

HarrixOptimizationAlgorithms. Сборник описаний алгоритмов оптимизации нулевого порядка. v. 1.5

А. Б. Сергиенко

5 февраля 2014 г.

Аннотация

В данном документе дано собрано множество описаний стандартных, нестандартных алгоритмов, модификаций стандартных. Здесь приведено лишь описание алгоритмов, а не их исследование эффективности. Большинство алгоритмов неэффективны. Это своеобразная "свалка" алгоритмов, которые используются автором. На данный документ можно ссылаться в своих работах, чтобы указать, что тот или иной алгоритм оптимизации подробно описан в этом документе. Тут нет исследований эффективности алгоритмов с данными операторами — это задача иных проектов. Здесь представлено только описание алгоритмов.

Содержание

1 Введение	3
2 Условные обозначения	3
3 Некоторая вводная информация	4
4 Постановка задачи оптимизации	5
5 Стандартный генетический алгоритм	6
5.1 Стандартный генетический алгоритм для решения задач на бинарных строках .	6
5.2 Стандартный генетический алгоритм для решения задач на вещественных строках	6
6 Модификации генетического алгоритма	7
6.1 Генетический алгоритм для решения задач на бинарных строках с изменяющимся соотношением числа поколений и размера популяции	7
6.2 Генетический алгоритм для решения задач на вещественных строках с изменяющимся соотношением числа поколений и размера популяции	8
6.3 Генетический алгоритм для решения задач на бинарных строках с турнирной селекцией, где размер турнира изменяется от 2 до размера популяции	9

6.4	Генетический алгоритм для решения задач на вещественных строках с турнирной селекцией, где размер турнира изменяется от 2 до размера популяции	10
6.5	Генетический алгоритм для решения задач на бинарных строках, в котором есть только два вида скрещивания: одноточечное и двухточечное скрещивание с возможностью полного копирования одного из родителей	11
6.6	Генетический алгоритм для решения задач на вещественных строках, в котором есть только два вида скрещивания: одноточечное и двухточечное скрещивание с возможностью полного копирования одного из родителей	12
6.7	Генетический алгоритм для решения задач на бинарных строках с турнирной селекцией с возвращением, где размер турнира изменяется от 2 до размера популяции	13
6.8	Генетический алгоритм для решения задач на вещественных строках с турнирной селекцией с возвращением, где размер турнира изменяется от 2 до размера популяции	14
7	Алгоритмы Монте-Карло	16
7.1	Метод Монте-Карло (Monte-Carlo) для решения задач на бинарных строках . . .	16
7.2	Метод Монте-Карло (Monte-Carlo) для решения задач на вещественных строках	16
	Список литературы	16

1 Введение

Это своеобразная «свалка» алгоритмов оптимизации, которые используются автором. Большинство алгоритмов неэффективны. Здесь они приведены, чтобы можно было ссылаться на них.

Данный документ представляет его версию 1.0 от 5 февраля 2014 г.

Последнюю версию документа можно найти по адресу:

<https://github.com/Harrix/HarrixOptimizationAlgorithms>

С автором можно связаться по адресу sergienkoanton@mail.ru или <http://vk.com/harrix>.

Сайт автора, где публикуются последние новости: <http://blog.harrix.org/>, а проекты располагаются по адресу <http://harrix.org/>.

2 Условные обозначения

$a \in A$ — элемент a принадлежит множеству A .

\bar{x} — обозначение вектора.

$\arg f(x)$ — возвращает аргумент x , при котором функция принимает значение $f(x)$.

$Random(X)$ — случайный выбор элемента из множества X с равной вероятностью.

$Random(\{x^i \mid p^i\})$ — случайный выбор элемента x^i из множества X , при условии, что каждый элемент $x^i \in X$ имеет вероятность выбора равную p^i , то есть это обозначение равнозначно предыдущему.

$random(a, b)$ — случайное действительное число из интервала $[a; b]$.

$int(a)$ — целая часть действительного числа a .

$\mu(X)$ — мощность множества X .

Замечание. Оператор присваивания обозначается через знак « $=$ », так же как и знак равенства.

Замечание. Индексация всех массивов в документе начинается с 1. Это стоит помнить при реализации алгоритма на C-подобных языках программирования, где индексация начинается с нуля.

Замечание. Вызывание трех функций: $Random(X)$, $Random(\{x_i \mid p_i\})$, $random(a, b)$ — происходит каждый раз, когда по ходу выполнения формул, они встречаются. Если формула итерационная, то нельзя перед ее вызовом один раз определить, например, $random(a, b)$ как константу и потом её использовать на протяжении всех итераций неизменной.

Замечание. Надстрочный индекс может обозначать как возведение в степень, так и индекс элемента. Конкретное обозначение определяется в контексте текста, в котором используется формула с надстрочным индексом.

Замечание. Если у нас имеется множество векторов, то подстрочный индекс обозначает номер компоненты конкретного вектора, а надстрочный индекс обозначает номер вектора во

множестве, например, $\bar{x}^i \in X$ ($i = \overline{1, N}$), $\bar{x}_j^i \in \{0; 1\}$, ($j = \overline{1, n}$). В случае, если вектор имеет свое обозначение в виде подстрочной надписи, то компоненты вектора проставляются за скобками, например, $(\bar{x}_{max})_j = 0$ ($j = \overline{1, n}$).

Замечание. При выводе матриц и векторов элементы могут разделяться как пробелом, так и точкой с запятой, то есть обе записи $(1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1)^T$ и $(1; 1; 1; 1; 1; 1; 1; 1)^T$ допустимы.

Замечание. При выводе множеств элементы разделяются только точкой с запятой, то есть допустима только такая запись: $\{1; 1; 1; 1; 1; 1; 1; 1\}^T$.

3 Некоторая вводная информация

В каждом классе решаемых задач (задачи бинарной оптимизации, задачи вещественной оптимизации и др.) определен некий основной алгоритм. Обычно им является стандартный генетический алгоритм. И с ним сравниваются все остальные алгоритмы оптимизации, чтобы можно было выявить лучший алгоритм на множестве тестовых задач при определенных фиксированных настройках. Алгоритмы, которые ср

Алгоритмы представленные в данной работе бывают нескольких типов, которые описаны ниже.

Основной алгоритм оптимизации — некий алгоритм в классе решаемых задач (задачи бинарной оптимизации, задачи вещественной оптимизации и др.) относительно которого производится сравнение всех остальных алгоритмов. Данный алгоритм может со временем меняться. Это происходит в случае, если обнаруживается алгоритм, который по эффективности превосходит (или не хуже) предыдущий основной алгоритм оптимизации по всем параметрам на всех тестовых функциях.

Сравниваемый алгоритм оптимизации — некий алгоритм, который сравнивается по эффективности с основным алгоритмом и другими сравниваемыми алгоритмами по эффективности.

Добавочный алгоритм оптимизации — алгоритм оптимизации, который не сравнивается по эффективности с основным алгоритмом и другими сравниваемыми алгоритмами по эффективности. Этот алгоритм является промежуточным, и в нем проверяется эффективность какой-нибудь настройки алгоритма. Например, в стандартном генетическом алгоритме есть три вида скрещивания: одноточечное, двухточечное и равномерное. А мы решили проверить трехточечное скрещивание. Для этого создает добавочный алгоритм, в котором есть только один вид скрещивания — трехточечным, и проводим полное тестирование алгоритма. И в сравнении с обычным алгоритмом можем оценить эффектность данного оператора. Если покажет эффективность, то уже можем создать сравниваемый алгоритм, который или уберет какой-то параметр или внесет трехточечное скрещивание на равноправных правах с другими видами скрещивания, или же, если на всех тестовых задачах трехточечное скрещивание покажет преимущество, то добавочный алгоритм станет сравниваемым алгоритмом. При этом отметим, что если просто добавим этот оператор в наравне с другими операторами, то нам не нужно будет пересчитывать весь алгоритм, так как просто добавим исследования из предыдущего исследования основного алгоритма.

Исследовательский алгоритм оптимизации — также алгоритм оптимизации, который не сравнивается по эффективности с основным алгоритмом и другими сравниваемыми алгорит-

мами по эффективности. Его особенность, что в этом алгоритме «вшито» множество разных настроек, эффективность которых мы не знаем. Мы проводим полное исследование данного алгоритма, убираем неэффективные настройки или комбинации настроек и формируем уже сравниваемый алгоритм оптимизации.

4 Постановка задачи оптимизации

Рассмотрим постановку задачи оптимизации в общем случае, решаемую алгоритмами оптимизации.

Необходимо найти:

$$\begin{aligned} \bar{x}_{max} &= \arg \max_{\bar{x} \in X} f(\bar{x}), \text{ где} \\ g_i(\bar{x}) &\leq 0, i = \overline{1, m_1}, \\ h_j(\bar{x}) &= 0, j = \overline{1, m_2}. \end{aligned} \quad (1)$$

Здесь:

X — множество всех возможных решений,

$f(\bar{x})$ — функционал, определенный на данном множестве, возвращающий действительное число из интервала $(-\infty; \infty)$,

m_1 — число ограничений в виде неравенств,

m_2 — число ограничений в виде равенств.

$\bar{x} \in X$ имеет вид

$$\bar{x} = (x_1; \dots; x_i; \dots; x_n)^T. \quad (2)$$

В случае, если $m_1 = 0$ и $m_2 = 0$, имеем задачу безусловной оптимизации. В противном случае — условной оптимизации.

В случае, если X — множество всех бинарных векторов длины n таких что $x_i \in \{0; 1\}$ ($i = \overline{1, n}$), то имеем задачу бинарной оптимизации (мощность поискового пространства X равна $\mu(X) = 2^n$). Если X — множество всех вещественных векторов длины n таких что $x_i \in \{Left_i; Right_i\}$ ($i = \overline{1, n}$), то имеем задачу вещественной оптимизации и т. д.

Будем предполагать в дальнейшем, что $f(\bar{x})$ может представлять собой многоэкстремальный функционал, и вычислению подлежат только значения $f(\bar{x})$ (нет возможности вычислить производные от функционала и т. д.).

Если требуется найти минимум функционала $f(\bar{x})$, то решаем задачу нахождения максимума функционала $-f(\bar{x})$.

Также обычно при решении задачи оптимизации нам дано число *CountOfFitness* — максимальное число вычислений целевой функции, которое нам дозволено использовать при запуске того или иного алгоритма оптимизации.

5 Стандартный генетический алгоритм

5.1 Стандартный генетический алгоритм для решения задач на бинарных строках

Тип алгоритма: основной алгоритм оптимизации.

Идентификатор: MHL_StandartBinaryGeneticAlgorithm.

Название: стандартный генетический алгоритм для решения задач на бинарных строках.

Подробное описание алгоритма представлено в данном проекте:

<https://github.com/Harrix/Standard-Genetic-Algorithm>.

Результат исследований алгоритма можно посмотреть тут:

<https://github.com/Harrix/HarrixDataOfOptimizationTesting>

В библиотеке HarrixMathLibrary данный алгоритм реализован в виде функции MHL_StandartBinaryGeneticAlgorithm. Библиотеку можно найти тут:

<https://github.com/Harrix/HarrixMathLibrary>

5.2 Стандартный генетический алгоритм для решения задач на вещественных строках

Тип алгоритма: основной алгоритм оптимизации.

Идентификатор: MHL_StandartRealGeneticAlgorithm.

Название: стандартный генетический алгоритм для решения задач на вещественных строках.

Подробное описание алгоритма представлено в данном проекте:

<https://github.com/Harrix/Standard-Genetic-Algorithm>.

Результат исследований алгоритма можно посмотреть тут:

<https://github.com/Harrix/HarrixDataOfOptimizationTesting>

В библиотеке HarrixMathLibrary данный алгоритм реализован в виде функции MHL_StandartBinaryGeneticAlgorithm. Библиотеку можно найти тут:

<https://github.com/Harrix/HarrixMathLibrary>

6 Модификации генетического алгоритма

6.1 Генетический алгоритм для решения задач на бинарных строках с изменяющимся соотношением числа поколений и размера популяции

Тип алгоритма: исследовательский алгоритм оптимизации.

Идентификатор: MHL_BinaryGeneticAlgorithmWDPOfNOFPS.

Название: генетический алгоритм для решения задач на бинарных строках с изменяющимся соотношением числа поколений и размера популяции.

Основан на стандартном генетическом алгоритме на бинарных строках: <https://github.com/Harrix/Standard-Genetic-Algorithm>.

Отличается от стандартного генетического алгоритма, тем, что размер популяции и число поколений рассчитывается из числа вычислений целевой функции не как одинаковые величины (извлечение квадратного корня), а через некоторое соотношение.

Число поколений определяется по формуле:

$$NumberOfGenerations = \text{int} (CountOfFitness^{Proportion}) . \quad (3)$$

Число поколений, соответственно, определяется по формуле:

$$PopulationSize = \text{int} \left(\frac{CountOfFitness}{NumberOfGenerations} \right) . \quad (4)$$

Тут *CountOfFitness* — максимальное число вычислений целевой функции, а *Proportion* — **новый** параметр в алгоритме, который обозначает «долю» числа поколений от общего числа вычислений целевой функции.

Proportion может принимать значения в интервале $[0; 1]$, а именно:

$$Proportion \in \{0; 0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1\} . \quad (5)$$

То есть *Proportion* может принимать 11 значений.

По сравнению с стандартным генетическим алгоритмом число вариантов настроек алгоритма увеличивается в 11 раз и равно **594**.

Чем меньше *Proportion*, тем меньше будет число поколений.

При *Proportion* = 0.5 получим обычный стандартный генетический алгоритм. Число поколений будет равно $\sqrt{CountOfFitness}$ (без учета получения целой части числа).

При *Proportion* = 0 число поколений будет равно 1.

При *Proportion* = 1 число поколений будет равно *CountOfFitness*.

Результат исследований алгоритма можно посмотреть тут:

<https://github.com/Harrix/HarrixDataOfOptimizationTesting>

В библиотеке HarrixMathLibrary данный алгоритм реализован в виде функции MHL_BinaryGeneticAlgorithmWDPOfNOfGPS. Библиотеку можно найти тут:

<https://github.com/Harrix/HarrixMathLibrary>

6.2 Генетический алгоритм для решения задач на вещественных строках с изменяющимся соотношением числа поколений и размера популяции

Тип алгоритма: исследовательский алгоритм оптимизации.

Идентификатор: MHL_RealGeneticAlgorithmWDPOfNOfGPS.

Название: генетический алгоритм для решения задач на вещественных строках с изменяющимся соотношением числа поколений и размера популяции.

Основан на стандартном генетическом алгоритме на вещественных строках: <https://github.com/Harrix/Standard-Genetic-Algorithm>.

Отличается от стандартного генетического алгоритма, тем, что размер популяции и число поколений рассчитывается из числа вычислений целевой функции не как одинаковые величины (извлечение квадратного корня), а через некоторое соотношение.

Число поколений определяется по формуле:

$$NumberOfGenerations = \text{int} \left(CountOfFitness^{Proportion} \right). \quad (6)$$

Число поколений, соответственно, определяется по формуле:

$$PopulationSize = \text{int} \left(\frac{CountOfFitness}{NumberOfGenerations} \right). \quad (7)$$

Тут *CountOfFitness* — максимальное число вычислений целевой функции, а *Proportion* — **новый** параметр в алгоритме, который обозначает «долю» числа поколений от общего числа вычислений целевой функции.

Proportion может принимать значения в интервале $[0; 1]$, а именно:

$$Proportion \in \{0; 0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1\}. \quad (8)$$

То есть *Proportion* может принимать 11 значений.

По сравнению с стандартным генетическим алгоритмом число вариантов настроек алгоритма увеличивается в 11 раз и равно **1188**.

Чем меньше *Proportion*, тем меньше будет число поколений.

При *Proportion* = 0.5 получим обычный стандартный генетический алгоритм. Число поколений будет равно $\sqrt{CountOfFitness}$ (без учета получения целой части числа).

При *Proportion* = 0 число поколений будет равно 1.

При *Proportion* = 1 число поколений будет равно *CountOfFitness*.

Результат исследований алгоритма можно посмотреть тут:

<https://github.com/Harrix/HarrixDataOfOptimizationTesting>

В библиотеке `HarrixMathLibrary` данный алгоритм реализован в виде функции `MHL_RealGeneticAlgorithmWDPOfNOOfGPS`. Библиотеку можно найти тут:

<https://github.com/Harrix/HarrixMathLibrary>

6.3 Генетический алгоритм для решения задач на бинарных строках с турнирной селекцией, где размер турнира изменяется от 2 до размера популяции

Тип алгоритма: добавочный алгоритм оптимизации.

Идентификатор: `MHL_BinaryGeneticAlgorithmWDTs`.

Название: генетический алгоритм для решения задач на бинарных строках с турнирной селекцией, где размер турнира изменяется от 2 до размера популяции.

Основан на стандартном генетическом алгоритме на бинарных строках:
<https://github.com/Harrix/Standard-Genetic-Algorithm>.

Отличается от стандартного генетического алгоритма, тем, что присутствует только турнирная селекция (пропорциональная и ранговая отсутствуют), но размер турнира может изменяться.

Так как основан на стандартном генетическом алгоритме, то размер популяции вычисляется, как квадратный корень из максимального числа вычислений целевой функции. Поэтому размер турнира *SizeOfTournament* может теоретически изменяться в пределах:

$$SizeOfTournament = 1, \text{int} \left(\sqrt{CountOfFitness} \right). \quad (9)$$

Тут *CountOfFitness* — максимальное число вычислений целевой функции.

SizeOfTournament может принимать следующие значения в данном алгоритме:

$$SizeOfTournament \in \left\{ \begin{array}{c} 2 \\ 3 \\ 4 \\ 5 \\ 1/3 \text{ от популяции} \\ 1/2 \text{ от популяции} \\ 2/3 \text{ от популяции} \\ \text{Вся популяция} \end{array} \right\} \quad (10)$$

Если записывать строго, то получится следующее множество:

$$SizeOfTournament \in \left\{ \begin{array}{c} 2 \\ 3 \\ 4 \\ 5 \\ int\left(\frac{1}{3} \cdot \sqrt{CountOfFitness}\right) \\ int\left(\frac{1}{2} \cdot \sqrt{CountOfFitness}\right) \\ int\left(\frac{2}{3} \cdot \sqrt{CountOfFitness}\right) \\ int\left(\sqrt{CountOfFitness}\right) \end{array} \right\} \quad (11)$$

То есть *SizeOfTournament* может принимать 8 значений.

Число вариантов настроек алгоритма равно **144**.

При *SizeOfTournament* = 2 получим обычный стандартный генетический алгоритм (без ранговой и пропорциональной селекции).

Результат исследований алгоритма можно посмотреть тут:

<https://github.com/Harrix/HarrixDataOfOptimizationTesting>

В библиотеке *HarrixMathLibrary* данный алгоритм реализован в виде функции *MHL_BinaryGeneticAlgorithmWDTs*. Библиотеку можно найти тут:

<https://github.com/Harrix/HarrixMathLibrary>

6.4 Генетический алгоритм для решения задач на вещественных строках с турнирной селекцией, где размер турнира изменяется от 2 до размера популяции

Тип алгоритма: добавочный алгоритм оптимизации.

Идентификатор: *MHL_RealGeneticAlgorithmWDTs*.

Название: генетический алгоритм для решения задач на вещественных строках с турнирной селекцией, где размер турнира изменяется от 2 до размера популяции.

Основан на стандартном генетическом алгоритме на вещественных строках:
<https://github.com/Harrix/Standard-Genetic-Algorithm>.

Отличается от стандартного генетического алгоритма, тем, что присутствует только турнирная селекция (пропорциональная и ранговая отсутствуют), но размер турнира может изменяться.

Так как основан на стандартном генетическом алгоритме, то размер популяции вычисляется, как квадратный корень из максимального числа вычислений целевой функции. Поэтому размер турнира *SizeOfTournament* может теоретически изменяться в пределах:

$$SizeOfTournament = 1, int\left(\sqrt{CountOfFitness}\right). \quad (12)$$

Тут *CountOfFitness* — максимальное число вычислений целевой функции.

SizeOfTournament может принимать следующие значения в данном алгоритме:

$$SizeOfTournament \in \left\{ \begin{array}{c} 2 \\ 3 \\ 4 \\ 5 \\ 1/3 \text{ от популяции} \\ 1/2 \text{ от популяции} \\ 2/3 \text{ от популяции} \\ \text{Вся популяция} \end{array} \right\} \quad (13)$$

Если записывать строго, то получится следующее множество:

$$SizeOfTournament \in \left\{ \begin{array}{c} 2 \\ 3 \\ 4 \\ 5 \\ int\left(\frac{1}{3} \cdot \sqrt{CountOfFitness}\right) \\ int\left(\frac{1}{2} \cdot \sqrt{CountOfFitness}\right) \\ int\left(\frac{2}{3} \cdot \sqrt{CountOfFitness}\right) \\ int\left(\sqrt{CountOfFitness}\right) \end{array} \right\} \quad (14)$$

То есть *SizeOfTournament* может принимать 8 значений.

Число вариантов настроек алгоритма равно **288**.

При *SizeOfTournament* = 2 получим обычный стандартный генетический алгоритм (без ранговой и пропорциональной селекции).

Результат исследований алгоритма можно посмотреть тут:

<https://github.com/Harrix/HarrixDataOfOptimizationTesting>

В библиотеке *HarrixMathLibrary* данный алгоритм реализован в виде функции *MHL_RealGeneticAlgorithmWDTs*. Библиотеку можно найти тут:

<https://github.com/Harrix/HarrixMathLibrary>

6.5 Генетический алгоритм для решения задач на бинарных строках, в котором есть только два вида скрещивания: одноточечное и двухточечное скрещивание с возможностью полного копирования одного из родителей

Тип алгоритма: исследовательский алгоритм оптимизации.

Идентификатор: *MHL_BinaryGeneticAlgorithmWCC*.

Название: генетический алгоритм для решения задач на бинарных строках, в котором есть только два вида скрещивания: одноточечное и двухточечное скрещивание с возможностью полного копирования одного из родителей.

Основан на стандартном генетическом алгоритме на бинарных строках:
<https://github.com/Harrix/Standard-Genetic-Algorithm>.

Отличается от стандартного генетического алгоритма тем, что есть только два вида скрещивания: одноточечное и двухточечное скрещивание с возможностью полного копирования одного из родителей. Равномерное скрещивание отсутствует. То есть данным алгоритмом проверяем: есть ли разница в эффективности алгоритма, если точки разрыва при скрещивании делать и по краям родителей, а не только внутри хромосомы.

В качестве операторов-заменителей используются:

- SinglepointCrossoverWithCopying — одноточечное скрещивание с возможностью полного копирования одного из родителей;
- TwopointCrossoverWithCopying — двухточечное скрещивание с возможностью полного копирования одного из родителей;

Подробно прочитать о этих двух операторах с формулами и примерами можно тут:

<https://github.com/Harrix/HarrixSetOfOperatorsAlgorithms>

Число вариантов настроек алгоритма равно **36**.

Результат исследований алгоритма можно посмотреть тут:

<https://github.com/Harrix/HarrixDataOfOptimizationTesting>

В библиотеке HarrixMathLibrary данный алгоритм реализован в виде функции MHL_BinaryGeneticAlgorithmWCC. Библиотеку можно найти тут:

<https://github.com/Harrix/HarrixMathLibrary>

6.6 Генетический алгоритм для решения задач на вещественных строках, в котором есть только два вида скрещивания: одноточечное и двухточечное скрещивание с возможностью полного копирования одного из родителей

Тип алгоритма: исследовательский алгоритм оптимизации.

Идентификатор: MHL_RealGeneticAlgorithmWCC.

Название: генетический алгоритм для решения задач на вещественных строках, в котором есть только два вида скрещивания: одноточечное и двухточечное скрещивание с возможностью полного копирования одного из родителей.

Основан на стандартном генетическом алгоритме на бинарных строках:
<https://github.com/Harrix/Standard-Genetic-Algorithm>.

Отличается от стандартного генетического алгоритма тем, что есть только два вида скрещивания: одноточечное и двухточечное скрещивание с возможностью полного копирования одного из родителей. Равномерное скрещивание отсутствует. То есть данным алгоритмом проверяем: есть ли разница в эффективности алгоритма, если точки разрыва при скрещивании делать и по краям родителей, а не только внутри хромосомы.

В качестве операторов-заменителей используются:

- SinglepointCrossoverWithCopying — однотоочечное скрещивание с возможностью полного копирования одного из родителей;
- TwopointCrossoverWithCopying — двухточечное скрещивание с возможностью полного копирования одного из родителей;

Подробно прочитать о этих двух операторах с формулами и примерами можно тут:

<https://github.com/Harrix/HarrixSetOfOperatorsAlgorithms>

Число вариантов настроек алгоритма равно **72**.

Результат исследований алгоритма можно посмотреть тут:

<https://github.com/Harrix/HarrixDataOfOptimizationTesting>

В библиотеке HarrixMathLibrary данный алгоритм реализован в виде функции MHL_RealGeneticAlgorithmWCC. Библиотеку можно найти тут:

<https://github.com/Harrix/HarrixMathLibrary>

6.7 Генетический алгоритм для решения задач на бинарных строках с турнирной селекцией с возвращением, где размер турнира изменяется от 2 до размера популяции

Тип алгоритма: добавочный алгоритм оптимизации.

Идентификатор: MHL_BinaryGeneticAlgorithmTournamentSelectionWithReturn.

Название: генетический алгоритм для решения задач на бинарных строках с турнирной селекцией с возвращением, где размер турнира изменяется от 2 до размера популяции.

Основан на стандартном генетическом алгоритме на бинарных строках: <https://github.com/Harrix/Standard-Genetic-Algorithm>.

Отличается от стандартного генетического алгоритма, тем, что присутствует только турнирная селекция, но и она измененная, и размер турнира может изменяться.

Подробное описание турнирной селекции с возвращением можно прочитать тут:

<https://github.com/Harrix/HarrixSetOfOperatorsAlgorithms>

Так как основан на стандартном генетическом алгоритме, то размер популяции вычисляется, как квадратный корень из максимального числа вычислений целевой функции. Поэтому размер турнира *SizeOfTournament* может теоретически изменяться в пределах:

$$SizeOfTournament = 1, int \left(\sqrt{CountOfFitness} \right). \quad (15)$$

Тут *CountOfFitness* — максимальное число вычислений целевой функции.

SizeOfTournament может принимать следующие значения в данном алгоритме:

$$SizeOfTournament \in \left\{ \begin{array}{c} 2 \\ 3 \\ 4 \\ 5 \\ 1/3 \text{ от популяции} \\ 1/2 \text{ от популяции} \\ 2/3 \text{ от популяции} \\ \text{Вся популяция} \end{array} \right\} \quad (16)$$

Если записывать строго, то получится следующее множество:

$$SizeOfTournament \in \left\{ \begin{array}{c} 2 \\ 3 \\ 4 \\ 5 \\ int\left(\frac{1}{3} \cdot \sqrt{CountOfFitness}\right) \\ int\left(\frac{1}{2} \cdot \sqrt{CountOfFitness}\right) \\ int\left(\frac{2}{3} \cdot \sqrt{CountOfFitness}\right) \\ int\left(\sqrt{CountOfFitness}\right) \end{array} \right\} \quad (17)$$

То есть *SizeOfTournament* может принимать 8 значений.

Число вариантов настроек алгоритма равно **144**.

Результат исследований алгоритма можно посмотреть тут:

<https://github.com/Harrix/HarrixDataOfOptimizationTesting>

В библиотеке *HarrixMathLibrary* данный алгоритм реализован в виде функции *MHL_BinaryGeneticAlgorithmTournamentSelectionWithReturn*. Библиотеку можно найти тут:

<https://github.com/Harrix/HarrixMathLibrary>

6.8 Генетический алгоритм для решения задач на вещественных строках с турнирной селекцией с возвращением, где размер турнира изменяется от 2 до размера популяции

Тип алгоритма: добавочный алгоритм оптимизации.

Идентификатор: *MHL_RealGeneticAlgorithmTournamentSelectionWithReturn*.

Название: генетический алгоритм для решения задач на вещественных строках с турнирной селекцией с возвращением, где размер турнира изменяется от 2 до размера популяции.

Основан на стандартном генетическом алгоритме на вещественных строках:
<https://github.com/Harrix/Standard-Genetic-Algorithm>.

Отличается от стандартного генетического алгоритма, тем, что присутствует только турнирная селекция, но и она измененная, и размер турнира может изменяться.

Подробное описание турнирной селекции с возвращением можно прочитать тут:

<https://github.com/Harrix/HarrixSetOfOperatorsAlgorithms>

Так как основан на стандартном генетическом алгоритме, то размер популяции вычисляется, как квадратный корень из максимального числа вычислений целевой функции. Поэтому размер турнира *SizeOfTournament* может теоретически изменяться в пределах:

$$SizeOfTournament = 1, \text{int} \left(\sqrt{CountOfFitness} \right). \quad (18)$$

Тут *CountOfFitness* — максимальное число вычислений целевой функции.

SizeOfTournament может принимать следующие значения в данном алгоритме:

$$SizeOfTournament \in \left\{ \begin{array}{c} 2 \\ 3 \\ 4 \\ 5 \\ 1/3 \text{ от популяции} \\ 1/2 \text{ от популяции} \\ 2/3 \text{ от популяции} \\ \text{Вся популяция} \end{array} \right\} \quad (19)$$

Если записывать строго, то получится следующее множество:

$$SizeOfTournament \in \left\{ \begin{array}{c} 2 \\ 3 \\ 4 \\ 5 \\ \text{int} \left(\frac{1}{3} \cdot \sqrt{CountOfFitness} \right) \\ \text{int} \left(\frac{1}{2} \cdot \sqrt{CountOfFitness} \right) \\ \text{int} \left(\frac{2}{3} \cdot \sqrt{CountOfFitness} \right) \\ \text{int} \left(\sqrt{CountOfFitness} \right) \end{array} \right\} \quad (20)$$

То есть *SizeOfTournament* может принимать 8 значений.

Число вариантов настроек алгоритма равно **288**.

Результат исследований алгоритма можно посмотреть тут:

<https://github.com/Harrix/HarrixDataOfOptimizationTesting>

В библиотеке *HarrixMathLibrary* данный алгоритм реализован в виде функции *MHL_RealGeneticAlgorithmTournamentSelectionWithReturn*. Библиотеку можно найти тут:

<https://github.com/Harrix/HarrixMathLibrary>

7 Алгоритмы Монте-Карло

7.1 Метод Монте-Карло (Monte-Carlo) для решения задач на бинарных строках

Тип алгоритма: сравниваемый алгоритм оптимизации.

Идентификатор: MHL_BinaryMonteCarloAlgorithm.

Название: Метод Монте-Карло (Monte-Carlo) для решения задач на бинарных строках.

Наиболее простой алгоритм. Случайно генерируется по равномерному закону распределения *CountOfFitness* решений из допустимой области. Из них выбирается лучшее.

При сравнении с работой, например, стандартного генетического алгоритма *CountOfFitness* (число вычислений целевой функции) выбирается таким же, что и у стандартного генетического алгоритма.

Результат исследований алгоритма можно посмотреть тут:

<https://github.com/Harrix/HarrixDataOfOptimizationTesting>

В библиотеке HarrixMathLibrary данный алгоритм реализован в виде функции MHL_BinaryMonteCarloAlgorithm. Библиотеку можно найти тут:

<https://github.com/Harrix/HarrixMathLibrary>

7.2 Метод Монте-Карло (Monte-Carlo) для решения задач на вещественных строках

Тип алгоритма: сравниваемый алгоритм оптимизации.

Идентификатор: MHL_RealMonteCarloAlgorithm.

Название: Метод Монте-Карло (Monte-Carlo) для решения задач на вещественных строках.

Наиболее простой алгоритм. Случайно генерируется по равномерному закону распределения *CountOfFitness* решений из допустимой области. Из них выбирается лучшее.

При сравнении с работой, например, стандартного генетического алгоритма *CountOfFitness* (число вычислений целевой функции) выбирается таким же, что и у стандартного генетического алгоритма.

Результат исследований алгоритма можно посмотреть тут:

<https://github.com/Harrix/HarrixDataOfOptimizationTesting>

В библиотеке HarrixMathLibrary данный алгоритм реализован в виде функции MHL_RealMonteCarloAlgorithm. Библиотеку можно найти тут:

<https://github.com/Harrix/HarrixMathLibrary>