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 381706-1м

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

Подпись

**Научный руководитель:**

Ассистент кафедры МОСТ

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

Подпись

Нижний Новгород  
2018

Содержание

[Введение 3](#_Toc533447345)

[Постановка задачи 5](#_Toc533447346)

[1. Описание предметной области. 6](#_Toc533447347)

[1.1. Постановка задачи оптимизации с нелинейными ограничениями и несколькими критериями 6](#_Toc533447348)

[1.2. Численный метод математического программирования 7](#_Toc533447349)

[2. Архитектура программного комплекса Абсолют 9](#_Toc533447350)

[3. Внедрение многокритериальных задач в программный комплекс «Абсолют» 12](#_Toc533447351)

[3.1. Представление задачи многокритериальной оптимизации 12](#_Toc533447352)

[3.2. Интеграция с системой Globalizer 15](#_Toc533447353)

[4. Клиентское приложение для программного комплекса «Абсолют» 17](#_Toc533447354)

[4.1. Стратегия смены технологии пользовательского интерфейса 17](#_Toc533447355)

[4.2. Реализация приложения на технологии WPF 18](#_Toc533447356)

[Заключение 22](#_Toc533447357)

[Список литературы 23](#_Toc533447358)

[Приложение 1. Класс ExaminMcoAlg 25](#_Toc533447359)

# 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. Global Optimization Toolbox[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

В прошлом семестре в приложение Абсолют была добавлена поддержка задач глобальной оптимизации с ограничениями: расширена модель предметной области и соответственно адаптирована визуальная часть приложения.

В рамках практической работы в текущем семестре требовалось продолжить разработку системы Абсолют с целью расширения ее функциональности на случай многокритериальных задач глобальной оптимизации. Кроме того, ставилась задача перевода пользовательского интерфейса на технологию WPF с последующей интеграцией разработанных алгоритмов многокритериальной оптимизации в систему. 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 c ограничениями и несколькими критериями;
2. Визуальное представление задач многокритериальной оптимизации в клиентском приложении;
3. Интеграция со сторонними системами оптимизации.

# 1. Описание предметной области.

# 1.1. Постановка задачи оптимизации с нелинейными ограничениями и несколькими критериями

Задача математического программирования с учетом ограничений ставится следующим образом:

где D– некоторое множество в простой структуры:

.

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

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

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

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(y) на множестве D.

Многоэкстремальная функция f(y) 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. Хранение одиночных экспериментов, системные события и презентационный слой - AbsolutKernel.
2. Серийное проведение экспериментов - SeriesExperiment.
3. Алгоритмы оптимизации - AbsolutAlg.
4. Библиотека целевых функций и проблем оптимизации - Functions.
5. Визуализация функций и элементы пользовательского интерфейса - Rendering
6. Первая версия GUI приложения - AbsolutDemo
7. Разработанное в данном семестере GUI приложение - AbsolutUi
8. Консольное приложение AbsolutConsole для запуска вычислений в командной строке.

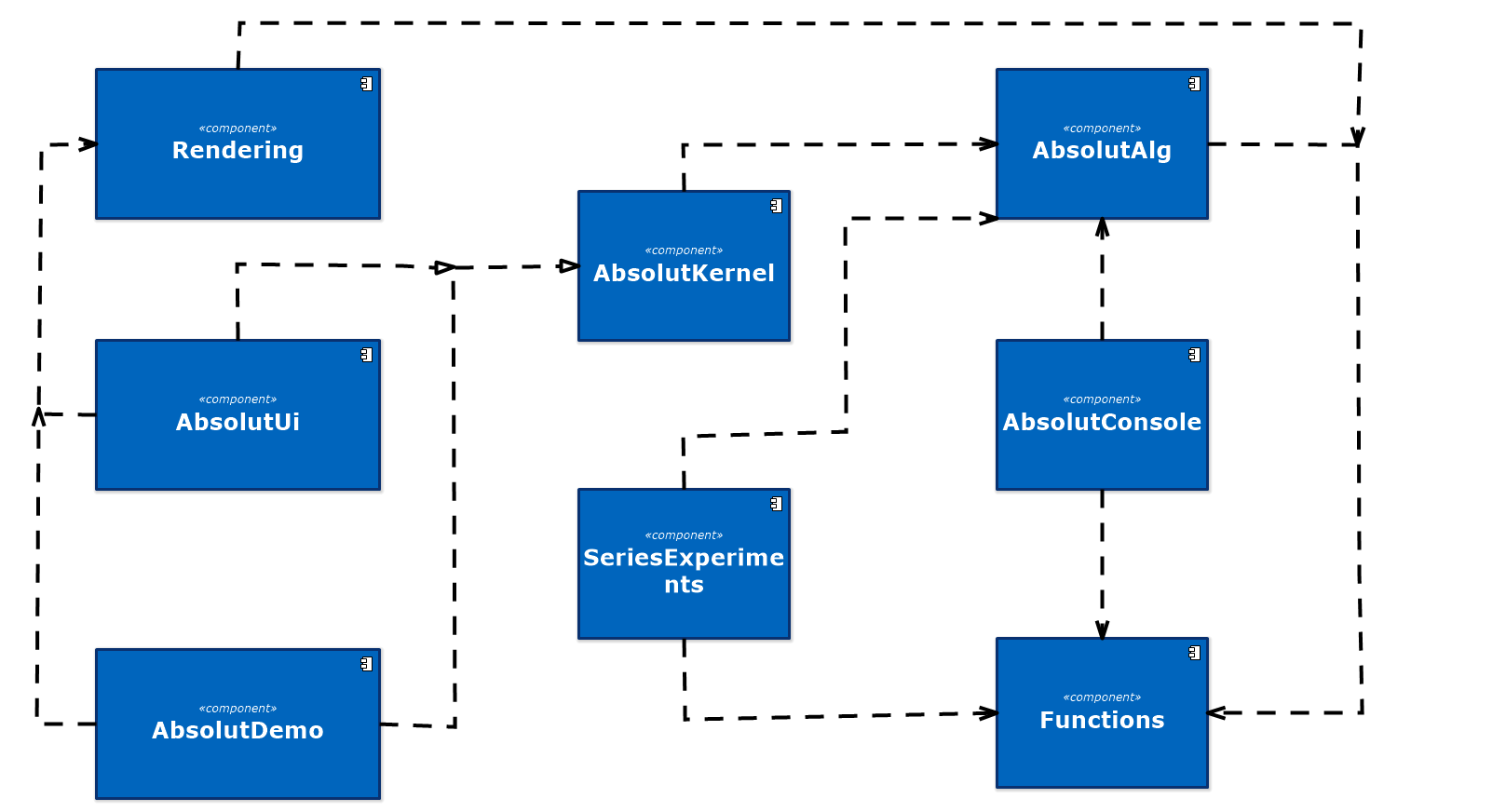


Рисунок 1. Диаграмма компонент

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 OpenGL позволило достичь динамического обновления графика целевой функции, линий уровня и карты температур во время решения задачи вместе с возможностью поворота поверхностей. В качестве основных технологий для создания пользовательских приложений использованы WinForms и WPF.

# 3. Внедрение многокритериальных задач в программный комплекс «Абсолют»

Далее описывается процесс расширения существующего приложения комплекса Абсолют функционалом для решения задач многокритериальной оптимизации и интеграцией с модулем решения многокритериальных задач системы Globalizer. Кратко описано текущее представление задачи математического программирования в системе, необходимые изменения для расширения постановки задачи

# 3.1. Представление задачи многокритериальной оптимизации

В программном комплексе Абсолют задача оптимизации представляется в виде объекта с интерфейсом IProblem (листинг 1), который представляет собой набор функций и их область определения D.

Листинг 1. 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.

Интерфейс задачи оптимизации сходен с форматом задач системы Globalizer для облегчения их интеграции.

Результат испытания численного метода представлен в системе классом MethodPoint (листинг 2).

Листинг 2. 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;

// номер процесса

public int id\_process;

// значения критериев в точке

public List<double> criterions;

// значения ограничений в точке

public List<double> constraints;

}

После внедрения задач с нелинейными ограничениями единственным требуемым изменением в структуре точки испытания становится способ хранения вычисленных значений критериев. Ранее для этих целей использовалось поле y, поэтому в клиентском коде все обращения к нему были заменены на поле Criterions.

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.

Далее рассмотрим вопрос изменения уже реализованных в системе численных методов математического программирования – расчета функции свертки и вычисление области Пеано, если алгоритм оптимизации поддерживает данные возможности.

Так как использование функции свертки и расчет области Пеано могут не поддерживаться большинством алгоритмов (рассчитанных на однокритериальную оптимизацию), был введен механизм поддерживаемых операций.

В интерфейсе ISearchAlg добавлен метод получения поддерживаемых операций и два метода для получения используемой функции свертки и списка точке области Пеано. (листинг 3). Базовый класс численных методов GenericAlg предоставляет пустой список доступных операций и заглушки для операционных методов с предупреждением пользователя о неподдерживаемой операции. Благодаря такому способу не потребовалось значительных изменений уже разработанных алгоритмов.

Листинг 3. Интерфейс ISearchAlg.

// виды поддерживаемых операций

public enum SupportedFeatures

{

Convolution,

ParetoFront

}

public interface ISearchAlg

{

string Name { get; }

int IterationsPerformed { get; }

IProblem Problem { get; set; }

void NextStep();

void PrevStep();

bool HasNext();

int CurrentIter();

MethodPoint LastPoint();

MethodPoint BestPoint();

List<MethodPoint> BestPoints();

void RecalcPoints();

IEnumerable<MethodPoint> CalculatedPoints(int start = 0);

void SetProperty(string name, object value);

void SetProperties(Dictionary<string, object> propsValues);

void SetPropertyWithoutRecalc(string name, object value);

void SetPropertiesWithoutRecalc(Dictionary<string, object> propsValues);

object GetProperty(string name);

List<PropertyInfo> GetPropertiesInfo();

// получение списка поддерживаемых операций

List<SupportedFeatures> GetSupportedFeatures();

// формирование области Парето

List<MethodPoint> GetParetoFront();

// доступ к фунцкии свертки

IFunction GetConvolution();

}

Текущая реализация подразумевает, что в случае работы со свертками алгоритм самостоятельно формирует класс, реализующий интерфейс IFunction с необходимой логикой вычисления значения функции в зависимости от типа используемой свертки. В качестве примера на листинге 4 приведен пример реализации свертки для алгоритмов из библиотеки MOEA.

Листинг 4. Пример расширения алгоритма функцией свертки

class MoeaConvolutionFunction : MultidimFunction

{

private IProblem \_problem;

public MoeaConvolutionFunction(IProblem problem)

{

\_problem = problem;

Left = problem.LowerBound;

Right = problem.UpperBound;

Lambdas = new List<double>();

for (int i = 0; i < problem.NumberOfCriterions; i++)

Lambdas.Add(1.0 / problem.NumberOfCriterions);

}

public override double Calc(List<double> arg)

{

double sum = 0;

int lambdaIndex = 0;

foreach (var problemCriterion in \_problem.Criterions)

{

sum += problemCriterion.Calc(arg) \* Lambdas[lambdaIndex];

lambdaIndex++;

}

return sum;

}

}

# 3.2. Интеграция с системой Globalizer

Интеграция с системой произведена путем запуска консольного приложения Globalizer [13] из системы Абсолют в отдельном потоке с ключом вывода в файл и дальнейшем считывании полученной из файла информации – точки испытания и точки области Парето.

Разбор выходных файлов Globalizer было решено вынести за пределы алгоритма оптимизации, так как и многокритериальный решатель и однокритериальный решатель используют практически одинаковый формат файлов (пример представлен на листинге 5). В начале файла указано количество точек испытания и размерность задачи. Затем для каждой точки испытания указаны координаты точки и после разделителя – вычисленные критерии. Файл с точками области Парето не имеет разделителя и представляет собой строки с координатами точек. Разбор файлов осуществляется в классе ExaminOutputParser, который используется для обоих алгоритмов, связанных с Globalizer.

Листинг 5. Часть выходного файла системы Globalizer

1424 3

-0.968506 -0.968506 | -4.48512 -4.26115 3.494

-0.999756 -0.937256 | -4.51594 -4.27508 3.41096

-0.937256 -0.937256 | -4.5287 -4.32736 3.38522

-0.874756 -0.999756 | -4.45535 -4.2878 3.56302

-0.809814 -0.932373 | -4.53631 -4.41599 3.34199

-0.874756 -0.874756 | -4.60415 -4.44804 3.17939

-0.809326 -0.869873 | -4.60614 -4.49185 3.15464

-0.811279 -0.804932 | -4.67069 -4.56111 2.96849

-0.815674 -0.751221 | -4.71799 -4.61056 2.81002

-0.876709 -0.751709 | -4.72006 -4.57399 2.76798

Так как MCO приложение использует различные ключи командной строки с перечислением номеров функций (для формирования многокритериальной задачи из DLL файла указанием номеров нужных функций), были добавлен новый вид свойств – списки.

В итоге для интеграции новой системы кроме написания разбора файла с областью Парето потребовалось введение нового типа свойств, новая генерация строки запуска консольной системы и дополнительный вызов парсинга файла после расчета. Последним пунктом интеграции стало добавление функции свертки аналогично уже описанному подходу. Полный текст класса доступен в Приложении 1.

# 4. Клиентское приложение для программного комплекса «Абсолют»

Далее описывается процесс расширения существующего приложения комплекса Абсолют клиентским приложением на технологии WPF [7], призванного заменить использованную раннее версию пользовательского интерфейса с целью упрощения пользования приложением и облегчением внедрения новых технологий в дальнейшем.

# 4.1. Стратегия смены технологии пользовательского интерфейса

В комплексе Абсолют пользовательский интерфейс был представлен компонентой View, реализованной при помощи технологии Windows Forms. Ее использование позволило в короткие сроки реализовать большую часть UI и интегрировать OpenGL для сложных визуализаций. Все выполненные на данный момент элементы интерфейса сделаны с учетом Windows Forms (наследуются от соответствующих базовых элементов из этой технологии).

Но при дальнейшей разработке остро стал вопрос перехода с Windows Forms на более современный и поддерживаемый последними версиями операционной системы Windows стек технологий Windows Presentation Foundation. Причинами такого решения стали:

1. Разделение code-behind и разметки визуальной части при помощи отдельного файла разметки на языке XAML (не требуется редактор форм для правок в интерфейсе)
2. Большой выбор сторонних библиотек (в том числе Avalon Dock [8] с функциональностью вкладок для реализации разделения экрана)
3. Широкое использование архитектуры MVVM и возможности Binding для ее реализации.
4. Недостаточная гибкость Windows Forms в плане расположения и управления элементами интерфейса.

Для максимального переиспользования уже наработанного кода применялись следующие принципы:

1. Формы с OpenGL функционалом используются как содержимое контейнеров WPF при помощи WindowsFormsHost.
2. В тех формах с использованием MVVM, ViewModel пользуется уже существующими Presenters для подписки на события и взаимодействием с данными экспериментов.
3. Диалоговые окна в первой версии приложения переиспользуются полностью, на следующем этапе планируется реализация всех диалоговых окон на технологии WPF.
4. Общие элементы интерфейса переносятся в компоненту Rendering, общие Presenters – в ядро системы.
5. Функционал по отображению многокритериальных задач реализуется только в новом приложении.

# 4.2. Реализация приложения на технологии WPF

Новое клиентское приложение состоит из следующих основных элементов:

* Формы
  + Стартовая форма приложения с вкладками для серийных и одиночных экспериментов
  + Форма серийного эксперимента
  + Форма одиночного эксперимента
* Виджеты
  + Таблица точек испытания
  + Отображение функции свертки
  + Пункт меню для функционалов задачи
  + Отображение точек области Парето

Стартовая форма приложения практически полностью повторяет аналогичную форму из прошлой версии GUI - отличия состоят только в классах создаваемых форм и использовании вместо технологии MDI механизма вкладок.

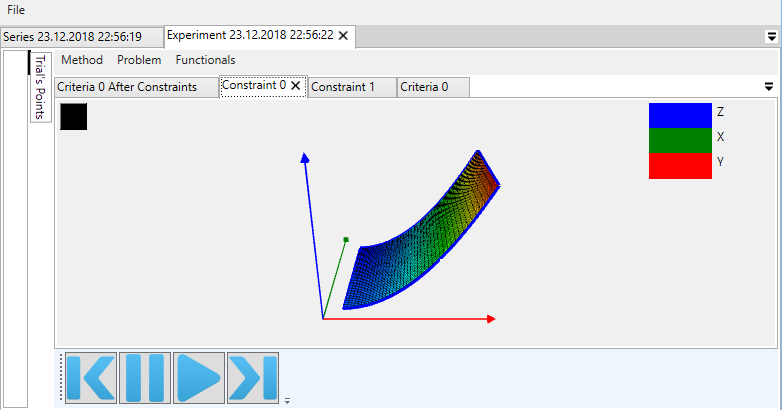


Рисунок 3. Форма эксперимента и механизм вкладок

Форма серийного эксперимента добавляется во вкладку при помощи WindowsFormsHost следующим образом (листинг 6)

Листинг 6. Использование форм WindowsForms как контента элементов WPF

public void ShowNewSeriesExperiment(Guid id, string expName, int expCount)

{

// создание формы серийного эксперимента

var child = new ExperimentsDemo(id, expCount)

{

Text = expName,

TopLevel = false,

FormBorderStyle = FormBorderStyle.None

};

// добавление новой вкладки с формой эксперимента

\_layoutPane.Children.Add(new LayoutDocument()

{

Title = expName,

Content = new WindowsFormsHost()

{

Child = child

}

});

}

Форма одиночного эксперимента была полностью переделана для использования WPF. В ней использована схема MVVM [9]. Набор вкладок и данные для виджетов связаны с полями класса ViewModel. ViewModel использует набор Presenters и интерфейсов View для получения событий от соответствующего эксперимента. При получении информации от Presenters ViewModel обновляет связанные поля, и форма эксперимента отображает актуальные данные. Отдельные виджеты имеют свой файл разметки и свою View Model (например, отображение функции свертки).

Рассмотрим этот процесс на реализации отображения точек границы Пеано. Презентационной слой для этой операции представлен в виде нового класса ParetoFrontPresenter. (листинг 7). Для получения информации от презентационного слоя класс View Model реализует интерфейс IParetoFrontView с двумя базовыми операциями:

* уведомление о том, что алгоритм не поддерживает данную операцию;
* обновить точки Парето-границы (список точек передается аргументом метода).
* обновить аппроксимацию точек Парето-границы (аналогично передается список в аргументы метода)

Если данная операция не поддерживается, View Model очищает списки точек. Реализации методов представлены на листинге 8.

Так как поля представляют собой объекты класса ObservableList, связанные элементы пользовательского интерфейса автоматически уведомляются об изменениях.

Листинг 7. Реализация презентационного слоя получения точек Парето-границы

// EventListenerAdapter содержит пустые реализации методов-обработчиков событий

public class ParetoFrontPresenter : EventListenerAdapter

{

// инициализация модели

private IModel \_m = ModelFactory.Build();

private Guid \_expId;

private IParetoFrontView \_view;

// подписка на эксперимент во время получения его id

public Guid ExpId

{

get

{

return \_expId;

}

set

{

\_expId = value;

\_m.SubscribeExperiment(this, value);

}

}

public ParetoFrontPresenter(IParetoFrontView view)

{

\_view = view;

}

// при изменении задачи необходимо получить

// новую оценку области Парето и передать ее

// в связанное отображение

public override void OnProblemChanged()

{

var searchAlg = this.\_m.GetAlg(this.\_expId);

if (searchAlg.GetSupportedFeatures().Contains(SupportedFeatures.ParetoFront))

{

\_view.UpdatePoints(searchAlg.GetParetoFront());

this.\_view.UpdateApproximatePoints(ParetoFrontHelper.LazyApproximateParetoFront(searchAlg.Problem));

}

else

{

\_view.ParetoNotSupported();

}

}

}

В качестве визуальных отображений была добавлена таблица со списком точек (аналогичная таблица используется для отображения точек испытания) и график расположения точек области Парето (для построения графика используется библиотека OxyPlot [10]).

Вкладки для этих виджетов создаются динамически и связывание с полями View Model так же происходит динамически во время установки алгоритма решения.

В остальных случаях используется стандартный механизм связывания с указанием нужного поля в файле разметки формы .xaml.

Листинг 8. Обновление связанных полей во View Model

public void ParetoNotSupported()

{

ParetoFront.Clear();

ApproximateParetoFront.Clear();

}

public void UpdatePoints(List<MethodPoint> paretoFront)

{

foreach (var methodPoint in paretoFront)

{

ParetoFront.Add(new MethodPointRow(methodPoint));

ApproximateScatterParetoFront.Add(

new ScatterPoint(methodPoint.Criterions[0], methodPoint.Criterions[1]) { Size = 5, Value = 4 });

}

}

public void UpdateApproximatePoints(List<MethodPoint> approximateParetoFront)

{

ApproximateParetoFront.Clear();

foreach (var methodPoint in approximateParetoFront)

{

if(methodPoint.Criterions.Count < 2) return;

ApproximateParetoFront.Add(new DataPoint(methodPoint.Criterions[0], methodPoint.Criterions[1]));

}

}

# Заключение

В ходе выполнения работы архитектура приложения Абсолют расширена новым клиентским приложением и алгоритмами многокритериальной оптимизации.

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

Вся разработка производилась в среде MSVisualStudio на языке С# с использованием библиотек .NET и библиотек для работы с OpenGL. Основной технологией для реализации пользовательского интерфейса стал Windows Presentation Foundation с библиотекой Xceed Toolkit. В качестве системы контроля версий использован Git и AppVeyor для непрерывной интеграции. Unit-тесты выполнены на встроенном в MSVisualStudio фреймворке MSTest.

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. Windows Presentation Foundation (заглавие с экрана) [Электронный ресурс] Режим доступа: [https://docs.microsoft.com/ru-ru/dotnet/framework/wpf](https://docs.microsoft.com/ru-ru/dotnet/framework/wpf/)/ (Дата обращения 23.12.18)
8. Avalon Dock (заглавие с экрана) [Электронный ресурс]. Режим доступа: [https://github.com/xceedsoftware/wpftoolkit/wiki/AvalonDock](https://github.com/xceedsoftware/wpftoolkit/wiki/AvalonDock%20) (Дата обращения 23.12.18)
9. Шаблон Model-View-ViewModel (заглавие с экрана) [Электронный ресурс]. Режим доступа: [https://docs.microsoft.com/ru-ru/xamarin/xamarin-forms/enterprise-application-patterns/mvvm](https://docs.microsoft.com/ru-ru/xamarin/xamarin-forms/enterprise-application-patterns/mvvm%20) (Дата обращения 23.12.18)
10. OxyPlot (заглавие с экрана) [Электронный ресурс]. Режим доступа: [http://www.oxyplot.org/](http://www.oxyplot.org/%20) (Дата обращения 23.12.18)
11. Global Optimization Toolbox (заглавие с экрана) [Электронный ресурс].Режим доступа <https://www.mathworks.com/products/global-optimization.html> (Дата обращения 10.06.2018)
12. Maple Global Optimization Toolbox (заглавие с экрана). [Электронный ресурс]. Режим доступа - <https://www.maplesoft.com/products/toolboxes/globaloptimization/> (Дата обращения 10.06.2018)
13. Practical genetic algorithms / Randy L. Haupt, Sue Ellen Haupt.—2nd ed.
14. 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.

# Приложение 1. Класс ExaminMcoAlg

public class ExaminMcoAlg : ExaminAlg

{

private const string ParetoFrontFileName = "optim\_pareto.txt";

private const string Delimeter = "\_";

private List<MethodPoint> \_paretoFrontPoints = new List<MethodPoint>();

private readonly List<SupportedFeatures> \_features =

new List<SupportedFeatures>() { SupportedFeatures.ParetoFront, SupportedFeatures.Convolution };

private IProblem \_problem1;

public ExaminMcoAlg()

{

// регистрация свойств алгоритма

Name = AbsolutAlgConstants.ExaminMcoMethodName;

RegisterProperty(new DoubleProperty(AbsolutAlgConstants.E, 0.000001, 0.1, 5, DoubleDeltaType.Log, 0.025));

RegisterProperty(new IntProperty(AbsolutAlgConstants.RLD, -100, 100, 1, 1));

RegisterProperty(new IntProperty(AbsolutAlgConstants.M, 1, 20, 1, 12));

var criterionsCount = 3;

RegisterProperty(

new IntProperty(AbsolutAlgConstants.NumberOfCriteria, 0, int.MaxValue, 1, criterionsCount));

RegisterProperty(new ListProperty<int>(AbsolutAlgConstants.FunctionNumber, new List<int>() { 1, 2, 3 }));

RegisterProperty(new IntProperty(AbsolutAlgConstants.ConstraintCount, 0, int.MaxValue, 1, 2));

RegisterProperty(new ListProperty<int>(AbsolutAlgConstants.ConstraintNum, new List<int>() { 4, 5 }));

RegisterProperty(

new DoubleProperty(AbsolutAlgConstants.Delta, 0.000001, 0.1, 5, DoubleDeltaType.Log, 0.025));

var lambdas = new List<double>();

for (int i = 0; i < criterionsCount; i++)

{

lambdas.Add(1.0 / criterionsCount);

}

RegisterProperty(new ListProperty<double>(AbsolutAlgConstants.LambdaList, lambdas));

SetProperty(AbsolutAlgConstants.R, 5.6);

}

// при установке задачи результирующая задача создается с учетом свойств алгоритма

// критерии и ограничения собираются соответственно спискам номеров функций

// в свойствах алгоритма

public override IProblem Problem

{

get

{

return \_problem1;

}

set

{

var constraints = (List<int>)GetProperty(AbsolutAlgConstants.ConstraintNum);

var criterions = (List<int>)GetProperty(AbsolutAlgConstants.FunctionNumber);

\_problem1 = ProblemFactory.BuildProblem(value, criterions, constraints);

}

}

// формирование строки запуска из значений свойств

protected override string BuildCommandLineArguments()

{

var buildCommandLineArguments = "-E " + (double)GetProperty(AbsolutAlgConstants.E) + " -tm MCO\_Method"

+ " -np 1" + " -N 2" + " -rld " + (int)GetProperty(AbsolutAlgConstants.RLD)

+ " -m " + (int)GetProperty(AbsolutAlgConstants.M) + " -r "

+ (double)GetProperty(AbsolutAlgConstants.R) + " -lib "

+ ((string)GetProperty(AbsolutAlgConstants.ExaminDll)) + " -NumberCriteria "

+ (int)GetProperty(AbsolutAlgConstants.NumberOfCriteria)

+ " -function\_number "

+ ((List<int>)GetProperty(AbsolutAlgConstants.FunctionNumber))

.Select(s => s.ToString()).Aggregate((i, j) => i + Delimeter + j)

+ " -constraint\_count "

+ (int)GetProperty(AbsolutAlgConstants.ConstraintCount)

+ " constraintNums "

+ ((List<int>)GetProperty(AbsolutAlgConstants.ConstraintNum))

.Select(s => s.ToString()).Aggregate((i, j) => i + Delimeter + j)

+ "-delta " + (double)GetProperty(AbsolutAlgConstants.Delta) + " -sip "

+ (string)GetProperty(AbsolutAlgConstants.OutputFile);

return buildCommandLineArguments;

}

// после загрузки точек испытания из файла идет парсинг файла с точками Парето

public override void RecalcPoints()

{

base.RecalcPoints();

if (!AreExaminPropertiesNotSet())

ParseParetoFrontFile();

}

// получение функции свертки

public override IFunction GetConvolution()

{

return new ExaminMcoConvolutionFunction(Problem)

{

Lambdas = (List<double>)GetProperty(

AbsolutAlgConstants.LambdaList)

};

}

// разбор файла с точками Парето

private void ParseParetoFrontFile()

{

var path = ParetoFrontFileName;

if (!File.Exists(path)) throw new FileNotFoundException();

var parser = new ExaminOutputParser(Problem);

using (var sr = File.OpenText(path))

{

while (!sr.EndOfStream)

{

var s = sr.ReadLine();

if (sr.EndOfStream)

continue;

var mp = parser.ParseParetoFrontFileLine(s);

mp.IdProcess = 0;

for (int i = 0; i < Problem.NumberOfCriterions; i++)

{

mp.Criterions = Problem.CalculateCriterions(mp.X);

}

\_paretoFrontPoints.Add(mp);

}

}

}

public override List<MethodPoint> GetParetoFront()

{

return \_paretoFrontPoints;

}

// поддерживаемые алгоритмом операции

public override List<SupportedFeatures> GetSupportedFeatures()

{

return \_features;

}

// класс функции свертки

class ExaminMcoConvolutionFunction : MultidimFunction

{

private IProblem \_problem;

public ExaminMcoConvolutionFunction(IProblem problem)

{

\_problem = problem;

Left = problem.LowerBound;

Right = problem.UpperBound;

}

public override double Calc(List<double> arg)

{

double sum = 0;

int lambdaIndex = 0;

foreach (var problemCriterion in \_problem.Criterions)

{

sum += problemCriterion.Calc(arg) \* Lambdas[lambdaIndex];

}

return sum;

}

}

}