mieści się w przedziale [αi,βi],
- nowa wartość genu mieści się w przedziale [α'i,β'i].

Sytuację tą opisuje dokładnie rysunek:



Rys. 3.1.1. Cztery możliwe przedziały efektów zastosowania operatorów krzyżowania w RCGA

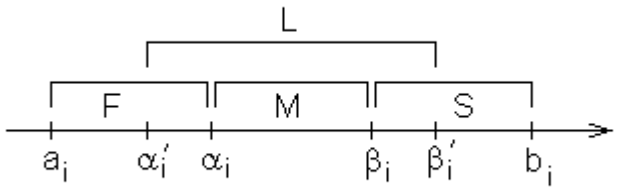
W 1. i 2. przypadku, jak widać, wynik nie leży pomiędzy wcześniejszymi wartościami genów - przypadki te można więc określić mianem badania nowych rozwiązań (ang. exploration). 3. przypadek to eksploatacja rozwiązań już posiadanych, 4. zaś to osłabiona eksploatacja rozwiązań (dopuszczone są pewne wartości z poza przedziału $[\alpha i,\beta i]$.

Należy zwrócić przy tym uwagę, że o ile przestrzeń rozwiązań zostanie znormalizowana (przedział $[ai,bi]=[0,1]$, co dodatkowo wpłynie na uproszczenie procesu konstrukcji i działań na genotypie), to za przedstawione wyżej działania odpowiadają funkcje logiki rozmytej:

- za badanie w obszarze $[ai,\alpha i]$ - t-norma (T),
- za badanie w obszarze $[\beta i,bi]$ - t-conorma (G),
- za eksploatację rozwiązań - funkcja uśredniająca (P),
- za osłabioną ekspolatację - uogólniony operator kompresji (Č).

Tu należy przypomnieć, że niektóre funkcje, spełniające odpowiednie warunki zaliczenia np. do t-norm, czy funkcji uśredniających, są parametryzowane. Parametr operatora logiki rozmytej pozwala na otrzymanie wyniku z zakresu maksymalnie od zwykłego operatora (np. $\wedge$) do tzw. silnego operatora (w tym wypadku $\otimes$). Poprzez podstawienie w czasie krzyżowania losowej wartości parametru można otrzymać zróżnicowanie wyników tej samej operacji (analogiczne do zróżnicowania operacji krzyżowania w algorytmach genetycznych z kodowaniem binarnym).

Przykładowe formuły wybrane z logiki rozmytej, wykorzystywane w RCGA, podane są poniżej (operator F odpowiada t-normie, S - t-conormie, M - funkcji uśredniającej, L - uogólnionemu operatorowi kompresji).



Rys. 3.1.2. Zakres wyników operacji krzyżowania generowanych przez poszczególne operatory RCGA

Rodzina	T-norma	T-conorma	F. uśredniająca	Uogóln. op. komp.
Logiczna	$T_1 = \min(x,y)$	$G_1 = \max(x,y)$	$P_1 = (1-\lambda)x+\lambda y$	$\check{C}_1 = T_1^{1-\lambda} G_1^{\lambda}$
Hamacher	$T_2 = \frac{xy}{x+y-xy}$	$G_2 = \frac{x+y-2xy}{1-xy}$	$P_2 = 1 / \left(\frac{y-y\lambda-xy+x\lambda}{xy}+1\right)$	$\check{C}_2 = P_2(T_2,G_2)$

Algebraiczna	$T_3 = xy$	$G_3 = x+y-xy$	$P_3 = x^{1-\lambda}y^{\lambda}$	$\check{C}_3 = P_3(T_3,G_3)$
Einstein	$T_4 = \frac{xy}{1+(1-x)(1-y)}$	$G_4 = \frac{x+y}{1+xy}$	$P_4 = 2 / (1+(\frac{2-x}{x})^{1-\lambda}(\frac{2-y}{y})^{\lambda}$	$\check{C}_4 = P_4(T_4,G_4)$

Operatory F1, S1, M1 i L1 dają przy tym największy nacisk na eksploatację starych rozwiązań (np. przedział L1 $[\alpha'i,\beta'i]=[\alpha i,\beta i]$). Skrajnym ich przeciwieństwem są F4, S4, M4 i L4 - tu wspomniany przedział $[\alpha'i,\beta'i] = [ai,bi]$. Operatory 2. i 3. wiersza plasują się pomiędzy 1. i 4.

Porównianie poszczególnych operatorów zostało zamieszczone w [Her 97]. Zostały tam również przytoczone propozycje strategii wykorzystujących te operatory (ogólną zasadą jest wybranie 50% populacji do operacji krzyżowania, nowopowstałe organizmy zajmują miejse rodziców, a w razie potrzeby także pozostałych 50% populacji):

- wykorzystanie operatorów F, S i M (2 osobniki z badań, 1 z eksploatacji),
- wykorzystanie wszystkich czterech operatorów: F, S, M, L (2 osobniki z badań, 2 z eksploatacji, do populacji trafiają 2 najlepsze),
- wykorzystanie operatorów F, S i dwukrotnie L (2 osobniki z badań, 2 z osłabionej eksploatacji, do populacji trafiają 2 najlepsze),
- wykorzystanie wszystkich czterech operatorów: F, S, M, L (do populacji trafiają wszystkie 4).

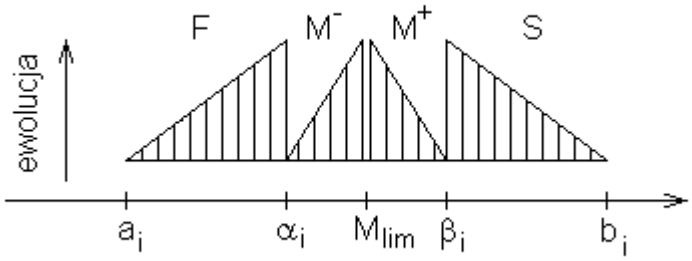
Dobór strategii uzależniony jest on twórcy algorytmu i przyjętego modelu ewolucji (regulacja ciśnienia selekcyjnego). Strategia może ponadto być zmieniana w czasie pracy algorytmu, przez co w początkowych pokoleniach może on kłaść największy nacisk na badanie (poszukiwanie różnych rozwiązań), a w drugiej na eksploatację (doskonalenie posiadanych rozwiązań).

3.2. Operatory dynamiczne oparte na logice rozmytej

Podczas obserwacji pracy algorytmów genetycznych, często zauważa się tendencję do prób wymuszania w początkowych etapach ewolucji tworzenia szerokiej gamy różnych rozwiązań (losowanie osobników początkowej populacji, niskie ciśnienie selekcyjne). W końcowej fazie pracy algorytmu zaś główny nacisk kładziony jest na doskonalenie znalezionych już rozwiązań (wysokie ciśnienie slekcyjne). Zagadnienie to zostało już wspomniane przy omawianiu strategii w poprzednim rozdziale.

Próbą utrzymania stałej tendencji do przechodzenia z badania do eksploatacji rozwiązań podczas pracy algorytmu genetycznego, są opisane w [Her ?1].

Rozwiązanie tam zaproponowane jest proste - poprzez użycie parametryzowanych operatorów krzyżowania F, S, M-, M+. Operator L w tym przypadku nie występuje, natomiast operatory M- i M+ odpowiadają w sumie wcześniej opisanemu operatorowi M (M- dąży do wartości średniej αi i βi (punkt M_{lim}) od strony ai, natomiast M+ od strony bi). Przestrzeń wyników działania tych operatorów w trakcie procesu ewolucji ilustruje rysunek poniżej :



Rys. 3.2.1. Zależność wyników dawanych przez operatory dynamiczne od etapu ewolucji

Zaproponowane w [Her ?1] strategie poszukiwań były następujące:

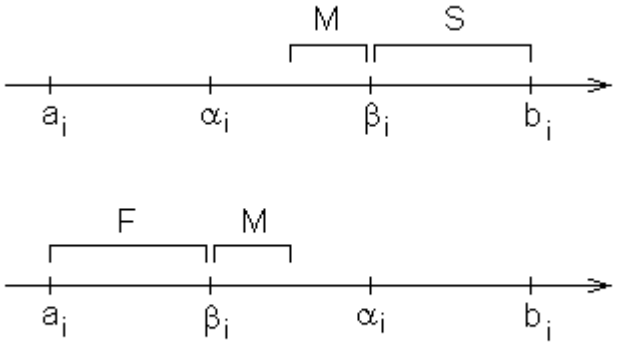
- dla każdej pary chromosomów z populacji generowane są wyniki działania wszystkich operatorów: F, S, M+ i M-. Z nich wybierane są 2 najbardziej obiecujące organizmy (o większym fitness) i zastępują rodziców,

- dla każdej pary z 50 % populacji generowane są wyniki działania wszystkich operatorów: F, S, M+ i M-. Dwa z nich zastępują w populacji rodziców, dwa-osobniki z pozostałych 50 % populacji.

3.3. Operatory heurystyczne

Dodatkowo, możliwe jest zastosowanie w algorytmach genetycznych elementów heurystyki. Możliwość ta polega na skierowaniu rezultatu procesu krzyżowania w okolicę lepszego genu, przystępującego do krzyżowania.

Przyjmijmy, że krzyżowane są dwa genotypy, przy czym lepszy jest organizm zawierający gen βi. Pozytywny efekt może mieć więc wymuszenie położenia wyniku krzyżowania w pobliżu genu lepszego ogranizmu (patrz rysunek):



Rys. 3.3.1. Heurystyka polega na wymuszeniu położenia wyników krzyżowania w pobliżu lepszych organizmów (tu: βi)

Przełożenie idei na wzory przedstawia się następująco (przypominam o założeniu, iż lepszy był organizm zawierający gen βi) :

$$h_1 i = \begin{cases} F(\alpha i,\beta i) \text{ jeżeli } \beta i < \alpha i, \\ S(\alpha i,\beta i) \text{ jeżeli } \beta i \geq \alpha i. \end{cases} \quad (3.3.1.)$$

$$h_2 i = M(\alpha i,\beta i) \text{ przy czym } |\alpha i - h_2 i| \geq |\beta i - h_2 i| \quad (3.3.2.)$$

gdzie $h_1 i$ i $h_2 i$ są nowymi genotypami, postałymi w wyniku krzyżowania.

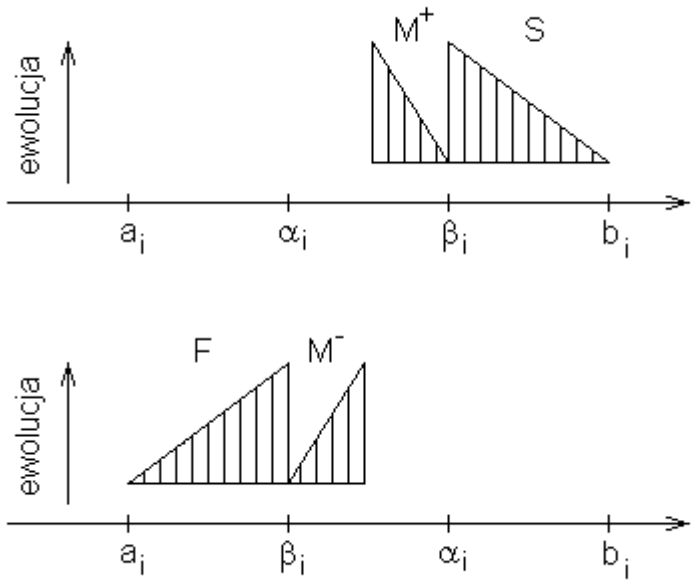
Należy jednak zaznaczyć, że:

- niekoniecznie wartość genu βi odpowiada za to, że był on lepszy - być może ten sam organizm, posiadając w tym miejscu genotypu gen αi byłby lepiej przystosowany,
- opracowanie tak skomplikowanego mechanizmu krzyżowania powoduje spowolnienie samego algorytmu genetycznego,
- heurystyczne operatory krzyżowania stają się ingerencją w proces ewolucji, który w praktyce sam powinien doprowadzić do znalezienia rozwiązań optymalnych lub quasi-optymalnych,
- sama idea polega na ograniczeniu przestrzeni poszukiwań - nie wydaje się więc, jakoby powinna być stosowana w początkowym okresie ewolucji.

Więcej na temat operatorów wykorzystujących techniki heurystyczne można znaleźć w [Her ?1].

3.4. Operatory heurystyczno-dynamiczne

Możliwe jest także połączenie technik heurystycznych z dynamicznym sterowaniem procesami ewolucji. Zakładając, podobnie jak w poprzednim rozdziale, że lepszy był osobnik zawierający gen βi, operatory krzyżowania działają jak na rysunku:



Rys. 3.4.1. Zależność wyników dawanych przez operatory heurystyczno-dynamiczne od etapu ewolucji

Zależności, według których wyznacza się wzory operatorów, wynikają bezpośrednio z rozdziałów 3.2. i 3.3. Więcej na ten temat można znaleźć w [Her ?1].

Niestety, podobnie jak w punkcie 3.3., operatory te nasuwają wiele obaw o naruszenie procesu ewolucji (jest to przecież sztuczna ingerencja w proces wzorowany na naturze), a połączenie obu technik powoduje dalsze spowolnienie pracy algorytmu (co nie znaczy, że wyniki nie mogą być otrzymane szybciej)...

3.5. Dynamiczna kontrola AG przy pomocy sterowników rozmytych

Przy wykorzystaniu sterowników rozmytych, opisanych w rozdziale 4. niniejszego opracowania, można kontrolować proces ewolucji w algorytmie genetycznym. Problem kontroli algorytmów genetycznych ma kluczowe znaczenie w zagadnieniach utrzymywania odpowiedniego zróżnicowania populacji, co wiąże się z problemem przedwczesnej jej zbieżności (zakończenie procesu ewolucji przed znalezieniem optimum globalnego).

Sterownik rozmyty posiada określoną bazę reguł, służącą do kontroli przebiegu procesu ewolucji, poprzez wpływ na wiążące parametry aplikacji algorytmu genetycznego, takie jak np. wielkość populacji. Przykładowe reguły dotyczące tego zagadnienia mogą być następujące:

IF (średni fitness/najlepszy fitness) jest DUŻY THEN rozmiar populacji powinien się zwiększyć,

IF (najgorszy fitness/średni fitness) jest MAŁY THEN rozmiar populacji powinien się zmniejszyć.

Należy tu jednak zauważyć, że poszukiwanie odpowiedniej bazy reguł (a szczególnie dostrajanie sterownika) jest w tym przypadku zadaniem niesłychanie skomplikowanym, można powiedzieć, że na miarę... algorytmu genetycznego. Niestety, w dostępnej literaturze nie ma praktycznie żadnych danych na temat próby zastosowania algorytmu genetycznego do optymalizacji takiego sterownika, poza pojedynczymi wzmiankami o samej idei.

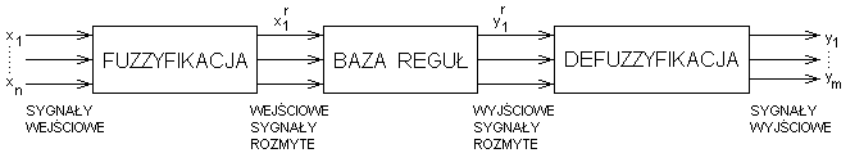
Więcej informacji na temat sterownika tworzonego w konwencjonalny sposób można znaleźć w [Lee 93], skąd zresztą pochodzą podane wyżej przykłady reguł.

4. Wykorzystanie algorytmów genetycznych w sterownikach rozmytych

Jednym z podstawowych zastosowań logiki - w tym rachunku zdań - w przemyśle są sterowniki przemysłowe. Silnie rozwinięta teoria sterowania w chwili obecnej dysponuje wieloma środkami, pozwalającymi na obsługę wszelkich praktycznie zastosowań.

Logika rozmyta znalazła zastosowanie w połączeniu z teorią sterowania ze względu na prosty sposób opisu reguł sterowania, możliwość zastosowania w przypadku systemów nieliniowych, a nawet systemów o nieznanym modelu matematycznym. Więcej na ten temat można dowiedzieć się z [Dri 96].

Klasyczny sterownik rozmyty można opisać za pomocą trzech bloków:



Rys. 4.1. Schematyczna budowa klasycznego sterownika rozmytego

Zasada działania sterownika rozmytego jest następująca: sygnały wejściowe (odczyty temperatury z termometrów, stężenia CH_4 z czujników itp.) docierają w formie sygnałów analogowych (najczęściej napięciowych, lub prądowych) albo cyfrowych do bloku fuzzyfikacji. Tam następuje proces fuzzyfikacji, opisany w rozdziale 2. tego opracowania. Rozmyte sygnały wejściowe, powstałe w bloku fuzzyfikacji, służą do wysterowania głównej części sterownika, związanej z bazą reguł. Blok ten generuje rozmyte sygnały wejściowe, które po przejściu procesu defuzzyfikacji (opis także w rozdziale 2.) są używane do wysterowywania rzeczywistych elementów wykonawczych (np. grzejników, wentylatorów).

Istnieją dwa typy baz reguł używanych w sterownikach rozmytych.

Pierwszy z nich, w pełni rozmyty, wykorzystuje się w sterownikach o strukturze przedstawionej na rysunku 4.1.. Poszczególne reguły w bazie mają zazwyczaj następującą postać:

IF wer1=Lwe1 AND wer2=Lwe2 AND ... AND wern=Lwen
THEN wyr1=Lwy1 AND wyr2=Lwy2 AND ... AND wyrm=Lwym (4.1.)

przy czym wer1÷wern są rozmytymi zmiennymi wejściowymi, Lwe1÷Lwen są terminami lingwistycznymi określającymi poszczególne zbiory rozmyte dotyczące zmiennych wejściowych, wyr1÷wyrm są rozmytymi zmiennymi wyjściowymi, Lwy1÷Lwym są terminami dotyczącymi zmiennych wyjściowych. Przykładem takiej reguły może być:

IF temperatura=”NISKA” AND wilgotność=”NISKA”
THEN ogrzewanie=”SILNE” AND wentylacja=”BRAK”. (4.2.)

Jak wiadomo, z procesu fuzzyfikacji sygnały do bazy reguł docierają w postaci wartości stopni przynależności do zbiorów rozmytych, co odpowiada stopniom zgodności z poszczególnymi terminami lingwistycznymi. Na podstawie wartości poszczególnych stopni zgodności zmiennych wejściowych, przy użyciu operatorów logiki romytej, określa się wartość całej reguły. I tak dla reguły (4.2.) przy przyjęciu wartości sygnałów wejściowych:

temperatura=”NISKA” 0,73
wilgotność=”NISKA” 0,16 (4.3.)

otrzymany z reguły (4.2.) wynik:

ogrzewanie=”SILNE” 0,16
wentylacja=”BRAK” 0,16 (4.4.)

ponieważ we wzorze (4.2.) pomiędzy równaniami zmiennych wejściowych stał operator AND (w tym wypadku użyto do obliczeń funkcji a AND b = min (a,b)).

Wyniki z poszczególnych reguł przekazywane są jako dane dla procesu defuzzyfikacji, który generuje już ostateczne wartości zmiennych wyjściowych.

Drugim typem reguł używanych w sterownikach rozmytych są reguły, w których rozmyta jest tylko ich pierwsza, warunkowa część. Opis zmiennych wyjściowych nie zawiera w tym wypadku odwołań do logiki rozmytej. Przykładem może być reguła:

IF temperatura=”NISKA” AND wilgotność=”WYSOKA”

THEN ogrzewanie=140-t1+2*t2 AND wentylacja=0,18*w1+1,47*w2 (4.5.)

Dokładniejszy opis tego typu sterowników można znaleźć w [Dri 96] i [Rut 97].

W tym rozdziale, wszystkie opisywane sterowniki rozmyte tworzone są na podstawie założeń pierwszego rodzaju. Przykład sterownika rozmytego częściowo, występuje jedynie w rozdziale 6. Optymalizacja tego typu sterowników może przebiegać analogicznie do niżej opisanych, przy czym dla optymalizacji zmiennych wyjściowych (poszukiwanie współczynników) można użyć algorytmów genetycznych wykorzystujących rzeczywiste kodowanie genotypu (RCGA), opisanych w rozdziale 3. tego opracowania.

Proces tworzenia sterowników rozmytych przebiega następująco :

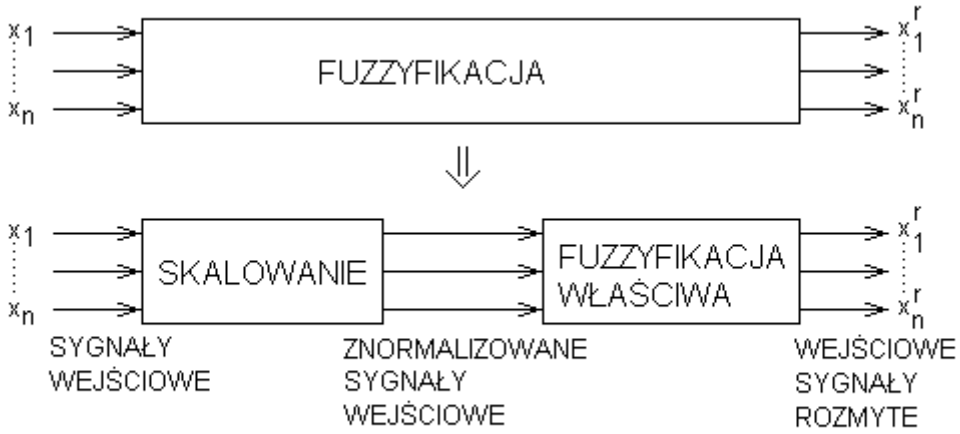
- wyznaczenie zmiennych wejścia, stanu (zapewniających sprzężenie zwrotne) i wyjścia,
- określenie przestrzeni zmiennych (ich dziedziny), związanych z nimi zbiorów rozmytych i ich zakresów przynależności do przestrzeni zmiennych,
- stworzenie bazy reguł rozmytych systemu,
- ocena systemu i ewentualne modyfikacje.

Jak widać, sam algorytm tworzenia sterowników rozmytych narzuca skojarzenia z procesem ewolucyjnym. Algorytmy genetyczne w przypadku sterowników rozmytych stosowane są do optymalizacji funkcji fuzzyfikacji, defuzzyfikacji, lub bazy reguł. Oczywiście, optymalizowane mogą być również jednocześnie dwie, lub nawet wszystkie trzy grupy w/w funkcji. Wszystko zależy od zastosowanego podejścia. Podobnie, różnią się w konkretnych zastosowaniach wykorzystywane funkcje fitness, oceniające stworzony system. Najczęściej ocena polega na porównaniu z założonym modelem docelowym (czasem istniejącym sterownikiem), lub symulacji pracy stworzonego sterownika i sprawdzeniu czasu doprowadzenia do stanu żądanego. Dodatkowym kryterium optymalizacji jest minimalizacja liczby reguł (szybkość działania sterownika rozmytego jest silnie zależna od liczby realizowanych reguł).

W dalszej części rozdziału opisane są możliwości optymalizacji poszczególnych grup parametrów sterowników rozmytych, a także konkretne przykłady zastosowań opisane w dostępnej literaturze.

4.1. Optymalizacja procesu fuzzyfikacji

Proces fuzzyfikazji może być przedstawiony jako złożenie procesu skalowania i fuzzyfikacji właściwej:



Rys. 4.1.1. Przebieg procesu fuzzyfikacji

Wyróżnienie skalowania jako osobnego bloku w sterowniku rozmytym (zarówno w procesie fuzzyfikacji, jak i defuzzyfikacji, o czym dalej) pozwala na sprowadzenie właściwego bloku fuzzyfikującego do zestawu funkcji postaci

Fij' : [0,1] -> [0,1] (4.1.1.)

zamiast Fij : Dij -> [0,1] (4.1.2.)

gdzie Fij jest podstawową fukcją fuzzyfikującą j-ty zbiór rozmyty i-tej zmiennej wejściowej, Dij- dziedziną j-tego zbioru i-tej zmiennej wejściowej, a Fij' - funkcją zmodyfikowaną, wykorzystującą wcześniejsze skalowanie:

Sij : Dij -> [0,1] (4.1.3.)

Korzyścią z wyodrębnienia skalowania poza proces właściwej fuzzyfikacji jest możliwość użycia dużo prostszego algorytmu genetycznego przy tworzeniu sterownika rozmytego. Algorytm w przypadku wykorzystania funkcji (4.1.2.) powinien bowiem zapewniać :

- zakodowanie położenia punktów charakterystycznych zbiorów rozmytych w rzeczywistej przestrzeni zmiennej (mogłoby okazać się konieczne użycie RCGA, opisanych w rozdziale 3.),
- eliminację osobników nie odpowiadających dopuszczalnemu zakresowi (dziedzinie) określającym daną zmienną.

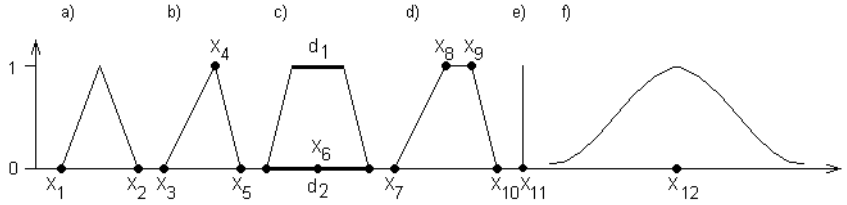
W przypadku wykorzystania współczynników skalujących zmienne (4.1.3.) przed właściwym procesem fuzzyfikacji (4.1.1.) algorytm genetyczny staje się prostszy do wykonania :

- współczynniki skalujące możemy określić wstępnie (znany zakres zmienności) danego sygnału wejściowego, lub poddać je procesowi ewolucji (zgodnie z zasadami kodowania liczb rzeczywistych),
- położenie punktów charakterystycznych zbiorów rozmytych określa się jedynie w przedziale [0,1], ze z góry założoną dokładnością. Pozwala to na kodowanie ich na zasadach liczb całkowitych (a nawet potęg 2, co pozwala na dalsze uproszczenie programu ewolucyjnego),
- nakładanie ograniczeń związanych z dziedziną zmiennej nie jest w tym przypadku konieczne - o prawidłowe działanie sterownika zadba sam algorytm genetyczny.

Położenie zbiorów rozmytych względem dziedziny zmiennej kodowane jest w genotypie poprzez podanie współrzędnych punktów charakterystycznych dla danego typu figur wybranych do opisu zbiorów rozmytych (rys. 4.1.2.). I tak:

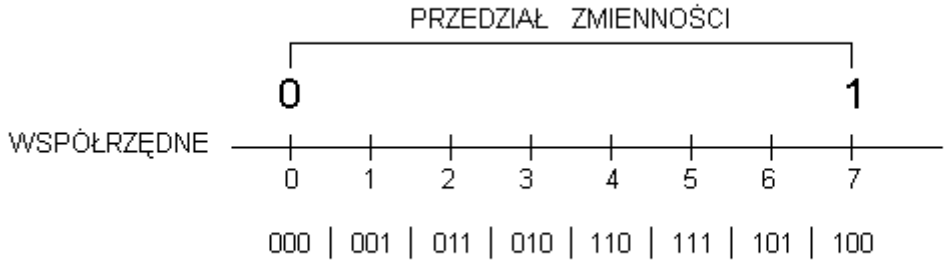
- dla trójkątów równoramiennych są to np. współrzędne wierzchołków przy podstawie (rys. 4.1.2.a),
- dla trójkątów dowolnych są to współrzędne wszystkich 3 wierzchołków (rys. 4.1.2.b),
- dla trapezów równoramienych są to np. współrzędna środka trapezu, długość podstawy górnej i dolnej (rys. 4.1.2.c),
- dla trapezów dowolnych są to współrzędne wszystkich czterech wierzchołków (rys. 4.1.2.d),

- dla singletonów jest to ich współrzędna położenia (rys. 4.1.2.e),
- dla krzywej Gaussa (rys. 4.1.2.f) są to parametry σ (określająca kształt) i μ (położenie). Dokładniejsze informacje na ten temat można znaleźć np. w [Bro 95].



Rys. 4.1.2. Punkty charakterystyczne w opisie zbiorów rozmytych: a)-trójkąt równoramienny, b)-trójkąt dowolny, c)- trapez równoramienny, d)- trapez dowolny, e)- sigleton, f)- krzywa Gaussa

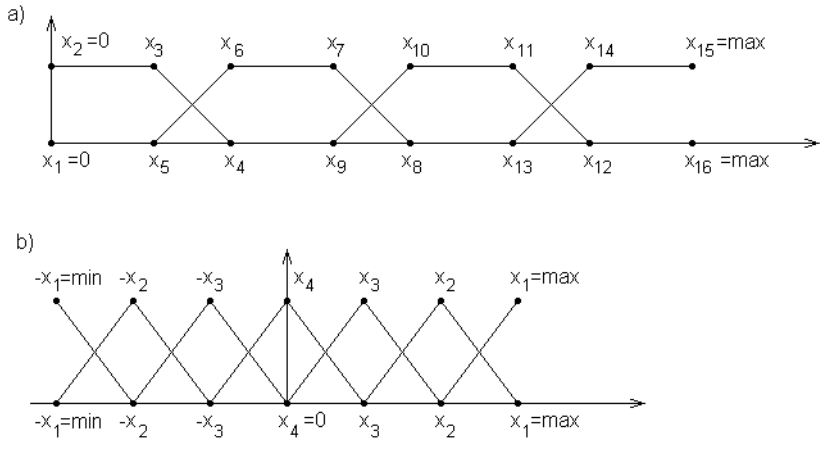
Jak widać na przykładzie krzywej Gaussa, używanie niestandardowych krzywych wiąże się z kłopotliwym wyrażaniem ich parametrów, m.in. odpowiadających za kształt,lecz bezpośrednio nie związanych z dziedziną (σ). W pozostałych wypadkach natomiast wszystkie parametry mogą być jednakowo odwzorowywane, na zasadach skończonego przedziału liczb całkowitych (także z wykorzystaniem kodu Graya):



Rys. 4.1.3. Odwzorowanie parametrów dla skończonego przedziału liczb całkowitych

Dodatkowo, należy zauważyć, że liczba współrzędnych podlegających optymalizacji może zostać obniżona ze względu na :

- istnienie zbiorów krańcowych (np. dla trapezów - rys. 4.1.4.a),
- chęć zachowania symetrii podziału (np. dla trójkątów - rys. 4.1.4.b),
- istnienie wartości neutralnej (np. zero dla temperatury - rys. 4.1.4.b).



Rys. 4.1.4. Możliwości obniżenia liczby zmiennych opisujących zbiory rozmyte, podlegających optymalizacji

4.2. Optymalizacja bazy reguł

Baza stworzona z reguł odpowiadających schematowi (4.1.) może być przedstawiona w postaci n-wymiarowej tabeli (n - liczba zmiennych wejściowych), w której każdym wymiarze wyróżniane są związane z daną zmienną zbiory rozmyte. Zapis ten jest analogiczny do zapisu sterowników opartych na układach logicznych (tablice Karnaugh'a).

Przykładem (za [Bie 96]) może być baza reguł dla dwóch zmiennych wejściowych:

temperatury - (zbiory rozmyte: Zimno, CHłodno, ODPowiednio, Ciepło, Gorąco) i liczby ludzi w muzeum - (zbiory rozmyte: Bardzo Mało, Mało, ŚRednio, Dużo, Bardzo Dużo). Zmienną wyjściową jest tu nawiew (zbiory rozmyte: GOrący, CIepły, Neutralny, CHłodny, ZImny). Tabela zależności wygląda następująco :

	Liczba zwiedzających :				
Temperatura :	Bardzo Mało	Mało	ŚRednio	Dużo	Bardzo Dużo
Zimno	GOrący	CIepły	Neutralny	Neutralny	CHłodny
CHłodno	CIepły	Neutralny	Neutralny	CHłodny	CHłodny
ODPowiednio	Neutralny	Neutralny	CHłodny	CHłodny	Zimny
Ciepło	Neutralny	CHłodny	CHłodny	Zimny	Zimny
Gorąco	CHłodny	CHłodny	Zimny	Zimny	Zimny

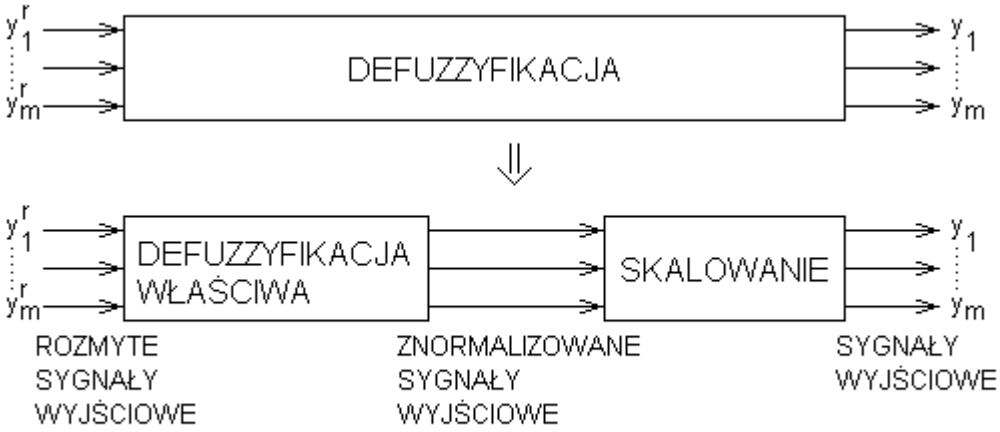
Ponadto, możliwe jest, by dla danych wartości zmiennych wejściowych reguła nie była określona (nie będzie się ona „odpalała" podczas pracy sterownika). Wygodnie jest zaznaczyć takie reguły w tabeli zależności za pomocą wyróżnionej wartości zmiennej wyjściowej.

Podczas optymalizacji za pomocą algorytmów genetycznych w chromosomie powinny zostać zakodowane wszystkie reguły. Próba kodowania jedynie reguł używanych (bez „nie odpalanych") wywoła bowiem możliwość zmiennej długości chromosomu, a w efekcie komplikację algorytmu i zwiększenie złożoności problemu.

W związku z tym, najczęściej przyjmuje się dla zmiennych wyróżnienie 2^x -1, $x \in N^+$ zbiorów rozmytych (w przypadku, kiedy dopuszcza się możliwość reguł nieokreślonych), lub 2^x, $x \in N^+$ w przeciwnym przypadku. Pozostała w pierwszym przypadku, wolna wartość, używana jest właśnie do zaznaczenia, że reguła w sterowniku będzie „nie odpalana". Istnieje także możliwość zakodowania możliwości używania danej reguły w osobnym bicie. Reguły w chromosomie kodowane są poprzez wartości zmiennej wyjściowej dla kolejnych pozycji tabeli zależności.

4.3. Optymalizacja procesu defuzzyfikacji

Proces defuzzyfikacji, podobnie jak fuzzyfikacji może być przedstawiony za pomocą złożenia procesu defuzzyfikacji właściwej i procesu skalowania:



Rys. 4.3.1. Przebieg procesu defuzzyfikacji

Co pozwala na opisanie bloku defuzzyfikacji funkcjami postaci:

Gij' : [0,1] -> [0,1] (4.3.1.)

zamiast Gij : [0,1] -> Dij (4.3.2.)

gdzie Gij jest podstawową fukcją defuzzyfikującą j-ty zbiór rozmyty i-tej zmiennej wyjściowej, Dij- dziedziną j-tego zbioru i-tej zmiennej wyjściowej, a Gij' - funkcją zmodyfikowaną, której wartość wykorzystywana jest następnie w procesie skalowania:

Sij : [0,1] -> Dij (4.3.3.)

Ze względu na analogię reguł defuzzyfikacji do występujących w procesie fuzzyfikacji (4.1.1.÷ 4.1.3.), tworzenie podstaw oraz optymalizacja może zostać przeprowadzona analogicznie (patrz rozdział 4.1.).

4.4. Tworzenie prostych sterowników rozmytych

Proste sterowniki rozmyte tworzone są do sterowania systemami poprzez wysyłanie odpowiednich sygnałów wyjściowych, na podstawie otrzymanych sygnałów wejściowych. Krótko zasadę ich działania przedstawiłem na początku rozdziału 4..

Tworzenie prostych sterowników rozmytych opisane zostało w [Hei ??] i [Hry 96]. Można wyróżnić w tym przypadku trzy etapy - przygotowanie podstaw, tworzenie genotypu i jego ocena. Etapy drugi i trzeci wykonywane są wielkrotnie, przy użyciu algorytmów genetycznych.

Przygotowanie podstaw obejmuje określenie:

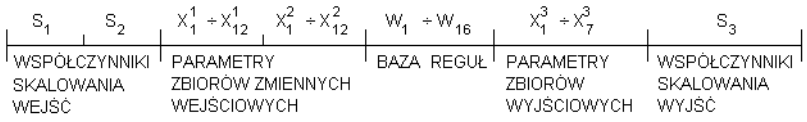
- liczby zmiennych wejściowych,
- przestrzeni poszczególnych zmiennych,
- liczby zbiorów rozmytych dla każdej zmiennej,
- typu zbiorów rozmytych (figur ich reprezentujących),
- gęstość próbkowania przestrzeni przy opisie zbiorów rozmytych (współrzędnych figur).

W pierwszym etapie dodatkowo może wystąpić opracowanie przez eksperta położenia zbiorów rozmytych (współrzędnych figur reprezentujących te zbiory - zarówno dla procesu fuzzyfikacji, jak i defuzzyfikacji), lub bazy reguł. W przypadku procesów de- i fuzzyfikacji możliwe jest również określenie przez eksperta tylko współczynników skalujących (dokładniejszy opis w rozdziałach 4.1. i 4.3.).

Tworzenie chromosomów polega na określeniu dla parametrów nie opracowanych dotąd przez eksperta:

- dla zbiorów rozmytych - współrzędnych je charakteryzujących,
- dla współczynników skalowania - wartości z odpowiedniego zakresu,
- dla bazy reguł - numerów zbiorów zmiennych wyjściowych, dla danej reguły, lub wartości wyróżnionej, jeśli reguła nie daje wyników (brak reguły).

Przykładowy genotyp przedstawiony jest poniżej:



Rys. 4.4.1. Przykład genotypu przy optymalizacji sterownika rozmytego

Ocena całych genotypów dokonywana jest na podstawie przeprowadzenia symulowanej pracy sterownika, z wykorzystaniem parametrów zakodowanych w poszczególnych chromosomach. Oceny dokonuje się przez:

- porównanie z opracowanymi charakterystykami teoretycznymi,
- porównanie z danymi z serii pomiarów doświadczalnych.

Funkcja fitness określana jest w tym przypadku za pomocą funkcji błędu i liczby reguł (podlegającej minimalizacji). Przykładowe funkcje przedstawione są poniżej :

$$F1 = g1 * \frac{\sqrt{(\Sigma_i (xi-xi')^2)}}{n} + g2 * lr$$

$$F2 = g1 * \frac{\Sigma_i |xi-xi'|}{n} + g2 * lr$$

Gdzie: xi jest wartością wzorcową dla danego pomiaru, xi' - wartością otrzymaną z optymalizowanego sterownika, n - liczba pomiarów, g1 - waga minimalizacji błędu, lr - liczba reguł, g2 - waga minimalizacji liczby reguł.

Należy tu zwrócić uwagę, że minimalizacja liczby reguł, choć korzystnie wpływająca na działanie sterowników rozmytych (szybkość), nie może zdominować procesu optymalizacji. Podobnie, optymalizacja wszystkich parametrów (fuzzyfikazji, bazy reguł, defuzzyfikacji), choć możliwa, może dać gorsze rezultaty, aniżeli optymalizacja tylko ich części.

Ciekawy przykład optymalizacji procesu tworzenia sterowników rozmytych podany jest w [Hei ??]. Sterownik posiadał trzy sygnały wejściowe i jeden wyjściowy. Każdej zmiennej odpowiadało siedem zbiorów rozmytych. Po procesie optymalizacji (wszystkich parametrów) otrzymano w 22614. pokoleniu następujące wyniki:

- sygnał wejściowy 1. - całkowicie odcięty,
- sygnał wejściowy 2. - 1 funkcja przynależności,
- sygnał wejściowy 3. - 2 funkcje przynależności,
- liczba reguł aktywnych - 3 z 512 możliwych,
- największe odchylenie od wartości wzorcowej - 1,77 %,
- średnie odchylenie od wartości wzorcowej - 0,61 %.

4.5. Sterowniki hybrydowe rozmyto - proporcjonalno - całkująco - różniczkujące

Pokaźne miejsce w teorii sterowania zajmują sterowniki PID. Skrót PID oznacza w tym przypadku „Proportional - Integral - Derivative", czyli proporcjonalno - całkująco - różniczkujące. Ich szerokie

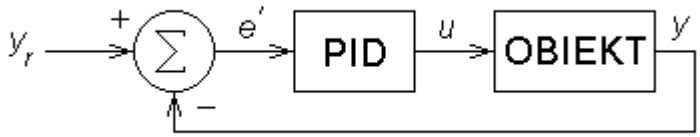
zastosowanie w przemyśle wynika z możliwości uwzględniania wartości zarówno z bardzo odległych w czasie pomiarów (człon całkujący), jak i dynamicznych zmian ostatniej chwili (człon różniczkujący).

Działanie sterowników PID można opisać następującym wzorem:

$$u(k)=Kp*e(k)+(Ki/T)*\Sigma_{n=0\div k}e(n)+(Kd*T)*\Delta e(k) \qquad (4.5.1.)$$

gdzie $e(k)=y(k)-y_r(k)$ oraz $\Delta e(k)=e(k)-e(k-1)$ (4.5.2.)

u(k) oznacza tu wyjście sterownika PID, $y_r(k)$ wartość żądaną, y(k) aktualną wartość z pomiaru.



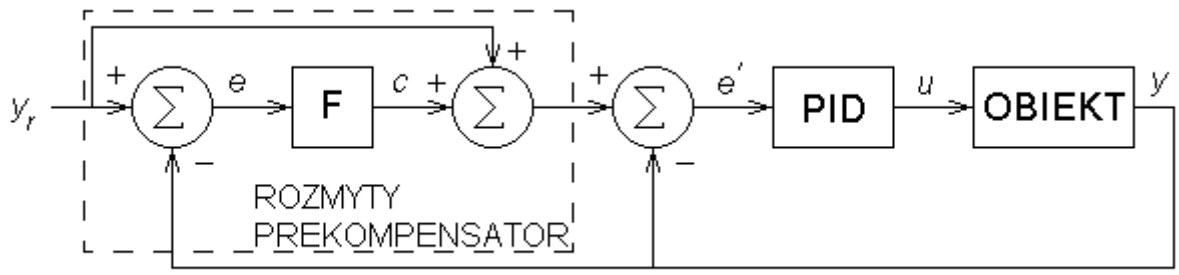
Rys. 4.5.1. Schemat blokowy sterownika PID

Dostrajanie sterowników PID jest zajęciem bardzo żmudnym, w związku z tym, dokonywane były próby zastosowania do tego celu sterowników rozmytych.

Temat ten został opisany w [Cho 97]. Zaproponowane rozwiązania można podzielić na dwa podejścia:

- modyfikacje wartości żądanej przez prekomprensator rozmyty,
- modyfikacje wzmocnienia.

Pierwszy pomysł, polegający na dodaniu do sterownika PID rozmytego prekompensatora, przedstawiony jest poniżej:



Rys. 4.5.2. Sterownik PID z rozmytym prekompensatorem

Modyfikacja wartości żądanej wprowadza nowe zmienne błędów:

$$ep = y_r(k) \blacklozenge Fp(e,\Delta e) - y(k)$$
$$ei = y_r(k) \blacklozenge Fi(e,\Delta e) - y(k)$$
$$ed = y_r(k) \blacklozenge Fd(e,\Delta e) - y(k) \qquad (4.5.3.)$$

gdzie ♦ może być działaniem dodawania, lub mnożenia, a Fp, Fi oraz Fd są funkcjami nieliniowymi e i Δe. Związany z nimi wzór określający działanie sterownika wygląda następująco:

$$u(k)= Kp*ep(k)+(Ki/T)*\Sigma_{n=0\div k}\, ei(n)+(Kd*T)*(ed(k)-ed(k-1)) \qquad (4.5.4.)$$

Druga metoda polega na modyfikacji wzmocnień. Sterownik rozmyty działa w tym wypadku równolegle ze sterownikiem PID. Wzór określający działanie hybrydowego sterownika przedstawiony jest poniżej:

$$u(k) = \begin{cases} Kp*e(k)+(Ki/T)*\Sigma_{n=0\div k}\, ei(n)+(Kd*T)*\Delta e(k) & \text{gdy } (e,\Delta e) \text{ należy do R} \\ G(e,\Delta e) & \text{w przeciwnym przypadku} \end{cases} \qquad (4.5.5.)$$

gdzie R jest częścią przestrzeni (e,Δe).

W tym przypadku dobrze dostrojony sterownik rozmyty przejmuje sterowanie (funkcja G(e,Δe)) poza obszarem okolic stanów ustalonych, w których sterowaniem zajmuje się sterownik PID. Pozwala to połączyć szybkość odpowiedzi sterownika rozmytego, reagującego na nagłe zmiany, z dokładnością sterownika PID, pozwalającą na dokładne ustalenie w obiekcie żądanej wartości.

Metoda ta nazywana jest modyfikacją wzmocnień, gdyż nowe wartości współczynników Kp, Ki oraz Kd wynoszą :

Kp' = Kp + Gp(e,Δe)
Ki' = Ki + Gi(e,Δe)
Kd' = Kd + Gd(e,Δe) (4.5.6.)

W [Cho 97] znaleźć można również opis eksperymentu - tworzenia prekompensatora rozmytego za pomocą algorytmów genetycznych.

Struktura reguły prekompensatora została ustalona następująco :

IF e=Le AND Δe=LΔe THEN c=Lc

gdzie Le, LΔe oraz Lc były nazwami zbiorów rozmytych zmiennych e, Δe i c.

Dla każdej zmiennej wyróżniono 7 zbiorów rozmytych (co dawało 49 reguł dla kompletnej bazy reguł).

Algorytm genetyczny działał dwufazowo :

- w pierwszej fazie optymalizacji poddano bazę reguł sterownika. Genotyp stanowiło 49 genów - pól tabeli zależności c(e,Δe).
- w drugiej fazie dokonano dostrajania sterownika przy pomocy regulacji parametrów funkcji przynależności do zbiorów rozmytych (małe modyfikacje współrzędnych trójkątów). Genotyp stanowiły 3 chromosomy, z których każdy stanowiło 19 genów - wartości modyfikacji współrzędnych 7 trójkątów).

Jako funkcję kosztu zastosowano cztery różne funkcje i ich kombinacje:

- minimalizującą błąd:
  $C1=(\Sigma\sqrt{(el^2)})/N$,
- opisującą stopień reakcji:
  $C2=\alpha*tr+\beta*Mp+\gamma*ts$
  gdzie tr - czas wzrostu, Mp - wartość szczytowa, ts - czas ustalania,
- wygładzającą bazę reguł:
  $C3=\Sigma_{i=1\div7}\,\Sigma_{j=1\div7;i+j\neq odd}\,(\Sigma_{k=i-1\div i+1}\,(Rij-Rkj)+\Sigma_{m=j-1\div j+1}\,(Rij-Rim))$
- minimalizującą energię sterowania:
  $C4=\Sigma ul^2/2$

Po wykonaniu prób, optymalna ze względu na wartość energii sterowania okazała się kombinacja 18*C1+0,005*C3+15*S4. Co ciekawe, wykluczenie z kryterium optymalizacji członu C3, odpowiadającego za wygładzenie bazy reguł, spowodowało zwiększenie wartości energii sterowania o prawie 5%...

4.6. Sterowniki PID realizowane za pomocą sterowników rozmytych

Sterowniki PID mogą być również realizowane w całości przy pomocy sterowników rozmytych. Przypominam - działanie sterowników PID, można przedstawić następująco:

$u(k)=Kp*e(k)+(Ki/T)*\Sigma_{n=0\div k}e(n)+(Kd*T)*\Delta e(k)$ (4.5.1.)

gdzie $e(k)=y(k)-y_r(k)$ oraz $\Delta e(k)=e(k)-e(k-1)$ (4.5.2.)

u(k) oznacza tu wyjście sterownika PID, $y_r(k)$ wartość żądaną, y(k) aktualną wartość z pomiaru.

Jeżeli sterownik przedstawimy w formie bazy reguł, zależnych od trzech sygnałów - całkującego, proporcjonalnego i różniczkującego, organizmy podlegające optymalizacji przez algorytm genetyczny, mogą składać się z następujących genów:

- wartości parametrów K (szczególnie w przypadku wykorzystywania rozszerzonej formy sygnału różniczkującego),
- wartości reguł - pól tablicy zależności,
- współczynniki skalujące (dla de- i fuzzyfikacji),
- współczynniki charakteryzujące zbiory rozmyte (np. popularne w tym zastosowaniu krzywe Gaussa).

Funkcja fitness określana jest na podstawie obserwacji odchyłek od wartości żądanej w okresie symulowanego sterowania. Dodatkowo, używane są uzależnienia od wartości energii sterującej (wartość, chwilowe przetężenia).

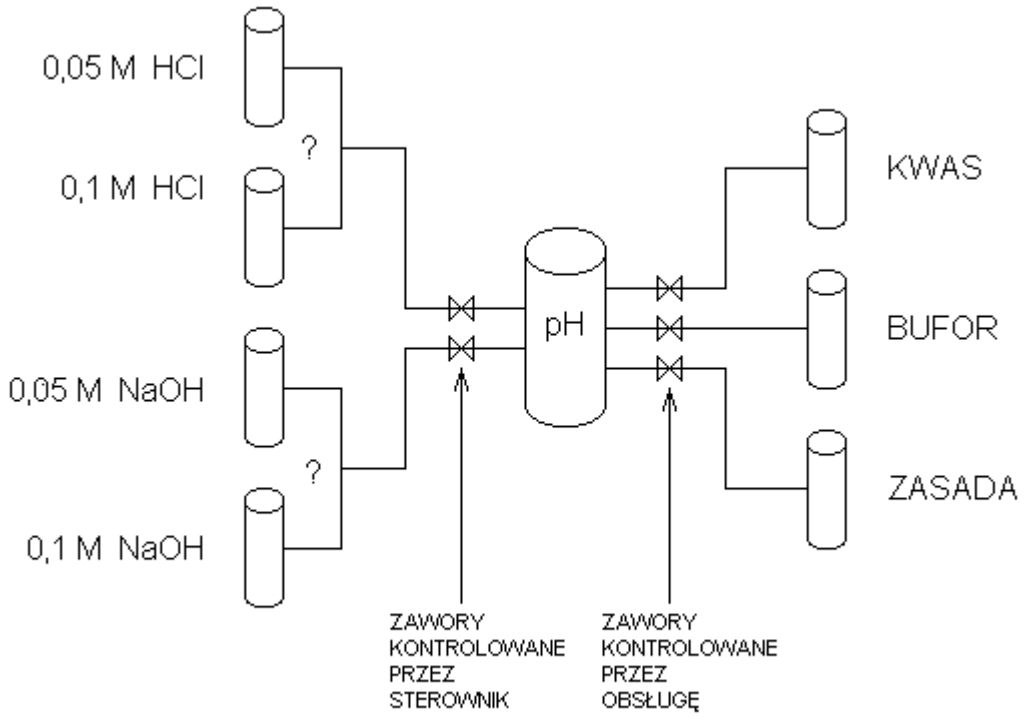
Informacje na temat tworzenia sterowników PD (proporcjonalno - różniczkujących), wraz z przypadkiem rozszerzonej formy sygnału, zawiera [Qi 97]. Natomiast wykorzystanie algorytmów genetycznych w sterownikach PI (proporcjonalno - całkujących), zawiera [Bon ??].

4.7. Sterowniki adaptacyjne

Przy projektowaniu sterowników zdarzają się sytuacje, w których pewne wielkości, od których zależy proces sterowania nie mogą być zmierzone. Zastosowanie algorytmów genetycznych do sterowników rozmytych pozwoliło na adaptację procesu sterowania do zaistniałych warunków, nawet w wypadku całkowitego braku informacji o ważnych czynnikach wpływających na system.

Przykład takiego systemu został opublikowany w [Kar ??]. Ze względu na potężną moc kryjącą się za tym rozwiązaniem, przytoczony zostanie tu skrócony opis, przedstawiający całą ideę właśnie tego przypadku.

Opisany w artykule Karra sterownik ma za zadanie kontrolowanie wartości pH w reaktorze oraz sterowanie zaworami, przez które do reaktora dopływać mogą dodatkowe ilości kwasu (HCl) i ługu (NaOH). Cały system przedstawiony jest poniżej :

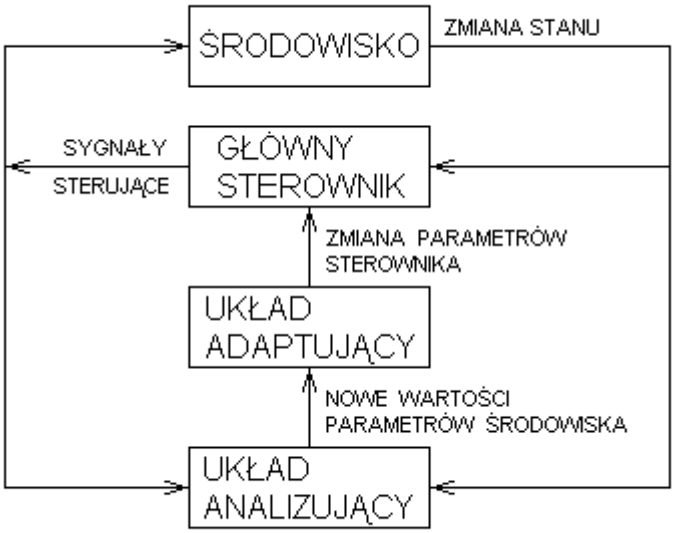


4.7.1. Instalacja reaktora chemicznego, kontrolowana przez adaptacyjny sterownik rozmyty

Obsługa może powodować w dowolnym momencie dopływ do reaktora kwasu, zasady, lub bufora. Bufor - jest to substancja zapewniająca utrzymanie pH w określonym zakresie. Bufor ma przy tym określoną pojemność. Po

przekroczeniu tej pojemności (ilość dodanego kwasu, lub zasady), pH przekroczy zakres. Sterownik, mający za zadanie utrzymanie w reaktorze stałego pH, może w tym celu dodawać zarówno kwas, jak i ług z własnych zasobów. W zasobach sterownika znajduje się kwas i zasada o stężeniu 0,05 M, lub 0,1 M.

Sterownik otrzymuje informacje jedynie o wartości i zmianie pH w danym momencie. Nie ma możliwości odczytu ilości dodanego przez obsługę kwasu, bufora albo zasady. Nie otrzymuje również informacji o aktualnym stężeniu kwasu ani ługu we własnych zbiornikach. Sterownik zarządza jedynie otwarciem i zamknięciem zaworów zbiorników dodatkowych.



Rys. 4.7.2. Struktura adaptacyjnego sterownika rozmytego

Z wyżej opisanych względów proces sterowania w przypadku próby zastosowania sterownika tradycyjnego wydaje się wprost nierozwiązywalny. Opracowano więc układ sterowników rozmytych, z użyciem algorytmów genetycznych, które zapewniają analizę sytuacji i dostosowanie pracy głównego sterownika do aktualnych warunków (rys. 4.7.2.).

Główny sterownik pracuje według stałej bazy reguł, opartej o zależność dwóch zmiennych wejściowych (pH (7 zbiorów rozmytych) i ΔpH (2 zbiory rozmyte)) i dwóch wyjściowych (Qa - przepływ kwasu (4 zbiory) i Qb - przepływ zasady (4 zbiory)). Na działanie głównego sterownika mają wpływ dane z układu adaptującego. Układ adaptujący może bowiem zmieniać współrzędne figur określających funkcje przynależności (zbiory rozmyte) zmiennych wejściowych i wyjściowych.

Układ analizujący na podstawie otrzymywanych wartości pH i ΔpH oraz sygnałów sterujących (Qa i Qb), wykorzystując algorytm genetyczny, odtwarza przebieg pracy reaktora. W genotypie kodowane są parametry, na które wpływ ma obsługa :

- 40 bitów - określających strumień główny kwasu,
- 40 bitów - określających strumień bufora,
- 40 bitów - określających strumień główny zasady,
- 40 bitów - stężenie kwasu zbiornika dodatkowego,
- 40 bitów - stężenie ługu zbiornika dodatkowego.

Funkcja fitness określa dla danego genotypu zgodność z odczytaną wartością i zmianami pH w reaktorze, w przedziale 100 sekund. Wynikiem działania algorytmu genetycznego są brakujące do analizy systemu dane (które były zakodowane w genotypie). Są one przekazywane do układu adaptującego.

Układ adaptujący na podstawie pełnej analizy systemu, dostarczonej przez układ analizujący, wykorzystując algorytmy genetyczne, dobiera optymalne współrzędne dla funkcji przynależności głównego sterownika. Tak więc genotyp zawiera w tym wypadku:

- 12 punktów po 7 bitów - określających współrzędne 7 zbiorów rozmytych opisujących pH (założenie o symetryczności współrzędnych, figury są trapezami dowolnymi);
- 4 punkty po 7 bitów - określających współrzędne 2 zbiorów rozmytych opisujących ΔpH (figury są trójkątami);
- 16 punktów po 7 bitów - określających współrzędne 4 zbiorów rozmytych opisujących przepływ kwasu i ługu ze zbiorników dodatkowych (założenie o jednakowych funkcjach przynależności dla przepływu kwasu i ługu, figury są trapezami dowolnymi).

Funkcja fitness określa dla danego genotypu odchyłkę od wartości żądanej w przeciągu najbliższych 100 sekund (symulacja sterownika głównego). Wynik - współrzędne fukcji przynależności - przekazywane są do głównego sterownika.

4.8. Zastosowania przemysłowe

W literaturze można znaleźć wiele różnych prób zastosowania połączenia algorytmów genetycznych i logiki rozmytej przy tworzeniu sterowników. Główne rodzaje sterowników oraz metody ich optymalizacji zostały przedstawione powyżej. Konkretne przykłady zastosowań w przemyśle oraz bogaty wybór materiałów na temat sterowników rozmytych można znaleźć w [Fuz 78].

Przykładem może być opisany w [Per 98] sterownik mający za zadanie sterowanie procesem mikrofiltracji surowej trzciny cukrowej, istotnym w przemyśle spożywczym.

5. Wykorzystanie AG i LR w zadawaniu pytań do baz danych

W ostatnich latach mamy do czynienia z coraz większym natłokiem informacji. Zalew informacji z różnych źródeł, związany z szybkim rozwojem technik informatycznych spowodował konieczność zmiany systemu gromadzenia i zarządzania danymi. Możliwość wyboru dokumentów na zasadzie dotyczy/nie dotyczy już nie wystarcza.

Rozpatrywany problem polega na wyszukiwaniu w bazie danych pozycji odpowiadających zadanym kryteriom. Jeżeli przyjmiemy:

D - jako zbiór dokumentów,
T - zbiór terminów (słów kluczowych),
R - relacja „dotyczenia" : $\mu r\ (di,A) \in [0,1]$ dla $(di,A) \in \mathbf{D*T}$, (5.1.)

co można zapisać także jako $\mu A(di) \in [0,1]$ - czyli na ile dokument di jest zgodny z terminem A. Np. $\mu A(d3)=0,8$ oznacza, że dokument d3 jest silnie zgodny pod względem tematycznym z terminem A.

Brakiem występującym w zwykłych systemach, w których można zadawać pytania połączone tylko spójnikami LUB albo I jest niemożliwość wymuszenia pewnych odpowiedzi [San 94]. Przykładem może być tu wyszukanie dokumentów odpowiadających pytaniu A LUB B, z nacikiem na A I B.

Aby dopuścić możliwość zadawania takich pytań, definiuje się dodatkowe operacje umożliwiające podanie istotności zgodności z danym terminem, z punktu widzenia całego pytania. I tak definiuje się:

- oparty na t-normie operator $A_a=T(a,A)$, np.

$\mu A_a(d)=\min(a,\mu A(d))$

(czyli maksymalne odchylenie wartości od zera), (5.2.)

- oparty na t-conormie operator $A^a = S(a,A)$, np.

$\mu A^a(d)=\max(1-a,\mu A(d))$

(czyli maksymalne odchylenie wartości od jedności). (5.3.)

Pełny opis algebry operacji A^a i A_a zawiera [San 94]. Przykład użycia tych opracji, zamieszczony poniżej, wykazuje celowość stosowania logiki rozmytej w systemach informacyjnych:

	d1	d2	d3	d4
A	0,2	1,0	0,8	0,3
B	0,8	0,3	1,0	0,2
$A \vee B$	0,8	1,0	1,0	0,3
$A \wedge B$	0,2	0,3	0,8	0,2
$A_{0,5} \vee B_{0,5} \vee (A \wedge B)$	0,5	0,5	0,8	0,3

Jak widać, wysoko są oceniane tu te dokumenty, w których oba terminy są silnie reprezentowane.

Kolejny raz logika rozmyta wkracza do działania przy prezentowaniu odpowiedzi użytkownikowi - zazwyczaj używa się zapytań z progiem Θ: prezentowanie dokumenty należą wtedy do Θ-obcięcia zbioru:

$$Q_\Theta = \{di \in \mathbf{D}, \mu Q(di) \geq \Theta\} \quad (5.4.)$$

gdzie Q-pytanie.

Prócz ustalenia granicznej Θ można założyć informowanie użytkownika o pewnej maksymalnej, z góry określonej ilości najwyżej punktowanych (zgodnych z pytaniem) odpowiedzi [San 95].

Dla systemów logiki dwuwartościowej nie istnieje możliwość wyboru wartości: Θ=1. Przy logice wielowartościowej o wartości Θ może zdecydować użytkownik - np. dla Θ=0,5 zostaną wybrane dla ostatniego pytania dokumenty d3,d1 i d2; natomiast dla Θ=0,75 - tylko d3. Jak widać, dokumenty są prezentowane w kolejności malejącej zgodności z pytaniem.

Efekty wyszukiwania mierzy się za pomocą miar precyzyjności i kompletności (ang. recall) odpowiedzi [San 94], [San 95]. Jeśli oznaczymy:

Rt - zbiór dokumentów wyszukanych (ang. retrieved),

Rl - zbiór dokumentów relewantnych (zgodnych z zadanym pytaniem).

to precyzyjność odpowiedzi: w zbiorach rozmytych:

$$P = \frac{|\mathbf{Rt} \cap \mathbf{Rl}|}{|\mathbf{Rt}|} \quad (\mathbf{Rt} \neq \emptyset) \qquad P = \frac{\Sigma_{dj \in Q\Theta}\ \mu Q(dj)}{|Q_\Theta|} \quad (5.5.)$$

kompletność: w zbiorach rozmytych:

$$R = \frac{|\mathbf{Rt} \cap \mathbf{Rl}|}{|\mathbf{Rl}|} \quad (\mathbf{Rl} \neq \emptyset) \qquad R = \frac{\Sigma_{dj \in Q\Theta}\ \mu Q(dj)}{\Sigma_{dj \in D}\ \mu Q(dj)} \quad (5.6.)$$

Precyzyjność i kompletność to kryteria efektywności systemów wyszukujących, nie można jednak kłaść zbyt dużego nacisku zwłaszcza na kompletność w dużych systemach informacyjnych (przy zbyt dużej ilości dokumentów wyszukanych użytkownik może mieć kłopoty z „przebiciem się" przez nie).

Z tego względu stosuje się funkcje oceniające będące kombinacją precyzyjności i kompletności:

$$E=\alpha*|\mathbf{Rt} \cap \mathbf{Rl}|-\beta*|\mathbf{Rt} \cap \overline{\mathbf{Rl}}|-\gamma*|\overline{\mathbf{Rt}} \cap \mathbf{Rl}|+\delta|\overline{\mathbf{Rt}} \cap \overline{\mathbf{Rl}}| \quad (5.7.)$$

gdzie E jest miarą użyteczności informacji. W sytuacjach rzeczywistych, kiedy dokumenty $\overline{\mathbf{Rt}}$ są nieznane, stosuje się ograniczoną postać formuły (5.7.):

$$E' = \alpha * |\mathbf{Rt} \cap \mathbf{Rl}| - \beta * |\mathbf{Rt} \cap \overline{\mathbf{Rl}}| \qquad (5.8.)$$

Funkcje te (5.7. i 5.8.) mogą zostać użyte w algorytmie genetycznym jako funkcje fitness. Nie jest to proste, w przypadku, gdy nie zastosowano w rozwiązaniu logiki rozmytej (dokładny opis w [Pet ??]), jeżeli bowiem przez ri oznaczymy wyszukanie przez system artykułu di (1 - gdy artykuł w odpowiedzi na dane pytanie został znaleziony, 0 - jeśli nie), a przez fi zgodność z kryteriami pytania (1 - gdy artykuł spełnia kryteria pytania, 0 - jeśli nie), okaże się, że funkcja fitness:

$$E = \alpha * \frac{\Sigma_{i=1 \div n} (ri * fi)}{\Sigma_{i=1 \div n} ri} + \beta * \frac{\Sigma_{i=1 \div n} (ri * fi)}{\Sigma_{i=1 \div n} fi} \qquad (5.9.)$$

nie będzie spełniała naszych oczekiwań, ze względu na to, że w logice dwuwartościowej fi wynosi 1 tylko przy całkowitej zgodności dokumentów z pytaniami - wartość niezerowa w liczniku będzie pojawiała się dosyć rzadko. W związku z tym, populacje będą zdominowane przez osobniki o zerowym fitness.

Z porównania z wzorem teorii schematów:

$$m_{t+1}(s) > m_t(s) * \frac{f}{f_{śr}} * \alpha \qquad (5.10.)$$

gdzie $m_j(s)$ - liczba wystąpień schematu s w populacji j, f - fitness organizmu s, $f_{śr}$ - średni fitness populacji (ciśnienie selekcyjne).

jasno wynika, że duża liczba osobników o zerowym fitness wywołuje bardzo niskie ciśnienie selekcyjne.

Działanie algorytmu genetycznego w tym przypadku można opisać schematem ([Pet ??]):

Rys. 5.1. Wykorzystanie algorytmów genetycznych w zadawaniu pytań

Obiektem działania algorytmu genetycznego są pytania, zadawane bazie danych. Optymalizacji mogą podlegać:

- wagi związane z poszczególnymi terminami w zapytaniu,
- wagi związane z poszczególnymi operatorami w zapytaniu,
- wagi i terminy zawarte w zapytaniu.

Pierwsze podejście (wagi terminów) jest obecnie najbardziej rozpowszechnione. Pytanie zadawane przez użytkownika na początku zawiera losowe wagi, przydzielone przez komputer. Proces ewolucyjny, wykorzystujący wyżej opisane funkcje fitness, poprzez optymalizację wag, odnajduje dokumenty najbardziej odpowiadające pytaniu użytkownika.

Drugie podejście (wagi operatorów) nie jest popularne, być może ze względu na kłopot z natychmiastową interpretacją przez człowieka takiego zapisu:

(t1 $\underline{\text{AND}}_{w1}$ (t3 $\underline{\text{OR}}_{w2}$ t5)) $\underline{\text{AND}}_{w3}$ (t4 $\underline{\text{OR}}_{w4}$ t5)

Trzecie podejście, dziś uznawane już za niezbyt właściwe, polegało na krzyżowaniu całych pytań, zapisanych w formie prefiksowej. Przykład takiej metody został przedstawiony w [Pet ?1]. W tym wypadku również możliwe jest użycie wag opisujących zarówno operatory, jak i terminy. Ze względu na możliwą zmienną długość genotypów opisujących pytania, operacja krzyżowania została zdefiniowana w sposób następujący :

- dla pytania Q1 o długości n1 odpowiadającego mu ciągu S1 i pytania Q2 o długości n2 odpowiadającego mu ciągu S2, losowo wybierane są liczby c1∈[1,n1] i c2∈[1,n2];
- nowe organizmy powstają przez wymianę najmniejszych możliwych fragmentów ciągów, związanej z wylosowaną pozycją w genotypie.

Oto przykład:

```
pozycje :  1    2    3    4    5    6    7    8    9    10   11   12   13
S1      :  (or  (and t1   t2)  (and t3   (or  t4   t5)))
S2      :  (or  (and (or  t1   t2)  (or  t1   t3)) (and t4   t5))
```

przy wylosowaniu c1=3 i c2=2 rezultatem krzyżowania będą:

```
S1'     :  (or  (and (or  t1   t2)  (or  t1   t3)) (and t3   (or  t4   t5)))
S2'     :  (or  (and t1   t2)  (and t4   t5))
```

Pokazane tu zasady krzyżowania związane są z zagadnieniem programowania genetycznego, opracowanego przez Kozę. Więcej informacji na ten temat można znaleźć w [Koz 93].

Konkretne zastosowanie algorytmów genetycznych i logiki rozmytej w problemie wyszukiwania informacji można znaleźć w [San 95]. Opisana jest tam aplikacja działająca w bazie patentów inżynierii biomedycznej.

6. Wykorzystanie AG i LR w modelowaniu reakcji chemicznych

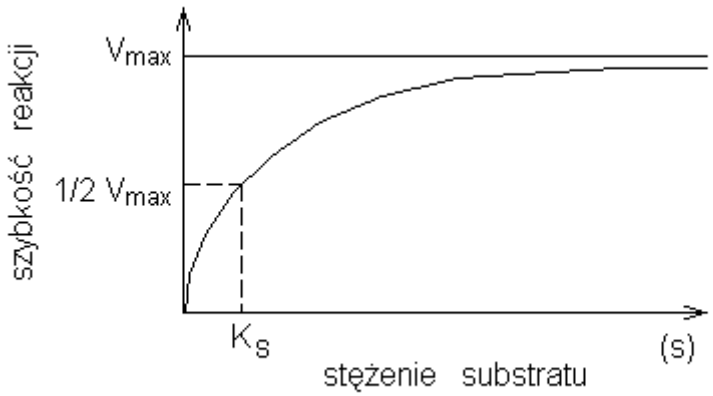
Wykorzystanie kombinacji algorytmów genetycznych i logiki rozmytej występuje również przy analizie przebiegu reakcji chemicznych. Podejście takie jest szczególnie użyteczne, w przypadku, gdy dysponuje się informacją niekompletną. Tak jest np. w przypadku modelowania reakcji związanych z metabolizmem.

Proces metabolizmu glukozy polega na jej przejściu, poprzez działanie z szeregiem enzymów, skończywszy na cyklu przemian, znanym jako cykl Krebsa. Przy tym przemiany te charakteryzują się dość specyficznym przebiegiem. Jednocześnie na danym etapie zachodzą bowiem trzy różne reakcje:

$E+S \rightarrow ES$ z szybkością k1,
$ES \rightarrow E+S$ z szybkością k2,
$ES \rightarrow E+P$ z szybkością k3, (6.1.)

przy czym E-enzym, S-substrat, P-produkt końcowy, ES-kompleks enzym-substrat. Biorąc pod uwagę, że jednocześnie proces metabolizmu przebiega na wielu różnych etapach, widać, że jego modelowanie nie jest absolutnie sprawą prostą.

Podstawy kinematyki enzymatycznej stworzyła hipoteza Michaelisa i Mentena w 1913 roku. Według niej szybkość reakcji enzymatycznej zależy od stężenia substratu:



Rys. 6.1. Zależność szybkości przebiegu reakcji od stężenia substratu

Dodatkowo, na szybkość reakcji wpływa stężenie aktywatorów.

Ze względu na brak możliwości uzyskania pełnych danych doświadczalnych (dostępne jedynie ograniczone wartości pomiarów z organizmu), wypróbowano połączenie logiki rozmytej z algorytmami genetycznymi dla uzyskania pełnych charakterystyk.

Problem został sprowadzony do stworzenia sterownika rozmytego, generującego krzywe zależności szybkości reakcji od stężenia substratu, dla danych stężeń aktywatorów. Sterownik ten był optymalizowany przez algorytm genetyczny, dokonujący porównania wartości generowanych przez sterownik z wartościami dyskretnymi, pochodzącymi z pomiarów.

Ciekawostką jest w tym przypadku, użycie niestandardowego sterownika rozmytego, określanego jako model „Sugeno-Takagi-Kang", a korzystającego z reguł postaci:

IF xr1=L1 AND xr2=L2 THEN z=a0+a1*x1+a2*x2 (6.2.)

gdzie x1,x2 są wartościami sygnałów wejściowych, xr1 oraz xr2 - wartościami rozmytymi sygnałów wejściowych, a L1 i L2 - terminami językowymi odpowiadającymi zbiorom rozmytym zmiennych xr1 i xr2; z oznacza tu wartość wyjściową (nie rozmytą), a a0,a1 oraz a2 wartości współczynników, optymalizowanych za pomocą algorytmu genetycznego.

Konkretna wartość wyprowadzana na wyjście jest średnią ważoną wartości otrzymanych z poszczególnych reguł.

W artykule [Yen ??] opisane jest wykorzystanie opisanych wyżej metod do modelowania dwóch reakcji kwasu 2-fosfoenolopirogronowego. Więcej na temat samej kinematyki procesów enzymatycznych można znaleźć w [Tro 78]. Tematowi optymalizacji sterowników rozmytych poświęcony jest natomiast rozdział 4. tego opracowania. Na temat zaś samego sterownika „Sugeno-Takagi-Kang" więcej informacji zawiera [Dri 96].

7. Wykorzystanie AG i LR w problemach podziału i klasyfikacji

Szerokim zastosowaniem algorytmów genetycznych i systemów rozmytych jest zagadnienie podziału zbiorów oraz pokrewne mu - klasyfikacji obiektów w określonej przestrzeni (w praktyce problem podziału przestrzeni).

Poniżej opisane są dwa przykłady zastosowania algorytmów genetycznych w tematyce podziału zbiorów. Pierwszy z nich dotyczy równomiernego rozdziału zadań pomiędzy np. grupy robocze. Drugi - dotyczy problemu klasyfikacji wzorów w przestrzeni, przy użyciu logiki rozmytej i algorytmów genetycznych.

Problem podziału zbiorów występuje często w przypadku potrzeby równomiernego rodziału zadań na maszyny liczące, rozkładu urządzeń na pokładzie statku (masa), napełnień poszczególnych komór w cysternie itp.. Prosty algorytm genetyczny służący do optymalizacji rozwiązań tego problemu opisany jest w rozdziale 7.1..

Problem klasyfikacji zaś występuje przy rozpoznawaniu obrazów, pisma (OCR), podziału elementów zbioru w zależności od pewnych własności itp.. Przykładowy algorytm stosowany do rozwiązywania tego rodzaju problemów można znaleźć w rozdziale 7.2..

7.1. Problem podziału zbioru

Problem podziału zbioru polega na równomiernym rozdziale zadań.
Jeżeli przez ES oznaczymy zbiór wszystkich zadań:

$ES=\{S_1, ... , S_n\}$ (7.1.1.)

problem polega na podziale na podzbiory spełniające następujące warunki:

$ES_i : \forall i.ES_i<>\phi , \forall i,j,i\neq j. ES_i \cap ES_j=\phi, \bigcup j\ ES_j=ES$ (7.1.2.)

Przy czym podział ten jest tym lepszy, im bardziej równomierny jest „ciężar" elementów w podzbiorach:

$b_j=\Sigma_{Sl \in ESi}\ fb(S_l)$ (7.1.3.)

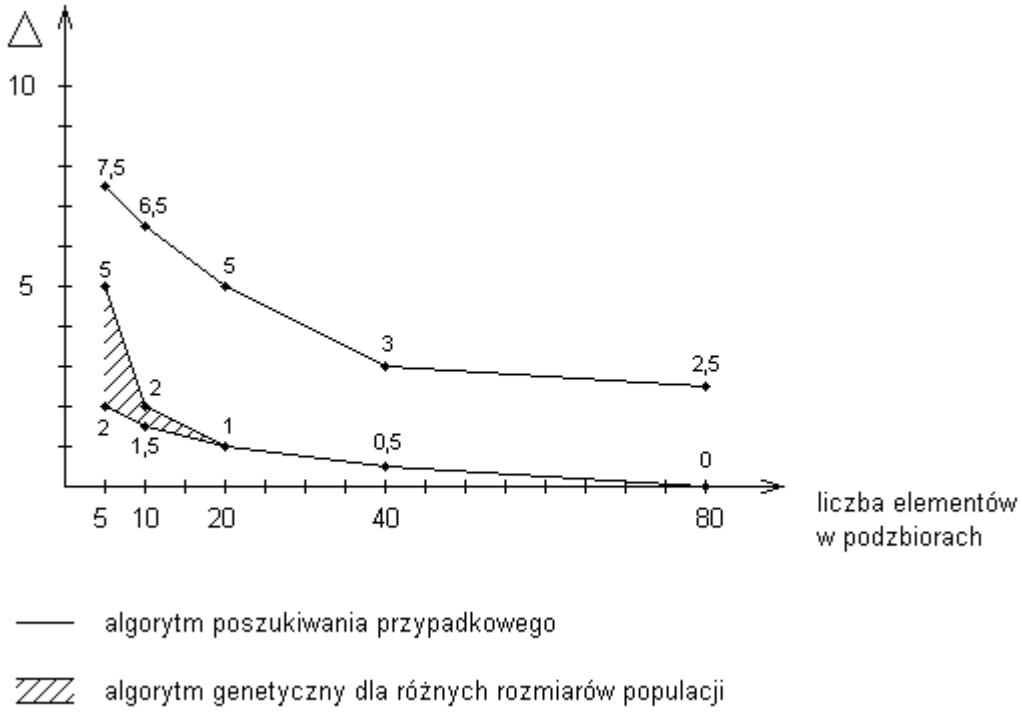
$\Delta=\Sigma_{i=1\div m-1}\ \Sigma_{j=i+1\div m}(|b_i-b_j|)$ (7.1.4.)

gdzie fb(S) jest funkcją wartościującą ciężar danego elementu S, b_i - ciężarem podzbioru zadań, Δ - określa różnice pomiędzy poszczególnymi podzbiorami (właśnie ta wartość podlega minimalizacji).

Algorytm genetyczny, zaproponowany w [Gom 98], opiera się na następujących założeniach:

- genotyp jest wektorem n liczb z przedziału [1,m] określającym przyporządkowania zadań $S_1 \div S_n$ do podzbiorów $ES_1 \div ES_m$,
- populacja początkowa o rozmiarze p (przypadkowe rozłożenie elementów S_l w zbiory ES_i),
- genotyp o $ES_j=\phi$ (dla dowolnego j) jest odrzucany i zastępowany innym (nie może istnieć podzbiór pusty),
- funkcja fitness - całkowita różnica podziału na zbiory (wzór 7.1.4.),
- w danej populacji 50% rozwiązań jest zastępowanych zmutowanymi,
- mutacja polega na przeniesieniu q wylosowanych elementów S_l z podzbioru ES_i do podzbioru ES_j,
- na początku działania algorytmu q jest równe 10, co 0,1 założonych etapów ewolucji jest zmniejszane o 1,
- zakończenie działania algorytmu następuje, gdy znalezione zostanie optimum (Δ=0), lub q zmaleje do 0.

Przeprowadzone badania sprawdzały zachowanie algorytmu ze względu na zmianę liczby elementów w podzbiorach, liczby podzbiorów, wariancji liczby elementów w każdym podzbiorze. Dodatkowo, porównaniu poddano algorytm poszukiwania przypadkowego. Bez względu na wyżej wymienione parametry algorytm genetyczny dawał w tej samej liczbie etapów ewolucji co algorytm poszukiwań przypadkowych wyniki Δ o około 5 razy mniejsze. W niektórych przypadkach stosunek ten był jeszcze bardziej korzystny dla algorytmu genetycznego i dochodził do 100 (!) krotnie mniejszej wartości parametru Δ dla algorytmu genetycznego. Przykładowy wynik badań dla wysokiej wariancji liczby elementów w podzbiorach przedstawiony jest na rysunku:



Rys. 7.1.1. Przykładowe wyniki porównania algorytmu genetycznego i algorytmu poszukiwania przypadkowego (za [Gom 98])

Pełne wyniki badań można oczywiście znaleźć w [Gom 98], wykres zamieszczony powyżej ilustruje jedynie tendencje wykryte w czasie badań.

7.2. Problem klasyfikacji elementów

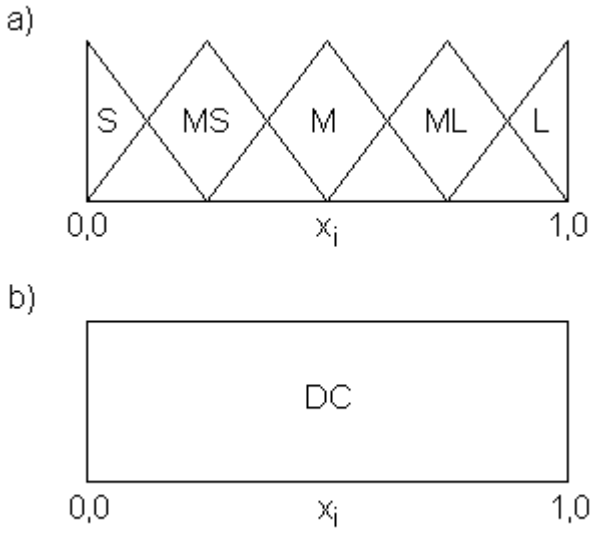
Ciekawym zagadnieniem, wykorzystywanym np. przy rozpoznawaniu obiektów takich jak obraz, dźwięk, pismo itp. jest problem klasyfikacji wzorów. Zakłada się tu, że wzór jest punktem w n-wymiarowej przestrzeni wzorów $[0,1]^n$. Można więc przyjąć, że wzór jest dowolnym obiektem, opisywanym za pomocą n znormalizowanych do zakresu [0,1] zmiennych.

Obiekty położone w określonych punktach przestrzeni $[0,1]^n$ mogą być zaliczone do określonych klas. Na tym polega m.in. zagadnienie rozpoznawania pisma (OCR).

Z punktu widzenia danej zmiennej (1 wymiar w przestrzeni $[0,1]^n$), określona klasa może być od niej zależna lub nie. Aby zachować typowy schemat sterownika rozmytego (patrz rozdział 4.):

IF x1=L1 AND x2=L2 AND ... AND xn=Ln THEN wzór=KLASA (7.2.1.)

gdzie x1÷xn to rozmyte zmienne wejściowe, a L1÷Ln - terminy lingwistyczne, opisujące związane z nimi zbiory rozmyte, konieczne jest określenie oprócz zwykłych funkcji przynależności (rys. 7.2.1.a) dodatkowej funkcji „Don't Care" (rys. 7.2.1.b).



Rys. 7.2.1. a)- pięć „tradycyjnych" zbiorów rozmytych kojarzonych z wartościami zmiennych rozmytych, b)- dodatkowy zbiór rozmyty dla zachowania jednakowego schematu wszystkich reguł

Optymalizowany sterownik, mający za zadanie klasyfikację wzorów, charakteryzuje się następującymi właściwościami:

- wszystkie reguły występują w postaci (7.2.1.),
- mogą istnieć reguły „nie odpalane",
- mogą istnieć reguły, klasyfikujące wzory do klasy pustej.

Reguły „nie odpalane" oraz klasyfikujące wzory do klasy pustej nie mają wpływu na klsyfikację danego obiektu. Reguły związane z klasą pustą są natomiast przydatne do określania obszarów podległych nie wyróżnianym w danym momencie klasom.

Algorytm genetyczny optymalizuje bazę reguł sterownika w następujący sposób:

- genotypy zawierają kompletną informację o bazie reguł, wraz z dodatkową zmienną, określającą przynależność danej reguły do grupy „ważych", związanych z klasą pustą oraz „nie odpalanych",
- funkcja fitness jest zależna od ilości (podlegającej maksymalizacji) dobrze sklasyfikowanych wzorów przykładowych (ciąg testujący) oraz ilości „ważnych" reguł (podlegającej minimalizacji).

Sterownik ukształtowany w ten sposób może być stosowany do rozpoznawania obiektów w sytuacjach rzeczywistych. Należy jednak wspomnieć tu o tym, że:

- jakość sterownika zależy silnie od zastosowanego ciągu testującego (musi on być reprezentatywny dla rozpoznawanych podczas rzeczywistej pracy obiektów),
- w przypadku możliwych zmian warunków pracy, można zastosować sterowniki adaptacyjne (patrz punkt 4.7.), które mogą radzić sobie np. z rozpoznawaniem pisma w systemie użytkowanym przez wiele osób.

Zagadnienie optymalizacji sterownika rozpoznającego obiekty opisane zostało szerzej w [Ish 97]. Często stosowane jest również połączenie logiki rozmytej oraz sieci neuronowych. Przykład taki został opisany w [Fuz ?1].

8. Zastosowania w ekonomii

9. Literatura

[Bie 96] Bielecki R., „Logika rozmyta w Systemach Ekspertowych", praca magisterska, Politechnika Wrocławska 1996 (maszynopis, ze zb. dr H. Kwaśnickiej).

[Bon ??] Bonissone P.P., Chiang K.H. „A knowledge-based system view of fuzzy logic controllers" w: Yager R.R., Zadeh L.A. (red.) „Fuzzy sets, neural networks and soft computing", (ze zb. dr H. Kwaśnickiej).

[Bro 95] Bronsztejn I.N., Siemiendiajew K.A. „Matematyka - Poradnik encyklopedyczny", PWN Warszawa 1995.

[Cho 97] Cho H.J., Cho K.B., Wang B.H. „Fuzzy-PID hybrid control: Automatic rule generation using genetic algorithms", Fuzzy Sets and Systems vol. 92 (1997), s. 305÷316.

[Cor ??] Cordon O., Herrera F., Lozano M. „On the bidirectional integration of genetic algorithms and fuzzy logic", (ze zb. dr H. Kwaśnickiej).

[Cor 96] Cordon O., Herrera F., Lozano M. „A classified review on the combination FL-GA bibliography", (1996), (http://decsai.ugr.es/~herrera/fl-ga.html).

[Dri 96] Driankov D., Fellendoorn H., Reinfrank M. „Wprowadzenie do sterowania rozmytego", WNT Warszawa 1996.

[Fuz 78] „Fuzzy Sets and Systems" (czasopismo), Elsevier Science, Amsterdam, 1978÷.

[Fuz ?1] „Fuzzy artmap: a synthesis of neural networks and fuzzy logic for supervised categorization and nonstationary prediction" w: Yager R.R., Zadeh L.A. „Fuzzy sets, neural networks and soft computing", (ze zb. dr H. Kwaśnickiej).

[Gom 98] Gomme P., Harrald P.G. „Applying evolutionary programming to selected set partitioning problems", Fuzzy Sets and Systems vol. 95 (1998), s. 67÷76.

[Hei ??] Heider H., Tryba V., Mühlenfeld E. „Automatic design of fuzzy systems by genetic algorithms", (ze zb. dr H. Kwaśnickiej).

[Her 97] Herrera F., Lozano M., Verdegay J.L. „Fuzzy connectives based crossover operators to model genetic algorithms population diversity", Fuzzy Sets and Systems, vol. 92 (1997), s. 21÷30.

[Her ?1] Herrera F., Lozano M., Verdegay J.L. „Dynamic and heuristic fuzzy connectives based crossover operators for controlling the diversity and convergence of real-coded genetic algorithms", (http://decsai.ugr.es/difuso/tr.html).

[Her ?2] Herrera F., Lozano M., Verdegay J.L. „Tackling real-coded genetic algorithms operators and tools for behavioural analysis", (http://decsai.ugr.es/difuso/tr.html).

[Her 95] Herrera F., Lozano M., Verdegay J.L. „The use of fuzzy connectives to design real-coded genetic algorithms", (http://decsai.ugr.es/difuso/tr.html).

[Hry 96] Hryń R. „Zastosowanie algorytmów genetycznych w systemach rozmytych", Politechnika Wrocławska 1996 (maszynopis, ze zb. dr H. Kwaśnickiej).

[Ish 97] Ishibuchi H., Murata T., Türksen I.B. „Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems", Fuzzy Sets and Systems vol. 89 (1997), s.135÷150.

[Kar ??] Karr C. „Adaptive control with fuzzy logic and genetic algorithms" w: Yager R.R., Zadeh L.A. „Fuzzy sets, neural networks and soft computing", (ze zb. dr H. Kwaśnickiej).

[Kin 98] Kindu S., Chen J. „Fuzzy logic or Lukasiewicz logic - a claryfication", Fuzzy Sets and Systems vol. 95 (1998), s. 369÷379.

[Koz 93] Koza J. R. „Genetic programming. On the programming of computers by means of natural selection", Bradford book Cambridge 1993.

[Lee 93] Lee M.A., Takagi H. „Dynamic control of genetic algorithms using fuzzy logic techniques", (http://www.bioele.nuee.nagoya-u.ac.jp/wsc1/papers/list_abst.html).

[Per 98] Perrot N., Me L., Trystram G., Trichard J.-M., Decloux M. „Optimal control of the microfiltration of sugar product using a controller combining fuzzy and genetic approaches", Fuzzy Sets and Systems vol. 94 (1998), s. 309÷322.

[Pet ??] Petry F.F. ,Buckles B.P., Prabhu D., Kraft D.H. „Generating fuzzy information retrieval queries via genetic programming", (ze zb. dr Kwaśnickiej).

[Pet ?1] Petry F.F. ,Buckles B.P., Prabhu D., Kraft D.H. „Fuzzy information retrieval using genetic algorithms and relevance feedback", (ze zb. dr Kwaśnickiej).

[Qi 97] Qi X.M., Chin T.C. „Genetic algorithms based fuzzy controller for high order systems", Fuzzy Sets and Systems vol. 91 (1997), s. 279÷284.

[Rut 97] Rutkowska D., Piliński M., Rutkowski L. „Sieci neuronowe, algorytmy genetyczne i systemy rozmyte", PWN Warszawa 1997.

[San 94] Sanchez E., Pierre P. „Fuzzy logic and genetic algorithms in information retrieval", (ze zb. dr Kwaśnickiej).

[San 95] Sanchez E., Miyano H., Brachet J.P. „Optimization of fuzzy queries with genetic algorithms. Appliction to a data base of patents in biomedical engineering", (ze zb. dr Kwaśnickiej).

[Tro 78] Trojanowski J. „Biochemia dla biologów", PWN Warszawa 1978.

[Woe 95] Woehr J. „Wizje Lotfiego" Software nr 4/1995, s. 28÷35.

[Yen ??] Yen J., Lee B., Liao J.C. „Using fuzzy logic and a hybrid genetic algorithm for metabolic modeling", (http://www.bioele.nuee.nagoya-u.ac.jp/wsc1/papers/list_abst.html).

Spis treści