МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Кафедра: Математического обеспечения ЭВМ и суперкомпьютерных технологий**

Направление подготовки: «Фундаментальная Информатика и Информационные Технологии»

Магистерская программа: «Инженерия Программного Обеспечения»

**ОТЧЕТ**

по производственной практике

на тему:

**«Разработка системы Абсолют   
для многомерных задачах ГО и МКО»**

**Выполнил(а):**студент(ка) группы 0836-1м

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Пронина М.В.

Подпись

**Научный руководитель:**

Ассистент кафедры МОСТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Козинов Е.А.

Подпись

Нижний Новгород  
2018

Содержание

[Введение 3](#_Toc517294672)

[Постановка задачи 5](#_Toc517294673)

[1. Описание предметной области. 6](#_Toc517294674)

[1.1. Постановка задачи оптимизации с нелинейными ограничениями 6](#_Toc517294675)

[1.2. Численный метод математического программирования 7](#_Toc517294676)

[2. Архитектура программного комплекса Абсолют 8](#_Toc517294677)

[3. Внедрение нелинейных ограничений в программный комплекс «Абсолют» 11](#_Toc517294678)

[3.1. Проектирование представление задачи оптимизации с нелинейными ограничениями 11](#_Toc517294679)

[3.2. Адаптация кодовой базы клиентского приложения 15](#_Toc517294680)

[4. Интеграция программного комплекса с существующими библиотеками оптимизации 18](#_Toc517294681)

[4.1. MOEAFramework 18](#_Toc517294682)

[4.2. AlgLib 20](#_Toc517294683)

[5. Серии экспериментов в приложении Абсолют 22](#_Toc517294684)

[5.1. Требования к модулю серийных экспериментов 22](#_Toc517294685)

[5.2. Проектирование модели данных модуля 23](#_Toc517294686)

[Заключение 26](#_Toc517294687)

[Список литературы 27](#_Toc517294688)

# Введение

Сейчас существует много задач, которые требуют применения сложных вычислений и суперкомпьютерных технологий. Одним из таких ресурсоемких методов является глобальная оптимизация, применяемая для решения задач в механике, экономике, проектировании инженерных сооружений. Применение глобальной оптимизации позволяет найти такие параметры систем различных типов, которые позволяет достичь требуемых значений каких-либо характеристик, например, сейсмоустойчивости задания или веса конструкции.

Цель задачи глобальная оптимизации (ГО) состоит в нахождении экстремума целевой функции с минимальным значением в некоторой области конечномерного векторного пространства, ограниченной набором неравенств. Сейчас разработано множество алгоритмов глобального поиска, например, метод Пиявского [1], генетические алгоритмы и их разновидности [12], информационно-статистический метод Стронгина [2,3]. Многие из них обладают различными наборами параметров и следуют разным стратегиям поиска, но все они достаточно сложны и требуют тщательного изучения перед их практическим использованием.

Для помощи в изучении алгоритмов глобальной оптимизации разрабатывается программный комплекс Абсолют предназначенный для изучения процесса оптимизации и проведения вычислительных экспериментов для лучшего понимания сфер применения тех или иных алгоритмов. Разработка подобных учебных комплексов актуальна для сложных численных методов - в [4] описывается программный комплекс для изучения работы численных методов с использованием технологий параллельного программирования, в [5] основной целью комплекса являются параллельные методы оптимизации.

Основная сложность разработки данной системы заключается в необходимости универсального представления численных методов оптимизации разных типов и реализованных на различных языках программирования и поддержки нескольких постановок задач оптимизации. Уже существует система PISA для универсального представления многокритериальных оптимизационных задач и генетических алгоритмов оптимизации [6], но она не обладает визуальным интерфейсом и достаточно сложна в настройке под конкретную задачу. В качестве других примеров программных комплексов для работы с задачами глобальной оптимизации можно привести наборы оптимизационных инструментов в составе математических пакетов. GlobalOptimizationToolbox[10] математической системы MatLab предоставляет широкий спектр возможностей по изучению глобальной оптимизации и включает в себя средства для работы с локальными методами, генетическими алгоритмами и многокритериальными задачами. Система Mapleв свою очередь предлагает свой набор инструментов в виде Maple Global Optimization Toolbox[11]. Оба вышеуказанных продукта не являются свободно распространяемыми и работают в среде математического пакетов. Кроме того, для работы в них требуются навыки программирования на языках соответствующих математических пакетов и имеются сложности с подключением задач в виде уже разработанных DLL-библиотек.

# Постановка задачи

После 2 лет работы над комплексом в рамках бакалаврской работы была разработана первая версия системы для работы с задачами оптимизации без ограничений размерности 2. Построенная архитектура приложения позволяет расширять систему новыми алгоритмами оптимизации путем единого представления численного метода и добавлять задачи оптимизации при помощи написания новых классов задач или динамически подключаемых библиотек. Также разработана визуальная часть системы для демонстрации поверхностей функций и процесса решения задачи.

В рамках практической работы требовалось продолжить разработку системы Абсолют с целью расширения ее функциональности на случай многомерных функций с ограничениями. Кроме того, ставилась задача внедрения новых алгоритмов оптимизации и поддержка работы с задачами размерности больше 2. Основная задача, поставленная перед приложением Абсолют - создать программный комплекс, обеспечивающий возможность изучения алгоритмов глобальной оптимизации, их сравнения и анализа, в удобной визуальной форме.

Поставленная задача включает в себя следующие этапы:

1. Поддержка постановки задачи оптимизации с ограничениями cвозможностью дальнейшего расширения постановки задачи до задачи с несколькими критериями;
2. Визуальное представление задач с ограничениями в клиентском приложении;
3. Интеграция со сторонними системами оптимизации.
4. Работа с функциями размерности больше 2.

# 1. Описание предметной области.

# 1.1. Постановка задачи оптимизации с нелинейными ограничениями

Задача математического программирования с учетом ограничений ставится следующим образом:

,

где D– некоторое множество в простой структуры:

.

Функция и ограничения удовлетворяют обобщенному условию Липшица на всем интервале поиска:

где и - любые числа из области определения, L - константа, имеет обратную себе .

является областью поиска решения и на ней определены все функции задачи – и

Для использования методов решения задач оптимизации одномерных функций для многомерных используется приведение функции многих переменных к функции одной переменной. Используя кривые типа развертки Пеано [3], однозначно отображающие отрезок [0,1] на N-мерный гиперкуб   
а также линейное преобразование координат, отображающее гиперпараллелепипед D на гиперкуб P, исходную задачу можно редуцировать к следующей одномерной задаче:  
.

Многомерная область определения функции переводится в отрезок вещественной оси. Подобласти отображаются в одномерные интервалы. В зависимости от степени разбиения m число подобластей, на которые делится исходная область, различно, оно составляет , где N — размерность отображаемой области. Таким образом, можно, выбрав параметр разбиения m, отобразить точку в центре каждой подобласти (сетку на гиперкубе) в точку в центре каждого интервала на отрезке (сетку на отрезке). Такое отображение будет взаимно однозначным. Многомерная целевая функция будет определена на одномерной сетке.

Рассматриваемая схема редукции размерности сопоставляет многомерной задаче с липшицевой минимизируемой функцией и липшицевыми ограничениями одномерную задачу, в которой соответствующие функции удовлетворяют равномерному условию Гельдера, т.е

где N есть размерность исходной многомерной задачи, а коэффициенты связаны с константами Липшица исходной задачи соотношениями [3].

# 1.2. Численный метод математического программирования

Представление численных методов в системе основывается на следующем подходе к определению численного метода оптимизации как итерационной процедуре, вычисляющей в точках области поиска характеристики минимизируемой функции, описанной в работах Стронгина Р.Г. [2,3].

Алгоритм решения задачи из класса Ф задается как набор в котором –совокупность правил выбора точек испытаний, правила построения оценки экстремума, – правила остановки вычислительного процесса. В качестве испытания принимаем операцию вычисления характеристик функции в конкретной точке. Получается следующая итерационная схема алгоритма[2]:

1. Выбор точки первого испытания , k = 1
2. На k-том шаге производится вычисление значения функции . Поисковая информация (совокупность точек и вычисленных значений функции) на данном шаге выглядит как
3. Определяется текущая оценка экстремума
4. Вычисляется точка очередного испытания

5. Если величина = 1, k = k+1 и вычисления продолжаются. Иначе процесс останавливается.

Основным классом алгоритмов, представленных в системе Абсолют, являются характеристические алгоритмы.

Алгоритм решения задачи называется характеристическим, если начиная с некоторого шага поиска , выбор координаты следующего испытания можно описать в виде следующей последовательности действий:

1. Задается набор конечного числа точек из области определения исследуемой функции и упорядочивается по возрастанию координаты. В него могут входить и точки, в которых не проводились испытания, например, границы области определения оптимизируемой функции.
2. Каждому интервалу из этого набора ставится в соответствие некоторое число , называемое характеристикой этого интервала
3. Определяется интервал с максимальной характеристикой
4. Следующее испытание проводится в точке интервала с максимальной характеристики, которая определяется по некоторому правилу.

Характеристическими являются такие алгоритмы, как метод перебора, метод Кушнера, метод ломаных и базовый информационно-статистический алгоритм глобального поиска Стронгина, подробно описанные в [2;3]. В последнем в качестве функции используется

где величина вычисляется следующим образом:

Где - параметр метода.

# 2. Архитектура программного комплекса Абсолют

Первая версия системы Абсолют предназначалась для проведения экспериментов по решению задач оптимизации без ограничений размерности равной двум.

Задача глобальной оптимизации в нем ставится как нахождение глобального минимума функции f(𝑥) на множестве D.

Многоэкстремальная функция f(𝑥) задается на своей области определения D. Область определения задается гиперкубом в многомерном пространстве левой и правой границей.

Для универсального представления численных методов решения задачи, методы в программном комплексе представляются как набор , в котором - правила выбора точек испытаний, - правила построения оценки экстремума, - правила остановки вычислительного процесса.

Архитектура приложения была разработана с целью легкого дальнейшего расширения комплекса новыми алгоритмами и задачами. Поэтому архитектура строилась по компонентному подходу с выделением следующих компонент (диаграмма компонент представлена на рисунке 1):

1. Модель, содержащая данные вычислительных экспериментов.
2. Алгоритмы оптимизации.
3. Библиотека целевых функций.
4. Набор графических компонент.

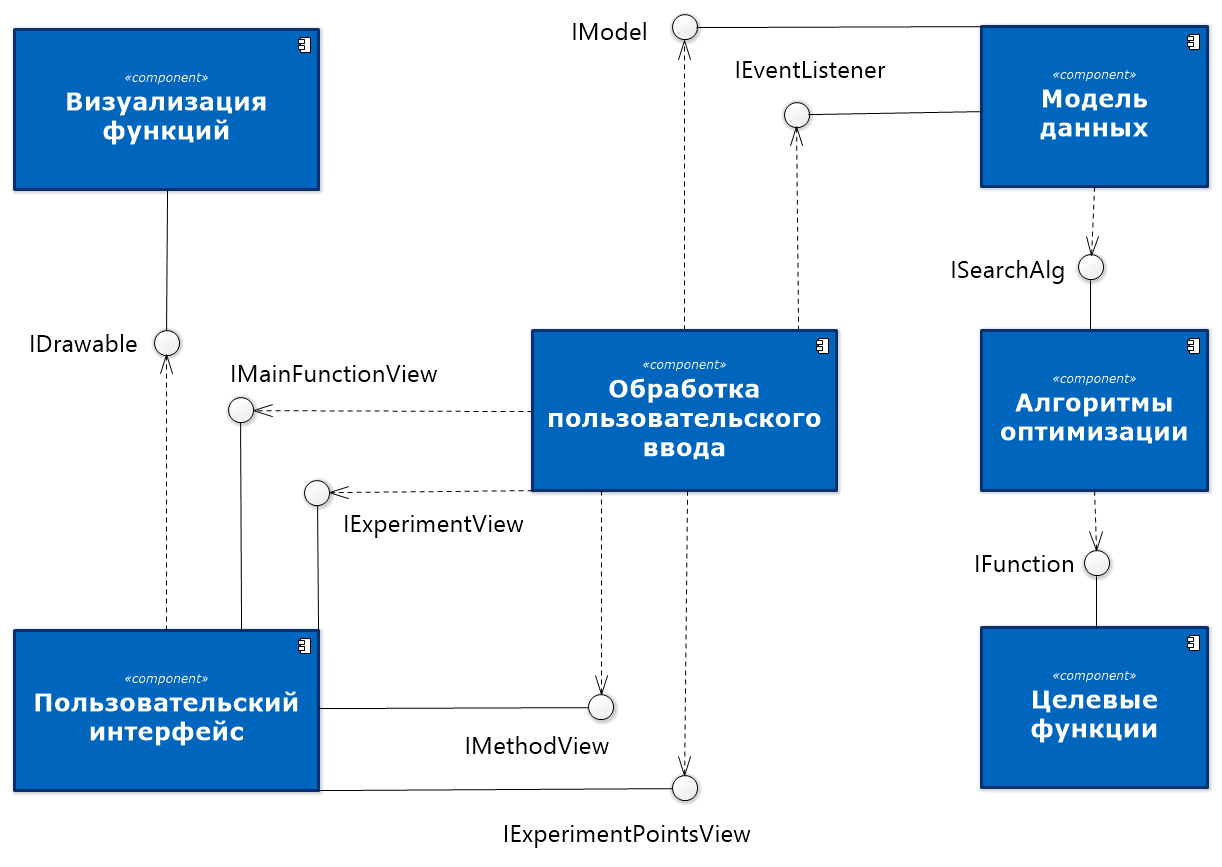


Рисунок 1. Диаграмма компонент

Компоненты организованы по слоям – слой данных, слой представления и слой отображения по паттерну Model-View-Presenter (рисунок 2). Взаимодействие между слоями происходит посредством выделенных интерфейсов.



Рисунок 2. Послойная архитектура системы.

Основными интерфейсами системы являются ISearchAlg (представление численного метода математического программирования), IFunction (представление задачи оптимизации) и IModel (управление данными экспериментов).

Графическая часть приложения состоит из нескольких визуальных элементов для отображения процесса решения задачи. Использование OpenGLпозволило достичь динамического обновления графика целевой функции, линий уровня и карты температур во время решения задачи вместе с возможностью поворота поверхностей.

# 3. Внедрение нелинейных ограничений в программный комплекс «Абсолют»

Далее описывается процесс расширения существующего приложения комплекса Абсолют новой функциональностью, связанный с численными методами решения задач с ограничениями. Описано текущее представление задачи математического программирования в системе и способы ее дополнения функциональными ограничениями с сохранением клиентского кода и возможностью введения многокритериальной оптимизации.

# 3.1. Проектирование представление задачи оптимизации с нелинейными ограничениями

В программном комплексе Абсолют задача оптимизации представляется в виде объекта с интерфейсом IFunction (листинг 1), который представляет собой функцию и ее области определения D.

Листинг 1. Интерфейс компоненты Functions.

public interfaceIFunction

{

// размерность задачи

int Dimension { get; set; }

// имя задачи

String Name { get; set; }

// границы допустимой области

List<double> Right { get; set; }

List<double> Left { get; set; }

// информация об известном глобальном минимуме

List<double> Min\_x { get; set; }

double? Min { get; set; }

// вычисление значения функции в точке

double Calc(List<double> arg);

// методы для представления многомерной функции как одномерной

double Calc(double x);

List<double> GetImage(double x);

double GetPrototype(List<double> x);

// методы для совместимости с форматом представления задач в системе Globalizer

int GetNumberOfFunctions();

void SetFunctionNumber(int index);

}

Для представления новой постановки задачи глобальной оптимизации с поддержкой ограничений одного интерфейса IFunction оказалось недостаточно, т.к. необходимо обеспечить хранение и вычисление функциональных ограничений. Существенным недостатком текущего подхода является и смешение представлений функционала и задачи оптимизации, что размывает назначение объектов с интерфейсом IFunction.

При выборе подхода к расширению постановки задачи необходимо обратить внимание на сохранение совместимости реализованного функционала и нового представления задачи математического программирования. Описанный выше интерфейс активно используется в графической компоненте и в клиентском приложения, поэтому можно предложить следующие варианты:

1. Использовать коллекцию объектов IFunctionкак представление набора из функционалов f(x) и g(x);
2. Обратиться к существующим методам GetNumberOfFunctions и SetFunctionNumber (реализованы для совместимости с форматом представления динамических библиотек с задачами оптимизации системы Globalizer[13]) для получения доступа к функционалам;
3. Создать новую сущность для представления задачи оптимизации, включающую в себя набор ограничений, целевую функцию (представленные интерфейсом IFunction) и другую информацию по задаче.

Первые два способа не требуют существенных изменений кода ядра программного комплекса и его клиентской части, но не лишены недостатков:

1. Представление задачи оптимизации в виде коллекции функционалов не позволяет сохранить информацию об известном решении задачи и смешивает ограничения и критерий задачи, что может вызывать ошибки при реализации численных методов;
2. Использование текущего представления с двумя методами расширения GetNumberOfFunctions и SetFunctionNumber объединяет понятия задачи оптимизации и функции в одном объекте IFunction, что противоречит парадигме объектно-ориентированного проектирования.

Поэтому было решено остановится на варианте номер 3 и добавить в систему новый интерфейс для задачи оптимизации (листинг 2).

Листинг2. Интерфейс IProblem.

public interface IProblem

{

// размерность задачи

int Dimension { get; set; }

// имя задачи

string Name { get; set; }

// границы допустимой области

List<double> LowerBound { get; }

List<double> UpperBound { get; }

// доступ к функционалам задачи - критерием и ограничениям

List<IFunction> Criterions { get; }

List<IFunction> Constraints { get; }

List<IFunction> Functionals { get; }

int NumberOfFunctions { get; }

int NumberOfConstratins { get; }

int NumberOfCriterions { get; }

// вычисление значений функционалов в точке

double CalculateFunctionals(List<double> y, int fNumber);

List<double> CalculateCriterions(List<double> x);

List<double> CalculateConstraints(List<double> toList);

// информация об известном решении задачи

double? OptimalValue { get; }

List<double> OptimalPoint { get; }

double GetOptimalValue(int index);

// начальная инициализация

void Initialize();

// путь к конфигурационному файлу для настройки динамически подключаемых задач

string ConfigPath { get; set; }

// путь к файлу динамической библиотеки

string DllPath { get; set; }

}

Он обеспечивает доступ к известному решению задачи, области определения и всем функционалам. В процессе программирования численных методов математического программирования для получения значений функционалов в точке предлагается использовать методы CalculateFunctionals, CalculateCriterions и CalculateConstraints.

Новый интерфейс совместим с форматом задач системы Globalizer для облегчения их интеграции.

Доступ к функционалам задачи в виде объектов с интерфейсом IFunction позволяет повторно использовать уже реализованные средства визуализации функций.

Разработчикам доступны несколько способов расширения системы новыми задачами оптимизации:

1. Создание класса новой задачи, реализующий интерфейс IProblem. Для этого можно воспользоваться базовым классом Problem в качестве родительского класса и использовать некоторые стандартные реализации. Примером такого способа является класс DllProblem для работы с задачами системы Globalizer.
2. Использование фабричных методов класса ProblemFactory для конструирования задачи из существующих функций.
3. Паттерн проектирования Bridge для трансляции обращений к интерфейсу IProblem в некоторую существующую реализацию задачи оптимизации.

Далее рассмотрим вопрос изменения уже реализованных в системе численных методов математического программирования для адаптации к новой постановке задачи оптимизации в виде объекта с интерфейсом IProblem.

Все разработанные алгоритмы реализуют интерфейс ISearchAlg для универсального представления численного метода оптимизации. ISearchAlg предоставляет информацию об оценке решения, вычисленных точках и позволяет устанавливать свойства метода – данные операции характерны для всех алгоритмов оптимизации.

Со стороны клиентского кода, процесс решения любой задачи любым из алгоритмов системы организован по выше приведенной схеме. В каждом из шагов схемы может быть обращение к объекту-задаче для вычисления значений функции, поэтому переход от постановки задачи без ограничений к задаче с ограничениями поднял довольно острый вопрос организации доступа к текущей задаче в объектах-алгоритмах.

Предыдущий подход заключался в регистрации в алгоритме специального свойства для сохранения целевой функции (используя уже описанный интерфейс IFunction) и дальнейшей работе с ним. Он показал себя несостоятельным – регистрация данного свойства не была автоматической и при реализации новых алгоритмов требовалось регистрировать его в коде с определенным именем.

В качестве решения в интерфейс ISearchAlgбыло добавлено свойство Problem нового типа IProblem, которое заменило регистрируемый вручную параметр. При этом не потребовалось значительных изменений клиентского кода – получения свойства MainFunction было заменено на Problem.GetCriterions().FirstOrDefault().

Еще одним изменением стал перенос методов развертки Пеано из IFunction в отдельный класс PeanoHelper. В дальнейшем планируется поддержка многошаговой редукции в качестве второго способа работы с многомерными функциями в численных методах.

Структура MethodPoint была расширена в соответствии с новой постановкой задачи и включает поля для вычисленных значений ограничений и критериев в точке (листинг 3).

Листинг3. Класс MethodPoint.

public class MethodPoint

{

// информация для индексного метода решения - на каком номере функции остановился расчет в этой точке

publicint IdFun;

// флаг окончания процесса поиска минимума на данной точке

public AlgEnd isEnd;

// координаты точки

public List<double> x;

// образ на развертке Пеано

public double evolventX;

// вычисленное значение критерия

public double? y;

// номер итерации

public int iteration;

// номерпроцесса

public int id\_process;

// значения критериев в точке

public List<double> criterions;

// значения ограничений в точке

public List<double> constraints;

}

# 3.2. Адаптация кодовой базы клиентского приложения

В клиентском коде программного комплекса значительные переработки коснулись главного окна эксперимента. До внедрения задач с ограничениями на нем располагались сведения о текущей точке испытания, лучшей точке, целевой функции (рисунок 3).

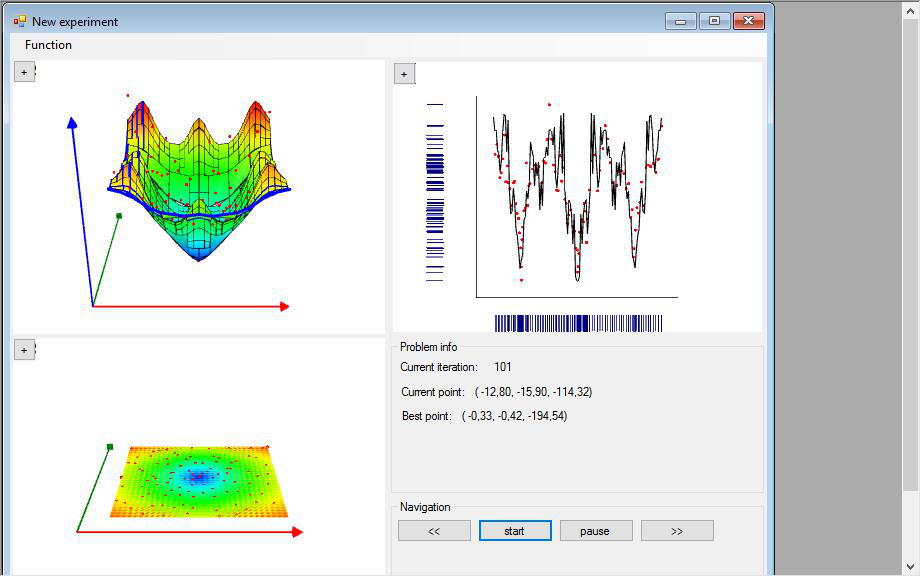


Рисунок 3. Представление решения задачи без ограничений в программном комплексе Абсолют.

Визуализация отображения целевой функции на результирующем множестве Dразработана на основе построения поверхности функции без учета ограничений. Для этого класс Surfaceиз библиотеки графических элементов Renderingбыл переработан в класс ProblemaSurface. ProblemaSurface использует интерфейс IProblem и учитывает значения ограничений в точках сетки для построения поверхности. Реализация учета ограничений потребовала изменений только в методе FillPolygons, который составляет сетку вершин. Остальной процесс отображения поверхности остался неизменным.

Пример построенной поверхности с учетом функциональных ограничений показан на рисунке 4.

|  |  |
| --- | --- |
| Criteria After Costraint05 | Criteria 005 |

Рисунок 4. Пример построения поверхности целевой функции с учетом ограничений / без учета ограничений

После интеграции нового типа задач в пользовательский интерфейс введено новое рабочее окно с информацией по всем функционалам системы и отображением целевой функции на множестве D. Для этого в пользовательском элементе управления визуализации поверхности критерия был изменен объект для отображения с типа Surface на тип ProblemSurface.

В пользовательском интерфейсе отображение организовано при помощи нескольких дочерних форм для каждого функционала задачи. Итоговая поверхность целевой функции доступна в отдельном окне. Все окна функционалов имеют синхронизированное вращение поверхностей.

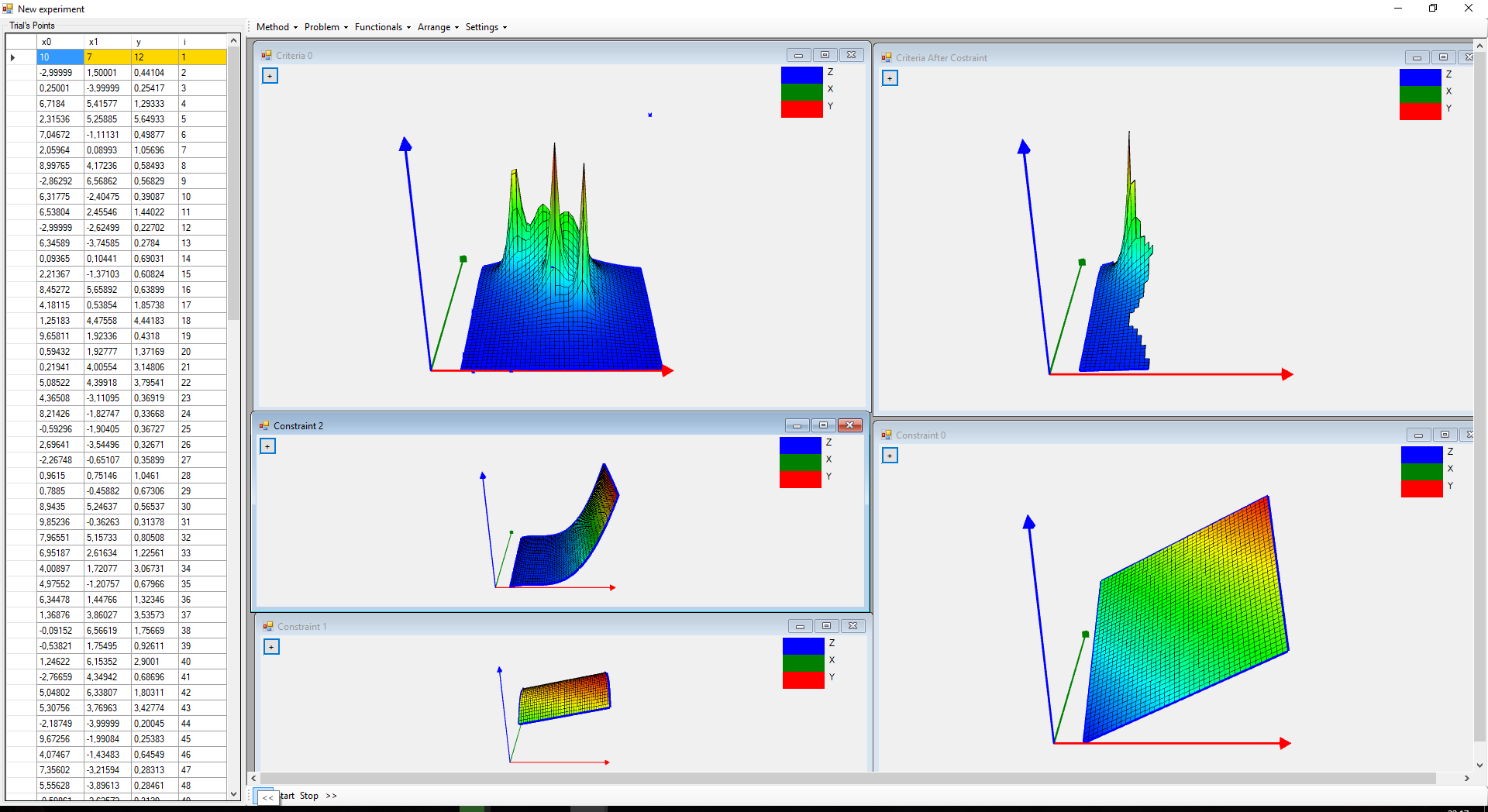


Рисунок 5. Форма эксперимента для задач с ограничениями.

Архитектура приложения, спроектированная с учетом особенностей предметной области и применением паттернов проектирования, позволила провести значительное изменение одного из основных интерфейсов практически без переписывания основных частей кода.

# 4. Интеграция программного комплекса с существующими библиотеками оптимизации

Одним из основных направлений развития программного комплекса Абсолют является взаимодействие с существующими системами оптимизации для наглядного сравнения и изучения процесса поиска решения. На данный момент была интегрирована система Globalizerпри помощи вызова исполняемого файла специально сформированной командной строкой и последующим импортом файла результатов. Работа в этом направлении была продолжена интеграцией библиотек AlgLib и MOEAFramework, внедрение которых осуществлялось путем написания специальных адаптеров для постановки задачи.

# 4.1. MOEAFramework

MOEAFramework - комплект библиотек для изучения, сравнения и использования различных генетических алгоритмов многокритериальной оптимизации на языке Java. Список доступных методов содержит около 20 наименований, среди которых NSGA-II, NSGA-III, ε-MOEA[7].

Интеграция производилась путем преобразования библиотек MOEAFramework в DLL-библиотеки для языка C# с использованием набора инструментов IKVM.NET[8]. Кроме такого подхода рассматривался способ, аналогичный работе с системой Globalizer - написание java-приложения для вызова из командной строки, которое обеспечивало бы доступ к функциональности MOEAFramework. Использование IKVM.NET оказалось предпочтительнее из-за прямого подключения библиотек к системе Абсолют без дополнительной разработки консольного приложения.

После конвертации библиотек интегрируемой системы требовалось привести интерфейс задачи оптимизации программного комплекса Абсолют IProblem к постановке задачи в рамках MOEAFramework (класс AbstractProblem) при помощи паттерна Адаптер.

Для этого реализован класс MoeaProblem (листинг 4), наследующий класс AbstractProblem. Исходная задача оптимизации хранится в виде поля класса и используется для вычисления значений функций в точках, найденных алгоритмом. Кроме этого в классе производится необходимая инициализация переменных для корректной работы метода.

В методе evaluate (листинг 4) происходит основная работа с задачей оптимизации – вычисление значений функций в точке. Кроме этого, метод имеет важное значение с точки зрения интеграции численного метода в систему – он регистрирует все точки, в которых вычисляется значение функции. Это необходимо, т.к. MOEAFramework не предполагает доступа к точкам испытаний алгоритма.

Листинг 4. Регистрация точек испытания внешней системы.

public override void evaluate(Solution solution)

{

// Получение точки испытания из текущего решения

double[] x = EncodingUtils.getReal(solution);

// Вычисление значений критериев и ограничений в точке испытания

double[] f = \_problem.CalculateCriterions(x.ToList()).ToArray();

double[] constr = \_problem.CalculateConstraints(x.ToList()).ToArray();

// Добавление точки испытания в список вычисленных точек, который в дальнейшем

// будет использован для сбора результатов работы алгоритма оптимизации

\_points.Add(new MethodPoint()

{

evolventX = 0,

id\_process = 0,

IdFun = 0,

isEnd = AlgEnd.no,

iteration = Iterations,

x = x.ToList(),

y = f[0],

constraints = \_problem.CalculateConstraints(x.ToList()),

criterions = \_problem.CalculateCriterions(x.ToList())

});

// Передача вычисленной информации в MOEA Framework

solution.setObjectives(f);

solution.setConstraints(constr);

Iterations++;

}

Для запуска вычислений в классе-наследнике GenericAlg в методе RecalcPoints осуществляется инициализация процесса вычисления в библиотеке MOEA (листинг 5):

Листинг 5. Использование библиотеки MOEA для реализации метода оптимизации

public override void RecalcPoints()

{

if (Problem == null) throw new InvalidOperationException(ExceptionMessages.ProblemIsNull);

// Установка задачи оптимизации в класс-адаптер

MoeaProblem.\_problem = Problem;

// Запуск вычислений классом Executor. Точки испытания будут получены из статического поля

// класса MoeaProblem (см. метод evalute)

NondominatedPopulation result = new Executor().withProblemClass(typeof(MoeaProblem)).withAlgorithm("NSGAII")

.withMaxEvaluations((int)GetProperty(StandartProperties.MaxIter))

.run();

// Передача результатов из MoeaProblem в поля класса GenericAlg

ExperimentPoints = MoeaProblem.\_points;

ExperimentPointsReal = MoeaProblem.\_points;

CurrentIteration = MoeaProblem.Iterations;

}

Обновление коллекции вычисленных точек производится в классе MoeaProblem при помощи статических полей.

Хотя такой подход позволяет быстро интегрировать стороннюю систему, написанную на языке Java, возможны проблем с конвертацией jar-файлов в DLL-библиотеки. В таких случаях следует обратится к уже упомянутому подходу с реализацией консольного приложения и передача ему определенных аргументов для запуска алгоритмов без конвертации исходных библиотек и использования сторонних инструментов.

# 4.2. AlgLib

В отличие от предыдущей системы математическая библиотека AlgLib доступна в виде Nuget-пакета для языка C#.

В случае библиотеки AlgLib основной сложностью стало получение вычисленной на итерации точки и связь интерфейса IProblem со способом вычисления функции Alglib.

Для решения первой задачи в вычислительный метод библиотеки передается Callbackфункция, которая вызывается после очередной итерации (листинг 6). В ней производится добавление точек испытания в коллекцию испытаний алгоритма.

Листинг 6. IterationCallback

publicvoid IterationCallback(double[] x, double func, object obj)

{

var methodPoint = new MethodPoint()

{

evolventX = 0,

IdFun = 0,

iteration = CurrentIteration + 1,

y = func,

x = x.ToList()

};

ExperimentPointsReal.Add(methodPoint);

ExperimentPoints.Add(methodPoint);

CurrentIteration++;

}

Для вычисления функций задачи в объекты Alglib передается указатель на функцию с определенной сигнатурой. Пример интеграции постановки задачи системы с внешней библиотекой представлена листинге 7.

Листинг 7. Адаптер между IProblem и представлением задачи в AlgLib

private void ConstrainedProblemAdapter(double[] x, double[] fi, object obj)

{

if (!(Problem != null&& Problem.NumberOfConstratins > 0 && Problem.NumberOfCriterions == 1))

throw new NotSupportedException();

// вычисление целевой функциивточке

fi[0] = Problem.Criterions[0].Calc(x.ToList());

// вычисление ограничений в точке

for (int i = 1; i <= Problem.NumberOfConstratins; i++)

{

fi[i] = Problem.Constraints[i - 1].Calc(x.ToList());

}

}

# 5. Серии экспериментов в приложении Абсолют

В изучении работы любых численных методов глобальной оптимизации часто необходимо проводить серийные эксперименты на наборе однотипных задач для вычисления операционной характеристики метода. Расширение приложения Абсолют функциональностью для проведения серий экспериментов позволит в наглядном виде изучать поведение алгоритмов глобальной оптимизации.

# 5.1. Требования к модулю серийных экспериментов

В текущем приложении исследователь может изучать поведение разных алгоритмов оптимизации на конкретных задачах. Но для сравнения численных методов распространены серии экспериментов на тестовых наборах данных или семействе тестовых задач. Имея запуски разных алгоритмов на одном семействе тестовых задач, можно делать выводы о операционных характеристиках алгоритмов, среднем количестве итераций и сравнивать результаты работы численных методов по этим параметрам.

На основании типичных действий во время проведения серийных экспериментов была составлена следующая use-caseдиаграмма (рисунок 4).

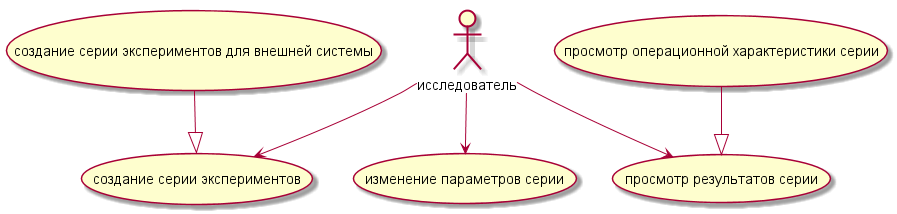


Рисунок 6. UseCase диаграмма использования серий экспериментов

По выделенным сценариям использования система должна поддерживать следующие операции:

1. Создание серии экспериментов с указанием семейства тестовых задач;
2. Отображение результатов серии и формирование на их основе операционной характеристики;
3. Изменение параметров алгоритма после получения результатов.

# 5.2. Проектирование модели данных модуля

Управление экспериментами в приложении Абсолют осуществляется при помощи компоненты AbsolutKernel и класса модели данных IModel. Указанные выше операции было решено не включать в уже существующий интерфейс IModel, а создать для новых сущностей собственную модель данных. Таким образом функциональность будет разделена и практически независима друг от друга.

Для серии экспериментов создана отдельная модель данных с интерфейсом, представленном на рисунке 7.

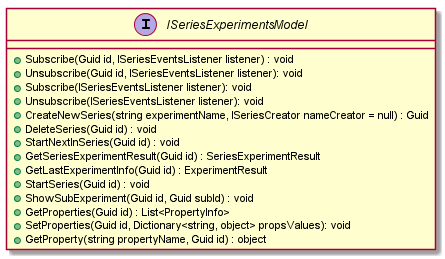


Рисунок 7. Интерфейс ISeriesExperimentsModel

Оповещение клиентского кода о создании серии экспериментов и связанных событиях производится путем подписки на события (паттерн Наблюдатель). Клиенты получают оповещения о создании серии, удалении серии, старте одиночного эксперимента и так далее. Доступные события описаны в интерфейсе ISeriesEventsListener(рисунок 8).

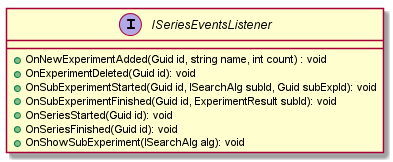


Рисунок 8. Интерфейс ISeriesListener

Каждая серия регистрируется в модели данных со своим уникальным id, по которому в дальнейшем клиентский код может получить необходимую информацию и подписаться на события конкретной серии. Аналогично одиночным экспериментам, модель серий является одиночкой, доступ к ней обеспечивается фабричным методом.

Сам серийный запуск представлен в системе классом SeriesExperiment. Он позволяет управлять исследуемым численным методом, набором функций для запуска серии и выдает результаты проведенных экспериментов в виде операционной характеристики и количества итераций на конкретных задачах.

В пользовательском интерфейсе для отображения нового вида сущностей добавлены два презентера – один для создания и удаления серий, второй для управления текущей серией экспериментов и обработки окна серии. Окно серии представлено на рисунке 9.

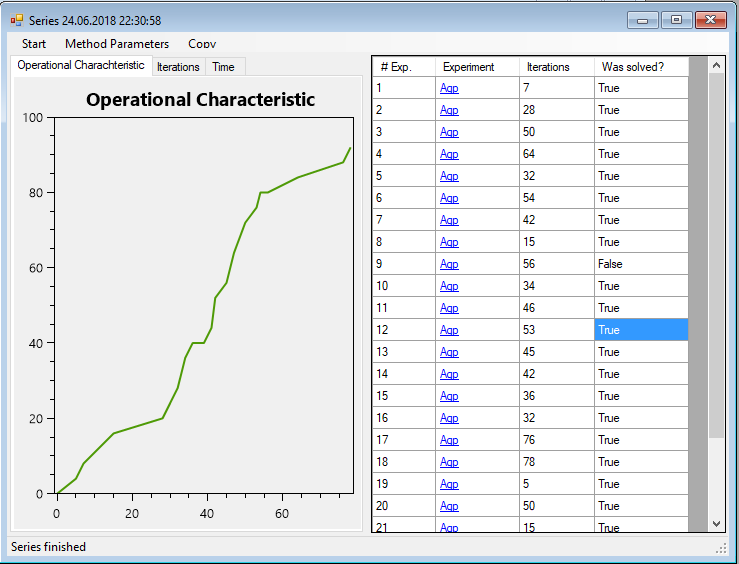


Рисунок 9. Окно серии экспериментов

Основной задачей стала разработка окна серии и связь серии экспериментов с одиночными экспериментами для открытия отдельного запуска. Для этого основная форма приложения, кроме подписки на события отдельных экспериментов, подписывается на события модели серии экспериментов и обрабатывает событие «показ нового дочернего эксперимента»

Операционная характеристика считается по окончанию всех запусков алгоритма путем подсчета расстояния между известным решением задачи и точками испытаний. Если находится точка, расстояние между которой и решением меньше заданного эпсилон, то задача считается решенной.

Запуск серии экспериментов осуществляется на семействе функций GKLS. Встроена поддержка серий для внешних систем оптимизации (в данный момент поддерживается система Globalizer). В последнем случае запуск осуществляется на подключаемых задачах в виде DLL-библиотек с указанием папки конфигурационных файлов.

В качестве примера результата работы модуля серий экспериментов далее приведены результаты запуска реализации алгоритма глобального поиска Стронгина Р.Г.[2] на 25 функциях из генератора задач GKLS с параметром r=2, epsilon = 0.0001.

Таблица результатов серии экспериментов:

|  |  |  |  |
| --- | --- | --- | --- |
| № эксп. | Алгоритм | Количество итераций | Найден мин. |
| 1 | Agp | 7 | True |
| 2 | Agp | 28 | True |
| 3 | Agp | 50 | True |
| 4 | Agp | 64 | True |
| 5 | Agp | 32 | True |
| 6 | Agp | 54 | True |
| 7 | Agp | 42 | True |
| 8 | Agp | 15 | True |
| 9 | Agp | 56 | False |
| 10 | Agp | 34 | True |
| 11 | Agp | 46 | True |
| 12 | Agp | 53 | True |
| 13 | Agp | 45 | True |
| 14 | Agp | 42 | True |
| 15 | Agp | 36 | True |
| 16 | Agp | 32 | True |
| 17 | Agp | 76 | True |
| 18 | Agp | 78 | True |
| 19 | Agp | 5 | True |
| 20 | Agp | 50 | True |
| 21 | Agp | 15 | True |
| 22 | Agp | 33 | True |
| 23 | Agp | 39 | False |
| 24 | Agp | 47 | True |
| 25 | Agp | 41 | True |

Таблица 1. Запуск 25 экспериментов на алгоритме АГП

Построенная в приложении операционная характеристика представлена на рисунке 10.

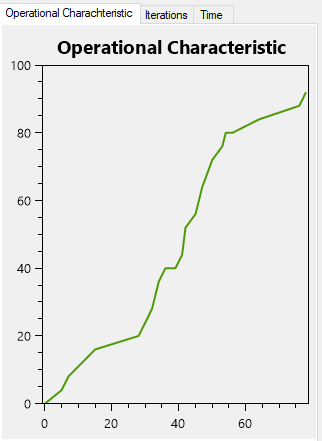


Рисунок 10. Операционная характеристика серии из 25 запусков

Изменив значение параметра r=5.95 в свойствах метода, получаем следующие результаты (рисунок 11).

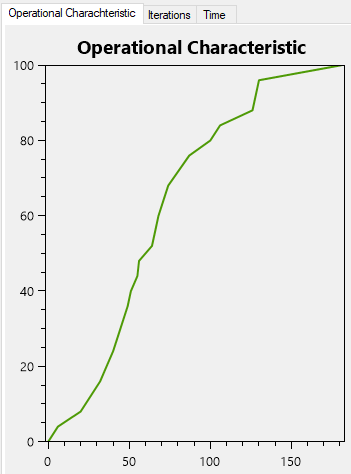


Рисунок 11. Операционная характеристика серии после изменения параметра r

Результаты отдельных экспериментов представлены в таблице 2. Изменение параметров метода доступно сразу же после окончания вычислений, что позволяет изучать поведение метода при различных значениях параметров.

Таблица 2. Результаты отдельных экспериментов после изменения параметра r

|  |  |  |  |
| --- | --- | --- | --- |
| № эксп. | Алгоритм | Количество итераций | Найден мин. |
| 1 | Agp | 36 | True |
| 2 | Agp | 74 | True |
| 3 | Agp | 55 | True |
| 4 | Agp | 20 | True |
| 5 | Agp | 126 | True |
| 6 | Agp | 181 | True |
| 7 | Agp | 130 | True |
| 8 | Agp | 32 | True |
| 9 | Agp | 100 | True |
| 10 | Agp | 66 | True |
| 11 | Agp | 51 | True |
| 12 | Agp | 128 | True |
| 13 | Agp | 40 | True |
| 14 | Agp | 64 | True |
| 15 | Agp | 6 | True |
| 16 | Agp | 49 | True |
| 17 | Agp | 68 | True |
| 18 | Agp | 87 | True |
| 19 | Agp | 56 | True |
| 20 | Agp | 43 | True |
| 21 | Agp | 106 | True |
| 22 | Agp | 87 | True |
| 23 | Agp | 49 | True |
| 24 | Agp | 71 | True |
| 25 | Agp | 26 | True |

# Заключение

В ходе выполнения работы архитектура приложения Абсолют расширена сущностями для постановки задачи оптимизации с ограничениями.

Кроме того, существующие отображения функций и точек испытания расширены возможностями визуализации целевой функции после применения ограничений.

В приложение интегрирован MOEAFramework и некоторые алгоритмы из библиотеки AlgLib.

Добавлена возможность проводить серийные эксперименты на наборе однотипных задач.

Вся разработка производилась в среде MSVisualStudio на языке С# с использованием библиотек .NET и библиотек для работы с OpenGL.

Визуализация задач оптимизации была использована в оформлении раздела сайта высокопроизводительных вычислений университета, посвященного глобальной оптимизации.

# Список литературы

1. Пиявский С. А. Один алгоритм отыскания абсолютного экстремума функций / С.А. Пиявский // Журнал вычислительной математики и математической физики – 1972- т.12, № 4 - стр. 885—896.
2. Параллельные вычисления в задачах глобальной оптимизации: монография / Р.Г. Стронгин [и др.] – М.: Издательство Московского университета, 2013. – 280 с.
3. Стронгин, Р.Г. Численные методы в многоэкстремальных задачах (информационно-статистические алгоритмы) / Р.Г. Стронгин. – М: Издательство «Наука», 1978 г. – 240 с.
4. Лабутина, А.А. ПараЛаб – среда для проведения вычислительных экспериментов для изучения и исследования параллельных методов решения сложных вычислительных задач / А. А. Лабутина, В.П. Гергель // Вестник Нижегородского университета им. Н.И. Лобачевского – 2011 - №3 [2] – с. 239-247
5. Сысоев, А.В. Программный комплекс параллельных вычислений в задачах выбора глобально-оптимальных решений / А.В. Сысоев // Вестник Нижегородского университета им. Н.И. Лобачевского – 2012 - №5 [2] – c. 425-431
6. Stefan Bleuler, Marco Laumanns, Lothar Thiele, EckartZitzler. PISA — A Platform and Programming Language Independent Interface for Search Algorithms / Stefan Bleuler, Marco Laumanns, Lothar Thiele, EckartZitzler. [Электронный ресурс] Режим доступа: <http://www.tik.ee.ethz.ch/sop/pisa/publications/pisa_paper.pdf> (Дата обращения 06.06.18)
7. MOEAFramework - A Free and OpenSource Java Framework for Multiobjective Optimization (заглавие с экрана) [Электронный ресурс] Режим доступа: <http://moeaframework.org/> (Дата обращения 23.12.17)
8. IKVM.NET (заглавие с экрана) [Электронный ресурс]. Режим доступа: <https://www.ikvm.net/>(Дата обращения 23.12.17)
9. AlgLib(заглавие с экрана) [Электронный ресурс]. Режим доступа: <http://www.alglib.net/> (Дата обращения 10.06.18)
10. Global Optimization Toolbox (заглавие с экрана) [Электронный ресурс].Режим доступа <https://www.mathworks.com/products/global-optimization.html> (Дата обращения 10.06.2018)
11. Maple Global Optimization Toolbox (заглавиесэкрана). [Электорнный ресурс]. Режим доступа - <https://www.maplesoft.com/products/toolboxes/globaloptimization/> (Дата обращения 10.06.2018)
12. Practical genetic algorithms / Randy L. Haupt, Sue Ellen Haupt.—2nd ed.
13. Sysoyev, A., Barkalov, K., Sovrasov, V., Lebedev, I., & Gergel, V. (2017, September). Globalizer–A Parallel Software System for Solving Global Optimization Problems. In International Conference on Parallel Computing Technologies (pp. 492-499). Springer, Cham.