Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Факультет информационных технологий и программирования
Кафедра компьютерных технологий

Петрова Ирина Анатольевна

Повышение эффективности алгоритмов многокритериальной оптимизации с помощью машинного обучения для решения задач составления расписаний

Научный руководитель: ассистент кафедры ТП М. В. Буздалов

Санкт-Петербург
2013

Содержание

Введение **5**

Глава 1. Обзор предметной области **6**

1.1 Задача оптимизации 6
1.1.1 Задача однокритериальной оптимизации 6
1.1.2 Задача многокритериальной оптимизации 6
1.1.3 Эволюционные алгоритмы 6
1.1.4 Эволюционные алгоритмы многокритериальной оптимизации 7
1.2 Вспомогательные критерии оптимизации 7
1.3 Задачи составления расписаний 8
1.3.1 Классические варианты задачи Job Shop 8
1.3.2 Другие вариации задачи Job Shop 9
1.3.3 Другие варианты задач составлении расписаний . . . 10
1.3.4 Графическое представленние задачи Job Shop 10
1.4 Методы решения задачи Job Shop 11
1.4.1 Применение эволюционных алгоритмов 11
1.4.2 Метод применения правил (Dispatching rules) 13
1.4.3 Метод удовлетворения ограничениям (Constraint satisfaction problem, CSP) 14
1.4.4 Нейронные сети 15
1.4.5 Сведение Job Shop к задаче многокритериальной оптимизации 15
1.5 Обучение с подкреплением 16
1.5.1 Марковский процесс принятия решений 18
1.6 Выводы по главе 1 18

Глава 2. Постановка задачи и описание метода ее решения . **20**

2.1 Требования, предъявляемые к разрабатываемому методу . 20
2.2 Общее описание метода 20
2.3 Метод MOEA+RL 21
2.4 Модификация метода MOEA+RL 22

2.5 Награды, состояния и агенты, применяемые в данных методах 23
2.6 Выводы по главе 2 26

Глава 3. Результаты применения метода 27

3.1 Общие конфигурации экспериментов 27
3.2 Эксперимент, демонстрирующий работу метода MOEA+RL 29
3.3 Эксперимент, демонстрирующий работу модификации метода MOEA+RL 30
3.4 Выводы по главе 3 32

Заключение 34

Список литературы 35

Введение

Существует метод, позволяющий повысить эффективность решения задачи однокритериальной оптимизации, сводя ее к задаче многокритериальной оптимизации, используя вспомогательные критерии [1—3]. Не всегда эффективно оптимизировать одновременно все критерии [1, 3], поэтому необходим метод выбора критериев в ходе процесса оптимизации. В работе [1] показано, что лучше всего использовать один дополнительный критерий, но оптимального способа выбора данного критерия еще не найдено [1, 2]. В настоящей работе предлагается метод повышения эффективности многокритериальных эволюционных алгоритмов с помощью обучения с подкреплением для выбора вспомогательного критерия, основанный на ранее предложенном методе повышения эффективности однокритериальных эволюционных алгоритмов [3, 4]. Эффективность разработанного метода демонстрируется на примере задачи Job Shop.

Существует несколько разновидностей задач составления расписаний, в частности, задача, известная под названием Job Shop. Эффективное решение задач составления расписаний важно для практических целей, таких как распределение задач в многопроцессорной системе, составление железнодорожных расписаний, управление воздушным движением [5].

Задача Job Shop NP-полна [6], поэтому на практике решается при помощи некоторых эвристик. Одна из них [1, 2] состоит в том, чтобы свести задачу Job Shop к задаче многокритериальной оптимизации, вводя вспомогательные критерии, выбираемые в ходе процесса оптимизации. В настоящей работе предлагается проверить применимость разработанного метода выбора вспомогательного критерия для решения данной задачи.

Глава 1. Обзор предметной области

1.1. Задача оптимизации

Рассмотрим различные варианты задачи оптимизации.

1.1.1. Задача однокритериальной оптимизации

Задача однокритериальной оптимизации формулируется следующим образом: максимизировать заданную функцию $f(x) : X \to \mathbb{R}$, где X — область допустимых значений функции f. Решением является $x^* \in X : f(x^*) \geq f(x)$ для всех $x \in X$.

1.1.2. Задача многокритериальной оптимизации

Предположим, что имеется совокупность функций $f = f_1(x), f_2(x), ...f_n(x)$, таких, что $f_i : X \to \mathbb{R}$. Тогда решением задачи многокритериальной оптимизации является $x^* \in X : f(x^*) \geq f(x)$. Для сравнения значений f может быть применен критерий оптимальности по Парето [7].

1.1.3. Эволюционные алгоритмы

Эволюционные алгоритмы [8] применяются для решения задач оптимизации и используют механизмы, основанные на принципах природной эволюции. Кандидаты на оптимальное решение задачи представляются особями. На каждой итерации алгоритма, определяющей поколение, существует множество особей, характеризующих данное поколение. Функция, показывающая насколько данное решение близко к оптимальному, называется функцией приспособленности (ФП). Алгоритм генерирует особи для следующего поколения, применяя к особям текущего поколения операторы отбора, скрещивания и мутации, что позволяет множеству особей эволюционировать от поколения к поколению. Эволюционный алгоритм закан-

чивает выполнение, когда достигнуто одно из условий останова. В данной работе использовались следующие условия останова:

- сгенерировано заданное число поколений;
- найдено решение задачи.

1.1.4. Эволюционные алгоритмы многокритериальной оптимизации

Метод, который предполагается улучшить в ходе выполнения работы, основан на применении эволюционных алгоритмов, осуществляющих многокритериальную оптимизацию. Одним из самых эффективных эволюционных алгоритмов многокритериальной оптимизации (Multi-objective evolutionary algorithm, MOEA), известных на данный момент [9], является NSGA-II [10]. В данном алгоритме при выборе особей используется понятие оптимальности по Парето. Оценка времени работы алгоритма NSGA-II — $O(GN \log^{M-1} N)$, где G — число поколений, N — количество особей в поколении, M — число оптимизируемых параметров.

1.2. Вспомогательные критерии оптимизации

При решении задач однокритериальной оптимизации при помощи эволюционных алгоритмов часто возникают проблемы, такие, как остановка поиска решения в локальном оптимуме, малое разнообразие особей. Для решения данных проблем существует метод сведения однокритериальной задачи оптимизации к многокритериальной [1]. Рассмотрим два подхода к использованию данного метода. Одним из них является разбиение задачи на подзадачи оптимизации и применение MOEA для одновременной оптимизации всех подзадач [11]. Однако, если для решения задачи оптимизация какого-либо из критериев неэффективна, то такой метод работает плохо [3, 4]. Второй подход применим при условии, что оптимизируемая функция некоторым образом зависит от некоторых дополнительных функций. Тогда эффективность решения задачи однокритериальной опти-

мизации можно повысить, сведя эту задачу к многокритериальной, взяв дополнительные функции в качестве вспомогательных критериев [1, 3, 4], а оптимизируемую функцию в качестве целевого критерия. Задача оптимизации вспомогательных критериев не ставится, они используются лишь для повышения эффективности оптимизации целевого критерия. Не всегда эффективно оптимизировать все критерии одновременно [3, 4]. Поэтому на каждом шаге алгоритма происходит выбор оптимизируемых критериев из списка вспомогательных. Таким образом, оптимизация происходит по целевому критерию и выбранным вспомогательным критериям.

В работах [3, 4] представлен метод для повышения эффективности решения задачи однокритериальной оптимизации с помощью эволюционных алгоритмов. В нем используется не одна ФП, а несколько и для каждого поколения ФП выбирается при помощи обучения с подкреплением.

В данной работе предлагается метод выбора вспомогательного критерия многокритериального оптимизационного эволюционного алгоритма, разработанный на основе существующего метода выбора ФП для однокритериальной оптимизации.

1.3. Задачи составления расписаний

Рассмотрим различные варианты задач составления расписаний.

1.3.1. Классические варианты задачи Job Shop

В задаче Job Shop рассматривается n работ и m машин. Для каждой работы i задана последовательность из m операций $(o_{i1}, o_{i2}, ..., o_{im})$. Каждая операция o_{ij} имеет время выполнения τ_{ij} и должна быть сделана на конкретной машине. Машины не могут выполнять более одной операции одновременно. Операции неделимы, то есть если операция начала выполняться, то она обязана выполниться до конца. Операции, относящиеся к одной и той же работе, не могут выполняться одновременно. Цель задачи Job Shop — составить расписание так, чтобы общее время выполнения

работ было минимальным. В качестве общего времени выполнения работ может быть выбрана длина периода обработки (makespan), а именно время прошедшее с момента начала выполнения первой операции до момента завершения последней. Также для задач, где необходимо минимизировать длину перерывов между выполнением операций работы [12], в качестве общего времени выполнения работ может использоваться F_Σ, которое вычисляется по формуле: $F_\Sigma = \sum_{i=1}^{n} F_i$, где F_i — время выполнения работы i, то есть время, прошедшее с момента начала выполнения работы до момента завершения последней операции работы i.

К общей постановке задачи могут быть добавлены дополнительные условия:

- Машины могут требовать определенного времени между выполнением работ, или, наоборот, отсутствия простоя.
- Целью задачи может являться минимизация не только суммарного времени, но некоторых других критериев.
- На работы могут накладываться ограничения. Например, работу i необходимо закончить до начала выполнения работы j.
- Время выполнения работы может быть фиксированное или вероятностное.

1.3.2. Другие вариации задачи Job Shop

На практике часто требуется решить более сложные задачи составления расписаний, базирующиеся на Job Shop [5]. Рассмотрим некоторые из них.

- Представим, что на заводе имеются машины, оборудованные некоторыми инструментами для выполнения работ, и каждую операцию в работе можно выполнять на машине с необходимыми инструментами. Иными словами, для каждой операции определена не одна машина, на которой ее можно выполнить, а список машин.

- Пусть работами являются компьютерные программы, а операциями — инструкции этих программ. У каждой операции есть список процессоров, на которых она может выполняться, притом она выполняется одновременно на всех процессорах из списка. Каждый процессор не может выполнять несколько задач одновременно. Таким образом, если у двух работ пересечение списков процессоров непусто, одновременно они выполняться не могут.
- Задача о составлении железнодорожного расписания. Пусть железная дорога разделена на некоторые участки пути, притом на каждом из таких участков одновременно может находиться только один поезд. Тогда в качестве работы можно рассмотреть передвижение поезда из точки отправления в точку назначения, а в качестве времени операции — время на преодоление некоторого участка пути.

1.3.3. Другие варианты задач составлении расписаний

Рассмотрим задачи составления расписаний, отличные от Job Shop.

- Flow Shop [6] — задача составления расписаний, где порядок выполнения работ четко определен. Целью задачи является минимизация времени простоя машин и времени выполнения работ. Примером Flow Shop является задача, где результаты работ передаются от одной машины к другой одним или несколькими роботами. Цель — составить расписание действий машин и роботов.
- Open Shop [13] — задача составления расписаний, где, в отличие от задачи Job Shop, порядок выполнения операций в работе не определен.

1.3.4. Графическое представленние задачи Job Shop

Одним из вариантов представления задачи Job Shop является диаграмма Гантта. На рис. 1.1 [14] приведен пример такой диаграммы для задачи состоящей из десяти работ (Job) и десяти машин (M0 — M9). Рас-

смотрим, к примеру, четвертую работу. Ее первая операция выполняется на машине M1. Вторая работа, которой необходима машина M1 для выполнения третьей операции, ждет, пока закончится выполнение первой операции четвертой работы, так как на машине может выполняться не более одной операции одновременно.

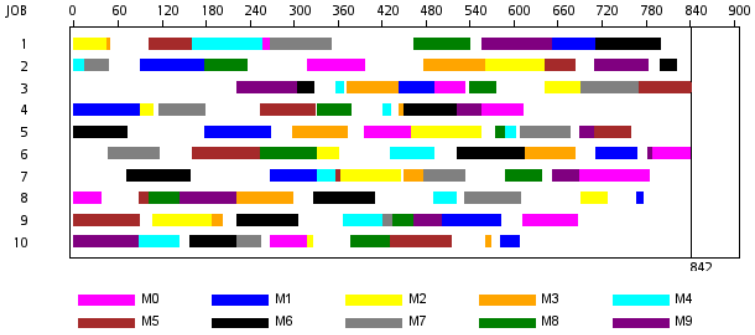

Рис. 1.1: Решение задачи Job Shop, состоящей из 10 работ и 10 машин

1.4. Методы решения задачи Job Shop

Рассмотрим некоторые методы решения задачи Job Shop.

1.4.1. Применение эволюционных алгоритмов

Наиболее распространенным методом решения задачи Job Shop является применение эволюционных алгоритмов. В качестве особи берется некоторое представление порядка выполнения работ. Наиболее широко используется метод кодирования особи при помощи перестановок с повторениями [1, 15]. Особь является вектором, состоящим из номеров работ. Если работа j имеет m операций, то j встречается в векторе ровно m раз. В качестве примера рассмотрим особь для задачи Job Shop, состоящей из трех работ и двух машин: $(1, 2, 1, 3, 2, 3)$. В данном примере каждая работа состоит из двух операций, поэтому номер каждой работы встречается в особи дважды. Для составления расписания выполнения операций по таким образом закодированной особи применяется алгоритм Гиффлера-Томпсона [1, 15]. Порядок работ в особи определяет приоритеты выполнения операций.

В приведенном выше примере наибольший приоритет имеет первая операция первой работы, за ней следует первая операция второй работы, далее вторая операция первой работы и т.д. Алгоритм Гиффлера-Томпсона применяется с учетом приоритетов выполнения операций. Преимуществом кодирования особи таким способом является то, что все особи являются корректными решениями задачи Job Shop.

При кодировании особи при помощи перестановок с повторениями используется оператор скрещивания, называемый Generalized Order Crossover (GOX) [16]. Кратко опишем принцип работы данного оператора. На рис. 1.2 изображены две родительские особи: $(1, 2, 1, 3, 2, 3)$ и $(2, 1, 3, 1, 3, 2)$. Индекс — номер операции в работе. Из первой особи выбирается подстрока s случайной длины, на примере это $(1, 3, 2)$. Она вставляется во вторую особь после работы, имеющей такой же номер и такой же индекс, как первая работа в s. Далее из полученной особи, из части, унаследованной от второй особи, удаляются те номера работ, которые имеют такой же номер и индекс, как в s.

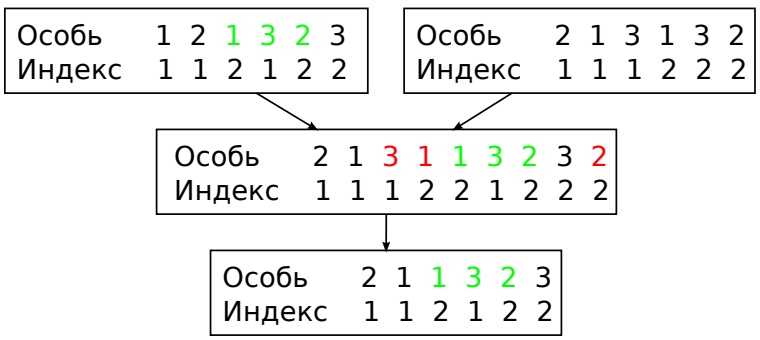

Рис. 1.2: Оператор скрещивания

В качестве оператора мутации используется Position Based Mutation [15]. Данный оператор мутации выбирает две случайных позиции в особи и меняет их местами.

Существует несколько различных вариантов повышения эффективности генетического алгоритма. Рассмотрим некоторые из них.

- Если не все особи в поколении являются корректными решениями задачи, можно использовать штрафную функцию [17]. К функции приспособленности особи, являющейся некорректным решением, добавляется штрафная функция, характеризующая насколько данная особь далека от того, чтобы являться решением. При кодировании особи при помощи перестановок с повторениями нет необходимости использовать данный подход, так как все особи являются корректными.
- Использование дифференциальной эволюции [18]. Кратко опишем идею дифференциальной эволюции. Для генерации нового поколения для каждой особи x_i из текущего поколения выбираются три различные особи x_1, x_2, x_3 , не равные x_i, и создается новая особь u, определяющаяся следующим образом: $u = x_1 + F \cdot (x_2 - x_3)$, где F — некоторая константа. Далее в результате скрещивания особей x_i и u создается новая особь x_i'. В случае если особь x_i' лучше приспособлена, чем x_i, то в новое поколение добавляется особь x_i', иначе x_i. Также известно применение дифференциальной эволюции совместно с методом применения правил для решения задачи Job Shop, где в качестве общего времени выполнения работ используется длина периода обработки [19]. Однако в настоящей работе в качестве общего времени выполнения работ используется F_Σ, поэтому сравнение с данным методом не производится.

1.4.2. Метод применения правил (Dispatching rules)

Данный метод широко применяется для решения задач составления расписаний [19, 20]. Кратко опишем идею данного подхода. Предположим, что освободилась машина, которую ждали несколько работ. Тогда для выбора работы, которая будет исполняться на освободившейся машине,

применяются правила. Наиболее часто использующиеся в задаче Job Shop правила:

- выбор той работы, которая ожидает дольше всего;
- выбор той работы, у которой выполняемая операция занимает меньше времени, чем у всех остальных работ.

Использование нескольких правил эффективнее, чем использование одного правила [21]. В таблице 1.1 приведено сравнение результатов, полученных при применении данного метода [21] и эволюционного алгоритма [17] для задач Job Shop, состоящих из десяти работ и десяти машин, где в качестве общего времени работ использовалась длина периода обработки. Можно видеть, что использование метода применения правил менее эффективно, чем использование эволюционных алгоритмов. Однако метод применения правил может использовать для повышения эффективности эволюционных алгоритмов [19].

Таблица 1.1: Сравнение метода применения правил и эволюционного алгоритма

Название задачи	Метод применения правил	Эволюционный алгоритм
ft10	1074	930
la16	1156	945
la17	913	784
la18	981	848
la19	940	844
la20	1000	907

1.4.3. Метод удовлетворения ограничениям (Constraint satisfaction problem, CSP)

Если в задаче Job Shop существуют дополнительные ограничения на минимально возможное начало выполнения работы и максимально возможное время окончания выполнения работы, то возможно применение алгоритмов решения задачи удовлетворения условий (Constraint Satisfaction Problem, CSP) к решению задачи Job Shop [22—24]. В рамках настоящей работы сравнение данного подхода и метода, использующего предложенный способ выбора вспомогательного критерия, не производится.

1.4.4. Нейронные сети

Одним из методов решения задачи Job Shop является применение нейронных сетей [25, 26]. Используются следующие типы нейронных сетей:

- нейронные сети с обратным распространением ошибки [27];
- нейронные сети Хопфилда [28].

Кратко опишем идею применения нейронной сети для решения задачи Job Shop. При помощи нейронной сети определяется приоритет каждой операции. На вход нейронной сети подается набор свойств операции, например, время выполнения операции, номер операции в работе и т.д. На выходе нейронная сеть выдает приоритет операции. Далее, учитывая приоритет каждой операции, алгоритм Гиффлера-Томпсона строит расписание выполнения операций [26]. Также для повышения эффективности решения задачи Job Shop при помощи нейронных сетей их применяют совместно с эволюционными алгоритмами [26]. Метод с использованием нейронных сетей для решения задачи Job Shop, по сравнению с эволюционными алгоритмами, менее эффективен [26].

1.4.5. Сведение Job Shop к задаче многокритериальной оптимизации

Одним из методов решения задачи Job Shop является сведение к задаче многокритериальной оптимизации. В работе [1] рассматривается вариант задачи Job Shop, описанный в разделе 1.3.1, где в качестве общего времени выполнения работ используется F_Σ. В качестве целевой функции берется F_Σ, а в качестве вспомогательных критериев используется время выполнения отдельных работ. В статье [1] показано, что наилучшие результаты достигаются при использовании одного вспомогательного критерия, выбираемого случайным образом из списка возможных критериев на каждой итерации эволюционного алгоритма многокритериальной оптимизации. Также показано, что при выборе в качестве общего времени выполнения работ F_Σ, данный метод работает лучше, чем традиционный

эволюционный алгоритм.

В статье [2] описывается способ повышения эффективности алгоритма [1]. В отличие от подхода, предложенного в статье [1], вспомогательные критерии выбираются не случайным образом, а в порядке возрастания минимально возможного времени выполнения работы, соответствующей вспомогательному критерию. Также в данной статье исследовался выбор в качестве вспомогательных критериев суммарного времени выполнения нескольких работ. Работы распределяются по вспомогательным критериям следующим образом: предположим, что каждый критерий состоит из p работ. Тогда первый критерий состоит из первых p среди отсортированных работ, второй — из следующих p и так далее. Результаты экспериментов показали, что наиболее эффективен метод со вспомогательными критериями, состоящими из одной работы.

Однако метод представленный в статье [2] является эвристикой случайного выбора. В данной работе предлагается применение нового автоматизированного подхода к выбору вспомогательного критерия, основанного на обучении с подкреплением к задаче Job Shop.

1.5. Обучение с подкреплением

Кратко опишем идею обучения с подкреплением [3, 29]:

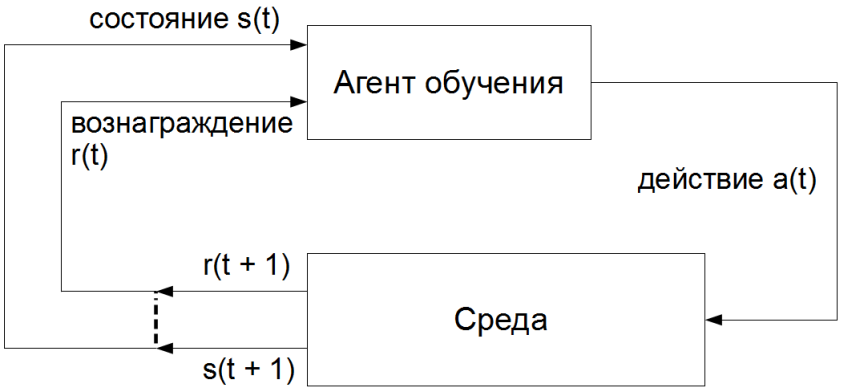

Рис. 1.3: Схема обучения с подкреплением

На каждом шаге агент обучения выбирает из имеющегося набора действий A некоторое действие a, на что среда сообщает ему, какую на-

граду r он за это получил и в каком состоянии оказался. Агент должен разработать тактику взаимодействия со средой, максимизирующую общую награду.

Рассмотрим некоторые алгоритмы обучения с подкреплением, применявшиеся в данной работе. Были выбраны именно такие алгоритмы, так как они хорошо зарекомендовали себя в задачах, связанных с выбором вспомогательных функций приспособленности [3, 4].

- Алгоритм Q-обучения. В данном алгоритме максимизируется функция $Q(s, a)$ — ожидаемая награда за действие a в состоянии s. Алгоритм обладает низкой скоростью сходимости, но не требует больших вычислительных ресурсов. Рассмотрим некоторые стратегии исследования среды, которые применялись в данной работе совместно с алгоритмом Q-обучения.
  - Жадная стратегия исследования среды. При таком подходе всегда выбирается действие, позволяющее получить максимальную ожидаемую награду. Минусом такой стратегии является то, что агент может остановиться в локальном оптимуме, не успев достаточно исследовать среду.
  - ε-жадная стратегия исследования среды. При таком подходе с вероятностью $1 - \varepsilon$ выбирается действие с максимальной ожидаемой наградой, а с вероятностью ε выбирается случайное действие. Такая стратегия в отличие от жадной позволяет агенту выбраться из локального оптимума.
  - Стратегия исследования среды по Больцману. При таком подходе в состоянии s действие a выбирается с вероятностью $\frac{\exp(\frac{G(s,a)}{\tau})}{\sum\limits_{a' \in A} \exp(\frac{G(s,a')}{\tau})}$, где τ — температурный коэффициент.
- Алгоритм отложенного Q-обучения [29, 30]. Данный алгоритм является модификацией алгоритма Q-обучения с жадной стратегией исследования среды. Преимуществом такого алгоритма является боль-

шая скорость сходимости, чем у алгоритма Q-обучения [30].

1.5.1. Марковский процесс принятия решений

Алгоритм Q-обучения работает в предположении, что взаимодействует со средой, описываемой марковским процессом принятия решений. Марковский процесс принятия решений (МППР) определяется следующими составляющими [29]:

- множество состояний S;
- множество действий A;
- функция вероятности перехода из состояния $s \in S$ в состояние $s' \in S$ в результате применения действия $a \in A$;
- функция вознаграждения $R(s, a)$ за применение действия a в состоянии s.

В МППР состояние, в которое перейдет система, находящаяся в состоянии s, при применении действия a, зависит только от текущего состояния s и примененного действия a.

1.6. Выводы по главе 1

Описаны задачи однокритериальной и многокритериальной оптимизации. Описан метод повышения эффективности однокритериальной оптимизации путем сведения к задаче многокритериальной оптимизации. Сведение производится с помощью введения вспомогательных критериев. Обоснована необходимость разработки метода адаптивного выбора вспомогательного критерия.

Описана задача составления расписаний, известная под названием Job Shop. Кратко описаны существующие подходы к решению данной задачи. Обозначена применимость разработанного метода выбора вспомогательного критерия для решения задачи Job Shop.

Кратко описана идея обучения с подкреплением. Перечислены использующиеся в данной работе алгоритмы обучения с подкреплением и стратегии исследования среды. Дано понятие марковского процесса принятия решений, которому соответствует задача обучения с подкреплением.

Глава 2. Постановка задачи и описание метода ее решения

В данной главе описывается разработанный метод выбора вспомогательного критерия в эволюционном алгоритме многокритериальной оптимизации при помощи обучения с подкреплением.

2.1. Требования, предъявляемые к разрабатываемому методу

Целью данной работы является разработка метода адаптивного выбора вспомогательного критерия в многокритериальном алгоритме при помощи обучения с подкреплением и проверка применимости разработанного метода на примере задачи Job Shop, где в качестве общего времени выполнения работ используется суммарное время выполнения всех работ. Требования к данной исследовательской работе:

- Разработать метод адаптивного выбора вспомогательного критерия в многокритериальном алгоритме.
- Предложить различные способы применения обучения с подкреплением для выбора вспомогательного критерия.
- Провести сравнение предложенных способов применения обучения с подкреплением на примере задачи Job Shop.
- Достичь улучшения результатов, полученных в статье [2].

2.2. Общее описание метода

В данной работе для оптимизации использовался один вспомогательный критерий, так как ранее [1, 2] было получено, что наилучшие результаты достигаются при использовании именно одного вспомогательного критерия.

Обозначим множество вспомогательных критериев как $\{h_i\}$. Таким образом, оптимизация происходит по целевому критерию, который является суммарным временем выполнения всех работ, и вспомогательному критерию, выбираемому на каждом шаге MOEA из множества $\{h_i\}$.

В качестве MOEA в данной работе использовался NSGA-II [10, 31], так как это один из наиболее эффективных эволюционных алгоритмов многокритериальной оптимизации. Также NSGA-II применялся в подходах, описанных в статьях [1, 2], с которыми сравнивается разработанный метод.

2.3. Метод MOEA+RL

Предлагаемый метод MOEA+RL основан на применении эволюционного алгоритма многокритериальной оптимизации, где выбор вспомогательного критерия осуществляется при помощи обучения с подкреплением. Схема предлагаемого метода представлена на рис. 2.1. Агент применяет действие — выбор вспомогательного критерия — к среде, ассоциируемой с алгоритмом многокритериальной оптимизации. Среда при помощи полученного критерия генерирует следующее поколение. Затем агент получает награду, зависящую от значения целевого критерия в новом поколении. Среда переходит в новое состояние, зависящее от значений критериев в сгенерированном поколении, и процесс повторяется. Псевдокод данного алгоритма приведен на листинге 2.1.

Листинг 2.1 Метод MOEA+RL

```
Сгенерировать начальное поколение G_0
Инициализировать номер поколения: n = 0
Вычислить начальное состояние среды
while (условие останова эволюционного алгоритма не выполнено) do
    Выбрать вспомогательный критерий и передать его среде
    Сгенерировать следующее поколение G_{n+1}
    Вычислить награду r(G_{n+1}) и передать ее агенту
    Вычислить и обновить состояние среды
    Обновить номер поколения: n ← n + 1
end while
```

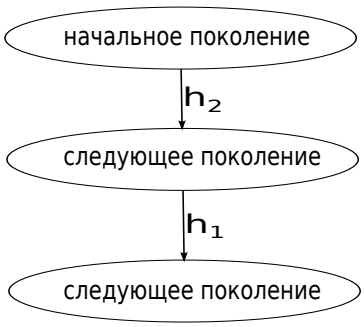

Рис. 2.1: Схема метода MOEA+RL

2.4. Модификация метода MOEA+RL

Недостатком метода MOEA+RL, описанного в разделе 2.3, является то, что при выборе неэффективного вспомогательного критерия особи могут заменяться на менее приспособленные. Для устранения описанного недостатка предлагается модификация метода MOEA+RL. Схема данного метода представлена на рис. 2.2. Предположим, что имеется c вспомогательных критериев. Для каждого критерия h_i среда t_i раз генерирует следующее поколение и возвращает агенту награду, зависящую от значения целевого критерия в новом поколении. Таким образом, в результате перебора всех критериев генерируется c новых поколений. Агент выбирает из них то поколение, на котором получена наибольшая награда, и для каждого h_i обновляет значение t_i. Среда переходит в новое состояние, соответствующее критерию, при помощи которого было сгенерировано выбранное поколение, и процесс повторяется. Псевдокод данного алгоритма приведен на листинге 2.2.

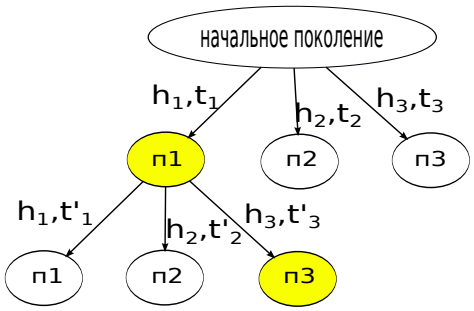

Рис. 2.2: Схема метода MOEA+RL

Листинг 2.2 Модификация метода MOEA+RL

1: Сгенерировать начальное поколение G_0
2: Инициализировать номер поколения: $n = 0$
3: Вычислить начальное состояние среды
4: **while** (условие останова эволюционного алгоритма не выполнено) **do**
5: **for** ($h_i \in h$) **do**
6: Инициализировать номер поколения: $j = 0$
7: **while** (j не равно t_i) **do**
8: Сгенерировать следующее поколение g_j
9: Вычислить награду $r(g_j)$ и передать ее агенту
10: Обновить номер поколения: $j \leftarrow j + 1$
11: **end while**
12: **end for**
13: Выбрать следующее поколение: $G_{n+1} = g_{max}$, где $r(g_{max}) = \max\limits_i r(g_{t_i})$
14: Обновить номер поколения: $n \leftarrow n + 1$
15: Вычислить и обновить состояние среды
16: Обновить значения t_i
17: **end while**

2.5. Награды, состояния и агенты, применяемые в данных методах

Для применения обучения с подкреплением необходимо определить то, как будут вычисляться состояния и награды среды, моделирующей эволюционный алгоритм многокритериальной оптимизации.

Определим несколько понятий, использующихся в описании наград и состояний.

- Насыщенность критериев определяется следующим образом: $sat(h_i) = \frac{h_i - optima(h_i)}{optima(h_i)}$, где $optima(h_i)$ — минимально возможное

значение h_i.

- Лучшая особь поколения — особь, у которой значение целевого критерия максимально.

Состояния:

- S_1 — одиночное состояние. При таком подходе у среды есть только одно состояние.
- S_2 — состояние, отражающее насыщенность критериев. При таком подходе состояние кодируется как вектор, состоящий из нулей и единиц, длина которого равна числу критериев. Если критерий в лучшей особи нового поколения более насыщен, чем в лучшей особи текущего поколения, то в ячейке вектора состояний, соответствующей данному критерию будет стоять единица, иначе ноль. Недостатком такого подхода является большое число состояний, вследствие чего агенту требуется большое число итераций обучения.
- S_3 — состояние, определяющееся наименее насыщенным критерием. При таком определении состояния число состояний среды равно числу вспомогательных критериев. Среда переходит в состояние, соответствующее наименее насыщенному критерию в новом поколении.
- S_4 — состояние, зависящее от номера поколения. Предположим, что всего в алгоритме создается n поколений и у среды имеется z состояний. Тогда функцией для определения состояния по поколению будет $s(t) = z - \lceil \frac{\log(n+1-t)}{\log((n+1)^{\frac{1}{z}})} \rceil$. На рис. 2.3 приведено соответствие состояния поколению для $n = 7$ и $z = 3$.

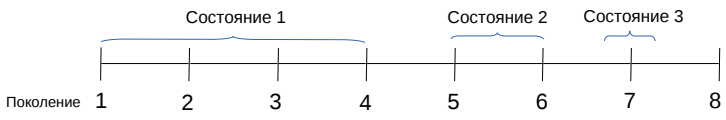

Рис. 2.3: Состояние, зависящее от номера поколения

- S_5 — состояние, завясящее от значения целевого критерия. Этот подход аналогичен предыдущему, только вместо номера состояния используется значение целевого критерия для лучшей особи нового поколения.

Награды:

- R_1 — награда, учитывающая приращение значений всех критериев для особей, составляющих Парето-фронт. Приращение значения каждого критерия считается следующим образом: $\text{diff}(h_i) = \frac{\text{cur}(h_i) - \text{new}(h_i)}{\text{cur}(h_i)}$, где $\text{cur}(h_i)$ — среднее значение критерия h_i по особям, составляющим Парето-фронт текущего поколения, $\text{new}(h_i)$ — среднее значение значение каждого критерия по особям, составляющим Парето-фронт нового поколения. Награда, возвращаемая агенту, определяется следующим образом: $r = \sum_{h_i} \text{diff}(h_i) \cdot \text{discount}(h_i)$, где $\text{discount}(h_i) \in [0; 1]$ — это коэффициент, определяющий, с каким весом учитывается приращение вспомогательных критериев. В случае, если h_i — целевой критерий, $\text{discount}(h_i) = 1$
- R_2 — награда, учитывающая приращение значения целевого критерия на особях, составляющих Парето-фронт. При таком подходе награда определется следующим образом $r = \text{diff}(\text{target})$, где target — целевой критерий, а diff определяется так, как и в предыдущей награде.
- R_3 — награда, учитывающая приращение значения целевого критерия в лучшей особи. Награда, возвращаемая агенту, определяется следующим образом $r = \text{new} - \text{cur}$, где new — значение целевого критерия в лучшей особи нового поколения, cur — значение целевого критерия в лучшей особи текущего поколения.

Обучение с подкреплением хорошо работает в ситуациях, которые можно представить в виде приближения МППР [32]. С учетом этого состояния и награды должны быть разработаны таким образом, чтобы в них

не учитывалась предыстория (предыдущие поколения эволюционного алгоритма). Описанные награды и состояния не учитывают предысторию, и поэтому могут быть использованы для обучения с подкреплением.

В данной работе применялись следующие агенты. Агенты A_1 и A_2 были специально разработаны для выбора того, сколько раз обучаться на каждом из критериев в модификации метода MOEA+RL.

- A_1 — агент, выбирающий случайным образом, сколько раз обучаться на каждом из критериев. Таким образом значения t_i выбираются случайным образом.
- A_2 — агент, обучающийся на каждом из критериев одинаковое число раз. Таким образом, все t_i имеют одинаковые значения.
- A_3 — агент, использующий Q-обучение с исследованием среды по Больцману.
- A_4 — агент, использующий Q-обучение с применением ε-жадной стратегии.
- A_5 — агент, использующий отложенное Q-обучение.

2.6. Выводы по главе 2

Формализована цель исследований: разработка метода адаптивного выбора вспомогательного критерия для повышения эффективности оптимизации целевого критерия с помощью многокритериальной оптимизации. Описаны требования, предъявляемые к данной работе.

Описан предлагаемый метод, использующий обучение с подкреплением. Описаны разработанные награды, состояния и агенты, необходимые для применения обучения с подкреплением.

Глава 3. Результаты применения метода

В данной главе описываются эксперименты, демонстрирующие работу предложенного метода выбора вспомогательного критерия в эволюционном алгоритме многокритериальной оптимизации при помощи обучения с подкреплением на примере задачи Job Shop.

3.1. Общие конфигурации экспериментов

Опишем конфигурации экспериментов, демонстрирующих применимость разработанных методов.

В данной работе в качестве эволюционного алгоритмов многокритериальной оптимизации использовался NSGA-II. В качестве вспомогательных критериев использовалось суммарное время выполнения k работ: $h_i = \sum_{j=k(i-1)+1}^{ik} F_j$. Общее число критериев, определяемых таким образом, равно $\lfloor \frac{n}{k} \rfloor$, где n — число работ.

Аналогично подходу в статье [1], для кодирования особи использовался метод перестановок с повторениями, для составления расписания по данной особи применялся метод Гиффлера-Томпсона. В качестве оператора отбора использовался турнирный отбор. Также, как и в статье [1], в качестве оператора скрещивания использовался Generalized Order Crossover, а в качестве оператора мутации — Position Based Mutation. Аналогично подходу в статье [1], после применения оператора скрещивания всегда применялся оператор мутации.

В ходе предварительных экспериментов было получено, что лучшие результаты достигаются при числе особей в поколении, равном 150. Эксперименты проводились на условиях задач Job Shop, использующихся в статьях [1, 2]. В таблице 3.1 приведено соответствие между условием задачи и ее размером.

Все алгоритмы были реализованы на языке программирования *Java*, также использовалась библиотека эволюционных алгоритмов *Watchmaker* [33]. Для проведения экспериментов была написана программа на языке *Scala*.

Таблица 3.1: Размеры задач Job Shop

Номер задачи	Размер
la01, la02	10×5
la26, la27, swv01, swv02	20×10
la11, la12, ft20	20×5
la16, la17, ft10	10×10
swv06, swv07	20×15

Для всех алгоритмов было произведено по 200 запусков на каждом условии задачи. В каждом из запусков генерировалось 500 поколений.

Результаты работы алгоритмов, приведенные в таблицах, посчитаны по формуле $\frac{\text{average}-\text{best}}{\text{best}} \cdot 100\%$, где average — средний результат, полученный в алгоритме, best — лучшее известное значение.

В таблице 3.2 приведены значения параметров эволюционного алгоритма, использовавшиеся в экспериментах.

Таблица 3.2: Значения параметров эволюционного алгоритма

Параметр	Значение
Число запусков алгоритма	200
Число особей в поколении	150
Число поколений	500
Вероятность кроссовера	0,8

В таблице 3.3 приведены значения параметров алгоритмов обучения, подобранные в ходе предварительных экспериментов.

Таблица 3.3: Значения параметров алгоритмов обучения

Параметр	Описание	Значение
Q-learning		
α	скорость обучения	0,6
γ	дисконтный фактор	0,7
ε	параметр для ε-жадной стратегии	0,25
τ	параметр для исследования среды по Больцману	10
Delayed Q-learning		
m	период обновления	5
γ	дисконтный фактор	0,7
ϵ	бонусное вознаграждение	0,1

3.2. Эксперимент, демонстрирующий работу метода MOEA+RL

Для демонстрации работы метода MOEA+RL был проведен ряд экспериментов, с использованием описанных выше состояний, наград и агентов.

В таблице 3.4 представлены наилучшие результаты работы метода MOEA+RL. Они были получены при применении агента A_5, использующего отложенное Q-обучение, и награды R_1, учитывающей приращение значений всех критериев на Парето-фронте. Наиболее эффективным оказалось использование одиночного состояния (S_1) и состояния (S_4), зависящего от номера поколения, при значении параметра z, равном трем.

В первой колонке таблицы указано условие задачи, во второй — наилучшее известное решение задачи. В третьей колонке приведен результат работы алгоритма, описанного в статье [2]. В остальных колонках приведены результаты применения метода MOEA+RL с использованием двух, пяти и десяти вспомогательных критериев соответственно. В скобках указано, какое состояние использовалось для получения данного результата.

Можно видеть, что результаты, полученные при использовании метода MOEA+RL сравнимы с результатами, полученными при применении алгоритма из статьи. Однако результаты работы метода MOEA+RL не превосходят результатов работы алгоритма из статьи.

Наилучшие результаты были получены при использовании двух вспомогательных критериев, агента A_5, награды R_1 и состояния S_4.

Таблица 3.4: Результаты применения метода MOEA+RL

Название	Лучшее известное значение	Алгоритм из статьи, % от лучшего значения	2 критерия, % от лучшего значения	5 критериев, % от лучшего значения	10 критериев, % от лучшего значения
la01 (10 × 5)	4832	1,702	2,018(S_4)	2,437(S_4)	2,700(S_1)
la02 (10 × 5)	4459	1,755	2,177(S_4)	3,057(S_1)	3,075(S_1)
la16 (10 × 10)	7428	4,460	4,793(S_1)	5,362(S_4)	5,607(S_1)
la17 (10 × 10)	6582	3,014	3,178(S_4)	3,470(S_4)	3,817(S_4)
ft10 (10 × 10)	7606	7,431	7,584(S_4)	8,609(S_1)	9,011(S_4)

3.3. Эксперимент, демонстрирующий работу модификации метода MOEA+RL

Опишем эксперименты, которые были проведены для демонстрации работы модификации метода MOEA+RL.

Рассматриваются две конфигурации этого эксперимента. В одной из них используется обучение при обновлении значений t_i (агенты A_3 и A_4), а в другой не используется (агенты A_1 и A_2). В алгоритме отложенного Q-обучения используется специфическая стратегия исследования среды, изменение этой стратегии не предполагается. В то же время в модификации метода MOEA+RL используется стратегия, предложенная в данной работе, и поэтому агент, использующий отложенное Q-обучение, неприменим в данном подходе. В данных экспериментах использовались награды $R_1 - R_3$.

Наилучшие результаты работы модификации метода MOEA+RL представлены в таблицах 3.5 и 3.6. Все они были получены при использовании награды R_3, учитывающей приращение значения целевого критерия, вычисленного для лучшей особи.

В таблице 3.5 представлены наилучшие результаты работы алгоритма при применении агентов A_3 и A_4, использующих результаты обучения при обновлении значений t_i. В скобках указано, какой агент использовался для получения данного результата.

Можно видеть, что результаты, полученные при применении модификации метода MOEA+RL с использованием агентов A_3 и A_4, сравнимы с результатами работы алгоритма из статьи, однако в большинстве случаев их не превосходят. Наилучшие результаты работы модификации метода MOEA+RL при использовании агентов, применяющих результаты обучения при обновлении значений t_i, были получены при использовании двух вспомогательных критериев и агента A_3, использующего Q-обучение с исследованием среды по Больцману.

Рассмотрим конфигурацию метода, в которой применялись агенты

Таблица 3.5: Результаты применения модификации метода MOEA+RL с использованием агентов A_3 и A_4

Название	Лучшее известное значение	Алгоритм из статьи, % от лучшего значения	2 критерия, % от лучшего значения	5 критериев, % от лучшего значения	10 критериев, % от лучшего значения
la01 (10 × 5)	4832	1,702	1,761(A_3)	2,296(A_3)	2,873(A_4)
la02 (10 × 5)	4459	1,755	2,025(A_3)	2,974(A_4)	3,311(A_4)
la16 (10 × 10)	7393	4,460	4,526(A_3)	5,341(A_4)	5,937(A_4)
la17 (10 × 10)	6555	3,014	2,882(A_4)	3,748(A_3)	4,165(A_3)
ft10 (10 × 10)	7501	7,431	7,752(A_3)	8,413(A_3)	9,421(A_4)
la11 (20 × 5)	14805	3,249	6,621(A_3)	7,761(A_4)	7,621(A_4)
la12 (20 × 5)	12484	3,434	7,912(A_3)	8,672(A_4)	8,594(A_4)
la26 (20 × 10)	20234	7,870	9,463(A_4)	9,683(A_4)	9,421(A_4)
la27 (20 × 10)	20844	7,658	9,107(A_4)	9,415(A_4)	9,451(A_4)
ft20 (20 × 5)	14279	7,031	10,986(A_3)	12,046(A_4)	12,458(A_4)
swv01 (20 × 10)	20688	17,042	18,929(A_3)	20,050(A_4)	20,427(A_4)
swv02 (20 × 10)	21682	14,370	16,675(A_3)	17,620(A_4)	17,830(A_4)
swv06 (20 × 15)	28863	14,632	15,888(A_3)	16,562(A_4)	17,067(A_4)
swv07 (20 × 15)	27385	15,780	17,394(A_4)	18,262(A_4)	18,025(A_4)

A_1 и A_2, не использующие результаты обучения при обновлении значений t_i. Наилучшие результаты получены при использовании агента A_1. Результаты представлены в таблице 3.6.

Таблица 3.6: Результаты применения модификации метода MOEA+RL с использованием агентов A_1 и A_2

Название	Лучшее известное значение	Алгоритм из статьи, % от лучшего значения	2 критерия, % от лучшего значения	5 критериев, % от лучшего значения	10 критериев, % от лучшего значения
la01 (10 × 5)	4832	1,702	1,831	1,943	2,112
la02 (10 × 5)	4459	1,755	1,720	2,497	2,663
la16 (10 × 10)	7393	4,460	4,283	4,887	5,407
la17 (10 × 10)	6555	3,014	2,566	3,402	3,703
ft10 (10 × 10)	7501	7,431	6,675	7,978	8,073
la11 (20 × 5)	14805	3,249	6,505	7,660	7,738
la12 (20 × 5)	12484	3,434	7,262	8,488	8,480
la26 (20 × 10)	20234	7,870	9,191	9,684	9,661
la27 (20 × 10)	20844	7,658	8,889	9,646	9,482
ft20 (20 × 5)	14279	7,031	10,798	12,022	12,117
swv01 (20 × 10)	20688	17,042	19,023	20,176	20,362
swv02 (20 × 10)	21682	14,370	16,291	17,416	17,535
swv06 (20 × 15)	28863	14,632	16,252	17,089	16,956
swv07 (20 × 15)	27385	15,780	17,439	17,883	18,172

Можно видеть, что результаты, полученные при применении модификации метода MOEA+RL с использованием агентов A_1 и A_2 сравнимы с результатами, полученными при применении алгоритма из статьи, и на небольших задачах их превосходят. Наилучшие результаты работы моди-

фикации метода MOEA+RL при использовании агентов, не применяющих результаты обучения при обновлении значений t_i, были получены при использовании двух вспомогательных критериев и агента A_1, выбирающего случайным образом значения t_i.

Также для задачи swv06 был проведен эксперимент, в котором генерировалось по 1000 поколений в каждом запуске. Наилучший результат работы модификации метода MOEA+RL, полученный при применении агента A_1, — 13,856%, превосходит результат, полученный при применении алгоритма из статьи, — 13,922%.

3.4. Выводы по главе 3

Были проведены эксперименты, демонстрирующие работу предложенного метода выбора вспомогательного критерия в эволюционном алгоритме многокритериальной оптимизации при помощи обучения с подкреплением на примере задачи Job Shop.

В результате экспериментов было получено, что при использовании модификации метода MOEA+RL достигаются лучшие результаты, чем при применении метода MOEA+RL.

Были рассмотрены две конфигурации модификации метода MOEA+RL — в одной из них использовались результаты обучения при обновлении значений t_i, а в другой не использовались. Наилучшие результаты для первой конфигурации достигаются при использовании модификации метода MOEA+RL с использованием агента A_3, применяющего Q-обучение с исследованием среды по Больцману. Наилучшие результаты для второй конфигурации получаются при использовании агента A_1, выбирающего случайным образом, сколько раз обучаться на каждом из критериев. В обеих конфигурациях наилучшие результаты были получены при использовании награды R_3, учитывающей приращение значения целевого критерия, вычисленного для лучшей особи.

Можно заметить, что на некоторых больших задачах, результаты,

полученные при использовании агентов, применяющих обучение при обновлении значений t_i, превышают результаты, полученные при применении агентов, не применяющих обучение при обновлении значений t_i.

Результаты, полученные при использовании модификации метода MOEA+RL с применением агента A_1 на ряде задач превосходят результаты, полученные при применении алгоритма из статьи [2].

Заключение

В работе предложен метод адаптивного выбора вспомогательного критерия в эволюционном алгоритме многокритериальной оптимизации при помощи обучения с подкреплением и проверена его применимость на примере задачи Job Shop, в которой в качестве общего времени выполнения работ используется суммарное время выполнения всех работ. На части тестовых данных применение предложенного метода более эффективно, чем использование разработанной ранее эвристики. Предлагаемый подход к выбору вспомогательного критерия отличается новизной, так как впервые для выбора вспомогательного критерия применено обучение с подкреплением.

С целью применения обучения с подкреплением для выбора вспомогательного критерия было разработано множество наград, состояний и агентов. Были проведены эксперименты, исследующие эффективность применения разработанных способов использования обучения с подкреплением для выбора вспомогательного критерия, на примере задачи Job Shop.

Было проведено сравнение предложенных способов применения обучения с подкреплением для выбора вспомогательного критерия. В результате было получено, что наиболее эффективной является модификация метода MOEA+RL, при применении награды, зависящей от значения целевого критерия в лучшей особи, и агента, выбирающего случайным образом, сколько раз обучаться на каждом из критериев. Результаты, полученные при применении такого варианта разработанного метода к ряду задач, превосходят результаты, полученные в статье [2]. Также разработанный метод выбора вспомогательного критерия успешно применялся в генерации тестов для олимпиадных задач [34].

Таким образом, данная работа удовлетворяет всем предъявленным требованиям.

Список литературы

1. *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. No.4. Pp. 323–347.
2. *F. D.*, *Lochtefeld*, *Ciarallo F. W.* Helper-objective optimization strategies for the Job-Shop Scheduling Problem // Appl. Soft Comput. 2011. No.6. Pp. 4161–4174.
3. *Afanasyeva A.*, *Buzdalov M.* Choosing Best Fitness Function with Reinforcement Learning / Proceedings of the Tenth International Conference on Machine Learning and Applications, ICMLA 2011. Vol. 2. Honolulu, HI, USA: IEEE Computer Society, 2011. Pp. 354–357.
4. *Buzdalova A.*, *Buzdalov M.* Increasing Efficiency of Evolutionary Algorithms by Choosing between Auxiliary Fitness Functions with Reinforcement Learning / Proceedings of the 11th International Conference on Machine Learning and Applications, ICMLA 2012. Vol. 1. 2012. Pp. 150–155.
5. *Brucker P.* The Job-Shop Problem: Old and New Challenges / In proceedings of the 3rd Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2007), 28 -31 August 2007, Paris, France. Под ред. P. Baptiste, G. Kendall, A. Munier-Kordon и F. Sourd. Paper. 2007. С. 15—22.
6. *Garey M. R.*, *Johnson D. S.*, *Sethi R.* The Complexity of Flowshop and Jobshop Scheduling // Math. of Op. Res. 1976. No.2. ctr127, pp. 117–129.
7. *Ногин В.* Принятие решений в многокритериальной среде: Количественный подход. Физматлит, 2002. ISBN: 9785922102742.
8. *Скобцов Ю. А.* Основы эволюционных вычислений. Донецк: ДонНТУ, 2008.
9. *Coello C.*, *Lamont G.*, *Van Veldhuisen D.* Evolutionary Algorithms for Solving Multi-objective Problems. Genetic and evolutionary computation series. Springer Science+Business Media, LLC, 2007. ISBN: 9780387367972.
10. *Deb K.*, *Pratap A.*, *Agarwal S.*, *Meyarivan T.* A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II // Transactions on Evolutionary Computation. 2002. №2. С. 182—197.
11. *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. EMO '01. London, UK: Springer-Verlag, 2001. Pp. 269–283.
12. *Crauwels H.*, *Verlinden B.*, *Cattrysse D.*, *Oudheusden D. V.* Sheet-Metal Shop Scheduling Considering Makespan and Flow Time Criteria // The Open Operational Research Journal. 2010. С. 12—24.
13. *Gonzalez T.*, *Sahni S.* Open Shop Scheduling to Minimize Finish Time // J. ACM. 1976. №4. С. 665—679. ISSN: 0004-5411.
14. Job Shop Gantt Chart. http://tinyurl.com/kgmswhp.
15. *Jensen M. T.*, *Dissertation P.*, *Jensen T.* Robust and Flexible Scheduling with Evolutionary Computation. Тех. отч. 2001.
16. *Bierwirth C.* A Generalized Permutation Approach to Job Shop Scheduling with Genetic Algorithms // OR Spektrum. 1995. С. 87—92.
17. *Sun L.*, *Cheng X.*, *Liang Y.* Solving Job Shop Scheduling Problem Using Genetic Algorithm with Penalty Function // IJIIP. 2010. №2. С. 65—77.
18. *Storn R.*, *Price K.* Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces. 1995.
19. *Ponsich A.*, *Tapia M. G. C.*, *Coello C. A. C.* Solving Permutation Problems with Differential Evolution: An Application to the Jobshop Scheduling Problem / Ninth International Conference on Intelligent Systems Design and Applications, ISDA 2009, Pisa, Italy , November 30-December 2, 2009. IEEE Computer Society, 2009. С. 25—30. ISBN: 978-0-7695-3872-3.

20. *Ingimundardottir H.*, *Runarsson T. P.* Supervised learning linear priority dispatch rules for job-shop scheduling / Proceedings of the 5th international conference on Learning and Intelligent Optimization. LION'05. Rome, Italy: Springer-Verlag, 2011. C. 263—277. ISBN: 978-3-642-25565-6.
21. *Kawai T.*, *Fujimoto Y.* An Efficient Combination of Dispatch Rules for Job-shop Scheduling Problem. 2005.
22. *Cheng C.-c.*, *Smith S. F.* Applying Constraint Satisfaction Techniques to job shop scheduling // Annals of Operations Research. 1997. C. 327—357.
23. *Pang W.*, *Goodwin S. D.* Application of CSP Algorithms to Job Shop Scheduling Problems.
24. *Sadeh N.*, *Sycara K.*, *Xiong Y.* Backtracking Techniques for the Job Shop Scheduling Constraint Satisfaction Problem. 1995.
25. *Jain A. S.*, *Meeran S.* Job-Shop Scheduling Using Neural Networks. 1998.
26. *Weckman G. R.*, *Ganduri C. V.*, *Koonce D. A.* A neural network job-shop scheduler // Journal of Intelligent Manufacturing. 2008. C. 191—201.
27. *Anilkumar K. G.*, *Tanprasert T.* A Subjective Scheduler Based on Backpropagation Neural Network for Formulating a Real-life Scheduling Situation //. 2008.
28. *Fnaiech N.*, *Hammami H.*, *Yahyaoui A.* New Hopfield Scheduling Neural Network for joint Job Shop of production and maintenance. 2012.
29. *Николенко С. И.*, *Тулупьев А. Л.* Самообучающиеся системы. М., 2009.
30. *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 (ICML 2006). 2006. Pp. 881–888.
31. *D'Souza R. G. L.*, *Sekaran K. C.*, *Kandasamy A.* Improved NSGA-II Based on a Novel Ranking Scheme // CoRR. 2010.
32. *Sutton R. S.*, *Barto A. G.* Introduction to Reinforcement Learning. 1st. Cambridge, MA, USA: MIT Press, 1998. ISBN: 0262193981.
33. Watchmaker Framework for Evolutionary Computation. http://watchmaker.uncommons.org.
34. *Buzdalov M.*, *Buzdalova A.*, *Petrova I.* Generation of Tests for Programming Challenge Tasks using Multi-Objective Optimization / Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2013. to be published, 06-10 July 2013.