ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет Информационных технологий и программирования

Кафедра Компьютерных технологий

Направление подготовки 01.03.02 Прикладная математика и информатика

ОТЧЕТ

по учебной практике

Тема задания: Адаптивная настройка вероятности мутации в эволюционных алгоритмах с помощью обучения с подкреплением

Студент Родионова Анна Дмитриевна, **группа №** М3339

Руководитель практики от университета: Буздалова Арина Сергеевна, сотрудник МНЛ КТ

Ответственный за практику от университета: Корнеев Г. А., зам. зав. каф. КТ по УР

Практика пройдена с оценкой _____

Дата _____

Санкт-Петербург
2018

1. **Цели и задачи практики**

Основной целью практики является повышение эффективности эволюционных алгоритмов за счет разработки новых механизмов адаптивной настройки вероятности мутации.

2. **Сведения об организации**

Университет ИТМО – один из ведущих российских вузов в сфере информационных технологий. Он известен своими достижениями в области олимпиадного программирования, а также международной научной лабораторией «Компьютерные технологии», которая ведет исследования по четырем направлениям: машинное обучение, биоинформатика, эволюционные вычисления, дискретная оптимизация.

3. **Занимаемая должность**

Занимаемся должность – программист-исследователь.

4. **Использованные технологии**

Для реализации алгоритмов использовался язык программирования C++.

5. **Цели проекта**

Разработать (1 + λ) эволюционный алгоритмов, использующий обучение с подкреплением для настройки вероятности мутации, и на примере задачи “One Max” сравнить его производительность с производительностью базовых (1 + λ) эволюционных алгоритмов:

- без адаптивной настройки вероятности мутации;
- с адаптивной настройкой вероятности мутации с использованием разделения на две подгруппы.

6. Описание выполненного проекта

Эволюционные алгоритмы (ЭА) – направление в искусственном интеллекте, которое использует и моделирует процессы естественного отбора. Примером такого алгоритма является (1 + λ) эволюционный алгоритм, являющийся вариацией эволюционной стратегии, или другими словами – алгоритмом поиска, основанным на идеях приспособления и эволюции. Суть его в том, что на каждой итерации генерируется λ промежуточных решений и среди них выбирается лучшее, которое становится “родителем” на следующей итерации.

Эффективность работы эволюционных алгоритмов сильно зависит от значений используемых параметров, в частности, от вероятности мутации. Для ряда эволюционных алгоритмов и простых задач оптимизации известны оптимальные функциональные зависимости значения вероятности мутации от приспособленности особей в текущем поколении. Однако такие зависимости сложно угадать. Вместо них можно использовать механизмы адаптивной настройки, согласно которым вероятность мутации меняется в процессе работы эволюционного алгоритма.

В выполненном проекте было проведено сравнение трех (1 + λ) эволюционных алгоритмов, использующих разные механизмы адаптивной настройки вероятности мутации на примере задачи “One Max”. Данная задача заключается в том, чтобы найти битовую строку длины *n*, состоящую только из единиц, имея изначально случайную битовую строку такой длины. Таким образом, функция приспособленности в задаче “One Max” – число единиц в текущем решении.

Рассмотренные (1 + λ) эволюционные алгоритмы:

- ***без адаптивной настройки вероятности мутации***

Ожидаемое время работы алгоритма [1]: $O(\frac{1}{2}\cdot\frac{n\cdot ln(ln(\lambda))}{ln(\lambda)}+\frac{e^{pn}}{pn}\cdot\frac{n\cdot ln(n)}{\lambda})$.

Здесь вероятность мутации *p* является константой на протяжении всей работы алгоритма. Доказано, что в таком случае оптимальное ее значение равняется *1/n*.

Псевдокод:

Select x uniformly at random from $\{0, 1\}^n$ and set $p \leftarrow 1/n$.

for $t \leftarrow 1, 2, \ldots$ **do**

  for $i \leftarrow 1, \ldots, \lambda$ **do**

    Create x_i by flipping each bit in a copy of *x* independently with probability *p*.

  $x^* \leftarrow argmax_{x_i} f(x_i)$ (breaking ties randomly).

  if $f(x^*) \geq f(x)$ **then**

    $x \leftarrow x^*$.

- ***с адаптивной настройкой вероятности мутации с использованием разделения на две подгруппы***

Ожидаемое время работы алгоритма [1]: $O(\frac{n}{ln(\lambda)}+\frac{n\cdot ln(n)}{\lambda})$.

В данном случае изначально заданная вероятность будет изменяться. На каждой итерации создается две примерно равных группы решений. Вероятность мутации для одной группы равняется *p/2*, а для другой *2p*, где *p* – текущая вероятность мутации. Равновероятно после текущей итерации вероятность мутации станет либо такой, какая была у группы, в которой было получено лучшее решение, либо равновероятно поменяется на случайную из *p/2* и *2p* (этот случай необходим, чтобы не застрять в зоне локального экстремума).

Псевдокод:

Select x uniformly at random from $\{0, 1\}^n$ and set $p \leftarrow 2/n$.

for $t \leftarrow 1, 2, \ldots$ **do**

for $i \leftarrow 1, \ldots, \lambda$ **do**

Create x_i by flipping each bit in a copy of x independently with probability $p/2$ if $i \leq \lambda/2$ and with probability $2p$ otherwise.

$x^* \leftarrow argmax_{x_i} f(x_i)$ (breaking ties randomly).

if $f(x^*) \geq f(x)$ **then**

$x \leftarrow x^*$.

Perform one of the following two actions with probability 0.5:

- Replace p with probability of mutation that x^* has been created with.
- Replace p with either $p/2$ or $2p$, each with probability 0.5.

Replace p with $min\{max\{2/n, p\}, 1/4\}$.

Замечания:

1) Значение p должно быть всегда в диапазоне $[2/n, 1/4]$. Таким образом, в случае, если выбранная операция выводит вероятность мутации за его границы, для следующей итерации значение p останется неизменным.
2) В реализованной версии начальное значение p равнялось $2/n$.

- ***с адаптивной настройкой вероятности мутации с использованием обучения с подкреплением***

Ожидаемое время работы алгоритма: неизвестно.

В качестве алгоритма обучения с подкреплением был использован алгоритм *Q-learning [2]*, в котором состояние s определялось как число потомков, лучших, чем родитель, а два возможных действия a – как увеличение или уменьшение вероятности мутации в два раза. Функция $Q(s, a)$ – ожидаемая награда агенту за действие a в состоянии s. Изначально она инициализируется нулями и в дальнейшем после каждой итерации изменяется по формуле:
$Q(s,a) := Q(s,a) + \alpha \cdot (r + \gamma \cdot max_{a'} Q(s',a') - Q(s,a))$, где

α– это фактор обучения (константа). Чем он выше, тем сильнее агент доверяет новой информации.

γ– это фактор дисконтирования (константа). Чем он меньше, тем меньше агент задумывается о выгоде от будущих своих действий.

s – состояние, в котором агент находился на момент начала итерации.

a – переход, который агент совершил в конце прошлой итерации.

s' – состояние, соответствующее новому набору потомков.

r – поощрение, полученное лучшим потомком.

В итоге следующий переход *a'* (способ изменения вероятности мутации) агент выбирает как $argmax_a Q(s', a)$.

Псевдокод:

Set $Q(s, a) \leftarrow 0$ for each state *s* and for each transition *a*.

Select x uniformly at random from $\{0, 1\}^n$ and set $p \leftarrow 2/n$.

for $t \leftarrow 1, 2, \ldots$ **do**

for $i \leftarrow 1, \ldots, \lambda$ **do**

Create x_i by flipping each bit in a copy of *x* independently with probability *p*.

$x^* \leftarrow argmax_{x_i} f(x_i)$ (breaking ties randomly).

s' ← the number of offspring better than parent.

r ← *f(x*) - f(x)*.

if $f(x^*) \geq f(x)$ **then**

$x \leftarrow x^*$.

For the old state s, the new state *s'*, the transition *a*, with which the probability of mutation and reward *r* was updated for the last time update the value of $Q(s, a)$:

$Q(s, a) \leftarrow Q(s, a) + 0.8 \cdot (r + 0.2 \cdot max_{a'} Q(s', a') - Q(s, a))$.

Choose an action *a'* for changing *p*:

$a' \leftarrow argmax_a Q(s', a)$.

Replace *p* with probability of mutation that is determined by the action of *a'*.

Replace *p* with *min{max{2/n, p}, 1/4}*.

Замечания:

1) Для вероятности мутации инициализируемое значение, диапазон и поведение в случае попытки выбранной операции вывести ее значение из диапазона – совпадают с описанными в предыдущем алгоритме.
2) В реализованной версии αпринята равной *0.8*, γ– *0.2*, а поощрение потомка определяется разностью значений оценочной функции для этого потомка и его родителя.
3) Изначально последний совершенный переход *a* не определен, поэтому после первой итерации пересчета функции *Q(s, a)* не происходит.

Источники:

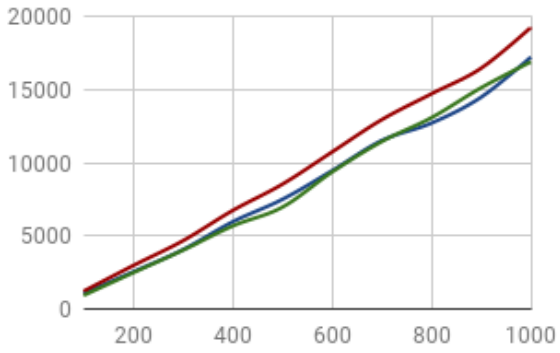
1) *Doerr B., Giessen C., Witt C., Yang J.* The (1+λ) Evolutionary Algorithm with Self-Adjusting Mutation Rate // Proceedings of Genetic and Evolutionary Computation Conference, 2017. – Pp. 1351-1358.
2) *Николенко С. И., Тулупьев А. Л.* Самообучающиеся системы. М., 2009.

Полученные результаты:

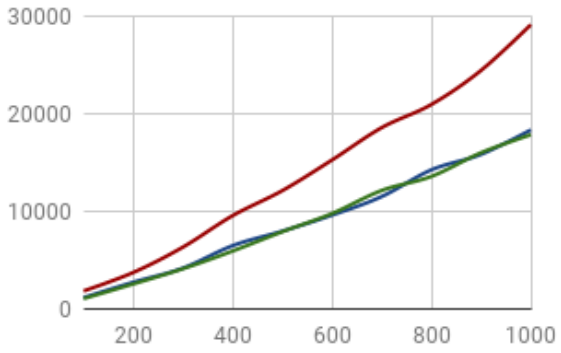
На рисунке 1 представлены сравнительные графики зависимости числа вычислений функции приспособленности от длины строки *n* для различных λ:

- без настройки *(синий)*;
- с разделением на две подгруппы *(красный)*;
- с обучением с подкреплением *(зеленый)*.

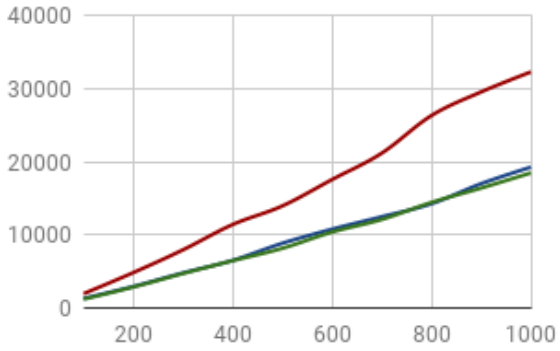
λ = 1



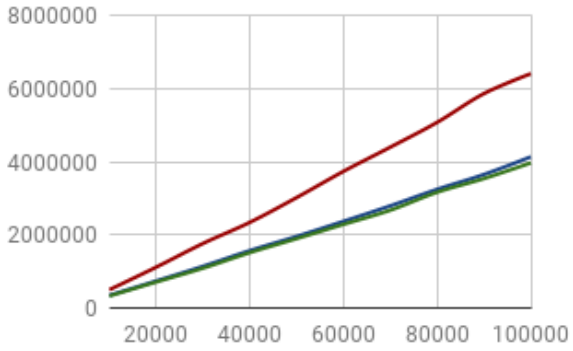
λ = 5



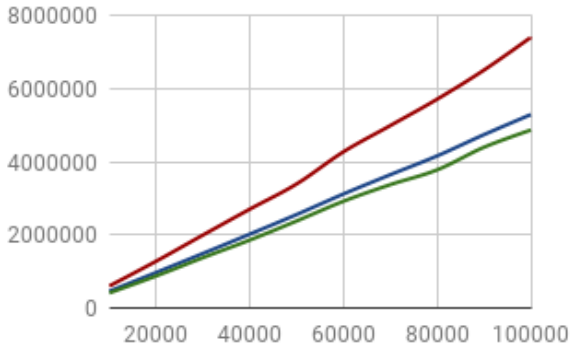
λ = 10



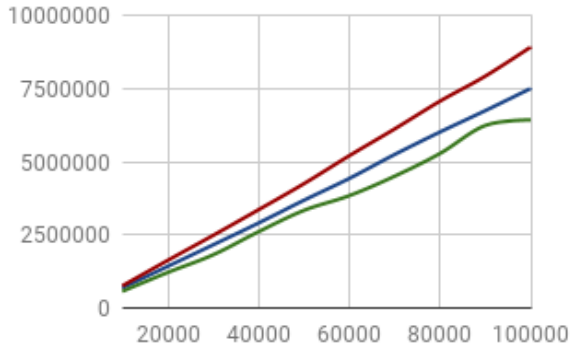
λ = 50



λ = 100



λ = 200



λ = 400

λ = 800

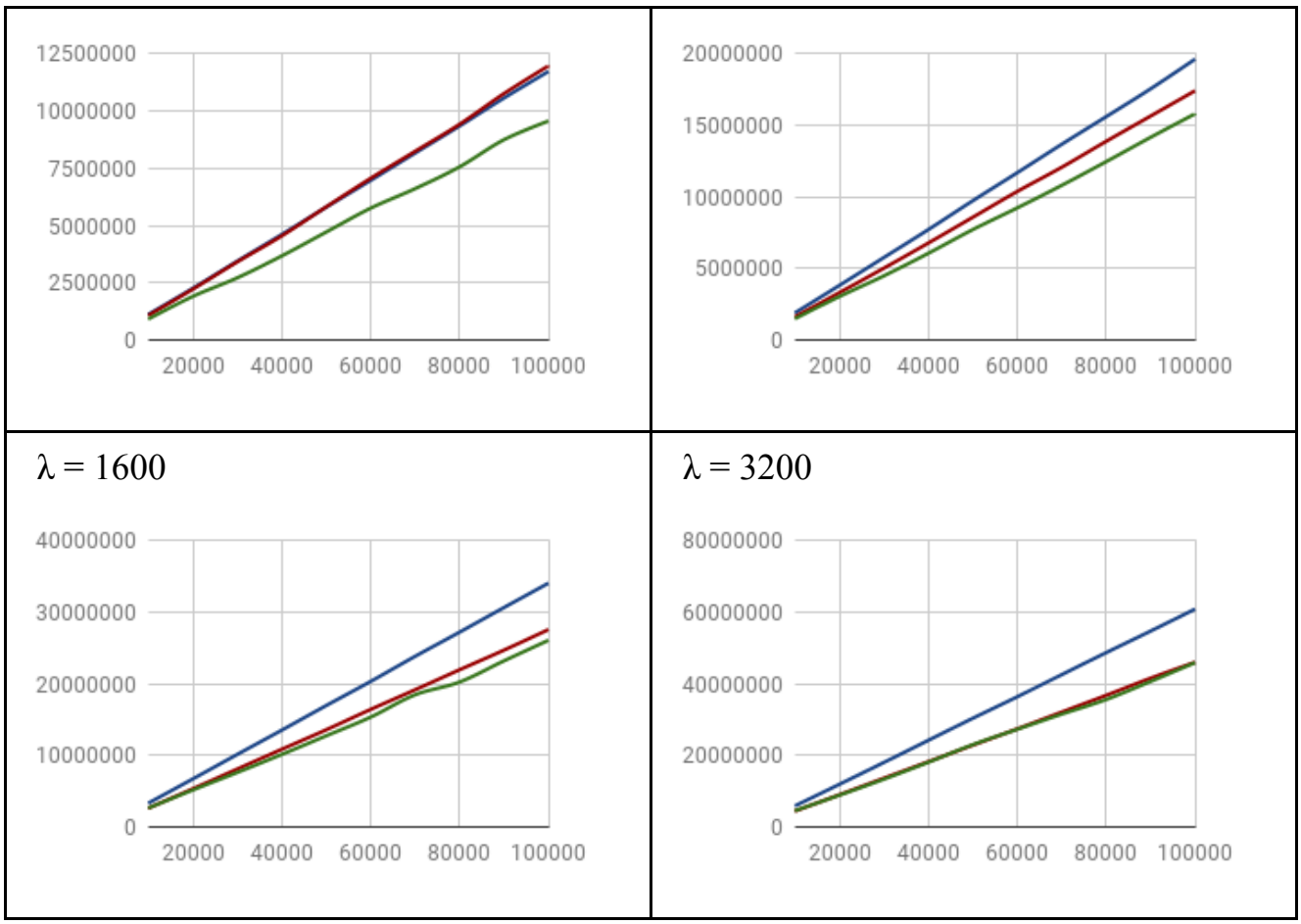


Рисунок 1 — зависимость числа вычислений функции приспособленности от длины строки n для различных λ.

Замечание:

Для возможности получения результатов для больших *n* за удовлетворительное время для мутации была использована оптимизация при которой вместо инвертирования каждого бита с вероятностью p находится каждый следующий инвертируемый бит путем округления вниз величины $i + 1 + log_{1-p}k$, где *k* – случайное число в диапазоне от 0 до 1, а *i* – индекс текущего инвертируемого бита, изначально равный *-1*. Таким образом, одна мутация будет происходить не за *n*, а за *np*, что близко к константе.

7. Выводы

По полученным графикам можно сделать два вывода:

1) Алгоритм с разделением на две подгруппы становится эффективен только при больших значениях λ и n, а при малых значениях он “проигрывает” алгоритму без настройки.
2) Алгоритм, использующий обучение с подкреплением для всех рассмотренных λ работал не хуже, чем лучший из двух базовых алгоритмов при текущей λ. Таким образом, результаты исследования показали, что такой механизм адаптивной настройки вероятности мутации в эволюционных алгоритмах является достаточно эффективным для рассматриваемой базовой задачи и, значит, имеет смысл изучение его производительности для более сложных задач.