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

О Т Ч Е Т
по учебной практике

Тема задания: Исследование механизмов адаптивной настройки вероятности мутации в эволюционных алгоритмах

Студент Антонов Кирилл Александрович, **группа №** М3338

Руководитель практики от организации: Буздалова Арина Сергеевна, сотрудник МНЛ КТ, место работы — ИТМО МНЛ КТ

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

Цели и задачи практики

1. Получить общее представление об эволюционных алгоритмах (например, воспользовавшись источником [1])

2. Ознакомиться с туториалом ([2] в списке источников.

3. Реализовать механизм настройки вероятности мутации из [3], описанный на слайде 67 туториала [2] для алгоритма (1+ λ) ЭА на задаче *OneMax*.

4. Реализовать механизм, предлагаемый после слова “Note” на слайде 67 туториала [2].

5. Провести вычислительный эксперимент: построить графики времени работы ЭА с использованием реализованных механизмов, сравнить эффективность механизмов.

Сведения об организации

Международная научная лаборатория "Компьютерные технологии" создана на основе кафедр «Компьютерные технологии», «Технологии программировании», «Программная инженерия и верификация программ» Университета ИТМО, лаборатории «Алгоритмы сборки геномных последовательностей», созданной на основе Решения учёного совета Университета ИТМО от 27.12.2011 г., на основе научно-исследовательского центра «Технологии программирования и искусственного интеллекта», организованного в рамках реализации программы развития Университета ИТМО на 2009-2018 годы, а также научно-образовательного центра «Разработка методов сборки генома, сборки транскриптома и динамического анализа протеома», созданного в рамках Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы. Мероприятие 1.1 «Поддержка исследований, проводимых коллективами научно-образовательных центров» по научному направлению «Науки о жизни (Живые системы)» в области «Геномные, протеомные и постгеномные технологии».

Лаборатория ведёт исследования по четырём направлениям: теория кодирования, биоинформатика, машинное обучение, технологии программирования

Занимаемая должность

Программист-исследователь

Основные поставленные задачи: эксперементально сравнить механизмы адаптивной настройки вероятности мутации при решении задачи *OneMax* алгоритмом (1+ λ) ЭА, построить графики времени работы алгоритма и сделать выводы.

Использованные технологии

Был выбран язык C++ чтобы, реализованные алгоритмы быстро работали. Для разработки на C++ была использована IDE qt-creator. Использовалась система сборки Cmake и фреймворки Qt, qcustomplot для автоматического простроения графиков. Так же был использован bash для создания скриптов, помогающий организовывать результаты, git и travis-ci для контроля изменений проекта. Были использованы некоторые возможности библиотеки Boost

Цели проекта

Повышение эффективности эволюционных алгоритмов за счёт разработки новых механизмов адаптивной настройки вероятности мутации.

Описание выполненного проекта

Изучение теории

Первоначальные умения в области эволюционных алгоритмов были получены в на занятиях на соответствующем курсе в университете. Для получения более глубоких знаний потребовался источник [1].
С помощь туториала из источника [2], было получено представление о 1 + λ алгоритмах и настройки их параметров.
Так же была изучена библиотека QCustomPlot для построения графиков при помощи C++.

Описание предметной области

Эффективность работы эволюционных алгоритмов (ЭА) 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 простые правила, или механизмы адаптивной настройки, согласно которым вероятность мутации меняется в процессе работы эволюционного алгоритма. Например, одним из известных простых механизмов является “правило одной пятой”: если за последние пять итераций увеличение приспособленности происходило больше, чем один раз -- увеличить вероятность мутации, в противном случае -- уменьшить. В данной работе был проведён анализ некоторых существующих механизмов адаптивной настройки вероятности мутации, и предложен новый не уступающие по эффективности известным функциональным зависимостям при решении различных задач оптимизации.

Описание подхода 1 + λ

Для получения определенной особи из исходного поколения, используя только мутации применяется подход 1 + λ. То есть из текущего поколения особей выбирается лучшая (с наибольшем значением функции приспособленности), возможно производится настройка параметров мутации, проводится мутация выбранной особи λ раз и получается новое поколение из λ особей. Процесс выбора лучшей особи, настройки параметров и ее мутации продолжается пока не будет получена особь с нужным значением функции приспособленности.
Типичной тестовой задачей, для проверки 1 + λ ЭА является задача OneMax.

Описание задачи OneMax

Особь — строка размера n из элементов 0 и 1. Один элемент строки будет называть битом.
Функция приспособленности — возвращает количество элементов 1 в особи.
Требуется получить особь состоящую полностью из 1.

Псевдокод простого (1 + λ) ЭА без настройки параметров мутации для решения этой задачи:

Algorithm 1: $(1 + \lambda)$ ЭА без настройки параметров мутации

$x[\lambda][n] \leftarrow init\ randomly$
$parent = x[random\_integer(0 \dots \lambda)]$
while $cnt(parent) \neq n$ **do**
    for $i = 0 \dots \lambda$ **do**
        $x[i] = flip(parent, \frac{1}{n})$
    end for
    $candidate = arg\_max(cnt(x[1]), \dots, cnt(x[\lambda]))$
    if $cnt(parent) \leq cnt(candidate)$ **then**
        $parent = candidate$
    end if
end while

Особь обозначается - *x[i]*, функция приспособленности — *cnt*, мутация производится с помощью функции flip которая принимает первым аргументов особь, а вторым

параметр т. е. вероятность с которой она будет проводить мутации каждого бита в особи.

Выполнение проекта

Первая задача была реализовать этот простой алгоритм и сравнить его количество вычислений функции приспособленности и количество вычислений функции приспособленности, которое даёт теоретическая оценка. Далее был реализован алгоритм из статьи источник [3], в котором присутствует настройка вероятности с которой производится мутация. Псевдокод алгоритма:

Algorithm 2: $(1 + \lambda)$ ЭА с настройкой вероятности мутации и делением на две субпопуляции

$x[\lambda][n] \leftarrow init\ randomly$
$parent = x[random\_integer(0 \dots \lambda)]$
$p = \frac{1}{n}$
while $cnt(parent) \neq n$ **do**
    for $i = 0 \dots \frac{\lambda}{2}$ **do**
        $x[i] = flip(parent, \frac{p}{2})$
    end for
    for $i = \frac{\lambda}{2} \dots \lambda$ **do**
        $x[i] = flip(parent, 2 \times p)$
    end for
    $candidate = arg\_max(cnt(x[1]), \dots, cnt(x[\lambda]))$
    if $cnt(parent) \leq cnt(candidate)$ **then**
        $parent = candidate$
    end if
    Сделать одно из следующих двух действий с вероятностью $\frac{1}{2}$

- *Заменить* p *на вероятность, с которой был создан candidate*
- *Заменить* p *на* $\frac{p}{2}$ *или на* $2 \times p$ *с вероятностью* $\frac{1}{2}$

    Заменить p *на* $\min(\max(\frac{2}{n}, p), \frac{1}{4})$
end while

По теоретической оценки количества вычислений функции приспособленности, приведённой в статье, этот алгоритм должен был делать меньшее их количество чем предыдущий. Чтобы это проверить на больших данных за разумное время, потребовалось оптимизировать реализации этих алгоритмов.

Оптимизация мутации: пользуясь приёмами теории вероятности, можно посчитать индекс следующего инвертируемого бита, если i -- индекс текущего инвертируемого бита. Он будет равен $i+1+[\log_{1-p}(r)]$, где r - случайное число в диапазоне от 0 до 1, результат округляется вниз. Тогда мутация выглядит примерно так: начинаем с i = -1, считаем следующий инвертируемый бит, применяем инвертирование, и так далее, пока не выйдем за границы строки. Таким образом, мутация стала проводится не за n итераций, а за n*p (p - вероятность мутации), что близко к константе, т.к. p = 1/(n*const).

Оптимизация обновления родителя: хранятся не особи, а один вектор-патч с лучшим значением функции приспособленности. В этом патче записываются мутации относительно самой первой особи, соответствующие ребенку с наилучшим значением функции приспособленности. Во время мутации формируется временный патч. Если его функция приспособленности не хуже, чем у лучшего патча, лучший патч обновляется.

Чтобы ускорить получение результатов тесты алгоритмов запускались параллельно в 8 потоков через thread pool из библиотеки Boost. Один поток запускал один из алгоритмов на одном из тестов.

Далее был реализован алгоритм предлагаемый после слова “Note” на слайде 67 туториала [2], с возможностью менять параметры мутации. Его псевдокод:

Algorithm 3: $(1 + \lambda)$ ЭА с настройкой вероятности мутации и делением на три субпопуляции

$x[\lambda][n] \leftarrow$ *init randomly*
$parent = x[random\_integer(0 \dots \lambda)]$
$p = \frac{1}{n}$
$params[3] \leftarrow \{1 < C_1, C_2 = 1, 0 < C_3 < 1\}$
while $cnt(parent) \neq n$ **do**
    for $i = 0 \dots \frac{\lambda}{3}$ **do**
        $x[i] = flip(parent, p \times params[0])$
    end for
    for $i = \frac{\lambda}{3} \dots 2 \times \frac{\lambda}{3}$ **do**
        $x[i] = flip(parent, p \times params[1])$
    end for
    for $i = 2 \times \frac{\lambda}{3} \dots \lambda$ **do**
        $x[i] = flip(parent, p \times params[2])$
    end for
    $candidate = arg\_max(cnt(x[1]), \dots, cnt(x[\lambda]))$
    if $cnt(parent) \leq cnt(candidate)$ **then**
        $parent = candidate$
    end if
    Сделать одно из следующих двух действий с вероятностью $\frac{1}{2}$

- *Заменить* p *на вероятность, с которой был создан candidate*
- *Заменить* p *на* $C_1 \times p$ *или на* $C_3 \times p$ *с вероятностью* $\frac{1}{2}$

    Заменить p *на* $\min(\max(\frac{2}{n}, p), \frac{1}{4})$
end while

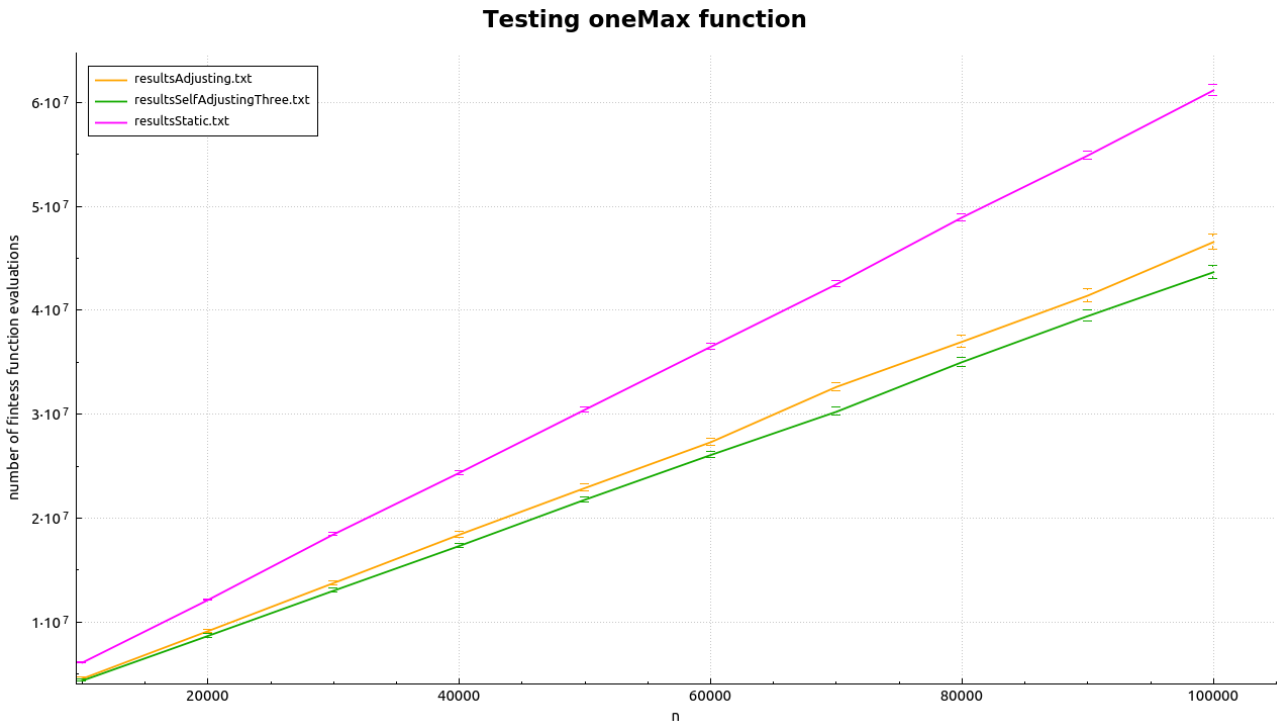
Реализация была оптимизирована так же, как и у предыдущих алгоритмов, чтобы его время работы было небольшое.

Поиск оптимальных констант для нового алгоритма

Чтобы искать оптимальные параметры была введена следующая функция расстояния, которая считалась для результатов алгоритма 2 (с делением на две субпопуляции) и алгоритма 3:
d(a[..], b[..]) -> sum (a[i] — b[i]), где a[i], b[i] – количество вычислений функции приспособленности для n = i алгоритма 2 и алгоритма 3.
Сперва, поиск оптимальных параметров производился выбором произвольных чисел из промежутков (1., 2) для первого параметра и (0.1, 1) для второго. Дальше стало понятно где примерно находятся оптимальные параметры и каждый «подозрительный» промежуток промерялся последовательным поиском. Получилось, что оптимальный первый параметр равен 1.4 с точностью 0.1, а второй 0.7 с точностью 0.05.

Выводы

В процессе выполнения проекта было реализовано несколько 1 + λ алгоритмов и сравнивалось количество раз, которое они производят вычислений функции приспособленности в зависимости от размера особи и λ. Результаты работы алгоритмов для λ = 3200 приведены на следующем графике [Testing oneMax function]. Алгоритм 3 с параметрами 1.4 и 0.7 показан зеленым цветом, алгоритм 2 оранжевым и алгоритм 1 серенивым.
На графике так же показаны стандартные отклонения.
В ходе экспериментов, выяснилось что при некоторых параметрах алгоритм 3 выигрывает у алгоритма 2. Так же выяснилось, что для алгоритма 3 оптимальная константа $C_1 = 1.4$ с точностью 0.1, а $C_2 = 0.7$ с точностью 0.05.



Источники

1. Конспект лекций Luke S. Essentials of Metaheuristics (https://cs.gmu.edu/~sean/book/metaheuristics/Essentials.pdf)

2. Туториал Doerr C. Non-static parameter choices in Evolutionary Computation // GECCO 2017 (http://www-ia.lip6.fr/~doerr/GECCO17tutorial.pdf)

3. Статья Doerr B., Gießen C., Witt C., Yang J. The (1 + λ) evolutionary algorithm with self-adjusting mutation rate // GECCO 2017