1. Разработать алгоритм и составить программу решения задачи линейного программирования с интервальными значениями коэффициентов целевой функции в соответствии с заданиями (приложение 1).

Вариант задания:

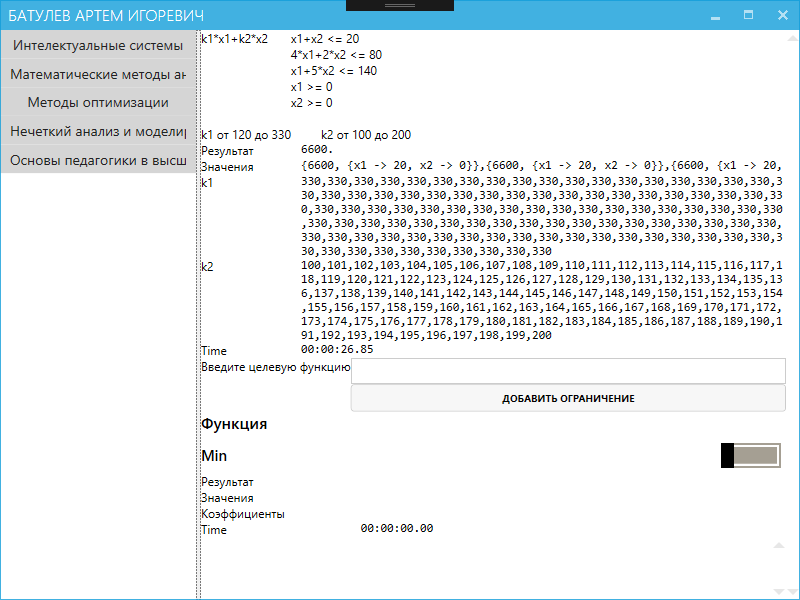


1. Алгоритм решения задачи представлен на рисунке 1.



Рис.1

На рисунке 2 представлена экранная форма с введенными интервальными значениями и ограничениями.

  
Рисунок 2 – Экранная форма с введенными исходными данными

На рисунке 3 представлена экранная форма с результатами решения задачи.

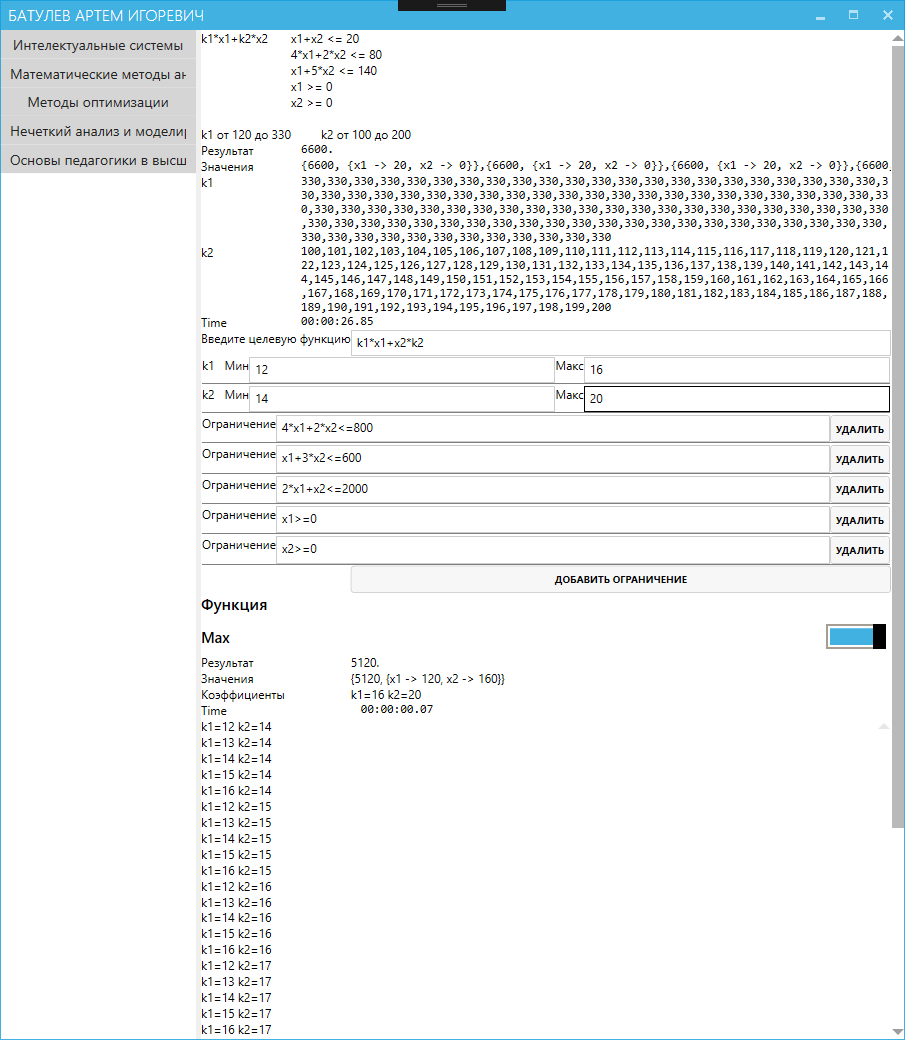


Рисунок 3 – Результаты решения задачи

Результат работы Wolfram Mathematica представлен на рисунке 4

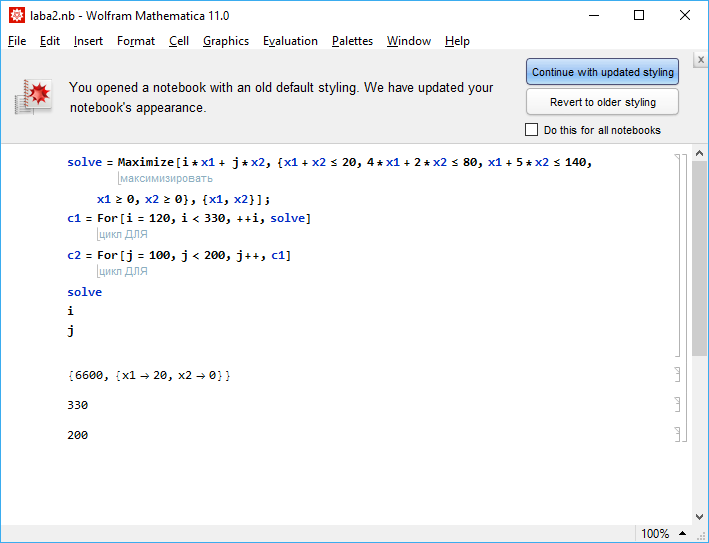


Рисунок 3

Выводы: Программа выводит время выполнения: в данном случае время – 26 c. Решение задачи полностью совпало с решением, выполненным в Wolfram Mathematica.

Приложение А

Текст программы

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Wolfram.NETLink;

namespace OptimizationMethods.ViewModels.Lab2

{

public class Lab2ViewModel : TimeBaseJob

{

private List<ResultType> results = new List<ResultType>();

private string \_result;

public Lab2ViewModel(MathKernel mathKernel)

: base(mathKernel)

{

Job = new Job(mathKernel);

}

public Job Job { get; }

public string Function { get; set; }

public string Result

{

get

{

EvaluateTime(() => \_result = Compute($"Max[{{ {GetResults()} }}]")); ;

OnPropertyChanged(nameof(ResultF));

OnPropertyChanged(nameof(K1));

OnPropertyChanged(nameof(K2));

return \_result;

}

}

public string ResultF => string.Join(",", results.Where(x => x.Max == \_result).Select(x => x.FMax));

public string K1 => string.Join(",", results.Where(x => x.Max == \_result).Select(x => x.k1));

public string K2 => string.Join(",", results.Where(x => x.Max == \_result).Select(x => x.k2));

private string GetResults()

{

results.Clear();

var sb = new StringBuilder();

for (int i = 120; i <= 330; i++)

{

for (int j = 100; j <= 200; j++)

{

var r = new ResultType

{

Max = Compute($"NMaximize[{{ {i} \* x1 + {j} \* x2, x1 + x2 <= 20 && 4 \* x1 + 2 \* x2 <= 80 && x1 + 5 \* x2 <= 140 && x1 >= 0 && x2 >= 0 }},{{x1,x2}}][[1]]"),

//Min =Compute($"NMinimize[{{ {i} \* x1 + {j} \* x2, x1 + x2 <= 20 && 4 \* x1 + 2 \* x2 <= 80 && x1 + 5 \* x2 <= 140 && x1 >= 0 && x2 >= 0 }},{{x1,x2}}][[1]]"),

FMax = Compute($"Maximize[{{ {i} \* x1 + {j} \* x2, x1 + x2 <= 20 && 4 \* x1 + 2 \* x2 <= 80 && x1 + 5 \* x2 <= 140 && x1 >= 0 && x2 >= 0 }},{{x1,x2}}]"),

//FMin =Compute($"Minimize[{{ {i} \* x1 + {j} \* x2, x1 + x2 <= 20 && 4 \* x1 + 2 \* x2 <= 80 && x1 + 5 \* x2 <= 140 && x1 >= 0 && x2 >= 0 }},{{x1,x2}}]"),

k1 = i,

k2 = j

};

results.Add(r);

sb.Append(r.Max + ",");

}

}

var result = sb.ToString();

if (result.Any() && result.Last() == ',')

{

result = result.Remove(result.Length - 1);

}

return result;

}

}

}