3. Kim C., Stern R.M. Robust Signal-to-Noise Ratio Estimation Based on Waveform Amplitude Distribution Analysis // Proc. INTERSPEECH-2008. – Brisbane, Australia, 2008. – P. 2598–2601.
4. Nemer E., Goubran R., Mahmoud S. SNR Estimation of Speech signals Using Subbands and Fourth-Order Statistics // IEEE Signal Processing Letters. – 1999. – V. 6. – № 7. – P. 171–174.
5. Hirsch H.G., Ehrlicher C. Noise estimation techniques for robust speech recognition // Proc. ICASSP. – Detroit, Michigan, 1995. – V. 1. – P. 153–156.
6. Hergolz C., Jeub M., Nelke C., Beaugeant C., Vary P. Evaluation of Single- and Dual-channel Noise Power Spectral Density Estimation Algorithms for Mobil Phones // Proc. Konferenz Elektronische Sprachsignalverarbeitung (EESV). – Aachen, Germany, 2011. – P. 1–10.

Столбов Михаил Борисович – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, ООО «ЦРТ-инновации», кандидат технических наук, ст. научный сотрудник, доцент, stolbov@speechpro.com

УДК 004.4'242

МЕТОД ПОСТРОЕНИЯ УПРАВЛЯЮЩИХ АВТОМАТОВ НА ОСНОВЕ МУРАВЬИНЫХ АЛГОРИТМОВ

Д.С. Чивилихин, В.И. Ульянцев

Предлагается метод построения управляющих конечных автоматов по выбранной функции приспособленности, основанный на применении муравьиных алгоритмов. Проводится его апробация на задаче об «Умном муравье». Показано, что метод обладает большей производительностью по сравнению с традиционным подходом к построению управляющих автоматов на основе генетических алгоритмов. Предлагаемый метод не требует реализации таких нетривиальных операторов, как, например, скрещивание или мутация.
Ключевые слова: конечные автоматы, муравьиные алгоритмы.

Введение

В последние годы для решения разнообразных задач все чаще применяется автоматное программирование [1]. В рамках этого подхода поведение программ описывается с помощью детерминированных конечных автоматов. При этом часто эвристическое построение автоматов затруднено, поэтому для этой цели применяются различные эволюционные алгоритмы, такие как генетические алгоритмы [2–5] и эволюционные стратегии [2, 6].

При построении автоматов с помощью эволюционных алгоритмов вводится функция, называемая функцией приспособленности, отражающая то, насколько автомат близок к решению задачи. Таким образом, построение автомата для той или иной задачи сводится к построению автомата с наибольшим или наименьшим значением функции приспособленности. Отдельно отметим, что введение функции приспособленности не предполагает наличие эталонного автомата. Примерами задач, для которых можно найти решение, представленное управляющим автоматом, с помощью эволюционных алгоритмов, являются:
- задача об «Умном муравье» [1, 3, 7];
- задача о завоевании ресурсов [2];
- задача о построении автомата на основе тестовых примеров [3, 8].

В начале работы эволюционный алгоритм генерирует некую начальную популяцию особей (в данном случае – конечных автоматов). Обычно особи начальной популяции генерируются случайным образом. Далее, пока не будет выполнено условие останова, выполняются операции мутации, скрещивания и селекции. Операции мутации и скрещивания по некоторым правилам изменяют особи в текущей популяции. Операция селекции определяет, какие особи перейдут в следующую популяцию.

При построении конечных автоматов с помощью эволюционных алгоритмов требуется разрабатывать операции мутации и скрещивания. В настоящей работе предлагается новый метод построения управляющих автоматов, основанный на муравьином алгоритме [9], не требующий реализации обозначенных операций.

Вопрос сходимости муравьиных алгоритмов изучен на данный момент сравнительно слабо. Доказательство сходимости некоторых классов муравьиных алгоритмов можно найти в [10]. Вопрос о сходимости других типов муравьиных алгоритмов остается на данный момент открытым. Сходимость предложенного алгоритма построения автоматов в настоящей работе не изучается.

Муравьиные алгоритмы

Муравьиные алгоритмы – группа алгоритмов оптимизации на графах, принцип работы которых основан на поведении муравьев, ищущих путь от муравейника до источника пищи. Примером задачи, которая может быть решена с помощью муравьиного алгоритма, является задача о коммивояжере [11].

В муравьиных алгоритмах решения строятся набором агентов-муравьев, использующих при выборе пути в графе некоторую стохастическую стратегию. Решения могут быть представлены как путями в графе, так и отдельными его вершинами. Каждому ребру графа (u, v) (здесь u и v – вершины графа) сопоставлено некоторое действительное число τ_{uv}, называемое значением феромона, и (необязательно) некоторое эвристическое расстояние η_{uv}. Значения феромона модифицируются муравьями в процессе поиска, в то время как эвристические расстояния задаются перед началом работы алгоритма и далее не изменяются.

Можно выделить три основных этапа работы муравьиного алгоритма, которые повторяются, пока не будет найдено допустимое решение или пока не выполнится условие останова. На первом этапе, назовем его `ConstructSolutions`, каждый муравей совершает переходы по ребрам графа, запоминая свой путь – последовательность ребер. Находясь в определенной вершине, каждый муравей определяет следующее ребро, исходя из эвристических расстояний и значений феромона на ребрах, инцидентных данной вершине. Определив ребро, муравей добавляет его к своему пути и переходит по этому ребру к следующей вершине.

На втором этапе – `UpdatePheromones` – значения феромона на всех ребрах графа изменяются. Значение феромона на ребре может увеличиться, если по этому ребру прошел муравей, или уменьшиться вследствие испарения. При испарении значения феромона на всех ребрах графа изменяются в заданное число раз. На третьем (необязательном) этапе – `DaemonActions` – применяется некоторая локальная оптимизация или другие действия, которые не могут быть выполнены отдельными муравьями.

Управляющие конечные автоматы

Будем называть управляющим детерминированный конечный автомат (ДКА), на каждом переходе которого записаны событие и последовательность выходных воздействий. События поступают на вход автомата из внешней среды. При поступлении события автомат переходит в новое состояние и передает выходные воздействия объекту управления. Пример управляющего конечного автомата приведен на рис. 1.

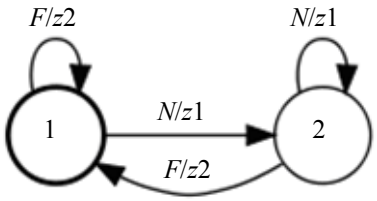

Рис. 1. Пример управляющего конечного автомата

Для данного автомата множество входных событий равно {F, N}, а множество выходных воздействий – {$z1$, $z2$}. Стартовое состояние автомата на рис. 1 и на всех других рисунках в данной работе отмечено жирной линией.

Предлагаемый метод построения управляющих автоматов на основе муравьиных алгоритмов

Рассмотрим задачу о построении управляющего конечного автомата [1] при фиксированном множестве входных событий, выходных воздействий и введенной функции приспособленности. Пусть задано число состояний N, множество входных событий *Events* и множество выходных воздействий *Actions* автомата. Задача состоит в поиске автомата с наибольшим значением функции приспособленности.

Рассмотрим представление конечного автомата в виде графа переходов. При этом подходе состояния автомата соответствуют вершинам графа, а переходы – его дугам. Рассмотрим множество всех автоматов с параметрами (N, *Events*, *Actions*). Любой автомат из выбранного множества как граф является подграфом полного недетерминированного конечного автомата (НКА) с теми же параметрами. Под полным НКА понимается автомат, в котором из каждого из N состояний по каждому входному событию из *Events* существует переход в каждое состояние с каждым выходным воздействием из *Actions*. Пример такого автомата с двумя состояниями приведен на рис. 2.

Задача оптимизации ДКА для муравьиного алгоритма может быть сформулирована следующим образом. Задано множество входных событий *Events*, множество выходных воздействий *Actions* и число состояний N. Также задана функция приспособленности, определенная на множестве всех ДКА с указанными параметрами, сопоставляющая автомату действительное число. На полном НКА с теми же параметрами требуется найти такой путь, чтобы ДКА, составленный из ребер и вершин найденного пути, имел наибольшую функцию приспособленности.

В начале работы алгоритма всем ребрам полного НКА приписывается некоторый одинаковый ненулевой вес – значение феромона в терминах муравьиных алгоритмов. В каждую вершину полного НКА помещается по муравью. Каждый муравей выбирает переходы из своего состояния по каждому входному

символу, исходя из значений феромона на переходах. Выбор производится с помощью метода «рулетки» [11]. Заметим, что, в отличие от классического муравьиного алгоритма, описанного в разделе «Муравьиные алгоритмы», каждый муравей строит переходы только из своего состояния, не перемещаясь в другие состояния (вершины).

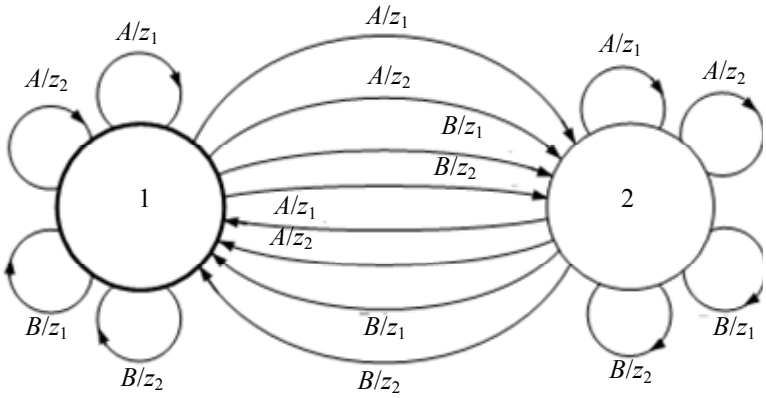

Рис. 2. Пример полного недетерминированного конечного автомата

По выбранным муравьями состояниям и переходам строится ДКА. Далее для ДКА вычисляется выбранная функция приспособленности. При этом запоминаются переходы, которые совершал автомат. К весам посещенных при вычислении функции приспособленности переходов НКА добавляются дополнительные веса, пропорциональные вычисленному значению. Отметим, что веса тех переходов, которые не были посещены при вычислении функции приспособленности, не изменяются, а сами переходы удаляются из ДКА. Также из ДКА удаляются непосещенные состояния.

На следующем этапе со всех ребер НКА «испаряется» феромон – веса всех ребер уменьшаются в одинаковое число раз.

Наконец, проверяются условия сходимости (достижения требуемого значения функции приспособленности) и стагнации. Под стагнацией понимается ситуация, при которой значение функции приспособленности лучшего автомата не увеличивается в течение некоторого фиксированного промежутка времени (числа шагов алгоритма). В таком случае алгоритм перезапускается. При достижении требуемого значения функции приспособленности алгоритм прекращает свою работу, выдавая лучший из построенных автоматов.

Экспериментальное исследование

В рамках экспериментального исследования предлагаемый метод был протестирован на задаче об «Умном муравье» [1, 4, 7]. Для вычислений использовался персональный компьютер с процессором Intel Core i7 2600 с тактовой частотой 3,4 ГГц.

Опишем задачу об «Умном муравье». Задано квадратное тороидальное поле размером 32×32 клетки. На поле вдоль заданной ломаной расположено 89 «яблок» – единиц еды, причем яблоки располагаются не в каждой клетке этой ломаной. В начале пути муравей находится в левой верхней клетке поля и «смотрит» на восток. Поле, начальное расположение муравья и расположение еды на поле показано на рис. 3. Белые клетки пусты, в черных клетках расположена еда, а серым цветом отмечены клетки ломаной, не содержащие еды.

Муравей видит одну клетку перед собой: он может определить, есть еда в следующей клетке или нет. У муравья есть 200 ходов. На каждом ходу он может выполнить одно из трех действий: повернуть налево, повернуть направо, перейти на одну клетку вперед. При этом если клетка, в которую перешел муравей, содержит еду, муравей съедает эту еду.

В данной задаче рассматриваются два входных события – **N** (в следующей клетке нет еды) и **F** (в следующей клетке есть еда) – а также три выходных воздействия: **L** (повернуть налево), **R** (повернуть направо) и **M** (перейти на одну клетку вперед).

Задача состоит в том, чтобы построить систему управления муравьем, которая позволит ему съесть всю еду за минимальное число шагов. В работе [3] предлагается решение данной задачи путем построения конечного автомата, управляющего муравьем, с помощью генетического программирования. В указанной работе был найден автомат, имеющий семь состояний и позволяющий муравью съесть всю еду за отведенное число шагов. Число вычислений функции приспособленности, потребовавшихся для построения такого автомата – порядка сотен миллионов ($160 \cdot 10^6$ и $250 \cdot 10^6$ для двух разных экспериментов).

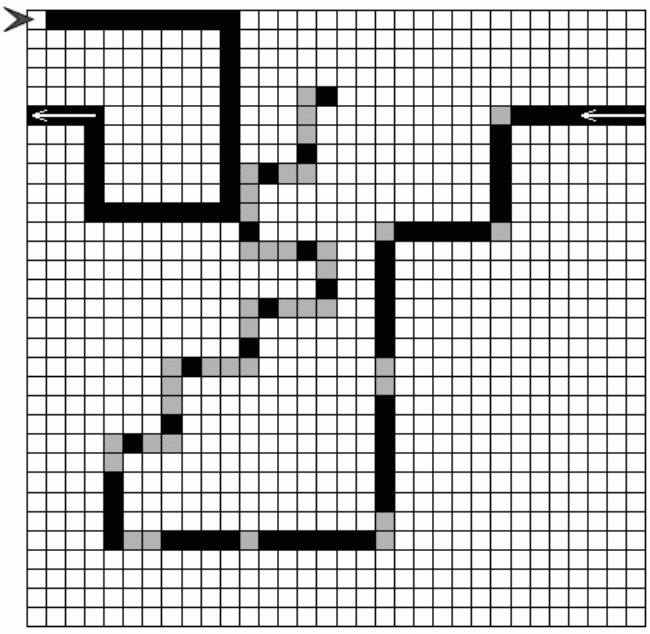

Рис. 3. Поле, начальное расположение муравья и еды в задаче об «Умном муравье»

Традиционно функция приспособленности автомата, управляющего умным муравьем, имеет вид [1]

$$f(fsm) = n + \frac{200 - numberOfSteps}{200}, \quad (1)$$

где *fsm* – конечный автомат; *n* – число съеденных яблок, а *numberOfSteps* – номер хода, на котором было съедено последнее яблоко.

При использовании функции приспособленности (1) предлагаемый муравьиный алгоритм не нашел оптимального решения задачи об «Умном муравье». В настоящей работе авторами предлагается использовать модифицированную функцию приспособленности, учитывающую число состояний автомата:

$$f(fsm) = n + \frac{200 - numberOfSteps}{200} + 0{,}1 \cdot (M - N), \quad (2)$$

где M – число состояний НКА, в котором проводится поиск с помощью муравьиного алгоритма, а N – число состояний ДКА, посещенных при вычислении функции приспособленности. При использовании выражения (2) оптимизация проводится одновременно по числу съеденных яблок, числу потребовавшихся для этого шагов, а также по числу использованных состояний автомата.

Авторами было проведено 15 экспериментов по построению оптимального автомата. В каждом эксперименте были использованы следующие параметры алгоритма:

- скорость испарения феромона – 0,5;
- стагнационный параметр – 10^6;
- число состояний полного НКА – 12.

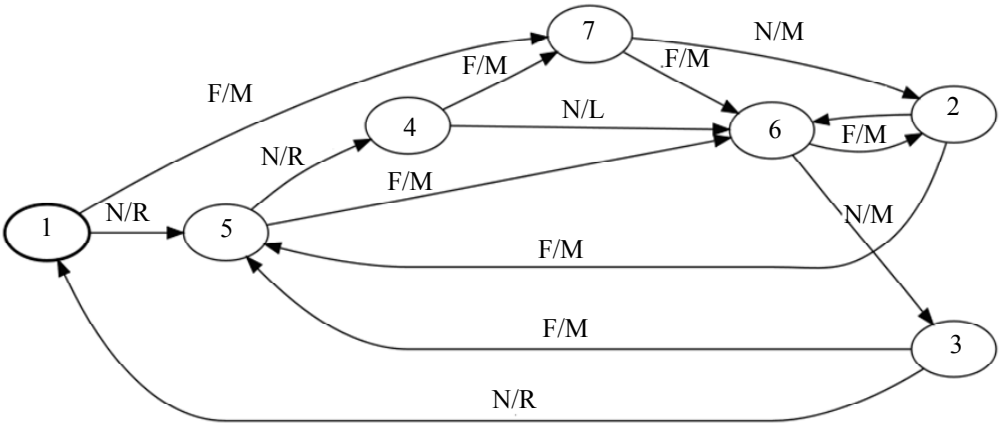

Рис. 4. Управляющий автомат для задачи об «Умном муравье»

В каждом из экспериментов был получен автомат, имеющий семь состояний, который съедает всю еду за 189 ходов. Минимальное число вычислений функции приспособленности, потребовавшееся для

построения оптимального автомата – $1,65 \cdot 10^6$, максимальное – $60,16 \cdot 10^6$, среднее значение – $26,49 \cdot 10^6$ (стандартное отклонение – $17,45 \cdot 10^6$). При этом среднее время построения оптимального автомата составило примерно 32 мин. Диаграмма переходов одного из построенных автоматов приведена на рис. 4. При этом отметим, что для построения такого автомата с помощью генетического программирования в работе [3] потребовалось в среднем $220 \cdot 10^6$ вычислений функции приспособленности. Таким образом, с помощью предложенного алгоритма удалось найти решение задачи об «Умном муравье» в 8,3 раза быстрее, чем при использовании генетического алгоритма.

Заключение

Разработан метод построения управляющих конечных автоматов, основанный на муравьиных алгоритмах. Разработано программное средство на языке Java, реализующее предлагаемый метод. Метод был протестирован на задаче об «Умном муравье». При использовании модифицированной функции приспособленности, учитывающей число состояний автомата, удалось найти решение этой задачи в 8,3 раза быстрее, чем с помощью генетического алгоритма.

Преимущество предлагаемого подхода перед генетическим алгоритмом также состоит в том, что для решения каждой конкретной задачи не требуется определять функции мутации и скрещивания, что обычно является нетривиальной задачей.

Исследование поддержано в рамках ФЦП «Научные и научно-педагогические кадры инновационной России на 2009–2013 годы».

Литература

1. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. – СПб: Питер, 2009. – 176 с.
2. Spears W.M., Gordon D.F. Evolving finite-state machine strategies for protecting resources // Proceedings of the International Symposium of Methodologies for Intelligent Systems 2000. ACM Special Interest Group on Artificial Intelligence. – Springer-Verlag, 2000. – P. 166–175.
3. Царев Ф.Н. Метод построения автоматов управления системами со сложным поведением на основе тестов с помощью генетического программирования // Материалы Международной научной конференции «Компьютерные науки и информационные технологии». – Саратов: СГУ, 2009. – С. 216–219.
4. Царев Ф.Н., Шалыто А.А. Применение генетического программирования для генерации автомата в задаче об «Умном муравье» // Сборник трудов IV-ой Международной научно-практической конференции «Интегрированные модели и мягкие вычисления в искусственном интеллекте». – М.: ФИЗМАТЛИТ, 2007. – Т. 2. – С. 590–597.
5. Александров А.В., Казаков С.В., Сергушичев А.А., Царев Ф.Н., Шалыто А.А. Генерация конечных автоматов для управления моделью беспилотного самолета // Научно-технический вестник СПбГУ ИТМО. – 2011. – № 2 (72). – С. 3–11.
6. Lucas S.M. Evolving finite state transducers: Some initial explorations. Genetic Programming // Lecture Notes in Computer Science. – Berlin: Springer, Heidelberg, 2003. – V. 2610. – P. 241–257.
7. Лобанов П.Г. Использование генетических алгоритмов для генерации конечных автоматов: Дис. … канд. тех. наук: 05.13.11. – СПбГУ ИТМО, 2008. – 114 с.
8. Егоров К.В., Царев Ф.Н. Совместное применение генетического программирования и верификации моделей для построения автоматов управления системами со сложным поведением // Сборник докладов конференции молодых ученых и специалистов «Информационные технологии и системы». – М.: ИППИ РАН, 2009. – С. 77–82.
9. Dorigo M. Optimization, Learning and Natural Algorithms // Ph. D. thesis. – Politecnico di Milano. Italy, 1992. – 140 p.
10. Dorigo M., Stützle T. Ant Colony Optimization. – MIT Press, 2004. – 305 p.
11. Dorigo M., Gambardella L.M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem // IEEE Transactions on Evolutionary Computation. – 1997. – V. 1. – № 1. – P. 53–66.
12. Baker J.E. Reducing Bias and Inefficiency in the Selection Algorithm // Proceedings of the Second International Conference on Genetic Algorithms and their Application. – USA: Lawrence Erlbaum Associates, 1987. – P. 14–21.

Чивилихин Даниил Сергеевич – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, студент, chivilikhin.daniil@gmail.com

Ульянцев Владимир Игоревич – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, студент, ulyantsev@rain.ifmo.ru