Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ТЕОРЕТИЧЕСКОЕ ИССЛЕДОВАНИЕ ЭВОЛЮЦИОННОГО АЛГОРИТМА (1+1) НА ДИНАМИЧЕСКИХ ПСЕВДОБУЛЕВЫХ ЗАДАЧАХ ОПТИМИЗАЦИИ

Автор: Торопин Константин Игоревич _____

Направление подготовки: 01.03.02 Прикладная математика и информатика

Квалификация: Бакалавр

Руководитель ВКР: Буздалов М.В., к.т.н. _____

Санкт-Петербург, 2021 г.

Обучающийся Торопин Константин Игоревич
Группа М3437 Факультет ИТиП

Направленность (профиль), специализация
Математические модели и алгоритмы в разработке программного обеспечения

ВКР принята «____» _____ 20___ г.

Оригинальность ВКР _____%

ВКР выполнена с оценкой _____

Дата защиты «____» _____ 20___ г.

Секретарь ГЭК Павлова О.Н. _____

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В практике применения эволюционных алгоритмов нередко встречаются динамические задачи оптимизации, в рамках которых оптимизируемая функция изменяется со временем. В теоретическом анализе времени работы эволюционных алгоритмов не так много работ уделяется этим задачам. И в этих работах центральным ставится вопрос может ли быть достигнут оптимум между изменениями функции приспособленности, а то, как в промежутке меняется значение приспособленности, не особо рассматривается, за вычетом приспособленностей, крайне близких к оптимуму. В частности не решается вопрос что должно происходить с различными способами адаптации параметров, которые применяются для повышения эффективности алгоритмов. Поэтому задача анализа алгоритма (1 + 1) на динамических псевдобулевых задачах оптимизации является актуальной.

ГЛАВА 1. ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Эволюционные алгоритмы могут применяться для поиска экстремума функции.

Однако в теории эволюционных алгоритмов существует проблема, которая заключается в том, что функция в процессе поиска экстремума может изменятся. Примером такой задачи может быть поиск оптимального маршрута в городе - искомый маршрут в зависимости от времени может отличаться. Поэтому возникает потребность проанализировать алгоритмы учитывая динамическое преобразование функции. Для начала был взят алгоритм (1 + 1) на задаче $OneMax$.

Данный алгоритм был выбран потому что это наиболее понятная и простая задача для анализа и часто является отправной точкой для исследования.

OneMax - задача поиска минимума функции количества несовпадающих бит с некоторым вектором f_{max}, длина которого n.

Алгоритм (1 + 1) решает её следующим образом - берётся случайный вектор длины n и далее итеративно мутируется - каждый бит независимо меняется с некоторой вероятностью $\frac{r}{n}$ (r рассматривается в данном исследовании $o(1)$ по отношению к n). Если количество совпадающих бит увеличивается, берётся мутируемый. Добавим следующую динамику для задачи - пусть каждые T итераций алгоритма вектор f_{max} меняет один случайный бит. Существуют несколько статей на похожую тему [3] и [7] где также исследуется вопрос скорости работы динамических алгоритмов, однако в этих статьях нет исследований самого поведения алгоритма. В рамках исследования были выявлены две ситуации - когда T слишком мала, то алгоритм "застревает"на некотором значении, расстояние до f_{max} колеблются на уровне некоторого $\cdot\, n$. Значение c будем называть точкой стабилизации или плато. То есть в этой ситуации с некоторого момента примерное отношение бит которые не совпадают с вектором f_{max} ко всем битам будет равна c. Вторая ситуация - T оказывается достаточно большой, что алгоритм доходит до нужного оптимума, однако медленнее чем если бы динамики не было. Помимо этого был рассмотрена ситуация когда алгоритм не заканчивается на достижении оптимума (что вполне можно увидеть на реальных задачах). Тогда будет разумным вопрос - какая вероятность, что при изменении одного бита алгоритм успеет найти новый оптимум до следующего изменения.

По первому случаю был установлен факт, что значение c не зависит от параметра T (если $n > 1/c$), это значит что если мы фиксируем какое-то значение T, то насколько сильно мы бы не увеличивали n, значение c будет оставаться тем же. Были найдены некоторые оценки на значение c, а также теоретически обоснован факт почему алгоритм фиксируется и почти не уходит от этого значения.

Что касается второго случая - асимптотика EA-алгоритмов обычно оценивается с помощью некоторых теорем которые несут общее название drift-analysis. Были взяты теоремы Additive Drift и Variable Drift и рассмотрены с точки зрения того, что математическое ожидание изменения функции на некоторых шагах отличается от обычного. Если рассматривать данную конкретную задачу, то в ней каждые K шагов математическое ожидание будет отличаться на некоторое значение $\leqslant 1$. Эти теоремы в дальнейшим могут помочь исследовать скорость нахождения первого оптимума и в более сложных алгоритмах в которых будет введена динамика. С помощью видоизмененных теорем была доказано, что если T больше некоторого значения, то асимптотика алгоритма не изменится, то есть останется $O(n \cdot log(n))$

Экспериментальные данные показывают, что плато находится на тех значениях, которые предсказывает теория.

ГЛАВА 2. ТЕОРЕТИЧЕСКОЕ ИССЛЕДОВАНИЕ

2.1. Постановка задачи

Для начала рассмотрим базовую задачу $OneMax$ в том, чтобы найти заданный вектор, когда у нас есть только ”черный ящик”который говорит сколько бит совпадает с оптимальным вектором. Суть базового алгоритма $(1+1)EA$ в том берём случайный вектор, и каждую итерацию изменяем каждый бит независимо с некоторой вероятностью $\frac{r}{n}$, где n - длина вектора, а r - некоторая константа Более наглядно показано с помощью псевдокода

Листинг 1 – OneMax (1 + 1)EA

Require: x_{max} - required vector; n — dimensionality of vector; r - constant
1: Sample first individual x randomly
2: Calculate fitness of x: $f_{best} \leftarrow samebits(x, x_{max})$
3: **while** $x \neq x_{max}$ **do**
4: $y \leftarrow mutate(x, \frac{r}{n})$
5: $f_y \leftarrow samebits(y)$
6: **if** $f_y > f_{best}$ **then**
7: $x \leftarrow y$
8: $f_{best} \leftarrow samebits(y, x_{max})$
9: **end if**
10: **end while**

Листинг 2 – OneMax (1 + 1)EA with dynamic

Require: x^{opt} - required vector; r - constant; T - constant
11: Sample first individual x randomly
12: Calculate fitness of x: $f_{best} \leftarrow samebits(x, x_{max})$
13: $i \leftarrow 1$
14: **while** $x \neq x^{opt}$ **do**
15: **if** $i \mod T = 0$ **then**
16: $x_{max} \leftarrow change(x^{opt})$
17: **end if**
18: $y \leftarrow mutate(x, \frac{r}{n})$
19: $f_y \leftarrow samebits(y)$
20: **if** $f_y > f_{best}$ **then**
21: $x \leftarrow y$
22: $f_{best} \leftarrow samebits(y, x^{opt})$
23: **end if**
24: **end while**

Суть алгоритмов $(1+1)EA$ и задачи $OneMax$ детально расписана в [10] и [12].

Далее введём для данного алгоритма динамику схожую с той, которую вводят в [5] [6]. Раз в T итераций алгоритм случайным образом изменяет в необходимом векторе ровно один бит.

В ходе практического анализа алгоритма была обнаружена закономерность, что алгоритм при некоторых значениях параметра T алгоритм "останавливается"в некоторых точках.

Для дальнейшего анализа нам потребуются обозначения:

$(x_t^{opt})_{t \in N}$ - оптимальный вектор, который алгоритм пытается найти на итерации t.

x_0^{opt} считаем равным изначальному искомому вектору

Множество всевозможных пар x_t, x_t^{opt} будет задавать пространство исходов Ω , вероятность конкретного исхода как и вся σ-алгебра естественным образов выводится из фильтрации, которую пораждает случайный процеесс.

Таким образом построено вероятностное пространство $(\Omega, \mathcal{F}, \mathbb{P})$.

$samebits(a, b)$ - количество совпадающих бит у двух векторов.

$X_t := samebits(x_t, x_t^{opt})$, последовательность случайных величин обозначающих количество совпадающих бит текущего вектора на итерации t.

Y_t - случайная величина, обозначающая прирост величины X_t за счёт изменения вектора x_t. Формально

$$Y_t := \begin{cases} samebits(x_t, x_{t-1}^{opt}) - X_{t-1}, & \text{if } t > 0 \\ 0, & \text{if } t0 \end{cases}.$$

Z_t - случайная величина, обозначающая уменьшение величины X_t за счёт изменения x_t^{opt}. Формально

$$Z_t := \begin{cases} samebits(x_t, x_t^{opt}) - samebits(x_t, x_{t-1}^{opt}), & \text{if } t > 0 \\ 0, & \text{if } t0 \end{cases}$$

Очевидно что в рамках данной задачи $Z_t = \{0, -1\}$ для всех t.

Так же можно увидеть что $X_t = X_{t-1} + Y_t + Z_t$.

Небольшая ремарка: в дальнейших рассуждениях будут использоваться формулы вида $E[X_{t+1} = a | x_t = I]$ то есть в них будет опускаться фильтра-

ция, хотя она, конечно, неявно подразумевается. Введём обозначения $p = \frac{r}{n}$, $q = 1 - p$.

2.2. Поиск плато

В ходе практического исследования алгоритма был установлен факт, что для некоторых T значение алгоритм останавливается на некотором значении и не может далеко уйти за эти рамки. Назовём значение этой точки "плато"и будем искать его в следующей точке: пусть I- количество совпадающий бит, n - длина массива где математическое ожидание "подняться вверх"то есть на сколько больше бит будет будет равняться оптимальному будет равно значению $\frac{1}{T \cdot n}$. Физический смысл этого можно понять так: пусть мы рассматриваем задачу, когда динамика может происходить каждый раз, но с вероятностью 1/T. Данная задача также исследуется в [2] [11]. Вероятность того, что динамика ухудшит ответ будет $\frac{I}{T \cdot n}$ и математическое ожидание будет процесса динамики будет равно этому же $\frac{I}{T \cdot n}$. Так что суммарное математическое ожидание будет равно 0.

На самом деле задача стоит другим образом и честная оценка $E[X_{t+T} - X_t, x_t = I]$ через T итераций будет ниже чем $T \cdot E[Y_t | X_t = I]$ потому что она убывает от роста I, но заглядывая в будущее, окажется что этого приближения достаточно, и уйти от данной точки на линейную длину алгоритму будет экспоненциально сложно.

Пусть в данный момент у алгоритма величина $X_t = I$, то есть количество бит текущего, которые совпадают с текущим оптимальным вектором. Введём обозначения - для конкретной I

$$Bin(m, n) \equiv C_m^n \cdot p^m \cdot q^{n-m} = C_m^n \cdot p^m \cdot q^{n-m}$$
$$A(i) \equiv Bin(i, I)$$
$$Q(i) \equiv Bin(i, n - I)$$
$$= \frac{I}{n}$$

Значение c можно рассматривать как процент правильных бит. Значение $A(i)$ как вероятность что среди I бит произойдёт ровно i изменений. Значение $Q(i)$ как вероятность что среди $n - I$ бит произойдёт ровно i изменений.

Так как вероятность изменения значения вектора ровно на i равна вероятности того что среди ещё не найденных бит изменится ровно k бит, а в найденных изменится $k - i$ для всех возможных k.

$$P(Y_t = i|X_t = I) = \sum_{j=1}^{\min(I-i,n-I-i)} A(i+j)Q(i)$$

Давайте рассматривать только такие I что

$$c > 0.5$$
$$r = O(1), n \to \infty$$

Очевидно что рассматривать алгоритм в тех случаях, когда $c < 0.5$ не имеет смысла, так как уже начальное приближение задачи при инициализации вектора будет иметь математическое ожидание количества совпадающих бит уже $\frac{n}{2}$. По поводу второго приближения, все изучения задачи $OneMax$ всегда используют $r = o(1)$, а ещё чаще равному 1.

Заметим что

$$P(Y_t = i|X_t = I) = \sum_{j=1}^{I-i} A(i+j)Q(i)$$

Тогда значение $E[Y_t|X_t = I]$ можно расписать в виде: $E[Y_t|X_t = I] = \sum_{i=1}^{I} i \cdot \sum_{j=1}^{I-i} A(i+j)Q(i) = \sum_{i=1}^{I} Q(i) \sum_{j=i}^{I} j \cdot A(j-i) =$

$$\begin{aligned} = Q(0)A(1) + \; & 2 \cdot Q(0)A(2) \; + 3 \cdot Q(0)A(3) \; + \ldots \; + I \cdot Q(0)A(I) + \\ & Q(1)A(2) \; + 2 \cdot Q(1)A(3) \; + \ldots \; + (I-1) \cdot Q(1)A(I) + \\ & \qquad\qquad + \ldots \; + A(I)Q(n-I) \end{aligned}$$

Так как $\sum_{j=1}^{I} j \cdot A(j)$ это определение матожидание случайной величины A, которая равна pI.

Заметим что первая строка равна

$\tau^* \equiv Q(0) \cdot \sum_{j=1}^{I} j \cdot A(j) = Q(0) \cdot pI = q^{n-I} \cdot c \cdot r$. Это будет нашей нижней оценкой на $E[Y_t|X_t = I]$

Далее оценим сумму остальных строк. Оценим сверху для каждой строки во сколько изменяется элемент в строке относительно предыдущего, взяв отношение элемента в строке $j+1$ и поделим на элемент в строке j:

Для строки $i \geqslant 2$

$$\frac{(j+1)\cdot Q(i)\cdot A(i+j+1)}{j\cdot Q(i)\cdot A(i+j)}$$
$$= \frac{j+1}{j}\cdot\frac{I!}{(I-(i+j+1))!\cdot(i+j+1)!}\cdot p^{I-(i+j+1)}q^{i+j+1}\cdot$$
$$\cdot\frac{(I-(i+j))!\cdot(i+j)!}{I!}\cdot\frac{1}{p^{I-(i+j)}q^{i+j}}$$
$$= \frac{j+1}{j}\cdot\frac{I-i-j}{i+j+1}\cdot(\frac{q}{p}=\frac{r\cdot n}{n\cdot(n-r)}=\frac{r}{n-r})$$
$$= \frac{j+1}{j\cdot(i+j+1)}\cdot\frac{c\cdot(n-r)+c\cdot r-i-j)}{n-r}\cdot r\leqslant$$
$$\leqslant\frac{1}{2}\cdot\frac{c\cdot(n-r)}{n-r}\cdot r+o(1)\approx\frac{c\cdot r}{2}$$

Все попытки анализа данной модели для r равной произвольной константе от n не увенчались успехом, скорее всего для анализа необходимо использовать модель по аналогии с той, которая делалась в [1]

Если принять гипотезу что $r<2$, тогда каждая строка i:

$$a_i\equiv\sum_{j=1}^{I-i}(j\cdot Q(i)\cdot A(i+j))\leqslant$$
$$\leqslant\sum_{j=1}^{inf}(Q(i)\cdot A(i+1)\cdot(\frac{c\cdot r}{2})^j)$$
$$=Q(i)\cdot A(i+1)\cdot\frac{1}{1-\frac{c\cdot r}{2}}$$

Аналогично сравним первые элементы рядов i и $i+1$:

$$\frac{Q(i+1)\cdot A(i+2)}{Q(i)\cdot A(i+1)}=\frac{(n-I-i)\cdot(I-i-1)}{(i+1)\cdot i}\cdot(\frac{r}{n-r})^2$$
$$\leqslant\frac{(1-c)\cdot(n-1)2\cdot c\cdot r^2}{(n-r)^2}=c\cdot(1-c)\cdot r^2$$

Получившийся коэффициент сходится благодаря тому что $c \cdot (1-c) < 1/4$, а $r < 2$ как мы приняли.

Аналогично получается геометрическая прогрессия, которая сходится, теперь посчитаем сумму всех рядов:

$$Q(1)A(2) \cdot \frac{1}{1 - c \cdot (1-c) \cdot r^2} \cdot \frac{1}{1 - c \cdot r/2} =$$
$$= \frac{(n-I) \cdot I \cdot (I-1)}{2} \cdot p^3 \cdot q^{n-3} \cdot constants \leqslant$$
$$\leqslant c^2 \cdot r^3 \cdot q^{n-3} = (q^{n-I} \cdot c \cdot r) \cdot (q^{I-3} \cdot c \cdot r^2) = \tau^* \cdot (q^{I-3} \cdot c \cdot r^2)$$

Значение $constants < 1$ его можно использовать, чтобы брать более качественные оценки на c_{lower} о котором будет говорится позже.

Оценки τ^*:

Введем значение $\tau \equiv \exp(-r + r \cdot c) \cdot c \cdot r$

$$\tau^* = q^{n-I} \cdot c \cdot r = (1 - r/n)^n \cdot (1-c) \cdot c \cdot r < \exp(-r + r \cdot c) \cdot c \cdot r = \tau$$
$$\tau^* = (1 - r/n)^{n-r} \cdot (1-c) \cdot c \cdot r \cdot (1 - r/n)^r \cdot (1-c)$$
$$> \exp(-r + r \cdot c) \cdot c \cdot r \cdot (1 - r^2/2n)^{1-c} > \exp(-r + r \cdot c) \cdot c \cdot r \cdot (1 - r^2/2n) >$$
$$> \exp(-r + r \cdot c) \cdot c \cdot r \cdot (1 - o(1)) = \tau \cdot (1 - o(1))$$

Заметим также:

$$q^{I-3} = (1 - r/n)^{c-3} < \exp(-r \cdot c) \cdot (1 - r/n)^{-3} < \exp(-r \cdot c) + o(1)$$

Обозначим $k \equiv \exp(-r \cdot c) \cdot c \cdot r^2$

Тогда сумма всей таблицы оценивается сверху как:

$$\tau^* \cdot (1 + q^{I-3} \cdot c \cdot r^2) < \exp(-r + r \cdot c) \cdot c \cdot r \cdot (1 + \exp(-r \cdot c) + o(1)) \cdot c \cdot r^2) <$$
$$< \tau \cdot (1 + k + o(1)$$

Из чего сделаем вывод что $E[Y_t | X_t = I] \in (\tau \cdot (1 - o(1)), \tau \cdot (1 + k + o(1))$

Рассмотрим $E[Z_t, X_t = I]$.

Для начала давайте рассматривать задачу в которой каждый шаг с вероятностью $1/T$ меняется один бит в векторе x^{opt}. Так как вероятность что случайно выбранный бит будет из тех, которые уже достигли оптимума равняется $1 - I/n = (1 - c)$.

Тогда $E[Z_t, X_t = I] = (1 - c)/T$.

Заметим что $E[Z_t, X_t = I]$возрастает с убыванием c, когда как $E[Y_t|X_t = I]$ убывает.

Второй факт доказывается тем что при $p > 1/2$ вероятность изменить бит $> \frac{1}{2}$, а количество необходимых бит убывает.

Тогда рассмотрим для фиксированного T такие два значения c_{upper} и c_{lower} для которых мы можем быть уверены что:

$$E[_\Delta(X_t)|X_t = n \cdot c_{upper}] \geqslant 0$$
$$E[_\Delta(X_t)|X_t = n \cdot c_{lower}] \leqslant 0$$

Грубо говоря вне этого промежутка алгоритм будет ”тянуть”его назад.

$$E[_\Delta(X_t)|X_t = n \cdot c] = E[(Y_t)|X_t = n \cdot c] - \frac{(1 - c)}{K}$$
$$E[Y_t|X_t = I] \in (\tau - o(1), \tau \cdot (1 + k + o(1))$$
$$0 \leqslant [_\Delta(X_t)|X_t = n \cdot c] < \tau - o(1) - \frac{(1 - c)}{K}$$
$$K \cdot (\tau - o(1)) > (1 - c)$$
$$K > \frac{(1 - c)}{\tau \cdot (1 - o(1))}$$

Рассмотрим $g_{lower}(c) = \frac{(1-c)}{\tau} = \frac{(1-c)}{\exp(-r+r\cdot c)\cdot c\cdot r}$

Тогда $c_{lower} = g^{-1}_{lower}(K) + \epsilon$ где $\epsilon \to 0$

Аналогично $g_{upper}(c) = \frac{(1-c)}{\tau\cdot(1+k)} = \frac{(1-c)}{\exp(-r+r\cdot c)\cdot c\cdot r\cdot(1+\exp(-r\cdot c)\cdot c\cdot r^2)}$
$c_{upper} = g^{-1}_{upper}(K)$

Заметим что величины c_{upper} и c_{lower} не зависят от n, а значит мы можем сделать вывод что плато будет начинаться в одинаковых значениях вне зависимости от величины n.

Далее оценим величину:

$$P(Y_t \geqslant j | X_t = I)$$

Установим следующий факт:

Теорема 1.

$$r, \eta > 0, I > n/2 :$$
$$P(Y_t \geqslant j | X_t = I) \leqslant \frac{r}{(1+\eta)^j}$$

Доказательство.

$$P(j)P(Y = j | X_t = I) = \sum_{i=0}^{I-j} A(i+j)Q(i)$$
$$\frac{A(i)}{A(i+1)} = \frac{n!}{i! \cdot (n-i)!} \cdot p^i \cdot q^{n-i} \cdot \frac{(i+1)! \cdot (n-i-1)!}{n!} \cdot \frac{1}{p^{i+1} \cdot q^{n-i-1}} =$$
$$= \frac{i+1}{n-i} \cdot \frac{n-r}{r} P(Y_t \geqslant j | X_t = I)$$

Тогда используя Negative-drift из [9] теорему мы можем доказать что уйти на любое значение от плато займёт экспоненциальное время.

Теорема 2. Пусть $X_{t_0} = c : c \in (c_{lower}, c_{upper})$ Тогда найдётся такое большое n, что:

$$1. \forall k : k > 1/2, k \cdot n < c_{lower} \cdot n - 1,$$
$$T_a = \min t > t_0 | X_t \leqslant k \cdot n \exists l :$$
$$P[T_a \leqslant e^{l \cdot n}] < e^{-l \cdot n}$$
$$2. \forall k : k \leqslant 1, k \cdot n > c_{upper} \cdot n + 1,$$
$$T_a = \min t > t_0 | X_t \geqslant k \cdot n \exists l :$$
$$P[T_a \leqslant e^{l \cdot n}] < e^{-l \cdot n}$$

Доказательство. Из факта выше следует, что для всех $I > 1/2$
$P(Y_t \geqslant j | X_t = I) \leqslant \frac{r}{(1+\eta)}$

Также заметим что Z_t не может превышать по модулю 1.

$$P(Z_t \geqslant 1) \leqslant \frac{1}{K}$$
$$P(Z_t \geqslant j > 1) = 0$$

Таким образом:

$P(|_\Delta X| \geqslant j | X_t = I) \leqslant \frac{\max(r, 1/K)}{(1+\eta)}$

1. Рассмотрим второй случай, так как $E[_\Delta X_t | X_t = I]$убывает и $E[_\Delta X | X_t = c_{upper} \cdot n] \equiv \delta < 0$

И по монотонности для всех

$$I \in (c_{upper} \cdot n, k \cdot n)$$
$$E[_\Delta X | X_t = I] < \delta < 0$$

Таким образом все условия Negative-Drift теоремы выполнены.

2. Аналогично, только рассмотрим вместо X_t случайную величину $W_t = -X_t$, тогда

$$E[_\Delta W | W_t = c_{lower} \cdot n] \equiv \delta < 0$$
$$I \in (k \cdot n, c_{lower} \cdot n)$$
$$E[_\Delta W | W_t = I] \equiv \delta < 0$$

.

2.3. Оценки скорости нахождения первого оптимума

Для оценки скорости эволюционных алгоритмов часто используют так называемые Drift-теоремы например в [4] подробнее теоремы можно изучить в [9] и [8]. Рассмотрим классические Drift-теоремы с точки зрения динамической задачи задачи.

Теорема 3. Пусть $(X_t)_{t \in N}$, последовательность случайных величин с $S \in [0, \infty)$ состояниями

$X \in Z$; $s_{\min} = \inf(S \backslash \{0\})$; Пусть $T := \inf t > 0 | X_t = 0$; Для любого $t \geqslant 0$:

$$\delta_t(s) = E[X_t - X_{t+1} | t\%K \neq K-1, X_t = s]$$
$$\Delta_t(s) = E[X_t - X_{t+1} | t\%K = K-1, X_t = s]$$

Если существуют такие константы ϵ и δ, что:

$$\epsilon \cdot K > \delta$$
$$\delta_t(s) > \epsilon$$
$$\Delta_t(s) > \epsilon - \delta$$

Тогда $E[T] \leqslant \frac{E[X_0]}{\delta - \epsilon/K}$

Доказательство. Теорему можно свести к классической теореме об Аддитивном дрифте. [8] Рассмотрим случайную величину $Y_i = X_{i \cdot K}$

$$E[Y_0] = E[X_0]$$
$$Y_{i+1} = X_i \cdot k + \sum_{t=i \cdot K}^{i \cdot (K+1) - 1} (X_{t+1} - X_t) + \delta$$
$$E[Y_i - Y_{i+1}] \geqslant K \cdot \epsilon - \delta > 0$$

Последнее неравенство из условия.
Тогда если I - первое такое i, что $Y_i = 0$, $E[I] \leqslant \frac{E[X_0]}{K \cdot \epsilon - \delta}$.
А значит $E[T] \leqslant E[I] \cdot K \leqslant \frac{E[X_0]}{\epsilon - \frac{\delta}{K}}$

Теперь рассмотрим вариативный дрифт:

Теорема 4. Пусть $(X_t)_{t \in N}$, последовательность с $S \in [0, \infty)$ состояниями, число K

$s_{\min} = \inf(S \backslash \{0\})$

Пусть $T := \inf t > 0 | X_t = 0$.

И для любого $t \geqslant 0$:

$$\delta_t(s) = E[X_t - X_{t+1} | t\%K \neq K-1, X_t = s]$$
$$\Delta_t(s) = E[X_t - X_{t+1} | t\%K = K-1, X_t = s]$$

Если существуют такая константа ϵ и монотонно возрастающая функция $h : R^+ \to R^+$ что:

$$\delta_t(s) > h(s)$$
$$\Delta_t(s) > h(s) - \epsilon$$
$$K > \frac{\epsilon}{h(s_{\min})}$$

Тогда:

$$E[T] \leqslant \frac{1}{1 - \frac{\epsilon}{K \cdot h(s_{\min})}} \cdot (\frac{s_{\min}}{h(s_{\min})} + E[\int_{s_{\min}}^{X_0} \frac{1}{h(\sigma)} d\sigma])$$

Доказательство. Воспользуемся доказательством классического вариативного дрифта. [9] Если рассматривать функцию $g(s) := \begin{cases} \frac{s_{\min}}{h(s_{\min})} + \int_{s_{\min}}^{s} \frac{1}{h(\sigma)} d\sigma, & \text{if } s \geqslant s_{\min} \\ \frac{s_{\min}}{h(s_{\min})}, & \text{if } 0 \leqslant s \leqslant s_{\min} \end{cases}$ То по свойствам функции из теоремы получается что если брать $Y_t = g(X_t)$:
Для $t \mod K \neq K - 1$:

$$E[Y_t - Y_{t+1} | Y_t = g(s)] = E[g(X_t) - g(X_{t+1}) | g(X_t) = g(s)]$$
$$\geqslant E[\frac{X_t - X_{t+1}}{h(X_t)}] = \frac{\delta_t(s)}{h(s)} \geqslant 1$$

Для $t \mod K = K - 1$:

$$E[Y_t - Y_{t+1} | Y_t = g(s)] \geqslant \frac{\Delta_t(s)}{h(s)} \geqslant 1 - \frac{\epsilon}{h(s_{\min})}$$

Тогда можем применяя (3) выше со следующими условиями:

$$\epsilon_{new} \equiv 1$$

$$\delta_{new} \equiv \frac{\epsilon}{h(s_{\min})}$$
$$K_{new} \equiv K$$
$$K > \frac{\epsilon}{h(s_{\min})} > \delta_{new}$$
$$\epsilon_{new} \cdot K_{new} > \delta_{new}$$

Значит:

$$E[T] \leqslant \frac{1}{1 - \frac{\epsilon}{K \cdot h(s_{\min})}} \cdot (\frac{s_{\min}}{h(s_{\min})} + E[\int_{s_{\min}}^{X_0} \frac{1}{h(\sigma)} d\sigma])$$

Применим данную теорему к рассматриваемой задаче:

Теорема 5. Если Алгоритм $OneMax(K)$ имеет $K = O(n)$, $K > \frac{n \cdot \exp(r)}{r} + O(n)$, то $E[T] = O(n \cdot \log(n))$

Доказательство.

$$\delta_t(s) \geqslant A(1)Q(0) = i \cdot p \cdot q^{n-1} = i \cdot r/n \cdot (1 - r/n)^{n-1}$$
$$\geqslant \frac{i}{n} \cdot r \cdot \exp(-r) - \phi(n)$$
$$\phi(n) = o(1/n)$$
$$\Delta_t(s) = \delta_t(s) - \frac{n-s}{n} \geqslant \delta_t(s) - 1$$
$$\epsilon = 1$$
$$s_{min} = 1$$
$$h(s) = r \cdot \frac{s}{n} \cdot \exp(-r) - \phi(n)$$
$$h(s_{min}) = \frac{r \cdot \exp(-r)}{n} - \phi(n)$$
$$K > \frac{1}{h(s_{\min})}$$
$$\exists c \in (0,1) : \frac{1}{1 - \frac{1}{K \cdot h(s_{\min})}} < c$$

Значит мы можем применить теорему (4):

$$E[T] \leqslant \frac{1}{1 - \frac{\epsilon}{K \cdot h(s_{\min})}} \cdot (\frac{s_{\min}}{h(s_{\min})} + E[\int_{s_{\min}}^{X_0} \frac{1}{h(\sigma)} d\sigma]) =$$
$$= c \cdot (O(n \cdot \log(n))) = O(n \cdot \log(n))$$

2.4. Добавочные леммы

Леммы, используемые в доказательстве:

Лемма 6. $(1 - r/n)^n < e^{-r}$

Лемма 7. $(1 - r/n)^{n-r} > e^{-r}$

Лемма 8. $(1 - r/n)^r > 1 - 2r/n$

Доказательство. Пусть a - такое что $a = 2^k, a \leqslant r, 2 \cdot a > r$, где $k \in \mathbb{N}$

$$(1 - r/n)^r \geqslant (1 - r/n)^a = (1 - 2r/n + r^2/n^2)^{a/2} > (1 - 2r/n)^{a/2}$$
$$> (1 - 4r/n)^{a/4} > 1 - a \cdot r/n > 1 - 2 \cdot r^2/n$$

Лемма 9. $(1 - r/n)^n > e^{-r} \cdot (1 - o(1/n))$

Доказательство.

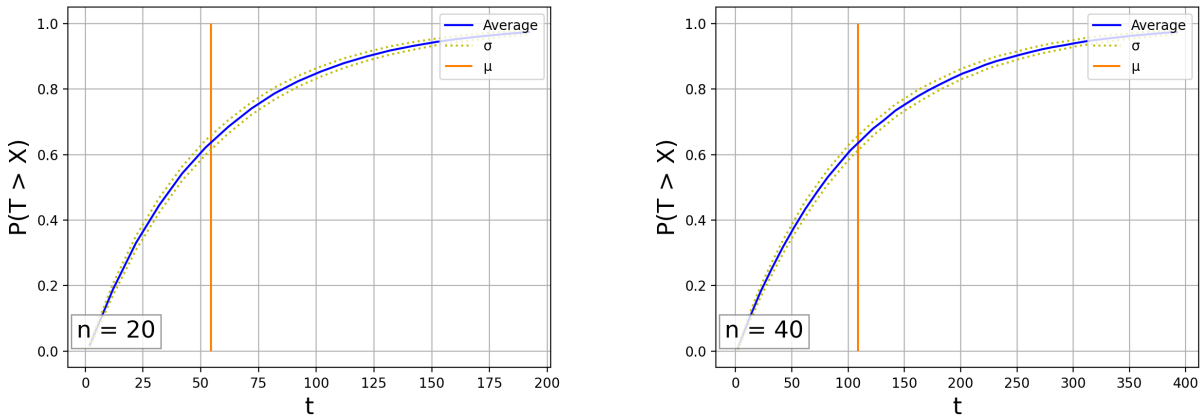
$$(1 - r/n)^n > e^{-r} \cdot (1 - r/n)^r > e^{-r} \cdot (1 - 2 \cdot r^2/n) > e^{-r} - o(r^2/n) = e^{-r} - o(1)$$

ГЛАВА 3. ПРОВЕРКА ГИПОТЕЗ НА РЕАЛЬНЫХ ДАННЫХ

Для того чтобы посмотреть насколько быстро будет возвращаться алгоритм после первого достижения оптимума были проведён следующий эксперимент:

Алгоритм запускается несколько раз на одних и тех же n и T. И рассматривается количество раз, когда алгоритм успевает найти новый оптимум.

μ - матожидание первого срабатывания.



n = 20 n 200

Рисунок 1 – Как часто алгоритм найти успевает новый оптимум

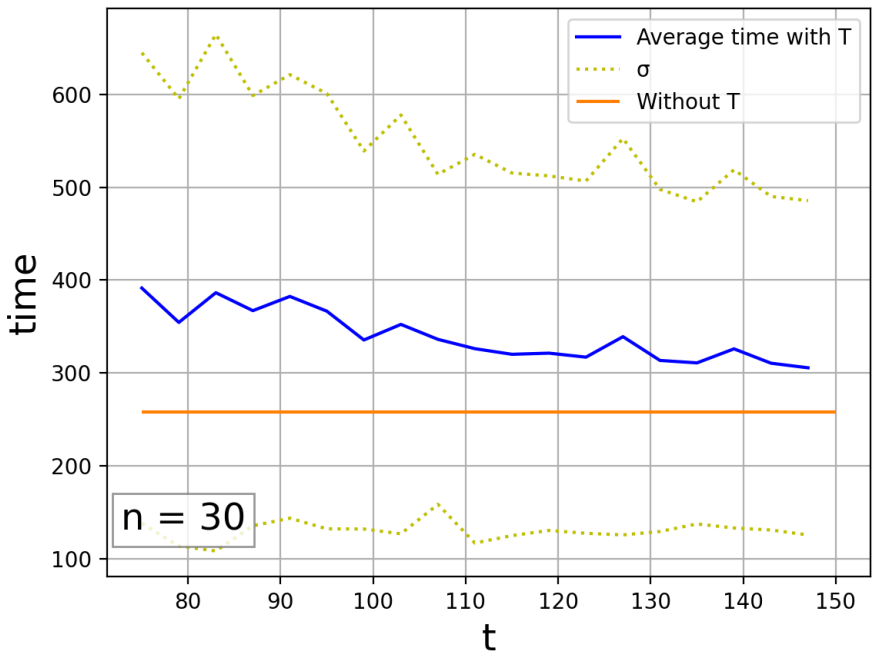


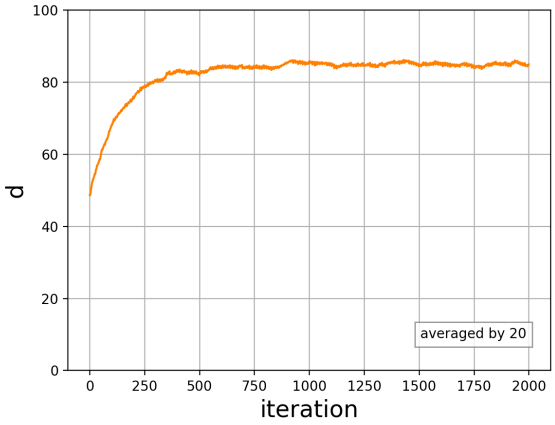
Рисунок 2 – Время работы алгоритма с n = 30

Для оценки скорости алгоритма рассматриваются разные значения T

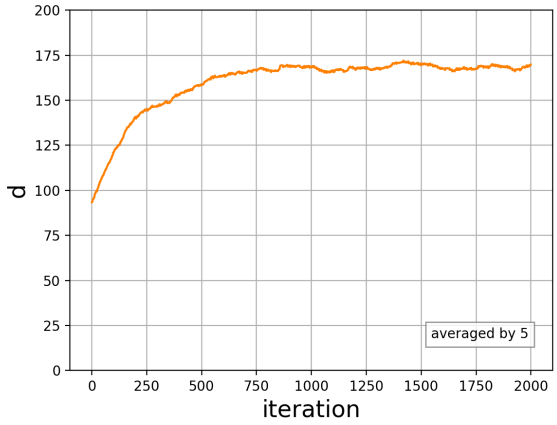
В качестве демонстрации основной гипотезы о независимости значения плато от n поставлены несколько экспериментов с разными n, но с одной и той же $T = 10$.

На графиках изображено усредненное поведение функции d - количество разных бит у x и x^{opt} за 20 запусков.

Как видно из экспериментов они все останавливаются примерно на одном уровне по отношению к n.

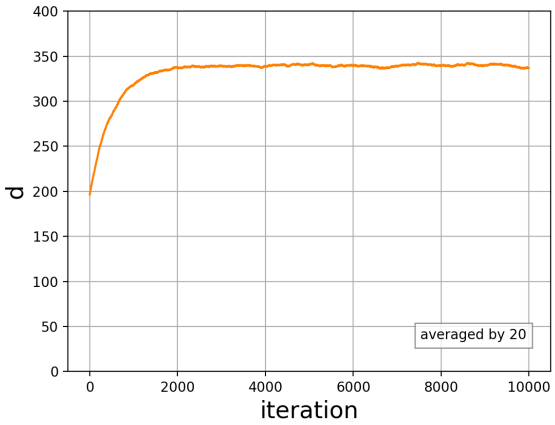


n = 100

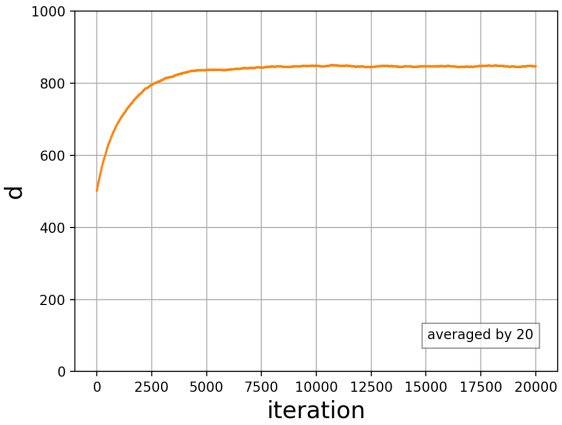


n 200

Рисунок 3 – Усредненное поведение алгоритма по 20 итерациям при T = 10



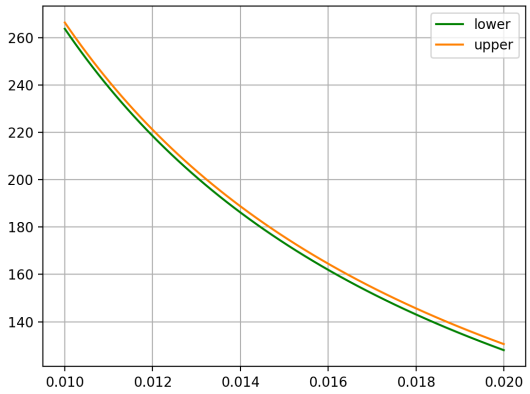
n = 100

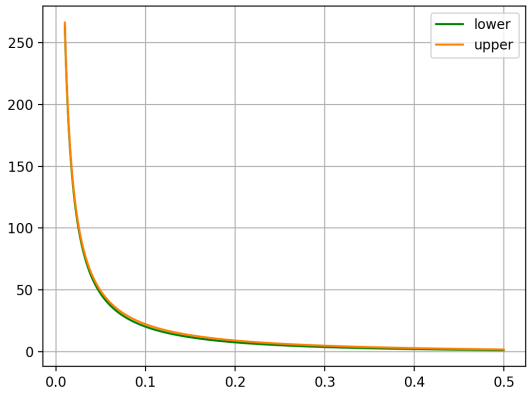


n = 200

Рисунок 4 – Усредненное поведение алгоритма по 20 итерациям при T = 10

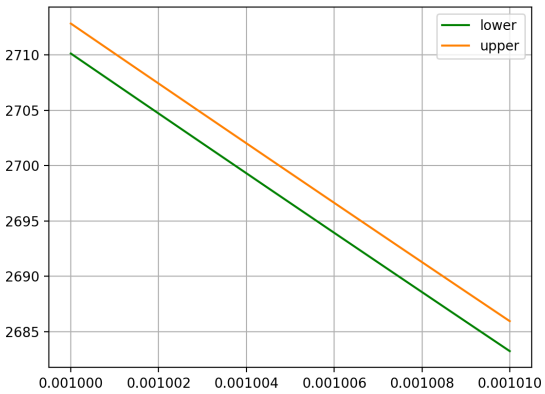
В рамках анализа поведения было установлено, что если бы мы решали обратную задачу, то есть для данного c, пытались бы найти T, то оно лежало бы в промежутке (g_{lower}, g_{upper}) где $g_{lower} = \frac{(1-c)}{\exp(-r+r\cdot c)\cdot c\cdot r}$; $g_{upper}(c) = \frac{(1-c)}{\exp(-r+r\cdot c)\cdot c\cdot r\cdot(1+\exp(-r\cdot c)\cdot c\cdot r^2)}$ так что для нахождения. T взять обратную функцию от данной Для анализа функций g_{upper} и g_{lower} приведены следующие графики:

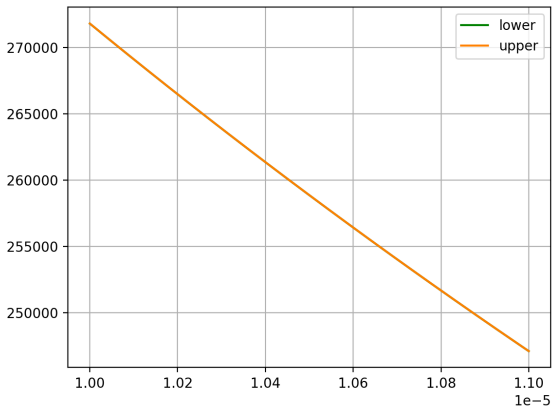




с от 1 до 2 с от 1 до 5

Рисунок 5 – Зависимость T от c

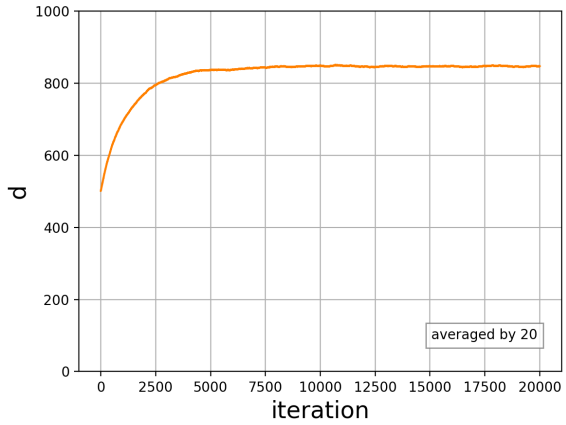




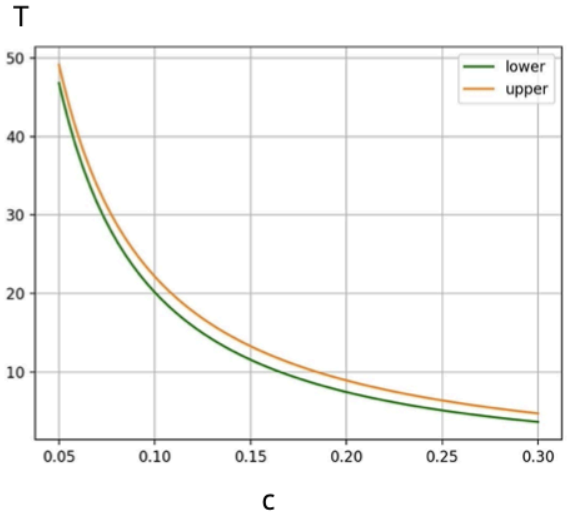
с от 0.001 до 0.00101 с от 0.0001 до 0.00011

Рисунок 6 – Зависимость T от c

Рассмотрим как предсказанная функция ведёт себя на практике



Усреднённое значение по 20 итерациям



c от 0.05 до 0.30

Рисунок 7 – Анализ качества предсказанной функции

Если взять значение из графика справа, то для $T = 10$ значение c должно быть равно примерно 0.17, то есть остановиться на $1000 \cdot (1 - 0.17) = 830$, что и показывается на практике.

ЗАКЛЮЧЕНИЕ

В результате были сформуилорованы и доказаны несколько теорем, которые помогут в дальнейшим анализировать эволюционные алгоритмы с динамическими изменениями. Даны оценки кака точка стабилизация будует находиться для данного T. То есть в каком месте алгоритм будет находиться в независимости от n. Выведены пару drift-теорем, которые могут помочь дальнейшим исследованиям динамических оптимизаций, в частности теоремы применены к данной задаче в следствие чего была установлены такие значение T, при которых алгоритм будет находить оптимумы за то же асимптотическое время. Оценки были подтверждены экспериментальными данными.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 A Blend of Markov-Chain and Drift Analysis // International Conference on Parallel Problem Solving from Nature. — 2008. — Т. 5199. — С. 41–51.

2 A general noise model and its effects on evolution strategy performance / C. Qian [и др.] // Evolutionary Computation. — 2017. — С. 1399–1406.

3 *Droste S.* Analysis of the (1+1) EA for a Dynamically Bitwise Changing OneMax // Genetic and Evolutionary Computation. — 2003. — Т. 1, № 1. — С. 909–921.

4 *Gießen C.*, *Witt C.* The Interplay of Population Size and Mutation Probability in the (1+λ) EA on OneMax // Algorithmica. — 2017. — Т. 78, № 2. — С. 587–609.

5 *Jansen T.*, *Dang D. C.*, *Lehre P.* Populations Can Be Essential in Tracking Dynamic Optima // Algorithmica. — 2017. — Т. 78, № 2. — С. 660–680.

6 *Jansen T.*, *Zarges C.* Analysis of Randomised Search Heuristics for Dynamic Optimisation // Evolutionary Computation. — 2015. — Т. 23. — С. 513–541.

7 *Kötzing T.*, *Lissovoi A.*, *Witt C.* (1+1) EA on Generalized Dynamic OneMax // the 2015 ACM Conference on Foundations of Genetic Algorithms XIII (FOGA ’15). — 2015. — С. 40–51.

8 *Lehre P. K.*, *Witt C.* General Drift Analysis with Tail Bounds. — 2013. — http://arxiv.org/abs/1307.2559.

9 *Lengler J.* Drift Analysis. — 2018. — `https://arxiv.org/abs/1712.009`.

10 *Muhlenbein H.* How genetic algorithms really work: I. mutation and hill climbing. — 1992.

11 Optimal resampling for the noisy OneMax problem / J. Liu [и др.] // CoRR. — 2016.

12 *S. Droste T. J.*, *Wegener I.* On the analysis of the (1+1) evolutionary algorithm // Theoretical Computer Science. — 2002. — Т. 276, № 1. — С. 51–81.