УДК 004.021

К. С. Гайдук, студент
О. Г. Шевченко, ст. викладач
Донецький національний технічний університет, Україна
ks.gayduk@gmail.com

Генерація детермінованих кінцевих автоматів

Розглянуто можливість застосування генетичних алгоритмів для генерації довільних детермінованих кінцевих автоматів Мілі. Виконано дослідження впливу параметрів розроблених алгоритмів на швидкість їх збіжності. Проаналізовано обмеження, що накладаються на структуру автомата, при використанні зазначених алгоритмів.

Ключові слова: кінцевий автомат, граф переходів, генетичний алгоритм, кросинговер, мутація, фітнес-функція.

Вступ

Вирішення ряду наукових та інженерних задач вимагає дослідження кінцевих автоматів з великою кількістю станів і вхідних сигналів. Прикладами областей, в яких виникають подібні задачі, можуть служити: проектування мікропроцесорів [1], розпізнавання мов [2,3], побудова кінцевих перетворювачів та ін. У зв'язку з чим, представляється актуальною проблема генерації довільних автоматів великої розмірності, на базі яких в подальшому можливе проведення різноманітних досліджень - зокрема, апробування алгоритмів декомпозиції, мінімізації кількості станів та ін.

Залежно від використовуваного класу генетичних алгоритмів, методи генерації кінцевих автоматів можна розподілити на кілька груп [4]:

1. Еволюційні алгоритми (за Л. Фогелем) [5], в яких виникнення нових особин популяції відбувається виключно за рахунок мутації. Характеризуються низькою швидкістю роботи та високою ймовірністю потрапляння в локальні екстремуми;

2. Генетичні алгоритми - окрім оператора мутації, використовують також оператор кросинговеру (схрещування). Мають більш високу, у порівнянні з еволюційними, швидкість роботи, однак вимагають використання популяції великого розміру;

3. Генетичне програмування - підклас генетичних алгоритмів, в яких особини описуються за допомогою дерев рішень.

У разі генерації автоматів, особини зазвичай представляються за допомогою бітових рядків, графів переходу або дерев рішень. При цьому, графи переходів можуть представлятися як у вигляді масивів вершин і дуг, так і кодуватися у вигляді бітових рядків [6].

В даній роботі досліджена можливість представлення графів переходів у вигляді матриць суміжності та допоміжних матриць для зберігання інформації про вхідні та вихідні сигнали, що зіставлені дугам.

Теоретичні засади

Як відомо [7], абстрактний автомат задається п'ятимісцевим кортежем:

$$S =< Z, W, Q, \delta, \lambda >, \quad (1)$$

де Z – вхідний алфавіт, W – вихідний алфавіт, Q – множина станів, δ – функція переходів, λ – функція виходів. Інколи [8] також вказується початковий стан автомата q_0.

У разі автомата Мілі, функції переходів і виходів виконують наступні відображення:

$$q(t+1) = \delta(z(t), q(t)), t = 0,1,2, \ldots, \quad (2)$$

$$w(t) = \lambda(z(t), q(t)), t = 0,1,2, \ldots, \quad (3)$$

де $z(t)$, $w(t)$, $q(t)$ – значення вхідного сигналу, вихідного сигналу та стану в поточний момент часу t відповідно, $q(t+1)$ – стан автомата в момент часу $t+1$.

За умови повної визначеності автомата [7], функції δ та λ визначені на всій множині $Z \times Q$ (інакше кажучи, таблиці переходів і виходів повністю заповнені), з чого випливає, що ступінь виходу кожної вершини графа переходів $\vec{G}$ в такому випадку дорівнює потужності вхідного алфавіту:

$$deg_-(v_i) = |Z|, i = \overline{1,n}, n = |Q|, v_i \in V, \quad (4)$$

$$deg_-(v_i) + deg_+(v_i) = deg(v_i), i = \overline{1,n}, \quad (5)$$

де $deg_+(v_i)$ та $deg_-(v_i)$ – ступені входу та виходу вершини v_i відповідно, $\deg(v_i)$ – ступінь вершини v_i, V – множина вершин.

Відомо, що сума ступенів входу всіх вершин дорівнює сумі ступенів виходу всіх вершин, і дорівнює кількості ребер:

$$\sum_{i=1}^{n} deg_+(v_i) = \sum_{i=1}^{n} deg_-(v_i) = |U| = |X \cup L|, \quad (6)$$

де U – множина ребер, X – множина дуг, L – множина петель.

З урахуванням (4), маємо:

$$\sum_{i=1}^{n} deg_+(v_i) = n|Z| = |U| = |Z \times Q| = |Z|n. \quad (7)$$

За умови $deg_+(v_i) = |Z|, i = \overline{1,n}$ рівність (7) виконується, і для кожної вершини стають можливими два випадки:

1. Вершина інцидентна $|Z|$ вхідним дугам і $|Z|$ вихідним дугам;

2. Вершина інцидентна $|Z|-1$ вхідним дугам, $|Z|-1$ вихідним дугам і одній петлі.

В обох випадках буде справедливою рівність

$$deg(v_i) = 2|Z|, i = \overline{1,n}. \quad (8)$$

Нехай, граф переходів $\vec{G}$ представлений за допомогою матриці суміжності $\vec{A}$ [9]. Позначимо через $G =< V, U >$ граф, отриманий в результаті дезорієнтації дуг [10] графа переходів, а відповідну йому матрицю суміжності як A. Якщо

розглядати A як матрицю цілих випадкових чисел розподілених в інтервалі $[0;1]$, і позначити ймовірність рівності елемента матриці одиниці як $p_s = p(A_{i,j} = 1), i = \overline{1,n}, j = \overline{1,n},$ то справедливим буде вираз

$$|U| = |X| + |L| = p_s \frac{n(n-1)}{2} + p_s n = |Z|n. \quad (9)$$

Виконавши найпростіші перетворення, отримаємо:

$$p_s = \frac{2|Z|}{n+1} = \frac{2|Z|}{|Q|+1}. \quad (10)$$

Опис алгоритму

Пропонований алгоритм генерації довільного детермінованого кінцевого автомата (ДКА) складається з наступних кроків:

1. Генерація неорієнтованого (базового) графа G, в якого ступені всіх вершин рівні $2|Z|$.

2. Отримання орієнтованого графа $\vec{G}^*$, шляхом орієнтації ребер графа G таким чином, щоб вхідна та вихідна ступені кожної вершини були рівні $|Z|$.

3. Зіставлення вихідним дугам кожної вершини $\vec{G}^*$ букв вхідного алфавіту таким чином, щоб виконувалася умова

$$\bigcap_{j=1}^{|Z|} u_{ij}^{-} = \emptyset, \quad (11)$$

де u_{ij}^{-} – j-й елемент множини букв, зіставлених вихідним дугам i-ої вершини.

4. Зіставлення вихідним дугам кожної вершини $\vec{G}^*$ букв вихідного алфавіту (виконується випадковим чином).

5. Побудова таблиць переходів і виходів.

В результаті виконання алгоритму буде отримано довільний ДКА Мілі та відповідний йому граф переходів $\vec{G} =< \vec{G}^*, \delta, \lambda >$.

Генерація базового графа

Для генерації базового графа був використаний генетичний алгоритм (ГА) [11]. Кожна особина початкової популяції створювалася в два етапи:

1. Генерація матриці випадкових чисел R розміром $n\,x\,n$, в якій ймовірність рівності елемента одиниці дорівнює p_s. З ймовірністю $1 - p_s$ елементи матриці R дорівнюють нулю.

2. Формування матриці суміжності A:

$$H = \begin{cases} R_{i,j}, j \le i, i = \overline{1,n}, \\ 0, j > i, i = \overline{1,n}, \end{cases} \quad (12)$$

$$A = H \vee H^T, \quad (13)$$

де H – матриця, в якій елементи на головній діагоналі, а також нижче неї, рівні відповідним елементам матриці R, а елементи вище головної діагоналі дорівнюють нулю, A – результат логічного складання матриці H зі своєю транспонованою копією.

Оскільки $p(R_{i,j} = 1) = p_s$ для усіх $i = \overline{1,n}, j = \overline{1,n}$, справедливим буде і вираз $p(A_{i,j} = 1) = p_s, i = \overline{1,n}, j = \overline{1,n}$.

Фітнес-функція $f_1 = f(A)$ розраховувалася як кількість вершин, ступінь яких дорівнює $m = 2|Z|$:

$$M = E_{n\,x\,n} + I_{n\,x\,n}, \quad (14)$$

$$P = A \circ M, \quad (15)$$

$$D = P \cdot I_{n\,x\,1}, \quad (16)$$

$$D' = \{x | x \in D \wedge x = m\}, \quad (17)$$

$$f_1 = |D'|, \quad (18)$$

де $E_{n\,x\,n}$ – одинична матриця, $I_{n\,x\,n}$ – матриця, всі елементи якої дорівнюють одиниці, P – результат поелементного множення A та M, D – результат матричного множення матриці P на вектор-стовпець одиниць $I_{n\,x\,1}$, D' - підмножина вектор-стовпця D, елементи якого дорівнюють m.

Поділ хромосом в операторі кросинговеру (ОК) визначений в такий спосіб:

$$s = rand(1-n; n-2), s \in \mathbb{Z}, \quad (19)$$

$$A_l = \begin{cases} A_{i,j}, i + j \le n + 1 - s, i = \overline{1,n}, \\ 0, i + j > n + 1 - s, i = \overline{1,n}, \end{cases} \quad (20)$$

$$A_r = \begin{cases} A_{i,j}, i + j > n + 1 - s, i = \overline{1,n}, \\ 0, i + j \le n + 1 - s, i = \overline{1,n}, \end{cases} \quad (21)$$

де s – випадкове ціле число в інтервалі $[1-n; n-2]$ (розподіл чисел при генерації рівномірний), A_l – ліва половина хромосоми, A_r – права половина хромосоми. Поділ хромосоми проілюстровано на рис. 1 (а-в).

Сам ОК виконується відповідно до рівнянь, наведених нижче:

$$\begin{gathered} \{A_l; A_r\} = chrom\_div(A, s), \\ \{B_l; B_r\} = chrom\_div(B, s), \\ C = A_l + B_r, \\ D = B_l + A_r, \end{gathered} \quad (22)$$

де $chrom\_div(A, s)$ та $chrom\_div(B, s)$ – поділ батьківських хромосом, виконаний відповідно до (20) і (21), C і D – хромосоми, що відповідають двом особинам нової популяції.

Ймовірність виконання кросинговеру на етапі генерації базового графа позначимо як p_{c1}.

В ході виконання оператора мутації (ОМ) з однаковою ймовірністю виконується додавання нового ребра або видалення існуючого.

Додавання нового ребра виконується відповідно до алгоритму:

1. Якщо $\sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} = n^2$, - вихід, тому як граф повний і додавання ребра неможливо;

2. Розрахунок значень ступеня кожної вершини відповідно до (14) - (16), отримання вектора D;

3. $q = \min(D)$ – знаходження значення мінімального ступеня;

4. Отримання підмножини вершин T, ступінь яких дорівнює мінімальному;

5. $r_1 = T(rand(1, |T|))$ – вибір випадкової вершини з множини T;

6. Вибір множини вершин V', не інцидентних r_1 (тобто, таких, для яких $A_{r_1 j} = 0, j \in \overline{1,n}$);

7. Знаходження значення мінімального ступеня q' для вершин з множини V';

	1	2	3	4	5
1	1	0	1	1	0
2	0	0	0	1	1
3	1	0	1	0	0
4	1	1	0	0	1
5	0	1	0	1	1

а)

	1	2	3	4	5
1	1	0	1	1	0
2	0	0	0	1	1
3	1	0	1	0	0
4	1	1	0	0	1
5	0	1	0	1	1

б)

	1	2	3	4	5
1	1	0	1	1	0
2	0	0	0	1	1
3	1	0	1	0	0
4	1	1	0	0	1
5	0	1	0	1	1

в)

Рисунок 1 – Випадки поділу хромосоми на ліву та праву частини при $s = -4$, $s = 0$ та $s = 3$ відповідно.

8. Отримання підмножини вершин $T' = \{x | x \in V', \deg(x) = q'\}$;

9. $r_2 = T'(rand(1, |T'|))$ – вибір випадкової вершини з множини T';

10. Якщо $r_1 = r_2$, то $A_{r_1 r_2} = \neg A_{r_1 r_2}$ (додавання петлі); інакше, $A_{r_1 r_2} = \neg A_{r_1 r_2}$ та $A_{r_2 r_1} = \neg A_{r_2 r_1}$ (додавання ланки).

Видалення існуючого ребра виконується аналогічно, за тим лише винятком, що на першому кроці виконується перевірка умови $\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij} = 0$ (якщо матриця суміжності описує нуль-граф, то вихід), і виконується пошук вершин з максимальним значенням ступеня, а не мінімальним. Крім того, в п. 6 виконується вибір інцидентних вершин.

Якщо r_1 та r_2 генерувати випадковим чином, то, в разі розрідженого графа, ребра будуть частіше додаватися, ніж видалятися (а в разі щільного графа - частіше видалятися, ніж додаватися), що різко знизить швидкість збіжності алгоритму - аж до того, що він стане непридатним для практичного використання.

Імовірність мутації на етапі генерації базового графа позначимо як p_{m1}.

Оператор репродукції (ОР) визначено наступним чином:

1. Вектор значень фітнес-функції кожної особини нормується таким чином, щоб сума його елементів була рівною одиниці; отриманий нормований вектор розглядається як вектор значень ймовірності участі кожної особини в схрещуванні;

2. На підставі отриманого вектора ймовірностей, методом Монте-Карло розігрується два цілих випадкових числа з інтервалу $[1; n]$, що визначають номери особин для схрещування;

3. Для обраної пари з ймовірністю p_{c1} виконується оператор кросинговеру;

4. Для кожної з двох утворених дочірніх особин з ймовірністю p_{m1} виконується оператор мутації;

5. пп. 2-4 повторюються до тих пір, поки не буде згенеровано нову популяцію необхідної потужності (потужність початкової популяції N_1 дорівнює потужності наступних популяцій, і кратна двом).

Вихід з циклу ГА виконується за умовою

$$f_1(A_{ld}) = n, \quad (23)$$

де A_{ld} – особина-лідер.

Орієнтація базового графу

Орієнтація базового графа, як і його генерація, виконується за допомогою ГА. В цілому, алгоритм аналогічний попередньому, проте є відмінності в способі генерації початкової популяції, а також способі обчислення фітнес-функції і виконанні оператора мутації.

На вхід алгоритм приймає базовий неорієнтовані граф G, представлений матрицею суміжності A. На етапі формування початкової популяції створюється N_2 (за числом особин в популяції) копій матриці A, після чого виконується випадкова орієнтація кожного з графів наступним чином:

$$r = rand(0; 1), r \in \mathbb{R}, r \in (0; 1), \quad (24)$$

$$\vec{A} := A, \quad (25)$$

$$\vec{A} = \begin{cases} \vec{A}_{ij} := 0, if\ r < 0.5, \\ \vec{A}_{ji} := 0, if\ r \geq 0.5, \\ i = \overline{1, n-1}, j = \overline{i+1, n}, \end{cases} \quad (26)$$

де $:=$ – оператор присвоювання.

В алгоритмі випадкової орієнтації графа використано властивість симетрії матриці суміжності: в матриці суміжності A, що відповідає неорієнтованому графу, у разі наявності в графі ланки, що з'єднує вершини i та j, елементи матриці суміжності A_{ij} та A_{ji}, розташовані симетрично відносно головної діагоналі, дорівнюватимуть одиниці. Алгоритм пробігає по всіх елементах вище головної діагоналі (елементи самої головної діагоналі відповідають петлям, і тому не обробляються), і при обробці кожного елемента $\vec{A}_{ij}$ з однаковою ймовірністю виконується дія $\vec{A}_{ij} := 0$ або $\vec{A}_{ji} := 0$ – в результаті, якщо до обробки матриці $\vec{A}$, отриманої після (26), малася рівність $\vec{A}_{ij} = \vec{A}_{ji} = 0$ (тобто, відповідної ланки не було), то воно збережеться. В іншому випадку, одна з двох одиниць, відповідних одній ланці, буде обнулена. Випадкове число r генерується заново при обробці кожного елемента $\vec{A}_{ij}$.

В результаті виконання (25) - (26) для кожної матриці A, буде сформована початкова популяція потужністю N_2 ($N_2 \in 2k, k = 1,2, \ldots$).

Фітнес-функція особини $f_2 = f(\vec{A})$ оцінюється як кількість вершин, ступінь виходу яких дорівнює $|Z| = 0.5m$:

$$f_2 = \left|\{x | x \in \vec{A} \cdot I_{n\,x\,1} \wedge x = 0.5m\}\right|. \quad (27)$$

Добуток $\vec{A} \cdot I_{n\,x\,1}$ – це не що інше, як вектор-стовпець значень ступенів виходу для кожної вершини. f_2 дорівнює потужності підмножини множини $\vec{A} \cdot I_{n\,x\,1}$, елементи якої дорівнюють $|Z|$.

Виконання оператора мутації полягає в зміні орієнтації однієї з дуг графа. В алгоритмі оператора мутації виділено два варіанти зміни орієнтації дуги: "розумний" і випадковий. У разі "розумного" варіанту, вершини, ступінь виходу яких менше або дорівнює $|Z|$, залишаються незмінніми (якщо ступінь виходу вершини дорівнює $|Z|$, то вона вже задовольняє вимогам, що висуваються до кожної вершини шуканого графа; якщо ступінь виходу вершини менше $|Z|$, то зміна орієнтації однієї з дуг зменшить ступінь виходу ще на одиницю, тим самим погіршивши показник, який оптимізується для вершини). У разі випадкової зміни орієнтації дуги, вибираються довільні дві вершини, пов'язані дугою (за винятком петель). Ймовірність виконання "розумного" варіанту задається параметром p_i. Використання одного тільки "розумного" варіанту небажано, тому що може виникнути ситуація, в якій ближче до кінця роботи ГА у деяких вершин зі ступенем виходу більше $|Z|$ не буде інцидентних їм вершин, ступінь виходу яких був би відмінним від $|Z|$ – алгоритм в такому випадку зійдеться, але буде потрібно більше ітерацій, ніж у разі комбінування "розумного" і випадкового варіантів.

Алгоритм "розумної" зміни орієнтації дуги:

1. $\Omega = \vec{A} \cdot I_{n\,x\,1}$ – отримання вектор-стовпця значень ступенів виходу кожної з вершин;

2. $\beta = \{x | x \in \Omega \wedge x > |Z|\}$ – множина вершин, ступінь виходу яких більше $|Z|$; якщо $\beta = \emptyset$, вихід;

3. $r_1 = \beta(rand(1, |\beta|))$ – вибір випадкової вершини з множини β;

4. $\gamma = \{x | x = v_i \in V, i = \overline{1,n}, \vec{A}_{r_1 i} = 1\}$ – вибір множини вершин, інцидентних r_1, причому дуги, що зв'язують r_1 і вершини множини γ, для r_1 будуть вихідними;

5. $r_2 = \gamma(rand(1, |\gamma|))$ – вибір випадкової вершини з множини γ;

6. пп. 3-5 виконується доти, поки $r_1 = r_2$ або $\sum_j^n \vec{A}_{r_2 j} = |Z|$; щоб уникнути зациклення, перебір виконується не більше ніж n раз.

Алгоритм випадкової зміни орієнтації дуги:

1. $\Omega = \vec{A} \cdot I_{n\,x\,1}$ – отримання вектор-стовпця значень ступенів виходу кожної з вершин;

2 $\beta = \{x | x \in \Omega \wedge x > 0\}$ – множина вершин, ступінь виходу яких більше 0; якщо $\beta = \emptyset$, вихід; умова $x > 0$ накладається з тієї причини, що для вершин, у яких немає вихідних дуг, при використовуваному алгоритмі не вдасться знайти суміжні вершини;

3. $r_1 = \beta(rand(1, |\beta|))$ – вибір випадкової вершини з множини β;

4. $\gamma = \{x | x = v_i \in V, i = \overline{1,n}, \vec{A}_{r_1 i} = 1\}$ – вибір множини вершин, інцидентних r_1, причому дуги, що зв'язують r_1 і вершини множини γ, для r_1 будуть вихідними;

5. $r_2 = \gamma(rand(1, |\gamma|))$ – вибір випадкової вершини з множини γ;

6. пп. 3-5 виконується доти, поки $r_1 = r_2$; щоб уникнути зациклення, перебір виконується не більше ніж n раз.

Ймовірність мутації на етапі орієнтації базового графа позначимо як p_{m2}. Ймовірність кросинговеру – p_{c2}.

Вихід з циклу ГА виконується за умовою

$$f_2(\vec{A}_{ld}) = n, \quad (28)$$

де $\vec{A}_{ld}$ – особина-лідер.

Формування таблиць переходів та виходів

Зіставлення вхідних сигналів з дугами графа $\vec{G}^*$ виконується наступним чином:

1. Генерується вектор $Z = \{1; \dots; l\}, l = |Z|$;

2. Генерується ціле випадкове число s в діапазоні $[1; l-1]$;

3. Виконується розбиття

$\{\{z(1), \dots, z(s)\}, \{z(s+1), \dots, z(l)\}\}$;

4. Блоки отриманого розбиття міняються місцями:

$Z' = \{\{z(s+1), \dots, z(l)\}, \{z(1), \dots, z(s)\}\}$;

5. Елементи отриманої множини Z' ставляться у відповідність вихідним дугам оброблюваної вершини графа в порядку появи одиниць у відповідному рядку матриці суміжності;

6. пп. 2-5 виконуються для кожної вершини графа.

Кожній дузі також ставиться у відповідність випадкове число з діапазону $[1; |W|]$, що визначає вихід автомата для відповідного переходу.

Результати та аналіз

Для алгоритму генерації базового графа були досліджені наступні залежності:

1. Залежність кількості ітерацій k від потужності популяції N_1, а також від кількості станів автомата n, що генерується (рис. 2 (а));

2. Залежність логарифма кількості ітерацій $\log(k+1)$ від кількості станів автомата n, що генерується, а також потужності вхідного алфавіту $|Z|$ (рис. 2 (б));

3. Залежність логарифма кількості ітерацій $\log(k+1)$ від ймовірностей кросинговеру p_{c1} та мутації p_{m1} (рис. 2 (в));

4. Залежність фітнес-функції особи-лідера $f_1(A_{ld})$ від номера ітерації, а також значень p_{c1} (рис. 2 (г)).

З рис. 2 (а) видно, що немає сенсу збільшувати кількість особин популяції пропорційно кількості станів n. Для розглянутого випадку ($n \leq 40$) достатньо прийняти $N_1 = 10$.

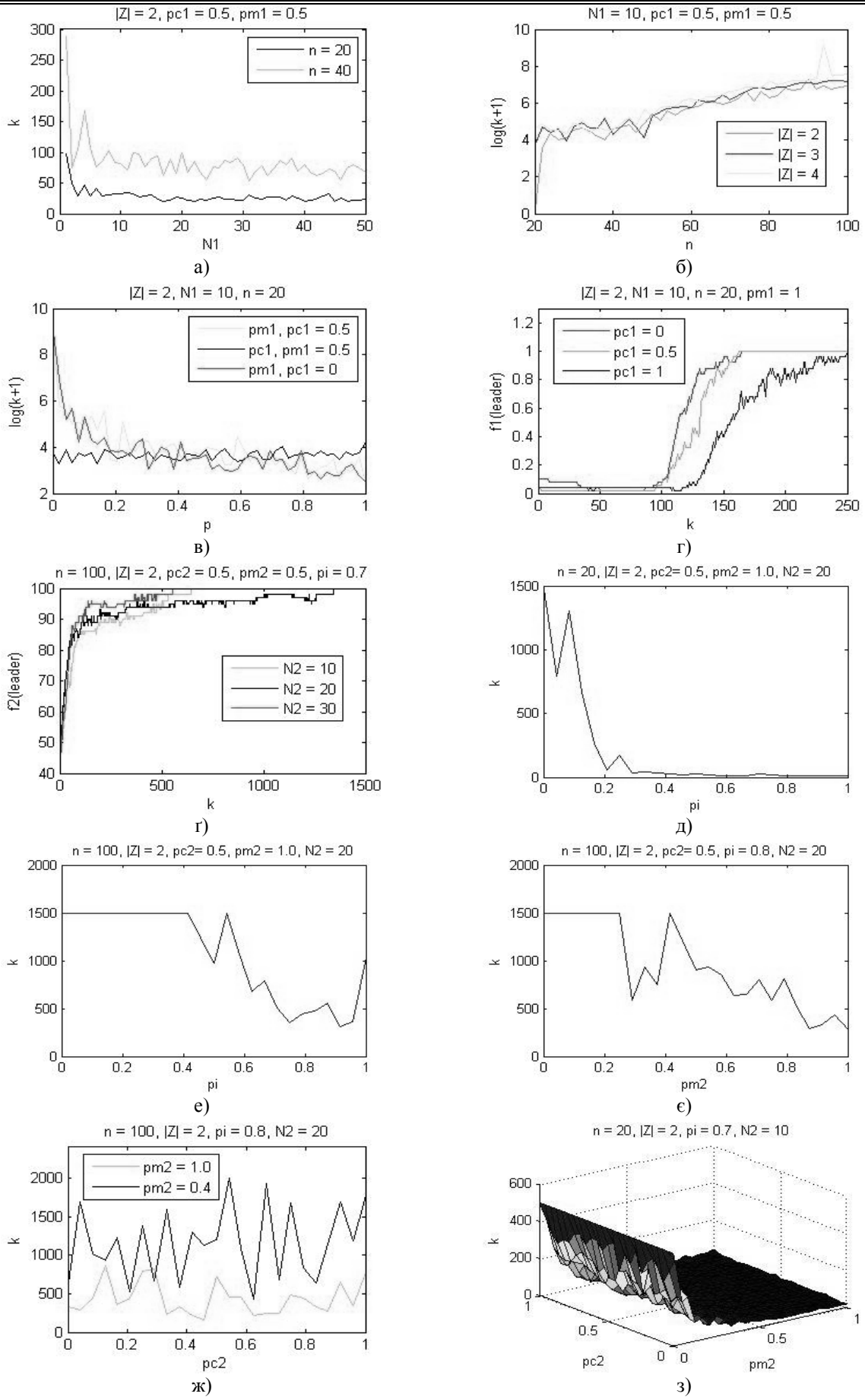

Рисунок 2 – Дослідження алгоритмів генерації та орієнтації базового графа.

Тут також варто розуміти, що збільшення потужності популяції може призвести до зменшення числа ітерацій алгоритму, але може не скоротити тривалість його виконання.

Залежність кількості ітерацій від потужності вхідного алфавіту носить експонентний характер, що видно з рис. 2 (б). Так, при $n = 100$ та $|Z| = 2$ знадобилося 994 ітерації, при $|Z| = 3$ – 1319 ітерацій, при $|Z| = 4$ – 1975 ітерації алгоритму. З (10) також випливає, що завжди варто враховувати нерівність $2|Z| \le n + 1$ – в іншому випадку, алгоритм не зійдеться.

Було встановлено (рис. 2 (г)), що кросинговер зменшує швидкість збіжності алгоритму, але не збільшує її. У разі $p_{c1} = 0$, обміну генетичним матеріалом між особинами не відбувається, і алгоритм назвати генетичним можна лише умовно. Т.ч., на етапі генерації базового графа параметри p_{c1} та p_{m1} варто прийняти рівними 0 та 1 відповідно.

Для алгоритму орієнтації базового графа були досліджені наступні залежності:

1. Залежність фітнес-функції особи-лідера $f_2(\vec{A}_{ld})$ від номера ітерації, а також від потужності популяції N_2 (рис. 2 (г));
2. Залежність кількості ітерацій k від ймовірності вибору "розумного" варіанту зміни орієнтації дуг p_i в операторі мутації (рис. 2 (д, е));
3. Залежність кількості ітерацій k від ймовірності мутації p_{m2} (рис. 2 (є));
4. Залежність кількості ітерацій k від ймовірності кросинговеру p_{c2} (рис. 2 (ж));
5. Залежність кількості ітерацій k від ймовірностей p_{c2} та p_{m2} (рис. 3 (з)).

З рис. 2 (г) видно, що, зі збільшенням потужності популяції, крутизна функції $f_2(\vec{A}_{ld}, k)$ на початку збільшується, проте це не гарантує збільшення швидкості збіжності алгоритму. Наприклад, при $N_2 = 30$ крутизна функції $f_2(\vec{A}_{ld}, k)$ на інтервалі $k \in [1; 200]$ була максимальної, але алгоритм зійшовся пізніше, ніж при $N_2 = 10$ та $N_2 = 20$.

При $n = 20$ залежність кількості ітерацій алгоритму від ймовірності p_i виглядає досить тривіально (рис. 2 (д)): при $p_i \ge 0{,}2$ спостерігається різке скорочення кількості ітерацій. Однак при $n = 100$ (рис. 2 (е)) проявляється ситуація, про яку було згадано вище: ближче до кінця роботи ГА у деяких вершин зі ступенем виходу більше $|Z|$ не буде інцидентних їм вершин, ступінь виходу яких була б відмінною від $|Z|$, і це знижує швидкість збіжності алгоритму. Т.ч., для випадку $n = 100$ p_i краще прийняти рівним 0,7-0,9. При побудові графіка рис. 2 (е) для кожної точки виконувалося усереднення за результатами п'яти вимірювань. Максимально допустима кількість ітерацій алгоритму k_{lim} (обмеження, встановлене з метою запобігання зациклення і надмірно тривалого пошуку рішення) було прийнято рівним 1500.

Впливу ймовірності кросинговеру на швидкість збіжності алгоритму не виявлено (рис. 2 (ж, з)).

Результати вимірювання часу виконання алгоритмів генерації і орієнтації базового графа представлені в табл. 1. В ході експериментів були використані наступні значення параметрів: $|Z| = 2$, $N_1 = N_2 = 50$, $p_{m1} = p_{m2} = 1.0$, $p_{c1} = p_{c2} = 0$, $p_i = 0.8$. Усереднення виконувалося за результатами трьох вимірів.

Таблиця 1 – Залежність швидкості збіжності алгоритмів від кількості вершин графа переходів.

n	Генерація базового графа			Орієнтація базового графа		
	$\bar{k}$	$\bar{t}$, с	$\bar{t}_{it}$, с	$\bar{k}$	$\bar{t}$, с	$\bar{t}_{it}$, с
50	37	0.6	0,02	62	1.9	0.03
100	87	2.6	0,03	459	25.9	0.06
200	183	15.8	0,09	1835	248.8	0.14

Позначення, прийняті в табл. 1: $\bar{k}$ - середня кількість ітерацій; $\bar{t}$ - середній час виконання, с; $\bar{t}_{it}$ - середній час однієї ітерації, с.

З прикладами згенерованих автоматів, представлених у вигляді графів переходів в форматі DOT, можна ознайомитися за посиланням [12].

Висновки

Розглянуто можливість застосування генетичних алгоритмів для генерації довільних автоматів Мілі. Представлений в роботі спосіб дозволяє досягти необхідного результату, проте має ряд обмежень, серед яких неможливість генерації графів з кратними ребрами та генерація виключно ейлерових графів.

Реалізація алгоритмів і їх дослідження проводилися в середовищі MATLAB на машині з 2,1 ГГц ЦП та 2 Гб ОЗП.

Список використаної літератури

1. Бурдонов И. Б. Исследование графа набором автоматов / И. Б. Бурдонов, А. С. Косачев, В. В. Кулямин // Программирование. – 2015. – №6. – С. 3-8.
2. Поликарпова Н.И. Метод сокращения таблиц для генерации автоматов с большим числом входных переменных на основе генетического программирования / Н. И. Поликарпова, В. Н. Точилин, А. А. Шалыто // Известия РАН. Теория и системы управления. – 2010. – №2. – С. 100-117.
3. Александров Д. Е. Сложность распознавания принадлежности слова регулярному языку в системах обнаружения вторжений : дис. кандидата ф.-м. наук : 05.13.19 / Александров Дмитрий Евгеньевич ; Московський держ. університет ім. М. В. Ломоносова. – Москва, 2015.

4. Лобанов П. Г. Использование генетических алгоритмов для генерации конечных автоматов : дис. к. т. н. : 05.13.11 / Лобанов Павел Геннадьевич ; Санкт-Петербурзький державний університет інформаційних технологій механіки і оптики. – СПб, 2008.
5. Фогель Л. Искусственный интеллект и эволюционное моделирование / Фогель Л., Оуэнс А., Уолш М. – Москва : Мир, 1969. – 230 с.
6. Jefferson D. Evolution as a Theme in Artificial Life [Електронний ресурс] / D. Jefferson, R. Collins, C. Cooper та ін. // Лос-Анджелес, 1992. – Режим доступу: http://web.cs.ucla.edu/~dyer/Papers/AlifeTracker/Alife91Jefferson.html
7. Гуренко В. В. Введение в теорию автоматов / В. В. Гуренко. – М. : МДТУ ім. Н.Е. Баумана, 2013. – 62 с.
8. Баранов С. И. Синтез микропрограммных автоматов / С. И. Баранов. – Ленинград : "Энергия", 1979. – 232 с.
9. Кормен Т. Алгоритмы: построение и анализ / Т. Кормен та ін. – М. : ООО "И. Д. Вильямс", 2013. – 1328 с.
10. Глушков В. М. Энциклопедия кибернетики : в 2 т. / В. М. Глушков та ін. – Київ : Головна редакція УРЕ, 1974. – Т. 1. – 608 с.
11. Скобцов Ю. А. Основы эволюционных вычислений / Ю.А. Скобцов. – Донецьк : ДонНТУ, 2008. – 326 с.
12. Гайдук К. С. Довільні ДКА Мілі / К. С. Гайдук // Покровськ, 2017. – Режим доступу: https://github.com/ks-gayduk/dfsm_miles

Надійшла до редакції 10.04.2017

К.С. ГАЙДУК, О.Г. ШЕВЧЕНКО
Донецкий национальный технический университет, г. Покровск, Украина

ГЕНЕРАЦИЯ ДЕТЕРМИНИРОВАННЫХ КОНЕЧНЫХ АВТОМАТОВ

Рассмотрена возможность применения генетических алгоритмов для генерации произвольных детерминированных конечных автоматов Мили. Выполнено исследование влияния параметров разработанных алгоритмов на скорость их сходимости. Проанализированы ограничения, накладываемые на структуру генерируемого автомата при использовании указанных алгоритмов.

Ключевые слова: конечный автомат, граф переходов, генетический алгоритм, кроссинговер, мутация, фитнесс-функция.

K. GAYDUK, O. SHEVCHENKO
Donetsk National Technical University, Pokrovsk, Ukraine

GENERATION OF DETERMINATED FINITE AUTOMATES

The solution of number scientific and engineering problems requires the study of finite state machines with large number of states and input signals. Examples of areas in which such problems may be: the designing of microprocessors, the creation of artificial intelligence systems, the recognition of regular languages, etc. In this connection, it is an urgent problem to generate random high-dimensional machines, on which then may conduct various studies - in particular testing algorithms decomposition, minimizing the number of states and others.

This paper presents and investigates an algorithm generating random deterministic finite Mealy machines of a predetermined number of states. The algorithm consists of three main stages: the generation of a random non-oriented graph with the number of vertices corresponding to the required number of states of the automaton; transformation of the obtained base graph into oriented; comparison of the arcs of the oriented graph of the values of the input and output signals.

Generation and orientation of the base graph was performed with using of genetic algorithms. Each individual was described by a two-dimensional chromosome, represented by the adjacency matrix of the corresponding graph.

The influence of the parameters of the genetic algorithms used on the rate of their convergence was investigated. Established that a given machine representation in computer memory, and also for given definitions of genetic operators and ways of calculating the fitness function, the crossing-over operator reduces the rate of convergence of the algorithm in the case of generation of the base graph, and does not effect on the rate of convergence of the algorithm in the case of the orientation of the base graph.

Restrictions imposed on the graph conversion machine used method of filing machine in computer memory are analyzed and indicated.

The constraints imposed on the automaton transition graph by the method used to represent the automaton in the computer memory.

The results of measurements of the execution time of algorithms and the rate of their convergence are presented depending on the number of states of the generated automaton.

Keywords: finite state machine, transition graph, genetic algorithm, crossing-over, mutation, fitness function.

REFERENCES

1. Burdonov, I.B.,Kosachev, A.S., Kulyamin ,V.V.(2015), *A study of a graph by a set of machine* [*Issledovanie grafa naborom avtomatov*], Programming, №6, pp. 3-8.
2. Polikarpova, N.I., Tochilin V.N., Shalyito A.A.(2010), *The method of shortening tables for generating machine with a large number of input variables based on genetic programming* [*Metod sokrascheniya tablits dlya generatsii avtomatov s bolshim chislom vhodnyih peremennyih na osnove geneticheskogo programmirovaniya*], Izvestiya PAN. Teoriya i sistemyi upravleniya, №2, pp.100-117.
3. Aleksandrov,D.E. (2015), *The difficulty of recognizing the word's belonging to a regular language in intrusion detection systems* [*Sloghnost raspoznavaniya prinadleghnosti slova regulyarnomu yazyiku v sistemah obnarugheniya vtorgheniy*],dis.candidate f.-m. sciences: 05.13.19, Moscow State University at Lomonosov M.V.
4. Lobanov, P.G. (2008) ,T*he use of genetic algorithms for the generation of finite machine* [*Isspolzovanie geneticheskih algoritmov dlya generatsii konechnyih avtomatov]*, dis. c.t.s: 05.13.11, Petersburg State University of Informatics Technology Mechanics and Optics.
5. Vogel L (1969). *Artificial intelligence and evolutionary modeling* [*Isskustvennyiy intellect i evolyutsionnoe modelirovanie*], Moscow, 1969, -230p.
6. Jefferson D. *Evolution as a Theme in Artificial Life* (1992) / D. Jefferson, R. Collins, C. Cooper- Los-Angeles,1992, available at: http://web.cs.ucla.edu/~dyer/Papers/AlifeTracker/ Alife91Jefferson.html
7. Gurenko, V.V. (2013) *Introduction to the theory of machine* [*Vvedenie v teoriyu avtomatov*], Moscow, MDTU at Bauman, 2013, -62p.
8. Baranov, S.I. (1979) *Synthesis of microprogram machine* [*Sintez mikroprogramnyih avtomatov*], Leningrad,1979. - 232 p
9. Kormen T. (2013) *Algorithms: construction and analysis* [*Algoritmyi: postroenie i analiz*], Moscow, 2013. - 1328 p.
10. Glushkov, V.M. (1974) *Encyclopedia of cybernetics: 2 vol* [*Entsiklopediya kibernetiki: v 2 t.*], Kiev, 1974.-V.1.-608p.
11. Skobtsov ,Yu.A. (2008) *Fundamentals of evolutionary computations* [*Osnovyi evolyutsionnyih vyichisleniy*], Donetsk: DonNTU, 2008. – 326p.
12. Gaiduk K. S. *Arbitrary RFA Mealy* [*Dovilni DKA Mili*], available at : https://github.com/ks-gayduk/dfsm_miles.