I Krajowa Konferencja
Algorytmy Ewolucyjne
Murzasichle, 12-15.06.1996

Optymalizacja reguł decyzyjnych dla systemów rozmytych przy użyciu algorytmu ewolucyjnego ze zmienną wielkością populacji.

Marcin Przeradzki, e-mail: przeradz@bolek.ii.pw.edu.pl

1. WSTĘP

Wygenerowanie optymalnego zbioru reguł dla systemów rozmytych nie jest zadaniem prostym, gdyż wymaga rozpatrzenia wielu przypadków. Algorytm ewolucyjny pozwala w szybki sposób znaleźć w przestrzeni rozwiązań obszar, który spełnia pożądane warunki. Szukamy takiego zbioru reguł, który daje jak najmniejszy błąd i jego wielkość jest jak najmniejsza. W artykule pokazano sposób wyznaczania przy pomocy algorytmu ewolucyjnego bazy reguł aproksymujących na zadanym przedziale funkcję nieliniową dwóch zmiennych.

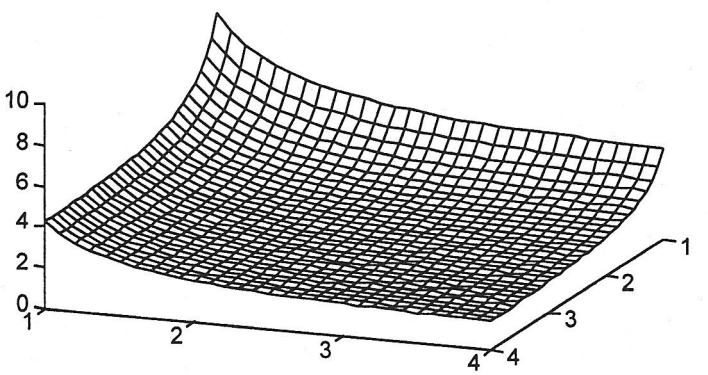
2. SFORMUŁOWANIE PROBLEMU

Należy wygenerować bazę reguł dla systemu opartego na logice rozmytej[2] aproksymującą funkcję:

$$z = (1 + x^{-2} + y^{-1.5})^2 \qquad (1)$$

gdzie: $x \in \langle 1;4 \rangle$, $y \in \langle 1;4 \rangle$.

Wartości lingwistyczne są ustalane w etapie wstępnym i pozostają bez zmian w trakcie optymalizacji bazy reguł.



Rys. 1 Wygląd aproksymowanej funkcji.

3. ROZWIĄZANIE PROBLEMU

Do znalezienia zbioru reguł użyto podejścia „Michigan approach"[3], charakteryzującego się tym, że każda reguła jest reprezentowana przez pojedynczy chromosom. Wykorzystano algorytm ewolucyjny ze zmienną wielkością populacji - EVaPS [1]. Każdy chromosom po utworzeniu ma obliczany czas życia wyrażony w generacjach.

Długość życia jest tym większa, im lepsze dopasowanie chromosomu (większa wartość funkcji dopasowania). Zwiększa to szansę lepszych chromosomów na częstszą rekombinację.

```
procedure EVaPS
begin
  t:=0;
  inicjacja Pop(t);
  ewaluacja Pop(t);
  ∀P ∈ Pop(0) : a(P) := 0
  while not kryterium stopu do
        begin
              Offs(t) := reprodukcja Pop(t);
              rekombinacja Offs(t);
              ewaluacja Offs(t);
              ∀C ∈ Offs(t) : a(C) := 0;
              Pop(t+1) := Pop(t) ∪ Offs(t)
              t:=t+1;
              ∀P ∈ Pop(t) : a(P) := a(P) + 1;
              Pop(t) := Pop(t) \ {P ∈ Pop(t): a(P) > l(P)};
        end;
end;

gdzie:
  a(C) - aktualny wiek chromosomu C
  l(C) - długość życia chromosomu C
```

Rys. 2: Struktura algorytmu EVaPS.

3.1. KODOWANIE

W chromosomie zastosowano kodowanie całkowitoliczbowe , na przykład reguła:

IF x is 1 AND y is 3 THEN z is 0

(gdzie x, y, z - zmienne lingwistyczne) ma postać [1 3 0].

3.2. OPERATORY GENETYCZNE

Zastosowano operator krzyżowania równomiernego polegający na wymianie odpowiadających sobie genów w chromosomach macierzystych. Prawdopodobieństwo wymiany jest stałe i równe p_{cross}.

P_1: [1 2 0 2]	C_1: [1 1 0 0]
P_2: [2 1 4 0]	C_2: [2 2 4 2]
P_1,P_2-chromosomy macierzyste;	
C_1,C_2-chromosomy potomne	

Rys. 3: **Przykład działania operatora krzyżowania**

Mutacja jest wykonywana z prawdopodobieństwem p_{mut} dla każdego genu niezależnie. Mutacja polega na zmianie wartości genu na losową wartość wygenerowaną ze zbioru wartości lingwistycznych dopuszczalnych dla mutowanego genu.

3.3. OCENA JAKOŚCI ZBIORU REGUŁ

Kryterium do oceny zbioru reguł jest błąd średniokwadratowy:

$$\varepsilon = \frac{1}{N}\sum_{i=1}^{N}(z_i - z_i')^2$$

N - wielkość zbioru trenującego (ilość przykładów);

z_i - wynik otrzymany po zastosowaniu zbioru reguł na i-tym przykładzie ze zbioru trenującego;

z'_i - wynik pożądany po zastosowaniu zbioru reguł na i-tym przykładzie ze zbioru trenującego; obliczane na podstawie wzoru (1).

Zadanie optymalizacji polega na dobraniu bazy reguł minimalizując wartość błędu średniokwadratowego.

3.4. OCENA PRZYSTOSOWANIA CHROMOSOMU

Każdy chromosom reprezentuje zakodowaną regułę. Zatem miarą jego jakości jest to, na ile polepsza jakość bazy reguł B. Dopasowanie chromosomu jest różnicą błędu bazy reguł przed i po wstawieniu do niego reguły w nim zakodowanej. Jeżeli polepsza ona w wystarczający sposób jakość bazy reguł, to jest do niego wstawiana. Rozpatrywane są dwa przypadki: dołożenie nowej reguły do bazy reguł albo zamiana z regułą najsłabszą. Oto szczegółowy algorytm postępowania podczas obliczania dopasowania dla chromosomu C z zakodowaną regułą R:

- jeżeli taka sama reguła już jest w bazie reguł, to przepisz jej dopasowanie do chromosomu C i zakończ wykonywanie procedury (w bazie nie może być dwóch takich samych reguł);
- ε_B - początkowy błąd zbioru reguł (znany z wcześniejszego wywołania procedury);
- policz ε_1 - błąd bazy reguł bez reguły najsłabszej - $B \setminus \{R_S\}$;
- policz ε_2 - błąd bazy reguł z wstawioną regułą R na miejsce najsłabszej - $B \cup \{R\} \setminus \{R_S\}$;
- policz ε_3 - błąd całej bazy reguł plus reguła - $B \cup \{R\}$;
- jeżeli $\varepsilon_1 - \varepsilon_2 > \Phi(R_S)$ ($\Phi(R_S)$ - wartość funkcji dopasowania dla R_s) i $\varepsilon_2 < \varepsilon_3$ to:
  1. wstaw R na miejsce R_s - $B' = B \cup \{R\} \setminus \{R_S\}$,
  2. policz wartości dopasowania dla wszystkich reguł w bazie i usuń te z mniejszym dopasowaniem niż zadane δ_{min_fit} - $B'' = \{R \in B' : \Phi(R) > \delta_{min\_fit}\}$,
  3. zakończ wykonywanie procedury;
- jeżeli $\varepsilon - \varepsilon_3 > \delta_{min\_fit}$ (δ_{min_fit} -zadane parametrycznie minimalne dopasowanie, z którym reguła zostaje włączona do bazy), to
  4. dodaj R do bazy reguł - $B' = B \cup \{R\}$,
  5. policz wartości dopasowania dla wszystkich reguł w bazie i usuń te z mniejszym dopasowaniem niż zadane δ_{min_fit} - $B'' = \{R \in B' : \Phi(R) > \delta_{min\_fit}\}$,
  6. zakończ wykonywanie procedury;

Przy każdorazowej modyfikacji bazy reguł wartości dopasowania reguł w bazie ulegają zmianie. Dlatego po każdej takiej modyfikacji niezbędne jest powtórne obliczenie wartości dopasowania dla wszystkich reguł. Jeżeli po przeliczeniu okazuje się, że dopasowanie jest mniejsze, bądź równe δ_{min_fit}, to reguła zostaje usunięta z bazy. Jeżeli ocena jakości reguły okaże się pozytywna (reguła wejdzie do bazy), to ilość wywołań procedury liczącej błąd zbioru reguł jest maksymalnie M+4 (liczy się błędy ε_B, ε_1, ε_2, ε_3 oraz dla każdej reguły z osobna - M to ilość reguł w bazie).

3.5. KRYTERIUM STOPU

Obliczenia trwają dopóki nie zostanie spełniony jeden z trzech warunków:

1. błąd średniokwadratowy zejdzie poniżej zadanego poziomu - $\varepsilon_B < \varepsilon_{gr}$,
2. numer generacji jest większy od zadanego - $t > t_{gr}$,
3. jakość bazy reguł nie poprawia się przez daną liczbę generacji, czyli $\varepsilon_B(t+\tau) = \varepsilon_B(t)$, $\tau > \tau_{gr}$.

W praktycznych badaniach symulacyjnych najczęściej decydujące było trzecie kryterium.

4. WYNIKI BADAŃ SYMULACYJNYCH

4.1. PARAMETRY ALGORYTMU EWOLUCYJNEGO

Pierwszym zadaniem było dobranie takich parametrów algorytmu EVaPS, aby obliczenia przebiegały jak najlepiej. Po kilkunastu próbach uznano, że poniższe wartości dają zadowalające wyniki:

- λ - wartość określająca wielkość populacji potomnej, a tym samym ogólnie decydująca o wielkości populacji; najlepsze rezultaty dla λ=10 lub λ=15;

- p_{mut}=0.4 - prawdopodobieństwo mutacji;
- p_{cross}=0.8 - prawdopodobieństwo krzyżowania chromosomu;
- p_{exch}=0.5 - prawdopodobieństwo wymiany genów w czasie krzyżowania;
- l_{min}=0 - minimalny czas życia chromosomu;
- l_{max}=10 - maksymalny czas życia chromosomu ;
- zastosowano biliniową strategię obliczania czasu życia dla chromosomów[1];
- badano zarówno globalne, jak i lokalne strategie obliczania czasu życia, tzn. brano pod uwagę wszystkie generacje albo tylko generację ostatnią (uzyskano dobre wyniki dla obydwu ustawień);

4.2. WARTOŚCI LINGWISTYCZNE

Wartości lingwistyczne systemu rozmytego zostały dobrane a priori tak, aby równomiernie pokrywały dziedzinę zmiennych wejściowych <1;4> oraz zmiennej wyjściowej <1;9>. Przyjęto, że będą miały kształt trójkątów. Na wejściach było ich po 4 , a na wyjściu 9. Zatem ilość wszystkich możliwych do utworzenia reguł jest 4*4*9=144. Początkowe próby z inną ilością pokazały, że:

- gdy było ich za mało albo miały kształt trapezów - nie dało się uzyskać zadowalającego spadku wartości błędu dla zbioru reguł;
- gdy było ich więcej - wydłużał się czas obliczeń, błąd dla zbioru trenującego malał nawet do wartości 0.01, ale za to dla zbioru testującego okazał się bardzo duży (około 0.35).

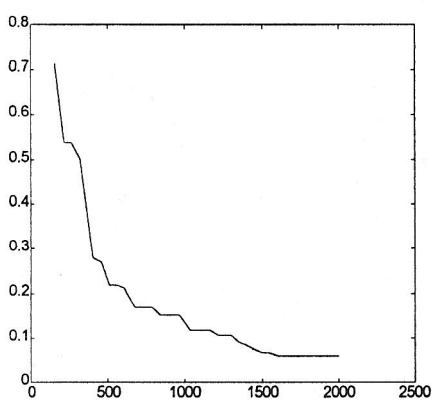
W drugim przypadku wystąpiło zjawisko przeuczenia. Powstało dużo reguł (kilkadziesiąt), które nauczyły się dobrze aproksymować dane ze zbioru trenującego, ale nie posiadały zdolności dobrej generalizacji dla przykładów testujących.

4.3. ZBIÓR TRENUJĄCY I TESTUJĄCY

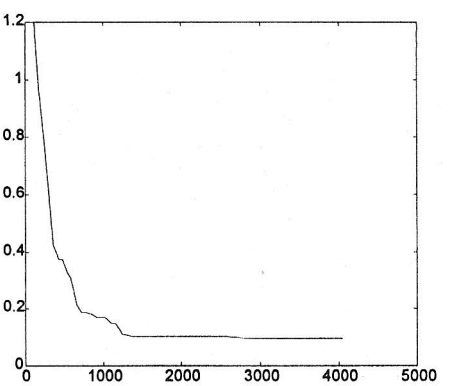
Zbiór trenujący stanowiło 100 przykładów wygenerowanych dla punktów pokrywających równomierną siatką całą dziedzinę funkcji. Zbiór testujący to 500 przykładów dla wygenerowanych losowo punktów.

4.4. BŁĄD I WIELKOŚĆ ZBIORU REGUŁ W FUNKCJI ILOŚCI WYWOŁAŃ PROCEDURY OBLICZENIOWEJ

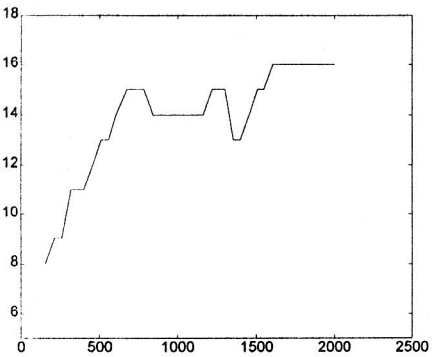
Symulacje prowadzono dla dwóch wartości parametru minimalnego dopasowania - dla δ_{min_fit}=0 i δ_{min_fit}=0.005. Wartość zerowa tego parametru dopuszcza do bazy reguł wszystkie te, które zmniejszają jego błąd. Powoduje to często, że baza reguł urasta do wielkości około 30, z czego wiele z nich ma minimalny przyczynek (rzędu tysięcznych) do zmniejszenia jej błędu. Nie jest to zjawisko korzystne, gdyż duża baza reguł zwalnia w sposób znaczący obliczanie odpowiedzi reguł na dane wejściowe i ilość wywołań funkcji obliczającej błąd bazy reguł. Ustawienie wartości parametru δ_{min_fit}=0.005 spowodowało zmniejszenie wielkości bazy reguł (nigdy nie przekroczyła 20 reguł), przy nieznacznym zwiększeniu błędu (około 0,06).



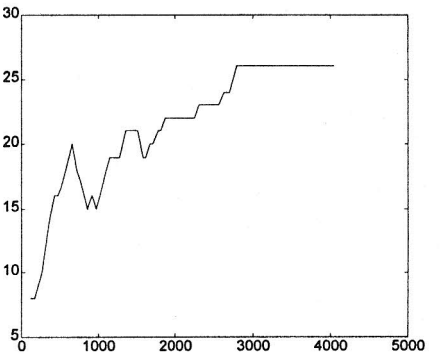
Rys. 4: Błąd w zależności od ilości wywołań funkcji obliczeniowej dla δ_{min_fit}=0.005.



Rys. 5: Błąd w zależności od ilości wywołań funkcji obliczeniowej δ_{min_fit}=0.



Rys. 6: Wielkość zbioru reguł . w zależności od ilości wywołań funkcji obliczeniowej dla δ_{min_fit}=0.005



Rys. 7: Wielkość zbioru reguł w zależności od ilości wywołań funkcji obliczeniowej dla δ_{min_fit}=0

4.5. PRZYKŁADOWE WYNIKI SYMULACJI

Wykonano szereg symulacji komputerowych. Dla wszystkich przyjęto wartość parametru δ_{min_fit} równą 0.005 (z powodów opisanych w poprzednim punkcie).

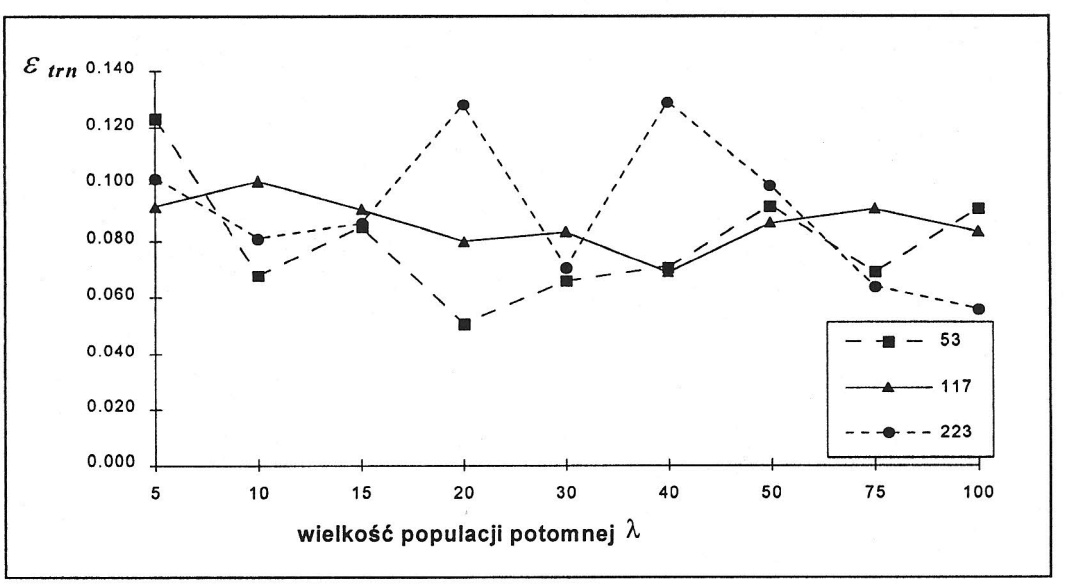
Okazało się, że duży wpływ na wyniki działania algorytmu ma początkowa faza obliczeń, a w szczególności inicjalna zawartość bazy reguł. Dlatego obliczenia prowadzono dla trzech różnych wartości ustawiających generator liczb losowych $\Omega_{seed} \in \{53, 117, 223\}$.

Obliczenia były prowadzone w zależności od wielkości populacji potomnej λ oraz w zależności od maksymalnego czasu życia chromosomów l_{max}. Następnie zbadano zależność błędu zbioru testującego ε_{tst} i trenującego ε_{trn}.

Zastosowanym kryterium stopu był brak poprawy zbioru reguł przez co najmniej 35 generacji (dla mniejszych wielkości populacji np. dla parametrów l_{max}=5 lub λ<=10 ilość ta wynosiła 50).

4.5.1. JAKOŚĆ BAZY REGUŁ W ZALEŻNOŚCI OD WIELKOŚCI POPULACJI POTOMNEJ λ

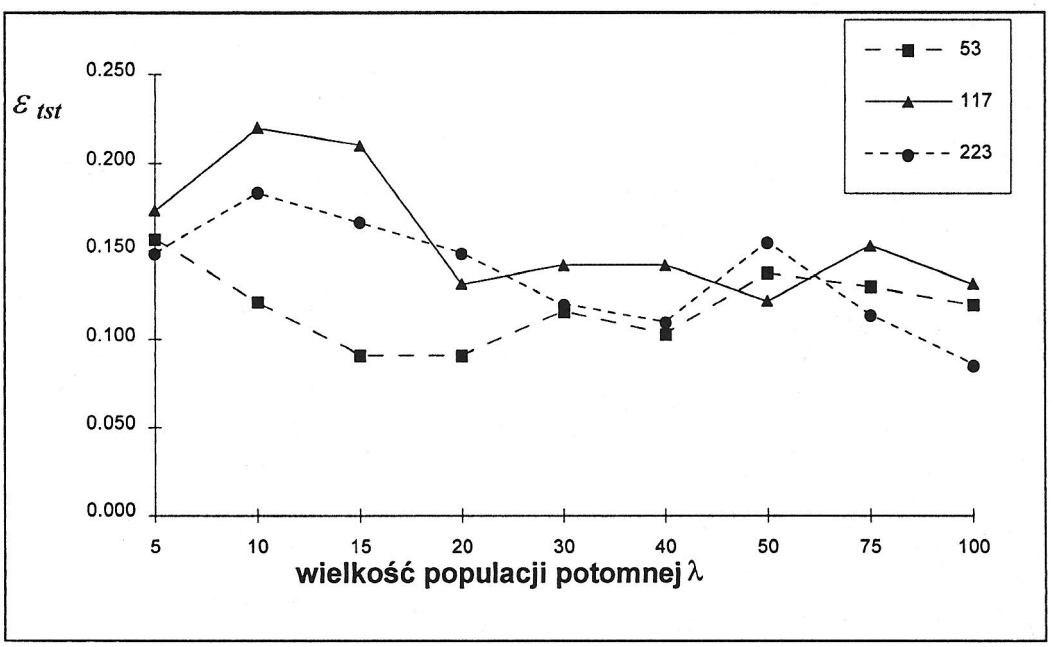
Badano zależność generowanej bazy reguł i wielkości populacji potomnej λ. Pozostałe parametry algorytmu ewolucyjnego EVaPS były stałe i miały wartości jak w punkcie 4.1. Brano pod uwagę błąd zbioru trenującego ε_{trn} i testującego ε_{tst}, ilość wywołań funkcji obliczającej błąd bazy reguł M, wielkość bazy reguł ω i wielkość ostatniej populacji Π.



Z danych widać, że wartości błędów są bliskie sobie dla wszystkich λ. Błąd zbioru testującego mieści się zwykle w przedziale od 0.09 do 0.2. Im większa wartość λ, tym błąd jest mniej zróżnicowany dla poszczególnych Ω_{seed}.

Ω_{seed}	badane wielkości	Wielkości populacji potomnej λ								
		5	10	15	20	30	40	50	75	100
53	ε_{trn}	0.123	0.068	0.085	0.051	0.066	0.071	0.092	0.069	0.091
	ε_{tst}	0.157	0.121	0.091	0.091	0.116	0.103	0.137	0.130	0.120
	M	2240	2494	1088	2942	5186	2253	2202	2057	4267
	ω	14	15	15	16	15	12	18	16	16
	Π	18	38	56	73	113	150	183	279	366
117	ε_{trn}	0.092	0.101	0.09	0.080	0.083	0.069	0.086	0.091	0.083
	ε_{tst}	0.173	0.220	0.210	0.131	0.142	0.142	0.122	0.153	0.131
	M	398	730	720	2337	1421	1656	1841	1048	3762
	ω	12	13	14	13	14	16	14	12	11
	Π	21	37	52	73	113	146	187	283	365
223	ε_{trn}	0.102	0.081	0.086	0.128	0.071	0.129	0.099	0.064	0.056
	ε_{tst}	0.149	0.183	0.166	0.149	0.120	0.110	0.155	0.114	0.086
	M	1286	2604	5172	2310	2360	1720	1551	1624	5571
	ω	15	14	15	13	14	15	16	16	15
	Π	19	39	54	75	112	150	182	283	370

Tabela 1. Wyniki symulacji przy różnej wielkości populacji potomnej λ.

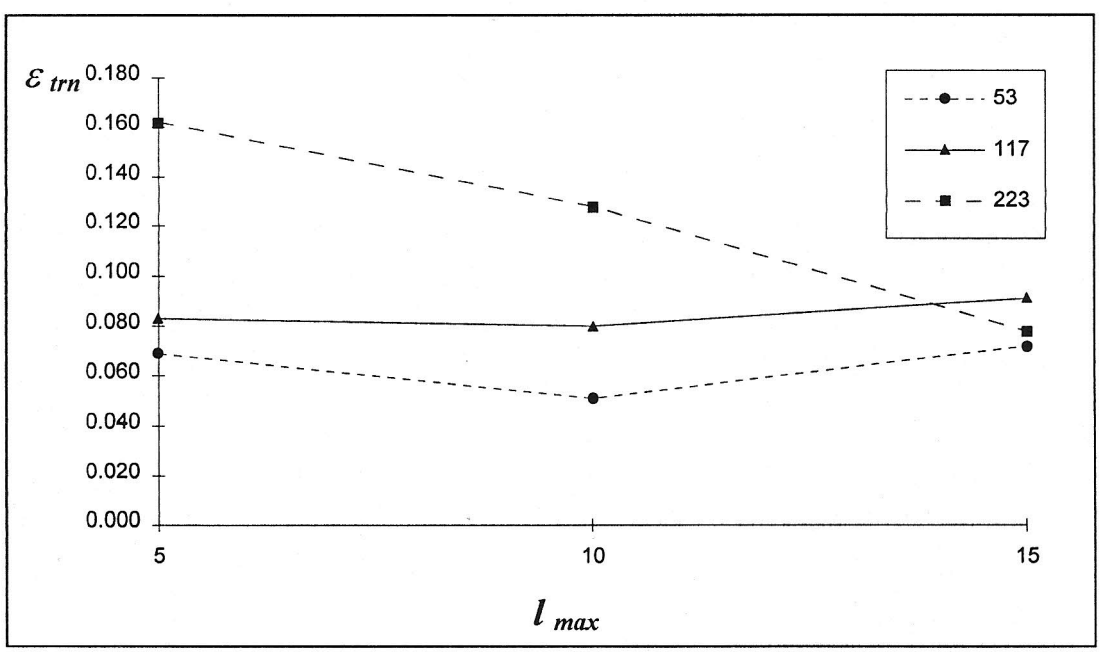


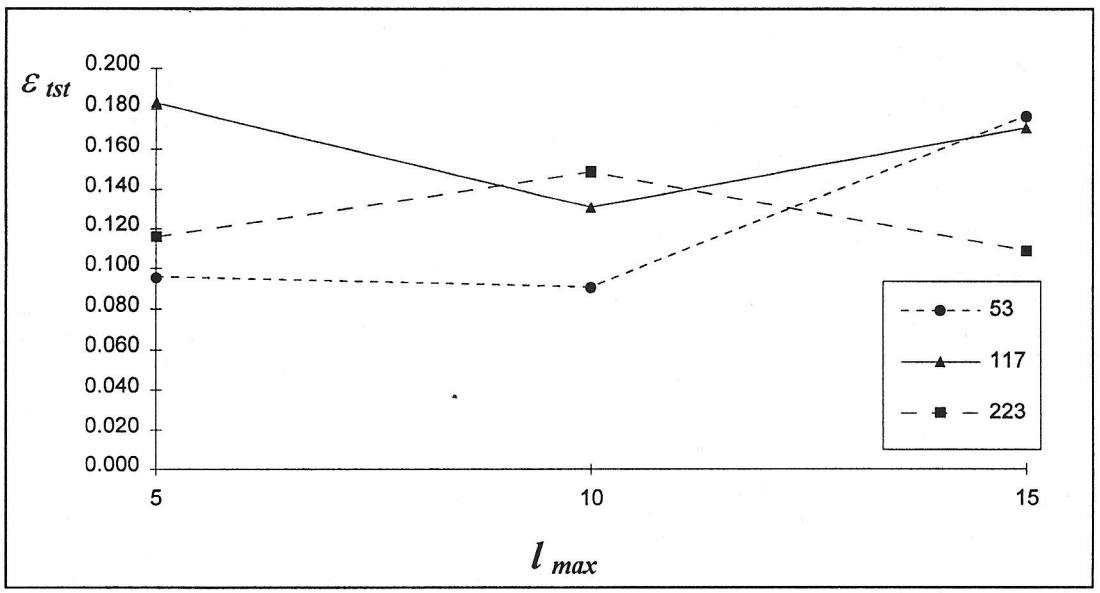
4.5.2. JAKOŚĆ BAZY REGUŁ W ZALEŻNOŚCI OD MAKSYMALNEGO CZASU ŻYCIA CHROMOSOMÓW l_{max}

Badania prowadzono dla trzech wartości parametru $l_{max} \in \{5, 10, 15\}$. Jak w poprzednim punkcie brano pod uwagę błąd zbioru trenującego ε_{trn} i testującego ε_{tst}, ilość wywołań funkcji obliczającej błąd zbioru reguł M, wielkość bazy reguł ω i wielkość ostatniej populacji Π.

Ω_{seed}	badane wielkości	Maksymalna długość życia chromosomu l_{max}		
		5	10	15
53	ε_{trn}	0.069	0.051	0.072
	ε_{tst}	0.096	0.091	0.177
	M	2159	2942	3312
	ω	17	16	19
	Π	38	73	132
117	ε_{trn}	0.083	0.080	0.091
	ε_{tst}	0.183	0.131	0.171
	M	3526	2337	1403
	ω	14	13	11
	Π	35	73	135
223	ε_{trn}	0.162	0.128	0.078
	ε_{tst}	0.116	0.149	0.109
	M	2103	2310	2688
	ω	13	13	15
	Π	36	75	130

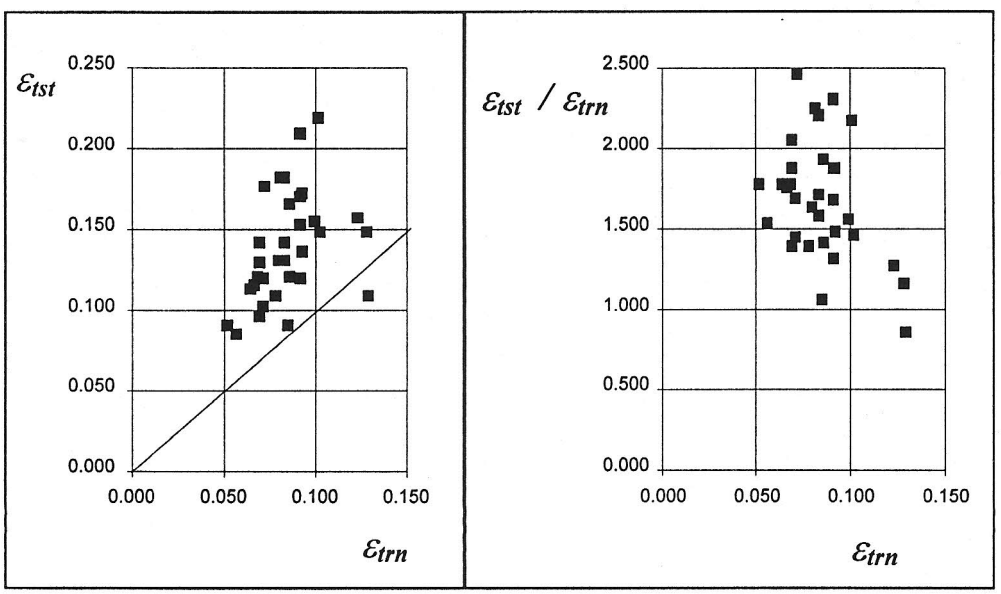
Tabela 2. Wyniki symulacji przy różnej wartości maksymalnego czasu życia chromosomów l_{max}.





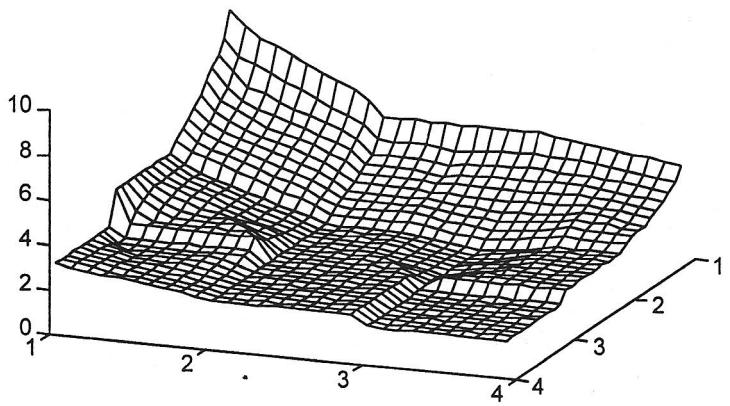
4.5.3. BŁĄD ZBIORU TESTUJĄCEGO I TRENUJĄCEGO

Poniższe wykresy pokazują zależności między ε_{trn} i ε_{tst} dla danych z Tabela 1 i Tabela 2



4.6. APROKSYMACJA

Przy błędzie zbioru reguł od 0.05 do 0.1 na zbiorze trenującym, błąd zbioru testującego był od 30% do 100% większy, chociaż w dwóch przypadkach był on mniejszy (patrz tabele).



Rys. 8: Wygląd funkcji aproksymującej dla błędu zbioru testowego 0.09

5. PODSUMOWANIE

W artykule niniejszym opisano sposób generacji bazy reguł do systemów rozmytych przy wykorzystaniu algorytmu ewolucyjnego ze zmienną wielkością populacji. Zostały przedstawione wyniki badań symulacyjnych. Badania te polegały na generacji bazy reguł aproksymującej możliwie dokładnie funkcję nieliniową dwóch zmiennych. Baza reguł powinna mieć jak najmniejszy błąd i jej wielkość powinna być jak najmniejsza. Poprzez ustawienie minimalnego progu δ_{min_fit} uzyskano prawie dwukrotne zmniejszenie ilości reguł kosztem nieznacznego zwiększenia błędu bazy reguł. Wartość funkcji dopasowania chromosomu musi przewyższać ten próg, aby zakodowana w nim reguła weszła do bazy. Dodatkowym pozytywnym efektem wprowadzenia progu było skrócenie czasu obliczeń.

Metodą kilkudziesięciu prób ustalono wartości parametrów algorytmu ewolucyjnego, dla których działa on w sposób możliwie optymalny (patrz 4.1). Przeprowadzono dwie serie symulacji: dla różnej wielkości populacji potomnej oraz dla różnych wartości maksymalnego czasu życia chromosomów. Okazało się, że ważnym czynnikiem decydującym o jakości generowanej bazy reguł jest początkowy etap działania algorytmu, szczególnie dla małych wartości λ i l_{max}.

Algorytm jest zbieżny dla szerokiego zakresu parametrów λ i l_{max}. O jego skuteczności w dużej mierze, oprócz algorytmu ewolucyjnego, decyduje sposób obliczania wartości funkcji dopasowania reguł oraz kryteria wstawiania ich do bazy.

6. LITERATURA

[1] Arabas J. „Algorytmy ewolucyjne ze zmienną licznością populacji i zmiennym zasięgiem krzyżowania" rozprawa doktorska, *Politechnika Warszawska*, Warszawa 1995.

[2] Bellman, Zadeh „Decision-making in fuzzy environment", *Management Science 17*, 1970.

[3] De Jong K. „An Introduction To Evolutionary Computation and Its Applications in Fuzzy Logic", Theorie und Praxis, edit B.Reusch, *Springer Verlag*, Berlin Heidelberg 1994.