РАЗДЕЛ «ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ»

УДК 004.4′242

ЧЕРНОМУРОВА Л.О., к.ф.-м.н., доцент
ІВАНІН Д.О., к.ф.-м.н., доцент
САВИЦЬКА К.В., магістр

Дніпродзержинський державний технічний університет

ПОБУДОВА КІНЦЕВИХ АВТОМАТІВ З ВИКОРИСТАННЯМ ЙМОВІРНІСНИХ АЛГОРИТМІВ

Вступ. В останній час все ширше застосовується автоматне програмування, в межах якого поведінка програм описується за допомогою скінчених детермінованих автоматів. Такі автомати знаходять широке застосування для розпізнавання регулярних мов, при побудові дискретних пристроїв керування. Зокрема, теорія побудови компіляторів базується на ієрархії моделей автоматів.

Для багатьох задач автомати вдається будувати евристично. Наочним прикладом, в якому автоматична побудова логіки сутності зі складною поведінкою має вирішальне значення, є одна з класичних задач із області спільного використання ймовірнісних алгоритмів і кінцевих автоматів – задача про «Розумну мураху» [1] та модифікації цієї задачі.

Постановка задачі. Наведемо найпростішу постановку задачі про «Розумну мураху». Задано клітинне поле, по якому може пересуватися мураха. Поле має розміри $n \times n$, воно перебуває на поверхні двовимірного тора і на ньому знаходиться N яблук. В деяких клітинах поля розміщена їжа (яблуко).

На рис.1 зображено приклад поля з їжею, що розміщена в заповнених клітинах.

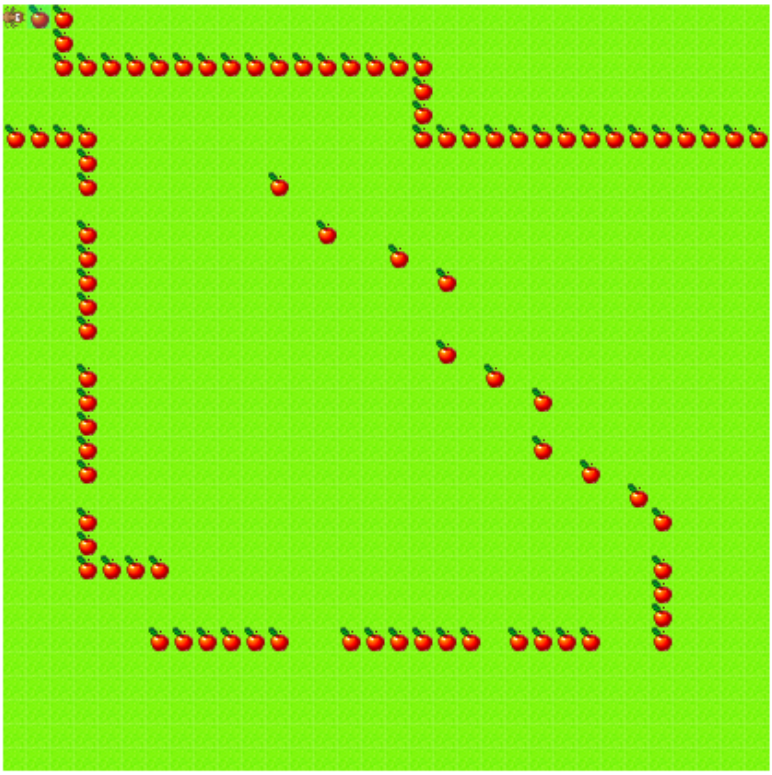

Рисунок 1 – Приклад поля для «Розумної мурахи»

З'їдена мурахою їжа не поповнюється, мураха живе на протязі всієї гри. Вона може пересуватися по будь-яким клітинам поля та обирати напрямок руху. За один хід вона здійснює одну з трьох дій:

– перейти в клітину, що знаходиться перед нею;

– повернути ліворуч;

– повернути праворуч.

Мурасі відомо, чи знаходиться їжа в клітині перед нею. Задача мурахи – почати рух з краю поля, за k ходів з'їсти якомога більше одиниць їжі (яблук), тобто відвідати максимальне число клітин, які містять їжу. Після проходження k ходів підраховується кількість їжі, що з'їдена мурахою. Це значення і є результатом гри.

Для даної задачі цільову функцію можна представити у наступному вигляді:

$$F = eaten - \frac{steps}{k}, \tag{1}$$

де *eaten* – кількість з'їденої їжі (яблук);

steps – номер кроку, на якому було з'їдено останнє яблуко;

k – задана кількість кроків.

Один із способів опису поведінки мурахи – кінцевий автомат з діями на переходах (автомат Мілі), у якого є одна вхідна змінна логічного типу (чи знаходиться їжа перед мурахою), а множина вихідних впливів складається з перерахованих дій. Приклад такого автомату представлено у вигляді графа переходів на рис.2.

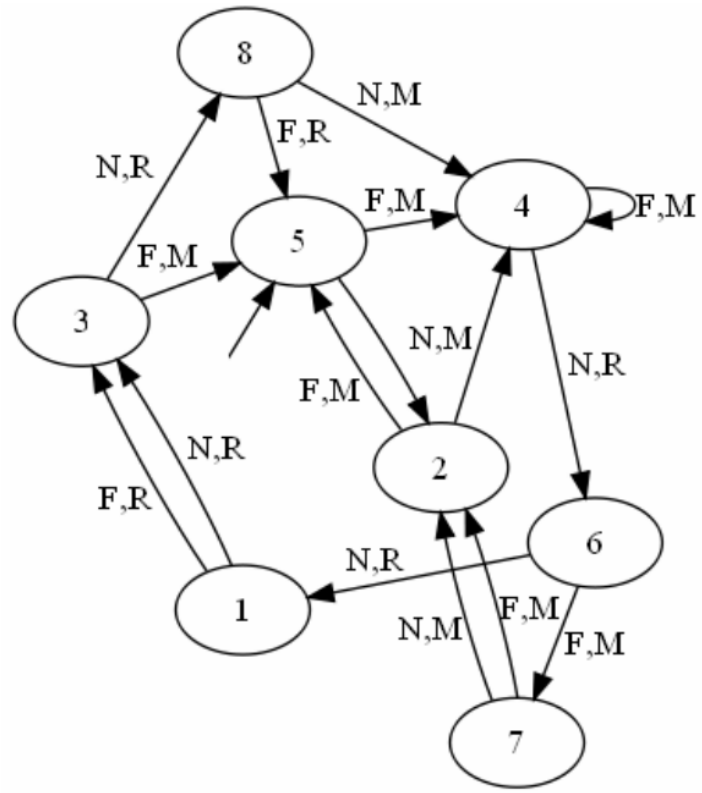

Рисунок 2 – Представлення автомата у вигляді графа переходів

Необхідно з використанням ймовірнісних алгоритмів побудувати автомат Мілі, близький до оптимального, що керує мурахою. Оптимальним вважається автомат, який має фіксоване число станів, а мураха, керована цим автоматом, з'їдає максимальне число яблук.

Результати роботи. В даній роботі для розв'язання поставленої задачі були застосовані такі ймовірнісні алгоритми: простий та клітинний генетичні алгоритми, алгоритм імітації відпалу [2]. Головну увагу було зосереджено на алгоритмі імітації відпалу та порівнянні його ефективності з генетичними алгоритмами.

Метод відпалу – техніка оптимізації, що використовує упорядкований випадковий пошук по аналогії з процесом утворення в речовині кристалічної структури при її охолодженні. Наведемо опис алгоритму імітації відпалу для поставленої задачі. На самому початку задається температура T (параметр, що визначає межу області пошуку рішення), початкове значення якої досить велике, але поступово знижується. Процес імітації починається з вибору деякого рішення задачі, для якого генерується випадковим чином автомат з діями на переходах та наперед заданою кількістю станів. Це рішення називається поточним. Для цього рішення (отриманого автомату) підраховується значення цільової функції (1). Наступним кроком є створення робочого рішення шляхом зміни поточного рішення. Зміна рішення здійснюється в такий спосіб: випадково змінюються початковий стан автомату і перехід автомату в інший стан з діями на переходах. Вибір початкового стану і переходу розігрується за допомогою випадкової величини, що має нормальний розподіл, як показано на рис.3.

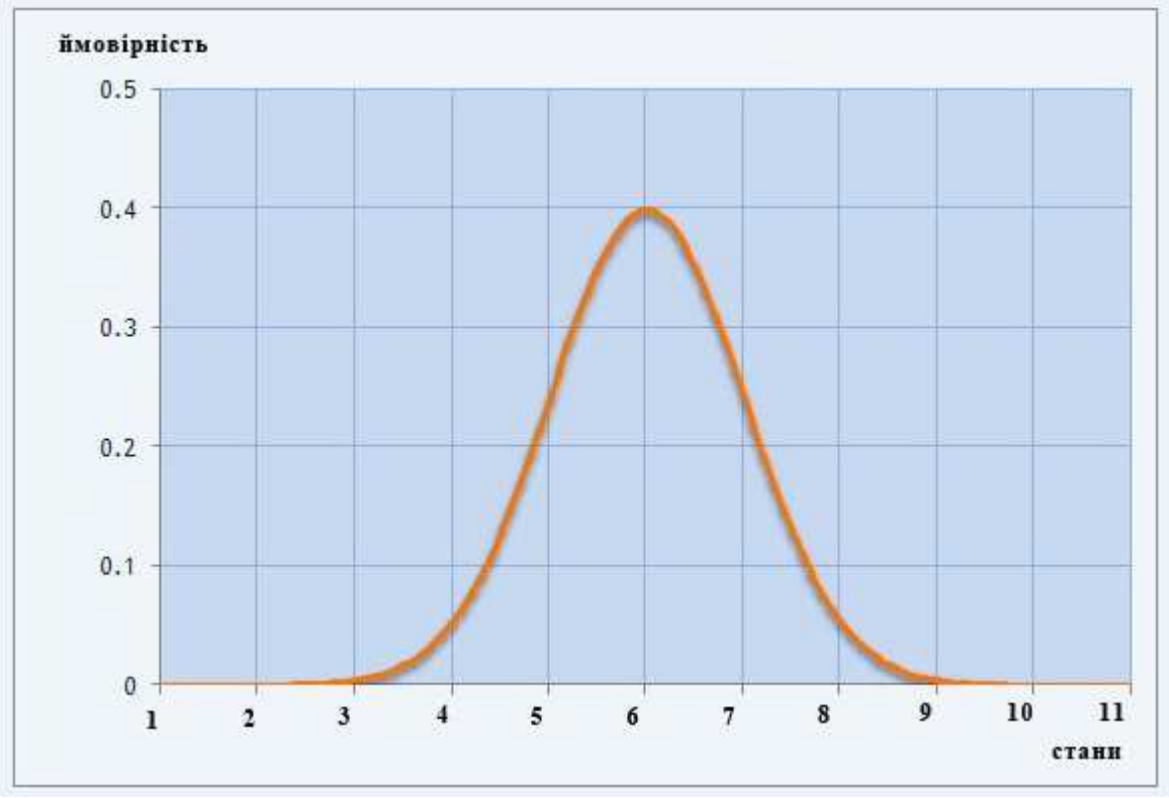

Рисунок 3 –Графік нормального розподілу

На осі x позначені стани автомата. Математичне очікування – це поточний стан автомата, а дисперсія розраховується за наступною формулою:

$$D = \frac{n}{2} T_{\text{пот}},$$

де $T_{\text{пот}}$ – значення поточної температури;

n – кількість станів автомата.

При високій температурі далекі від оптимальних рішення отримуються частіше, але здійснюється більш повний пошук рішень. Тобто при високій температурі існує більша ймовірність переходу в будь-який стан. При зниженні температури отримуємо рішення з кращим значенням цільової функції $F_{\text{пот}}$.Тобто діапазон станів, в який може перейти автомат, значно зменшується за рахунок зменшення значення дисперсії.

Для створеного робочого рішення підраховується значення цільової функції $F_{\text{роб}}$. Якщо значення цільової функції робочого рішення більше, ніж у поточного рішення, тоді робоче рішення стає поточним, а температура знижується. Якщо навпаки, тоді застосовується критерій допуску, згідно з яким випадковим чином розігрується величина в діапазоні [0;1], і якщо ця величина менша, ніж $\exp\left(\frac{F_{\text{роб}} - F_{\text{пот}}}{T_{\text{пот}}}\right)$, тоді робоче рішення стає поточним.

Ймовірність прийняття робочого рішення у вигляді умови записується наступним чином:

$$P = \begin{cases} 1, & F_{\text{роб}} \geq F_{\text{пот}}, \\ \exp\left(\frac{F_{\text{роб}} - F_{\text{пот}}}{T_{\text{пот}}}\right), & F_{\text{роб}} < F_{\text{пот}}. \end{cases}$$

Робота алгоритму продовжується доти, доки температура не стане рівною нулю, або буде отримано прийнятне рішення.

Для розв'язання поставленої задачі була розроблена програма у середовищі Micro-softVisual C# 2008 Express Edition на мові програмування C#. Проведені комп'ютерні дослідження (рис.4) показали, що генетичні алгоритми дозволяють отримати прийнятні результати досить швидко.

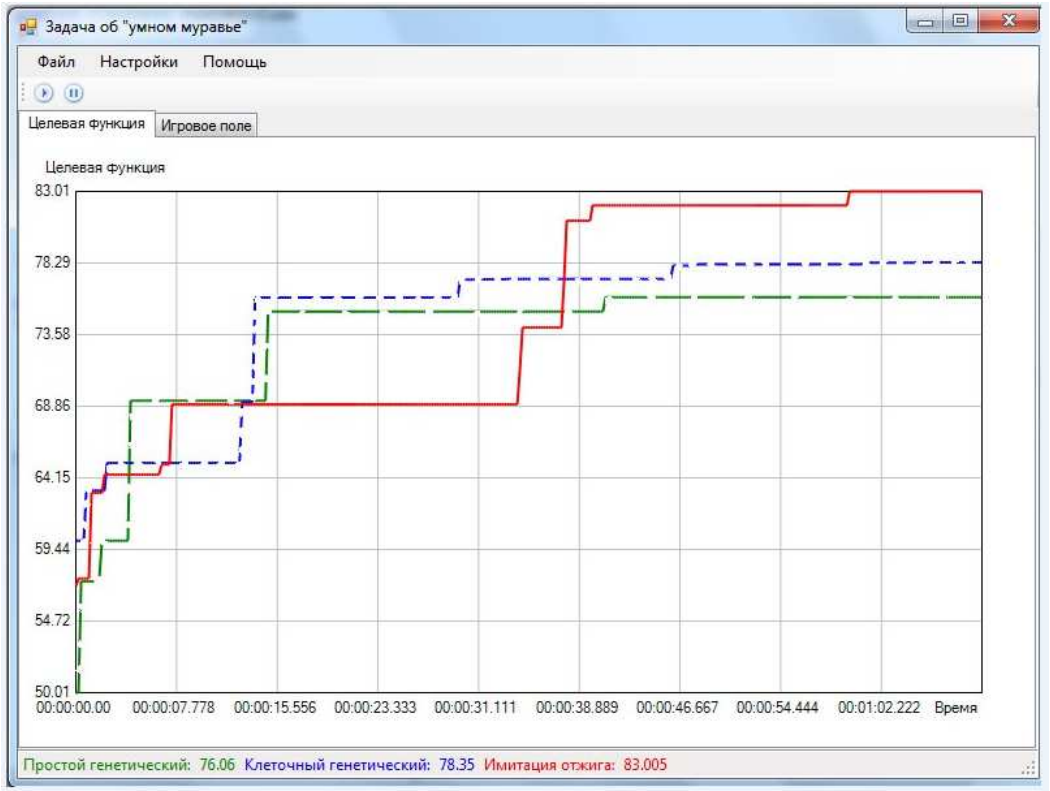

Рисунок 4 –Графіки порівнянь ймовірнісних алгоритмів

Зокрема, клітинний генетичний алгоритм на самому початку роботи, частіше за все, дає краще значення цільової функції у порівнянні з простим, але у подальшому процесі роботи в цілому генетичні алгоритми дають майже однаковий результат. Алгоритм імітації відпалу на початку процесу поступається генетичним алгоритмам, але під час подальшої роботи цей алгоритм в більшості випадків показує краще значення цільової функції, ніж генетичні алгоритми.

Висновки. Чисельні дослідження, проведені в даній роботі, показали, що алгоритм імітації відпалу приблизно на 5% ефективніший, ніж генетичні алгоритми для задачі про «Розумну мураху». В даній роботі метод відпалу був застосований та показав також свою ефективність для розв'язання деяких модифікацій задачі про «Розумну мураху», зокрема, «Мураха – 2» та «Мураха – 3».

Перевагою методу відпалу є властивість уникати "пасток" в локальних максимумах (мінімумах) функції, що оптимізується, і продовжувати пошук глобального максимуму (мінімуму). Ще однією перевагою є те, що навіть в умовах браку обчислювальних ресурсів для знаходження глобального максимуму (мінімуму) метод відпалу, як правило, дозволяє отримати досить непогане рішення. Метод відпалу і його модифікації є одним з найбільш ефективних методів випадкового пошуку оптимального рішення для великого класу задач. Результати роботи алгоритмів показали, що метод відпалу не програє генетичним алгоритмам, а для багатьох задач і виграє.

ЛІТЕРАТУРА

1. Бедный Ю.Д. Применение генетических алгоритмов для построения автоматов в задаче «Умный муравей» / Бедный Ю.Д., Шалыто А.А. // СПбГУ: ИТМО. – 2007. – [Електронний ресурс]. – Режим доступу: http://is.ifmo.ru/works/ant
2. А.С.Лопатин. Метод отжига в задачах оптимизации / А.С.Лопатин. – 2004. – [Електронний ресурс] – Режим доступу: http://www.math.spbu.ru/user/gran/students/cothesis.pdf

Надійшла до редколегії 06.06.2012.

УДК 519.8, 004.4

КАДОЧНИКОВА Я.Е., к.ф.-м.н, доцент
ЛЕОНОВ А.В., студент

Днепродзержинский государственный технический университет

ИСПОЛЬЗОВАНИЕ ВОЗМОЖНОСТЕЙ ASP.NET ДЛЯ ВИЗУАЛИЗАЦИИ РЕШЕНИЯ ОДНОЙ ЗАДАЧИ ОПТИМАЛЬНОГО РАЗБИЕНИЯ МНОЖЕСТВ

Введение. Интерес к непрерывным задачам оптимального разбиения множества (ОРМ) на непересекающиеся подмножества с целью минимизации некоторого критерия качества разбиения вызван тем, что к ним сводится достаточно широкий класс как теоретически, так и практически важных задач оптимизации, а также задач из других разделов прикладных наук. Отметим, что на протяжении последних тридцати лет Днепропетровской научной школой под руководством Е.М.Киселёвой получены существенные результаты в направлении развития теории ОРМ. Широкий обзор задач теории и практики, сводящихся к математическим моделям оптимального разбиения множеств, а также алгоритмы их решения приведены в монографии [1].