

**Konstrukcja automatu deterministycznego skończonego sprawdzającego
zachodzenie relacji indukowanej przez język dla słów z danego języka
(dokumentacja uzupełniająca)
Teoria algorytmów i obliczeń**

Anna Zawadzka, Sylwia Nowak, Pavel Kuzmich, and Piotr Waszkiewicz

Spis treści

Lecture Notes in Computer Science	1
<i>Authors' Instructions</i>	
1 Wstęp	2
1.1 Cel projektu	2
1.2 Sposób realizacji w etapach	2
2 Szczegółowy opis sposobu realizacji zadania	3
2.1 Automat wejściowy	3
2.2 Zbiór treningowy i zbiór testowy	3
2.3 Opis przestrzeni poszukiwania dla algorytmu PSO	3
2.4 Odległość w przeszukiwanej przestrzeni	4
2.5 Przeszukiwanie przestrzeni w algorytmie PSO	4
3 Wyniki przeprowadzonych testów	6
3.1 Skuteczność aproksymacji automatu o 4 stanach	8
3.2 Skuteczność aproksymacji automatu o 6 stanach	8
3.3 Skuteczność aproksymacji automatu o 10 stanach	8
3.4 Skuteczność aproksymacji automatu o 15 stanach	8
3.5 Skuteczność aproksymacji automatu o 20 stanach	8
3.6 Skuteczność aproksymacji automatu o 30 stanach	10
3.7 Skuteczność aproksymacji automatu o 50 stanach	12
3.8 Skuteczność aproksymacji automatu o 80 stanach	14

1 Wstęp

1.1 Cel projektu

Celem projektu jest stworzenie deterministycznego automatu skończonego realizującego funkcję relacji indukowanej przez język na podstawie odpowiedzi udzielanych przez gotowe, nieznane narzędzie pełniące tę samą rolę.

1.2 Sposób realizacji w etapach

- Wprowadzenie narzędzia (automatu), pozwalającego sprawdzić czy $\forall_{x,y \in \Sigma^*} xR_L y$.
- Utworzenie treningowego i testowego zbioru słów.
- Znalezienie skończonych automatów deterministycznych spełniających cel projektu z wykorzystaniem algorytmu PSO (Particle Swarm Optimization) i zbioru treningowego słów.
- Wybranie najlepszego utworzonego automatu wykorzystując zbiór testowy słów.

2 Szczegółowy opis sposobu realizacji zadania

2.1 Automat wejściowy

Na podstawie automatu wejściowego zostanie stworzone narzędzie sprawdzające, czy dwa słowa są ze sobą w relacji. Informacje o automacie wejściowym będą przechowywane w pliku tekstowym podawanym na wejście programu. Format reprezentacji automatu jest następujący:

- Stany numerowane liczbami $1, 2, \dots, n$, gdzie n to liczba stanów, 1 – stan początkowy.
- r – liczba symboli w alfabecie.
- Tabela funkcji przejścia.

Po wprowadzeniu pliku z opisem automatu będzie można zobaczyć jego graficzną reprezentację w postaci grafu.

2.2 Zbiór treningowy i zbiór testowy

Utworzenie treningowego i testowego zbioru słów nastąpi poprzez wygenerowanie odpowiednich zbiorów. Będą one dostępne (nie zmienia się) dopóki zbiór symboli alfabetu pozostanie ten sam.

Zbiór treningowy zawierać będzie wszystkie słowa nad alfabetem o długości mniejszej bądź równej długości podanej przez użytkownika (nie większej niż 5).

Zbiór testowy składać się będzie ze słów będących wariacjami bez powtórzeń dla słów o długości nie mniejszej niż 7 i nie większej niż maksymalna ilość stanów poszukiwanego automatu. W przypadku mniejszej ilości symboli alfabetu niż wymagałyby tego wariacje następuje konstrukcja nowego, tymczasowego zbioru poprzez powielenie wszystkich symboli alfabetu i dodanie ich do istniejących już symboli. Nowe symbole będą dodatkowo indeksowane numerem oznaczających w której iteracji zostały dodane. Zapewni to ich rozróżnialność podczas konstrukcji zbiorów.

Przykład zbioru: 0, 1. Zbiór po przygotowaniu do utworzenia wariacji bez powtórzeń dla słów długości 6: 0, 1, 0₁, 1₁, 0₂, 1₂.

Dodatkowo do zbioru treningowego zostały dodane losowe słowa (zawierające losowe litery alfabetu) o losowej długości z przedziału $[x_1, x_2]$, gdzie x_1 jest maksymalną długością słowa treningowego generowanego poprzednią metodą (przy użyciu wariacji z powtórzeniami) a x_2 jest nowym parametrem oznaczającym maksymalną długość losowego słowa. W przypadku gdy $x_2 < x_1$ nie generowane są żadne dodatkowe słowa. Użytkownik może również określić maksymalną ilość słów losowych. Dodatkowo dla każdego nowego słowa s_1 dopisanego do zbioru treningowego tworzone jest słowo s_2 tej samej długości lecz również o losowej zawartości, które jest dopisywane do zbioru słów testowych.

Zbiór treningowy i testowy są zbiorami rozłącznymi.

2.3 Opis przestrzeni poszukiwania dla algorytmu PSO

Przestrzeń poszukiwania w algorytmie PSO będą stanowić wszystkie deterministyczne automaty skończone o określonej liczbie stanów, z alfabetem wejściowym równoważnym z alfabetem języka L . Zapis cyfrowy każdego automatu można przedstawić jako zbiór macierzy binarnych o wymiarach $|Q|^2$ (gdzie $|Q|$ to ilość stanów automatu). Zbiór ten jest równoliczny ze zbiorem symboli alfabetu automatu (każdemu symbolowi odpowiadać będzie inna macierz). Jeżeli jako dodatkowe założenie przyjmujemy, że w każdym wierszu każdej macierzy znajdować może się tylko jedna jedynka, to wtedy taką macierz traktować możemy jako część tabelki funkcji przejścia automatu, która dla wiersza o numerze i oraz jedynce w kolumnie j odpowiada przejściu ze stanu q_i do stanu q_j po wczytaniu symbolu przypisanego do tej macierzy. Ponieważ następująca reprezentacja jest nieoptymalna pod względem zużycia pamięci zapis można uprościć do tablicy jednowymiarowej która dla każdego indeksu odpowiadającego numerowi stanu w którym automat znajduje się przed wczytaniem symbolu, przyporządkowuje numer stanu w którym automat znajdzie się po wczytaniu elementu z taśmy.

Dla przykładu, po zastosowaniu takiej optymalizacji dla tabeli funkcji przejścia

symbol	q_1	q_2	q_3
q_1	0	1	0
q_2	1	0	0
q_3	0	0	1

Tablica 1. Przykładowa tabela funkcji przejścia

otrzymamy jako wynik wektor $[1 \ 0 \ 2]$.

Przyjmijmy $d = |\Sigma|$ liczba symboli alfabetu, $f = |Q|$ liczba stanów automatu. Tak więc przestrzeń wszystkich automatów stanowiąc będą wszystkie możliwe ciągi liczb o długości $d * f$ i wartościach z przedziału $[0, f)$. Częstki poruszające się w przestrzeni będą przyjmowały pozycje będące reprezentacją tych automatów. Dla każdej cząsteczki jej prędkość definiowana będzie jako $V_p = (V_1, V_2, \dots, V_d)$ gdzie $V_{i(1 \leq i \leq d)}$ oznacza prędkość cząstki na płaszczyźnie odpowiadającej symbolowi o numerze i , definiowanej jako $V_{pi} = (V_1, V_2, \dots, V_f)$. $V_{j(1 \leq j \leq f)}$ reprezentuje tutaj prędkość cząstki dla wiersza numer j w macierzy zdefiniowanej wcześniej jako reprezentacja tabeli funkcji przejścia dla danego symbolu. Tak zdefiniowaną prędkość można (podobnie jak położenie) przedstawić w postaci tablicy jednowymiarowej o długości $d * f$ (gdzie d i f - zdefiniowane wcześniej). Zmiana położenia cząstki następuje więc poprzez dodanie wektora prędkości do wektora położenia tym samym otrzymując nowe położenie.

2.4 Odległość w przeszukiwanej przestrzeni

Dla tak zdefiniowanej przestrzeni i ciągów liczb długości $d*f$, które reprezentować będą automaty wprowadzimy metrykę, nazywaną metryką Hamminga. Dla dwóch ciągów a i b z tej przestrzeni odległość Hamminga jest równa liczbie jedynek w słowie $aXORb$, np. odległość pomiędzy ciągami 10011101 i 10111001 wynosi 2.

2.5 Przeszukiwanie przestrzeni w algorytmie PSO

Przeszukiwanie przestrzeni automatów realizowane jest poprzez wygenerowanie pewnej ilości cząstek (nazywanych rojem) poruszających się według określonych zasad i dążących do osiągnięcia postawionego celu. Za cel najczęściej przyjmuje się minimalizację wartości pewnej funkcji nazywanej funkcją celu. W tym projekcie będzie to funkcja $f : A \rightarrow N$ (A - zbiór automatów, N - liczby naturalne), która dla danego automatu zwraca liczbę błędnie zaklasyfikowanych słów.

W trakcie kolejnych kroków zdyskretyzowanego czasu cząstki przemieszczają się do nowych pozycji, symulując adaptację roju do środowiska poszukując optimum. Każda cząstka pamięta swoje dotychczasowe najlepsze położenie (miejsce w przestrzeni gdzie znaleziony automat miał najniższą wartość funkcji celu). "Liderem" roju zostaje ta o najlepszym położeniu z całego roju. Położenia cząstek w obszarze przeszukiwania stanowią potencjalne rozwiązania.

Algorytm PSO funkcjonuje według następującego schematu:

1. Nadanie cząstkom roju losowych położeń
2. Dokonanie oceny położenia cząsteczek za pomocą funkcji dopasowania (fitness)
3. Zmiana zapisu w pamięci cząstek dotycząca najlepszych własnych położeń
4. Wyłonienie lidera roju
5. Aktualizacja wektora prędkości każdej cząstki

$$v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[]) \quad (1)$$

6. Aktualizacja położenia każdej cząstki - jeżeli położenie nie uległo zmianie przez przynajmniej trzy iteracje losowane zostaje nowe położenie

$$present[] = present[] + v[] \quad (2)$$

gdzie

$present$ - wektor położenia i -tej cząstki,

v - odpowiedni wektor prędkości danej cząstki,

$pbest$ - najlepsze dotąd znalezione położenie i -tej cząstki,

$gbest$ - najlepsze dotąd znalezione położenie lidera roju,

$c1, c2$ - współczynniki wagowe, określane na poziomie i -tej cząstki.

Współczynniki $c1, c2$ są ustalonymi mnożnikami wagowymi. Współczynniki są liczbami losowymi o rozkładzie równomiernym w przedziale $[0, 1]$.

Jeżeli w trakcie działania algorytmu PSO któraś z cząstek nie będzie zyskiwać lepszych położeń przez więcej niż trzy iteracje zostaje jej przydzielone nowe, zupełnie losowe położenie. Dzięki temu zyskuje się lepszą zbieżność do najlepszego automatu, oraz unika ryzyka uwięzienia w lokalnym minimum. Cząstki które w danym ruchu miałyby wyjść poza obszar poszukiwań zamiast być stopowane zostają umieszczone na początku tego obszaru z przeciwległej strony.

Punkty 2-6 należy wykonywać w pętli dopóki oba warunki będą spełnione:

- któraś z cząstek nie osiągnie założonej wartości minimalnej błędu,
- nie zostanie osiągnięta z góry wyznaczona ilość iteracji ruchu.

Jeden przebieg pętli odpowiada krokowi czasowemu dla poruszającego się roju. Wartość kroku czasowego przyjmuje się jako jednostkową. Cząstki poruszają się ruchem odcinkowo prostoliniowym. Postać równań (1) i (2) wynika z drugiej zasady dynamiki Newtona, gdzie wypadkowa siła wywołująca przyspieszenie każdej cząstki powstaje wskutek działania sił od “naciągniętych sprężyn” pomiędzy położeniami aktualnym a najlepszymi położeniami: własnym, lidera sąsiada oraz lidera roju. Kolejny wektor prędkości czasu jest wynikiem jej przyspieszenia w kierunku nowego, potencjalnie lepszego położenia, bazując na stale zmieniających się najlepszych wcześniej znanych położeniach.

Dla każdej wartości $i = 1, 2, 3, \dots$ algorytm PSO uruchamiany będzie w przestrzeni z automatami o i stanach. Warto zwrócić uwagę, że istnieje możliwość znalezienia kilku automatów spełniających warunki zakończenia algorytmu PSO. Aby zminimalizować ryzyko wybrania niewłaściwego automatu, po zakończeniu obliczeń dla każdej wartości parametru i najlepszy automat dla danej ilości stanów będzie testowany na zbiorze testowym. Ostateczna decyzja wyboru najlepszego rozwiązania podjęta będzie na podstawie uzyskanej wartości funkcji celu na tym zbiorze.

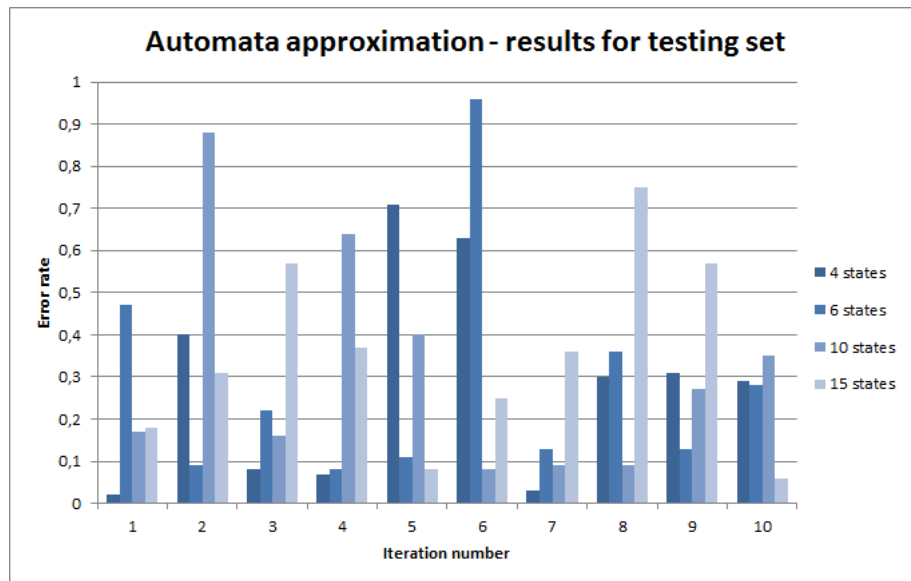
3 Wyniki przeprowadzonych testów

Podczas sprawdzania skuteczności zaproponowanego rozwiązania wykonano szereg następujących testów.

Dla wszystkich symulacji przyjęto alfabet pięcioletni. Wykonano po 10 iteracji przeszukiwania przestrzeni dla każdej klasy automatów o 4, 6, 10, 15 stanach z losową funkcją przejścia. Dla algorytmu PSO rozważane były przestrzenie automatów z ilością stanów z zakresu 3 - 15. Zbiór treningowy oraz testowy konstruowane były zgodnie z zasadami przedstawionymi we wcześniejszych sekcjach tego dokumentu.

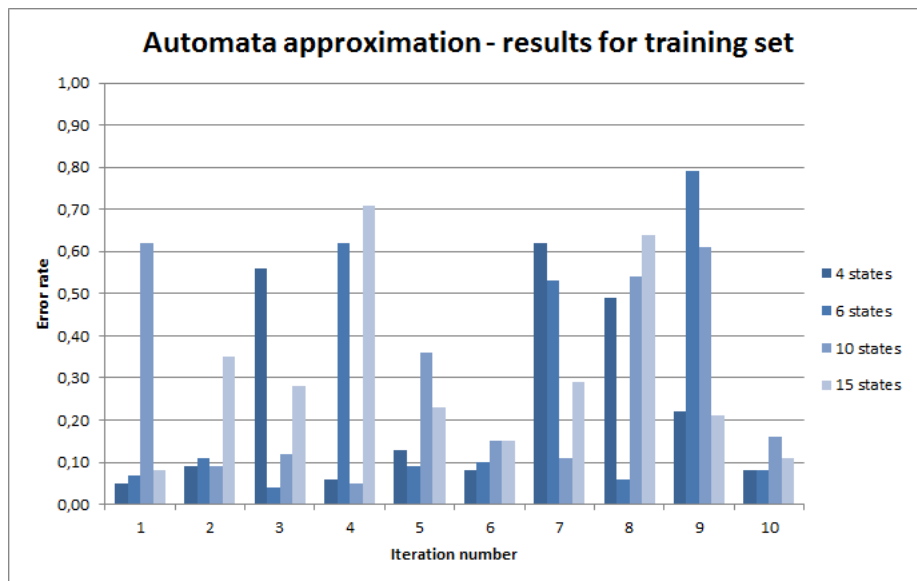
Tablica 2. Automata approximation - results for testing set

	4 states	6 states	10 states	15 states
1	0.02	0.47	0.17	0.18
2	0.4	0.09	0.88	0.31
3	0.08	0.22	0.16	0.57
4	0.07	0.08	0.64	0.37
5	0.71	0.11	0.4	0.08
6	0.63	0.96	0.08	0.25
7	0.03	0.13	0.09	0.36
8	0.3	0.36	0.09	0.75
9	0.31	0.13	0.27	0.57
10	0.29	0.28	0.35	0.06
Average	0.284	0.283	0.313	0.35



Tablica 3. Automata approximation - results for training set

	4 states	6 states	10 states	15 states
1	0,05	0,07	0,62	0,08
2	0,09	0,11	0,09	0,35
3	0,56	0,04	0,12	0,28
4	0,06	0,62	0,05	0,71
5	0,13	0,09	0,36	0,23
6	0,08	0,10	0,15	0,15
7	0,62	0,53	0,11	0,29
8	0,49	0,06	0,54	0,64
9	0,22	0,79	0,61	0,21
10	0,08	0,08	0,16	0,11
Average	0,24	0,25	0,28	0,31



3.1 Skuteczność aproksymacji automatu o 4 stanach

3.2 Skuteczność aproksymacji automatu o 6 stanach

3.3 Skuteczność aproksymacji automatu o 10 stanach

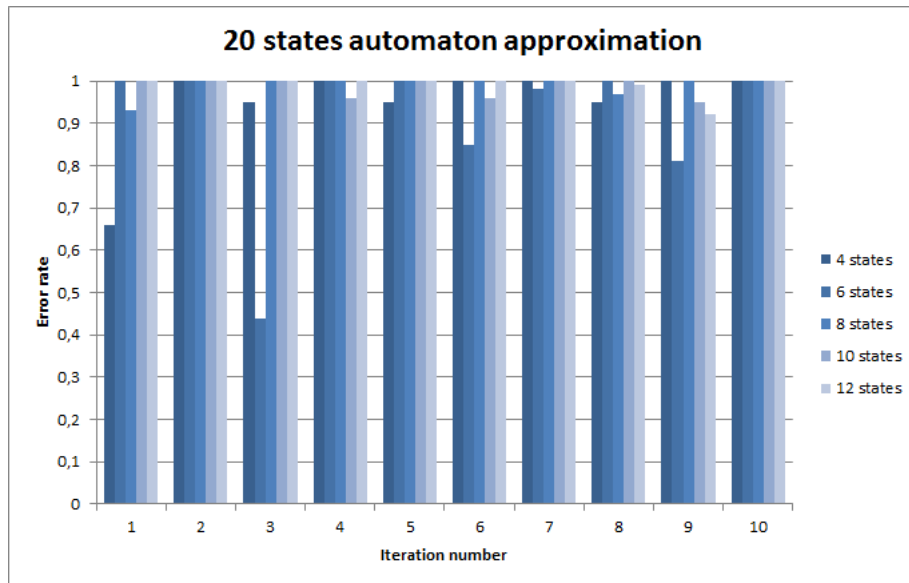
3.4 Skuteczność aproksymacji automatu o 15 stanach

Dodatkowo przeprowadzone zostały próby aproksymacji automatów o 20, 30, 50, 80 stanach przez inne, o mniejszej liczbie stanów - 4, 6, 8, 10, 12.

3.5 Skuteczność aproksymacji automatu o 20 stanach

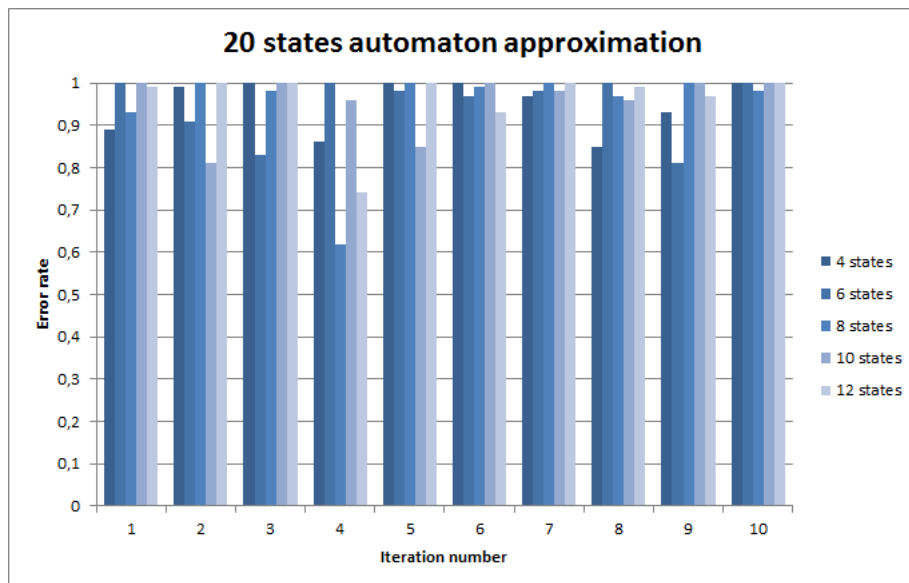
Tablica 4. 20 states automaton approximation testing set

	4 states	6 states	8 states	10 states	12 states
1	0,66	1	0,93	1	1
2	1	1	1	1	1
3	0,95	0,44	1	1	1
4	1	1	1	0,96	1
5	0,95	1	1	1	1
6	1	0,85	1	0,96	1
7	1	0,98	1	1	1
8	0,95	1	0,97	1	0,99
9	1	0,81	1	0,95	0,92
10	1	1	1	1	1
Average	0,951	0,908	0,99	0,987	0,991



Tablica 5. 20 states automaton approximation training set

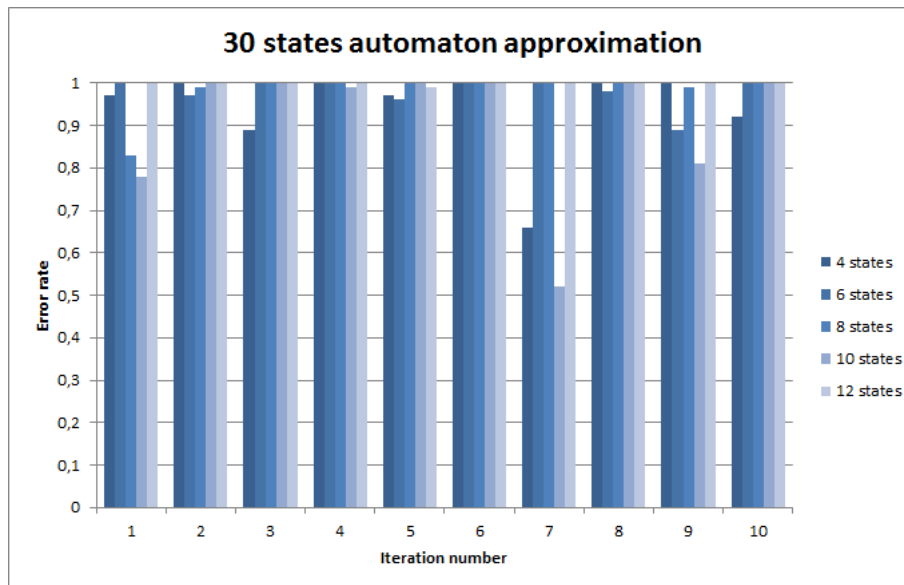
	4 states	6 states	8 states	10 states	12 states
1	0,89	1	0,93	1	0,99
2	0,99	0,91	1	0,81	1
3	1	0,83	0,98	1	1
4	0,86	1	0,62	0,96	0,74
5	1	0,98	1	0,85	1
6	1	0,97	0,99	1	0,93
7	0,97	0,98	1	0,98	1
8	0,85	1	0,97	0,96	0,99
9	0,93	0,81	1	1	0,97
10	1	1	0,98	1	1
Average	0,949	0,948	0,947	0,956	0,962



3.6 Skuteczność aproksymacji automatu o 30 stanach

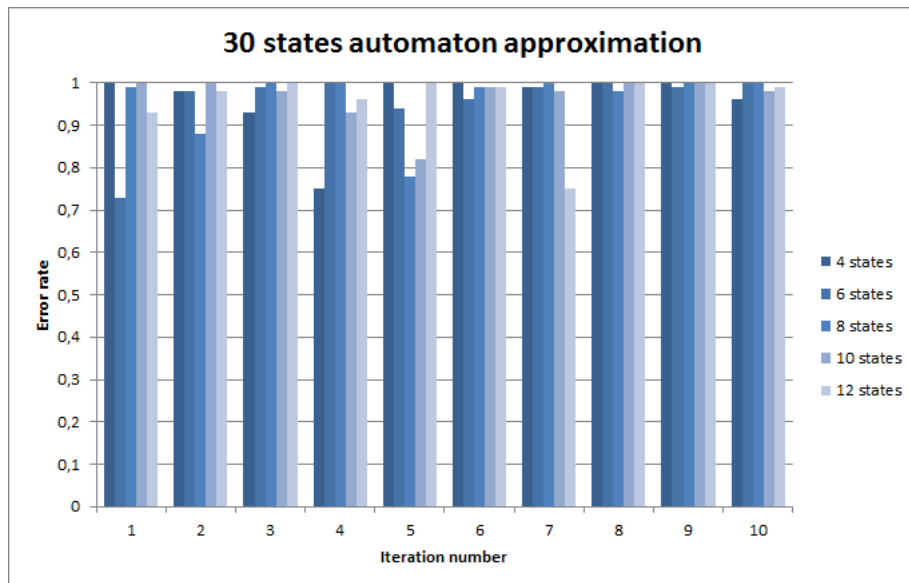
Tablica 6. 30 states automaton approximation testing set

	4 states	6 states	8 states	10 states	12 states
1	0,97	1	0,83	0,78	1
2	1	0,97	0,99	1	1
3	0,89	1	1	1	1
4	1	1	1	0,99	1
5	0,97	0,96	1	1	0,99
6	1	1	1	1	1
7	0,66	1	1	0,52	1
8	1	0,98	1	1	1
9	1	0,89	0,99	0,81	1
10	0,92	1	1	1	1
Average	0,941	0,98	0,981	0,91	0,999



Tablica 7. 30 states automaton approximation training set

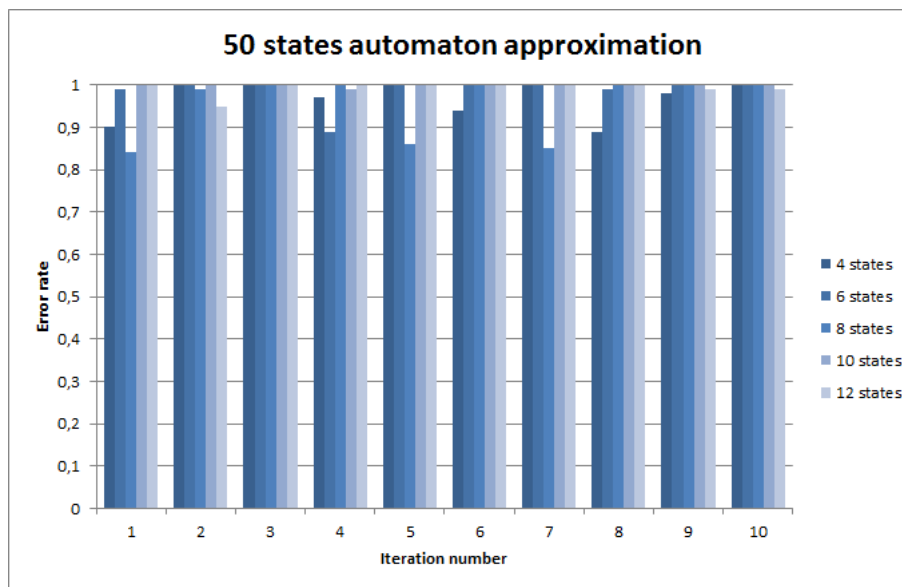
	4 states	6 states	8 states	10 states	12 states
1	1	0,73	0,99	1	0,93
2	0,98	0,98	0,88	1	0,98
3	0,93	0,99	1	0,98	1
4	0,75	1	1	0,93	0,96
5	1	0,94	0,78	0,82	1
6	1	0,96	0,99	0,99	0,99
7	0,99	0,99	1	0,98	0,75
8	1	1	0,98	1	1
9	1	0,99	1	1	1
10	0,96	1	1	0,98	0,99
Average	0,961	0,958	0,962	0,968	0,96



3.7 Skuteczność aproksymacji automatu o 50 stanach

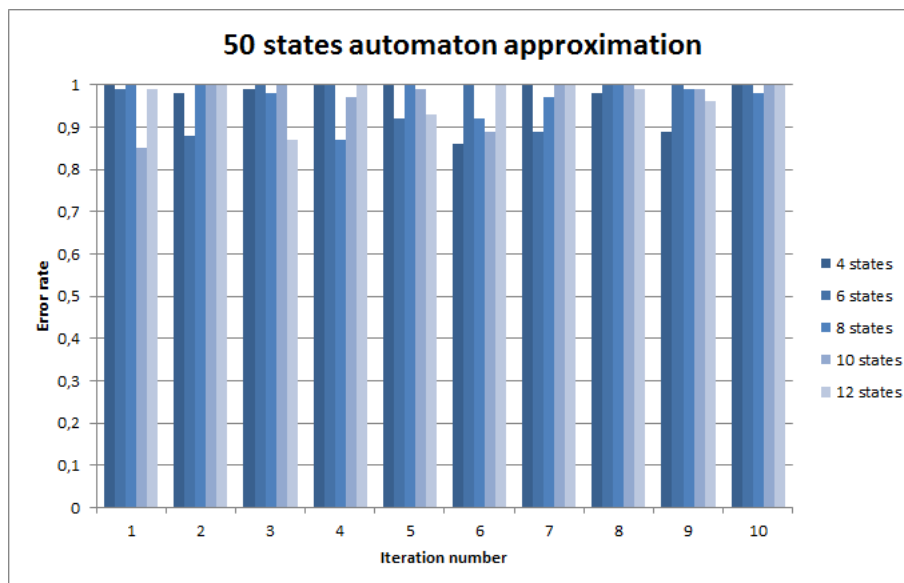
Tablica 8. 50 states automaton approximation testing set

	4 states	6 states	8 states	10 states	12 states
1	0,9	0,99	0,84	1	1
2	1	1	0,99	1	0,95
3	1	1	1	1	1
4	0,97	0,89	1	0,99	1
5	1	1	0,86	1	1
6	0,94	1	1	1	1
7	1	1	0,85	1	1
8	0,89	0,99	1	1	1
9	0,98	1	1	1	0,99
10	1	1	1	1	0,99
Average	0,968	0,987	0,954	0,999	0,993



Tablica 9. 50 states automaton approximation training set

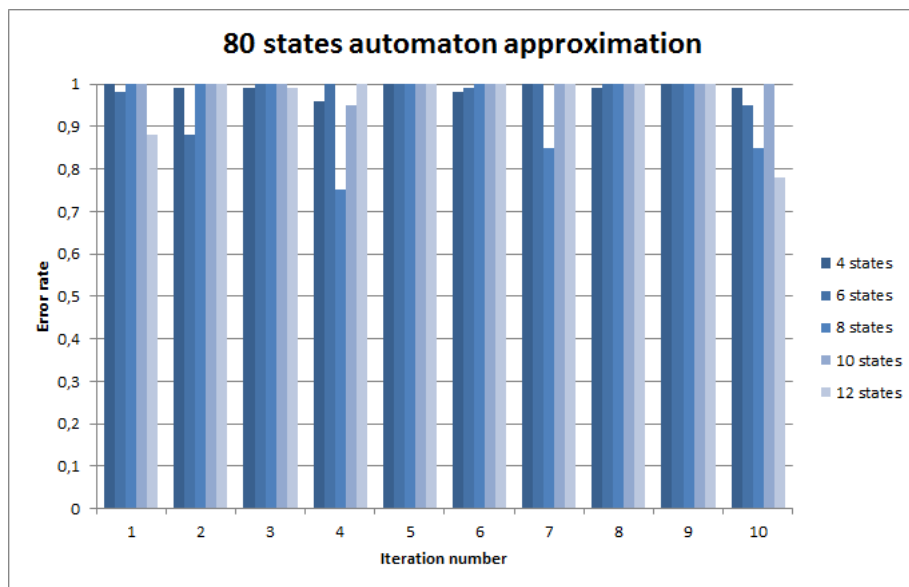
	4 states	6 states	8 states	10 states	12 states
1	1	0,99	1	0,85	0,99
2	0,98	0,88	1	1	1
3	0,99	1	0,98	1	0,87
4	1	1	0,87	0,97	1
5	1	0,92	1	0,99	0,93
6	0,86	1	0,92	0,89	1
7	1	0,89	0,97	1	1
8	0,98	1	1	1	0,99
9	0,89	1	0,99	0,99	0,96
10	1	1	0,98	1	1
Average	0,97	0,968	0,971	0,969	0,974



3.8 Skuteczność aproksymacji automatu o 80 stanach

Tablica 10. 80 states automaton approximation testing set

	4 states	6 states	8 states	10 states	12 states
1	1	0,98	1	1	0,88
2	0,99	0,88	1	1	1
3	0,99	1	1	1	0,99
4	0,96	1	0,75	0,95	1
5	1	1	1	1	1
6	0,98	0,99	1	1	1
7	1	1	0,85	1	1
8	0,99	1	1	1	1
9	1	1	1	1	1
10	0,99	0,95	0,85	1	0,78
Average	0,99	0,98	0,945	0,995	0,965



Tablica 11. 80 states automaton approximation training set

	4 states	6 states	8 states	10 states	12 states
1	0,99	1	0,99	0,86	0,97
2	1	1	0,97	1	1
3	1	0,98	1	0,98	0,92
4	0,96	0,99	1	0,99	1
5	1	0,99	0,98	1	0,98
6	0,92	1	1	0,97	1
7	1	1	0,99	1	0,97
8	1	0,94	1	0,99	1
9	0,99	0,97	0,96	1	0,98
10	0,98	0,99	0,99	1	1
Average	0,984	0,986	0,988	0,979	0,982

