5 ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

УДК 004.85

ВЫБОР ФУНКЦИИ ПРИСПОСОБЛЕННОСТИ ОСОБЕЙ ГЕНЕТИЧЕСКОГО АЛГОРИТМА С ПОМОЩЬЮ ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ

А.С. Афанасьева, М.В. Буздалов

Предлагается метод, позволяющий динамически выбирать вспомогательную функцию приспособленности, наиболее выгодную для использования в генетическом алгоритме. Метод основан на применении обучения с подкреплением. Приведены результаты его использования для решения модельной задачи.

Ключевые слова: генетические алгоритмы, машинное обучение, обучение с подкреплением.

Введение

В теории оптимизации известны задачи скалярной и многокритериальной оптимизации [1]. На практике возникает необходимость решать модифицированную скалярную задачу оптимизации, предполагающую наличие дополнительных критериев [2]. Она отличается от многокритериальной задачи тем, что целью ее решения является максимизация единственной целевой функции, а дополнительные критерии имеют лишь вспомогательное значение. В таких задачах целевая функция может некоторым образом зависеть от дополнительных критериев, поэтому в ряде случаев вместо максимизации целевой функции оказывается выгодным оптимизировать дополнительные критерии.

Важным классом алгоритмов оптимизации являются генетические алгоритмы (ГА) [3], где в качестве критерия выступает функция приспособленности (ФП). При наличии нескольких вспомогательных ФП выбор наиболее выгодной из них приходится производить вручную [4]. Подобный подход не вполне эффективен, так как предполагает многократный перезапуск ГА. В представляемой работе предлагается метод, позволяющий автоматизировать этот процесс. Более того, метод позволяет осуществлять динамический выбор в случае, когда на разных этапах выполнения алгоритма выгодны разные ФП.

Введем некоторые понятия. Пусть имеется набор функций приспособленности. Целью применения ГА является получение особи, имеющей как можно лучшее (например, наибольшее) значение одной из этих функций. Будем называть такую ФП целевой ФП (ЦФП), остальные ФП будем называть вспомогательными. Функцию приспособленности, используемую в ГА для отбора особей в данный момент времени, будем называть текущей ФП.

Будем рассматривать случай, в котором применение одной из вспомогательных ФП в качестве текущей приводит к более быстрому получению особей с более высокими значениями ЦФП, нежели применение самой ЦФП. Целью настоящей работы является демонстрация на модельной задаче того, что применение обучения с подкреплением [5] позволяет выбирать оптимальные текущие ФП во время работы ГА.

Модельная задача

Рассмотрим следующую модельную задачу. Пусть особь представляется битовой строкой фиксированной длины n. Будем обозначать число единиц в такой строке как x. Функции приспособленности особи будем задавать как функции от x. ЦФП задается формулой $g(x) = \left\lfloor \frac{x}{k} \right\rfloor$. Вспомогательные ФП имеют следующий вид: $h_1(x) = \begin{cases} x, x < p \\ p, x \geq p \end{cases}$ и $h_2(x) = \begin{cases} p, x < p \\ x, x \geq p \end{cases}$, где p – положительное целое число. Будем называть p точкой переключения.

Можно видеть, что для особей с числом единиц, меньшим значения точки переключения, выгоднее использовать функцию h_1 в качестве текущей ФП. Для особей, число единиц в представлении которых превышает значение точки переключения, выгодно использовать h_2. Применение этих функций будет быстрее приводить к более высоким значениям целевой функции g.

Задачей разрабатываемого метода является динамическое переключение на текущую ФП, наиболее выгодную для особей, составляющих текущее поколение. Другими словами, метод должен сначала выбирать функцию h_1 в качестве текущей, а после достижения точки переключения выбирать функцию h_2.

Обучение с подкреплением

Большинство алгоритмов обучения с подкреплением относится к методам машинного обучения, не требующим наличия заранее подготовленного набора тестовых примеров [5, 7]. Инкрементальный характер этих алгоритмов позволяет обеспечить динамический характер переключения функций приспо-

собленности и освобождает от необходимости подготовки набора тестовых примеров, требующей выполнения многократных запусков ГА.

Кратко опишем идеи обучения с подкреплением. Агент обучения применяет действия к среде, которая отвечает на каждое действие поощрением. Целью агента является максимизация суммарного поощрения. Для эффективного применения использованных в данной работе алгоритмов необходимо, чтобы задача могла быть представлена в виде марковского процесса принятия решений.

Марковский процесс принятия решений состоит из:

- множества состояний S;
- множества действий A;
- функции поощрения $W : S \times A \to R$;
- функции перехода между состояниями $T : S \times A \times S \to R$, заданной таким образом, что вероятность перехода среды из состояния s в состояние s' после действия агента a равна $T(s,a,s')$.

В следующем разделе будет показано, как описанная ранее модельная задача была представлена в виде марковского процесса принятия решений.

Представление модельной задачи в терминах обучения с подкреплением

Множество действий агента задается с помощью функций приспособленности: $A = \{g, h_1, h_2\}$. Применение того или иного действия означает выбор соответствующей ФП в качестве текущей и последующее формирование нового поколения ГА.

Состояние среды соответствует состоянию ГА. Оно однозначно определяется упорядоченным вектором значений производных всех применяемых ФП в точке x, задаваемой числом единиц в лучшем индивидууме поколения. Например, если выполняется неравенство $h_1'(x) \le g'(x) \le h_2'(x)$, то состояние имеет вид $\langle h_1, g, h_2 \rangle$.

Значение функции поощрения зависит от изменения значений функций приспособленности при переходе из одного состояния в другое, вызванном действием агента. Введем вспомогательную функцию D_f, такую, что

$$D_f(x_1, x_2) = \begin{cases} 0 : f(x_2) - f(x_1) < 0 \\ 0{,}5 : f(x_2) - f(x_1) = 0 \\ 1 : f(x_2) - f(x_1) > 0 \end{cases},$$

где f – некоторая фиксированная ФП. Функция поощрения имеет вид $W(s,a) = D_g(x_s, x_{s'}) + c\left(D_{h_1}(x_s, x_{s'}) + D_{h_2}(x_s, x_{s'})\right)$, где $c \in [0,1]$ – вещественный коэффициент, позволяющий варьировать вклад вспомогательных функций в величину поощрения; s' – состояние, в которое переходит среда из состояния s после действия a; x_s и $x_{s'}$ – значения, соответствующие лучшим особям в состояниях s и s'.

Наряду с приведенными определениями рассматривались и альтернативные: состояние среды определялось упорядоченным вектором значений функций приспособленности, значение функции поощрения – изменением значения ЦФП. Однако такие определения в ходе предварительного эксперимента не позволяли эффективно бороться с зашумленностью среды и приводили к непостоянству значений функции поощрения, иными словами, к переменному характеру среды. Для борьбы с подобным эффектом важно, чтобы после точки переключения среда переходила в другое состояние, в котором функция поощрения может быть определена по-новому без ущерба для постоянства среды. Предложенные в конечном итоге определения удовлетворяют этому требованию.

Описание реализации алгоритма

Для решения поставленной задачи было применено Q-обучение, которое является разновидностью инкрементальных алгоритмов обучения с подкреплением, не строящих модель среды. Особенностью таких алгоритмов является малая требовательность к вычислительным ресурсам на каждом шаге [7]. Эта особенность важна для того, чтобы не замедлять работу ГА.

Использовалось две разновидности Q-обучения: Q-обучение с ε-жадным правилом [5, 8] и отложенное Q-обучение, предложенное в работе [9]. Был реализован инкрементальный алгоритм. Каждый шаг этого алгоритма состоит из шага обучения, задачей которого является выбор главной ФП, и шага ГА, состоящего в генерации следующего поколения.

Разработаны независимые интерфейсы среды и агента обучения с подкреплением, что должно позволить данному методу работать не только с ГА, но и с другими алгоритмами оптимизации. Для добавления нового типа алгоритма следует реализовать соответствующий интерфейс среды, что предполагает

разработку состояний среды и функции поощрения. После этого к новой среде можно будет применять ранее реализованные алгоритмы обучения с подкреплением.

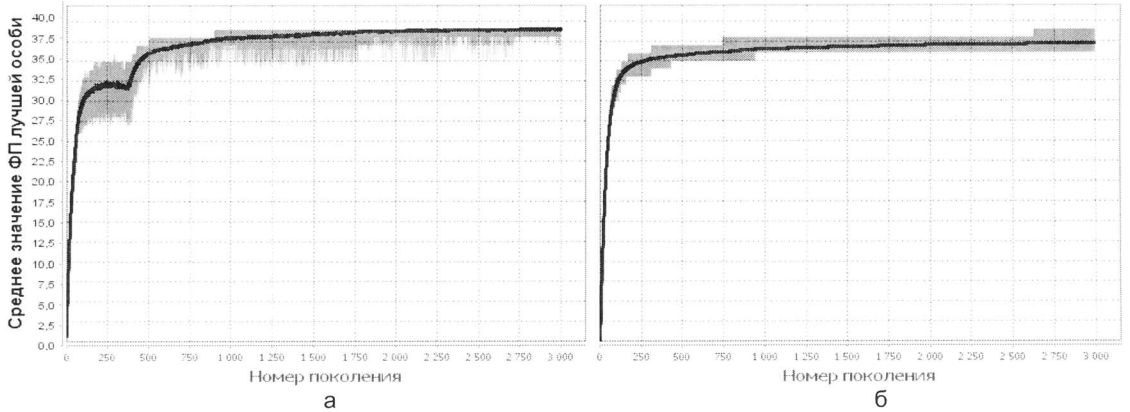
Описание и результаты эксперимента

В качестве эксперимента модельная задача решалась с помощью двух алгоритмов, один из которых основан на отложенном Q-обучении, другой – на ε-жадном. Была собрана статистика, описывающая запуски ГА с применением этих алгоритмов при различных комбинациях значений параметров как самой задачи, так и использовавшихся алгоритмов обучения. Вероятности кроссовера и мутации, применявшиеся в ГА, составляли 70% и 3% соответственно.

В качестве параметров задачи выступали длина строки l, представляющей особь ГА, значение точки переключения p, а также делитель k, входящий в определение целевой функции $g(x)$. Характер параметров обучения зависел от конкретного алгоритма. В случае отложенного обучения настраивались такие параметры, как период обновления значений Q-функции m, дополнительное поощрение ε_1 и величина дисконтного фактора γ [9]. Для ε-жадного обучения задавались вероятность выбора случайного действия ε, скорость обучения α и также величина дисконтного фактора γ [8].

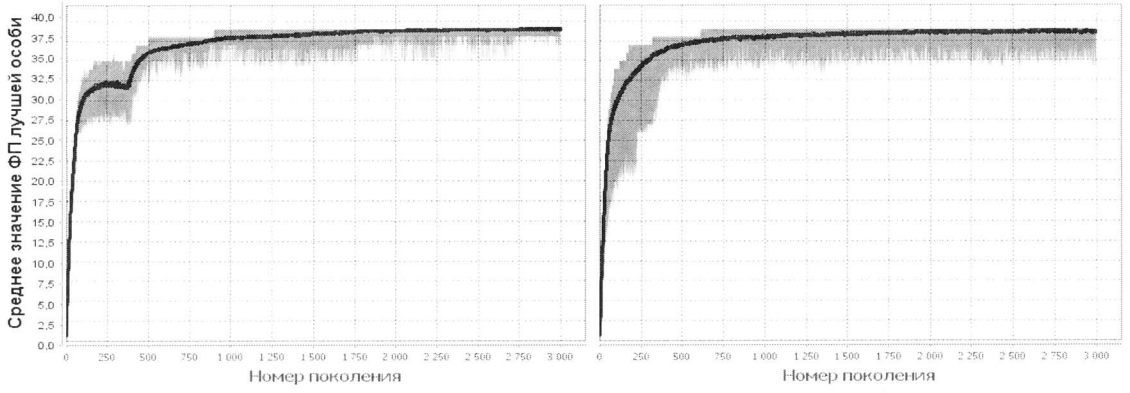
Во время предварительного эксперимента было выявлено, что в качестве значения константы c, входящей в определение функции поощрения W, разумно брать значение $c = 0{,}5$. Подобный выбор позволяет детальнее учитывать состояние ГА, чем выбор, при котором значения c близки к нулю, и сохранить преимущество целевой функции перед вспомогательными, что не достигается при значениях c, близких к единице.

В ходе основного эксперимента было перебрано более тысячи комбинаций различных значений параметров, каждая конфигурация запускалась 100 раз с целью усреднения. Рассмотрим результаты для параметров задачи $l = 400$; $p = 266$; $k = 10$.



а б

Рис. 1. Графики работы ГА, к которому применен алгоритм, основанный на отложенном Q-обучении (а), и обычного ГА (б)



а б

Рис. 2. Графики работы ГА, к которому применены алгоритмы, основанные на отложенном (а) и на ε-жадном (б) Q-обучении

На рис. 1 показан сравнительный график работы ГА без обучения (рис. 1, б) и работы алгоритма с отложенным Q-обучением (рис. 1, а). Для построения графика использованы оптимальные параметры алгоритма отложенного обучения, выявленные в ходе эксперимента: $m = 100$; $\varepsilon_1 = 0{,}2$; $\gamma = 0{,}01$. Можно видеть, что скорость работы ГА с обучением превосходит скорость работы обычного ГА. Это объясняет-

ся тем, что обучение справляется с задачей выбора оптимальной ФП. Отставание отложенного обучения на промежутке до 375-го поколения соответствует периоду накопления опыта агентом.

На рис. 2 представлен сравнительный график работы алгоритмов, основанных на отложенном и ε-жадном Q-обучении. Параметры отложенного обучения – те же, что и в предыдущем примере. Для жадного обучения были подобраны параметры $\varepsilon = 0{,}1$; $\alpha = 0{,}1$; $\gamma = 0{,}01$. Жадное обучение быстрее реагирует на изменение состояния среды, комбинируя накопление опыта с его применением. В целях исследования оно выбирает случайное действие с вероятностью ε. Этим объясняется большой разброс значений ФП особей, и, как следствие, отставание среднего значения ФП от значения, обеспечиваемого другим алгоритмом. Алгоритм, применяющий отложенное Q-обучение, характеризуется меньшим разбросом значений ФП особей. После прохождения периода накопления опыта он показывает лучшие результаты, чем алгоритм с жадным обучением.

На рис. 3 представлены графики, показывающие число переключений на ту или иную ФП. ФП 1 на рисунке обозначает $h_1(x) = \begin{cases} x, x < p \\ p, x \geq p \end{cases}$; ФП 2 соответствует $h_2(x) = \begin{cases} p, x < p \\ x, x \geq p \end{cases}$; ФП 3 представляет собой целевую функцию $g(x) = \left\lfloor \frac{x}{k} \right\rfloor$, где $p = 266$; $k = 10$. Можно видеть, что ε-жадное обучение быстрее и стабильнее выбирает ФП 1 на первом промежутке значений ЦФП особей, расположенном перед точкой переключения. Отложенное обучение на этом промежутке находится в стадии накопления опыта, что отражается на графике в виде примерно равного числа переключений на все три ФП.

Отложенное обучение после накопления опыта выбирает $g(x)$ в качестве главной ФП, чем объясняется отсутствие роста графика, соответствующего отложенному обучению, на промежутке от 125-го до 375-го поколений на рис. 1, 2, но затем достаточно быстро переключается на оптимальную $h_2(x)$. Таким образом, оба алгоритма обеспечивают стабильный выбор наиболее эффективной ФП 2 на втором участке значений ЦФП особей, следующем после точки переключения.

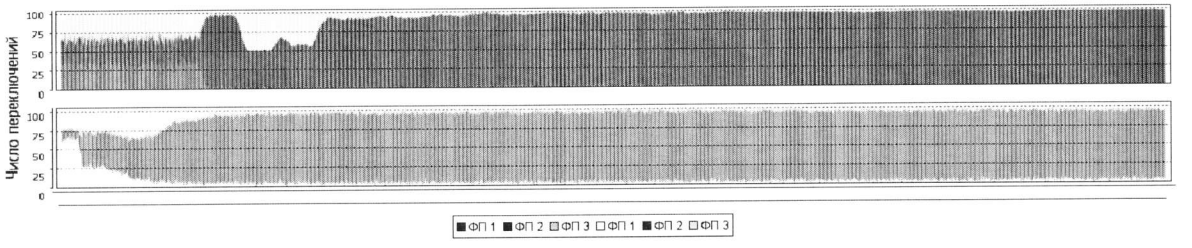


Рис. 3. Переключение ФП с применением отложенного (верхняя диаграмма) и ε-жадного (нижняя диаграмма) Q-обучения

Можно сделать вывод о том, что оба алгоритма способны переключаться на оптимальную ФП, что приводит к повышению производительности ГА. Использование ε-жадного обучения позволяет переключаться на оптимальную ФП практически сразу. Алгоритм, использующий отложенное Q-обучение, переключается при условии предоставления достаточного времени для накопления опыта, однако он показывает более стабильные и высокие результаты на длительном временном промежутке.

Заключение

В работе описан метод, позволяющий динамически настраивать генетический алгоритм с целью ускорения поиска наиболее приспособленных особей. Метод основан на применении обучения с подкреплением для выявления наиболее выгодной вспомогательной функции приспособленности. В предлагаемый метод заложены возможности расширения для совместного использования с произвольным алгоритмом оптимизации. Проведенный эксперимент показал, что метод позволяет эффективно решить поставленную модельную задачу. Это позволяет судить о перспективности применения методов машинного обучения к выбору вспомогательных оптимизируемых величин. Ниже перечислены возможные направления дальнейшей работы.

- Реализация затухающего в рамках фиксированного состояния Q-обучения с ε-жадным правилом, которое могло бы совместить в себе стабильность отложенного Q-обучения и быстроту реакции ε-жадного Q-обучения.
- Исследование целесообразности применения обучения с подкреплением, строящего модель среды. Этот подход позволяет более детально учитывать особенности среды, однако обычно характеризуется большими требованиями к вычислительной производительности [7].
- Применение обучения, ориентированного на меняющуюся среду [10].

– Применение предложенного подхода к решению практических задач, в частности, применение метода к генетическому выращиванию тестов, описанных в работе [4].

Литература

1. Лотов А.В., Поспелова И.И. Многокритериальные задачи принятия решений: Учебное пособие. – М.: МАКС Пресс, 2008. – 197 с.
2. Буздалов М.В. Генерация тестов для олимпиадных задач по теории графов с использованием эволюционных алгоритмов. Магистерская диссертация. СПбГУ ИТМО, 2011 [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/papers/2011-master-buzdalov/, св. Яз. рус. (дата обращения 25.11.2011).
3. Mitchell M. An Introduction to Genetic Algorithms. – MA: MIT Press, 1999. – 221 p.
4. Буздалов М.В. Генерация тестов для олимпиадных задач по программированию с использованием генетических алгоритмов // Научно-технический вестник СПбГУ ИТМО. – 2011. – № 2 (72). – С. 72–77.
5. Sutton R.S., Barto A.G. Reinforcement Learning: An Introduction. – MA: MIT Press, 1998. – 322 p.
6. Mitchell T.M. Machine Learning. – McGraw Hill, 1997. – 432 p.
7. Kaelbling L.P., Littman M.L., Moore A.W. Reinforcement Learning: A Survey // Journal of Artificial Intelligence Research. – 1996. – V. 4. – P. 237–285.
8. От моделей поведения к искусственному интеллекту / Под ред. В.Г. Редько. – 2-е. изд. – М.: КомКнига, 2010. – 456 с.
9. Strehl A.L., Li L., Wiewora E., Langford J., Littman M.L. PAC model-free reinforcement learning // Proceedings of the 23rd international conference on Machine learning. ICML'06. – 2006. – P. 881–888.
10. Da Silva B.C., Basso E.W., Bazzan A.L.C., Engel P.M. Dealing with non-stationary environments using context detection // Proceedings of the 23rd international conference on Machine learning. ICML'06. – 2006. – P. 217–224.

Афанасьева Арина Сергеевна – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, студент, afanasyevarina@gmail.com

Буздалов Максим Викторович – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, аспирант, mbuzdalov@gmail.com

УДК 004.415.53

АВТОМАТИЧЕСКИЕ МЕТОДЫ МОДИФИКАЦИИ РЕШЕНИЙ ДЛЯ ТЕСТИРОВАНИЯ ПРОВЕРЯЮЩИХ ПРОГРАММ

А.А. Ахи, А.С. Станкевич, А.А. Шалыто

Предлагается метод автоматизированного тестирования проверяющих программ для олимпиадных задач по программированию, предназначенный для выявления ошибок в тестируемых программах. Предлагаемый метод основан на использовании модификации решений, проверяемых проверяющей программой. Описывается использование метода для тестирования олимпиадных задач полуфинала чемпионата мира ACM ICPC NEERC 2011 и соревнований Russian Code Cup 2011, при этом в ряде проверяющих программ были обнаружены ошибки.

Ключевые слова: олимпиады по программированию, олимпиадные задачи, проверяющие программы, тестирование.

Введение

Олимпиады по программированию проводятся в большом числе по всему миру. Они способствуют выявлению талантливых программистов среди школьников и студентов. Одной из наиболее значимых является международная студенческая олимпиада по программированию International Collegiate Programming Contest [1], которая проводится Association for Computing Machinery (далее олимпиада будет упоминаться как ACM ICPC). Данная олимпиада обладает обширной сетью отборочных соревнований, в число которых входят полуфинальные соревнования North East European Regional Contest [2] (NEERC), проходящие на территории России. Значимой олимпиадой на территории России также является соревнование Russian Code Cup [3], проводимое корпорацией Mail.ru Group.

На олимпиадах по программированию предлагается решить несколько задач. Формулировка задачи предполагает чтение входных данных, формат которых описан в условии задачи, и получение требуемых выходных данных, формат которых также описан в условии задачи. Решением задачи является программа, написанная на одном из языков программирования (например, в соревновании ACM ICPC используются языки C, C++ и Java [4]) и преобразующая входные данные в выходные.

Для проверки программа запускается на наборе заранее подготовленных тестов, не известных участникам. Решение считается прошедшим определенный тест, если оно, при работе с ним, не нарушило ограничений, указанных в условии задачи, завершилось корректно (без ошибок времени выполнения), а