μ	0,04			
	Функция приспособленности			
Число состояний	2	4	8	16
Полные таблицы	17,18	15,94	15,03	13,68
Деревья решений	18,28	20,28	18,60	20,18
Предложенный метод	18,82	16,44	19,75	15,46

Таблица. Результаты экспериментов

Анализ результатов показывает, что предложенный метод является более эффективным по сравнению с представлением функции переходов полными таблицами. При некоторых значениях числа состояний предложенный метод более эффективен по сравнению с методом представления функции переходов деревьями решений.

Исследование выполнено по Федеральной целевой программе «Научные и научно-педагогические кадры инновационной России на 2009–2013 годы» в рамках государственного контракта П1188 от 27 августа 2009 года.

Литература

1. Koza J.R. Genetic programming: On the Programming of Computers by Means of Natural Selection. – Cambridge: MIT Press, 1992.
2. Гладков Л.А, Курейчик В.В., Курейчик В.М. Генетические алгоритмы. – М.: Физматлит, 2006.
3. Поликарпова Н.И., Точилин В.Н., Шалыто А.А. Применение генетического программирования для генерации автоматов с большим числом входных переменных // Научно-технический вестник СПбГУ ИТМО. – 2008. – № 53. – С. 24–42.
4. Данилов В.Р. Метод представления автоматов деревьями решений для использования в генетическом программировании // Научно-технический вестник СПбГУ ИТМО. – 2008. – № 53. – С. 103–108.
5. Шалыто А.А. Логическое управление. Методы аппаратной и программной реализации. – СПб: Наука, 2000. – 780 с.
6. Bryant R.E. Graph-based algorithms for boolean function manipulation // IEEE Transactions on Computers. – 1986. – № 8. – P. 677–691.
7. Бедный Ю.Д., Шалыто А.А. Применение генетических алгоритмов для построения автоматов в задаче «Умный муравей». – СПбГУ ИТМО, 2007 [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/works/ant, свободный. Яз. рус. (дата обращения 09.02.2011).

Данилов Владимир Рюрикович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, аспирант, Vladimir.Daniloff@gmail.com

Шалыто Анатолий Абрамович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, доктор технических наук, профессор, зав. кафедрой, shalyto@mail.ifmo.ru

УДК 004.4’242

АВТОМАТИЧЕСКИЙ ПОДБОР ПАРАМЕТРОВ ВНЕШНЕЙ СРЕДЫ ПРИ ГЕНЕРАЦИИ АВТОМАТНЫХ ПРОГРАММ С ПОМОЩЬЮ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

В.А. Кулев

Предлагается новый метод генетического программирования, позволяющий улучшить способ построения функции приспособленности для генетических алгоритмов, проводящих тестирование особей путем симуляции окружающей среды. Метод позволяет автоматически подбирать параметры внешней среды, что приводит к улучшению качества генерируемых особей. В ходе исследований на примере задачи об «Итерированной дилемме узника» было показано, что разработанный подход применим для генерации автоматных программ в задачах, где поведение окружающей среды может быть описано с помощью конечного автомата.

Ключевые слова: генетическое программирование, конечный автомат, автоматное программирование.

Введение

Вычисление функции приспособленности на основе симуляции внешней среды получило наибольшее распространение благодаря своей очевидности и простоте. В некоторых ситуациях этот подход также представляется единственным возможным, при этом множество параметров среды, которое используется для вычисления функции приспособленности, намного меньше множества возможных параметров среды. Это множество, как правило, задается априори, исходя из эвристических соображений.

Целью данной работы является создание метода, позволяющего охватить большую часть пространства возможных параметров внешней среды, при этом сохраняя приемлемую скорость работы генетического алгоритма.

При применении автоматов в программировании, как правило, используются не абстрактные модели автоматов, а модель автоматизированного объекта [1], объединяющая управляющий автомат и объект управления.

Важная черта устройств управления всех абстрактных машин – их конечность. Управляющий автомат не только имеет конечное число состояний, но, кроме того, реализуемые им функции переходов и выходов оперируют исключительно конечными множествами. Именно это свойство позволяет описывать логику поведения машины явно – в виде таблицы или графа переходов. Поэтому свойство конечности устройства управления необходимо сохранить при построении модели автоматизированного объекта. Более того, это свойство целесообразно усилить следующим неформальным требованием: число управляющих состояний, входных и выходных воздействий должно быть небольшим (обозримым). Управляющий автомат с тысячей состояний, безусловно, является конечным, однако изобразить его граф переходов практически невозможно, что сводит на нет преимущества явного выделения управляющих состояний.

Напротив, число состояний объекта управления может быть сколь угодно большим (при переходе от модели к программной реализации оно с необходимостью станет конечным, однако может остаться необозримым). В процессе работы управляющему автомату требуется получать информацию о состоянии объекта управления и изменять его. Однако в силу свойства конечности автомат не может напрямую считывать и записывать это состояние. Для этого и требуются операции объекта управления. Небольшое число запросов, возвращающих конечные значения, позволяет автомату получать информацию о состояниях объекта управления, которую он способен обработать. Небольшое число команд используется для «косвенного» изменения состояний объекта управления.

На рис. 1 сплошными стрелками обозначены типичные для программных реализаций виды взаимодействия между автоматом, объектом управления и внешней средой. Автомат получает входные воздействия как со стороны среды, так и от объекта управления.

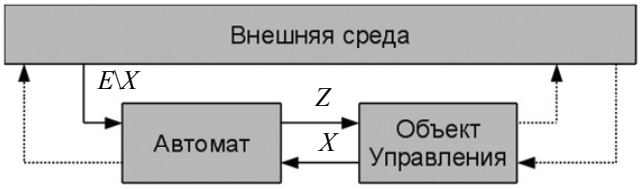

Рис. 1. Модель автоматизированного объекта

Пунктирными стрелками обозначены менее распространенные, хотя и возможные, варианты взаимодействия. Автомат может оказывать выходное воздействие и на внешнюю среду. Однако таких связей обычно можно избежать, включив все управляемые автоматом сущности в состав его объекта управления. Отметим, что в программировании, в общем случае, различие между объектом управления и внешней средой носит скорее концептуальный, а не формальный характер. Создавая модель системы со сложным поведением, разработчик производит ее декомпозицию на автоматизированные объекты, определяя тем самым объект управления каждого автомата. В целях минимизации связей между модулями программной системы целесообразно проводить декомпозицию таким образом, чтобы автомат оказывал выходные воздействия только на собственный объект управления. Кроме того, объект управления может взаимодействовать с внешней средой напрямую.

Предлагаемый метод

В рамках данной работы рассматривается ситуация, в которой поведение внешней среды автоматизированного объекта может быть представлено в виде конечного автомата. В этом случае внешняя среда сама может рассматриваться как автоматизированный объект, что порождает схему, изображенную на рис. 2.

Отметим, что полученная модель очень удобна ввиду естественности взаимодействия ее компонент. Обмен информацией между автоматизированным объектом и внешней средой ведется через входные и выходные воздействия управляющих автоматов, а также через взаимодействие объектов управления.

Также можно выделить класс задач, в которых подразумевается наличие нескольких автоматных программ с одинаковыми наборами входных и выходных воздействий, играющих в некую игру. В этом случае для каждого из игроков внешней средой будут являться его оппоненты, а входными воздействиями управляющего его автомата – выходные воздействия автоматов оппонентов, или их функция $f_z : Z^{N-1} \to X$, где N – число игроков; Z – множество выходных воздействий; X – множество входных воздействий.

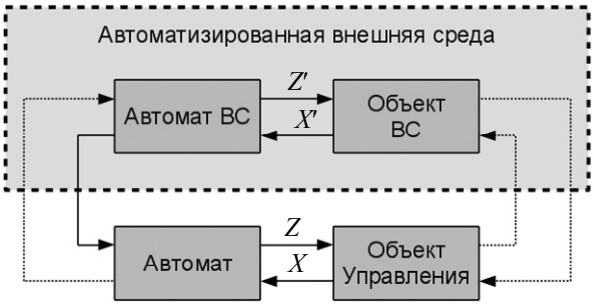

Рис. 2. Модель с автоматизированной внешней средой

Пусть задана функция приспособленности $f_t(i,e)$ для особи $i \in P_t$ и управляющего автомата внешней среды $e \in P_{env}$. Если известно максимальное возможное значение функции f_t для всех i и e, $f_{\max} = \max_{(i,e)}[f_t(i,e)]$, то уровень отклонения особи i от оптимального поведения можно оценить как $f_{inv}(i,e) = f_{\max} - f_t(i,e)$. Тогда качество теста с автоматом e для всей популяции P_t может вычисляться как среднее значение: $f_{inv}(e) = \dfrac{\sum_{i \in P_t} f_{inv}(i,e)}{|P_t|}$.

Однако подбор одиночного управляющего автомата внешней среды не является эффективным способом всестороннего тестирования целевых особей. В связи с этим предлагается хранить множество таких автоматов, которые будут образовывать популяцию особей внешней среды P_{env}. К этой популяции можно применить метод генетического программирования, используя функцию $f_{inv}(e)$ в качестве функции приспособленности. Также необходимо определить модифицированную функцию приспособленности для особей из популяции P_t, вычисляемую как среднее значение: $f_t(i) = \dfrac{\sum_{e \in P_{env}} f_t(i,e)}{|P_{env}|}$.

Если использовать генетический алгоритм для популяции P_{env} как составную часть основного алгоритма, то получится так называемый «мета-генетический» алгоритм [2]. Однако этот алгоритм работает достаточно длительное время. Для описываемой задачи более эффективно работает модификация классического алгоритма, в которой эволюция обеих популяций производится параллельно.

Сформулируем этапы модифицированного генетического алгоритма, автоматически подбирающего управляющие автоматы внешней среды.

1. **Инициализация.** Создаются две случайные начальные популяции целевых автоматов и управляющих автоматов внешней среды с соответствующими входными и выходными воздействиями. Размер популяций и параметры генерации случайных особей могут различаться.
2. **Тестирование.** На этапе тестирования происходит вычисление функций приспособленности $f_t(i)$ и $f_{inv}(e)$ для всех особей $i \in P_t$ и $e \in P_{env}$. Для этого производится моделирование поведения каждого целевого автоматизированного объекта i в каждой автоматизированной внешней среде e, результатом которой является функция $f_t(i,e)$. Далее искомые значения функций приспособленности вычисляются по приведенным выше формулам.
3. **Отбор.** Выбор особей для проведения генетических операций и формирования нового поколения производится независимо для каждой из популяций, поэтому алгоритм отбора не требует модификации.
4. **Кроссовер и мутация.** Способ проведения кроссоверов и мутаций также не отличается от классического алгоритма, однако можно сформулировать рекомендации по настройке его параметров. Как правило, при применении оператора мутации задается ее «сила», определяющая, насколько большая доля хромосомы будет подвергнута случайной модификации. Данный параметр, в совокупности с общим количеством производимых мутаций, определяет, насколько быстро изменяется генотип популяции в ходе эволюционного процесса. Для обеспечения большей стабильности работы модифицированного генетического алгоритма предлагается устанавливать этот параметр для популяции управляющих автоматов внешней среды ниже, чем для популяции целевых автоматов.

Принципиальная схема модифицированного генетического алгоритма изображена на рис. 3. На ней явно обозначены части алгоритма, выполняющиеся независимо для обеих популяций, и точка, в которой происходит их взаимодействие.

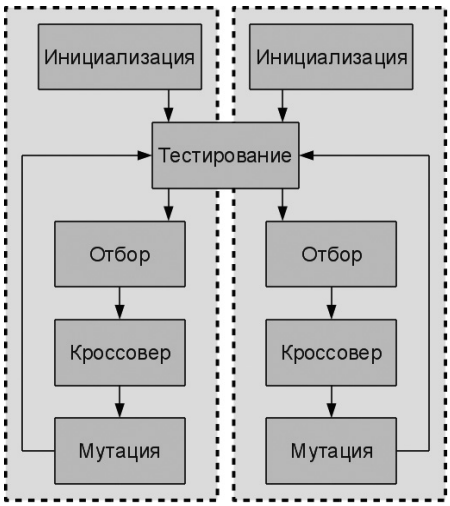

Рис. 3. Схема модифицированного генетического алгоритма

Экспериментальное исследование

Экспериментальное исследование построенного алгоритма было проведено на примере некооперативной матричной игры с ненулевой суммой, обычно называемой «Итерированная дилемма узника» («Iterated prisoner's dilemma») [3].

В статье [4] описывается эксперимент с применением генетического алгоритма для генерации стратегий управления узником. В качестве хромосомы использовалась битовая строка, кодирующая управляющую функцию $\{S, P, R, T\}^3 \to \{Cooperate, Defect\}$. На вход функции предоставляются исходы трех предыдущих раундов игры, которые могут быть следующими:

– $S(Sucker)$ – игрок отрицал, когда другой сознался;
– $P(Punishment)$ – оба игрока сознались;
– $R(Reward)$ – оба игрока отрицали;
– $T(Temptation)$ – игрок сознался, когда другой отрицал.

Функция приспособленности особей вычислялась путем турнира между особями популяции. Стратегии, полученные в результате работы построенного алгоритма, имели поведение, близкое к стратегии «Око за око». В турнире многие особи набирали даже большее количество баллов, чем «Око за око», за счет того, что они успешнее противостояли особям из той же популяции.

Эксперимент для изучения работы модифицированного алгоритма был проведен в двух вариациях. Общими настройками алгоритма были:

– размер популяции – 300;
– число состояний автомата – 5;
– вероятность скрещивания – 0,5;
– вероятность мутации – 0,5;
– доля элитизма – 0,125.

Для оценки получаемого результата также подсчитывался выигрыш лучшей особи из популяции против стратегии «Око за око».

В первом эксперименте функция приспособленности для обеих популяций вычислялась по одинаковому алгоритму, который далее будем называть «прямым». В этом случае приспособленность особи определяется как ее средний выигрыш при игре со всеми особями из противоположной популяции. Графики, отражающие ход эксперимента, приведены на рис. 4 и 5.

Три коррелирующие линии (1, 2 и 4) обозначают соответственно минимальное, максимальное и среднее значение функции приспособленности в популяции. Линия 3 обозначает приспособленность лучшей особи из популяции пре игра против стратегии «Око за око».

Во втором эксперименте функция приспособленности одной из популяций была заменена на обратную, которая вычисляется не как средний выигрыш игрока, а как средний проигрыш оппонента.

Таким образом, одна из популяций берет на себя четко обозначенную роль контроля качества особей из противоположной популяции. По графикам на рис. 6 и 7, полученным в ходе эксперимента, можно судить о пользе такого подхода, так как получаемые решения лучше ведут себя в игре против неизвестной им стратегии «Око за око».

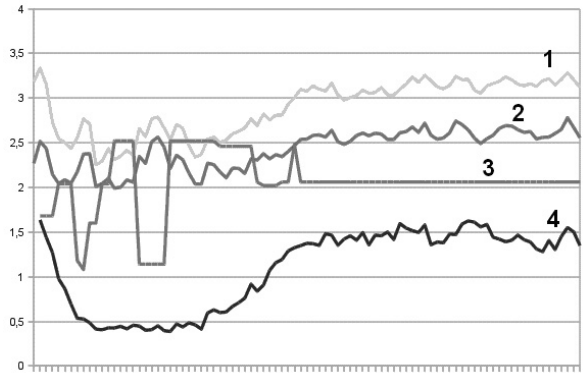

Рис. 4. Показания первой популяции с симметричной функцией приспособленности

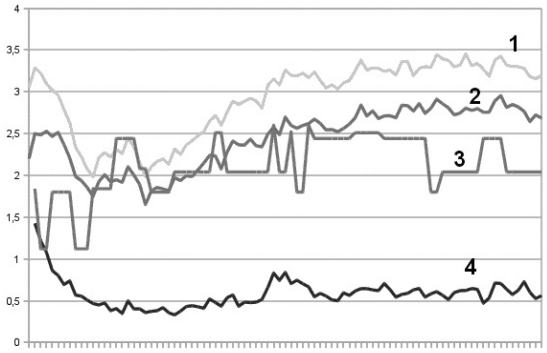

Рис. 5. Показания второй популяции с симметричной функцией приспособленности

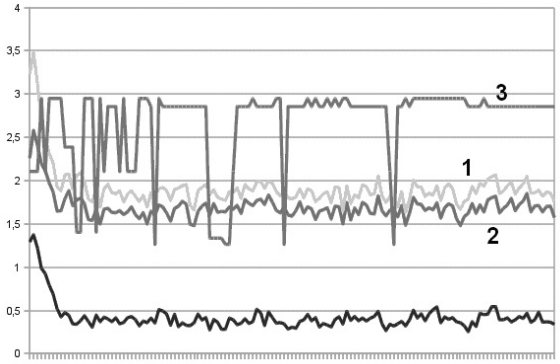

Рис. 6. Показания популяции с «прямой» функцией приспособленности

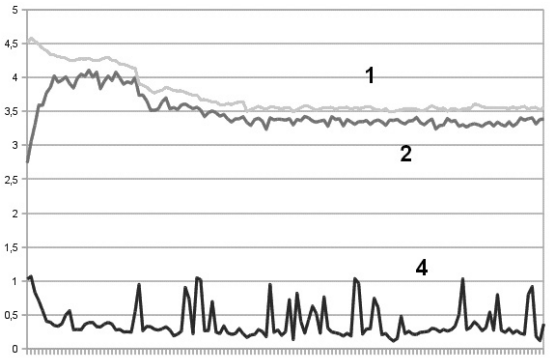

Рис. 7. Показания популяции с «обратной» функцией приспособленности

Заключение

Таким образом, применение модифицированного генетического алгоритма к задаче об «итерированной дилемме узника» позволило получить особей, способных успешно соревноваться не только со своими собратьями, но и с неизвестными им противниками, что было показано на примере стратегии «Око за око». Исходя из этих результатов, можно сделать вывод о том, что предложенный метод является эффективным решением проблемы, описанной в начале статьи.

Литература

1. Поликарпова Н.И., Точилин В.Н., Шалыто А.А. Разработка библиотеки для генерации автоматов методом генетического программирования // Сборник докладов X международной конференции по мягким вычислениям и измерениям. – СПб: СПбГЭТУ, 2007. – Т. 2. – С. 84–87.
2. Bäck T. Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. – Oxford University Press, 1996.
3. Axelrod R., Hamilton W.D. The evolution of cooperation // Science. – 1981. – V. 211. – № 4489. – P. 1390.
4. Axelrod R. The evolution of strategies in the iterated prisoner's dilemma. – Cambridge university press, 1987. – P. 1–16.

Кулев Владимир Анатольевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, аспирант, me@lightoze.net

УДК 004.4'242

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО ПОДХОДА ДЛЯ ГЕНЕРАЦИИ КЛЕТОЧНЫХ АВТОМАТОВ

А.В. Тихомиров, А.А. Шалыто

Рассматривается метод генерации произвольных клеточных автоматов на основе тестовых наборов при помощи генетических алгоритмов. Описаны основные проблемы при использовании стандартного генетического алгоритма для решения поставленной задачи. Предложены модифицированные генетические операторы для устранения данных недостатков. Произведена апробация на нескольких обучающих примерах.
Ключевые слова: клеточные автоматы, генетические алгоритмы.

Введение

В настоящее время широко распространено использование клеточных автоматов для симуляции многих физических процессов, таких как диффузия энергии, разнообразные химические реакции, рост кристаллов и т.д. [1–4]. Однако использование клеточных автоматов затрудняется тогда, когда физическая система известна, а описывающий ее клеточный автомат – нет, или построение его обычными эвристическими методами затруднительно, так как автомат может иметь большое число состояний, переходов, условий и действий на переход [5].

Цель настоящей работы – устранить этот недостаток, используя генетическое программирование с использованием обучающих наборов.

Постановка задачи

Задана двумерная плоскость определенной размерности с координатной сеткой, которая делит плоскость на квадраты [6, 7]. При этом задаются данные для различных временных шагов, т.е. состояние системы в начале, несколько промежуточных состояний и конечное состояние системы.

Каждая ячейка плоскости управляется одинаковым автоматом.

Цель поставленной задачи – вырастить клеточный автомат, который наиболее точно описывает заданную физическую систему.

Строение хромосомы клеточного автомата

Обычно при использовании генетического подхода используется подход, при котором данные в хромосоме хранятся в виде строки [5]. Однако в рассматриваемой задаче длина хромосомы может варьироваться в зависимости от количества переходов между состояниями. Для более удобного описания данных ген представляется не строкой, а специализированной структурой, что отличается от стандартного метода генетического программирования. При этом некоторые виды генов являются составными, т.е. содержат другие, более простые составляющие.

Количество базовых генов соответствует количеству состояний в клеточном автомате. Таким образом, базовый ген под названием «StateGene» описывает определенное состояние автомата. «StateGene» состоит из генов переходов клеточного автомата, которые называются «TransitionGene» (рис. 1).