SCM-2008 Saint-Petersburg, 23-25 June 2008

Материал опубликован в сборнике докладов XI Международной конференции по мягким вычислениям и измерениям (SCM'2008). СПб.: СПбГЭТУ. 2008, с. 248-251.

МЕТОД ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ ДЛЯ ГЕНЕРАЦИИ АВТОМАТОВ, ПРЕДСТАВЛЕННЫХ ДЕРЕВЬЯМИ РЕШЕНИЙ

В. Р. Данилов, А. А. Шалыто

Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Abstract. In this paper authors present a new representation of Finite State Machines as individuals of an evolutionary algorithm. A new technology of genetic programming for automata evolving is developed.

Введение

Автоматное программирование [1] – быстроразвивающаяся парадигма программирования, использующая представление программ в виде формальной модели – автоматов. Применение генетического программирования [2] для генерации автоматов позволяет существенно сократить цикл разработки автоматных программ.

Для генерации автоматов могут применяться различные модификации эволюционных алгоритмов [3]. Однако, для большинства из этих методов размер хромосомы, требуемый для хранения автомата, экспоненциально зависит от числа входных переменных.

При решении задачи классификации плотности Дж. Козой [4] применялось представление функции переходов автомата с помощью деревьев разбора [5], представляющих булевы функции. Это представление обладает существенно большей выразительностью по сравнению с традиционными табличными методами, что позволило эффективно решить задачу. Однако этот метод может быть применен только к автоматам, имеющим два состояния. Целесообразно разработать метод эффективного представления автоматов, который может быть применен к решению более сложных задач.

Авторами предлагается способ применения деревьев решений [6] для представления автоматов с дискретными входными переменными. Производится сравнение разработанного способа с традиционными генетическими алгоритмами на примере задачи об умном муравье [7].

Представление автомата с помощью деревьев решений

Дерево решений является удобным способом задания дискретной функции, зависящей от конечного числа дискретных переменных.

Оно представляет собой помеченное дерево, метки в котором расставлены по следующему правилу:

- внутренние узлы помечены символами переменных;
- ребра – значениями переменных;
- листья – значениями искомой функции.

Для определения значения функции по значениям переменных необходимо спуститься от корня до листа, и сформировать значение, которым помечен полученный лист. При этом из вершины, помеченной переменной x переход производится по тому ребру, которое помечено тем же значением, что и значение

переменной x. На рис. 1 показано дерево, реализующее булеву функцию от переменных a, b и c.

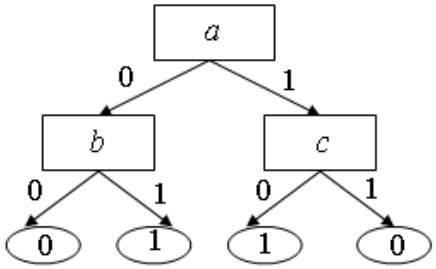

Рис. 1.Пример дерева решений

Опишем предлагаемый метод представления автомата с помощью деревьев решений. Для задания автомата необходимо выразить его функции переходов и выходов с помощью таких деревьев. Осуществим следующее преобразование автомата: вместо задания функций переходов и выходов для автомата в целом, представим эти функции для каждого состояния. Более формально это выглядит следующим образом: зададим для каждого состояния $q \in Q$ функцию $\sigma_q : X \to Q \times Y$, такую что $\sigma_q(x) = (\delta(q,x), \lambda(q,x))$ для $\forall x \in X$.

Функции σ_q соответствуют функциям переходов и выходов из состояния q. Каждая из этих функций может быть выражена с помощью дерева решений. В этих деревьях переменными являются входные переменные автомата, а множеством значений – все возможные пары (Новое Состояние, Действие). Таким образом, автомат в целом задается упорядоченным набором деревьев решений.

Генетические операции

Для использования представления автоматов в виде набора деревьев решений в генетических алгоритмах определим следующие операции:

- случайное порождение автомата – в каждом состоянии создается случайное дерево решений;
- скрещивание автоматов – скрещиваются деревья решений в соответствующих состояниях;
- мутация автомата – в случайном дереве решений производится мутация.

Здесь считается, что число состояний в автомате фиксировано. Поэтому противоречий при выполнении определенных таким образом операций не возникнет.

После определения операций над набором деревьев решений, определим генетические операции над отдельными деревьями решений. Их можно определить следующим образом:

- Случайное порождение дерева решений. При этом случайным образом выбирается метка: либо одно из возможных значений функции переходов, либо одна из переменных. После этого создается вершина, помеченная выбранной меткой. Если была выбрана переменная, то рекурсивно генерируются дети вершины, иначе вершина становится листом дерева.
- Мутация – выбирается случайный узел в поддереве. После этого поддерево, соответствующее выбранному узлу, заменяется на случайно сгенерированное (рис. 2).
- Скрещивание – в скрещиваемых деревьях выбираются два случайных узла. После этого поддерево, соответствующее выбранному узлу в первом дереве, заменяется поддеревом, соответствующим узлу второго дерева(рис. 3).

Метод генетического программирования для генерации автоматов, представленных деревьями решений

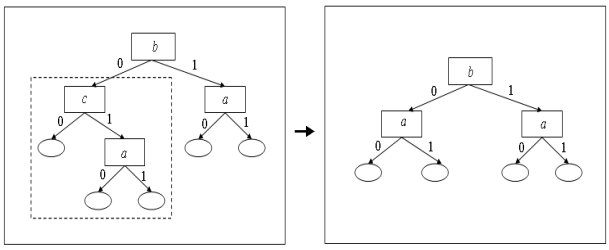

Рис. 2. Мутация деревьев решений

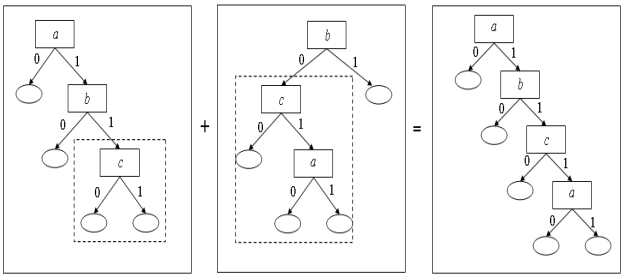

Рис. 3. Скрещивание деревьев решений

Отметим, что заданные таким образом операции могут порождать деревья, в которых некий атрибут встречается на пути от корня до листа дважды(например, дерево, полученное в результате скрещивания на рис. 3). Таким образом, необходимо ввести операцию обрезки – удаление недостижимых ветвей. Операция может быть выполнена следующим образом: узел, одна из дочерних вершин которого недостижима, заменяется достижимой дочерней вершиной. Операция обрезки должна выполняться после генетических операций скрещивания и мутации.

Апробация

Разработанный подход был протестирован на задаче «Умный муравей-2» [7]. Приведем описание задачи. Муравей находится на случайном игровом поле. Поле представляет собой тор размером 32×32 клетки. При этом муравей видит перед собой некоторую область(рис. 4).

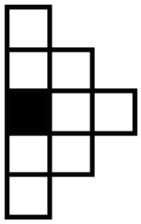

Рис. 4.Видимая муравью область

Еда в каждой клетке располагается с некоторой вероятностью μ. Значение μ является параметром задачи. Игра длится 200 ходов, за каждый из которых муравей может сделать одно из трех действий: поворот налево или направо, шаг вперед. Если после хода муравей попадает на клетку, где есть еда, то еда съедается. Целью задачи является построение стратегии поведения муравья, при которой математическое ожидание съеденной еды максимально.

Автомат управления муравьем в этой задаче имеет восемь(число видимых клеток) входных переменных – каждая из которых определяет, есть ли еда в клетке, соответствующей переменной. Все входные переменные имеют логический тип.

Предложенный подход сравнивался с генетическими алгоритмами над битовыми строками, и генетическими алгоритмами, оперирующими над таблицами переходов. Эксперимент заключался в сравнении полученных значений фитнесс-функции(объема съеденной еды) за фиксированное число шагов с одинаковыми настройками. Запуск алгоритмов производился со следующими настройками: стратегия отбора – элитизм, для размножения отбираются 25 % популяции, имеющих наибольшее значение фитнесс-функции; частота мутации – 2 %; размер популяции – 200 особей; число популяций – 100; фитнесс-функция – среднее значение съеденной еды на 200 случайных картах,

карты внутри одной популяции совпадают, карты различных популяций различны.

μ	0.02			
	Фитнесс-функция			
Число состояний	2	4	8	16
Битовые строки	7.66	8.38	7.95	6.98
Таблица переходов	6.12	7.32	7.24	7.28
Предложенный метод	7.68	8.04	7.32	8.25
μ	0.03			
	Фитнесс-функция			
Число состояний	2	4	8	16
Битовые строки	14.46	13.81	13.23	11.93
Таблица переходов	12.48	12.17	11.72	11.15
Предложенный метод	14.14	13.86	13.77	14.18
μ	0.04			
	Фитнесс-функция			
Число состояний	2	4	8	16
Битовые строки	19.11	18.68	17.47	15.10
Таблица переходов	17.18	15.94	15.03	13.68
Предложенный метод	18.28	20.28	18.60	20.18

Таблица 1.Результаты экспериментов

Результаты эксперимента приведены в табл. 1. Последнее измерение фитнесс-функции осуществлялось на случайном наборе из 2000 карт.

Анализ показывает, что в случаях, когда важны значения почти всех предикатов (при $\mu = 0.01$), предлагаемый метод работает хуже известных. Это можно объяснить тем, что искомые автоматы плохо описываются деревьями решений. Однако, при больших значениях μ предложенный метод работает лучше, особенно при большом числе состояний. Таким образом, выяснено, что метод работает лучше известных в большинстве случаев, за исключением ситуаций, когда функция переходов не может быть эффективно выражена деревом решений.

Литература

1. Шалыто А.А. Технология автоматного программирования //Труды первой Всероссийской научной конференции «Методы и средства обработки информации» – МГУ, 2003 – С. 528–535. http://is.ifmo.ru/works/tech_aut_prog/
2. Koza J. Genetic programming: On the Programming of Computers by Means of Natural Selection – MA: The MIT Press, 1992. – 849 c.
3. Гладков Л. А, Курейчик В. В., Курейчик В. М. Генетические алгоритмы. М.: Физматлит, 2006. – 319 с.
4. Andre D., Bennet F., Koza J. Discovery by Genetic Programming of a Cellular Automata Rule that is Better than any Known Rule for the Majority Classification Problem. http://citeseer.ist.psu.edu/33008.html
5. Хопкрофт Дж., Мотвани Р., Ульман Дж. Введение в теорию автоматов, языков и вычислений. – 2-е изд. – М.: Вильямс, 2002. – 528 с.
6. Шеннон К. Работы по теории информации и кибернетике. М.: Иностранная литература, 1963. – 836 с.
7. Бедный Ю.Д., Шалыто А.А. Применение генетических алгоритмов для построения автоматов в задаче «Умный муравей». – СПбГУ ИТМО,2007.http://is.ifmo.ru/works/ant