УДК 004.932.72'1

МЕТОДЫ ОПТИМИЗАЦИИ СТРАТЕГИЙ В ИГРАХ ДЛЯ ДВУХ УЧАСТНИКОВ С ИСПОЛЬЗОВАНИЕМ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

Д.А. Трофимов, А.А. Шалыто

Предложена модификация генетического алгоритма, позволяющая решать задачу оптимизации без задания целевой функции в явном виде, используя только функцию сравнения пары решений-кандидатов. Для этого используются так называемые турнирные схемы. Предложенный алгоритм позволяет применять основные принципы генетических алгоритмов в тех задачах, в которых применение генетических алгоритмов в классическом виде невозможно, либо слишком неэффективно. К указанному классу задач относятся задачи построения оптимальной стратегии против заранее неизвестного оппонента в играх двух участников.

Ключевые слова: генетические алгоритмы, игры для двух участников.

Введение

Во многих задачах многомерной оптимизации, где невозможно найти точное решение за приемлемое время, используются разнообразные эвристические алгоритмы. Достаточно часто для этой цели используются генетические алгоритмы, отличительной особенностью которых является наличие операций скрещивания и мутации между решениями-кандидатами [1].

К сожалению, не для всех задач генетические алгоритмы применимы в своем классическом виде. Это связано с тем, что для каждого решения необходимо вычислять значение так называемой функции приспособленности. При этом возникает проблема определения такой функции и алгоритма ее вычисления. Если функцию приспособленности задавать в точном соответствии с критерием оптимизации исходной задачи, временные затраты на вычисление функции в заданной точке могут быть слишком велики. Поскольку для получения качественного результата в ходе выполнения генетического алгоритма требуется выполнять такие вычисления большое число раз, общее время работы алгоритма становится неприемлемо высоким. По мнению авторов, это является главным недостатком генетических алгоритмов по сравнению с другими методами оптимизации.

В данной работе рассматривается класс задач, в которых при отсутствии заданного алгоритма вычисления функции приспособленности для одного отдельного взятого решения известен алгоритм сравнения двух решений, определяющий лучшее из них.

Известны следующие способы решения таких задач.

- Выбирается «эталонное» решение, далее задача сводится к нахождению такого решения, результат игры которого против эталонного максимален. Недостатки такого подхода: требуется наличие такого эталона, результат которого зависит от выбора эталона, если сравнение выполняется только качественно, но не количественно (т.е. можно определить, лучше ли данное решение, чем эталон, но нельзя определить, насколько), невозможно этим способом выбрать лучшее среди всех решений, побеждающих эталонное.
- Метод восхождения к вершине (hill climbing) [2] заключается в следующем. Случайным образом выбирается начальное решение. Далее на каждом шаге создается случайная модификация текущего решения. Полученное решение сравнивается с исходным и, если оно оказывается лучше, заменяет собой текущее, в противном случае оно отбрасывается. Процесс повторяется заданное число шагов.

В настоящей работе представлен способ модификации генетического алгоритма. Полученный алгоритм выполняет оптимизацию заданной игровой стратегии с выделенными параметрами оптимизации и не требует для своей работы вычисления в явном виде функции приспособленности отдельно взятой стратегии, используя только заданную функцию сравнения пар решений-кандидатов. Алгоритм принципиально отличается от обычного генетического алгоритма способом отбора. Алгоритм не требует возможности количественного сравнения и работает с недетерминированной (стохастической) функцией сравнения.

Рассматриваемый класс задач

Игра – процесс, в котором участвуют две стороны, ведущие борьбу за реализацию своих интересов [3]. Каждая из сторон имеет свою цель и использует некоторую стратегию, которая может вести к выигрышу или проигрышу в зависимости от поведения противоположной стороны.

Далее рассматриваются игры для двух игроков *с нулевой суммой*, в которых выигрыш одного участника равен проигрышу второго.

Стратегия – алгоритм поведения игрока, принимающего участие в игре, обладающий некоторым количеством настраиваемых параметров.

Во многих играх, в частности, в играх с большим количеством состояний или с неполной информацией невозможно (либо требует неприемлемо большого времени) построение детерминированной оптимальной стратегии. Для стратегий в таких играх, реализованных «вручную», типично наличие многих параметров (пороговых значений, весовых коэффициентов), которые подбираются эмпирически. Каждый такой параметр далее считается вещественным числом, принадлежащем промежутку [0, 1).

Участник – стратегия с зафиксированными значениями параметров.

Пространство множества участников для стратегии с K параметрами представляет из себя пространство векторов $[0, 1)^K$.

Множество возможных участников игры U – это объединение по всем стратегиям множеств участников для каждой стратегии. Каждый участник задается идентификатором своей стратегии и вектором параметров этой стратегии.

Судья (или функция сравнения) – алгоритм, на вход которому подаются два участника, выдающий результат игры между ними – выигрыш первого участника (или, что тоже самое, проигрыш второго участника). В зависимости от правил игры возможны следующие варианты множества допустимых результатов:

- $\{-1; 1\}$ – «–1» означает поражение первого из входных участников, «1» – второго;
- $\{-1; 0; 1\}$ – «–1» означает поражение первого из входных участников, «1» – второго, «0» – ничью;
- вещественное число в интервале $[-1, 1]$ – количественный выигрыш первого участника.

Единственным способом оценки качества того или иного участника является сравнение его с другими посредством вызовов судьи. Внутренняя механика игры, а также семантика отдельных параметров стратегий алгоритму оптимизации в общем случае неизвестна, судья для него представляет собой «черный ящик».

Результат игры в общем случае не является детерминированным, т.е. не гарантируется идентичность результата нескольких игр между одними и теми же участниками.

Постановка задачи оптимизации

Целью алгоритма оптимизации является выбор наилучшей стратегии и набора параметров для нее.

В условиях, когда правила игры не дают количественной оценки отдельно взятого участника, выбор целевой функции (какого участника считать «наилучшим») не является однозначным. Далее за целевую функцию оптимизации принимается средний результат игры данного участника против каждого возможного оппонента, т.е. следующая величина: $f(A) = \int\limits_{B \in U} E \dfrac{F(A,B) - F(B,A)}{2} dB$.

Рассматривается поиск максимума этой величины в слабом смысле, т.е. если максимум достигается для нескольких решений, оптимальным считается любое из них.

К сожалению, вычисление данной величины нереализуемо на практике. Далее рассматривается приближение этой задачи – выбор лучшего участника из конечного множества, заданного прямым перечислением. В этом случае значение функции для участника равно среднему выигрышу участника в играх против всех возможных оппонентов. Средний выигрыш вычисляется попарным сравнением всех участников из данного множества.

Модификация генетического алгоритма

Ниже описывается алгоритм, наследующий многие признаки генетических алгоритмов (селекция, мутация, кроссовер). Вместо заданной правилами игры функции приспособленности для одного участника, рассчитываемой независимо для каждого участника, алгоритм использует заданную правилами игры функцию сравнения двух участников.

На входе алгоритма – количество стратегий S, количество параметров для каждой стратегии $K_i (i \in [0..S-1])$, а также судья – функция, принимающая на вход упорядоченную пару участников и возвращающая результат игры между ними. Каждый участник кодируется парой $(i \in [0..S-1], V \in [0,1)^{K_i})$. На выходе алгоритма – наилучший участник, полученный алгоритмом к моменту останова.

Алгоритм выглядит следующим образом:

1. Создать начальную популяцию из N участников. N раз случайно выбирается стратегия, независимо один от другого выбираются параметры этой стратегии;
2. Провести турнир среди N участников, выбрать M лучших;
3. Применяя операции мутации и кроссовера к M отобранным участникам, создать новую популяцию из N участников;
4. Повторить п.п. 2–3 до наступления условий останова;
5. В последнем поколении провести турнир, выявляющий одного лучшего участника, вернуть его в качестве ответа.

Алгоритм получения следующего поколения участников определяется аналогично таковому из классической схемы генетического алгоритма. Далее рассматривается следующий вариант:

1. Выбрать из N участников M;
2. $N/2$ раз случайно равновероятно выбрать пару из M участников с одинаковой стратегией, получить нового участника кроссовером между участниками этой пары;

3. $N/2$ раз случайно равновероятно выбрать одного участника из M, создать нового участника мутацией выбранного;
4. N участников, полученных на предыдущих двух шагах – это и есть новое поколение.

Турнир – способ выделения из N участников M лучших, при этом M лучших между собой ранжировать не требуется. Часто используемые турнирные схемы [4, 5].

- Круговой турнир (round-robin, RR) – наиболее объективен, но требует $O(N^2)$ операций сравнения.
- Турниры с выбыванием после первого (single elimination, SE) или после двух поражений (double elimination, DE), требуют порядка $O(N)$ операций сравнения.
- Швейцарский турнир (Swiss-system, SW) – аналог кругового, но с меньшим количеством туров и специальным разбиением на пары, требует порядка $O(N\log NM)$ операций сравнения.

Мутация и кроссовер определены аналогично обычному генетическому алгоритму. Мутация заключается в замене одного случайного параметра участника случайным числом. Кроссовер заключается в том, что значение каждого параметра нового участника выбирается случайным образом между значениями параметров участников-«родителей».

Чтобы сравнить между собой результаты работы нескольких вариаций данного алгоритма, следует прерывать алгоритм спустя некоторое фиксированное, одинаковое для всех алгоритмов время. Результаты работы алгоритмов – лучших участников – сравниваются либо проведением кругового турнира между ними, либо сравнением суммарного результата игр против некоего тестового множества участников. Для измерения времени удобнее использовать не секунды, а число вызовов функции сравнения.

Стохастические функции сравнения

В ряде случаев функция сравнения может возвращать различные результаты при повторных вызовах с одними и теми же значениями аргументов. Это может быть обусловлено следующими причинами:

- случайность заложена в правила игры (например, в карточных играх, где раздача карт в каждой игре генерируется заново);
- даже если правила игры не используют случайных величин, их могут использовать один или оба участника. Например, участник делает не наилучший с его точки зрения возможный ход, а произвольный, при этом вероятность совершения того или иного хода пропорциональна его «качеству» с точки зрения участника.

В случае двух возможных исходов игры – «победа» или «поражение» первого из участников, результат сравнения участников A и B – случайная величина, задаваемая одним параметром $P(A, B)$ – вероятностью победы участника A. Назовем «истинным» более вероятный из двух исходов. Далее рассматривается частный случай, когда $P(A, B)=\text{const}(A, B)=p>0{,}5$. Величина $q=1-p$ – вероятность функции сравнения выдать результат, противоположный истинному, далее называется *уровнем шума.*

При уровне шума выше некоторого порога сходимость алгоритмов, имевшая место в случае детерминированной функции, может нарушиться. Для борьбы с этим эффектом уровень шума функции сравнения уменьшается следующим алгоритмом: R раз провести игру между одними и теми же участниками, выдать более часто встретившийся исход. Оптимальное значение R для алгоритма с заданной турнирной схемой и ограничением на число вызовов функции сравнения подбирается экспериментальным путем.

Результаты

Для сравнения алгоритмов использовалась следующая игра. Два игрока называют по K вещественных чисел от 0 до 1, побеждает тот, у кого окажется больше позиций, в которых названное им на этой позиции число превысило таковое у противника. Стратегия для данной игры имеет K параметров – вектор чисел, называемых участником. Для моделирования стохастической функции сравнения рассматривался вариант, когда с заданной вероятностью $1-P$ возвращался результат, противоположный результату исходной игры.

Оценкой алгоритма на всех этапах считался максимум величины $V(A)$ по всем векторам из последнего поколения, где $V(A)$ – медианный элемент вектора A, упорядоченного по возрастанию чисел.

Было поставлено несколько задач оптимизации, каждая из которых задавалась параметром P – вероятностью функции сравнения вернуть истинный результат и параметром C – максимально допустимым числом вызовов функции сравнения. Алгоритм решения задачи оптимизации задается четверкой (T, N, M, R), где T – турнирная схема; N – число участников в одном поколении; M – число участников, порождающих следующее поколение; R – число повторных вызовов функции сравнения для уменьшения ее шума.

В ряде случаев рассмотренный в данной работе алгоритм показал лучшие результаты, чем простой алгоритм локального поиска (hill climbing, HC), заключающийся в следующем: взять случайное начальное решение, далее взять случайную мутацию текущего решения, сравнить с исходным, оставить в качестве текущего победившее, повторять процесс до истечения лимита времени.

Ниже приведены графики зависимости качества лучшего решения от времени для значения $C=10000$ при $P=0{,}55$ (рис. 1) и $P=0{,}8$ (рис. 2) (показаны лучшие алгоритмы для каждой турнирной схемы).

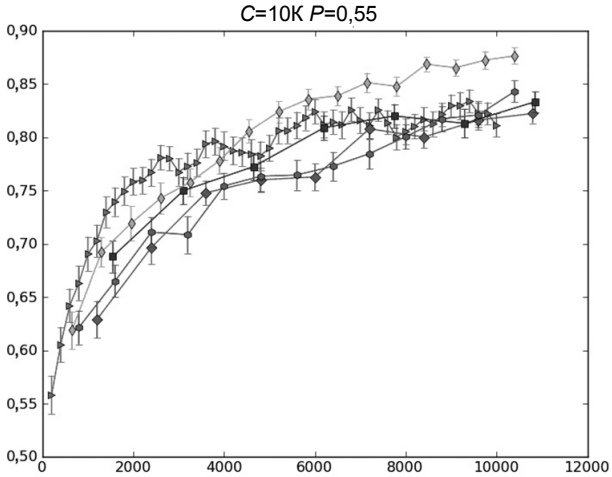

Рис. 1. График зависимости качества лучшего решения от времени для C=10000 и P=0,55;
HC R=100; RR R=10 N=16 M=4; SE R=50 N=32 M=8;
DE R=50 N=8 M=2; SW R=10 N=32 M=4

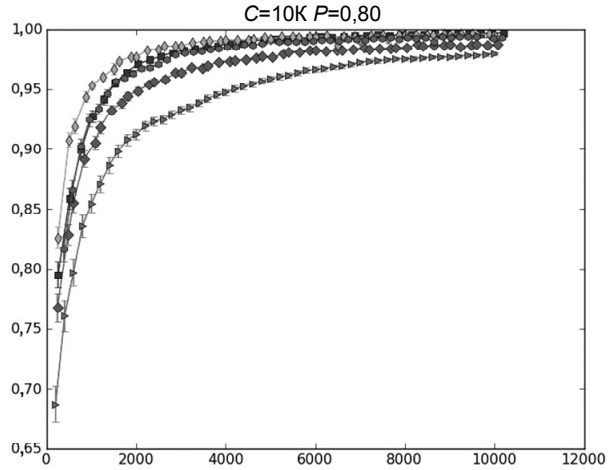

Рис. 2. График зависимости качества лучшего решения от времени для C=10000 и P=0,8;
HC R=50; RR R=1 N=16 M=4; SE R=1 N=256 M=32;
DE R=1 N=64 M=8; SW R=1 N=64 M=8

Выводы

На основе полученных результатов сделаны следующие наблюдения и выводы:

- алгоритмы со временем улучшали значение приведенной выше целевой функции, несмотря на то, что они не выполняли ее вычисление ни для одного участника;
- среди рассмотренных турнирных схем не нашлось универсальной, оказавшейся лучшей для всех задач. Выбор оптимальной турнирной схемы зависит от свойств задачи, в том числе от ограничения на время работы;
- при значении $P \geq 0{,}8$ уменьшение шума функции сравнения оказалось бесполезным для всех турнирных схем (лучшие результаты показали варианты алгоритма с R=1).

Открытым вопросом является возможность применения полученного алгоритма на более широком классе задач. Например, представляют интерес задачи, где стохастические функции сравнения имеют более сложный характер, чем рассмотренная простая модель с одинаковым во всех точках уровнем шума. В ходе их исследования в дальнейшем можно использовать рассмотренные в данной работе алгоритмы и выводы.

Литература

1. Goldberg D.E. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading. – MA: Addison-Wesley, 1989.
2. Гилл Ф., Мюррей У., Райт М. Практическая оптимизация. – М.: Мир, 1985.
3. Петросян Л.А., Зенкевич Н.А., Семина Е.А. Теория игр: Учеб. пособие для ун-тов. – М.: Высшая школа, Книжный дом «Университет», 1998.

.

4. Fayers M. Elimination Tournaments Requiring a Fixed Number of Wins [Электронный ресурс]. – Режим доступа: http://www.maths.qmul.ac.uk/~mf/papers/meko.pdf, свободный. Яз. англ. (дата обращения 09.02.2011).
5. Brad L. Miller, David E. Goldberg. Genetic Algorithms, Tournament Selection, and the Effects of Noise [Электронный ресурс]. – Режим доступа: http://www.illigal.uiuc.edu/web /technical-reports/1995/01/24/genetic-algorithms-tournament-selection-and-the-effects-of-noise-13pp/, свободный. Яз. англ. (дата обращения 09.02.2011).

Трофимов Дмитрий Алексеевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, студент, dmitriy.tref@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`2

РАЗРАБОТКА МЕТОДОВ ПОСТРОЕНИЯ КОНЕЧНЫХ АВТОМАТОВ С ИСПОЛЬЗОВАНИЕМ АЛГОРИТМА ИМИТАЦИИ ОТЖИГА НА ПРИМЕРЕ ИГРЫ «ВОЙНА ЗА РЕСУРСЫ»

А.К. Заикин

Представленные в работе алгоритмы имитации отжига применяются для генерации автоматов управления защитником в игре «Война за ресурсы». Рассматривается вопрос о применении исследуемых схем алгоритма имитации отжига к задаче построения конечного автомата, управляющего защитником в данной игре, и последующий анализ полученных результатов.

Ключевые слова: алгоритм имитации отжига, конечный автомат, игра «Война за ресурсы».

Введение

Эволюционные вычисления являются одним из активно развивающихся и перспективных направлений в искусственном интеллекте и программировании. Они доказали свою эффективность на практике при решении широкого спектра интересных и сложных задач в различных областях.

В работе [1] предложена игра «Война за ресурсы», на примере которой можно строить эффективные стратегии защиты от нападающего. Целью настоящей работы является применение схем алгоритма имитации отжига [2] для генерации автоматов управления защитником в этой игре. В данной работе исследовался отжиг Коши и его модификации. Для получения энергии решения были рассмотрены два способа оценки полученных автоматов. Выполнены оценка эффективности рассмотренных методов и сравнение их с генетическими алгоритмами.

Постановка задачи

«Война за ресурсы» – это игра для двух игроков на поверхности тора размером N на N. Каждая клетка представляет собой ресурс, за который борются соперники, и может быть свободна или захвачена одним из игроков. В начале игры все клетки, кроме занятых игроками, свободны. Первый игрок (защитник) занимает клетку – (1, 1), а второй (нападающий) – (N, N).

Каждый игрок видит состояния четырех клеток – с севера, юга, запада и востока от себя. В процессе игры противники ходят по очереди. На каждом шаге участник осматривает видимые клетки и перемещается на одну из свободных или ранее захваченных им. Игрок, вставший на свободную клетку, захватывает ее до конца игры.

Игра заканчивается, когда на поле не остается свободных клеток или достигается ограничение по числу шагов. Защитник побеждает, если он захватил больше клеток, чем нападающий.

Нападающий действует по жадной стохастической стратегии: если он видит свободные клетки, то он ходит на произвольную из них; если свободных клеток в поле видимости нападающего нет, то он ходит на любую захваченную им клетку.

Максимальные значения функции приспособленности для полученных автоматов представлены в табл. 1.

	Число состояний									
	1	2	3	4	5	6	7	8	9	10
Значение	0,806	0,867	0,893	0,888	0,901	0,903	0,899	0,905	0,896	0,883

Таблица 1. Значения функции приспособленности лучшего найденного автомата