Университет ИТМО

Факультет информационных технологий и программирования
Кафедра компьютерных технологий

Буздалова Арина Сергеевна

Выбор вспомогательных оптимизируемых величин для повышения эффективности эволюционных алгоритмов с помощью обучения с подкреплением

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

Санкт-Петербург
2014

Содержание

Введение **5**

Глава 1. Обзор предметной области **6**

1.1 Эволюционные алгоритмы 6
1.2 Методы использования дополнительных критериев 7
1.3 Обучение с подкреплением 9
1.3.1 Общие понятия обучения с подкреплением 9
1.3.2 Способы оценки суммарной награды 10
1.3.3 Стратегии исследования среды 11
1.4 Настройка эволюционных алгоритмов с помощью обучения с подкреплением 12
1.5 Выводы по главе 1 12

Глава 2. Задача выбора вспомогательных оптимизируемых величин **14**

2.1 Постановка задачи 14
2.2 Требования, предъявляемые к методу выбора 14
2.3 Выводы по главе 2 15

Глава 3. Метод выбора вспомогательных оптимизируемых величин EA+RL **16**

3.1 Описание метода 16
3.2 Определение награды 17
3.3 Определение состояния 19
3.4 Выводы по главе 3 19

Глава 4. Теоретический анализ предлагаемого метода **20**

4.1 Задача с мешающим критерием 21

4.1.1 Схема доказательства 21
4.1.2 Лемма об обучении 22
4.1.3 Цепь Маркова 24
4.1.4 Упрощение цепи Маркова 25
4.1.5 Асимптотическая оценка числа итераций EA+RL . . 27
4.2 Задача с помогающим критерием 30
4.2.1 Схема доказательства 31
4.2.2 Устранение рекурсии 34
4.2.3 Модификация выражений 35
4.2.4 Асимптотическая оценка числа вычислений функции приспособленности 36
4.2.5 Отношение числа вычислений функции приспособленности 38
4.3 Выводы по главе 4 39

Глава 5. Применение метода EA+RL к задаче о генерации тестов 40

5.1 Задача о генерации тестов 40
5.2 Описание эксперимента 41
5.2.1 Эволюционный алгоритм 41
5.2.2 Метод, с которым производится сравнение 42
5.2.3 Параметры обучения с подкреплением 42
5.2.4 Вычисление среднего числа поколений 42
5.3 Результаты 43
5.4 Выводы по главе 5 44

Заключение 46

Список литературы 47

Введение

В данной работе предлагается метод динамического выбора вспомогательных оптимизируемых величин, которые для краткости далее будем называть дополнительными критериями. Предполагается, что набор дополнительных критериев задан заранее, однако свойства критериев неизвестны. Ставится задача оптимизации некоторого целевого критерия с помощью эволюционного алгоритма. Использование дополнительных критериев может позволить найти точку оптимума целевого критерия за меньшее число поколений эволюционного алгоритма. Задача оптимизации самих дополнительных критериев не ставится, они используются лишь для повышения эффективности оптимизации целевого критерия.

Предлагается метод выбора дополнительных критериев с помощью обучения с подкреплением. Проводится теоретический анализ метода для двух модельных задач. Приводятся результаты применения метода к задаче о генерации тестов против решений олимпиадных задач по программированию.

Глава 1. Обзор предметной области

В данном разделе описывается рассматриваемая предметная область. Дается краткое описание эволюционных алгоритмов, приводится обзор подходов, позволяющих повысить их эффективность с использованием вспомогательных оптимизируемых величин. Также излагаются основные идеи обучения с подкреплением, на котором основан предлагаемый в работе метод, и приводится обзор других методов, позволяющих настраивать эволюционные алгоритмы с помощью обучения с подкреплением.

1.1. Эволюционные алгоритмы

Существуют практически значимые задачи оптимизации, в которых информация о градиенте оптимизируемой функции недоступна, в частности, задачи дискретной оптимизации. К ним относятся, например, задачи комбинаторной оптимизации (например, построение расписаний), а также задачи поисковой инженерии программного обеспечения, где требуется оптимизировать программы или входные данные для них. Точные алгоритмы решения многих из этих задач являются слишком неэффективными для их применения в практических приложениях, а для некоторых классов задач таких алгоритмов доказуемо не существует. Для получения решений приемлемого качества можно использовать эволюционные алгоритмы [1, 2].

Для решения задачи оптимизации некоторого критерия с помощью эволюционного алгоритма необходимо переформулировать ее в терминах задачи оптимизации *функции приспособленности* (ФП), позволяющей сравнивать точки из пространства поиска. Далее будем считать термины "критерий оптимизации"и "функция приспособленности"синонимами, если не указано иное. Точки из пространства поиска, являющиеся кандидатами в решение задачи оптимизации, кодируются в виде *особей* эволюционного алгоритма. Текущий набор особей называется *поколением*.

Во время выполнения эволюционного алгоритма к особям текуще-

го поколения применяются эволюционные операторы *скрещивания (кроссовера)* и *мутации*, в результате чего образуются новые особи-*потомки*. Формирование следующего поколение выполняется путем отбора особей, основанного на их *приспособленности*, то есть, на том, какие значения принимает функция приспособленности, посчитанная на данных особях. К следующему поколению применяются эволюционные операторы и процесс повторяется.

Наиболее распространенными условиями останова эволюционного алгоритма являются нахождение точки оптимума функции приспособленности или выполнение фиксированного числа итераций [1]. Второе условие удобно в тех случаях, когда значение оптимума функции приспособленности неизвестно или время, необходимое для его нахождения, слишком велико.

В качестве меры эффективности эволюционного алгоритма можно рассматривать два подхода, соответствующих различным условиям останова: число поколений, необходимое для нахождения точки оптимума функции приспособленности; или же значение функции приспособленности, соответствующее особям последнего сгенерированного поколения. Обычно производится некоторое число запусков эволюционного алгоритма и названные величины усредняются [3].

1.2. Методы использования дополнительных критериев

Эффективность однокритериальной оптимизации может быть повышена с использованием дополнительных критериев. Насколько известно автору, первое исследование возможности использования дополнительных критериев для повышения эффективности оптимизации, относится к работе [4]. Однако в этой работе рассматривается оптимизация непрерывных функций, производящаяся классическими методами, предполагающими возможность вычисления производной.

Рассмотрим современное состояние исследований в области эволюционных алгоритмов, с помощью которых проводится оптимизация дискретных функций, для которых невозможно вычислить производную. Для сравнения потенциальных решений задачи оптимизации в классических эволюционных алгоритмах используется функция приспособленности. Будем называть эту функцию приспособленности целевой функцией приспособленности, а соответствующий ей критерий оптимизации — целевым критерием оптимизации.

Известны исследования по повышению эффективности эволюционных алгоритмов с помощью дополнительных критериев [5—7]. В ряде существующих подходов дополнительные критерии создаются вручную и оптимизируются одновременно с помощью многокритериальных эволюционных алгоритмов [8—10].

Также существует подход, в котором дополнительные критерии оптимизируются не одновременно, а выбираются динамически в процессе работы эволюционного алгоритма [11]. Еще одной особенностью этого метода является то, что дополнительные критерии оптимизируются одновременно с целевым критерием. Дополнительные критерии, используемые для текущего этапа оптимизации, выбираются случайным образом из списка всех доступных дополнительных критериев. Подобный подход не позволяет учесть особенности решаемой задачи оптимизации. В других работах предлагается использовать частные подходы: порядок выбора дополнительных критериев зависит от решаемой задачи [12, 13]. Такие подходы являются недостаточно общими и не могут быть модифицированы для решения задач, отличных от изначально решаемой задачи.

Общим недостатком перечисленных подходов повышения эффективности оптимизации целевого критерия с помощью дополнительных является то, что в них используются многокритериальные эволюционные алгоритмы, основанные на сравнении по Парето. Итерации таких алгоритмов имеют большую вычислительную сложность, чем итерации однокритери-

альных эволюционных алгоритмов, которые будут использоваться в предлагаемом проекте. Также в существующих подходах предполагается, что использование любого из дополнительных критериев должно приводить к повышению эффективности оптимизации целевого критерия, по крайней мере, на некотором этапе оптимизации. Убедиться в выполнении этого требования достаточно сложно. Также сложно разрабатывать соответствующие критерии [8, 11]. Таким образом, как отмечается в выводах к работе [11], необходим метод, позволяющий автоматически выбирать эффективные дополнительные критерии и игнорировать неэффективные. Такой метод можно было бы применять, в том числе, к выбору критериев, сгенерированных автоматически, свойства которых заранее не известны.

1.3. Обучение с подкреплением

Предлагаемый в данной работе метод выбора дополнительных критериев основан на обучении с подкреплением. Опишем основные идеи алгоритмов обучения с подкреплением.

1.3.1. Общие понятия обучения с подкреплением

Обучение с подкреплением используется для решения задач взаимодействия с динамической средой. Общая схема алгоритмов обучения с подкреплением такова:*агент* применяет *действие* к *среде*, после чего получает от среды *награду* и некоторое представление текущего *состояния* среды [14]. Агент использует полученную информацию для обучения и цикл взаимодействия со средой повторяется. Задачей агента является максимизация суммарной награды. Схема взаимодействия агента со средой представлена на рис. 1.1, где t — номер итерации алгоритма обучения с подкреплением. Далее введенные понятия будут описаны более подробно.

Действие может как выбираться из некоторого конечного дискретного набора, так и, например, принадлежать множеству векторов вещественных чисел R^n [15, 16]. В данной работе в качестве действия рассмат-

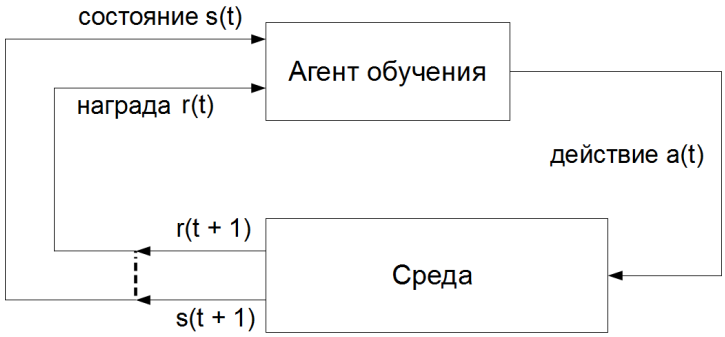

Рис. 1.1: Схема взаимодействия агента со средой в алгоритмах обучения с подкреплением

ривается выбор критерия оптимизации из некоторого конечного заранее заданного набора. Поэтому будем рассматривать алгоритмы обучения с подкреплением, предназначенные для работы с конечным дискретным набором действий.

Состояние также может быть как элементом конечного дискретного набора, так и представляться вектором вещественных чисел [15]. В данной работе предлагаются способы определения состояния эволюционного алгоритма, позволяющие получить конечный дискретный набор состояний.

Награда обычно представляет собой целое или вещественное число [14]. Далее будем считать, что награда вещественная. Положительная награда свидетельствует об эффективности действий агента, отрицательная — о неэффективности.

1.3.2. Способы оценки суммарной награды

Обозначим *мгновенную* награду, полученную на $t-$й итерации алгоритма обучения с подкреплением как r_t. Задача максимизации суммарной награды может быть формализована различными способами, соответствующими модели конечного горизонта ($\max E(\sum_{t=0}^{h} r_t)$, где h — число итераций алгоритма), бесконечного горизонта ($\max E(\sum_{t=0}^{\infty} \gamma^t r_t)$, где $0 < \gamma < 1$ — дисконтный фактор) и достаточно редко использующейся модели среднего вознаграждения ($\max \lim_{h\to 0} E(\frac{1}{h} \sum_{t=0}^{h} r_t)$) [17].

В данной работе награда формируется эволюционным алгоритмом, время завершения которого не всегда известно заранее. Например, вре-

мя завершения работы эволюционного алгоритма практически невозможно точно определить в случае, когда условием останова является достижение оптимального значения функции приспособленности [1]. Поэтому модель конечного горизонта неприменима. Предлагается использовать модель бесконечного горизонта.

1.3.3. Стратегии исследования среды

В алгоритмах обучения с подкреплением применение накопленного опыта совмещается с необходимостью исследовать еще не посещенные состояния среды. Эти задачи зачастую конфликтуют. Существуют различные стратегии совмещения применения опыта с исследованием среды.

Обозначим как $Q(s, a)$ ожидаемую эффективность применения действия a в состоянии s. Одна из простейших стратегий заключается в том, чтобы не использовать накопленный опыт, а всегда выбирать действие, наиболее эффективное в текущем состоянии. Такая стратегия называется *жадной* [14, 17]. Этот подход может привести к тому, что будет найдена локально оптимальная стратегия поведения, не позволяющая в полной мере максимизировать суммарную награду.

Существует также $\varepsilon-$жадная стратегия исследования среды [14, 17], при которой с вероятностью ε агент выбирает случайное действие, а в остальных случаях придерживается жадной стратегии. Данная стратегия позволяет избежать остановки в локальном оптимуме. Однако выбор случайного действия теряет смысл, когда агент полностью исследовал среду.

Еще одним способом исследования среды является исследование по Больцману [17]. В соответствии с этим способом, вероятность выбора действия a в состоянии s составляет $p(a) = \frac{e^{Q(s,a)/T}}{\sum_{a' \in A} e^{Q(s,a')/T}}$, где T — температура, параметр, убывающий со временем. Данный подход лишен недостатков описанных выше стратегий, однако его эффективность зависит от подбора параметра скорости убывания температуры.

Перечисленные стратегии могут применяться в различных алгорит-

мах обучения с подкреплением, которые не накладывают ограничения на то, какая стратегия используется. Также в некоторых алгоритмах могут использоваться специальные стратегии исследования среды [18].

1.4. Настройка эволюционных алгоритмов с помощью обучения с подкреплением

Существуют методы, в которых обучение с подкреплением используется для настройки эволюционных алгоритмов [16, 19, 20]. Эти методы позволяют с помощью обучения с подкреплением настраивать некоторые числовые параметры эволюционных алгоритмов, такие как вероятность применения генетических операторов.

Насколько известно автору, ни для одного из соответствующих алгоритмов не существует теоретических оценок времени их работы. Существуют некоторые работы, в которых оценивается скорость сходимости алгоритмов обучения с подкреплением (в основном, скорость сходимости алгоритма Q-learning при решении ряда модельных задач) [18, 21—24]. Также существует достаточно большое число работ, в которых оценивается время работы эволюционных алгоритмов при решении модельных задач, многие из которых перечислены, например, в обзоре [25]. Однако время работы эволюционного алгоритма, руководимого обучением с подкреплением, остается открытым вопросом.

1.5. Выводы по главе 1

Описаны основные методы, рассматриваемые в данной работе: эволюционные алгоритмы, эффективность которых должна быть повышена с помощью предлагаемого в работе метода, и обучение с подкреплением, которое используется в этом методе. Приведен обзор существующих методов, позволяющих настраивать эволюционные алгоритмы с помощью обучения с подкреплением. Отмечено, что в соответствующей области, насколько

известно автору, пока не получены теоретические оценки времени работы алгоритмов. Также приведен обзор существующих подходов, позволяющих повысить эффективность эволюционных алгоритмов с использованием дополнительных критериев. Упомянутые недостатки существующих подходов подтверждают актуальность разработки метода автоматического выбора дополнительных критериев.

Глава 2. Задача выбора вспомогательных оптимизируемых величин

2.1. Постановка задачи

Имеется целевой критерий оптимизации $t : W \to \mathbb{R}$, где W — дискретное пространство поиска решений. Необходимо найти точку глобального оптимума целевого критерия из пространства W. В случае, когда существует несколько точек глобального оптимума, достаточно найти хотя бы одну из них. Далее, говоря о точках оптимума и оптимальных значениях некоторых критериев, будем иметь в виду точки глобальных оптимумов и глобальные оптимальные значения, если не указано иное. Оптимизация целевого критерия производится с помощью эволюционного алгоритма.

Также имеется конечный набор H вспомогательных оптимизируемых величин, или *дополнительных (вспомогательных) критериев*, $h_i \in H$. Предполагается, что оптимизация некоторых из них может привести к нахождению точки оптимума целевого критерия за меньшее число итераций эволюционного алгоритма.

2.2. Требования, предъявляемые к методу выбора

Требуется разработать метод выбора дополнительных критериев, обладающий следующими характеристиками:

- критерии должны выбираться динамически, во время работы эволюционного алгоритма;
- разработанный метод должен быть основан на однокритериальном эволюционном алгоритме, использование многокритериальных алгоритмов нежелательно.

Использование многокритериальных эволюционных алгоритмов нежелательно по причине того, что они требуют больше вычислительных ресурсов для решения рассматриваемой задачи, чем однокритериальные. Однако предполагается, что разрабатываемый метод можно будет применять также и в случае использования многокритериальных алгоритмов. Требования к производительности метода в последнем случае аналогичны требованиям к производительности метода в случае использования однокритериальных эволюционных алгоритмов, приведенным ниже.

Сформулируем требования, предъявляемые к производительности метода. Назовем *временем работы алгоритма* число вычислений функции приспособленности, необходимое для нахождения оптимального значения целевого критерия. Далее будем называть эволюционным алгоритмом однокритериальный эволюционный алгоритм, оптимизирующий целевой критерий и не использующий дополнительные критерии. Разрабатываемый метод должен удовлетворять следующим требованиям:

- при наличии хотя бы одного дополнительного критерия, оптимизация которого позволяет получить оптимальное значение целевого критерия за меньшее число итераций эволюционного алгоритма, время работы предлагаемого метода должно быть меньше времени работы эволюционного алгоритма;
- при отсутствии описанного в предыдущем пункте дополнительного критерия, разрабатываемый метод должен работать не медленнее, чем эволюционный алгоритм.

2.3. Выводы по главе 2

Поставлена задача выбора дополнительных критериев оптимизации. Сформулированы ожидаемые характеристики метода выбора и требования, предъявляемые к нему.

Глава 3. Метод выбора вспомогательных оптимизируемых величин EA+RL

Опишем предлагаемый метод выбора дополнительных критериев эволюционного алгоритма (evolutionary algorithm, EA). Метод основан на обучении с подкреплением (reinforcement learning, RL). Будем называть предлагаемый метод EA+RL [26—28].

3.1. Описание метода

Пусть решается задача максимизации целевого критерия. Представим задачу выбора дополнительных критериев как задачу обучения с подкреплением [14]. В качестве среды будем рассматривать эволюционный алгоритм. Применение действия соответствует выбору критерия из множества, состоящего из целевого критерия и дополнительных критериев. Награда пропорциональна росту целевого критерия по сравнению с предыдущим поколением эволюционного алгоритма. Более подробно различные определения награды обсуждаются в разделе 3.2. Состояние в общем случае также зависит от значения целевого критерия (см. раздел 3.3).

Опишем общую схему метода EA+RL. Агент обучения с подкреплением выбирает некоторый критерий, который передается эволюционному алгоритму. Выбранный критерий используется в эволюционном алгоритме для формирования следующего поколения. После формирования следующего поколения, вычисляются значения награды и состояния, которые передаются агенту. Агент обновляет оценку выгодности действий, и процесс повторяется до тех пор, пока не выполнится критерий останова. При работе с эволюционными алгоритмами критерием останова чаще всего является достижение оптимума целевого критерия или превышение ограничения на число поколений. Схема метода EA+RL представлена на рис. 3.1.

Заметим, что в соответствии с алгоритмом обучения с подкреплени-

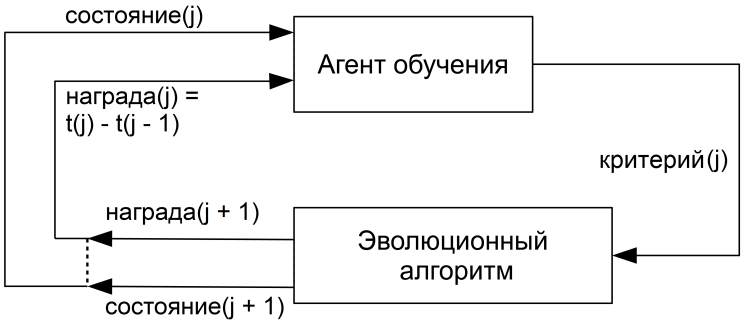

Рис. 3.1: Схема метода EA+RL, t — целевой критерий, j — номер итерации

ем (см. раздел 1.3), агент выбирает критерии так, чтобы максимизировать суммарную ожидаемую награду, зависящую от роста целевого критерия. Таким образом, действия агента должны приводить к максимизации целевого критерия. Это позволяет не оптимизировать целевой критерий явно на каждой итерации, а значит, использовать однокритериальный эволюционный алгоритм. Напомним, что некоторые подходы, описанные в разделе 1.2, вынуждены использовать многокритериальные алгоритмы, где одним из критериев оптимизации является целевой.

3.2. Определение награды

Для формализации понятия награды, введем некоторые обозначения. Пусть A — множество действий агента, $z \in Z$ — состояние среды. Тогда функция награды имеет вид $R : Z \times A \to \mathbb{R}$. Значения функции награды оцениваются с помощью информации о значениях мгновенной награды (см. Раздел 1.3).

Множество действий агента соответствует множеству всех критериев оптимизации, включая дополнительные критерии из множества H и целевой критерий t. Таким образом, $A = H \cup \{t\}$.

Обозначим i-е поколение как G_i. Применение действия агентом означает выбор некоторого критерия $f_i \in A$ в качестве функции приспособленности для поколения G_i. Нашей задачей является определение функции награды R, значение которой вычисляется после выбора критерия f_i в состоянии z_{i-1} и формирования поколения G_i.

Награда должна быть тем выше, чем выше значение целевого критерия, полученного в результате применения соответствующего критерия оптимизации. В методе EA+RL могут использоваться различные функции награды, удовлетворяющие этому требованию.

Награду можно вычислять, основываясь на приспособленности лучшей особи или на средней приспособленности особей, составляющих поколение. Пусть $B(f,i)$ — наибольшее значение критерия f в поколении G_i. Тогда формула мгновенной награды в первом случае имеет следующий вид:

$$r(t,i) = \frac{B(t,i) - B(t,i-1)}{B(t,i)} + k \sum_{f \in H} \frac{B(f,i) - B(f,i-1)}{B(f,i)},$$

где $0 \leq k \leq 1$ — параметр.

Пусть x — особь эволюционного алгоритма. Приведем формулу мгновенной награды, основанной на средней приспособленности поколения:

$$r(t,i) = \frac{\sum_{x \in G_i} t(x) - \sum_{x \in G_{i-1}} t(x)}{\sum_{x \in G_i} t(x)} + \\ + k \sum_{f \in H} \frac{\sum_{x \in G_i} f(x) - \sum_{x \in G_{i-1}} f(x)}{\sum_{x \in G_i} f(x)},$$

где $0 \leq k \leq 1$ — параметр.

Параметр k в обоих случаях влияет на то, будут ли учитываться значения дополнительных критериев в награде. Ненулевое значение параметра k имеет смысл использовать в случаях, когда известно, что дополнительные критерии положительно коррелируют с целевым критерием. Однако чаще о свойствах критериев заранее не известно, в таких случаях рекомендуется использовать $k = 0$.

В ходе экспериментальных исследований было получено, что для разных задач оптимизации эффективны разные способы определения награды [26, 29—31].

3.3. Определение состояния

Одним из простейших способов задания состояния является использование единственного состояния. Этот подход наиболее эффективен в случаях, когда во время всего процесса оптимизации наиболее выгодным является один и тот же критерий.

Также состояние можно определить как значение какого-либо из критериев, посчитанное на лучшей особи или усредненное для всех особей поколения. Подобный подход удобен для теоретического анализа и используется в следующем разделе. В случае, когда поколение состоит из одной особи, определение состояния таким образом позволяет получить Марковский процесс принятия решений [17].

В ходе ряда экспериментальных исследований показало свою эффективность определение состояния в виде вектора критериев, отсортированных по величине изменения их значений в двух последовательных поколениях [29].

Как и для способов определения награды, для разных задач оказались эффективными разные способы определения состояния.

3.4. Выводы по главе 3

Предложен метод выбора дополнительных критериев, основанный на обучении с подкреплением. Описана общая схема метода, способы определения награды и состояния. Предложенный подход является достаточно общим и не накладывает ограничения на используемый алгоритм обучения с подкреплением, как и на настраиваемый эволюционный алгоритм. Метод соответствует характеристикам, сформулированным в главе 2.

Глава 4. Теоретический анализ предлагаемого метода

В данной главе доказываются некоторые факты о предлагаемом методе на примере двух модельных задач, в которых требуется максимизировать некоторый целевой критерий. Пространство поиска в обеих задачах составляют битовые строки длины n. Поиск решения задачи оптимизации начинается с особи, состоящей из всех нулей.

В первой рассматриваемой задаче помимо целевого критерия оптимизации присутствует также дополнительный критерий, использование которого может затруднить оптимизацию целевого. Будем назвать такой критерий *мешающим*. Показывается, что предлагаемый метод EA+RL позволяет игнорировать мешающий критерий и проводить оптимизацию асимптотически за такое же время, как и без него [31].

Во второй рассматриваемой в этом разделе задаче наряду с целевым критерием оптимизации присутствует дополнительный критерий, использование которого может позволить быстрее найти точку оптимума целевого критерия. Будем называть такой критерий *помогающим*. Показывается, что предлагаемый метод позволяет использовать преимущества помогающего критерия и решает задачу оптимизации асимптотически быстрее, чем традиционный алгоритм, не использующий дополнительных критериев [32].

В качестве эталонного алгоритма оптимизации, с которым сравнивается предлагаемый метод, будем рассматривать метод спуска со случайными мутациями. Предлагаемый метод также будем применять к выбору функции приспособленности в методе спуска со случайными мутациями. Таким образом, анализируемый алгоритм будет отличаться от эталонного алгоритма оптимизации только в части выбора дополнительных критериев.

В качестве алгоритма обучения с подкреплением будем использо-

вать $Q-learning$ с жадной стратегией исследования среды [14]. В качестве состояния рассматривается число единиц в текущей битовой строке. Псевдокод анализируемого алгоритма, составленного в соответствии с предлагаемым методом EA+RL, приведен ниже.

Листинг 1 Алгоритм, реализующий метод EA+RL с использованием жадного Q-обучения и метода спуска со случайными мутациями

Вход: t — целевой критерий оптимизации; F — множество всех критериев оптимизации (содержит t); S — множество состояний; α — скорость обучения; γ — дисконтный фактор.

1: Инициализировать $Q(s, f)$ нулями для всех $s \in S$, $f \in F$
2: Инициализировать текущую особь: $x = 00...0$
3: **while** $(t(x) \neq \max)$ **do**
4: Запомнить текущую особь: prev = x
5: Состояние среды $s =$ число единиц в x
6: Выбрать критерий: $f = \arg\max_f Q(s, f)$
7: Инвертировать случайный бит x: y = flipOneBit(x)
8: Выбрать текущую особь: если $f(y) \geq f(x)$, то $x = y$
9: Вычислить награду: $r = t(x) - t(prev)$
10: Вычислить новое состояние: $s' =$ число единиц в x
11: Обновить Q: $Q(s, f) = Q(s, f) + \alpha(r + \gamma \max_{f'} Q(s', f') - Q(s, f))$
12: **end while**

4.1. Задача с мешающим критерием

Рассмотрим модификацию задачи OneMax [25]. Пространство поиска — битовые строки длины n. Пусть x — число единиц в битовой строке. Как и в задаче OneMax, требуется максимизировать целевой критерий: $f_1 = x$. Однако в рассматриваемой задаче помимо целевого критерия присутствует дополнительный. В качестве дополнительного критерия будем использовать критерий $f_0 = n - x$, значение которого соответствует числу нулей в битовой строке. Заметим, что при увеличении критерия f_0 на некоторую величину целевой критерий f_1 уменьшается на такую же величину. Таким образом, критерий f_0 является мешающим.

4.1.1. Схема доказательства

В разделе 4.1.2 формулируется и доказывается лемма об обучении. На основе леммы в разделе 4.1.3 строится цепь Маркова, моделирующая процесс оптимизации, управляемый с помощью EA+RL. Получившаяся

цепь содержит альтернативные пути, что усложняет ее анализ. В разделе 4.1.4 вычисляется математическое ожидание числа итераций EA+RL, необходимого для прохождения по альтернативному пути. Полученное выражение позволяет построить упрощенную цепь, в которой участки альтернативных путей заменены на взвешенные ребра.

На основе упрощенной цепи в разделе 4.1.5 вычисляется математическое ожидание числа итераций EA+RL, необходимое для максимизации целевого критерия. Полученное выражение сравнивается с аналогичным выражением для метода спуска со случайными мутациями. На основе сравнения делается вывод о том, что асимптотика времени работы метода EA+RL при решении рассматриваемой задачи с мешающим критерием совпадает с асимптотикой времени работы метода спуска со случайными мутациями при решении задачи OneMax без мешающего критерия. Таким образом, метод EA+RL позволяет успешно игнорировать мешающий критерий в рассмотренной задаче.

4.1.2. Лемма об обучении

Лемма 4.1. *Пусть агент обучения посещает состояние s и покидает его. Тогда при всех последующих посещениях s агент будет выбирать эффективный критерий f_1.*

Рассмотрим случай, когда состояние s посещается в первый раз. В этом случае $Q(s, f_0) = Q(s, f_1) = 0$, так как изначально значения Q проинициализированы нулями и агент еще не посещал состояние s, а значит, не мог изменить начальные значения. Таким образом, равновероятно может быть выбран как критерий f_0, так и критерий f_1. В каждом из этих случаев, оператор мутации может инвертировать либо единичный бит, либо нулевой. Следовательно, из состояния s возможны четыре вида переходов, как показано на рис. 4.1.

Рассмотрим четыре описанных случая:

- Выбирается критерий f_0, оператор мутации инвертирует ноль в еди-

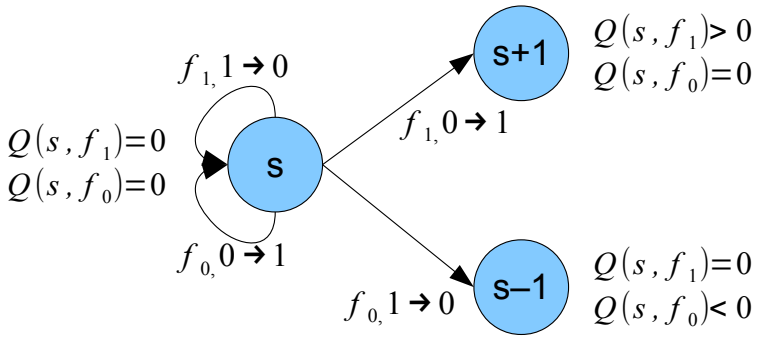

Рис. 4.1: Переходы из состояния, посещенного впервые

ницу. Поскольку полученная особь обладает меньшим числом нулей, она уступает родителю с точки зрения выбранного критерия. Полученная особь отвергается, алгоритм остается в состоянии s. Награда нулевая, значения Q не меняются.

- Выбирается критерий f_0, оператор мутации инвертирует единицу в ноль. Поскольку полученная особь обладает более высоким значением f_0, чем родительская особь, полученная особь становится текущей и алгоритм переходит в состояние $s-1$. Награда отрицательна, так как значение целевого критерия f_1, соответственно, уменьшается на единицу. Таким образом, значение $Q(s, f_0)$ становится отрицательным после обновления, значение $Q(s, f_1)$ остается равным нулю.
- Выбирается критерий f_1, оператор мутации инвертирует ноль в единицу. Поскольку полученной особи соответствует более высокое значение f_1, полученная особь становится текущей и алгоритм переходит в состояние $s+1$. Награда положительна, значение $Q(s, f_1)$ становится положительным, значение $Q(s, f_0)$ остается равным нулю.
- Выбирается критерий f_1, оператор мутации инвертирует единицу в ноль. Поскольку полученной особи соответствует меньшее значение f_1, родительская особь остается текущей, алгоритм остается в состоянии s. Награда нулевая, значения Q не меняются.

В двух из рассмотренных выше случаях происходит выход из состояния s, что соответствует формулировке леммы. В обоих случаях имеем

$Q(s, f_1) > Q(s, f_0)$, так как одно из значений Q нулевое, а другое положительное или отрицательное. При следующем посещении состояния s агент выберет эффективный критерий f_1, потому что ему соответствует большее значение Q. Неравенство между $Q(s, f_1)$ и $Q(s, f_0)$ не изменится, так как агент при применении f_1 будет получать либо нулевую, либо положительную награду, которая может только увеличить разрыв между $Q(s, f_1)$ и $Q(s, f_0)$. Таким образом, эффективный критерий f_1 будет выбираться при всех будущих посещениях состояния s.

4.1.3. Цепь Маркова

Рассмотрим цепь Маркова, соответствующую процессу оптимизации, выполняемому с помощью описанного алгоритма (см. рис. 4.2). Числа, написанные в вершинах цепи, соответствуют числу единиц в текущей особи. Первая особь полностью состоит из нулей, так что стартовая вершина содержит значение ноль.

Цепь состоит из трех групп вершин. Первая группа состоит из вершин, которые посещаются впервые, переход в них осуществляется из вершин, расположенных ниже (соответствующих меньшему числу единиц). Вторая группа состоит из вершин, посещаемых в результате того, что агент выбрал мешающий критерий f_0. Переход в такие вершины соответствует переходу вниз. Третья группа состоит из вершин, посещаемых при "восстановлении"после выбора мешающего критерия. Переход в такие вершины соответствует переходу вверх из вершин второй группы.

Эта цепь построена с использованием Леммы 4.1. Вершины из второй и третьей групп уже были посещены. Поэтому в таких вершинах агент всегда будет выбирать эффективный критерий f_1 и перемещаться в вершины, соответствующие большему числу единиц.

Следует отличать вершины цепи Маркова от состояний, введенных ранее. Состояние однозначно определяется числом единиц в текущей особи. Вершина цепи определяется не только числом единиц в текущей особи, но

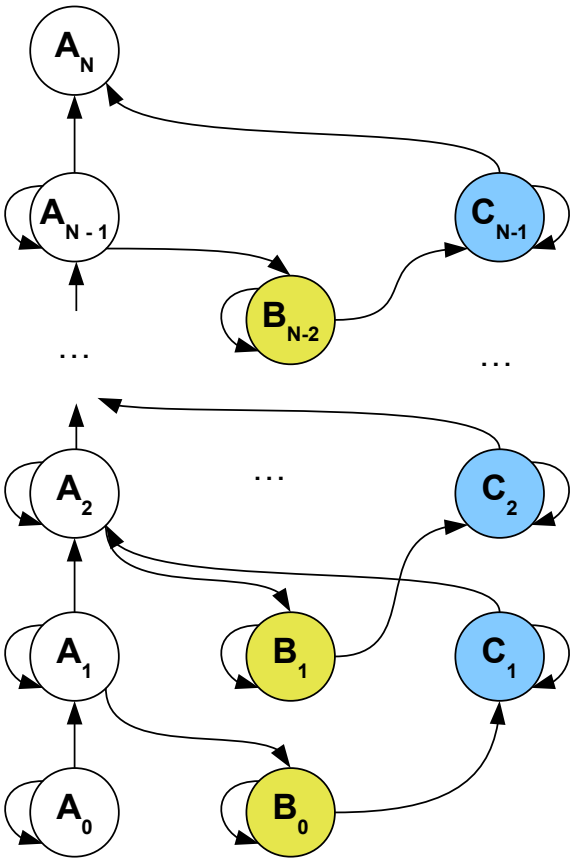

Рис. 4.2: Марковская цепь, соответствующая процессу оптимизации критерия f_1 с мешающим критерием f_0 с помощью метода EA+RL

и принадлежностью к одной из трех описанных выше групп.

4.1.4. Упрощение цепи Маркова

В этом разделе производится приведение цепи Маркова к удобному для дальнейшего анализа виду. Нашей целью является замена путей, проходящих через вершины второй и третьей группы, на взвешенные ребра, соединяющие вершины первой группы. Вес ребер будет соответствовать математическому ожиданию числа вычислений функции приспособленности, необходимого для прохода по заменяемому пути.

Заметим, что все заменяемые пути имеют одинаковую структуру. Рассмотрим один такой путь, представленный на рис. 4.3. Рядом с каждым переходом (ребром) на рисунке указан выбранный критерий оптимизации и результат применения оператора мутации.

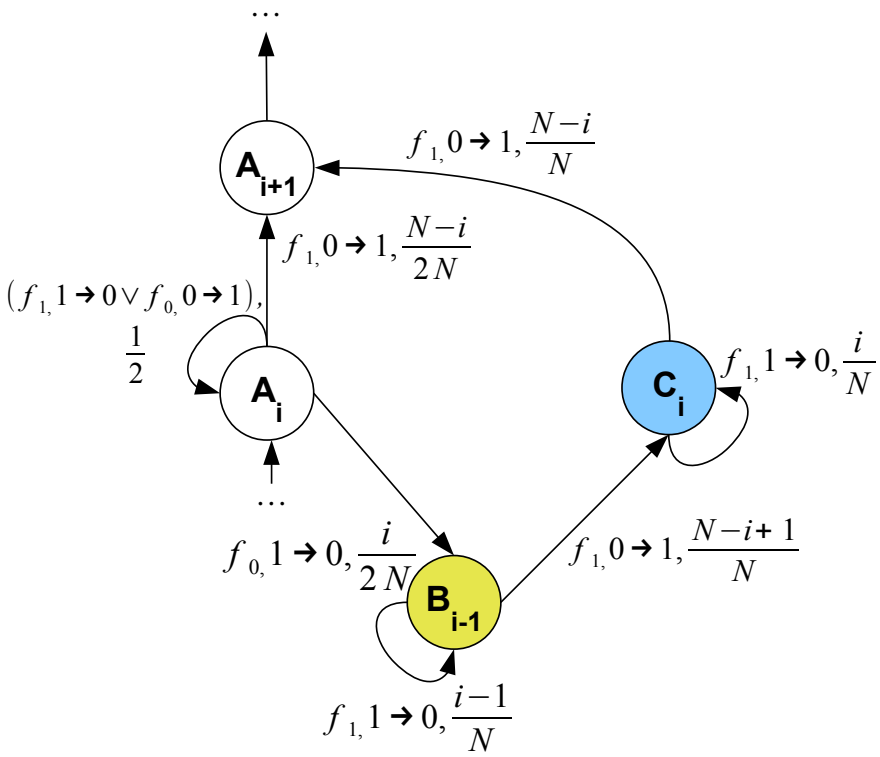

Рис. 4.3: Вероятности переходов в альтернативном пути из A_i в A_{i+1}, проходящем через B_{i-1} и C_i

Вычислим математическое ожидание числа вычислений функции приспособленности, необходимого для перехода из вершины B_{i-1} в вершину A_{i+1} через промежуточную вершину C_i. Для этого сначала вычислим математическое ожидание числа вычислений функции приспособленности для перехода из B_{i-1} в C_i:

$$E(B_{i-1} \to C_i) = 1 \times \frac{n-i+1}{n} + (1 + E(B_{i-1} \to C_i)) \times \frac{i-1}{n};$$

$$E(B_{i-1} \to C_i) = \frac{n}{n-i+1}.$$

Затем вычислим математическое ожидание числа вычислений функции приспособленности для перехода из C_i в A_{i+1}:

$$E(C_i \to A_{i+1}) = 1 \times \frac{n-i}{n} + (1 + E(C_i \to A_{i+1})) \times \frac{i}{n};$$

$$E(C_i \to A_{i+1}) = \frac{n}{n-i}.$$

Наконец, получим искомую величину, просуммировав промежуточные результаты:

$$E(B_{i-1} \to A_{i+1}) = E(B_{i-1} \to C_i) + E(C_i \to A_{i+1});$$

$$E(B_{i-1} \to A_{i+1}) = \frac{n}{n-i+1} + \frac{n}{n-i}.$$

Теперь мы можем заменить рассмотренный альтернативный путь из вершины A_i в вершину A_{i+1} на ребро соответствующего веса. Вероятность прохода по этому ребру равна вероятности выбора ребра $A_i \to B_{i-1}$ из замененного альтернативного пути, которая составляет $p = \frac{i}{2n}$.

В предположении, что ребро $A_i \to B_{i-1}$ выбрано с вероятностью p, его вес равен единице. Соответственно, вес ребра между A_i и A_{i+1} составляет $1 + E(B_{i-1} \to A_{i+1})$, как показано на рис. 4.4. Заменяя все альтернативные пути аналогичным образом, получаем упрощенную марковскую цепь.

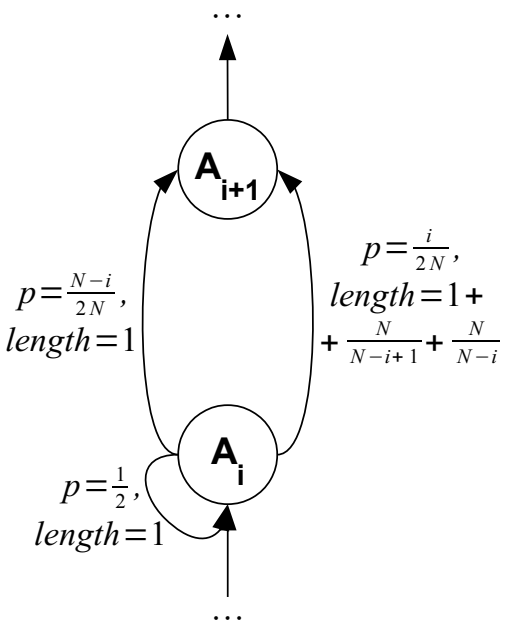

Рис. 4.4: Часть упрощенной марковской цепи

4.1.5. Асимптотическая оценка числа итераций EA+RL

В данном разделе вычисляется математическое ожидание числа вычислений функции приспособленности, необходимого для прохода по всем вершинам марковской цепи, начиная с вершины A_0 до вершины A_n. Другими словами, вычисляется математическое ожидание времени выполнения алгоритма EA+RL.

Пусть $Z(i)$ — математическое ожидание числа вычислений функции приспособленности, необходимого для перехода из вершины с номером i в

вершину с номером $i+1$. Используя вероятности переходов и веса ребер, указанные на рис. 4.4, получаем следующее уравнение:

$$Z(i) = \frac{1+Z(i)}{2} + \frac{n-i}{2n} + \left(1 + \frac{n}{n-i+1} + \frac{n}{n-i}\right) \times \frac{i}{2n}.$$

Решая приведенное выше уравнение, получаем выражение для $Z(i)$ в закрытой форме:

$$Z(i) = 2 + \frac{i}{n-i+1} + \frac{i}{n-i}. \tag{4.1}$$

Рассмотрим частный случай начальной вершины, изображенный на рис. 4.5.

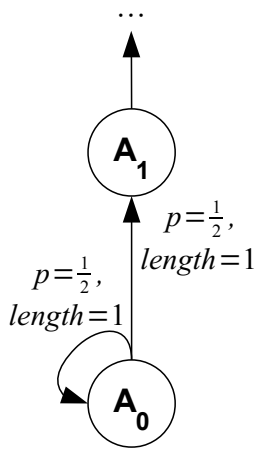

Рис. 4.5: Первые две вершины упрощенной марковской цепи

Аналогично, решим уравнение для $Z(0)$, чтобы получить выражение для $Z(0)$ в закрытой форме:

$$Z(0) = 1 \times \frac{1}{2} + (1 + Z(0)) \times \frac{1}{2};$$

$$Z(0) = 2.$$

Заметим, что подставляя $i = 0$ в выражение 4.1, также получим $Z(0) = 2$. Таким образом, выражение 4.1 верно и для вершины $Z(0)$.

Вычислим математическое ожидание общего числа вычислений функции приспособленности, необходимого для перехода из вершины с номером 0 в вершину с номером n как следующую сумму $Z(i)$:

$$T_{EA+RL}(n) = \sum_{i=0}^{n-1} \left(2 + \frac{i}{n-i+1} + \frac{i}{n-i} \right). \tag{4.2}$$

Приведем асимптотическую оценку полученного результата. Рассмотрим математическое ожидание числа вычислений функции приспособленности, необходимое для решения задачи OneMax в традиционной постановке (без дополнительных критериев) с помощью метода спуска со случайными мутациями:

$$T_EA(n) = \sum_{i=0}^{n-1} \left(1 + \frac{i}{n-i} \right); \tag{4.3}$$

$$T_EA(n) = \Theta(n \log n). \tag{4.4}$$

Проанализируем аргумент суммы из выражения 4.2, соответствующего времени работы алгоритма EA+RL:

$$T_{EA+RL}(n) = \sum_{i=0}^{n-1} \left(2 + \frac{i}{n-i+1} + \frac{i}{n-i} \right);$$

$$\begin{aligned} 1 + \frac{i}{n-i} \;&<\; 2 + \frac{i}{n-i+1} + \frac{i}{n-i} < \\ &<\; 2 + 2\frac{i}{n-i} = 2(1 + \frac{i}{n-i}). \end{aligned}$$

Сравнивая неравенства, приведенные выше, с временем решения задачи OneMax в традиционной постановке (см. выражения 4.3, 4.4), получаем верхнюю и нижнюю границы на время работы алгоритма EA+RL:

$$T_EA(n) < T_{EA+RL}(n) < 2 \cdot T_EA(n);$$

$$T_{EA+RL}(n) = \Theta(n \log n).$$

Таким образом, удалось показать, что время, необходимое для решения модифицированной задачи OneMax с мешающим критерием с помощью алгоритма EA+RL, составляет $\Theta(n \log n)$, что совпадает со временем решения традиционной задачи OneMax без дополнительных критериев. Полученный результат подтверждает, что метод EA+RL позволяет успешно игнорировать мешающий критерий.

4.2. Задача с помогающим критерием

Опишем модельную задачу XdivK с одним дополнительным критерием. Пространство поиска — битовые строки длины n. Пусть x — число единиц в битовой строке. Требуется максимизировать целевой критерий: $t = \lfloor \frac{x}{k} \rfloor$, где k — целочисленный параметр, $n \bmod k = 0$. В качестве дополнительного критерия будем использовать значение x.

Заметим, что оптимизация по дополнительному критерию позволяет достичь оптимума целевого критерия за меньшее число итераций. Пусть есть две битовые строки a и b с числом единиц y и z соответственно, $y < z$ (см. рис. 4.6). Пусть также с точки зрения целевого критерия эти строки не различаются. Однако использование строки b с большим числом единиц позволяет улучшить значение целевого критерия с большей вероятностью, чем использование строки a. Дополнительный критерий x ускоряет процесс оптимизации целевого критерия за счет того, что он позволяет выбрать строку b. Таким образом, критерий x является помогающим.

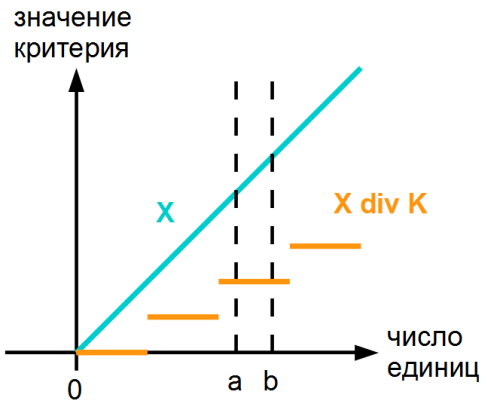

Рис. 4.6: Целевой и дополнительный критерии в задаче XdivK

Далее на примере описанной модельной задачи будет сравнено время работы метода EA+RL и эволюционного алгоритма, не использующего помогающий критерий. Требуется показать, что метод EA+RL позволит использовать преимущества помогающего критерия и найти оптимальное значение целевого критерия за меньшее число итераций.

В качестве эволюционного алгоритма, как и в предыдущем разделе, будем рассматривать метод спуска со случайными мутациями [8].

Напомним, что в качестве обучения с подкреплением используется алгоритм $\varepsilon-$жадного Q-обучения [14], поддерживающий текущую оценку ожидаемой награды, которая обновляется в соответствии с формулой: $Q(s,f) = Q(s,f) + \alpha(r + \gamma \max_{f'} Q(s',f') - Q(f,a))$, где f – выбранный критерий, состояние s — значение целевого критерия, награда r – разность значений целевого критерия на двух последовательных итерациях. Изначально Q инициализируется нулями.

4.2.1. Схема доказательства

Представим процесс оптимизации как цепь Маркова (см. рис. 4.7). Общий вид цепи одинаков как для метода EA+RL, так и для метода спуска со случайными мутациями. Вершины цепи соответствуют числу единиц в строке. Цветом выделены вершины, в которых x нацело делится на k. Заметим, что из этих вершин отсутствуют переходы в вершины с меньшим значением x, так как эволюционный алгоритм выбирает строки с большим или равным значением текущего критерия.

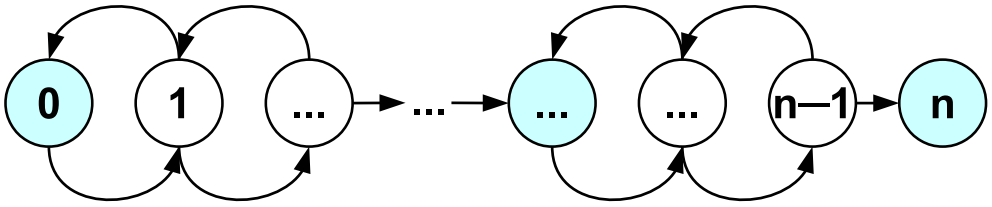

Рис. 4.7: Цепь Маркова, соответствующая процессу оптимизации в задаче XDIVK

Рассмотрим число итераций, необходимое для перехода из вершины x в вершину $x+1$. Пусть $Z_E(x)$ — математическое ожидание числа итераций метода спуска, $Z_R(x)$ — математическое ожидание числа итераций метода $EA+RL$. На рис. 4.8 и 4.9 для метода спуска и метода EA+RL рядом с каждым переходом цепи Маркова подписаны выбранный критерий, мутация и вероятность соответствующего перехода.

Оценим вероятности переходов в случае использования метода спуска (см. рис. 4.8). Для состояний, в которых x делится нацело на k

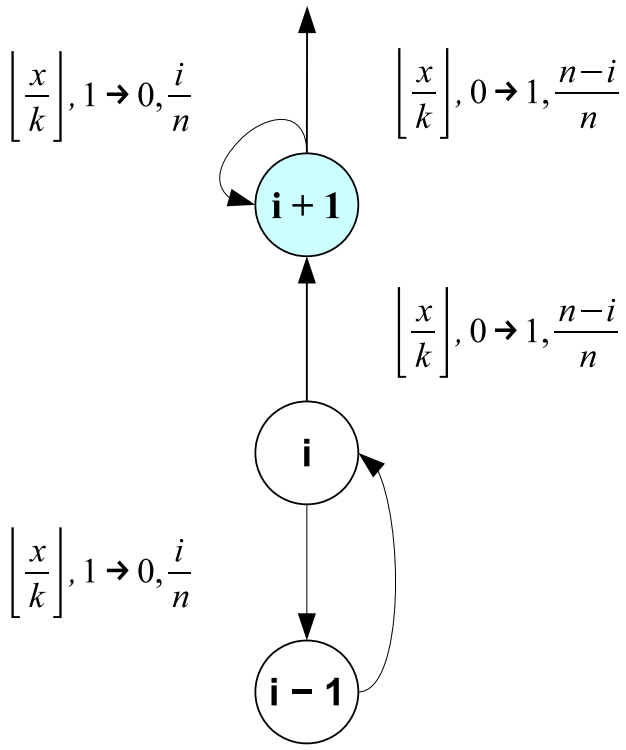

Рис. 4.8: Вероятности переходов при решении задачи XDIVK без дополнительных критериев

$(x \bmod k = 0)$, выполняется соотношение:

$$Z_E(x) = \frac{n-x}{n} + \frac{x}{n}(1 + Z_E(x)) = \frac{n}{n-x}, \tag{4.5}$$

в то время как для остальных состояний выполняется:

$$Z_E(x) = \frac{n-x}{n} + \frac{x}{n}(1 + Z_E(x-1) + Z_E(x)) = \frac{n}{n-x} + Z_E(x-1)\frac{x}{n-x}. \tag{4.6}$$

Рассмотрим теперь вероятности переходов в случае использования метода EA+RL, выбирающего на каждой итерации дополнительный или целевой критерий и передающий этот критерий в метод спуска со случайными мутациями (см. рис. 4.9). В этом случае вероятность перехода в состояние с меньшим числом единиц ниже, так как при выборе дополнительного критерия переход в строку с меньшим числом единиц осуществляться не будет.

Для состояний, в которых x делится нацело на k $(x \bmod k = 0)$, выполняется соотношение:

$$Z_R(x) = \frac{n-x}{n} + \frac{x}{n}(1 + Z_R(x)) = \frac{n}{n-x}, \tag{4.7}$$

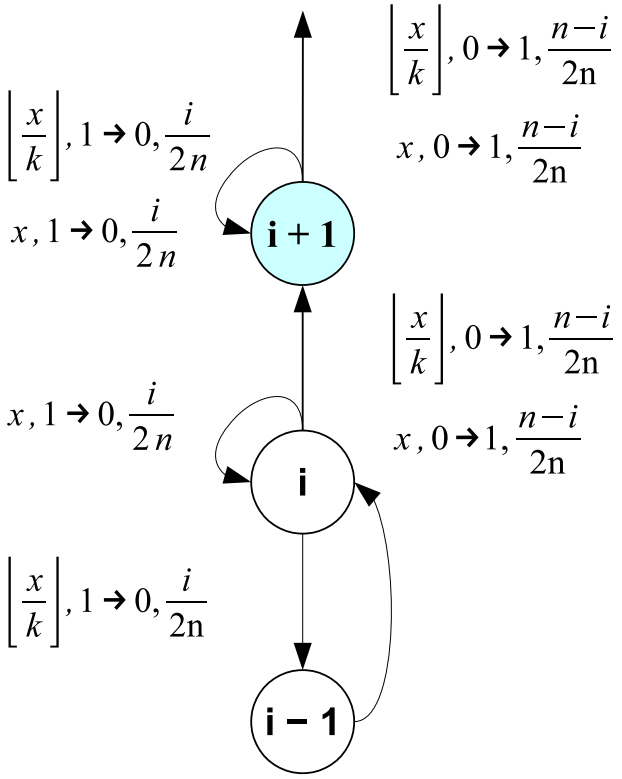

Рис. 4.9: Вероятности переходов при решении задачи XdivK с помогающим критерием

в то время как для остальных состояний выполняется:

$$Z_R(x) = \frac{n-x}{n} + \frac{x}{2n}(1 + Z_R(x)) + \frac{x}{2n}(1 + Z_R(x-1) + Z_R(x));$$

$$Z_R(x) = \frac{n}{n-x} + Z_R(x-1)\frac{x}{2(n-x)}. \tag{4.8}$$

Приведенные выражения представляются как:

$$Z_E(x) = \sum_{i=0}^{x \bmod k} \frac{\binom{n}{x-i}}{\binom{n-1}{x}}, Z_R(x) = \sum_{i=0}^{x \bmod k} 2^{-i}\frac{\binom{n}{x-i}}{\binom{n-1}{x}},$$

что можно доказать по индукции. Доказательство приводится в разделе 4.2.2

Для вычисления общего числа итераций, необходимого для достижения максимального значения целевого критерия, следует просуммировать полученные выражения вдоль рассмотренной цепи Маркова: $T_E(x) = \sum_{x=0}^{n-1} Z_E(x)$ и $T_R(x) = \sum_{x=0}^{n-1} Z_R(x)$. В разделе 4.2.3 эти суммы модифицируется для проведения дальнейшего анализа.

Можно показать, что $T_E(n,k) = T_R(n,k) = \Omega(n^k) = O(n^{k+1})$, где k — константа (см. раздел 4.2.4). В разделе 4.2.5 показывается, что если

$n \to \infty$ при фиксированном k, то $\frac{T_E(n,k)}{T_R(n,k)} \geq 2^{k-2}(1 - o(1))$. Последнее выражение означает, что метод EA+RL позволяет решить модельную задачу XDIVK не менее, чем в 2^{k-2} раза быстрее, чем метод спуска со случайными мутациями. Был проведен численный эксперимент, который показал, что данную оценку можно попытаться улучшить до 2^{k-1}.

4.2.2. Устранение рекурсии

Устраним рекурсию в выражениях для Z_E и Z_R.

Теорема 4.2. *Следующее равенство выполняется для всех x:*

$$Z_E(x) = \sum_{i=0}^{x \bmod k} \frac{\binom{n}{x-i}}{\binom{n-1}{x}}. \tag{4.9}$$

Доказательство. *Докажем по индукции. Для всех x, таких что $x \bmod k = 0$, база индукции следует из выражения (4.5):*

$$Z_E(x) = \frac{n}{n-x} = \frac{\binom{n}{x}}{\binom{n-1}{x}} = \sum_{i=0}^{x \bmod k} \frac{\binom{n}{x-i}}{\binom{n-1}{x}}.$$

Предположим, что $v > 0$ и теорема доказана для всех x', таких что $x' \bmod k = v - 1$. Тогда для всех x, удовлетворяющих равенству $x \bmod k = v$, выполняем шаг индукции, используя выражение 4.6:

$$\begin{aligned}
Z_E(x) &= \frac{n}{n-x} + Z_E(x-1)\frac{x}{n-x} = \\
&= \frac{n}{n-x} + \frac{x}{n-x}\left(\sum_{i=0}^{(x-1) \bmod k} \frac{\binom{n}{x-1-i}}{\binom{n-1}{x-1}}\right) = \\
&= \frac{n}{n-x} + \frac{x}{n-x}\left(\sum_{i=1}^{x \bmod k} \frac{\binom{n}{x-i}}{\binom{n-1}{x-1}}\right) = \\
&= \frac{n}{n-x} + \sum_{i=1}^{x \bmod k} \frac{\binom{n}{x-i}}{\binom{n-1}{x}} = \\
&= \frac{\binom{n}{x}}{\binom{n-1}{x}} + \sum_{i=1}^{x \bmod k} \frac{\binom{n}{x-i}}{\binom{n-1}{x}} = \sum_{i=0}^{x \bmod k} \frac{\binom{n}{x-i}}{\binom{n-1}{x}}. \quad \square
\end{aligned}$$

Теорема 4.3. *Следующее равенство выполняется для всех x:*

$$Z_R(x) = \sum_{i=0}^{x \bmod k} 2^{-i} \frac{\binom{n}{x-i}}{\binom{n-1}{x}}. \tag{4.10}$$

Доказательство. *Теорема доказывается по индукции аналогично Теореме 4.2 с использованием выражений (4.7) и (4.8).*

4.2.3. Модификация выражений

Пусть используется метод спуска со случайными мутациями, и начальная особь состоит из всех нулей. Тогда суммарное число вычислений функции приспособленности, необходимое для нахождения оптимума целевой функции в задаче XDIVK, составляет:

$$T_E(n,k) = \sum_{x=0}^{n-1} Z_E(x) = \sum_{x=0}^{n-1} \sum_{i=0}^{x \bmod k} \frac{\binom{n}{x-i}}{\binom{n-1}{x}}. \tag{4.11}$$

Аналогично, пусть применяется метод EA+RL и начальная особь состоит из всех нулей. Тогда суммарное число вычислений функции приспособленности, необходимое для нахождения оптимума целевой функции в задаче XDIVK, составляет:

$$T_R(n,k) = \sum_{x=0}^{n-1} Z_R(x) = \sum_{x=0}^{n-1} \sum_{i=0}^{x \bmod k} 2^{-i} \frac{\binom{n}{x-i}}{\binom{n-1}{x}}. \tag{4.12}$$

Эти выражения достаточно сложно анализировать, особенно если требуется сравнить асимптотическое поведение T_E и T_R. Перепишем $T_E(n,k)$ в следующем виде:

$$T_E(n,k) = \sum_{x=0}^{n-1} \sum_{i=0}^{x \bmod k} \frac{\binom{n}{x-i}}{\binom{n-1}{x}} =$$
$$= \sum_{m=0}^{\frac{n}{k}-1} \sum_{j=0}^{k-1} \sum_{i=0}^{j} \frac{\binom{n}{mk+j-i}}{\binom{n-1}{mk+j}} =$$
$$= \sum_{m=0}^{\frac{n}{k}-1} \sum_{i=0}^{k-1} \sum_{j=i}^{k-1} \frac{\binom{n}{mk+j-i}}{\binom{n-1}{mk+j}} =$$
$$= \sum_{i=0}^{k-1} \sum_{j=i}^{k-1} \sum_{m=0}^{\frac{n}{k}-1} \frac{\binom{n}{mk+j-i}}{\binom{n-1}{mk+j}}. \tag{4.13}$$

Аналогично, переписав $T_R(n,k)$, получаем:

$$T_R(n,k) = \sum_{x=0}^{n-1} \sum_{i=0}^{x \bmod k} 2^{-i} \frac{\binom{n}{x-i}}{\binom{n-1}{x}} =$$
$$= \sum_{i=0}^{k-1} 2^{-i} \sum_{j=i}^{k-1} \sum_{m=0}^{\frac{n}{k}-1} \frac{\binom{n}{mk+j-i}}{\binom{n-1}{mk+j}}. \tag{4.14}$$

В результате выполнения приведенных выше преобразований удалось вынести 2^{-i}, насколько это было возможно, наружу, что позволяет упростить оценку асимптотического поведения T_E и T_R.

4.2.4. Асимптотическая оценка числа вычислений функции приспособленности

В этом разделе приводится оценка асимптотического поведения T_E и T_R.

Определим $V(n,k,i)$ следующим образом:

$$V(n,k,i) = \sum_{j=i}^{k-1} \sum_{m=0}^{\frac{n}{k}-1} \frac{\binom{n}{mk+j-i}}{\binom{n-1}{mk+j}}. \tag{4.15}$$

Используя выражения (4.13) и (4.15), получаем:

$$T_E(n,k) = \sum_{i=0}^{k-1} V(n,k,i). \tag{4.16}$$

Аналогично, используя выражения (4.14) и (4.15), получаем:

$$T_R(n,k) = \sum_{i=0}^{k-1} 2^{-i} V(n,k,i). \tag{4.17}$$

Лемма 4.4. *Для всех i, x, таких что $0 \le i < k$, $0 \le x < n-i-1$, выполняется следующее:*

$$\frac{\binom{n}{x}}{\binom{n-1}{x+i}} < \frac{\binom{n}{x+1}}{\binom{n-1}{x+i+1}}. \tag{4.18}$$

Доказательство.

$$\frac{\frac{\binom{n}{x}}{\binom{n-1}{x+i}}}{\frac{\binom{n}{x+1}}{\binom{n-1}{x+i+1}}} = \frac{(x+1)(n-x-i-1)}{(n-x)(x+i+1)} =$$

$$= \left(1 - \frac{i}{x+i+1}\right)\left(1 - \frac{i+1}{n-x}\right) < 1. \quad \square$$

Из выражения (4.18) следует, что

$$\frac{\binom{n}{x}}{\binom{n-1}{x+i}} \le \binom{n}{n-i-1}. \tag{4.19}$$

Теорема 4.5. *Если k является константой в зависимости от n, то*

$$V(n,k,i) = \Omega(n^{i+1}) = O(n^{i+2}). \tag{4.20}$$

Доказательство. *Рассмотрим в выражении (4.15) случай, когда $j = k-1$ и $m = n/k - 1$, тогда отношение биноминальных коэффициентов составляет $\binom{n}{n-i-1}$, так что:*

$$\binom{n}{n-i-1} \le V(n,k,i).$$

Из (4.19) следует, что

$$V(n,k,i) \le \binom{n}{n-i-1}(k-i)\frac{n}{k}.$$

Если k зависит от n как константа, тогда из $i < k$ следует, что $\binom{n}{n-i-1} = \Theta(n^{i+1})$. Из этого факта и предыдущего выражения следует доказываемая асимптотическая оценка.

Теорема 4.6. *Асимптотическая оценка сложности $T_E(n,k)$ и $T_R(n,k)$ для фиксированного k составляет $\Omega(n^k)$ и $O(n^{k+1})$ соответственно.*
Доказательство. *Следует из (4.16), (4.17) и (4.20).*

4.2.5. Отношение числа вычислений функции приспособленности

Получим отношение числа вычислений функций приспособленности, необходимого для достижения оптимума целевого критерия в задаче $XdivK$ при использовании метода спуска со случайными мутациями и метода EA+RL.

Теорема 4.7. *Для достаточно больших n и фиксированного k*

$$T_E(n,k) \geq 2^{k-2}(1-o(1))T_R(n,k).$$

Доказательство.

$$T_E(n,k) = V(n,k,k-1) + V(n,k,k-2) + o(n^k);$$
$$T_R(n,k) = \frac{V(n,k,k-1)}{2^{k-1}} + \frac{V(n,k,k-2)}{2^{k-2}} + o(n^k).$$

Рассмотрим три случая:

1. $V(n,k,k-1) = \Theta(n^k)$, $V(n,k,k-2) = \Theta(n^k)$. *В этом случае оценим отношение следующим образом:*

$$\frac{T_E(n,k)}{T_R(n,k)} = \frac{V(n,k,k-1) + V(n,k,k-2) + o(n^k)}{\frac{V(n,k,k-1)}{2^{k-1}} + \frac{V(n,k,k-2)}{2^{k-2}} + o(n^k)} \geq$$
$$\geq \frac{V(n,k,k-1) + V(n,k,k-2) + o(n^k)}{\frac{V(n,k,k-1)+V(n,k,k-2)}{2^{k-2}} + o(n^k)} =$$
$$= \frac{2^{k-2} + \frac{o(n^k)}{V(n,k,k-1)+V(n,k,k-2)}}{1 + \frac{o(n^k)}{V(n,k,k-1)+V(n,k,k-2)}} =$$
$$= \frac{2^{k-2} + o(1)}{1 + o(1)} \geq \frac{2^{k-2}}{1 + o(1)} = 2^{k-2}(1 - o(1)).$$

2. $V(n,k,k-1) = \Theta(n^k)$, $V(n,k,k-2) = o(n^k)$. *В этом случае оценим отношение как*

$$\frac{T_E(n,k)}{T_R(n,k)} = \frac{V(n,k,k-1) + o(n^k)}{\frac{V(n,k,k-1)}{2^{k-1}} + o(n^k)} \geq$$
$$\geq \frac{2^{k-1} + \frac{o(n^k)}{V(n,k,k-1)}}{1 + \frac{o(n^k)}{V(n,k,k-1)}} =$$
$$= \frac{2^{k-1} + o(1)}{1 + o(1)} \geq \frac{2^{k-1}}{1 + o(1)} = 2^{k-1}(1 - o(1)).$$

3. $V(n,k,k-1) = \omega(n^k)$. *В этом случае оценим отношение следующим образом:*

$$\frac{T_E(n,k)}{T_R(n,k)} = \frac{V(n,k,k-1) + O(n^k)}{\frac{V(n,k,k-1)}{2^{k-1}} + O(n^k)} \geq$$
$$\geq \frac{2^{k-1} + \frac{O(n^k)}{V(n,k,k-1)}}{1 + \frac{O(n^k)}{V(n,k,k-1)}} =$$
$$= \frac{2^{k-1} + o(1)}{1 + o(1)} \geq \frac{2^{k-1}}{1 + o(1)} = 2^{k-1}(1 - o(1)).$$

Таким образом, в первом случае получаем оценку отношения $2^{k-2}(1 - o(1))$, *в двух оставшихся случаях получаем* $2^{k-1}(1 - o(1))$.

4.3. Выводы по главе 4

Доказана эффективность предложенного метода на примере двух модельных задач. Рассмотрены задачи с критерием, мешающим процессу оптимизации целевого критерия, и с критерием, помогающим процессу оптимизации целевого критерия. Проведенный анализ основан на представлении процесса оптимизации в виде марковской цепи. Показано, что метод позволяет игнорировать мешающий критерий и выбирать помогающий, что приводит к асимптотическому улучшению времени оптимизации целевого критерия. Таким образом, поведение метода при решении рассмотренных задач соответствует требованиям, предъявляемым в главе 2.

Глава 5. Применение метода EA+RL к задаче о генерации тестов

В данной главе рассматривается практическое применение метода EA+RL. Метод применяется для повышения эффективности решения задачи о генерации тестов производительности против решений олимпиадных задач по программированию [30, 33—35].

5.1. Задача о генерации тестов

Требуется сгенерировать тест, на котором время работы тестируемого решения олимпиадной задачи по программированию превышает заданный порог, равный пяти секундам. Тесты можно представить в виде особей эволюционного алгоритма. Тогда целевым критерием будет время работы решения t, его необходимо максимизировать.

Несмотря на то, что время работы решения на тесте является целевым критерием оптимизации, его неэффективно использовать в качестве функции приспособленности, так как его значения зашумлены, квантованы и зависят от платформы, на которой запускается решение. Поэтому в работе [33] было предложено использовать дополнительные критерии оптимизации, представляющие собой счетчики итераций циклов и рекурсивных вызовов функций. Как было показано в той же работе, на разных этапах оптимизации могут быть выгодны разные дополнительные критерии. Поэтому возникает необходимость применения метода адаптивного выбора критериев. В качестве такого метода будем использовать предлагаемый в настоящей работе метод EA+RL.

В качестве олимпиадной задачи по программированию рассматривается задача *Ships. Version 2* [36]. Дополнительные критерии вставляются вручную в код решения задачи. Более подробная информация об использованных дополнительных критериях содержится в работе [30].

5.2. Описание эксперимента

В ходе эксперимента генерация тестов проводилась как с одной зафиксированной функцией приспособленности, так и с помощью методов, позволяющих выбирать текущую функцию приспособленности (критерий оптимизации). К последним относятся предлагаемый в настоящей работе метод EA+RL, а также метод, предложенный в [11], и случайный выбор.

Каждый алгоритм запускался 100 раз, затем производилось усреднение результатов. Выполнение алгоритма продолжалось до тех пор, пока не находился тест, на котором время работы решения превышало установленный порог, или же пока не достигался предел на число поколений, равный 10000. Поколение состояло из 200 особей. Для генерации каждого поколения требовалось одинаковое число вычислений функции приспособленности, поэтому в дальнейшем будем сравнивать алгоритмы, основываясь на числе поколений, понадобившемся для достижения оптимального значения целевой функции приспособленности.

Далее приводится информация о значениях параметров использованных алгоритмов. Значения параметров были подобраны в ходе предварительного эксперимента.

5.2.1. Эволюционный алгоритм

В данном алгоритме для формирования следующего поколения применяется турнирный отбор с размером турнира 2 и с вероятностью выбора лучшей особи, равной 0,9. Операторы мутации и скрещивания применяются с вероятностью 1,0. Используется элитизм, отбирается пять лучших особей [2]. Если на протяжении 10000 поколений лучшее полученное значение функции приспособленности не изменяется, то текущее поколение очищается и заполняется случайными новыми особями.

5.2.2. Метод, с которым производится сравнение

Производится сравнение с методом, описанным в работе [11]. Согласно этому методу, одновременно оптимизируются целевой критерий и дополнительный критерий, выбираемый динамически. Дополнительный критерий выбирается случайным образом и используется для генерации 50 поколений особей. Затем случайным образом выбирается другой критерий, отличный от уже использовавшихся, и процесс повторяется. После того, как все критерии были выбраны, процесс выбора начинается заново. Для одновременной оптимизации нескольких функций приспособленности используется ускоренная вариация алгоритма *NSGA-II* [37]. Применяется стратегия выбора особей, основанная на доминировании по Парето.

5.2.3. Параметры обучения с подкреплением

В качестве алгоритмов обучения с подкреплением использовались *Q-learning* [14] с $\varepsilon-$жадной стратегией исследования среды и с параметрами $\varepsilon = 0.3, \alpha = 0.4, \gamma = 0.001$, а также *Delayed Q-learning* с параметрами $m = 5, \varepsilon = 0.001, \gamma = 0.1$ [18]. В обоих алгоритмах использовалась награда, основанная на средней приспособленности поколения с параметром $k = 0.5$ (см. раздел 3.2) и единственное состояние.

5.2.4. Вычисление среднего числа поколений

Как было сказано ранее, число поколений было ограничено до 10000. По этой причине в некоторых запусках не было достигнуто искомое значение целевого критерия. Таким образом, среднее число поколений, понадобившееся для достижения оптимального значения целевой функции приспособленности, не может быть точно посчитано. Однако можно оценить это значение, если предположить, что алгоритм перезапускается, если после 10000 поколений оптимум не достигнут.

Обозначим как E_S среднее число поколений, за которое удалось найти оптимум до достижения ограничения на число поколений. Пусть R —

доля успешных запусков, G — максимальное число поколений до перезапуска (в рамках эксперимента $G = 10000$). Тогда математическое ожидание E числа поколений, необходимого для нахождения оптимума, может быть получено из следующего уравнения:

$$E = E_S \cdot R + (G + E) \cdot (1 - R),$$

решением которого является

$$E = E_S + \frac{1 - R}{R} G. \tag{5.1}$$

Для нахождения аналогичной формулы стандартного отклонения, посчитаем математическое ожидание Q квадрата числа поколений, необходимого для нахождения оптимума. Пусть Q_S — среднее число поколений в успешных запусках, завершившихся до достижения ограничения на число поколений G. Тогда

$$Q = Q_S \cdot R + (Q + 2GE + G^2) \cdot (1 - R),$$

решая уравнение, получаем

$$Q = Q_S + \frac{1 - R}{R}(G^2 + 2GE). \tag{5.2}$$

Пусть стандартное отклонение числа поколений в успешных запусках составляет D_S и стандартное отклонение числа поколений, необходимых для достижения оптимума составляет D. По определению стандартного отклонения, $D_S^2 = Q_S - E_S^2$ и $D^2 = Q - E^2$. В последнем равенстве подставим Q из (5.2):

$$D^2 = Q_S + \frac{1 - R}{R}(G^2 + 2GE) - E^2.$$

Заменяя Q_S на $D_S^2 + E_S^2$ и извлекая квадратный корень, получаем формулу для D:

$$D = \sqrt{E_S^2 - E^2 + D_S^2 + \frac{1 - R}{R}(G^2 + 2GE)}. \tag{5.3}$$

5.3. Результаты

Результаты эксперимента приведены в таблице 5.1. Запуск считается успешным, если удалось получить тест, на котором время работы решения

превышает установленный порог. Среднее число поколений и стандартное отклонение σ вычислены в соответствии с формулами (5.1) и (5.3) соответственно.

Таблица 5.1: Результаты генерации тестов

Алгоритм	Критерии	Успешные запуски, %	Среднее число поколений	σ	Медиана
Критерии фиксированы					
GA	tuple	89	4497	4716	3013,5
GA	iterations	74	8182	7459	6273,5
GA	length	51	15030	13969	9783,5
GA	time	1	990319	994987	> 10000
Критерии выбираются					
GA + Delayed Q-learning	все	90	4411	4352	2919,0
GA + Q-learning	все	85	4580	5186	2308,0
GA + Random	все	76	6074	6788	3277,0
NSGA-II+Jensen	все	75	6103	7076	3251,5

Можно видеть, что оптимизация с использованием целевого критерия t неэффективна, как и предполагалось. Среди дополнительных критериев наиболее эффективен критерий *tuple*. Однако в общем случае мы не можем использовать этот результат, так как чтобы узнать о том, какой критерий наиболее эффективен, необходимо перебрать все критерии, что требует дополнительного времени.

Использование методов, выбирающих критерии оптимизации, позволяет ограничиться одним запуском эволюционного алгоритма. Можно видеть, что среди подобных методов наибольшую эффективность показал предложенный в настоящей работе метод EA+RL, которому соответствуют алгоритмы *GA+Delayed Q-learning* и *GA+Q-learning*. Алгоритм *NSGA-II+Jensen*, с которым производилось сравнение предложенного метода, оказался наименее эффективным среди рассмотренных.

5.4. Выводы по главе 5

Представлены результаты практического применения предложенного метода. Метод позволил повысить эффективность генерации тестов против решений олимпиадных задач по программированию. Кроме того, проведено сравнение предложенного метода с другим существующим методом

выбора дополнительных критериев. В рамках рассмотренной задачи, предложенный метод оказался эффективнее. Результаты экспериментов подтверждают, что метод соответствует требованиям, сформулированным в главе 2.

Заключение

Предложен метод выбора вспомогательных оптимизируемых величин, основанный на применении обучения с подкреплением. Метод предназначен для повышения эффективности оптимизации целевого критерия с помощью эволюционных алгоритмов.

На примере двух модельных задач показано, что метод позволяет игнорировать вспомогательные величины, мешающие оптимизации целевого критерия, и выбирать вспомогательные величины, помогающие оптимизации целевого критерия. В частности, для первой задачи показано, что асимптотика времени работы предложенного метода такая же, как в случае отсутствия плохого критерия, и составляет $\Theta(n \log n)$, где n — размер решения. Для второй задачи получено отношения времени работы метода спуска со случайными мутациями ко времени работы предложенного метода, показывающее, что предложенный метод решает задачу не менее, чем в 2^{k-2} раз быстрее, где k — параметр задачи.

Также предложенный метод использован для повышения эффективности генерации тестов против решений олимпиадных задач по программированию. Показано, что он позволяет решать поставленную задачу эффективнее, чем другие рассмотренные методы.

Список литературы

1. *Mitchell M.* An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press, 1996.
2. *Eiben A. E., Smith J. E.* Introduction to Evolutionary Computing. Berlin, Heidelberg, New York: Springer-Verlag, 2007.
3. *Derrac J., Garcia S., Molina D., Herrera F.* A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms // Swarm and Evolutionary Computation. 2011. №1. С. 3—18.
4. *Евтушенко Ю. Г., Жадан В. Г.* Точные вспомогательные функции в задачах оптимизации // Журнал вычислительной математкики и математической физики. 1990. №1. С. 43—57.
5. *Segura C., Coello C. A. C., Miranda G., Léon C.* Using multi-objective evolutionary algorithms for single-objective optimization // 4OR. 2013. №11. С. 201—228.
6. *Neumann F., Wegener I.* Can Single-Objective Optimization Profit from Multiobjective Optimization? B: Multiobjective Problem Solving from Nature. Natural Computing Series. Springer Berlin Heidelberg, 2008. С. 115—130.
7. *Brockhoff D., Friedrich T., Hebbinghaus N., Klein C., Neumann F., Zitzler E.* On the Effects of Adding Objectives to Plateau Functions // Transactions on Evolutionary Computation. 2009. №3. С. 591—603.
8. *Knowles J. D., Watson R. A., Corne D.* Reducing Local Optima in Single-Objective Problems by Multi-objectivization / Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization. Springer-Verlag, 2001. С. 269—283.
9. *Lochtefeld D. F., Ciarallo F. W.* Multiobjectivization via helperobjectives with the tunable objectives problem // IEEE Trans. Evolutionary Computation. 2012. №16. С. 373—390.
10. *Handl J., Lovell S. C., Knowles J. D.* Multiobjectivization by Decomposition of Scalar Cost Functions. B: Parallel Problem Solving from Nature – PPSN X. Т. 5199. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008. С. 31—40.
11. *Jensen M. T.* Helper-Objectives: Using Multi-Objective Evolutionary Algorithms for Single-Objective Optimisation: Evolutionary Computation Combinatorial Optimization // Journal of Mathematical Modelling and Algorithms. 2004. №4. С. 323—347.
12. *Lochtefeld D. F., Ciarallo F. W.* Deterministic Helper-Objective Sequences Applied to Job-Shop Scheduling / Proceedings of Genetic and Evolutionary Computation Conference. ACM, 2010. С. 431—438.
13. *Lochtefeld D. F., Ciarallo F. W.* Helper-Objective Optimization Strategies for the Job-Shop Scheduling Problem // Applied Soft Computing. 2011. №6. С. 4161—4174.
14. *Sutton R. S., Barto A. G.* Reinforcement Learning: An Introduction. Cambridge, MA, USA: MIT Press, 1998.
15. *Kaelbling L. P., Littman M. L., Moore A. W.* Reinforcement Learning: A Survey // Journal of Artificial Intelligence Research. 1996. С. 237—285.
16. *Eiben A. E., Horvath M., Kowalczyk W., Schut M. C.* Reinforcement Learning for Online Control of Evolutionary Algorithms / Proceedings of the 4th international conference on Engineering self-organising systems. Springer-Verlag, Berlin, Heidelberg, 2006. С. 151—160.
17. *Николенко С. И., Тулупьев А. Л.* Самообучающиеся системы. М., 2009.
18. *Strehl A. L., Li L., Wiewiora E., Langford J., Littman M. L.* PAC Model-free Reinforcement Learning / Proceedings of the 23rd International Conference on Machine Learning. 2006. С. 881—888.
19. *Müller S., Schraudolph N. N., Koumoutsakos P. D.* Step Size Adaptation in Evolution Strategies using Reinforcement Learning / Proceedings of the Congress on Evolutionary Computation. IEEE, 2002. С. 151—156.

20. *Sakurai Y.*, *Takada K.*, *Kawabe T.*, *Tsuruta S.* A Method to Control Parameters of Evolutionary Algorithms by Using Reinforcement Learning / Proceedings of 2010 Sixth International Conference on Signal-Image Technology and Internet-Based Systems (SITIS). 2010. С. 74—79.
21. *Gosavi A.* Reinforcement Learning: A Tutorial Survey and Recent Advances // INFORMS Journal on Computing. 2009. №2. С. 178—192.
22. *Singh S.*, *Jaakkola T.*, *Littman M. L.*, *Szepesvári C.* Convergence Results for Single-Step On-Policy Reinforcement Learning Algorithms // Mach. Learn. 2000. №3. С. 287—308.
23. *Kearns M.*, *Singh S.* Finite-Sample Convergence Rates for Q-learning and Indirect Algorithms / Proceedings of the 1998 conference on Advances in neural information processing systems II. Cambridge, MA, USA: MIT Press, 1999. С. 996—1002.
24. *Even-Dar E.*, *Mansour Y.* Learning Rates for Q-learning // J. Mach. Learn. Res. 2004. С. 1—25.
25. *Oliveto P. S.*, *He J.*, *Yao X.* Time Complexity of Evolutionary Algorithms for Combinatorial Optimization: A Decade of Results // International Journal of Automation and Computing. 2007. №3. С. 281—293.
26. *Buzdalova A.*, *Buzdalov M.* Increasing Efficiency of Evolutionary Algorithms by Choosing between Auxiliary Fitness Functions with Reinforcement Learning / Proceedings of the International Conference on Machine Learning and Applications. Т. 1. 2012. С. 150—155.
27. *Buzdalova A.*, *Buzdalov M.* Adaptive Selection of Helper-Objectives with Reinforcement Learning / Proceedings of the International Conference on Machine Learning and Applications. Т. 2. IEEE, 2012. С. 66—67.
28. *Afanasyeva A.*, *Buzdalov M.* Optimization with Auxiliary Criteria using Evolutionary Algorithms and Reinforcement Learning / Proceedings of 18th International Conference on Soft Computing MENDEL 2012. Brno, Czech Republic, 2012. С. 58—63.
29. *Afanasyeva A.*, *Buzdalov M.* Choosing Best Fitness Function with Reinforcement Learning / Proceedings of the Tenth International Conference on Machine Learning and Applications. Т. 2. Honolulu, HI, USA: IEEE Computer Society, 2011. С. 354—357.
30. *Buzdalov M.*, *Buzdalova A.* Adaptive Selection of Helper-Objectives for Test Case Generation / 2013 IEEE Congress on Evolutionary Computation. Т. 1. 2013. С. 2245—2250.
31. *Buzdalov M.*, *Buzdalova A.*, *Shalyto A.* A First Step towards the Runtime Analysis of Evolutionary Algorithm Adjusted with Reinforcement Learning / Proceedings of the International Conference on Machine Learning and Applications. Т. 1. IEEE Computer Society, 2013. С. 203—208.
32. *Buzdalov M.*, *Buzdalova A.* OneMax Helps Optimizing XdivK: Theoretical Runtime Analysis for RMHC and EA+RL / Proceedings of Genetic and Evolutionary Computation Conference (to be published). ACM, 2014.
33. *Buzdalov M.* Generation of Tests for Programming Challenge Tasks Using Evolution Algorithms / Proceedings of Genetic and Evolutionary Computation Conference Companion. New York, US, ACM, 2011. С. 763—766.
34. *Buzdalov M.*, *Buzdalova A.*, *Petrova I.* Generation of Tests for Programming Challenge Tasks Using Multi-Objective Optimization / Proceedings of Genetic and Evolutionary Computation Conference Companion. ACM, 2013. С. 1655—1658.
35. *Buzdalova A.*, *Buzdalov M.*, *Parfenov V.* Generation of Tests for Programming Challenge Tasks Using Helper-Objectives. B: 5th International Symposium on Search-Based Software Engineering. Т. 8084. Lecture Notes in Computer Science. Springer, 2013. С. 300—305.
36. Timus Online Judge. Problem “Ships. Version 2”. http://acm.timus.ru/problem.aspx?num=1394.
37. *Deb K.*, *Pratap A.*, *Agarwal S.*, *Meyarivan T.* A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II // Transactions on Evolutionary Computation. 2000. С. 182—197.