Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего

образования «Нижегородский государственный университет им. Н.И. Лобачевского»

Институт Информационных Технологий, Математики и Механики

Лабораторная работа

«Алгоритм глобального поиска (Стронгина) для одномерных задач оптимизации. Распараллеливание путем разделения области поиска»

Выполнила:

студентка группы 381506-3

Федорова О.П.

Нижний Новгород

2018

Оглавление

[Введение 3](#_Toc514451825)

[Постановка задачи 4](#_Toc514451826)

[Вспомогательные элементы 5](#_Toc514451827)

[1. Checker 5](#_Toc514451828)

[2. Generator 5](#_Toc514451829)

[Метод решения 6](#_Toc514451830)

[Описание реализации последовательного алгоритма 8](#_Toc514451831)

[Описание реализации параллельного алгоритма 9](#_Toc514451832)

[Подтверждение корректности 10](#_Toc514451833)

[Результаты экспериментов по оценке масштабируемости 14](#_Toc514451834)

[Вывод 15](#_Toc514451835)

# Введение

Стандартная математическая задача оптимизации формулируется таким образом. Среди элементов χ, образующих множества Χ, найти такой элемент χ\*, который доставляет минимальное значение f(χ\*) заданной функции f(χ). Для того, чтобы корректно поставить задачу оптимизации, необходимо задать:

1. Допустимое множество — множество  {\displaystyle \mathbb {X} =\{{\vec {x}}|\;g\_{i}({\vec {x}})\leq 0,\;i=1,\ldots ,m\}\subset \mathbb {R} ^{n}}

;

1. Целевую функцию — отображение {\displaystyle f:\;\mathbb {X} \to \mathbb {R} } ;
2. [Критерий поиска](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D1%82%D0%B5%D1%80%D0%B8%D0%B9_%D0%BE%D0%BF%D1%82%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8) (max или min).

Тогда решить задачу {\displaystyle f(x)\to \min \_{{\vec {x}}\in \mathrm {X} }}  означает одно из:

1. Показать, что  {\displaystyle \mathbb {X} =\varnothing }.
2. Показать, что целевая функция *f(χ)* {\displaystyle f({\vec {x}})} не ограничена снизу.
3. Найти  {\displaystyle {\vec {x}}^{\*}\in \mathbb {X} :\;f({\vec {x}}^{\*})=\min \_{{\vec {x}}\in \mathbb {X} }f({\vec {x}})}
4. Если  {\displaystyle \nexists {\vec {x}}^{\*}}, то найти  {\displaystyle \inf \_{{\vec {x}}\in \mathbb {X} }f({\vec {x}})}.

Общая запись задач оптимизации задаёт большое разнообразие их классов. От класса задачи зависит подбор метода (эффективность её решения). Классификацию задач определяют: целевая функция и допустимая область (задаётся системой неравенств и равенств или более сложным алгоритмом).

# Постановка задачи

В данной работе необходимо реализовать последовательный алгоритм и параллельный алгоритм с использованием библиотек Open MP и TBB , позволяющий найти глобальный минимум функции. Для примера были взяты следующие функции:

0)

1)

2)

3)

4)

5)

6)

7) .

# Вспомогательные элементы

## 1. Checker

Проверяет корректность результата, сравнивая результаты работы алгоритма полного перебора и алгоритм Стронгина на основе тестов, которые создал генератор. Сравнения осуществляются по переменной , допустимая погрешность 0.0015. Записывает вердикт(AC = Accepted = Решение выдаёт корректный результат на данном тесте;

WA = Wrong Answer = Решение выдаёт некорректный результат на данном тесте; NO = No verdict = Вердикт отсутствует), комментарий вида корректен результат или нет и действительную разницу между -ами.

## 2. Generator

Генератор тестов. Генерирует границы интервала поиска в зависимости от выбранного номера функции, сам номер функции, параметр алгоритма r, параметр останова epsilon. После того, как генерация была осуществленна записывает все переменные в файл под номером теста в папку tests.

# Метод решения

Рассмотрев определение задачи оптимизации, перейдем к проблеме ее решения. Различные формулировки определений численного метода оптимизации даны многими авторами. Общим во всех формулировках является представление метода как некоторой итерационной процедуры, которая ( в общем случае последовательно) осуществляет вычисление в точках области поиска определённых характеристик минимизируемой функции (такими характеристиками могут быть значение функции, ее градиента, матрицы вторых производных и т.п.). Назовем операцию вычисления характеристик функции в точке поисковым испытанием, а совокупность значений характеристик в этой точке – результатом испытания.

Вычислительная схема алгоритма:

1. Выбирается точка первого испытания , где Ф – множество (класс), к которому принадлежит задача, {, k = 1, 2,… – семейство функционалов, описывающее совокупность правил выбора точек испытаний.
2. Пусть выбрана точка k-ого испытания Производится вычисление значения функции . После этого имеем следующую поисковую информацию о функции f:

.

Полученная информация позволяет сузить класс, которому принадлежит функция f(y), до множества

Ф(

1. Определяется текущая оценка экстремума (приближенное решение)

,

где {, k = 1, 2,… – последовательность отображений, задающая совокупность правил построения приближенного решения.

1. Вычисляется точка очередного испытания
2. Определяется величина принимающая одно из двух возможных значений: ноль или единица, где {, k = 1, 2,… - набор, определяющий совокупность правил остановки вычислительного процесса. Если = 0, вычисления прекращаются, и в качестве решения задачи берется оценка .

**Алгоритм метода Стронгина**

Пусть , , любого следующего испытания определяется следующими правилами:

* 1. Перенумеровать точки k предшествующих испытаний нижними индексами в порядке увеличения значений координаты:

a = = b и сопоставить им значения , i = 1, …, k.

* 1. Для каждого целого числа , определить соответствующее ему множество {\displaystyle I\_{\nu }} нижних индексов точек, в которых вычислялись значения функций   :
  2. Для каждого интервала вычислить характеристику

где , r > 1 – параметр метода.

* 1. Определить интервал  которому соответствует максимальная характеристика {\displaystyle R(t)=\max\{R(i),\;1\leqslant i\leqslant k\}}.
  2. Провести очередное в точке

увеличить {\displaystyle k}k на 1.

* 1. Если < (— заданная точность метода), то прекратить выполнение алгоритма, иначе перейти на шаг 1.

# Описание реализации последовательного алгоритма

1.Создается вектор, в который записываются пара из левой и правой границ интервала и соответствующих им

2.Вызываем сам алгоритм, описанный выше, который в данный вектор записывает новые, вычисленные значения по и соответствующие ему

3. В новом векторе ищем минимальный элемент по и достаем соответствующий ему

# Описание реализации параллельного алгоритма

1.На каждом потоке создается вектор, в который записываются правая и левая границы интервала, которые вычисляются для каждого потока исходя из того, что размер порции должен быть (правая граница – левая граница) / число потоков

2.Для каждого вектора вызываем сам алгоритм, описанный выше, который в данный вектор записывает новые, вычисленные значения по и соответствующие ему

3. Собираем все вектора в один, сортируя при этом по значению

4. В новом векторе ищем минимальный элемент по и достаем соответствующий ему

# Подтверждение корректности

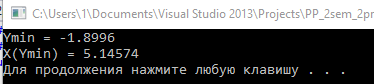
Проверим корректность задачи на основе приведенных в пункте «Постановка задачи» функциях на определенных интервалах и сравним их с результатами, которые выдает программа.

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Интервал | Ymin | X(Ymin) |
|  | [2,7 ; 7,5] | -1,899599 | 5,145735 |
|  | [0 ; 1,2] | -1,48907 | 0,96609 |
|  | [-10 ; 10] | -0,824239 | 0,679560 |
|  | [2,7 ; 7,5] | 5,19978 | -1,60130 |
|  | [0 ; 4] | -0,788685 | 0,224885 |
|  | [-5 ; 5] | -0,03553 | 2,41420 |
|  | [0 ; 6,5] | -7,815670 | 5,877287 |
|  | [-3 ; 3] | 7,515924 | 1,590700 |

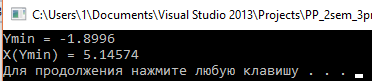
На следующих скриншотах представлены результаты моей программы с использованием библиотеки а) Open MP б) TBB. Эксперименты проводились на 4 процессах, параметром r = 1,8 и остановом epsilon = 0,001.

0)

А)

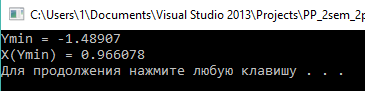


Б)

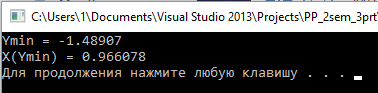


1)

А)

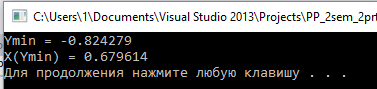


Б)

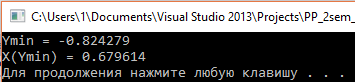


2)

А)

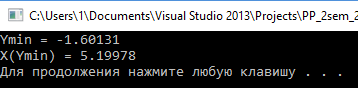


Б)

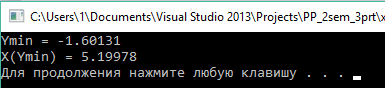


3)

А)

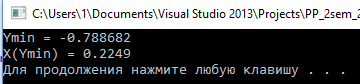


Б)

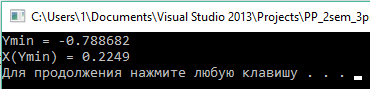


4)

А)

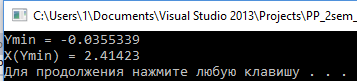


Б)

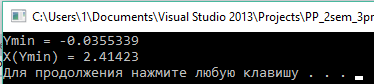


5)

А)

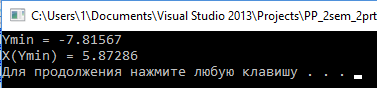


Б)

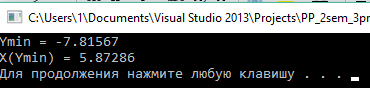


6)

А)

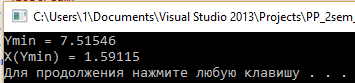


Б)

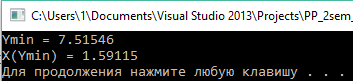


7)

А)



Б)



Погрешность не составляет больше 0,01. Исходя из этого можно судить о правильности реализации алгоритма.

# Результаты экспериментов по оценке масштабируемости

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Библиотека** | **1 процесс**  **сек.** | **Параллельный алгоритм** | | | | | | | |
| **2 процесса** | | **4 процесса** | | **8 процессов** | | **16 процессов** | |
| **T, сек.** | **Уск.** | **T, сек.** | **Уск.** | **T, сек.** | **Уск.** | **T, сек.** | **Уск.** |
| **Open MP** | 2.43955 | 2.64322 | 0.922 | 2.87731 | 0.803 | 3.21507 | 0.758 | 2.80438 | 0.871 |
| **TBB** | 2.43003 | 2.75331 | 0.8825 | 2.97446 | 0.817 | 3.4103 | 0.7125 | 2.83706 | 0.8564 |

Для имитирования вычислительной сложности функции в код оптимизируемых функций вставлен «холостой» цикл, который выполняет некоторые вычисления, не влияющие на значение функции. Все вычисления проводились для функции .

В зависимости от выбранной функции ускорение может быть отрицательным, как и получилось в данном случае.

# Вывод

Из полученных результатов можно сказать, что в рамках данной задачи использование параллелизма может быть рационально только для некоторых функций. В нашем случае оно оказалось нерациональным.