Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Кафедра математического обеспечения и суперкомпьютерных технологий**

**Отчет по учебной практике**

**«Параллельные вычисления в задачах глобальной оптимизации»**

**Выполнил**:

Студент группы 381503-3

Лалыкин Олег Вадимович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись

**Научный руководитель**:

Доцент кафедры МОСТ, к.т.н.

Сысоев Александр Владимирович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись

Нижний Новгород

2018

**Содержание**

[**Введение** 3](#_Toc513469835)

[**Постановка задачи** 4](#_Toc513469836)

[**Описание алгоритмов** 5](#_Toc513469837)

[**Алгоритм глобального поиска** 5](#_Toc513469838)

[**Адаптация к двумерной задаче** 6](#_Toc513469839)

[**Развертки Пеано** 7](#_Toc513469840)

[**Схемы распараллеливания одномерной задачи** 9](#_Toc513469841)

[**Распараллеливание по области поиска** 9](#_Toc513469842)

[**Распараллеливание по характеристикам** 9](#_Toc513469843)

[**Программная реализация** 10](#_Toc513469844)

[**Структура программы** 10](#_Toc513469845)

[**Описание структур данных** 11](#_Toc513469846)

[**Описание методов** 11](#_Toc513469847)

[**Результаты** 13](#_Toc513469848)

[**Заключение** 14](#_Toc513469849)

[**Перспективы развития** 15](#_Toc513469850)

[**Литература** 16](#_Toc513469851)

[**Приложение** 17](#_Toc513469852)

# **Введение**

Большое количество постановок технических или научных проблем можно сформулировать как задачу глобальной оптимизации. В связи с чем, алгоритмы глобальной оптимизации находят широкое применение в разнообразных областях науки и техники, везде, где необходимо получить наилучший результат целевой функции, оценивающей качество принимаемого решения.

К настоящему времени разработано большое количество алгоритмов и методов решения задачи многоэкстремальной оптимизации. Сегодня разработка методов глобальной оптимизации стимулируется развитием электронно-вычислительных средств и во многом связано с доступностью параллельных компьютерных систем высокой производительности. Появилось большое количество разнообразных подходов к распараллеливанию алгоритмов глобальной оптимизации.

Для одномерных функций существуют эффективные стратегии многоэкстремальной оптимизации. Лидером методов одномерной глобальной оптимизации является информационный алгоритм Р. Г. Стронгина. Решение многомерной задачи может быть сведено к решению одномерной задачи путем редукции размерности с использованием кривых Пеано.

# **Постановка задачи**

1. Изучение характеристически представимых методов решения одномерных задач глобальной оптимизации.
2. Изучение типовых схем распараллеливания характеристически представимых методов решения одномерных задач глобальной оптимизации.
3. Реализация некоторых типовых схем распараллеливания для решения одномерных задач глобальной оптимизации.
4. Изучение методов и типовых схем распараллеливания решения многомерных задач глобальной оптимизации.
5. Реализация решения многомерной задачи глобальной оптимизации.

В рамках данной работы требуется реализовать:

1. Последовательный и параллельный алгоритм одномерной задачи глобальной оптимизации Р. Г. Стронгина.
2. Последовательный и параллельный алгоритм двумерной задачи глобальной оптимизации с использованием схемы редукции размерности кривыми Пеано.
3. Тестирование для проверки корректности, используя функции Гришагина.

# **Описание алгоритмов**

## **Алгоритм глобального поиска[[1]](#footnote-1)**

Рассмотрим одномерную задачу минимизации функции на отрезке.

Согласно алгоритму, два первых испытания проводятся на кон­цах отрезка, то есть

,, далее вычисляются значения функции , , и количество проведенных испытаний к полагается равным 2.

Пусть проведено испытаний и получена информация

Для выбора точки нового испытания необходимо выполнить следующие действия.

1. Перенумеровать нижним индексом (начиная с нулевого зна­чения) точки

, , в порядке возрастания:

1. Полагая, вычислить величины:

и

где является заданным параметром метода.

1. Для каждого интервала вычислить характеристику
2. Найти интервал которому соответствует максимальная характеристика
3. Провести новое испытание в точке:
4. Вычислить значение и увеличить номер шага на единицу

.

Операции пунктов 1­­­-5 описывают решающее правило АГП. Правило остановки задается в форме:

где – заданная точность поиска (по координате).

В качестве оценки экстремума выбирается пара:.

- минимальное вычисленное значение функции:.

- координата этого значения:.

## **Адаптация к двумерной задаче [[2]](#footnote-2)**

Описанный алгоритм глобального поиска одномерной функции можно использовать для решения многомерной задачи.

Для этого используется способ редукции размерности отображением многомерной области поиска на одномерный интервал с помощью кривых Пеано.

Пусть дана задача глобальной оптимизации вида:

ϕ∗=ϕ(*y*∗)=min{ϕ(*y*): *y*∈*D*, *g j* (*y*)≤0, 1≤*j*≤*m*},

где *D*={*y*∈*R N* : *a i* ≤*y i* ≤*b i* , 1≤*i*≤*N*} - область в N-мерном пространстве

Целевая функция ϕ(*y*) удовлетворяют условию Липшица с соответствующими константами.

Используя кривые типа развертки Пеано *y*(*x*), однозначно отображающие отрезок [0,1] на *N*-мерный гиперкуб *D*

*D*={*y*∈*R N* : −2− 1 ≤*y i* ≤2− 1 , 1≤*i*≤*N*}={*y*(*x*): 0≤*x*≤1},

исходную задачу можно редуцировать к следующей одномерной задаче:

ϕ(*y*(*x*∗))=min{ϕ(*y*(*x*)): *x*∈[0,1], *g j* (*y*(*x*))≤0, 1≤*j*≤*m*}.

Пусть пространство в которой определена целевая функция ϕ(*y1*, *y2*) является двумерным (N=2), тогда область на которую отображается отрезок [0,1] имеет два ограничения : a≤ *y1*≤*b* ,c≤ *y2*≤*d*.

Таким образом, одномерный АГП адаптируется для решения двумерной задачи.

Алгоритм решения имеет вид:

Решается одномерная задача на единичном отрезке [0,1], где целевая функция ϕ(*y1*, *y2*) определена на двумерной области с заданными ограничениями a≤ *y1*≤*b* ,c≤ *y2*≤*d*. Проводится итерация в соответствии с АГП, где очередная точка испытания *x*∈[0,1] отображается на двумерную область a≤ *y1*≤*b* ,c≤ *y2*≤*d*. В результате отображения берутся соответствующие координаты (*y1*, *y2*) из двумерной области и используются для отыскания значения целевой функции ϕ(*y1*, *y2*) на текущей итерации. Полученное значение целевой двумерной функции используется для определения очередной точки последующего испытания.

## **Развертки Пеано [[3]](#footnote-3)**

Само построение развертки, отображающей единичный отрезок [0,1] на

*N*-мерный гиперкуб *D*

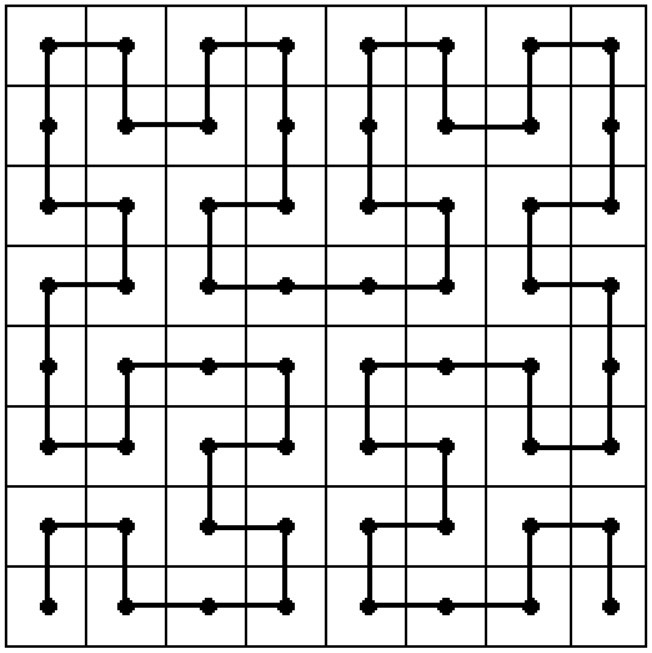
*D*={*y*∈*R N* : −2− 1 ≤*y i* ≤2− 1 , 1≤*i*≤*N*}={*y*(*x*): 0≤*x*≤1}

cводится к следующим операциям:

1. Разбиение гиперкуба *D* с длиной ребра равной 1 на 2 N гиперкубов первого разбиения с длиной ребра равной ½.
2. Далее каждый гиперкуб первого разбиения снова делится на 2 N следующего. Процесс деления продолжается пока гиперкубы не станут нужного разбиения.
3. Теперь аналогично поступают с единичным отрезком [0,1] – его так же делят на 2 N равных и продолжают делить каждую часть пока не достигнут нужного уровня разбиения.
4. Фиксируют, что точка *y(x)*, лежащая в гиперкубе уровня разбиения *n*, однозначно соответствует точке *x,* лежащей на отрезке того же разбиения *n.*
5. Фиксируют, что номера центров гиперкубов те же, что и номера соответствующих отрезков (на одном уровне разбиения).
6. Получают, что отображение отрезка [0,1] в гиперкуб *D* означает следующее: образ любого подынтервала из [0,1] является линейным отрезком, соединяющим центры-узлы гиперкуба.

В итоге строится кусочно-линейная кривая, которая соединяет центры-узлы гиперкуба. Такая кривая является численным приближением к кривой Пеано с заданной точностью, которая зависит от плотности развертки.

Пример кусочно-линейной развертки для двумерной функции:



**Рисунок 1 Развертка размерности 2 и плотности 3**

# **Схемы распараллеливания одномерной задачи**

## **Распараллеливание по области поиска**

Алгоритм заключается в распределении исходного отрезка на независимые участки по потокам и выполнении последовательного АГП в каждом.

Основные действия схемы:

1. Исходный отрезок разбивается на отдельные участки одинаковой длины в количестве равном числу выделенных потоков.
2. Каждый участок присваивается соответствующему потоку.
3. Каждый поток независимо от остальных выполняет последовательный АГП на своём участке.
4. Когда критерий остановки алгоритма выполнится во всех потоках, тогда происходит запись результатов.
5. Среди полученных результатов выбирается решение с наименьшим полученным значением целевой функции, он и считается решением всей задачи.

## **Распараллеливание по характеристикам**

Алгоритм заключается в параллельном нахождении сразу нескольких характеристик , из которых в выбирается максимальная.

Основные действия схемы:

1. Параллельный поиск оценок константы Липшица

, где

1. Параллельно вычисляются значения характеристик и для каждого потока запоминается индекс и значение максимальной характеристики
2. Проводится новое испытание в точке , где – индекс максимальной характеристики

# **Программная реализация**

## **Структура программы**

Evolvent.h – содержит заимствованный метод GetImage, реализованный в проекте Globalizer.

Grishagin\_function.h – содержит методы, используемые для тестирования алгоритма на двумерных функциях Гришагина.

Search.h – содержит все реализованные в данной работе алгоритмы.

Описание класса “Search”

private:

double left; // левая граница одномерной задачи

double right; // правая граница одномерной задачи

double\* Left; // ограничение области слева двумерной задачи

double\* Right; // ограничение области справа двумерной задачи

double r; // параметр метода

int procs;//число потоков

public:

Search(const double \_left, const double \_right, const double \_r, const int \_procs)//конструктор для одномерной задачи

Search(const double \*\_left, const double \*\_right, const double \_r, const int \_procs)//конструктор для двумерной задачи

double Func(const double &\_x)//Одномерные функции

double Func(const double \*\_y, vagrish::GrishaginFunction \*func) //Двумерные функции Гришагина

PointerOneDim Serial\_searchMin(const double \_left, const double \_right, const int \_N\_max, const double \_Eps);//Последовательный поиск одномерной задачи

PointerOneDim Simple\_Par\_searchMin(const double \_left, const double \_right, const int \_N\_max, const double \_Eps, const int \_threads);//ПП по отрезкам одномерной задачи

PointerOneDim Ch\_SearchMin(const double \_Epsilon, const int \_Steps, const int threads);//ПП по характеристикам одномерной задачи

PointerTwoDim Two\_Dim\_Search(const double \_Epsilon, const int \_Steps, const int threads, vagrish::GrishaginFunction \*func);//Решатель двумерной задачи

double R(const double &\_m\_small, const double &\_z\_curr, const double &\_z\_prev, const double &\_x\_curr, const double &\_x\_prev)//Характеристика

double M(const double &\_z\_curr, const double &\_z\_prev, const double &\_x\_curr, const double &\_x\_prev)//Параметр для оценки константы Липшица

double New\_x(double &x\_right, double &x\_left, double &z\_right, double &z\_left, double &m\_small)//Новая точка испытания

void Lipschitz(double &M\_big, double &m\_small)//Оценка константы Липшица

## **Описание структур данных**

double left - левая граница отрезка для одномерной задачи

double right - правая граница отрезка для одномерной задачи

double\* Left - левые ограничения области для двумерной задачи

double\* Right - правые ограничения области для двумерной задачи

double r - параметр метода

int procs - количество потоков

struct PointerOneDim – Структура для сбора результатов в одномерной задаче.

struct PointerTwoDim– Структура для сбора результатов в двумерной задаче.

struct GroupOneDim – Структура для организации параллельной реализации одномерной задачи. Распараллеливание по характеристикам.

struct borders – Структура для организации параллельной реализации одномерной задачи. Распараллеливание по области поиска.

struct GroupTwoDim – Структура для организации решения двумерной задачи.

## **Описание методов**

Search(const double \_left, const double \_right, const double \_r, const int \_procs) – конструктор для одномерной задачи, инициализирует указанные входные параметры.

Search(const double \*\_left, const double \*\_right, const double \_r, const int \_procs) – конструктор для двумерной задачи, инициализирует указанные входные параметры.

double Func(const double &\_x) – Одномерные целевые функции, зависящие от входного параметра \_x

double Func(const double \*\_y, vagrish::GrishaginFunction \*func) – Двумерные целевые функции Гришагина, зависящие от входного параметра \*\_y. Экземпляр \*func передается для вызова соответствующего метода подсчёта значения целевой сгенерированной функции Гришагина.

PointerOneDim Serial\_searchMin(const double \_left, const double \_right, const int \_N\_max, const double \_Eps) – Последовательное решение одномерной задачи с заданными ограничениями.

PointerOneDim Simple\_Par\_searchMin(const double \_left, const double \_right, const int \_N\_max, const double \_Eps, const int \_threads) – Параллельное решение одномерной задачи с заданными ограничениями. Распараллеливание по области поиска.

PointerOneDim Ch\_SearchMin(const double \_Epsilon, const int \_Steps, const int threads) – Параллельное решение одномерной задачи с заданными ограничениями. Распараллеливание по характеристикам.

PointerTwoDim Two\_Dim\_Search(const double \_Epsilon, const int \_Steps, const int threads, vagrish::GrishaginFunction \*func) – Решение двумерной задачи с заданными ограничениями для указанной функции Гришагина.

double R(const double &\_m\_small, const double &\_z\_curr, const double &\_z\_prev, const double &\_x\_curr, const double &\_x\_prev) – Подсчёт характеристики.

double M(const double &\_z\_curr, const double &\_z\_prev, const double &\_x\_curr, const double &\_x\_prev) – Подсчёт параметра М для оценки константы Липшица.

double New\_x(double &x\_right, double &x\_left, double &z\_right, double &z\_left, double &m\_small) – Подсчёт следующей точки испытания.

void Lipschitz(double &M\_big, double &m\_small) – Подсчёт оценки константы Липшица.

# **Результаты**

## **Тестирование алгоритма решения одномерной задачи**

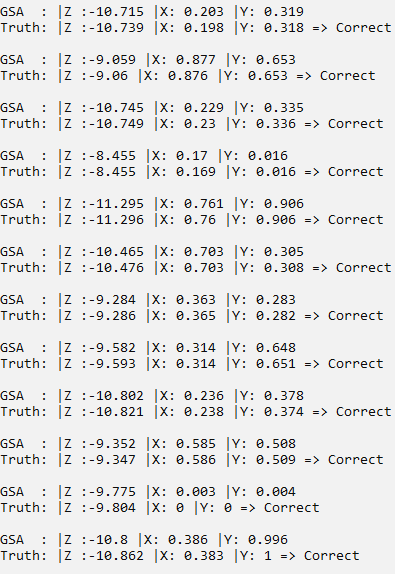
## **Тестирование алгоритма решения двумерной задачи**

Тестирование метода для функций Гришагина заключается в следующих действиях:

1. Выбор номера функции для генератора,
2. Получение входных параметров для функции – линейные ограничения на аргументы,
3. Проведение поиска глобального минимума для сгенерированной функции,
4. Сравнение полученного результата с известным-истинным,
5. Вынос результата – если разность между решениями не превышает заданную допустимую погрешность, то эксперимент успешен, иначе – провал.

Тест, проведенный для 100 функций Гришагина показал 100% правильность при следующих входных параметрах:

* Плотность развертки m = 20,
* Параметр метода r = 1.5,
* Критерий остановки по точности eps = 0.00001,
* Критерий остановки по числу итераций n = 1000,
* Число потоков - 4, распараллеливание по характеристикам,
* Условие корректности – погрешность не выше 0,1.

Пример тестирования для допустимой погрешности 0,01

**Рисунок 2Результаты теста из 12 функций**

# **Заключение**

В результате выполнения данной работы был разработан программный комплекс, реализующий поиск оптимума для одномерной и двумерной задач.

Проведя множество экспериментов, выполнив сравнение с заявленным оптимумом и найденным, можно судить о корректности реализованных алгоритмов.

# **Перспективы развития**

# **Литература**

1. Р. Г. Стронгин, В. П. Гергель, В. А. Гришагин, К. А. Баркалов **«**Параллельные вычисления в задачах глобальной оптимизации**»**
2. Гергель В.П., Стронгин Р.Г Основы параллельных вычислений для многопроцессорных вычислительных систем Н. Новгород: Изд-во ННГУ, 2001

# **Приложение**

1. Источник – литература [1] [↑](#footnote-ref-1)
2. Источник – литература [1] [↑](#footnote-ref-2)
3. Источник – литература [1] [↑](#footnote-ref-3)