

## ОЦЕНКА ВРЕМЕНИ РАБОТЫ ЭВОЛЮЦИОННОГО АЛГОРИТМА RMHC ПОД УПРАВЛЕНИЕМ АЛГОРИТМА Q-LEARNING НА ЗАДАЧЕ ONEMAX С МЕШАЮЩИМ КРИТЕРИЕМ ОПТИМИЗАЦИИ

**Буздалов М.В.**

*ассистент кафедры компьютерных технологий НИУ ИТМО,  
mbuzdalov@gmail.com*

**Буздалова А.С.**

*студентка кафедры компьютерных технологий НИУ ИТМО,  
abuzdalova@gmail.com*

**Аннотация:** В данной работе доказывается, что эволюционный алгоритм с двумя критериями оптимизации — целевым и мешающим, управляемый обучением с подкреплением, использующим жадную стратегию исследования среды, находит оптимальное решение столь же эффективно (в асимптотическом смысле), что и эволюционный алгоритм, оптимизирующий только целевой критерий.

### Введение

Метод EA+RL [1, 2], разработанный авторами настоящей статьи, успешно применяется в практических задачах. Метод основан на обучении с подкреплением [3] и предназначен для повышения эффективности эволюционных алгоритмов [4]. Было экспериментально подтверждено, что с помощью указанного метода можно эффективно решить ряд модельных задач, однако до настоящего времени для метода не было разработано теоретического обоснования. В данной работе описаны первые результаты по этому направлению.

### Модельная задача оптимизации

Исследуется модификация модельной задачи оптимизации, известная как OneMax [4] и широко применяемая в теоретических исследованиях эволюционных алгоритмов. Пространство оптимизации в этой задаче — множество векторов длиной  $N$ , состоящее из нулей и единиц. Оптимизируемая величина (целевой критерий оптимизации) — число единиц в векторе. Над вектором определена операция мутации, инвертирующая один случайно выбранный бит.

Данная задача модифицируется путем добавления дополнительного критерия оптимизации — числа нулей в векторе. Целевой и дополнительный критерии оптимизации противоречат друг другу. Цель метода EA+RL

— выбирать такой критерий, чтобы при оптимизации по нему была достигнута цель оптимизации – максимизация целевого критерия. С точки зрения метода неизвестно, какой критерий «оптимальнее», так как на практике целевой критерий не всегда является оптимальным [1, 2].

### Эволюционный алгоритм

В качестве алгоритма оптимизации для указанной модельной задачи используется алгоритм Random Mutation Hill Climbing, управляемый обучением с подкреплением. В качестве алгоритма обучения с подкреплением выбран алгоритм Q-Learning с жадной стратегией исследования среды [3]. Состоянием обучения с подкреплением является число единиц в текущей особи.

Схема алгоритма оптимизации такова:

- Инициализация:
  - $X$  — текущая особь, изначально равна вектору из  $N$  нулей;
  - $Q$  — матрица оценки выгодности переходов размером  $N \times 2$ , изначально заполнена нулями;
  - $F_0$  — функция приспособленности, равная числу нулей в векторе;
  - $F_1$  — функция приспособленности, равная числу единиц в векторе.
  - $M(X)$  — оператор мутации (возвращает вектор, равный вектору  $X$  с одним инвертированным битом);
  - $\alpha \in (0;1)$ ,  $\gamma \in (0;1)$  — параметры алгоритма.
- Итерация:
  - $S = F_1(X)$ ;
  - Если  $S = N$ , заканчиваем работу;
  - $Y = M(X)$ ;
  - $X_0 = X$
  - Если  $Q(S,0) = Q(S,1)$  то  $I =$  равновероятно 0 или 1;
  - Если  $Q(S,0) > Q(S,1)$ , то  $I = 0$ , иначе  $I = 1$ ;
  - Если  $I = 0$ , то  $F = F_0$ , иначе  $F = F_1$ ;
  - Если  $F(Y) \geq F(X)$ , то  $X = Y$ ;
  - $R = F_1(X) - F_1(X_0)$
  - $Q(S,I) = Q(S,I) \cdot (1 - \alpha) + \alpha \cdot (R + \gamma \cdot \max(Q(S,0), Q(S,1)))$

### Лемма об обучении

Сформулируем следующую лемму:

**Лемма об обучении.** Пусть алгоритм впервые посетил состояние  $S$ , а затем вышел из него в какое-либо другое состояние. Тогда при всех следующих посещениях состояния  $S$  обучение с подкреплением выберет функцию  $F_1$ .

Доказательство. При первом посещении состояния  $S$  выполняется равенство  $Q(S,0) = Q(S,1) = 0$ . Следовательно, равновероятно может быть вы-

брана как функция  $F_0$ , так и функция  $F_1$ . В зависимости от того, чему равнялся мутировавший бит — нулю или единице, алгоритм может как остаться в состоянии  $S$ , так и перейти в состояние  $S+1$  или  $S-1$ . Рассмотрим эти случаи в зависимости от нового состояния:

1.  $S, R=0$ . Максимум из  $Q(S, 0), Q(S, 1)$  также равен нулю, поэтому значения  $Q$  изменены не будут, и этот случай не влияет на дальнейший выбор обучения с подкреплением.
2.  $S+1, R=1$ . Следовательно, была выбрана функция  $F_1$ . Тогда  $Q(S, 1)$  будет равен  $\alpha \cdot R$ , что является положительной величиной, и  $Q(S, 1) > Q(S, 0)$ .
3.  $S-1, R=-1$ . Следовательно, была выбрана функция  $F_0$ . Тогда  $Q(S, 0)$  будет равен  $\alpha \cdot R$ , что является отрицательной величиной, и  $Q(S, 1) > Q(S, 0)$ .

Таким образом, при первом выходе из состояния  $S$  будет выполняться  $Q(S, 1) > Q(S, 0)$ . При последующих переходах из состояния  $S$  всегда будет выбираться функция  $F_1$ , причем будет выполняться неравенство  $R \geq 0$ . Если  $Q(S, 1) > 0$  до обновления, то оно будет положительным и после обновления. Если  $Q(S, 1) = 0$ , то оно может стать положительным или остаться нулевым, но в этом случае  $Q(S, 0) < 0$ . Таким образом, неравенство  $Q(S, 1) > Q(S, 0)$  будет всегда выполняться, что приведет к тому, что алгоритм будет всегда выбирать функцию  $F_1$ , что и требовалось доказать.

## Моделирование алгоритма цепью Маркова

Для вычисления матожидания числа итераций алгоритма до достижения оптимума можно воспользоваться цепью Маркова. Перейдем к описанию цепи, моделирующей рассматриваемый алгоритм. Для различения состояния обучения с подкреплением в моделируемом алгоритме и состояния цепи Маркова последние будем называть «вершинами».

Первая группа вершин цепи  $A_i, 0 \leq i \leq N$  соответствует тому, что алгоритм пришел в состояние  $S=i$  в первый раз. В этом случае алгоритм еще не знает, какой критерий следует выбрать. Однако, когда алгоритм покинет это состояние, он уже будет знать, что делать в случае  $S=i$ , согласно лемме об обучении.

Вторая группа вершин цепи  $B_i, 0 \leq i \leq N-2$  соответствует тому, что алгоритм вышел из вершины  $A_{i+1}$ , воспользовавшись функцией  $F_0$ . Так как достичь состояния  $i+1$  алгоритм может, только однажды побывав в состоянии  $i$ , то в этом случае алгоритм будет всегда выбирать функцию  $F_1$ . При этом он может либо остаться в состоянии  $i$ , либо перейти в состояние  $i+1$ .

При попадании в состояние с номером  $i+1$  из вершины  $B_i$  алгоритм также будет всегда выбирать функцию  $F_1$ , так как он уже заведомо побывал в состоянии  $i+1$ . Для таких ситуаций требуется завести третью группу вершин цепи  $C_i, 1 \leq i \leq N-1$ .

Опишем переходы между указанными вершинами.

1. Из вершины  $A_0$  имеется переход по вероятности  $\frac{1}{2}$  в саму себя и по вероятности  $\frac{1}{2}$  в вершину  $A_1$ . Действительно, любая мутация особи, состоящей из нулей, приводит к превращении нуля в единицу. С вероятностью  $\frac{1}{2}$  будет выбрана функция  $F_0$ , которая отклонит эту мутацию, в противном случае будет выбрана функция  $F_1$ , которая эту мутацию примет.
2. Из вершины  $A_i$ ,  $1 \leq i \leq N-1$ , имеется три перехода. С вероятностью  $(N-i)/(2N)$  одновременно будет выбрана функция  $F_1$  и будет инвертирован бит со значением 0, таким образом, состоится переход в  $A_{i+1}$ . С вероятностью  $i/(2N)$  одновременно будет выбрана функция  $F_0$  и будет инвертирован бит со значением 1, таким образом, состоится переход в  $B_{i-1}$ . Во всех остальных случаях, которые составляют вероятность  $\frac{1}{2}$ , алгоритм останется в вершине  $A_i$ .
3. Из вершины  $B_i$ ,  $0 \leq i \leq N-2$ , имеется два перехода. С вероятностью  $(N-i)/N$  будет инвертирован бит со значением 0 и состоится переход в вершину  $C_{i+1}$ . С оставшейся вероятностью алгоритм останется в вершине  $B_i$ .
4. Из вершины  $C_i$ ,  $1 \leq i \leq N-1$ , также имеется два перехода — с вероятностью  $(N-i)/N$  в вершину  $A_{i+1}$ , с оставшейся вероятностью в себя.

### Сведение к линейной взвешенной цепи Маркова

Заметим, что вершины цепи Маркова из групп  $B$  и  $C$  имеют по одному входящему переходу и одному выходящему переходу, если не рассматривать переходы-петли. Кроме того, они принимают участие только в цепочке вершин  $A_i \rightarrow B_{i-1} \rightarrow C_i \rightarrow A_{i+1}$ . Целью построения исходной цепи Маркова является подсчет матожидания числа переходов из состояния  $A_0$  в состояние  $A_N$ . Следовательно, мы можем упростить построенную цепь Маркова, исключив вершины из групп  $B$  и  $C$ , но взамен каждому переходу, помимо вероятности его осуществления, потребуется сопоставить его длину.

Длина перехода  $A_i \rightarrow B_{i-1}$  равняется единице. Из вершины  $B_{i-1}$  можно выйти с вероятностью  $(N-i+1)/N$ , следовательно, матожидание числа итераций и, следовательно, длины перехода  $B_{i-1} \rightarrow C_i$  будет равно  $N/(N-i+1)$ . Аналогично, длина перехода  $C_i \rightarrow A_{i+1}$  будет равна  $N/(N-i)$ . Общая длина перехода  $A_i \rightarrow A_{i+1}$  равна  $1 + N/(N-i+1) + N/(N-i)$ .

В полученной упрощенной цепи Маркова переходы ведут либо из состояния  $A_i$  в себя, либо из  $A_i$  в  $A_{i+1}$ . Таким образом, структура цепи стала линейной, что существенно упрощает расчет матожидания.

Опишем переходы в упрощенной цепи. Из вершины  $A_0$ , как и ранее, имеется два перехода: один в себя с вероятностью  $\frac{1}{2}$  и длиной 1, второй в  $A_1$  с вероятностью  $\frac{1}{2}$  и длиной 1. Из вершины  $A_i$ ,  $1 \leq i \leq N-1$ , имеется три перехода:

- в вершину  $A_i$  с вероятностью  $\frac{1}{2}$  и длиной 1;
- в вершину  $A_{i+1}$  с вероятностью  $\frac{N-i}{2N}$  и длиной 1;
- в вершину  $A_{i+1}$  с вероятностью  $\frac{i}{2N}$  и длиной  $1 + \frac{N}{N-i+1} + \frac{N}{N-i}$ .

Определим функцию  $Z(i)$  как матожидание числа шагов, требуемых для того, чтобы перейти из состояния  $A_i$  в состояние  $A_{i+1}$ . Для  $Z(0)$  верно соотношение  $Z(0) = \frac{1}{2}(1 + Z(0)) + \frac{1}{2}$ , следовательно,  $Z(0) = 2$ . Для  $x > 0$  выполняется соотношение:

$$Z(i) = \frac{1}{2}(1 + Z(i)) + \frac{N-i}{2N} + i\left(\frac{1 + \frac{N}{N-i+1} + \frac{N}{N-i}}{2N}\right),$$

что эквивалентно  $Z(i) = 2 + \frac{i}{N-i+1} + \frac{i}{N-i}$ .

Общее число итераций, таким образом, равняется

$$T_R(N) = \sum_{i=0}^{N-1} \left( 2 + \frac{i}{N-i+1} + \frac{i}{N-i} \right). \quad (1)$$

### Асимптотическая оценка

Для работы простого эволюционного алгоритма, Random Mutation Hill Climbing, на задаче OneMax (без введения дополнительного, мешающего критерия оптимизации) существует следующая оценка матожидания числа итераций до нахождения оптимального ответа:

$$T_0(N) = \sum_{i=0}^{N-1} \frac{N}{N-i} = \sum_{i=0}^{N-1} \left( 1 + \frac{i}{N-i} \right) = \Theta(N \log N). \quad (2)$$

Отметим, что для каждого слагаемого из формулы (1) можно определить верхнюю и нижнюю асимптотические оценки с использованием соответствующего слагаемого из формулы (2):

$$1 + \frac{i}{N-i} < 2 + \frac{i}{N-i+1} + \frac{i}{N-i} < 2 \cdot \left( 1 + \frac{i}{N-i} \right).$$

Отсюда следует, что  $T_0(N) < T_R(N) < 2 \cdot T_0(N)$ .

### Заключение

В работе показано, что асимптотика числа итераций до завершения работы эволюционного алгоритма Random Mutation Hill Climbing под управлением обучения с подкреплением, использующего жадную стратегию,

на модельной задаче OneMax с мешающим критерием равна  $\Theta(N \log N)$ , а число итераций  $T_R(N)$  не превышает  $2 \cdot T_0(N)$ , где  $T_0(N)$  — число итераций того же эволюционного алгоритма без обучения с подкреплением на модельной задаче без мешающего критерия оптимизации.

### Литература

1. *Buzdalova A., Buzdalov M.* Increasing Efficiency of Evolutionary Algorithms by Choosing between Auxiliary Fitness Functions with Reinforcement Learning // Proceedings of the Eleventh International Conference on Machine Learning and Applications, ICMLA 2012. — Boca Raton: IEEE Computer Society, 2012. — Vol. 1. — P. 150-155.
  2. *Буздалова А.С., Буздалов М.В.* Метод повышения эффективности эволюционных алгоритмов с помощью обучения с подкреплением // Научно-технический вестник информационных технологий, механики и оптики — Санкт-Петербург, 2012. — № 5(81) — С. 115–119.
  3. *Sutton R.S., Barto A.G.* Reinforcement Learning: An Introduction. — MIT Press, Cambridge, MA, 1998. — 322 p.
  4. *Eiben A.E., Smith J.E.* Introduction to Evolutionary Computing. — Springer-Verlag, Berlin, Heidelberg, New York, 2007. — 304 p.
-