MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 27, NO. 4, 2020
journal homepage: www.mais-journal.ru

THEORY OF COMPUTING

The "One-fifth Rule" with Rollbacks for Self-Adjustment of the Population Size in the $(1 + (\lambda, \lambda))$ Genetic Algorithm

A. O. Bassin[1], M. V. Buzdalov[1], A. A. Shalyto[1]

DOI: 10.18255/1818-1015-2020-4-488-508

[1]ITMO University, 49 Kronverkskiy ave., Saint Petersburg 197101, Russia.

MSC2020: 60G40, 90C56
Research article
Full text in Russian

Received October 22, 2020
After revision November 18, 2020
Accepted December 16, 2020

Self-adjustment of parameters can significantly improve the performance of evolutionary algorithms. A notable example is the $(1 + (\lambda, \lambda))$ genetic algorithm, where adaptation of the population size helps to achieve the linear running time on the OneMax problem. However, on problems which interfere with the assumptions behind the self-adjustment procedure, its usage can lead to the performance degradation. In particular, this is the case with the "one-fifth rule" on problems with weak fitness-distance correlation.

We propose a modification of the "one-fifth rule" in order to have less negative impact on the performance in the cases where the original rule is destructive. Our modification, while still yielding a provable linear runtime on OneMax, shows better results on linear function with random weights, as well as on random satisfiable MAX-3SAT problems.

Keywords: parameter adaptation; $(1 + (\lambda, \lambda))$ GA; linear functions; MAX-3SAT

INFORMATION ABOUT THE AUTHORS

Anton Olegovich Bassin correspondence author	orcid.org/0000-0002-6697-6714. E-mail: anton.bassin@gmail.com PhD student.
Maxim Viktorovich Buzdalov correspondence author	orcid.org/0000-0002-7120-8824. E-mail: mbuzdalov@gmail.com Researcher, PhD.
Anatoly Abramovich Shalyto	orcid.org/0000-0002-2723-2077. E-mail: shalyto@mail.ifmo.ru Chief researcher, Doctor.

Funding: This research was supported by the Russian Scientific Foundation, agreement No.17-71-20178.

For citation: A. O. Bassin, M. V. Buzdalov, and A. A. Shalyto, "The "One-fifth Rule" with Rollbacks for Self-Adjustment of the Population Size in the $(1 + (\lambda, \lambda))$ Genetic Algorithm", *Modeling and analysis of information systems*, vol. 27, no. 4, pp. 488-508, 2020.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 27, № 4, 2020
сайт журнала: www.mais-journal.ru

THEORY OF COMPUTING

Правило «одной пятой» с возвратами для настройки размера популяции в генетическом алгоритме $(1 + (\lambda, \lambda))$

А. О. Басин[1], М. В. Буздалов[1], А. А. Шалыто[1]

DOI: 10.18255/1818-1015-2020-4-488-508

[1]Университет ИТМО, Кронверкский пр., д. 49, Санкт-Петербург, 197101 Россия.

УДК 004.023:004.85
Научная статья
Полный текст на русском языке

Получена 22 октября 2020 г.
После доработки 18 ноября 2020 г.
Принята к публикации 16 декабря 2020 г.

Известно, что настройка параметров может существенно улучшить время работы эволюционных алгоритмов. Ярким примером этого является генетический алгоритм $(1 + (\lambda, \lambda))$, где адаптация размера популяции в процессе работы помогает достичь линейного времени работы на задаче OneMax. Однако если свойства решаемой задачи вступают в конфликт с принципами работы используемого метода настройки параметров, производительность эволюционного алгоритма может существенно ухудшаться. Так, например, происходит при использовании правила «одной пятой» в упомянутом алгоритме при решении задач со слабой корреляцией между приспособленностью и расстоянием до оптимума.

В данной работе предлагается модификация правила «одной пятой», существенно снижающая отрицательные эффекты от его использования при их наличии. Показывается, что данная модификация также достигает линейного времени работы на задаче OneMax, при этом ее использование приводит к улучшению производительности на линейных псевдобулевых функциях со случайными весами, а также на некотором классе задач MAX-3SAT.

Ключевые слова: настройка параметров; $(1 + (\lambda, \lambda))$-ГА; линейные функции; MAX-3SAT

ИНФОРМАЦИЯ ОБ АВТОРАХ

Антон Олегович Басин
автор для корреспонденции
orcid.org/0000-0002-6697-6714. E-mail: anton.bassin@gmail.com
Аспирант.

Максим Викторович Буздалов
автор для корреспонденции
orcid.org/0000-0002-7120-8824. E-mail: mbuzdalov@gmail.com
Научный сотрудник, кандидат технических наук.

Анатолий Абрамович Шалыто
orcid.org/0000-0002-2723-2077. E-mail: shalyto@mail.ifmo.ru
Главный научный сотрудник, доктор технических наук.

Финансирование: Исследование поддержано Российским научным фондом, соглашение №17-71-20178.

Для цитирования: A. O. Bassin, M. V. Buzdalov, and A. A. Shalyto, “The “One-fifth Rule” with Rollbacks for Self-Adjustment of the Population Size in the $(1 + (\lambda, \lambda))$ Genetic Algorithm”, *Modeling and analysis of information systems*, vol. 27, no. 4, pp. 488-508, 2020.

Введение

Эволюционные алгоритмы — это обширный подкласс стохастических методов оптимизации, как правило, взаимодействующих с оптимизируемой функцией как с «черным ящиком». Под эволюционными алгоритмами в узком смысле чаще всего подразумевают генетические алгоритмы [1], эволюционные стратегии [2, 3], эволюционное программирование [4] и генетическое программирование [5]. В широком смысле данный термин также включает в себя методы роевого интеллекта [6—8], методы на основе физических моделей [9, 10], а также методы, разработанные без привлечения каких-либо аналогий с природными процессами или явлениями [11—13]. Данную совокупность методов также называют метаэвристиками [14] или рандомизированными оптимизационными эвристиками. Эволюционные алгоритмы являются разновидностью так называемых мягких вычислений, а в более широком смысле — частью предметной области, все чаще называемой «вычислительный интеллект» (англ. computational intelligence).

Эволюционные алгоритмы применяются для решения множества практических задач, таких как составление расписаний в промышленности [15, 16], оптимизация разводки печатных плат [17—20], построение систем управления [21—27], сегментирование изображений [28], а также приближенного решения других задач комбинаторной [29—32] и вещественной оптимизации [33—35].

Как правило, в работах, посвященных эволюционным алгоритмам, используется набор терминов, заимствованных из теории эволюции. Так, пробные решения называются *особями*, а их составные части — *генами*. Операторы, вносящие небольшие изменения в особи, называются *операторами мутации*, а операторы, создающие новые особи на основе двух и более исходных (*родительских*) особей, называются *операторами скрещивания*, *рекомбинации* или *кроссинговера*. Функция оценки решения, которую требуется оптимизировать, называется *функцией приспособленности*. Многие эволюционные алгоритмы оперируют наборами решений ограниченного размера, которые в этом случае называются *популяциями*, а различные функции, выбирающие особи из популяции или строящие новую популяцию, называются *операторами отбора*.

В настоящей работе особое внимание уделяется настройке параметров в эволюционных алгоритмах. Необходимо отметить, что во многих ранних работах по эволюционным алгоритмам считалось, что их эффективность не зависит значительным образом от конкретных значений параметров (таких как, например, вероятность мутации каждого гена). Данная парадигма мышления объясняется тем, что даже при очевидно неправильных значениях параметров результаты работы таких алгоритмов могут быть вполне удовлетворительными [36, стр. 22]. Следующим этапом были попытки поиска универсальных оптимальных значений параметров [37], далее была осознана необходимость ставить значения параметров в зависимость от размера задачи [38]. В настоящее время общепризнано, что настройка параметров играет одну из ключевых ролей в повышении производительности эволюционных алгоритмов [39]. Исследованиями в этом направлении занимается множество исследовательских групп по всему миру [40—52].

Эволюционные алгоритмы являются прежде всего алгоритмами, предназначенными для применения на практике. Однако для объяснения их принципов работы было развито несколько ветвей теоретических исследований, из которых стоит особо отметить математический анализ времени работы эволюционных алгоритмов, вплотную связанный с именем И. Вегенера [53]. С обширным перечнем известных результатов и их обсуждением можно ознакомиться в обзорах [54, 55]. Среди российских ученых стоит отметить В. Редько [56, 57], Е. Семенкина [58, 59], С. Родзина [60, 61], А. Еремеева [62—65].

Польза анализа времени работы эволюционных алгоритмов не ограничивается констатацией фактов об оценках времени нахождения оптимума той или иной задачи тем или иным алгоритмом. С помощью сравнения фактических оценок времени работы и оценок вычислительной сложности задач оптимизации для алгоритмов различных классов можно ставить вопросы о причинах их

недостаточной эффективности, формулировать гипотезы о наиболее перспективных направлениях исследований и, наконец, разрабатывать новые, более эффективные эволюционные алгоритмы.

Ярким примером эволюционного алгоритма, разработанного исходя из теоретических соображений, является генетический алгоритм $(1 + (\lambda, \lambda))$, предложенный в работе [66] (также см. конференционную версию [67]). Данный алгоритм обладает рядом уникальных свойств. Во-первых, на модельной задаче OneMax, наиболее часто рассматриваемой в теории эволюционных вычислений, данному алгоритму достаточно линейного, относительно размера задачи n, числа запросов к функции приспособленности для нахождения оптимума, в то время как все другие ранее известные эволюционные алгоритмы требуют как минимум $\Omega(n \log n)$ запросов. Во-вторых, известно, что для достижения этого успеха в данном алгоритме требуется выполнять настройку параметров в процессе работы — оптимальный статический выбор параметров дает следующую оценку необходимого числа запросов [68]:

$$\Theta\left(n\sqrt{\frac{\log n \log\log\log n}{\log\log n}}\right),$$

в то время как линейное число запросов достигается при использовании правила «одной пятой» [69] или же стохастического выбора параметра из распределения со степенным законом [70].

Помимо задачи OneMax, данный алгоритм рассматривался также и на других задачах. В работе [66] он, в частности, исследовался экспериментальным способом на линейных псевдобулевых функциях со случайными весами из диапазона $[1; 2]$, а также на функциях из семейства Royal Road. В работе [71] было показано, что данный алгоритм хорошо справляется с некоторым подклассом задач MAX-3SAT (состоящих в нахождении подстановки булевых переменных, максимизирующей число дизъюнктов в заданной КНФ-формуле), а в последовавшей за этим работе [72] это утверждение было доказано (хотя и для несколько иного подкласса этих задач). Однако применение данного алгоритма на практике все еще нельзя назвать широким: среди инженерных применений можно отметить лишь работу [73], посвященную генерации конструкции башен, также можно отметить работу [74] из области поисковой инженерии программного обеспечения. Одним из факторов можно назвать то, что правило «одной пятой», используемое для настройки параметров с целью достижения максимальной производительности, способно существенно ухудшать поведение алгоритма на задачах с малой корреляцией между приспособленностью и расстоянием Хэмминга до оптимума. Отчасти это было показано уже в работе [72], в которой для надлежащей эффективности потребовалось дополнительно ограничить сверху значение параметра. Польза подобного ограничения была также показана в работе [75] для другого класса задач.

В данной работе, являющейся расширенной версией работы [76], мы проводим углубленное изучение этой проблемы. После введения определений, обозначений, а также описания рассматриваемых в работе алгоритмов и модельных задач (раздел 1), в разделе 2 выполняется анализ зависимости эффективности генетического алгоритма $(1 + (\lambda, \lambda))$ от значения параметра λ при различных значениях расстояния Хэмминга до оптимума, из которого делаются выводы о возможных причинах неэффективного поведения данного алгоритма и их связи с настройкой параметров. В разделе 3 излагается модификация правила «одной пятой», используемого в рассматриваемом алгоритме для настройки параметров. Экспериментальные исследования, результаты которых изложены в разделе 4, показывают, что предложенная модификация действительно приводит к повышению эффективности генетического алгоритма $(1 + (\lambda, \lambda))$ на задачах, отличных от OneMax. В то же время в разделе 5 показано, что на задаче OneMax предложенная модификация сохраняет асимптотическую эффективность, и среднее число запросов к функции приспособленности, необходимое для нахождения ее оптимума, остается линейным. В разделе 6 дополнительно обсуждается то, насколько эффективность предложенной модификации близка к теоретически достижимым значениям, и потенциал дальнейшего улучшения данного алгоритма. Итоги работы подводятся в заключении.

1. Определения и обозначения

Будем обозначать множество целых чисел от 1 до n как $[1..n]$. Логарифм по неопределенному основанию, который может встречаться в асимптотических обозначениях, обозначается как log, в то время как натуральный логарифм обозначается как ln. Значение x, округленное вниз, записывается как $\lfloor x \rfloor$. Если x — логическое значение, то $[x]$ — единица, если x истинно, иначе ноль. В псевдокодах алгоритмов операция UniformRandom обозначает случайную выборку элемента из множества, причем каждый элемент выбирается с равной вероятностью.

1.1. Исследуемые модельные задачи

Ранее упомянутая во введении задача OneMax является классической задачей, изучаемой во многих теоретических и экспериментальных работах из области эволюционных вычислений. Экземплярами данной задачи являются функции $\mathrm{OneMax}_{n,z}$, $z \in \{0,1\}^n$, определенные на битовых строках размера n следующим образом:

$$\mathrm{OneMax}_{n,z} : \{0,1\}^n \longrightarrow \mathbb{R}; x \mapsto |\{i \in [1..n] \mid x_i = z_i\}|.$$

Таким образом, функция $\mathrm{OneMax}_{n,z}$ возвращает число битовых позиций, на которых битовые строки x и z совпадают. Множество всех функций $\mathrm{OneMax}_{n,z}$ для фиксированного значения n составляет задачу OneMax размера n.

Функции, составляющие OneMax, могут быть интерпретированы как полиномиальные псевдобулевы функции с максимальной степенью монома, равной единице, или *линейные* псевдобулевы функции. Линейные псевдобулевы функции могут быть представлены в следующем общем виде:

$$\mathrm{Linear}_{n,z,w} : \{0,1\}^n \longrightarrow \mathbb{R}; x \mapsto \sum_{i \in [1..n]} w_i \cdot [x_i = z_i],$$

где w — вектор положительных весов, ассоциированных с битовыми позициями. В данной работе рассмотрены подклассы линейных псевдобулевых функций следующего вида: элементы вектора w являются целочисленными и выбираются равновероятно из множества $[1..W]$ для некоторого значения W. Будем обозначать задачу, состоящую из всех таких функций, как LinInt_W. Отметим, что $\mathrm{OneMax} = \mathrm{LinInt}_1$.

Также в работе будет рассмотрен специальный класс задач выполнимости булевой формулы [77], являющийся подмножеством класса MAX-3SAT. Класс MAX-3SAT состоит из псевдобулевых задач оптимизации, в которых, путем подбора подстановки переменных, требуется максимизировать число выполненных дизъюнктов булевой формулы, записанной в конъюнктивной нормальной форме с тремя литералами в каждом дизъюнкте (т.е. 3-КНФ). Пусть число переменных равно n, а число дизъюнктов равно m. В общем виде данные задачи являются NP-трудными, однако существуют относительно простые классы таких задач [78]. Одним из таких классов являются случайно сгенерированные выполнимые формулы в так называемой модели *planted solution*. Суть данной модели состоит в следующем: вначале генерируется некоторая подстановка x^*, которая будет заведомо являться решением. Далее, каждый из m дизъюнктов генерируется путем равновероятной и независимой выборки каждого из литералов из множества всех литералов с учетом того, что дизъюнкт должен удовлетворяться подстановкой x^*. *Плотностью дизъюнктов* называется значение m/n, равное среднему числу дизъюнктов, приходящееся на одну переменную. Выбор данной модели обусловлен тем, что она ранее изучалась в теории эволюционных вычислений [72, 79, 80], и, в частности, ее свойства достаточно сильно похожи на свойства задачи OneMax в том случае, когда формула является *плотной*, то есть $m \geq n^2$. При рассматриваемой в данной работе комбинации параметров, $m = 4n \ln n$, которая также изучалась в указанных работах, данное сходство также выражено, но не столь сильно, поэтому задача является более сложной, чем OneMax.

1.2. Эволюционные методы локального спуска

В силу относительной простоты связанного с ними анализа, в теоретических исследованиях эволюционных алгоритмов нередко фигурируют алгоритм рандомизированного локального поиска (англ. randomized local search, далее RLS, рисунок 1) и так называемый эволюционный алгоритм (1+1) (далее (1 + 1)-ЭА, рисунок 2). В обоих алгоритмах между итерациями переходит только одна особь, на каждой итерации с помощью оператора мутации порождается одна дочерняя особь, и если ее приспособленность не хуже, чем у родительской, она заменяет родительскую. В алгоритме RLS оператор мутации локальный, в нем инвертируется один случайно выбранный бит, в (1 + 1)-ЭА оператор мутации глобальный — каждый бит инвертируется независимо с вероятностью $1/n$.

Известно, что оба данных алгоритма оптимизируют линейные псевдобулевы функции, включая OneMax, в среднем за $\Theta(n \log n)$ запросов к функции приспособленности. Для рассматриваемого в данной работе класса задач MAX-3SAT для (1 + 1)-ЭА также была доказана оценка $\Theta(n \log n)$ для числа запросов в среднем [80]. Отметим, что для RLS эта оценка также выполняется с поправкой на то, что каждый запуск завершается успешно лишь с вероятностью $1 - e^{-\Omega(n)}$, то есть вероятность неуспеха экспоненциально убывает с ростом числа переменных n.

Require: n: the problem size, $f : \{0,1\}^n \to \mathbb{R}$: the function to maximize
1: $x \leftarrow$ UniformRandom($\{0,1\}^n$)
2: **for** $t \leftarrow 1, 2, 3, \ldots$ **do**
3: $\quad i \leftarrow$ UniformRandom($[1..n]$)
4: $\quad y \leftarrow x$ with i-th bit flipped
5: $\quad$ **if** $f(y) \geq f(x)$ **then**
6: $\quad\quad x \leftarrow y$
7: $\quad$ **end if**
8: **end for**

Fig. 1. Randomized local search

Рис. 1. Рандомизированный случайный поиск

Require: n: the problem size, $f : \{0,1\}^n \to \mathbb{R}$: the function to maximize
1: $x \leftarrow$ UniformRandom($\{0,1\}^n$)
2: **for** $t \leftarrow 1, 2, 3, \ldots$ **do**
3: $\quad y \leftarrow x$
4: $\quad$ **for** $i \in [1..n]$ **do**
5: $\quad\quad$ **with the probability of** $1/n$ **do**
6: $\quad\quad\quad$ flip i-th bit in y
7: $\quad\quad$ **done**
8: $\quad$ **end for**
9: $\quad$ **if** $f(y) \geq f(x)$ **then**
10: $\quad\quad x \leftarrow y$
11: $\quad$ **end if**
12: **end for**

Fig. 2. The (1 + 1) evolutionary algorithm

Рис. 2. Эволюционный алгоритм (1 + 1)

1.3. Генетический алгоритм $(1 + (\lambda, \lambda))$

Генетический алгоритм $(1+(\lambda, \lambda))$ (для краткости будем обозначать его как $(1+(\lambda, \lambda))$-ГА) был впервые предложен в работе [67], журнальная версия которой опубликована двумя годами позже [66].

Его основное отличие от существующих эволюционных алгоритмов, предназначенных для оптимизации псевдобулевых функций, можно кратко сформулировать следующим образом. Во-первых, в операторе мутации используется более высокая, чем обычно, вероятность инвертирования битов, что увеличивает скорость исследования пространства поиска. Во-вторых, в алгоритме используется оператор скрещивания, настроенный таким образом, чтобы нивелировать отрицательные эффекты от такого оператора мутации. Более подробно этот алгоритм будет описан далее.

В канонической версии данного алгоритма имеется, по сути, один параметр λ, а все остальные параметры выводятся из него и из размера задачи n. В работах, в которых данный алгоритм был впервые предложен [66, 67], основной объем теоретического анализа был проделан для фиксированного значения параметра λ. В частности, было доказано, что при некотором выборе параметра λ в зависимости от n матожидание времени работы алгоритма, требуемое для решения задачи OneMax, составляет $O(n\sqrt{\log n})$, что асимптотически меньше, чем у любого другого известного на тот момент эволюционного алгоритма. Данная оценка была несколько улучшена в последующих работах [68, 81], в них показана не только верхняя оценка, но и асимптотически равная ей нижняя.

В то же время, в работах [66, 67] было показано и то, что если выбирать значение λ в зависимости не только от размера задачи n, но и от значения приспособленности лучшей особи (то есть, *динамически*, с изменением данного параметра в процессе работы), то матожидание времени работы такого алгоритма на задаче OneMax составляет $O(n)$. Таким образом, показано, что динамический выбор параметра лучше, чем наилучший возможный статический выбор. Также с помощью экспериментов было показано, что известное из области вещественнозначных эволюционных алгоритмов так называемое правило «одной пятой», примененное для управления параметром λ, способно поддерживать значение этого параметра в непосредственной близости от оптимального значения, не используя знание об оптимальной для данной задачи зависимости. Данная особенность, а вместе с ней и оценка $O(n)$ на матожидание времени работы на OneMax, была впоследствии доказана теоретически [68, 69]. Линейная оценка для матожидания времени работы была также доказана для описанного выше класса задач MAX-3SAT [72], однако для того, чтобы она выполнялась, параметр λ необходимо дополнительно ограничивать некоторым достаточно медленно растущим значением (в работе [72] использовалось значение $\overline{\lambda} = 2\ln(n+1)$).

В данной работе исследуется адаптивная версия $(1 + (\lambda, \lambda))$-ГА, использующая ограничение на значение λ (рисунок 3). Опишем ее подробнее. В данном алгоритме используются следующие операторы, заданные на битовых строках:

- оператор мутации ℓ бит: оператор $\mathrm{Mutate}(x, \ell)$ создает на основе битовой строки $x \in \{0,1\}^n$ новую битовую строку y путем инвертирования ровно ℓ бит, таким образом, что любое множество бит размером ℓ выбирается равновероятно;
- смещенный оператор скрещивания: оператор $\mathrm{Crossover}(x, x', c)$, параметризованный *смещением* $c \in [0,1]$ создает новую битовую строку y на основе двух данных битовых строк x и x' путем независимого выбора для каждой позиции $i \in [1..n]$ значения из второго аргумента ($y_i = x'_i$) с вероятностью c и значения из первого аргумента ($y_i = x_i$) в противном случае.

Алгоритм начинает работу со случайной инициализации начальной популяции, состоящей из одного элемента x. На каждой итерации выполняются следующие действия.

- На *фазе мутации*, на основе родительской особи x генерируется λ потомков путем применения оператора мутации, для которого параметр ℓ выбирается одинаковым для всех потомков и выбирается из биномиального распределения $\mathcal{B}(n, \lambda/n)$. В данной работе, следуя недавно предложенным практико-ориентированным рекомендациям [82], которые уже привели к ряду нетривиальных результатов [83], значение ℓ выбирается из условного распределения, которое может быть описано как $\ell \sim \mathcal{B}(n, \lambda/n) \mid \ell > 0$, то есть всякий раз, когда ℓ выбирается равным нулю, выборка повторяется.

- Фаза мутации оканчивается *промежуточным отбором*, на котором из λ потомков остается потомок, имеющий максимальное значение приспособленности. Данный потомок обозначается как x'. Если потомков с максимальной приспособленностью несколько, один из них выбирается равновероятно.
- На *фазе скрещивания*, на основе родительской особи x и лучшего из потомков из предыдущей фазы x' генерируется λ новых потомков путем применения оператора скрещивания. Смещение в операторе скрещивания выбирается равным $c = 1/\lambda$. Данный выбор был обоснован (по крайней мере, для решения задач, подобных OneMax) в работах [66, 84]. Используемая в данной работе практико-ориентированная модификация состоит в том, что при попытке выбора нуля бит из особи x' получившаяся особь, идентичная родительской, игнорируется и строится заново, а если получающаяся особь идентична особи x', то ее приспособленность повторно не вычисляется.
- После этого осуществляется *элитарный отбор* в следующее поколение: лучший из потомков, сгенерированных на фазе скрещивания, замещает собой родительскую особь x, если его приспособленность не хуже, чем у x.

Require: n: the problem size, $f : \{0,1\}^n \to \mathbb{R}$: the function to maximize

$F \leftarrow$ a constant $\in (1; 2)$ ▷ The adaptation speed
$U \leftarrow 5$ ▷ The value of 5 from the 1/5-th rule
$\lambda \leftarrow 1$ ▷ The initial value of λ
$x \leftarrow \text{UniformRandom}(\{0,1\}^n)$
for $t \leftarrow 1, 2, 3, \ldots$ **do**
  $p \leftarrow \lambda/n$, $c \leftarrow 1/\lambda$, $\lambda' \leftarrow \lfloor \lambda \rfloor$, $\ell \sim \mathcal{B}(n, p)$
  for $i \in [1..\lambda']$ **do** ▷ Phase 1: Mutation
    $x^{(i)} \leftarrow \text{Mutate}(x, \ell)$
  end for
  $x' \leftarrow \text{UniformRandom}(\{x^{(j)} \mid f(x^{(j)}) = \max\{f(x^{(i)})\}\})$
  for $i \in [1..\lambda']$ **do** ▷ Phase 2: Crossover
    $y^{(i)} \leftarrow \text{Crossover}(x, x', c)$
  end for
  $y \leftarrow \text{UniformRandom}(\{y^{(j)} \mid f(y^{(j)}) = \max\{f(y^{(i)})\}\})$
  if $f(y) > f(x)$ **then** ▷ Selection and adaptation
    $x \leftarrow y$, $\lambda \leftarrow \max\{\lambda/F, 1\}$
  else if $f(y) = f(x)$ **then**
    $x \leftarrow y$, $\lambda \leftarrow \min\{\lambda F^{1/(U-1)}, \overline{\lambda}\}$
  else
    $\lambda \leftarrow \min\{\lambda F^{1/(U-1)}, \overline{\lambda}\}$
  end if
end for

Fig. 3. The $(1 + (\lambda, \lambda))$ genetic algorithm with the adaptive choice of $\lambda \le \overline{\lambda}$

Рис. 3. Генетический алгоритм $(1 + (\lambda, \lambda))$ с адаптивным выбором параметра $\lambda \le \overline{\lambda}$

Правило «одной пятой», используемое для адаптивной настройки параметра λ, устроено следующим образом. Если лучшая приспособленность на текущей итерации увеличилась, то значение λ уменьшается в F раз. Если же улучшения не было, то значение λ увеличивается в $F^{1/(U-1)}$ раз, где U — константа (как правило, $U = 5$). В результате, если средняя вероятность улучшения составляет $1/U$, то значение λ поддерживается на постоянном уровне. В дополнение к этому, значение λ ограничено снизу единицей, а сверху — значением $\overline{\lambda}$, которое, как правило, зависит от n и не превосходит n.

2. Исследование ландшафта параметров

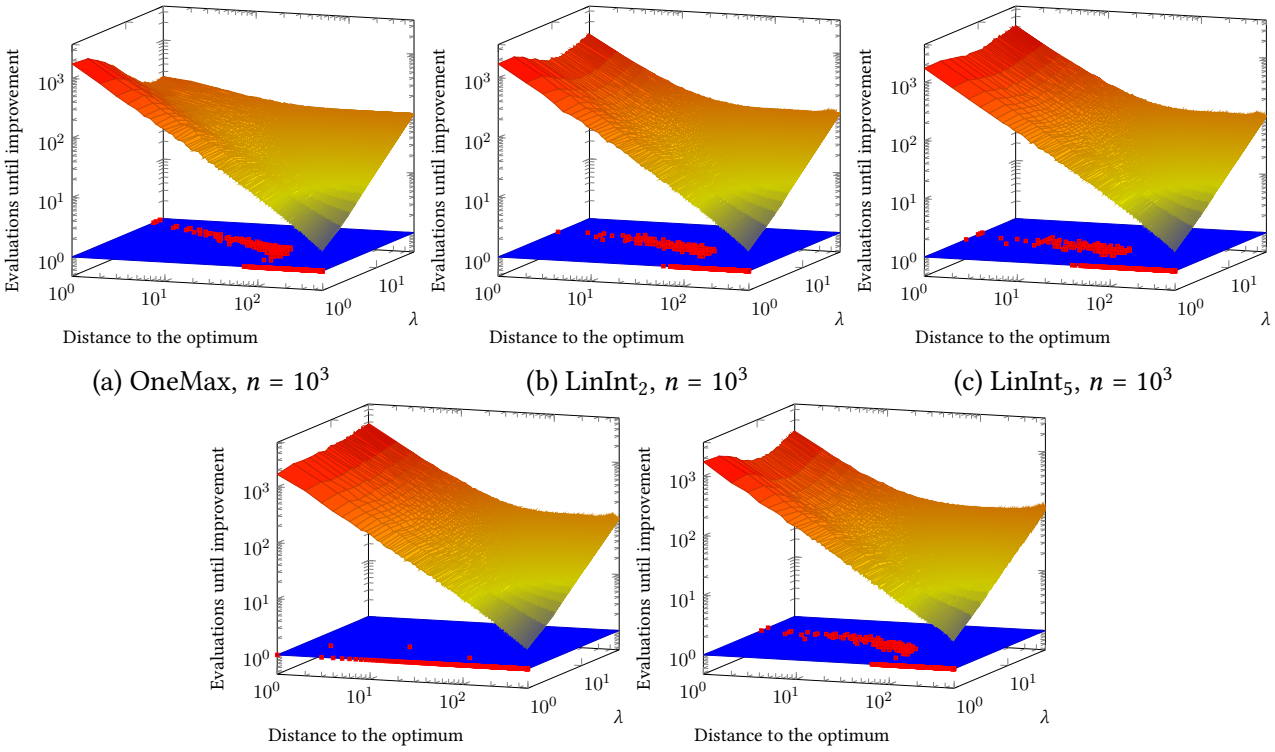
С целью анализа того, какую эффективность имеют различные значения параметра λ в зависимости от расстояния Хэмминга до оптимума, были проведены предварительные экспериментальные исследования, результаты которых представлены на рисунке 4. Данные исследования были проведены для всех рассматриваемых в данной работе задач: OneMax, LinInt_2, LinInt_5, LinInt_n и MAX-3SAT. Для всех задач использовался единый размер задачи $n = 1000$. Для каждой задачи и для всех целочисленных значений $\lambda_0 \in [1..50]$ было выполнено 10^3 запусков генетического алгоритма $(1 + (\lambda, \lambda))$ с фиксированным значением $\lambda = \lambda_0$. В процессе данных запусков для всех расстояний Хэмминга до оптимума $1 \le d \le 500$ и всех исследованных значений λ_0 было вычислено, сколько вычислений функции приспособленности $E(d, \lambda_0)$ было выполнено, когда расстояние между родительской особью и оптимумом было равно d при $\lambda = \lambda_0$, а также число $I(d, \lambda_0)$, описывающее, сколько раз алгоритм покидал данное состояние, переходя в состояние с меньшим расстоянием Хэмминга до оптимума.

На рисунке 4 верхние поверхности сложной формы отражают полученные в результате описанных экспериментов аппроксимации *матожидания числа вычислений функции приспособленности, необходимых для улучшения*, вычисленные как $E(d, \lambda_0)/I(d, \lambda_0)$. Нижние поверхности синего цвета отображают лучшие (т.е. минимальные) среди полученных таким образом значений: для каждого значения d сначала находится такое λ^*, что $E(d, \lambda^*)/I(d, \lambda^*)$ минимально, затем определяются такие достаточно хорошие λ_+, что $E(d, \lambda^+)/I(d, \lambda^+) \le 1.02 \cdot E(d, \lambda^*)/I(d, \lambda^*)$, то есть эффективность которых находится в пределах 2% от оптимального выбора. Такие значения λ+ на указанных рисунках отображаются в виде красных кубиков на синей поверхности.

Внешний вид рисунка 4a, иллюстрирующего результаты на задаче OneMax, соответствует известным фактам относительно поведения $(1 + (\lambda, \lambda))$-ГА на заданной задаче. За исключением некоторых отклонений при больших расстояниях Хэмминга до оптимума, вызванных дискретными значениями параметра λ и использованием практико-ориентированных модификаций, значения λ, близкие к оптимальным, находятся в окрестности прямой в логарифмических осях, проведенной от точки $d = 500, \lambda = 1$ до точки $d = 1, \lambda \approx 30 \ldots 40$. Данные наблюдения соответствуют известному факту, что значения $\lambda \approx \sqrt{n/(n - f(x))}$ оптимальны для задачи OneMax.

Результаты для задач LinInt_2 (рисунок 4b) и LinInt_5 (рисунок 4c) уже показывают несколько иную картину. Пока расстояние Хэмминга до оптимума превышает примерно 50, оптимальные значения ведут себя близко к тому, как это было на OneMax. Однако это поведение изменяется по мере приближения к оптимуму. Визуально общая тенденция поведения все еще напоминает выстраивание вдоль некоторой прямой в логарифмических осях, но наклон этой прямой уже существенно меньше: на единичном расстоянии до оптимума лучшее значение λ для задачи LinInt_2 лишь немного превышает 10, в то время как для задачи LinInt_5 лучшее значение уже меньше десяти. Соответствующие оценки матожиданий числа вычислений функции приспособленности, необходимые для улучшения (то есть для достижения оптимума), примерно равны 10^3 в случае LinInt_2 и достигают $2 \cdot 10^3$ в случае LinInt_5, в то время как для OneMax эта оценка была около 200. Данные наблюдения подсказывают, что уже для линейных функций улучшение наступает недостаточно быстро для того, чтобы правило «одной пятой» поддерживало λ в окрестности оптимальных значений.

В более абстрактных терминах, у задач LinInt_2 и LinInt_5 наблюдается недостаточно хорошая корреляция между функцией приспособленности и расстоянием Хэмминга до оптимума, в то время как у задачи OneMax она по определению идеальна (т.е. равна минус единице). Что же касается задачи LinInt_n, то у нее эта корреляция уже весьма мало отличается от нуля, и, судя по рисунку 4d, сама структура $(1 + (\lambda, \lambda))$-ГА уже недостаточно хороша, чтобы решать эту задачу эффективно: оптимальные значения λ сконцентрированы около единицы, что практически превращает данный алгоритм в алгоритм локального поиска, подобный (1 + 1)-ЭА.



(a) OneMax, $n = 10^3$ (b) LinInt_2, $n = 10^3$ (c) LinInt_5, $n = 10^3$

(d) LinInt_n, $n = 10^3$ (e) MAX-3SAT, $n = 10^3$

Fig. 4. Parameter landscapes for different problems. Red cubes indicate values of $\lambda(d)$ within 2% of the best

Рис. 4. Ландшафты параметров для различных задач. Красные кубики соответствуют значениям $\lambda(d)$ в пределах 2% от лучших

На задаче MAX-3SAT с логарифмической плотностью дизъюнктов поведение $(1 + (\lambda, \lambda))$-ГА представляется более интересным. Из рисунка 4e можно сделать вывод, что поведение данного алгоритма, исходя из величин λ, находится где-то между поведением на задаче OneMax и на LinInt_2. Кроме того, линия, вокруг которой сконцентрированы оптимальные значения λ, представляется скорее как кривая, выпуклая вверх. Исходя из данных результатов, положительный эффект от ограничения значений λ значением $2\ln(n + 1)$, показанный в работе [72], представляется вполне естественным.

3. Предлагаемая модификация

В данной работе предлагается модификация адаптивной настройки параметров по правилу «одной пятой» в $(1 + (\lambda, \lambda))$-ГА (рисунок 5). Основная идея состоит в том, чтобы препятствовать непрерывному росту λ при длительных сериях неудачных итераций, в то же время позволяя повышать λ до произвольно больших значений, если это действительно необходимо. Если итерация была успешной (приспособленность лучшего из потомков превзошла приспособленность родителя), обновленное значение λ также сохраняется в переменной λ_0. Каждая же непрерывная последовательность неудачных итераций разбивается на *отрезки* линейно увеличивающейся длины: каждый следующий отрезок на единицу длиннее предыдущего. В рамках каждого отрезка значение λ растет экспоненциально, как в исходном алгоритме, однако в начале каждого такого отрезка значение λ сбрасывается в значение λ_0. Начальный размер отрезка равен константному значению 10, чтобы соответствовать исходному $(1 + (\lambda, \lambda))$-ГА в случае, когда адаптация работает эффективно. Описанная стратегия ограничивает скорость, с которой растет максимальное значение λ, в то же время сохраняется возможность выполнить итерацию с относительно небольшим значением λ, что может быть полезно в тех ситуациях, когда малые λ выгоднее.

Require: n: the problem size, $f : \{0,1\}^n \to \mathbb{R}$: the function to maximize
1: $F \leftarrow$ a constant $\in (1;2)$ ▷ The adaptation speed
2: $U \leftarrow 5$ ▷ The value of 5 from the 1/5-th rule
3: $\lambda \leftarrow 1$ ▷ The initial value of λ
4: $B \leftarrow 0$ ▷ The number of consecutive bad iterations
5: $\Delta \leftarrow 10$ ▷ The limit of growth for λ
6: $\lambda_0 \leftarrow 1$ ▷ The base value of λ
7: $x \leftarrow \text{UniformRandom}(\{0,1\}^n)$
8: **for** $t \leftarrow 1, 2, 3, \ldots$ **do**
9: $p \leftarrow \lambda/n$, $c \leftarrow 1/\lambda$, $\lambda' \leftarrow \lfloor \lambda \rfloor$, $\ell \sim \mathcal{B}(n, p)$
10: **for** $i \in [1..\lambda']$ **do** ▷ Phase 1: Mutation
11: $x^{(i)} \leftarrow \text{Mutate}(x, \ell)$
12: **end for**
13: $x' \leftarrow \text{UniformRandom}(\{x^{(j)} \mid f(x^{(j)}) = \max\{f(x^{(i)})\}\})$
14: **for** $i \in [1..\lambda']$ **do** ▷ Phase 2: Crossover
15: $y^{(i)} \leftarrow \text{Crossover}(x, x', c)$
16: **end for**
17: $y \leftarrow \text{UniformRandom}(\{y^{(j)} \mid f(y^{(j)}) = \max\{f(y^{(i)})\}\})$
18: **if** $f(y) > f(x)$ **then** ▷ Selection and adaptation
19: $x \leftarrow y$, $\lambda \leftarrow \max\{\lambda/F, 1\}$, $\lambda_0 \leftarrow \lambda$, $B \leftarrow 0$, $\Delta \leftarrow 10$
20: **else**
21: **if** $f(y) = f(x)$ **then**
22: $x \leftarrow y$
23: **end if**
24: $B \leftarrow B + 1$
25: **if** $B = \Delta$ **then**
26: $B \leftarrow 0$, $\Delta \leftarrow \Delta + 1$
27: **end if**
28: $\lambda \leftarrow \min\{\lambda_0 F^{B/(U-1)}, \overline{\lambda}\}$
29: **end if**
30: **end for**

Fig. 5. The $(1 + (\lambda, \lambda))$ genetic algorithm with the modified adaptive choice of $\lambda \le \overline{\lambda}$

Рис. 5. Генетический алгоритм $(1 + (\lambda, \lambda))$ с модифицированным адаптивным выбором параметра $\lambda \le \overline{\lambda}$

Данная модификация также оставляет возможность ограничения значения λ сверху некоторым заданным значением $\overline{\lambda}$. В рамках экспериментального исследования будут рассмотрены два варианта ограничения: $\overline{\lambda} = n$ и $\overline{\lambda} = 2\ln(n+1)$, что позволяет оценить влияние предложенной стратегии в совокупности с влиянием ограничения наибольшего значения λ, и при независимом влиянии новой стратегии.

Рисунок 6 представляет поведение значений размера популяции λ на сериях итераций генетического алгоритма, не находящих лучшее решение. Эта модификация приблизительно возводит в квадрат число итераций, необходимых $(1 + (\lambda, \lambda))$-ГА для достижения некоторого достаточно большого значения λ. Как следствие, в ситуациях, когда λ превышает оптимальное значение для текущего расстояния до оптимума, предлагаемый метод адаптации, в отличие от исходной стратегии настройки параметров, все еще способен регулярно выполнять итерации с использованием значения λ, эффективного в данный момент времени, даже если улучшение приспособленности еще не наступило.

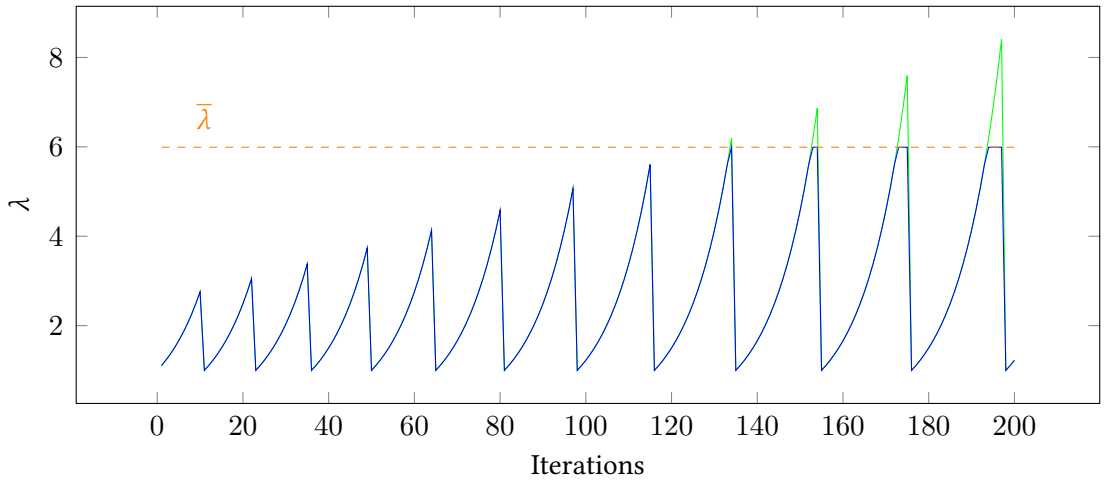


Fig. 6. The behavior of λ in the proposed modification in the case of no improvements

Рис. 6. График поведения λ в предлагаемом методе адаптации при отсутствии улучшений

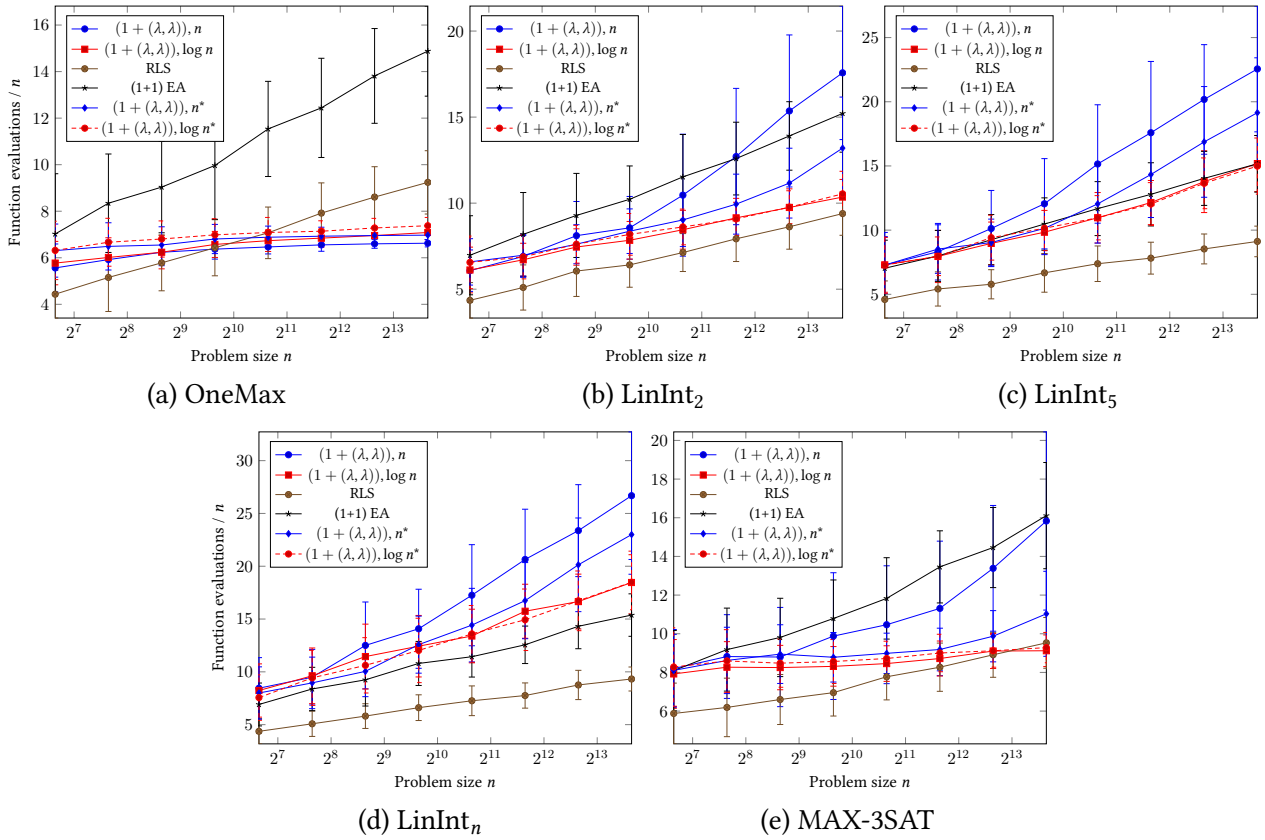
4. Экспериментальные результаты

В данном разделе описывается постановка и результаты экспериментального исследования, нацеленного на сравнение исходной версии $(1 + (\lambda, \lambda))$-ГА и предложенной в данной работе модификации. Каждый из алгоритмов рассматривался в двух вариантах: ограничение сверху на размер популяции составляло $\overline{\lambda} = n$ и $\overline{\lambda} = 2\ln(n + 1)$. Для краткости первый вариант будем называть *неограниченным*, а второй — *с логарифмическим ограничением*. Используется скорость адаптации $F = 1.5$. Для полноты картины в сравнении также участвовали алгоритмы $(1 + 1)$ ЭА и RLS, причем в первом из них, с целью приведения всех алгоритмов в одинаковые условия, стандартная битовая мутация использовалась также в практико-ориентированной модификации: выборка числа инвертируемых бит из биномиального распределения осуществлялась, пока это число не станет ненулевым.

В качестве задач оптимизации использовались те же пять задач, что и в разделе об исследовании ландшафта параметров: OneMax, LinInt_2, LinInt_5, LinInt_n, MAX-3SAT. Рассматривались размеры задач из множества $n \in \{100, 200, 400, 800, 1600, 3200, 6400, 12800\}$. Было выполнено 100 независимых запусков каждого алгоритма на каждой задаче для каждого из указанных размеров n.

Результаты представлены на рисунке 7. Алгоритмы с предложенной модификацией обозначены звездочкой. При решении задачи OneMax все вариации $(1 + (\lambda, \lambda))$-ГА ожидаемо показали хороший результат (рисунок 7a). В частности, варианты с логарифмическим ограничением немного уступают неограниченным вариантам, что также было ожидаемо, поскольку на последних итерациях варианты с логарифмическим ограничением используют субоптимальные значения λ. Предлагаемый алгоритм немного уступает исходному при сравнении соответствующих вариантов, но различие остается небольшим. Предполагается, что с увеличением размера задачи разница стремится приблизительно к 10%. Как будет показано в следующем разделе, асимптотическое поведение модифицированного $(1 + (\lambda, \lambda))$-ГА остается линейным на задаче OneMax.

На задаче LinInt_2 (рисунок 7b) обе версии $(1 + (\lambda, \lambda))$-ГА с логарифмическим ограничением попрежнему показывают достаточно высокую производительность, хотя асимптотика их времени работы, по-видимому, уже отличается от линейной. На рассмотренных размерах задачи их производительность лучше, чем у $(1 + 1)$-ЭА, и немного хуже, чем у RLS, однако тенденции, выраженные наклоном графиков, показывают, что при достаточно больших размерах задач данные алгоритмы превзойдут по производительности и RLS. Неограниченные варианты алгоритма показывают заметно худшие результаты. Исходный неограниченный вариант $(1 + (\lambda, \lambda))$-ГА проявляет тенденцию



(a) OneMax (b) LinInt_2 (c) LinInt_5

(d) LinInt_n (e) MAX-3SAT

Fig. 7. Runtime plots of the algorithms on the benchmark problems. Modified algorithms are marked with a star

Рис. 7. Графики времени работы алгоритмов на модельных задачах. Модифицированные алгоритмы обозначены звездочкой

к замедлению уже при $n = 1600$ и становится хуже, чем (1+1)-ЭА, при $n = 6400$. В свою очередь, модифицированный неограниченный вариант $(1 + (\lambda, \lambda))$-ГА только начинает замедляться при $n = 6400$, но все еще решает задачу быстрее, чем (1 + 1) ЭА, поэтому можно утверждать, что, по крайней мере, константный множитель при асимптотическом выражении для времени выполнения у модифицированной версии лучше, чем у исходной версии $(1 + (\lambda, \lambda))$-ГА.

Аналогичная ситуация наблюдается на рисунке 7c при решении задачи LinInt_5 и на рисунке 7d для LinInt_n. На этих задачах неограниченные варианты $(1 + (\lambda, \lambda))$-ГА демонстрируют худшее время работы по сравнению с (1 + 1)-ЭА, начиная с некоторого значения размера задачи. При этом стоит отметить, что предложенная в данной работе модификация стратегии настройки параметров всегда показывает лучший результат, чем исходная стратегия, в случае неограниченных вариантов, и не отличается от исходной стратегии в случае вариантов с логарифмическим ограничением. Асимптотическое поведение всех рассмотренных алгоритмов, судя по форме графиков, одинаково и составляет $\Theta(n \log n)$, таким образом, влияние модификаций и ограничений на этих задачах ограничивается вкладом в константный множитель при асимптотике и слагаемых меньших порядков.

Поведение алгоритмов на задаче MAX-3SAT (рисунок 7e) выглядит немного иначе, поскольку график неограниченного исходного варианта $(1 + (\lambda, \lambda))$-ГА растет быстрее (возможно, даже асимптотически) по сравнению с неограниченным модифицированным вариантом. Дальнейшее исследование этого поведения представляет интерес, однако вычисление функции приспособленности, в том числе ее пересчитывание при изменении небольшого числа бит, является для MAX-3SAT более трудоемкой задачей. Также из графиков можно сделать нетривиальный вывод, что данная задача для рассматриваемых вариантов $(1 + (\lambda, \lambda))$-ГА является более простой, чем даже LinInt_2.

5. Анализ времени работы предложенной модификации на задаче OneMax

Данный раздел посвящен доказательству того, что $(1 + (\lambda, \lambda))$-ГА с использованием предложенного в настоящей работе метода адаптации параметров сохраняет линейное время работы на задаче OneMax. Приведенные доказательства опираются на теоремы и утверждения из работы [69], посвященной аналогичным доказательствам для исходной версии алгоритма. Отметим, что ряд предположений, на которые опираются теоремы в указанной работе, остаются в силе и в случае модифицированной версии: так, например, используются те же значения параметров, задающих силу мутации ($p = \lambda/n$) и смещенность скрещивания ($c = 1/\lambda$), также коэффициент скорости адаптации F находится в тех же пределах $(1; 2)$.

Теорема 1. *Число запросов к функции приспособленности, выполненных $(1 + (\lambda, \lambda))$-ГА с предложенным в настоящей работе методом настройки размера популяции λ, значением $F \in (1; 2)$ и зависимостями значений параметров от λ в виде $p = \lambda/n$ и $c = 1/\lambda$, составляет $O(n)$ при оптимизации функции* OneMax.

Доказательство. Принцип выполнения доказательства остается неизменным по сравнению с теоремой 5 из [69]. В частности, основным лейтмотивом является доказательство того, что размер популяции λ не отклоняется значительно от оптимального выбора $\lambda^* = \lceil \sqrt{n/n - f(x)} \rceil$, для которого линейное время работы доказано в теореме 2 из [69].

Пусть $x \in \{0, 1\}^n$, $\lambda \geq C_0 \lceil \sqrt{n/n - f(x)} \rceil$ и $q = q(\lambda)$ — вероятность того, что итерация ГА, начинающаяся с особью x, будет успешной. По лемме 6 из [69] следует, что $q > 1/5$ для всех $C_0 > C$. Такая достаточно высокая вероятность дает возможность предположить, что на одной из следующих итераций значение λ уменьшится и устремится к λ^* для случаев, когда λ намного больше λ^*.

Аналогично тому, как это было сделано в [69], в данном доказательстве процесс оптимизации делится на *фазы* двух типов. Первый тип называется *короткой* фазой — в ней неравенство $\lambda \leq C_0\lambda^*$ остается верным на протяжении всех итераций ГА, составляющих эту фазу. Пусть $\tilde{\lambda}$ — начальное значение λ, а t — число итераций в фазе. На каждой итерации ГА производится 2λ вычислений функции приспособленности, следовательно, общее число вычислений функции приспособленности для модифицированного алгоритма на этой фазе выглядит следующим образом.
Если $t \leq \tilde{d}$:

$$\sum_{i=0}^{t-1} 2\tilde{\lambda}F^{i/4} = 2\tilde{\lambda}\frac{F^{t/4} - 1}{F^{1/4} - 1} = O(\lambda^*), \tag{1}$$

иначе:

$$\sum_{d=\tilde{d}}^{k-1}\sum_{i=0}^{d-1} 2\tilde{\lambda}F^{i/4} + \sum_{j=0}^{c-1} 2\tilde{\lambda}F^{j/4} \leq \sum_{i=0}^{t-1} 2\tilde{\lambda}F^{i/4} = 2\tilde{\lambda}\frac{F^{t/4} - 1}{F^{1/4} - 1} = O(\lambda^*) \tag{2}$$

$$\sum_{d=\tilde{d}}^{k-1}\sum_{i=0}^{d-1} 2\tilde{\lambda}F^{i/4} + \sum_{j=0}^{c-1} 2\tilde{\lambda}F^{j/4} = 2\tilde{\lambda}\frac{\sum_{d=\tilde{d}}^{k-1}(F^{d/4} - 1) + F^{c/4} - 1}{F^{1/4} - 1} \geq O(C_0\lambda^*) = O(\lambda^*), \tag{3}$$

где $(k - \tilde{d})$ — число сбросов λ до последнего успешного значения, c — число итераций после последнего сброса λ, $\sum_{i=0}^{k}(\tilde{d} + k) + c = t$ и $\tilde{d} = 10$.

Вторым типом стадии оптимизации является *длительная* фаза. В длительной фазе неравенство $\lambda \leq C_0\lambda^*$ выполняется хотя бы на одной итерации. В свою очередь, каждая *длительная* фаза разделяется на *открывающую* подфазу, которая заканчивается вместе с последней итерацией, удовлетворяющей $\lambda < C_0\lambda^*$, и на *главную* подфазу, которая начинается с размера популяции $C_0\lambda^* \leq \lambda < C_0\lambda^* F^{1/4}$.

Для *открывающей* подфазы число вычислений функции приспособленности можно рассчитать аналогично неравенствам (1), (2) и (3): с поправкой на то, что число итераций равно $t = m$, где

$m = \max\{k \mid \lambda F^{k/4} < C_0\lambda^*\}$, общее число запросов к функции приспособленности составляет $O(\lambda^*)$. Таким же образом можно оценить затраты на *главной* подфазе с учетом того, что начальное значение λ не превосходит $C_0\lambda^* F^{1/4}$.

Рассмотрим случай, который отличается от исходной стратегии настройки параметров по правилу «одной пятой», а именно при $t > \tilde{d}$:

$$C_0\lambda^* F^{1/4}\left(\sum_{d=\tilde{d}}^{k-1}\sum_{i=0}^{d} F^{i/4} + \sum_{j=0}^{c} F^{j/4}\right) \le C_0\lambda^* F^{1/4}\sum_{i=1}^{t} F^{i/4} \le D' C_0\lambda^* F^{(t+1)/4}, \tag{4}$$

для $D' \ge 1/(F^{1/4} - 1)$, где $(k - \tilde{d})$ — число сбросов λ до последнего успешного значения, c — число итераций после последнего сброса λ, $\sum_{i=0}^{k}(\tilde{d} + k) + c = t$ и $\tilde{d} = 10$.

Неравенство (4) является эквивалентом утверждения 2.1 из [69]: $E[T \mid I = t] \le D\lambda^* F^{t/4}$ для достаточно большой константы D, где T — число вычислений функции приспособленности на протяжении *длительной* фазы, а I — число итераций в *главной* подфазе.

Для рассматриваемой модификации остается в силе и утверждение 2.2 из [69], которое определяет вероятность необходимости t итераций: $\Pr[I = t] = e^{-ct}, c > 0$. Данное утверждение следует из того, что, при отсутствии успешных итераций, λ со временем достигает произвольно больших значений, и уменьшение размера популяции возможно в ситуации, когда λ превышает $C_0\lambda^*$.

Общее число вычислений функции приспособленности в ходе *длительной* фазы составляет:

$$\sum_{t=1}^{\infty} E[T \mid I = t]\Pr[I = t] \le D\lambda^* \sum_{t=1}^{\infty} F^{t/4} e^{-ct},$$

что при $F^{1/4} < e^c$ равно $O(\lambda^*)$. □

6. Аппроксимация производительности при оптимальном выборе параметров

Несмотря на то, что предложенная в настоящей работе модифицированная версия правила «одной пятой» позволяет добиться прироста производительности $(1 + (\lambda, \lambda))$-ГА на задачах, более сложных, чем OneMax, вопрос о теоретическом пределе производительности этого алгоритма при оптимальной адаптации параметров, остается нерешенным (в отличие от задачи OneMax, для которой ответ уже известен [66]). С целью исследования этого вопроса было проведено повторное использование экспериментальных результатов из раздела 2: для каждого значения расстояния Хэмминга до оптимума были выбраны значения λ, соответствующие минимальному времени ожидания улучшения.

Далее, для всех алгоритмов, исследованных в разделе 4, а также для $(1 + (\lambda, \lambda))$-ГА, который выбирает значения λ из таблицы, рассчитанной по указанному выше алгоритму для решаемой задачи, на основе расстояния Хэмминга текущего лучшего решения до оптимума, были проведены дополнительные экспериментальные исследования при фиксированном размере задачи n для всех ранее рассмотренных задач: OneMax, LinInt_2, LinInt_5, LinInt_n, MAX-3SAT. Последний алгоритм назовем *псевдооптимальным* $(1 + (\lambda, \lambda))$-ГА.

Результаты представлены в таблице 1, где алгоритмы с предложенной в работе модификацией правила «одной пятой» отмечены звездочками, а псевдооптимальный $(1+(\lambda, \lambda))$-ГА помечен словом «pseudo». Представленные результаты показывают, что все алгоритмы (за исключением RLS на задачах, отличных от OneMax) уступают по производительности псевдооптимальному $(1+(\lambda, \lambda))$-ГА на всех задачах, включая задачу LinInt_n, для которой структура $(1 + (\lambda, \lambda))$-ГА уже, по-видимому, не является перспективной. Таким образом, приведенные наблюдения демонстрируют значительный потенциал методов адаптации параметров и мотивируют их дальнейшее улучшение в применении к $(1 + (\lambda, \lambda))$-ГА.

Table 1. Comparison of existing algorithms and the pseudo-optimal $(1 + (\lambda, \lambda))$ GA, problem size $n = 1000$

Problem	Algorithm	Evaluations
OneMax	RLS	6600 ± 1200
	(1+1) EA	10900 ± 2400
	$(1 + (\lambda, \lambda))$, n	6400 ± 400
	$(1 + (\lambda, \lambda))$, $\log n$	6600 ± 600
	$(1 + (\lambda, \lambda))$, n *	9300 ± 2200
	$(1 + (\lambda, \lambda))$, $\log n$ *	9700 ± 2600
	$(1 + (\lambda, \lambda))$, pseudo	5700 ± 400
LinInt_2	RLS	6700 ± 1100
	(1+1) EA	10900 ± 2000
	$(1 + (\lambda, \lambda))$, n	9200 ± 2300
	$(1 + (\lambda, \lambda))$, $\log n$	8100 ± 1100
	$(1 + (\lambda, \lambda))$, n *	9900 ± 2300
	$(1 + (\lambda, \lambda))$, $\log n$ *	9900 ± 1900
	$(1 + (\lambda, \lambda))$, pseudo	7500 ± 1200
LinInt_5	RLS	6900 ± 1300
	(1+1) EA	10700 ± 2300
	$(1 + (\lambda, \lambda))$, n	12600 ± 3300
	$(1 + (\lambda, \lambda))$, $\log n$	10400 ± 1700
LinInt_{1000}	RLS	6800 ± 1400
	(1+1) EA	11200 ± 2400
	$(1 + (\lambda, \lambda))$, n	15400 ± 4300
	$(1 + (\lambda, \lambda))$, $\log n$	12800 ± 2700
	$(1 + (\lambda, \lambda))$, n *	12400 ± 2600
	$(1 + (\lambda, \lambda))$, $\log n$ *	12300 ± 2400
	$(1 + (\lambda, \lambda))$, pseudo	11000 ± 2300
MAX-3SAT	RLS	7500 ± 1400
	(1+1) EA	11100 ± 2100
	$(1 + (\lambda, \lambda))$, n	9800 ± 3100
	$(1 + (\lambda, \lambda))$, $\log n$	8400 ± 1000
	$(1 + (\lambda, \lambda))$, n *	10300 ± 2000
	$(1 + (\lambda, \lambda))$, $\log n$ *	10600 ± 2100
	$(1 + (\lambda, \lambda))$, pseudo	7800 ± 900
LinInt_5	$(1 + (\lambda, \lambda))$, n *	11100 ± 2100
	$(1 + (\lambda, \lambda))$, $\log n$ *	11000 ± 2000
	$(1 + (\lambda, \lambda))$, pseudo	9500 ± 1600

Таблица 1. Сравнение существующих алгоритмов и псевдооптимального $(1 + (\lambda, \lambda))$-ГА, размерность задач $n = 1000$

7. Заключение

В работе предложена модификация правила «одной пятой», которая используется для настройки параметра λ в генетическом алгоритме $(1 + (\lambda, \lambda))$. Предлагаемый метод направлен на снижение нежелательных эффектов, приводящих к снижению производительности при решении как задач со сниженной корреляцией между приспособленностью и расстоянием до оптимума (линейные функции со случайными целочисленными весами, ограниченными константой), так и задач со слабой или нулевой корреляцией между приспособленностью и расстоянием до оптимума (линейные функции с линейными случайными весами), а также и других задач оптимизации (случайные экземпляры задачи MAX-3SAT в модели *planted solution* и логарифмической плотностью дизъюнктов). Основная идея метода состоит в том, чтобы замедлить рост λ при длительных сериях неудачных итераций. В терминах числа итераций генетического алгоритма $(1 + (\lambda, \lambda))$, скорость роста λ, измеренная по пикам, становится приблизительно равной квадратным корням от исходной скорости.

Несмотря на то, что предложенный метод определенно не является окончательным решением указанной проблемы, в работе показано стабильное улучшение производительности модифицированной версии по сравнению с исходным $(1 + (\lambda, \lambda))$-ГА на всех проблемных функциях. На OneMax предложенная стратегия работает приблизительно на 10% хуже, однако ее время выполнения остается линейным не только на практике, но и в теории, что было доказано по аналогии с соответствующим доказательством для исходной версии.

Также была исследована теоретически возможная производительность $(1 + (\lambda, \lambda))$-ГА в случае, когда выбор λ близок к оптимально возможному. Для этого для исследованных задач были предподсчитаны оптимальные значения λ в зависимости от расстояния Хэмминга текущего лучшего решения до оптимума, которые затем были использованы в экспериментах. Несмотря на то, что полученный вариант алгоритма не является практически значимым, данная часть исследования демонстрирует, что теоретически возможности структуры $(1+(\lambda, \lambda))$-ГА для решения сложных задач шире, чем представлялось ранее.

References

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan, 1975, 211 pp.

[2] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart: Fromman-Holzboorg Verlag, 1973.

[3] H.-P. Schwefel, «Binäre Optimierung durch somatische Mutation», Technical University of Berlin and Medical University of Hannover, Tech. Rep., May 1975.

[4] L. J. Fogel, «Autonomous automata», *Industrial Research*, vol. 4, pp. 14–19, 1962.

[5] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. Cambridge, MA, USA: MIT Press, 1992.

[6] R. C. Eberhart and J. Kennedy, «A new optimizer using particle swarm theory», in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[7] M. Dorigo and L. M. Gambardella, «Ant colony system: A cooperative learning approach to the traveling salesman problem», *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[8] R. Poli, J. Kennedy, and T. Blackwell, «Particle swarm optimization, An overview», *Swarm Intelligence*, vol. 1, pp. 33–57, 2007.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, «Optimization by simulated annealing», *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[10] J. M. Bishop, «Stochastic searching networks», in *Proceedings of 1st IEEE Conference on Artificial Neural Networks*, 1989, pp. 329–331.

[11] R. Storn and K. Price, «Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces», *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[12] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz, «Linkage problem, distribution estimation, and bayesian networks», *Evolutionary Computation*, vol. 8, no. 3, pp. 311–340, 2000.

[13] B. Doerr and M. S. Krejca, «Significance-based estimation-of-distribution algorithms», in *Proceedings of Genetic and Evolutionary Computation Conference*, 2018, pp. 1483–1490.

[14] S. Luke, *Essentials of Metaheuristics*. Lulu, 2009, 253 pp.

[15] A. Orlov, V. V. Kureichik, and A. Glushchenko, «Hybrid genetic algorithm for cutting stock and packaging problems», in *Proceedings of East-West Design & Test Symposium*, IEEE, 2016.

[16] L. A. Gladkov, N. V. Gladkova, and S. A. Gromov, «Hybrid models of solving optimization tasks on the basis of integrating evolutionary design and multiagent technologies», in *Proceedings of Computer Science Online Conference*, ser. Advances in Intelligent Systems and Computing 985, 2019, pp. 381–391.

[17] E. V. Kuliev, A. N. Dukkardt, V. V. Kureychik, and A. A. Legebokov, «Neighborhood research approach in swarm intelligence for solving the optimization problems», in *Proceedings of East-West Design & Test Symposium*, IEEE, 2014.

[18] E. V. Kuliev, V. V. Kureichik, and I. O. Kursitys, «Decision making in VLSI components placement problem based on grey wolf optimization», in *Proceedings of East-West Design & Test Symposium*, IEEE, 2019.

[19] V. V. Kureichik and D. V. Zaruba, «Combined approach to place electronic computing equipment circuit elements», in *Proceedings of East-West Design & Test Symposium*, IEEE, 2015.

[20] L. A. Gladkov, N. V. Gladkova, and S. Leiba, «Electronic computing equipment schemes elements placement based on hybrid intelligence approach», in *Proceedings of Computer Science Online Conference*, ser. Advances in Intelligent Systems and Computing 348, 2015, pp. 35–44.

[21] M. Semenkina, «Parallel version of self-configuring genetic algorithm application in spacecraft control system design», in *Proceedings of Genetic and Evolutionary Computation Conference Companion*, 2013, pp. 1751–1752.

[22] D. Chivilikhin, V. Ulyantsev, and A. Shalyto, «Modified ant colony algorithm for constructing finite state machines from execution scenarios and temporal formulas», *Automation and Remote Control*, vol. 77, no. 3, pp. 473–484, 2016.

[23] C. M. Fonseca and P. J. Fleming, «Nonlinear system identification with multiobjective genetic algorithm», in *Proceedings of the World Congress of the International Federation of Automatic Control*, 1996, pp. 187–192.

[24] I. Buzhinsky, V. Ulyantsev, D. Chivilikhin, and A. Shalyto, «Inducing finite state machines from training samples using ant colony optimization», *Journal of Computer and Systems Sciences International*, vol. 53, no. 2, pp. 256–266, 2014.

[25] D. Chivilikhin, V. Ulyantsev, and A. Shalyto, «Extended finite-state machine inference with parallel ant colony based algorithms», in *International Student Workshop on Bioinspired Optimization Methods and Their Applications*, 2014, pp. 117–126.

[26] D. Chivilikhin, V. Ulyantsev, and A. Shalyto, «Combining exact and metaheuristic techniques for learning extended finite state machines from test scenarios and temporal properties», in *Proceedings of International Conference on Machine Learning and Applications*, 2014, pp. 350–355.

[27] I. Buzhinsky, V. Ulyantsev, F. Tsarev, and A. Shalyto, «Search-based construction of finite-state machines with real-valued actions: New representation model», in *Proceedings of Genetic and Evolutionary Computation Conference Companion*, 2013, pp. 199–200.

[28] S. El-Khatib, Y. Skobtsov, and S. Rodzin, «Improved particle swarm medical image segmentation algorithm for decision making», in *Intelligent Distributed Computing XIII*, ser. Studies in Computational Intelligence 868, 2019, pp. 437–442.

[29] A. V. Eremeev and Y. V. Kovalenko, «Genetic algorithm with optimal recombination for the asymmetric travelling salesman problem», in *LSSC 2017: Large-Scale Scientific Computing*, ser. Lecture Notes in Computer Science 10665, 2017, pp. 341–349.

[30] A. V. Eremeev and Y. V. Kovalenko, «A memetic algorithm with optimal recombination for the asymmetric travelling salesman problem», *Memetic Computing*, vol. 12, no. 1, pp. 23–36, 2020.

[31] D. S. Sanches, D. Whitley, and R. Tinós, «Improving an exact solver for the traveling salesman problem using partition crossover», in *Proceedings of Genetic and Evolutionary Computation Conference*, 2017, pp. 337–344.

[32] L. D. Whitley, F. Chicano, and B. W. Goldman, «Gray box optimization for Mk landscapes (NK landscapes and MAX-kSAT)», *Evolutionary Computation*, vol. 24, no. 3, pp. 491–519, 2016.

[33] A. Glotić and A. Zamuda, «Short-term combined economic and emission hydrothermal optimization by surrogate differential evolution», *Applied Energy*, vol. 141, pp. 42–56, 2015.

[34] T. Liao, P. J. Egbelu, and P. Chang, «Two hybrid differential evolution algorithms for optimal inbound and outbound truck sequencing in cross docking operations», *Applied Soft Computing*, vol. 12, no. 11, pp. 3683–3697, 2012.

[35] V. Feoktistov, S. Pietravalle, and N. Heslot, «Optimal experimental design of field trials using differential evolution», in *Proceedings of Congress on Evolutionary Computation*, 2017, pp. 1690–1696.

[36] T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds., *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics Publishing, 2000, 339 pp.

[37] J. J. Grefenstette, «Optimization of control parameters for genetic algorithms», *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, pp. 122–128, 1986.

[38] H. Mühlenbein, «How genetic algorithms really work: Mutation and hillclimbing», in *Parallel Problem Solving from Nature – PPSN II*, Elsevier, 1992, pp. 15–26.

[39] Á. E. Eiben, R. Hinterding, and Z. Michalewicz, «Parameter control in evolutionary algorithms», *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, 1999.

[40] V. Stanovov, S. Akhmedova, E. Semenkin, and M. Semenkina, «Generalized Lehmer mean for success history based adaptive differential evolution», in *Proceedings of International Joint Conference on Computational Intelligence*, 2019, pp. 93–100.

[41] V. Stanovov, S. Akhmedova, and E. Semenkin, «LSHADE algorithm with rank-based selective pressure strategy for solving CEC 2017 benchmark problems», in *Proceedings of Congress on Evolutionary Computation*, IEEE, 2018.

[42] M. Semenkina and E. Semenkin, «Memetic self-configuring genetic programming for solving machine learning problems», in *Proceedings of International Congress on Advanced Applied Informatics*, 2015, pp. 599–604.

[43] M. Semenkina, S. Akhmedova, C. Brester, and E. Semenkin, «Choice of spacecraft control contour variant with self-configuring stochastic algorithms of multi-criteria optimization», in *Proceedings of International Conference on Informatics in Control, Automation and Robotics*, 2016, pp. 281–286.

[44] V. Stanovov, E. Semenkin, and O. Semenkina, «Self-configuring hybrid evolutionary algorithm for fuzzy imbalanced classification with adaptive instance selection», *Journal of Artificial Intelligence and Soft Computing Research*, vol. 6, no. 3, pp. 173–188, 2016.

[45] N. Hansen and A. Ostermeier, «Completely derandomized self-adaptation in evolution strategies», *Evolutionary Computation*, vol. 9, pp. 159–195, 2001.

[46] R. Tanabe and A. Fukunaga, «Success-history based parameter adaptation for differential evolution», in *Proceedings of IEEE Congress on Evolutionary Computation*, 2013, pp. 71–78.

[47] R. Senkerik, M. Pluhacek, T. Kadavy, and A. Zamuda, «Distance based parameter adaptation for success-history based differential evolution», *Swarm and Evolutionary Computation*, vol. 50, 2019. DOI: 10.1016/j.swevo.2018.10.013.

[48] N. Dang and C. Doerr, «Hyper-parameter tuning for the $(1 + (\lambda, \lambda))$ GA», in *Proceedings of Genetic and Evolutionary Computation Conference*, 2019, pp. 889–897.

[49] E. Ridge and D. Kudenko, «Tuning an algorithm using design of experiments», in *Experimental Methods for the Analysis of Optimization Algorithms*, T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, Eds. Springer, 2010, pp. 265–286.

[50] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, T. Stützle, and M. Birattari, «The irace package: Iterated racing for automatic algorithm configuration», *Operations Research Perspectives*, vol. 3, pp. 43–58, 2016.

[51] F. Hutter, H. H. Hoos, and K. Leyton-Brown, «Sequential model-based optimization for general algorithm configuration», in *Proceedings of Learning and Intelligent Optimization*, Springer, 2011, pp. 507–523.

[52] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle, «ParamILS: An automatic algorithm configuration framework», *Journal of Artificial Intelligence Research*, vol. 36, pp. 267–306, 2009.

[53] I. Wegener, «Methods for the analysis of evolutionary algorithms on pseudo-Boolean functions», in *Evolutionary Optimization*, ser. International Series in Operations Research & Management Science 48, 2003, pp. 349–369.

[54] A. Auger and B. Doerr, *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2011.

[55] B. Doerr and F. Neumann, Eds., *Theory of Evolutionary Computation—Recent Developments in Discrete Optimization*. Springer, 2020.

[56] V. G. Red'ko and Y. Tsoy, «Estimation of the evolution speed for the quasispecies model: Arbitrary alphabet case», in *Artificial Intelligence and Soft Computing – ICAISC 2006*, ser. Lecture Notes in Computer Science 4029, 2006, pp. 460–469.

[57] V. G. Red'ko, O. P. Mosalov, and D. V. Prokhorov, «Investigation of evolving populations of adaptive agents», in *Artificial Neural Networks: Biological Inspirations – ICANN 2005*, ser. Lecture Notes in Computer Science 3696, 2005, pp. 337–342.

[58] A. Antamoshkin, E. Antamoshkin, and E. Semenkin, «Local search efficiency when optimizing unimodal pseudoboolean functions», *Informatica*, vol. 9, no. 3, 1998.

[59] A. N. Antamoshkin, V. N. Saraev, and E. Semenkin, «Optimization of unimodal monotone pseudoboolean functions», *Kybernetika*, vol. 26, no. 5, pp. 432–442, 1990.

[60] S. Rodzin and L. Rodzina, «Theory of bionic optimization and its application to evolutionary synthesis of digital devices», in *Proceedings of East-West Design & Test Symposium*, IEEE, 2014.

[61] S. El-Khatib, Y. Skobtsov, S. Rodzin, and S. Potryasaev, «Theoretical and experimental evaluation of PSO-K-Means algorithm for MRI images segmentation using drift theorem», in *Proceedings of Computer Science Online Conference*, ser. Advances in Intelligent Systems and Computing 985, 2019, pp. 316–323.

[62] P. A. Borisovsky and A. V. Eremeev, «Comparing evolutionary algorithms to the (1+1)-EA», *Theoretical Computer Science*, vol. 403, no. 1, pp. 33–41, 2008. DOI: 10.1016/j.tcs.2008.03.008.

[63] D. Corus, D.-C. Dang, A. V. Eremeev, and P. K. Lehre, «Level-based analysis of genetic algorithms and other search processes», *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 5, pp. 707–719, 2018.

[64] A. V. Eremeev, «On non-elitist evolutionary algorithms optimizing fitness functions with a plateau», in *Mathematical Optimization Theory and Operations Research*, ser. Lecture Notes in Computer Science 12095, 2020, pp. 329–342.

[65] A. V. Eremeev, «On proportions of fit individuals in population of mutation-based evolutionary algorithm with tournament selection», *Evolutionary Computation*, vol. 26, no. 2, pp. 269–297, 2018.

[66] B. Doerr, C. Doerr, and F. Ebel, «From black-box complexity to designing new genetic algorithms», *Theoretical Computer Science*, vol. 567, pp. 87–104, 2015.

[67] B. Doerr, C. Doerr, and F. Ebel, «Lessons from the black-box: Fast crossover-based genetic algorithms», in *Proceedings of Genetic and Evolutionary Computation Conference*, 2013, pp. 781–788.

[68] B. Doerr and C. Doerr, «Optimal static and self-adjusting parameter choices for the $(1 + (\lambda, \lambda))$ genetic algorithm», *Algorithmica*, vol. 80, no. 5, pp. 1658–1709, 2018.

[69] B. Doerr and C. Doerr, «Optimal parameter choices through self-adjustment: Applying the 1/5-th rule in discrete settings», in *Proceedings of Genetic and Evolutionary Computation Conference*, 2015, pp. 1335–1342.

[70] D. Antipov, M. Buzdalov, and B. Doerr, «Fast mutation in crossover-based algorithms», in *Proceedings of Genetic and Evolutionary Computation Conference*, ACM, 2020, pp. 1268–1276.

[71] B. Goldman and W. Punch, «Parameter-less population pyramid», in *Proceedings of Genetic and Evolutionary Computation Conference*, 2014, pp. 785–792.

[72] M. Buzdalov and B. Doerr, «Runtime analysis of the $(1 + (\lambda, \lambda))$ genetic algorithm on random satisfiable 3-CNF formulas», in *Proceedings of Genetic and Evolutionary Computation Conference*, 2017, pp. 1343–1350.

[73] A. Gandomi and B. Goldman, «Parameter-less population pyramid for large-scale tower optimization», *Expert Systems with Applications*, vol. 96, pp. 175–184, 2018.

[74] V. Mironovich and M. Buzdalov, «Hard test generation for maximum flow algorithms with the fast crossover-based evolutionary algorithm», in *Proceedings of Genetic and Evolutionary Computation Conference Companion*, 2015, pp. 1229–1232.

[75] M. A. Hevia Fajardo and D. Sudholt, «On the choice of the parameter control mechanism in the $(1 + (\lambda, \lambda))$ genetic algorithm», in *Proceedings of Genetic and Evolutionary Computation Conference*, 2020, pp. 832–840.

[76] A. Bassin and M. Buzdalov, «The 1/5-th rule with rollbacks: On self-adjustment of the population size in the $(1 + (\lambda, \lambda))$ GA», in *Proceedings of Genetic and Evolutionary Computation Conference Companion*, 2019, pp. 277–278. [Online]. Available: https://arxiv.org/abs/1904.07284.

[77] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979, 338 pp.

[78] D. Mitchell, B. Selman, and H. Levesque, «Hard and easy distributions of SAT problems», in *Proceedings of AAAI Conference on Artificial Intelligence*, 1992, pp. 459–465.

[79] A. M. Sutton and F. Neumann, «Runtime analysis of evolutionary algorithms on randomly constructed high-density satisfiable 3-CNF formulas», in *Proceedings of Parallel Problem Solving from Nature*, ser. Lecture Notes in Computer Science 8672, 2014, pp. 942–951.

[80] B. Doerr, F. Neumann, and A. M. Sutton, «Improved runtime bounds for the (1+1) EA on random 3-CNF formulas based on fitness-distance correlation», in *Proceedings of Genetic and Evolutionary Computation Conference*, 2015, pp. 1415–1422.

[81] B. Doerr and C. Doerr, «A tight runtime analysis of the $(1 + (\lambda, \lambda))$ genetic algorithm on OneMax», in *Proceedings of Genetic and Evolutionary Computation Conference*, 2015, pp. 1423–1430.

[82] E. Carvalho Pinto and C. Doerr. (2018). Towards a more practice-aware runtime analysis of evolutionary algorithms, [Online]. Available: https://arxiv.org/abs/1812.00493.

[83] E. C. Pinto and C. Doerr, «A simple proof for the usefulness of crossover in black-box optimization», in *Parallel Problem Solving from Nature – PPSN XV, Vol. 2*, ser. Lecture Notes in Computer Science 11102, 2018, pp. 29–41.

[84] B. Doerr, «Optimal parameter settings for the $(1 + (\lambda, \lambda))$ genetic algorithm», in *Proceedings of Genetic and Evolutionary Computation Conference*, Full version available at http://arxiv.org/abs/1604.01088, 2016, pp. 1107–1114.