Университет ИТМО

Факультет информационных технологий и программирования

Кафедра компьютерных технологий

Матвеева Анна Александровна

Выбор вспомогательных оптимизируемых величин в эволюционных алгоритмах при помощи многокритериального обучения с подкреплением

Научный руководитель: зав. каф. ТП, д. т. н., проф. А. А. Шалыто

Санкт-Петербург

2015

Содержание

Введение

Существуют методы повышения эффективности эволюционных алгоритмов при помощи вспомогательных критериев. Одним из таких методов является метод EA+RL (Evolutionary Algorithm and Reinforcement Learning) [1], в котором для выбора вспомогательного критерия, используемого в качестве функции приспособленности на каждом шаге алгоритма, применяется обучение с подкреплением. В обучении с подкреплением агент обучения применяет действие к среде, в результате чего среда переходит в новое состояние и возвращает агенту численную награду. В методе EA+RL в роли среды обучения выступает эволюционный алгоритм, а действием является выбор функции приспособленности – целевой или одной из вспомогательных. Цель обучения с подкреплением – максимизация суммарной награды. Эффективность данного метода была продемонстрирована и теоретически доказана на ряде задач.

Существует ряд методов расчета функции награды. В качестве награды в методе EA+RL ранее использовалась скалярная величина, и приходилось ограничиваться одним способом определения награды. В данной работе предлагается использовать многомерную награду, что позволяет совмещать несколько хорошо зарекомендовавших себя одномерных функций наград.

Глава 1. Обзор предметной области

В данной главе рассматривается предметная область. Приводится краткое описание эволюционных алгоритмов. Дается обзор существующих подходов, применяемых для повышения эффективности эволюционных алгоритмов, основанных на использовании вспомогательных оптимизируемых критериев. Дается описание обучения с подкреплением, метода машинного обучения, на котором основан метод, предлагаемый в данной работе. Также в данной главе приводится подробное описание метода EA+RL, на котором основан предлагаемый в работе метод.

1.1 Эволюционные алгоритмы

Существуют такие задачи оптимизации, точные алгоритмы решения которых являются недостаточно эффективными для применения на практике, а для некоторых задач точных алгоритмов не существует. К таким задачам, например, относятся задачи комбинаторной оптимизации: задача коммивояжера, задача построения расписаний и др. Для решения таких задач можно применять эволюционные алгоритмы [4, 5].

Эволюционные алгоритмы (ЭА) имеют в своей основе принципы природной эволюции. Точки из пространства поиска решений задачи оптимизации представляются в виде особей эволюционного алгоритма. Чтобы определить, насколько особь близка к оптимальному решению, задается функция приспособленности (ФП) [8].

На каждой итерации эволюционный алгоритм работает с некоторым набором особей, который называется поколением. В ходе работы эволюционного алгоритма к особям текущего поколения применяются эволюционные операторы: мутации и скрещивания, в результате применения

которых образуются новые особи. Следующее поколение составляется путем отбора особей, который основывается на том, насколько приспособленной является каждая особь, то есть на значении функции приспособленности, посчитанной на данной особи [4].

В качестве условия останова эволюционного алгоритма, как правило, используются нахождение оптимального решения, стагнация в течение фиксированного числа поколений или выполнение алгоритмом заданного числа итераций. Последнее условие останова применяется, когда значение функции приспособленности оптимального решения заранее неизвестно или время, необходимое для нахождения оптимального решения, слишком велико. При применении этого условия оценка эффективности эволюционного алгоритма выполняется на основе лучшего значения функции приспособленности, полученного за фиксированное число поколений [1, 4, 5]. Если в качестве условия останова используется нахождение оптимального решения, в качестве меры эффективности эволюционного алгоритма, как правило, используется число поколений, необходимых для достижения оптимума.

1.2 Методы использования вспомогательных критериев

Существуют разные методы повышения эффективности эволюционных алгоритмов [15, 19, 20]. Для повышения эффективности решения задач однокритериальной оптимизации с помощью эволюционных алгоритмов можно использовать вспомогательные критерии. Впервые исследование возможности применения вспомогательных критериев оптимизации было проведено в работе [6]. С тех пор было предложено множество методов повышения эффективности оптимизации с помощью вспомогательных критериев, в том числе не накладывающих ограничений на вид оптимизируемых функций. Перейдем к краткому обзору некоторых из этих

методов, а именно: тех, которые применяются для повышения эффективности эволюционных алгоритмов.

В последнее десятилетие активно проводились исследования, направленные на повышение эффективности эволюционных алгоритмов при помощи вспомогательных критериев [2, 10-12]. Некоторые методы предполагают создание вспомогательных критериев оптимизации вручную. Созданные вспомогательные критерии оптимизируются одновременно с целевым критерием при помощи многокритериальных эволюционных алгоритмов [3, 16, 17]. Другие методы оптимизируют вспомогательные критерии не одновременно, а выбирают их динамически во время запуска эволюционного алгоритма. Один из таких методов описан в работе [14]. Данный метод на каждом этапе оптимизации случайным образом выбирает один из вспомогательных критериев и оптимизирует его одновременно с целевым критерием. Недостатком данного метода является то, что он никак не учитывает особенности задачи оптимизации, к которой он применяется. Также существуют методы, которые, напротив, применимы только для той задачи оптимизации, для которой они были разработаны [21]. То есть то, как выбираются вспомогательные критерии, зависит от конкретной задачи оптимизации.

1.3 Обучение с подкреплением

Метод, который предлагается в данной работе, использует для выбора вспомогательных критериев метод обучения с подкреплением. Обучение с подкреплением может применяться для повышения эффективности эволюционных алгоритмов [22, 25-27]. Далее приводится описание идей алгоритмов обучения с подкреплением.

Обучение с подкреплением применяется для решения задач взаимодействия с некоторой средой. Общая схема алгоритмов обучения с

подкреплением следующая: агент обучения применяет некоторое действие к среде. Среда в ответ на применение действия возвращает агенту свое текущее состояние и награду. Агент обучается на основе полученной от среды информации, после чего происходит следующая итерация взаимодействия со средой. Задачей агента обучения является максимизация суммарной награды [7]. На рисунке 1 приведена схема взаимодействия агента обучения со средой. На схеме t – номер итерации алгоритма обучения с подкреплением.

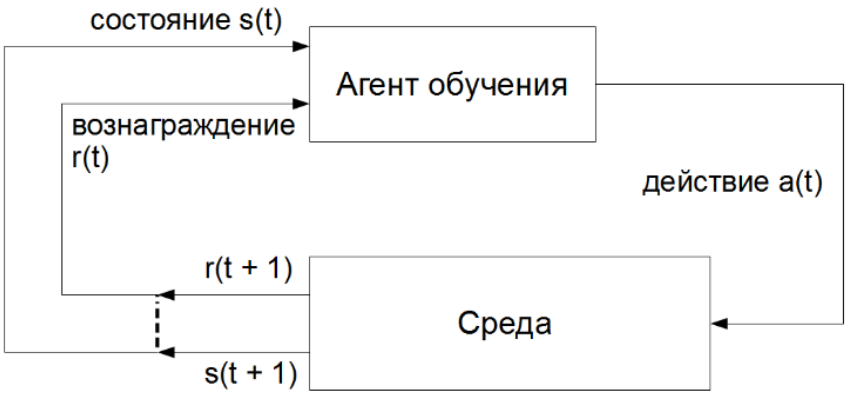

Рисунок 1. Схема взаимодействия агента обучения со средой

Действие, которое агент применяет к среде, может выбираться как из некоторого конечного дискретного набора, так и, например, из множества векторов вещественных чисел R^n [9]. В данной работе в качестве действия агента используется выбор критерия оптимизации из некоторого заранее заданного конечного набора. В связи с этим далее будем рассматривать только те алгоритмы обучения с подкреплением, которые предназначены для работы с конечным дискретным набором действий.

Как и действия агента, состояния среды могут принадлежать как конечному дискретному набору, так и являться векторами вещественных чисел. Метод, предлагаемый в данной работе, рассматривает среду как не меняющую свое состояние в течение работы алгоритма, поэтому не будем

более останавливаться на особенностях алгоритмов обучения с подкреплением, связанных с состояниями среды.

Награда, как правило, является целым или вещественным числом. Положительная награда говорит об эффективности действий агента, а отрицательная – об их неэффективности [7]. Метод, предлагаемый в данной работе, использует многомерную награду, подробное описание которой будет приведено далее.

Алгоритмы обучения с подкреплением совмещают применение накопленного в ходе работы алгоритма опыта с дальнейшим исследованием среды. Зачастую эти задачи конфликтуют между собой. Существует множество стратегий для совмещения исследования среды и применения накопленного опыта. Одна из самых распространенных стратегий – это ε-жадная стратегия исследования среды. Данная стратегия заключается в том, что с вероятностью ε агент обучения выбирает случайное действие, а в остальных случаях использует жадную стратегию исследования среды, то есть выбирает наиболее эффективное для текущего состояния действие. Использование ε-жадной стратегии позволяет избежать остановки в локальном оптимуме. Однако в случае, когда агент уже полностью исследовал среду, выбор случайного действия теряет смысл. [1, 7] В данной работе используется именно ε-жадная стратегия.

Далее приводится описание одного из популярных алгоритмов обучения с подкреплением: Q-обучения. Подробно остановимся на этом алгоритме, так как именно он используется в данной работе для апробации предлагаемого метода.

Алгоритм Q-обучения

Q-обучение является алгоритмом обучения с подкреплением, не строящим модель среды. Во время работы данного алгоритма

аппроксимируется функция $Q(s,a)$, $s \in S$, $a \in A$ – ожидаемое оптимальное вознаграждение за действие a в состоянии среды s, где S – множество состояний среды, A – множество возможных действия агента обучения [7]. Далее приводится псевдокод алгоритма.

Листинг 1. Алгоритм Q-обучения с ε-жадной стратегией исследования среды

Вход: ε – вероятность выбора случайного действия, α – скорость обучения, γ – дисконтный фактор

```
1  Инициализировать Q(s, a) для всех s ∈ S, a ∈ A
2  while (не достигнуто условие останова) do
3     Получить состояние среды s
4     p ← случайное вещественное число ∈ [0, 1]
5     if (p ≤ ε) then
6        a ← случайное действие ∈ A
7     else
8        a ← arg max_a Q(s, a)
9     end if
10    Применить действие a к среде
11    Получить от среды награду r и состояние s'
12    Q(s, a) ← Q(s, a) + α(r + γ max_a' Q(s', a') − Q(s, a))
13 end while
```

1.4 Метод EA+RL

Метод EA+RL [1] является одним из методов настройки эволюционных алгоритмов с помощью обучения с подкреплением. Предлагаемый в работе метод EA+MORL основывается на данном методе.

В данном методе используется однокритериальный эволюционный алгоритм, который настраивается при помощи обучения с подкреплением во время своего выполнения. Помимо целевого критерия оптимизации (целевой функции приспособленности), даны дополнительные критерии оптимизации, то есть дополнительные функции приспособленности. Агент обучения с подкреплением взаимодействует со средой, ассоциируемой с эволюционным алгоритмом. Агент передает эволюционному алгоритму одну из функций

приспособленности, целевую или вспомогательную, которая должна быть использована для формирования следующего поколения эволюционного алгоритма. После применения действия, то есть генерации следующего поколения с использованием полученной от агента обучения функции приспособленности, эволюционный алгоритм возвращает агенту обучения свое состояние и награду, которая в методе EA+RL является одномерной вещественной величиной. Чем выше награда, тем, следовательно, значительнее был рост целевой функции приспособленности. Следовательно, из наиболее оптимального выбора используемой функции приспособленности агентом обучения с подкреплением следует лучшая эффективность эволюционного алгоритма.

В листинге 2 приводится псевдокод метода EA+RL. Псевдокод приводится без привязки к конкретным алгоритму обучения с подкреплением или эволюционному алгоритму.

Листинг 2. Метод EA+RL

```
1  Инициализировать модуль обучения
2  Установить номер текущего поколения: i ← 0
3  Сгенерировать начальное поколение G_0
4  while (не достигнуто условие останова эволюционного алгоритма) do
5     Вычислить состояние s_i и передать его модулю обучения
6     Получить ФП для следующего поколения f_{i+1} из модуля обучения
7     Сгенерировать следующее поколение G_{i+1}
8     Вычислить награду r ← R(s_i, f_{i+1}) и передать ее модулю обучения
9     Обновить номер текущего поколения: i ← i + 1
10 end while
```

1.5 Выводы по главе 1

В данной главе дано описание основных методов, применяемых в данной работе: эволюционных алгоритмов и обучения с подкреплением. Дан обзор существующих методов, повышающих эффективность эволюционных алгоритмов с помощью использования вспомогательных критериев

оптимизации. Приведено описание метода повышения эффективности эволюционных алгоритмов с использованием обучения с подкреплением для выбора критерия оптимизации на каждой итерации алгоритма, на котором основан предлагаемый в данной работе метод.

Глава 2. Постановка задачи и описание метода выбора вспомогательных оптимизируемых величин

В данной главе приводится описание решаемой задачи, и перечисляются требования, предъявляемые к разрабатываемому методу ее решения.

Также описывается предлагаемый метод выбора вспомогательных оптимизируемых величин, дополнительных функций приспособленности эволюционного алгоритма, с использованием обучения с подкреплением. Предлагаемый метод основывается на методе EA+RL (Evolutionary Algorithm and Reinforcement learning). Будем называть предлагаемый метод EA+MORL (Evolutionary Algorithm and Multi-Objective Reinforcement learning).

2.1 Задача выбора вспомогательных оптимизируемых величин

Дана задача однокритериальной оптимизации с целевым критерием $g: W \to R$, где W – дискретное пространство поиска решений. Необходимо найти оптимальное решение. В случае существования более одного оптимального решения достаточно найти одно из них. Решение данной задачи ищется с помощью эволюционного алгоритма.

Также дан конечный набор H вспомогательных критериев оптимизации, $h_i: W \to R, h_i \in H$. Предполагается, что оптимизация этих вспомогательных критериев может привести к нахождению глобального

оптимума целевого критерия за меньшее число поколений эволюционного алгоритма.

2.2 Требования, предъявляемые к методу

В данной работе требуется разработать метод выбора вспомогательных критериев при помощи многокритериального обучения с подкреплением, обладающий следующими характеристиками:

- критерии должны выбираться динамически, во время работы эволюционного алгоритма;
- разработанный метод должен основываться на однокритериальном эволюционном алгоритме;
- число поколений эволюционного алгоритма, необходимое для нахождения оптимального решения при использовании многомерной награды $r = (r_0, r_1 \dots r_n)$ не должно превосходить или должно быть сравнимым с числом поколений, необходимым для нахождения оптимального решения при использовании скалярной награды r_k (метод EA+RL), которая является наименее эффективной из всех одномерных наград $r_i \in r$ для конкретной задачи.

2.3 Описание метода

Дана задача оптимизации целевого критерия. Также дан дискретный набор вспомогательных критериев. Задача решается при помощи обучения с подкреплением, где в качестве среды выступает эволюционный алгоритм. В предлагаемом методе считается, что среда всегда находится в одном состоянии. Для каждого поколения эволюционному алгоритму передается оптимизируемый на данной итерации критерий (функция

приспособленности). Для выбора функции приспособленности на каждой итерации используется обучение с подкреплением. Эволюционный алгоритм возвращает агенту обучения многомерную награду, определение которой будет дано далее.

Опишем схему работы метода EA+MORL. Агент обучения с подкреплением применяет действие к среде: выбирает функцию приспособленности и передает ее эволюционному алгоритму. Эволюционный алгоритм использует переданную ему функцию приспособленности для формирования следующего поколения. После формирования поколения значения награды вычисляются и передаются агенту обучения с подкреплением. Агент на основе полученной награды обновляет оценку выгодности действий, и процесс повторяется до тех пор, пока не будет выполнен критерий останова эволюционного алгоритма. На рисунке 2 приведена общая схема метода EA+MORL.

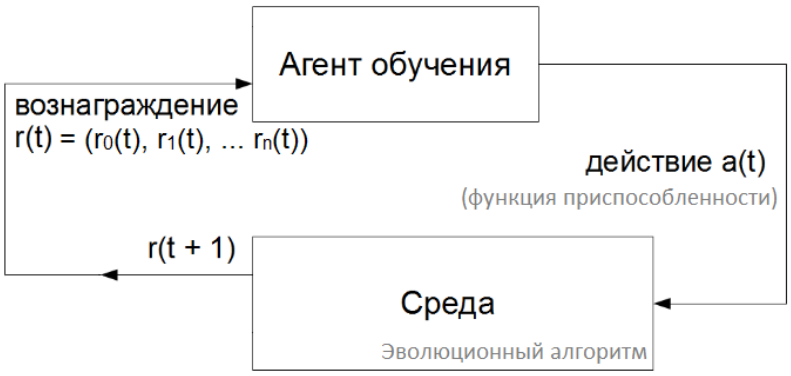

Рисунок 2. Схема метода EA+MORL

2.4 Определение награды

Введем определение понятия награды. Пусть A – множество действий агента обучения с подкреплением. Множество действий агента состоит из целевого критерия оптимизации g и множества вспомогательных критериев H. То есть $A = \{\{g\} \cup H\}$.

Пусть G_i – i-е поколение эволюционного алгоритма. Применение действия агентом обучения к эволюционному алгоритму означает передачу некоторого критерия оптимизации $f_i \in A$ в качестве функции приспособленности, которая используется для поколения G_i. Необходимо определить функцию награды $r = (r_0, r_1 \dots r_n)$, значение которой вычисляется после применения критерия f_i и формирования поколения G_{i+1}.

Функция награды должна обладать следующим свойством: награда должна быть тем выше, чем выше значение целевого критерия оптимизации, которое было получено в результате применения агентом обучения соответствующего действия. Были проведены исследования, которые теоретически и практически показали, что существует ряд функций одномерных наград, которые удовлетворяют этому свойству. Для предлагаемого в данной работе метода были выбраны четыре одномерные функции награды, которые хорошо зарекомендовали себя в методе EA+RL [23, 24, 28].

Приведем описание этих четырех функций награды. Пусть g – целевой критерий оптимизации, z_i – особь из поколения G_i с максимальным значением целевого критерия оптимизации, g_i^{avg} – среднее значение целевого критерия оптимизации по множеству особей поколения G_i, d_i – минимальное расстояние между значениями целевой функции приспособленности в поколении G_i. Запишем с использованием приведенных обозначений определение используемых в предлагаемом методе функций одномерных наград:

- Simple: $g(z_i) - g(z_{i-1})$;
- Mean: $g_i^{\text{avg}} - g_{i-1}^{\text{avg}}$;
- Sign: $\text{sign}\big(g(z_i) - g(z_{i-1})\big)$;
- Dmin: $d_i - d_{i-1}$.

Таким образом, в случае награды Simple награда вычисляется как разность лучших значений целевой функции приспособленности двух последовательных поколений эволюционного алгоритма. В случае функции награды Mean награда вычисляется как разность средних значений целевой функции приспособленности двух идущих подряд поколений эволюционного алгоритма. В случае функции награды Sign, награда вычисляется как знак разности лучших значений целевой функции приспособленности двух последовательных поколений эволюционного алгоритма, то есть как знак разности функции награды Simple. В случае функции награды Dmin награда определяется как разность минимальных расстояний между значениями целевой функции приспособленности двух последовательных поколений эволюционного алгоритма.

Были проведены исследования, которые показали, что для разных задач оптимизации эффективны разные способы определения награды [23, 24, 28]. Причем в общем случае без предварительных исследований невозможно достоверно определить, какой из способов определения награды будет наиболее эффективным. В связи с этим в данной работе в методе EA+MORL предлагается использовать многомерную функцию награды, которая объединяет в себе несколько одномерных функций наград и представляет собой вектор $r = (r_0, r_1 \dots r_n)$, где $r_i \in \{Simple, Mean, Sign, Dmin\}$.

2.5 Выводы по главе 2

Поставлена задача выбора вспомогательных критериев оптимизации с использованием многокритериального обучения с подкреплением. Сформулированы требования, предъявляемые к данному методу.

Описан предлагаемый метод выбора вспомогательных оптимизируемых критериев, основанный на многокритериальном обучении с подкреплением [13]. Приведена общая схема метода, описан способ

определения награды. Предложенный подход является достаточно общим и не накладывает ограничений ни на используемый эволюционный алгоритм, ни на используемый алгоритм обучения с подкреплением. Метод соответствует характеристикам, сформулированным в пункте 2.2.

Глава 3. Экспериментальные исследования предложенного метода

В данной главе описываются экспериментальные исследования и результаты применения предлагаемого метода EA+MORL для решения ряда модельных задач.

Для проведения экспериментальных исследований на языке программирования Java была написана реализация предлагаемого в работе метода, а также описанных далее модельных задач. Для работы с эволюционными алгоритмами использовался фреймворк для эволюционных вычислений «Watchmaker» [18].

Для проверки статистической различимости рассматриваемых методов был проведен непарный статистический тест Уилкоксона [29] по числу поколений эволюционного алгоритма, затраченных на нахождение оптимального решения задачи. Для проведения теста использовался язык программирования для статистической обработки данных R [30].

3.1 Используемый алгоритм обучения с подкреплением

В качестве алгоритма обучения с подкреплением был выбран алгоритм *Q*-обучения (*Q*-learning) с ε-жадной стратегией исследования среды, подробное описание которого приводится в пункте 1.3.1 данной работы. Для применения данного алгоритма к методу EA+MORL были внесены некоторые модификации, позволяющие работать с многомерной функцией награды.

Псевдокод данной реализации приведен в листинге 3. Стоит отметить, что, так как предлагаемый в работе метод рассматривает среду обучения как

имеющую только одно состояние, параметром функции $Q(a)$ является только действие $a \in A$, а состояние среды в данном случае не учитывается.

Листинг 3. Алгоритм многокритериального Q-обучения с ε-жадной стратегией исследования среды для метода EA+MORL

Вход: ε – вероятность выбора случайного действия, α – скорость обучения, γ – дисконтный фактор

Сформировать изначальное поколение эволюционного алгоритма G_0
Инициализировать $Q(a) \leftarrow 0$ для всех $a \epsilon A$
Инициализировать счетчик итераций $k \leftarrow 0$
while (не достигнуто условие останова) **do**
  $p \leftarrow$ случайное вещественное число $\in [0, 1]$
  if ($p \leq \varepsilon$) **then**
    $a \leftarrow$ случайное действие $\in A$
  else
    $\{a_{0\ldots j}\} \leftarrow \{\arg\max_a Q(a)\}$, т. е. $\forall i\ \forall a' : Q(a')_i \leq Q(a)_i$
    $a \leftarrow$ случайное действие $a_i \in \{a_{0\ldots j}\}$
  end if
  Передать функцию приспособленности a эволюционному алгоритму и сформировать поколение G_{k+1}
  Вычислить награду $r = (r_0, r_1 \ldots r_n)$ и передать ее агенту обучения
  for $i = 0 \ldots n$ **do**
    $Q(a)_i \leftarrow Q(a)_i + \alpha(r_i + \gamma(max_{a'} Q(a')_i - Q(a)_i)$
  end for
  $k \leftarrow k + 1$
end while

3.2 Применение метода EA+MORL для решения модельных задач

В данном разделе приводятся описания модельных задач, для которых проводились исследования, и результаты экспериментов, полученные при решении этих задач с помощью метода EA+MORL.

Для проведения экспериментов были выбраны задачи разной сложности [1, 23, 24, 31]:

- ONEMAX, решаемая (1+1) эволюционными стратегиями за $O(n\ logn)$;
- LEADINGONES, решаемая эволюционными алгоритмами за $O(n^2)$;

- XDIVK, решаемая эволюционными алгоритмами за $O(n^{k+1})$;
- H-IFF (Hierarchical-if-and-only-if function), имеющая много локальных оптимумов.

Для полноты экспериментов рассматривались задачи не только со вспомогательными критериями оптимизации, но и с мешающими, то есть такими, оптимизация которых препятствует оптимизации целевого критерия.

Результаты экспериментов сравнивались с результатами, полученными при решении задачи с помощью метода EA+RL, а также эволюционного алгоритма без применения обучения с подкреплением и дополнительных критериев оптимизации при аналогичных настройках эволюционного алгоритма и алгоритма обучения с подкреплением.

Во всех экспериментах использовалась однородная мутация, то есть при применении к особи оператора мутации каждый бит особи инвертировался с заданной вероятностью. В случаях, когда в эксперименте был применен оператор кроссовера, использовался одноточечный кроссовер. В качестве метода отбора особей использовался турнирный отбор. То есть из популяции случайным образом выбирались особи, и с заданной вероятностью p в следующее поколение попадала особь с лучшим значением функции приспособленности. С вероятностью $(1-p)$ метод отбора выбирал менее приспособленную особь.

3.2.1 Задача LEADINGONES

Рассмотрим задачу LEADINGONES. Для данной задачи особь представлена битовой строкой длины n. Пусть x первых бит особи равны единице, а $(x+1)$-й бит равен нулю. Тогда целевой критерий определим как $g = x$. В качестве вспомогательного критерия h будем использовать функцию ONEMAX: число единичных битов особи.

Для проведения эксперимента были выбраны следующие параметры эволюционного алгоритма и алгоритма обучения с подкреплением:

- длина особи: 300;
- размер популяции: 100;
- процент элитизма: 5;
- вероятность мутации: 0,007;
- вероятность кроссовера: 0,1;
- скорость обучения: 0,6;
- дисконтный фактор: 0,1;
- вероятность исследования среды: 0,05.

В таблице 1 приведены результаты эксперимента. Для каждого сочетания метода и типа награды было проведено по 45 запусков. Жирным шрифтом выделены сочетания метода и награды, которые показали наилучшие результаты.

Как видно из таблицы 1, для задачи с целевым критерием LEADINGONES и вспомогательным критерием ONEMAX лучшие результаты показывает предлагаемый в работе метод EA+MORL с векторными наградами $r = (Simple, Mean, Sign, Dmin)$ и $r = (Simple, Mean,\ Dmin)$. Для данной задачи метод EA+MORL в целом показывает лучшие результаты, чем использующий одномерные награды EA+RL или эволюционный алгоритм без вспомогательного критерия оптимизации. Стоит отметить, что для данной задачи метод EA+RL с функцией награды $Mean$ также показывает очень хорошие результаты.

В таблице 2 приведены результаты статистического теста Уилкоксона для метода EA+MORL со всеми типами многомерной награды и с методами

EA+RL, EA. Также в таблице 3 приводятся результаты теста Уилкоксона для сравнения трех самых эффективных для решения данной задачи алгоритмов и всех остальных рассматриваемых алгоритмов. Для результатов, представленных в таблице 3, применялась коррекция по методу Холма. Самые эффективные алгоритмы: с типами наград $(Simple, Mean, Sign, Dmin)$, $(Simple, Mean, Dmin)$ и $(Mean, Dmin)$ статистически различимы со всеми остальными алгоритмами.

Таблица 1. Результаты эксперимента для задачи LEADINGONES (число поколений, потребовавшееся для нахождения оптимального решения)

Алгоритм	**Тип награды**	**Медиана**	**Среднее значение**	**Среднеквадра-тическое отклонение**
EA	-	2949	2950,13	272,89
EA+RL	Simple	2501	2689,47	600,02
EA+RL	Mean	1676	1743,53	465,01
EA+RL	Sign	2768	2804,49	649,44
EA+RL	Dmin	1045	1087,36	198,67
EA+MORL	**Simple, Mean, Sign, Dmin**	**796**	**810,67**	**215,54**
EA+MORL	Simple, Sign, Dmin	1056	1032,60	253,78
EA+MORL	Simple, Mean	1137	1179,71	282,18
EA+MORL	Simple, Sign	2378	2463,22	533,86
EA+MORL	Simple, Dmin	1122	1146,16	250,42
EA+MORL	Simple, Mean, Sign	1150	1191,78	347,80
EA+MORL	**Simple, Mean, Dmin**	**800**	**848,47**	**224,73**
EA+MORL	**Mean, Dmin**	**929**	**890,91**	**213,46**
EA+MORL	Sign, Dmin	976	988,20	229,69
EA+MORL	Mean, Sign	1897	1921,73	305,95

Таблица 2. Результаты теста Уилкоксона для задачи LEADINGONES: сравнение метода EA+MORL и методов EA+RL, EA

p-value	EA+RL (Simple)	EA+RL (Mean)	EA+RL (Sign)	EA+RL (Dmin)	EA
EA+MORL (Simple, Mean, Sign, Dmin)	3,066e-16	3,065e-15	3,067e-16	5,353e-08	3,065e-16
EA+MORL (Simple, Sign, Dmin)	3,068e-16	6,008e-12	3,068e-16	0,2586	3,067e-16
EA+MORL (Simple, Mean)	3,28e-16	4,445e-09	3,749e-16	0,06879	3,067e-16
EA+MORL (Simple, Sign)	0,09806	1,578e-08	0,01825	6,373e-16	5,075e-07
EA+MORL (Simple, Dmin)	3,068e-16	5,447e-10	3,507e-16	0,2637	3,067e-16
EA+MORL (Simple, Mean, Sign)	1,5e-15	2,189e-08	2,2e-16	0,1222	3,506e-16
EA+MORL (Simple, Mean, Dmin)	3,067e-16	8,555e-15	3,068e-16	2,042e-06	3,066e-16
EA+MORL (Mean, Dmin)	3,067e-16	1,068e-14	3,068e-16	0,0001465	3,066e-16
EA+MORL (Sign, Dmin)	3,068e-16	3,266e-13	3,068e-16	0,0558	3,067e-16
EA+MORL (Mean, Sign)	1,741e-11	0,016	2,564e-11	6,809e-16	1,601e-15

Таблица 3. Результаты теста Уилкоксона для задачи LEADINGONES: сравнение самых эффективных алгоритмов и всех остальных рассматриваемых алгоритмов

p-value	EA+MORL (Simple, Mean, Sign, Dmin)	EA+MORL (Simple, Mean, Dmin)	EA+MORL (Mean, Dmin)
EA	4,2910e-15	4,2924e-15	4,2924e-15
EA+RL (Simple)	4,2910e-15	4,2924e-15	4,2924e-15
EA+RL (Mean)	2,7585e-14	7,6995e-14	9,6120e-14
EA+RL (Sign)	4,2910e-15	4,2924e-15	4,2924e-15
EA+RL (Dmin)	2,6765e-07	1,0210e-05	7,3250e-04
EA+MORL (Simple, Mean, Sign, Dmin)		4,6460e-01	1,5684e-01
EA+MORL (Simple, Sign, Dmin)	2,4208e-04	3,1968e-03	6,2600e-02
EA+MORL (Simple, Mean)	3,0359e-08	5,8720e-07	7,5680e-06
EA+MORL (Simple, Sign)	4,2910e-15	4,2924e-15	4,2924e-15
EA+MORL (Simple, Dmin)	1,7432e-08	8,5750e-07	1,8515e-05
EA+MORL (Simple, Mean, Sign)	5,4918e-08	1,9518e-06	1,9698e-05
EA+MORL (Simple, Mean, Dmin)	4,2900e-01		2,3230e-01
EA+MORL (Mean, Dmin)	1,0456e-01	4,6460e-01	
EA+MORL (Sign, Dmin)	6,8910e-04	9,3030e-03	1,5684e-01
EA+MORL (Mean, Sign)	4,2910e-15	4,2924e-15	4,2924e-15

3.2.2 Задача ONEMAX

Рассмотрим задачу ONEMAX. Для данной задачи особь представляется битовой строкой длины n. Пусть x бит особи равны единице, а остальные биты равны нулю. Тогда целевой критерий определим как $g = x$. В качестве дополнительного критерия h будем использовать функцию ZEROMAX, то есть число нулевых битов особи. Данный критерий является мешающим, а не вспомогательным, то есть препятствует оптимизации целевого критерия.

Метод EA+RL, на котором основан предлагаемый в работе метод EA+MORL, успешно распознает этот критерий как мешающий. При проведении данного эксперимента было необходимо убедиться, что метод EA+MORL также позволяет достаточно эффективно исключить из оптимизации мешающий критерий ZEROMAX.

Для проведения эксперимента были выбраны следующие параметры эволюционного алгоритма и алгоритма обучения с подкреплением:

- длина особи: 300;
- размер популяции: 100;
- процент элитизма: 5;
- вероятность мутации: 0,007;
- скорость обучения: 0,6;
- дисконтный фактор: 0,1;
- вероятность исследования среды: 0,01.

В таблице 4 приведены результаты эксперимента. Для каждого сочетания метода и типа награды было проведено по 45 запусков. Жирным шрифтом выделены сочетания метода и награды, которые показали наилучшие результаты.

Как видно из таблицы 4, для задачи с целевым критерием ONEMAX и мешающим критерием ZEROMAX лучшие результаты, как и ожидалось, показывает эволюционный алгоритм без применения дополнительных критериев оптимизации. Вторым по эффективности оказался метод EA+RL, использующий функцию награды $Mean$.

Метод EA+MORL с использованием векторной награды $r = (r_0 r_1 \dots r_n)$ показывает результаты, превосходящие или крайне близкие к результатам

метода EA+RL с использованием скалярной награды $r_i \in r$, показывающей наихудшие результаты по всем одномерным наградам, входящим в векторную награду $r = (r_0 r_1 \dots r_n)$.

В таблице 5 приведены результаты статистического теста Уилкоксона для метода EA+MORL со всем типами многомерной награды и с методами EA+RL, EA. Также в таблице 6 приводятся результаты теста Уилкоксона для сравнения трех самых эффективных для решения данной задачи алгоритмов и всех остальных рассматриваемых алгоритмов. Для результатов, представленных в таблице 6, применялась коррекция по методу Холма. Наиболее эффективный алгоритм метода EA+MORL: с типом награды $(Simple, Mean)$ статистически различим с алгоритмами метода EA+RL и сравнительно неэффективными алгоритмами метода EA+MORL.

Таблица 4. Результаты эксперимента для задачи ONEMAX (число поколений, потребовавшееся для нахождения оптимального решения)

Алгоритм	Тип награды	Медиана	Среднее значение	Среднеквадратическое отклонение
EA	**-**	**423**	**434,8**	**48,99**
EA+RL	Simple	944	960,53	398,04
EA+RL	**Mean**	**470**	**471,93**	**55,39**
EA+RL	Sign	925	909,13	280,42
EA+RL	Dmin	1076	1028,29	258,53
EA+MORL	Simple, Mean, Sign, Dmin	1009	1019,0	444,27
EA+MORL	Simple, Sign, Dmin	1150	1087,29	319,55
EA+MORL	**Simple, Mean**	**657**	**748,89**	**261,32**
EA+MORL	Simple, SIGN	896	911,04	349,49
EA+MORL	Simple, Dmin	1046	1082,96	351,68
EA+MORL	Simple, Mean, Sign	836	857,6	281,74
EA+MORL	Simple, Mean, Dmin	1077	1067,76	347,23
EA+MORL	Mean, Dmin	830	850,2	250,43
EA+MORL	Sign, Dmin	1056	1008,16	342,87
EA+MORL	Mean, Sign	749	787,27	289,14

Таблица 5. Результаты теста Уилкоксона для задачи OneMax: сравнение метода EA+MORL и методов EA+RL, EA

p-value	EA+RL (Simple)	EA+RL (Mean)	EA+RL (Sign)	EA+RL (Dmin)	EA
EA+MORL (Simple, Mean, Sign, Dmin)	0,4627	3,787e-13	0,1579	0,5004	2,919e-14
EA+MORL (Simple, Sign, Dmin)	0,03884	1,61e-14	0,00756	0,296	7,061e-15
EA+MORL (Simple, Mean)	0,004918	2,194e-10	0,004981	3,691e-06	4,954e-13
EA+MORL (Simple, Sign)	0,6426	1,246e-11	1,246e-11	0,8087	3,902e-13
EA+MORL (Simple, Dmin)	0,08418	1,763e-15	0,0236	0,7135	7,768e-16
EA+MORL (Simple, Mean, Sign)	0,2813	7,06e-13	0,2552	0,0006605	5,265e-14
EA+MORL (Simple, Mean, Dmin)	0,1101	7,713e-13	0,01354	0,5971	3,624e-14
EA+MORL (Mean, Dmin)	0,3209	7,488e-13	0,2672	0,0004974	3,409e-14
EA+MORL (Sign, Dmin)	0,2672	1,778e-10	0,08132	0,9903	1,433e-11
EA+MORL (Mean, Sign)	0,02539	1,557e-12	0,01102	2,697e-06	9,151e-14

Таблица 6. Результаты теста Уилкоксона для задачи OneMax: сравнение самых эффективных алгоритмов и всех остальных рассматриваемых алгоритмов

p-value	EA	EA+RL(Mean)	EA+MORL (Simple, Mean)
EA		6,8990e-04	6,9356e-12
EA+RL (Simple)	1,00850e-12	9,3420e-12	2,9508e-02
EA+RL (Mean)	6,89900e-04		2,8522e-09
EA+RL (Sign)	1,61040e-13	1,7952e-12	2,9508e-02
EA+RL (Dmin)	1,19054e-14	8,6073e-14	4,0601e-05
EA+MORL (Simple, Mean, Sign, Dmin)	2,91900e-13	3,7870e-12	9,3200e-04
EA+MORL (Simple, Sign, Dmin)	8,47320e-14	1,9320e-13	1,5756e-05
EA+MORL (Simple, Mean)	1,56080e-12	5,3340e-10	
EA+MORL (Simple, Sign)	1,56080e-12	4,9840e-11	1,1980e-01
EA+MORL (Simple, Dmin)	1,08752e-14	2,4682e-14	7,3690e-05
EA+MORL (Simple, Mean, Sign)	3,68550e-13	6,3540e-12	1,2843e-01
EA+MORL (Simple, Mean, Dmin)	3,06810e-13	6,3540e-12	1,0782e-04
EA+MORL (Mean, Dmin)	3,06810e-13	6,3540e-12	1,2843e-01
EA+MORL (Sign, Dmin)	2,86600e-11	5,3340e-10	1,3286e-03
EA+MORL (Mean, Sign)	5,49060e-13	9,3420e-12	3,6400e-01

3.2.3 Задача XDIVK

Рассмотрим задачу XDIVK. Для данной задачи особь представляется битовой строкой длины n. Также задается целое число k. Пусть x бит особи равны единице, а остальные биты равны нулю. Тогда целевой критерий оптимизации для данной задачи определим как $g = \left\lfloor \frac{x}{k} \right\rfloor$. В качестве

вспомогательного критерия h будем использовать функцию OneMax, то есть число единичных битов особи.

Для проведения эксперимента делитель k был выбран равным пяти. Также были выбраны следующие параметры эволюционного алгоритма и алгоритма обучения с подкреплением:

- длина особи: 100;
- размер популяции: 100;
- процент элитизма: 5;
- вероятность мутации: 0,01;
- скорость обучения: 0,6;
- дисконтный фактор: 0,1;
- вероятность исследования среды: 0,05.

В таблице 7 приведены результаты эксперимента. Для каждого сочетания метода и типа награды было проведено по 45 запусков. Жирным шрифтом выделены сочетания метода и награды, которые показали наилучшие результаты.

Как видно из таблицы 7, для задачи с целевым критерием XdivK и вспомогательным критерием OneMax лучшие результаты показывает метод EA+RL с типом награды $Mean$ и метод EA+MORL с типами награды $r = (Simple, Mean)$ и $r = (Simple, Mean,\ Sign)$.

В таблице 8 приведены результаты статистического теста Уилкоксона для метода EA+MORL со всем типами многомерной награды и с методами EA+RL, EA. Также в таблице 9 приводятся результаты теста Уилкоксона для сравнения трех самых эффективных для решения данной задачи алгоритмов и всех остальных рассматриваемых алгоритмов. Для результатов,

представленных в таблице 9, применялась коррекция по методу Холма. Как видно из таблицы 9, наиболее эффективные алгоритмы метода EA+MORL статистически неразличимы с многими алгоритмами метода EA+MORL с разными типами наград, в частности, с наградой ($Simple, Mean, Sign, Dmin$) и статистически различимы с алгоритмами метода EA+RL.

Таблица 7. Результаты эксперимента для задачи XDIVK (число поколений, потребовавшееся для нахождения оптимального решения)

Алгоритм	**Тип награды**	**Медиана**	**Среднее значение**	**Среднеквадратическое отклонение**
EA	-	37260	45607,96	38648,93
EA+RL	Simple	2322	3070,78	2678,77
EA+RL	**Mean**	**247**	**281,11**	**91,03**
EA+RL	Sign	2584	2689,96	1900,4
EA+RL	Dmin	2551	3346,78	2301,49
EA+MORL	Simple, Mean, Sign, Dmin	2126	1691,2	718,56
EA+MORL	Simple, Sign, Dmin	3371	4131,38	2903,92
EA+MORL	**Simple, Mean**	**1867**	**1740,27**	**628,96**
EA+MORL	Simple, Sign	2226	2955,56	2586,83
EA+MORL	Simple, Dmin	2989	3709,56	2849,83
EA+MORL	**Simple, Mean, Sign**	**1878**	**1643,64**	**701,99**
EA+MORL	Simple, Mean, Dmin	2141	1643,29	776,86
EA+MORL	Mean, Dmin	2011	1625,62	624,96
EA+MORL	Sign, Dmin	2523	2925,71	2378,22
EA+MORL	Mean, Sign	2010	1611,67	786,88

Таблица 8. Результаты теста Уилкоксона для задачи XDIVK: сравнение метода EA+MORL и методов EA+RL, EA

p-value	EA+RL (Simple)	EA+RL (Mean)	EA+RL (Sign)	EA+RL (Dmin)	EA
EA+MORL (Simple, Mean, Sign, Dmin)	0,01618	7,063e-15	7,063e-15	0,002097	2,2e-16
EA+MORL (Simple, Sign, Dmin)	0,03404	5,225e-16	0,01548	0,2211	4,906e-15
EA+MORL (Simple, Mean)	0,02005	9,472e-16	0,004619	0,002166	2,2e-16
EA+MORL (Simple, Sign)	0,7714	1,47e-12	0,968	0,2121	2,2e-16
EA+MORL (Simple, Dmin)	0,1355	2,071e-14	0,1339	0,6529	2,542e-16
EA+MORL (Simple, Mean, Sign)	0,008123	1,61e-14	0,001581	0,0006905	6,219e-15
EA+MORL (Simple, Mean, Dmin)	0,008933	2,498e-14	0,001766	0,0005526	7,534e-15
EA+MORL (Mean, Dmin)	0,002942	8,274e-15	0,001766	0,0005283	2,873e-15
EA+MORL (Sign, Dmin)	0,9389	1,512e-14	0,7683	0,2637	1,586e-13
EA+MORL (Mean, Sign)	0,006948	2,658e-14	0,001455	0,0003669	7,064e-15

Таблица 9. Результаты теста Уилкоксона для задачи XDIVK: сравнение самых эффективных алгоритмов и всех остальных рассматриваемых алгоритмов

p-value	EA+RL(Mean)	EA+MORL (Simple, Mean, Sign)	EA+MORL (Simple, Mean)
EA	4,29240e-15	8,7066e-14	3,08000e-15
EA+RL (Simple)	1,62450e-12	6,4984e-02	1,60400e-01
EA+RL (Mean)		2,0930e-13	1,23136e-14
EA+RL (Sign)	1,78820e-12	1,4229e-02	4,15710e-02
EA+RL (Dmin)	5,20520e-15	6,9050e-03	2,16600e-02
EA+MORL (Simple, Mean, Sign, Dmin)	7,06300e-14	1,0000e+00	1,00000e+00
EA+MORL (Simple, Sign, Dmin)	6,27000e-15	3,6612e-06	4,42440e-06
EA+MORL (Simple, Mean)	1,04192e-14	1,0000e+00	
EA+MORL (Simple, Sign)	1,78820e-12	5,5008e-01	1,00000e+00
EA+MORL (Simple, Dmin)	1,24260e-13	6,7034e-05	2,00970e-04
EA+MORL (Simple, Mean, Sign)	1,20960e-13		1,00000e+00
EA+MORL (Simple, Mean, Dmin)	1,24900e-13	1,0000e+00	1,00000e+00
EA+MORL (Mean, Dmin)	7,44660e-14	9,6250e-01	1,00000e+00
EA+MORL (Sign, Dmin)	1,20960e-13	1,7591e-01	3,87030e-01
EA+MORL (Mean, Sign)	1,24900e-13	1,0000e+00	1,00000e+00

3.2.4 Задача H-IFF со вспомогательными критериями

Рассмотрим задачу H-IFF (Hierarchical-if-and-only-if function). Для данной задачи особь представляется битовой строкой длины *n*. Функции H-IFF подается на вход битовая строка $B = b_1 b_2 \dots b_n$. Данная битовая строка интерпретируется как двоичное дерево, узлами и листьями которого являются блоки битовой строки. Вершиной дерева является исходная битовая строка B. Потомками каждого узла являются левая и правая половины родительского блока: B_L и B_R. Функция приспособленности вычисляется как

сумма длин блоков, состоящих только из нулей или только из единиц. Ниже приведена формула (4.1) вычисления функции приспособленности.

$$f(B) = \begin{cases} 1, \text{ если } |B| = 1 \\ |B| + f(B_L) + f(B_R), \text{ если } \forall i: b_i = 0 \text{ или } \forall i: b_i = 1 \\ f(B_L) + f(B_R) \text{ в остальных случаях} \end{cases} \quad (4.1)$$

Стоит отметить, что существует два оптимальных решения задачи H-IFF, а именно: строка, состоящая из нулевых битов, и строка, состоящая из единичных битов. Дадим определение модификации исходной задачи, которая называется MH-IFF. Данная модификация использует два критерия f_0 и f_1, каждый из которых учитывает только блоки, состоящие из нулевых битов, и блоки, состоящие из единичных битов, соответственно. Ниже приведена формула (4.2) вычисления этих критериев.

$$f_n(B) = \begin{cases} 0, \text{ если } |B| = 1 \text{ и } b_1 \neq n \\ 1, \text{ если } |B| = 1 \text{ и } b_1 = n \\ |B| + f_n(B_L) + f_n(B_R), \text{ если } \forall i: b_i = n \\ f_n(B_L) + f_n(B_R) \text{ в остальных случаях} \end{cases} \quad (4.2)$$

Таким образом, целевым критерием задачи H-IFF является $g = f$. Было показано, что для решения исходной задачи эффективно использовать вспомогательные критерии $H = \{h_i = f_i\}$.

Для проведения эксперимента были выбраны следующие параметры эволюционного алгоритма и алгоритма обучения с подкреплением:

- длина особи: 64;
- размер популяции: 100;
- процент элитизма: 5;
- вероятность мутации: 0,01;
- скорость обучения: 0,75;

- дисконтный фактор: 0,1;
- вероятность исследования среды: 0,01.

В таблице 10 приведены результаты эксперимента. Для каждого сочетания метода и типа награды было проведено по 45 запусков. Если за триста тысяч поколений эволюционного алгоритма оптимальное решение не было найдено, выполнение алгоритма останавливалось. Жирным шрифтом выделены сочетания метода и награды, которые показали наилучшие результаты.

Как видно из таблицы 10, для данной задачи наилучшие результаты показывает метод EA+RL с типом награды $Mean$. Метод EA+RL с типами награды $Simple$ и $MS1$ показывает хорошие результаты в смысле медианы, однако оптимальное решение находится лишь в 73% и 67% случаев соответственно. Метод EA+MORL для всех векторных наград, включающих в себя как одну из компонент награду $Mean$, находит лучшее решение в 100% случаев. По данным результатам можно сделать вывод, что включение в векторную награду эффективной для конкретной задачи награды приводит к нахождению оптимального решения за сравнительно небольшое число поколений.

В таблице 11 приведены результаты теста Уилкоксона для сравнения трех самых эффективных для решения данной задачи алгоритмов и всех остальных рассматриваемых алгоритмов, при использовании которых оптимальное решение было найдено в ста процентах случаев. Эффективными алгоритмами в данном случае также считаются именно те алгоритмы, которые позволили найти оптимальное решение в ста процентах случаев. Для результатов, представленных в таблице 11, применялась коррекция по методу Холма.

Таблица 10. Результаты эксперимента для задачи H-IFF со вспомогательными критериями (число поколений, потребовавшееся для нахождения оптимального решения)

Алгоритм	**Тип награды**	**Медиана**	**Среднее значение**	**Среднеквадратическое отклонение**	**% решения**
EA	-	-	-	-	0
EA+RL	Simple	2094	-	-	73
EA+RL	**Mean**	**562**	**765,07**	**785,49**	**100**
EA+RL	Sign	1353	-	-	67
EA+RL	Dmin	-	-	-	33
EA+MORL	Simple, Mean, Sign, Dmin	4766	4709,44	1799,42	100
EA+MORL	Simple, Sign, Dmin	13957	-	-	55
EA+MORL	Simple, Mean	4518	4404,07	2193,69	100
EA+MORL	Simple, Sign	2645	-	-	71
EA+MORL	Simple, Dmin	3630	-	-	73
EA+MORL	Simple, Mean, Sign	3879	4396,93	1791,25	100
EA+MORL	Simple, Mean, Dmin	3754	4463,62	2011,34	100
EA+MORL	Mean, Dmin	2530	2776,38	717,5	100
EA+MORL	Sign, Dmin	19338	-	-	55
EA+MORL	Mean, Sign	4252	4385,89	2003,79	100

Таблица 11. Результаты теста Уилкоксона для задачи H-IFF со вспомогательными критериями: сравнение самых эффективных алгоритмов и всех остальных рассматриваемых алгоритмов, которые позволили найти оптимальное решение в ста процентах случаев

p-value	EA+RL(Mean)	EA+MORL (Mean, Dmin)	EA+MORL (Simple, Mean, Dmin)
EA+RL (Mean)		1.3200e-15	1,1682e-14
EA+MORL (Simple, Mean, Sign, Dmin)	1,320e-15	4.1615e-08	1,0000e+00
EA+MORL (Simple, Mean)	1,320e-15	4.6300e-05	1,0000e+00
EA+MORL (Simple, Mean, Sign)	1,320e-15	1.1948e-06	1,0000e+00
EA+MORL (Simple, Mean, Dmin)	3,894e-15	2.8401e-06	
EA+MORL (Mean, Dmin)	1,320e-15		4,7335e-06
EA+MORL (Mean, Sign)	1,716e-14	2.8401e-06	1,0000e+00

3.2.5 Задача H-IFF со вспомогательными и мешающим критериями

Рассмотрим задачу H-IFF с целевым критерием $g = f$ и дополнительными критериями $H = \{h_0 = f_0,\ h_1 = f_1,\ h_2\}$. Формальное описание задачи, целевого критерия и вспомогательных критериев h_0, h_1 было дано в пункте 4.2.4. Дополнительно определим мешающий критерий h_2 как число совпадающих битов особи длины n, для которой вычисляется значение критерия, и битовой строкой $c = 101010 \ldots 1010$ длины n.

Для проведения эксперимента были выбраны следующие параметры эволюционного алгоритма и алгоритма обучения с подкреплением:

- длина особи: 64;
- размер популяции: 100;
- процент элитизма: 5;
- вероятность мутации: 0,01;
- скорость обучения: 0,75;
- дисконтный фактор: 0,1;
- вероятность исследования среды: 0,01.

В таблице 12 приведены результаты эксперимента. Для каждого сочетания метода и типа награды было проведено по 45 запусков. Если за триста тысяч поколений эволюционного алгоритма оптимальное решение не было найдено, выполнение алгоритма останавливалось. Жирным шрифтом выделены сочетания метода и награды, которые показали наилучшие результаты.

Как видно из таблицы 12, для данной задачи, как и в случае использования только целевого критерия g и вспомогательных критериев h_0,

h_1 (пункт 4.2.4), наилучшие результаты показывает метод EA+RL с типом награды $Mean$. В смысле эффективности метода EA+MORL для $\forall r = (r_0, r_1 \dots r_n)$ по сравнению с методом EA+RL с типами наград $\in r$ результаты также аналогичны результатам, полученным при проведении эксперимента для задачи H-IFF с использованием только целевого и вспомогательных критериев.

В таблице 13 приведены результаты теста Уилкоксона для сравнения трех самых эффективных для решения данной задачи алгоритмов и всех остальных рассматриваемых алгоритмов, при использовании которых оптимальное решение было найдено в ста процентах случаев. Эффективными алгоритмами, как и для задачи H-IFF с только вспомогательными критериями, считаются именно те алгоритмы, которые позволили найти оптимальное решение в ста процентах случаев. Для результатов, представленных в таблице 13, применялась коррекция по методу Холма.

Таблица 12. Результаты эксперимента для задачи H-IFF со вспомогательными и мешающим критериями (число поколений, потребовавшееся для нахождения оптимального решения)

Алгоритм	**Тип награды**	**Медиана**	**Среднее значение**	**Среднеквадра-тическое отклонение**	**% решения**
EA	-	-	-	-	0
EA+RL	Simple	1179	-	-	64
EA+RL	**Mean**	**542**	**715,93**	**610,41**	**100**
EA+RL	Sign	11084	-	-	60
EA+RL	Dmin	-	-	-	40
EA+MORL	Simple, Mean, Sign, Dmin	6149	5919,67	2172,51	100
EA+MORL	Simple, Sign, Dmin	4806	-	-	53
EA+MORL	Simple, Mean	4238	4505,27	1860,75	100
EA+MORL	Simple, Sign	3928	-	-	60
EA+MORL	Simple, Dmin	-	-	-	47
EA+MORL	Simple, Mean, Sign	4118	4718,67	2062,41	100
EA+MORL	Simple, Mean, Dmin	5161	5121,49	1953,41	100
EA+MORL	Mean, Dmin	3006	3186,11	609,09	100
EA+MORL	Sign, Dmin	-	-	-	38
EA+MORL	Mean, Sign	4433	4704,49	2359,51	100

Таблица 13. Результаты теста Уилкоксона для задачи H-IFF со вспомогательными и мешающим критериями: сравнение самых эффективных алгоритмов и всех остальных рассматриваемых алгоритмов, которые позволили найти оптимальное решение в ста процентах случаев

p-value	EA+RL(Mean)	EA+MORL (Mean, Dmin)	EA+MORL (Simple, Mean, Sign)
EA+RL (Mean)		1,4202e-14	1,2432e-13
EA+MORL (Simple, Mean, Sign, Dmin)	2,4868e-14	2,2145e-10	2,3708e-02
EA+MORL (Simple, Mean)	8,7560e-14	2,1693e-05	1,0000e+00
EA+MORL (Simple, Mean, Sign)	6,2160e-14	2,1693e-05	
EA+MORL (Simple, Mean, Dmin)	4,3692e-15	7,7240e-07	1,0000e+00
EA+MORL (Mean, Dmin)	1,1835e-14		3,6155e-05
EA+MORL (Mean, Sign)	2,1450e-13	2,2290e-05	1,0000e+00

3.3 Выводы по главе 3

Дано описание модифицированного для предлагаемого метода алгоритма *Q*-обучения, а также модельных задач, на которых проверялась эффективность метода. Приведены результаты экспериментов, в ходе которых предлагаемый метод применялся к описанным модельным задачам. Проведено сравнение эффективности с методом EA+RL, на котором основывается предлагаемый метод.

Экспериментально показано, что предлагаемый метод EA+MORL с векторной наградой $r = (r_0, r_1 \dots r_n)$ показывает эффективность, лучшую или сравнимую с эффективностью метода EA+RL с использованием в качестве награды $r_i \in r$, где r_i – одномерная награда, при использовании которой метод EA+RL показывает худшую эффективность по сравнению с использованием в качестве одномерной награды $\forall r_j \in r,\ r_j \neq r_i$. Таким образом, показано, что при отсутствии предварительной информации о том,

какая одномерная награда является наиболее эффективной для конкретной задачи, использование векторной награды, включающей в себя все потенциально используемые одномерные награды, оказывается предпочтительным.

Заключение

В работе предложен метод, основанный на многокритериальном обучении с подкреплением. Предлагаемый метод позволяет решать скалярную задачу оптимизации, для которой также дан набор дополнительных критериев оптимизации, при помощи эволюционных алгоритмов. Данный метод позволяет эффективно решать задачу без предварительных исследований того, какая функция награды алгоритма обучения с подкреплением будет наиболее эффективной.

Эффективность данного метода экспериментально проверена на ряде модельных задач разной сложности. Проведено сравнение предлагаемого метода с эволюционными алгоритмами, оптимизирующими только целевой критерий, без использования дополнительных критериев оптимизации, и с методом EA+RL, который основан на однокритериальном обучении с подкреплением. Результаты эксперимента проверены на статистическую значимость.

Список использованных источников

1. *Афанасьева А. С.* Выбор вспомогательных оптимизируемых величин для ускорения процесса оптимизации с помощью машинного обучения. Санкт-Петербург. 2012. http://is.ifmo.ru/diploma-theses/2012/bachelor/afanasyeva/thesis.pdf
2. *C. Segura, C. A. C. Coello, G. Miranda, C. Leon.* Using multi-objective evolutionary algorithms for single-objective optimization. 4OR, vol. 11, issue 3, pp. 201-228. 2013.
3. *Joshua D. Knowles, Richard A. Watson, David W. Corne.* Reducing Local Optima in Single-Objective Problems by Multi-objectivization. Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization. EMO ’01, pp. 269-283. London. 2001.
4. *M. Mitchell.* An introduction to Genetic Algorithms. MIT Press. Cambridge. 1996.
5. *A. E. Eiben, J. E. Smith.* Introduction to Evolutionary Computing. Springer. 2007.
6. *Ю. Г. Евтушенко и В. Г. Жадан*. Точные вспомогательные функции в задачах оптимизации. Журнал вычислительной математики и математической физики, том 30, стр. 43-57. 1990.
7. *R. S. Sutton, A. G. Barto.* Reinforcement Learning: An Introduction. MIT Press. Cambridge. 1998.
8. *Скобцов Ю. А.* Основы эволюционных вычислений. ДонНТУ. Донецк. 2008.
9. *A. E. Eiben, M. Horvath, W. Kowalczyk, M. C. Schut.* Reinforcement Learning for online control of evolutionary algorithms. Proceedings of the 4th international conference on Engineering self-organising systems ESOA’06. Pp. 151-160. 2006.

10. *D. Brockhoff, T. Friedrich, N. Hebbinghaus, C. Klein, F. Neumann, E. Zitzler*. On the Effects of Adding Objectives to Plateau Functions. Transactions of Evolutionary Computation. Vol. 13, pp. 591-603. 2009.
11. *F. Neumann, I. Wegener*. Can Single-Objective Optimization Profit from Multiobjective Optimization? Mutiobjective Problem Solving from Nature, Natural Computing Series. Pp. 115-130. 2008.
12. *M. Buzdalov, A. Buzdalova, I. Petrova.* Generation of Tests for Programming Challenge Tasks Using Multi-Objective Optimization. Proceedings of Genetic and Evolutionary Computation Conference Companion. Pp. 1655-1658. ACM, 2013.
13. *T. Brys, A. Harutyunyan, P. Vrancx, M. E. Taylor, D. Kudenko, A. Nowe.* Muti-Objectivization of reinforcement learning problems by reward shaping. 2014 International Joint Conference on Neural Networks. Pp. 2315-2322. 2014.
14. *M. T. Jensen.* Helper-Objectives: Using Multi-Objective Evolutionary Algorithms for Single-Objective Optimization: Evolutionary Computation Combinatorial Optimization. Journal of Mathematical Modelling and Algorithms. Vol. 3, pp. 323-347. 2004.
15. *G. Karaforias, M. Hoogendoorn, A. Eiben.* Parameter control in evolutionary algorithms: Trends and challenges. Evolutionary Computation, IEEE Transactions on. Pp. 99. 2014.
16. *D. F. Lochtefeld, F. W. Ciarallo.* Multiobjectivization via helperobjectives with the tunable objectives problem. IEEE Trans. Evolutionary Computation. Vol. 16, issue 3, pp. 373-390. 2012.
17. *J. Handl, S. C. Lovell, J. D Knowles.* Multiobjectivisation by Decomposition of Scalar Cost Functions. Parallel Problem Solving from Nature, Lecture Notes in Computer Science. Vol. 5199, pp. 31-40. 2008.
18. Watchmaker framework for evolutionary computation. http://watchmaker.uncommons.org

19. *S. Picek, D. Jakobovic.* From Fitness Landscape to Crossover Operator Choice. Proceedings of the 2014 conference on Genetic and evolutionary computation. Pp. 815-822. 2014.
20. *R. Ugolotti, S. Cagnoni.* Analysis of Evolutionary Algorithms using Multi-Objective Parameter Tuning. Proceedings of the 2014 conference on Genetic and evolutionary computation. Pp. 1343-1350. 2014.
21. *D. F. Lochtefeld, F. W. Cirallo.* Helper-Objective Optimization Strategies for the Job-Shop Scheduling Problem. Applied Soft Computing. Vol. 11, issue 6, pp. 4161-4174. 2011.
22. *J. A. Soria-Alcaraz, G. Ochoa, M. Carpio, H. Puga.* Evolvability Metrics in Adaptive Operator Selection. Proceedings of the 2014 conference on Genetic and evolutionary computation. Pp. 1327-1334. 2014.
23. *Мейнстер Д. Л.* Исследование различных способов определения состояния и награды в методе EA+RL на примере модельных задач. Санкт-Петербург. 2014.
24. *Подгорнова Е. С.* Исследование различных способов определения награды в методе EA+RL на примере модельных задач. Санкт-Петербург. 2014.
25. *Y. Sakurai, K. Takada, T. Kawabe, S. Tsuruta.* A method to control parameters of evolutionary algorithms by using reinforcement learning. Proceedings of the 2010 Sixth International Conference on Signal-Image Technology and Internet Based Systems. Pp. 74-79. 2010.
26. *S. Muller, N. N. Schraudolph, P. D. Koumoutsakos*. Step size adaptation in evolution strategies using reinforcement learning. Proceedings of the Congress on Evolutionary Computation. Pp. 151-156. 2002.
27. R. M. Everson, J. E. Pettinger. Controlling genetic algorithms with reinforcement learning. Proceedings of the Genetic and Evolutionary Computation Conference. Pp. 692. 2002.

28. *T. Ulrich, L. Thiele*. Maximizing Population Diversity in Single-Objective Optimization. Proceedings of the 13th annual conference on Genetic and evolutionary computation. Pp. 641-648. 2011.
29. *D. Molina, F. Herrera, J. Derrac, S. Garcia.* A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. 2011.
30. R: A language and environment for statistical computing. http://www.R-project.org. 2013.
31. *M. Buzdalov, A. Buzdalova*. OneMax Helps Optimizing XdivK: Theoretical Runtime Analysis for RMHC and EA+RL. GECCO 2014.