Московский авиационный институт

(национальный исследовательский университет)

Отчёт

по курсу «Математическое программирование и оптимальное управление»

Тема: «Решение задачи Динамического программирования I»

Выполнил: студент группы

М7О-407С-20

Понкращенков Д.Б.

Принял:

Барчев Н.Б.

1. Содержательная постановка оптимизационной задачи

Производственное предприятие может выпускать n видов продукции. Для производства используется однородное сырье, имеющееся на предприятии в ограниченном количестве. Известны технологические коэффициенты затрат сырья на производство единицы продукции каждого вида. Прибыль от реализации произведенной продукции каждого вида является монотонно возрастающей функцией с насыщением в зависимости от объема произведенной продукции. Требуется составить производственный план, определяющий объемы производства продукции каждого вида и обеспечивающий максимальную суммарную прибыль предприятия, при выполнении ограничения на используемое сырье.

,

Где:

z – суммарная прибыль предприятия;

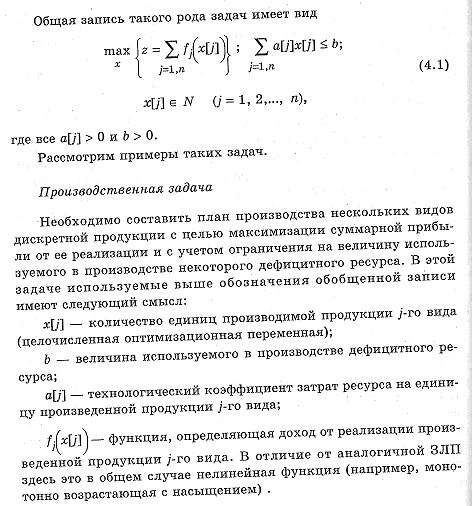
- максимальная прибыль от реализации продукции j-го вида,

 - коэффициент на интервале (0,1),

*x*[j] – объем выпуска продукции j-го вида.

1. Формализованная постановка оптимизационной задачи и ее характеристика

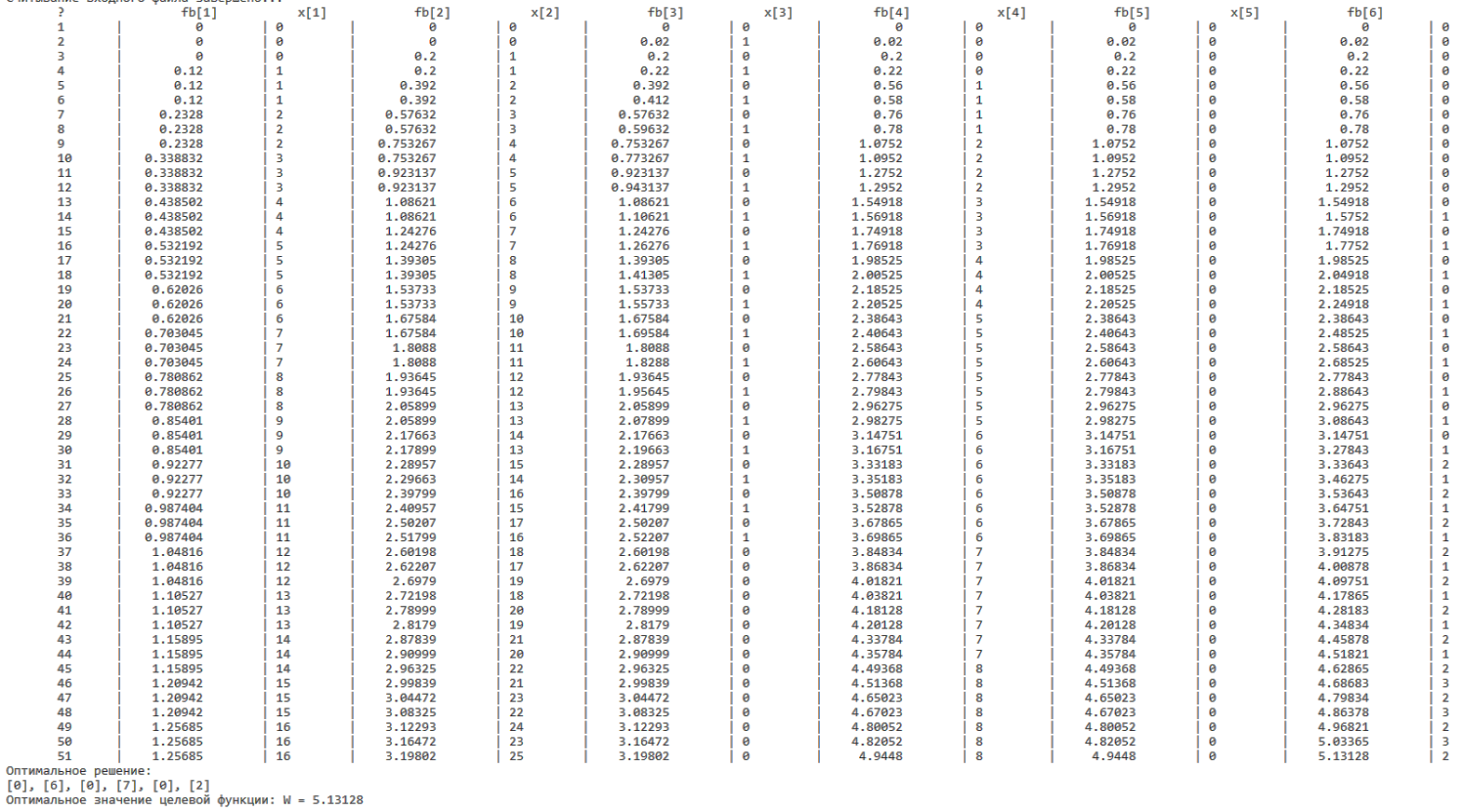
Данная задача относится к задачам с одним типом управлений и одним ограничением.



1. Исходные данные для проведения вычислительных экспериментов

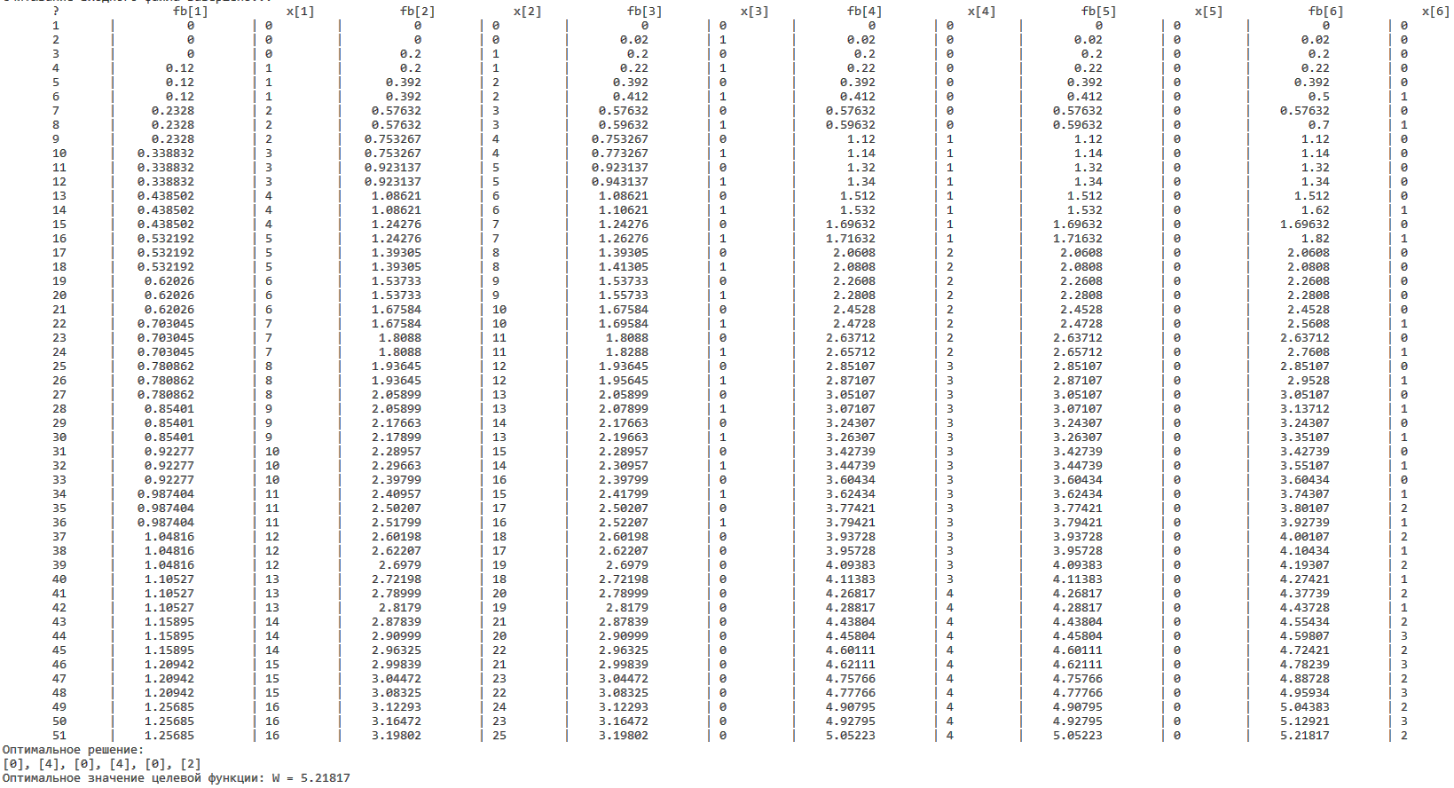
* Набор 1:
  + Количество продукции: n = 6
  + Количество выделенного ресурса: b = 50.
  + Коэффициенты в ограничении (затрат ресурсов на единицу продукции): a(J) = [3,2,1,4,6,5]
  + Коэффициент на интервале (0,1) K(J) = [0.06, 0.04, 0.02, 0.08, 0.12, 0.1]
  + Максимальная прибыль от реализации продукции j-го вида A[j] = [2,5,1,7,3,5]
* Набор 2:
  + Количество продукции: n = 6
  + Количество выделенного ресурса: b = 50.
  + Коэффициенты в ограничении (затрат ресурсов на единицу продукции): a(J) = [3,2,1,8,6,5]
  + Коэффициент на интервале (0,1) K(J) = [0.06, 0.04, 0.02, 0.16, 0.12, 0.1]
  + Максимальная прибыль от реализации продукции j-го вида A[j] = [2,5,1,7,3,5]

1. Результаты вычислительных экспериментов

1-ый набор:

Оптимальное решение: [0,6,0,7,0,2]

Оптимальное значение целевой функции: W = 5.131

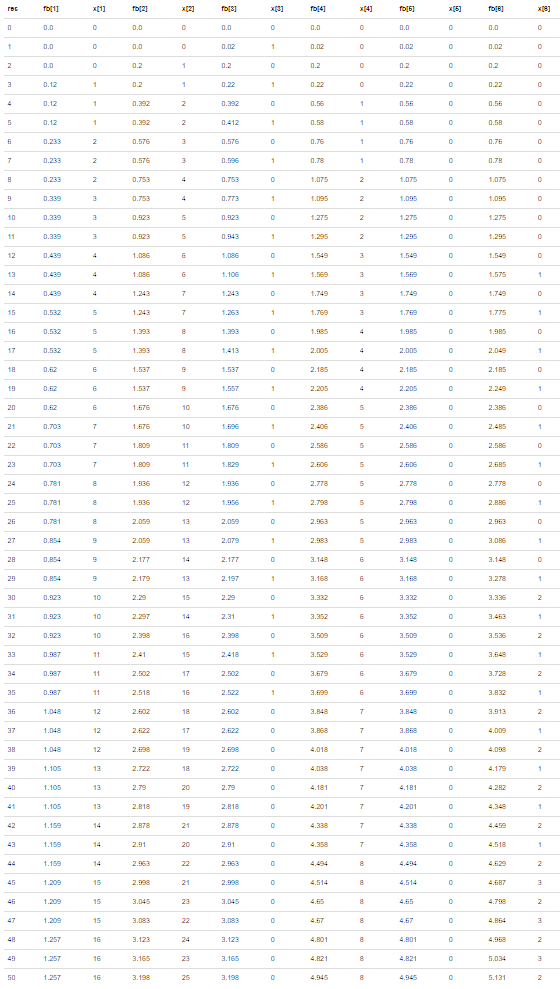
2-ый набор:

Оптимальное решение: [0,4,0,4,0,2]

Оптимальное значение целевой функции: W = 5.21817

Сравнение с результатами на сайте ws-dss:

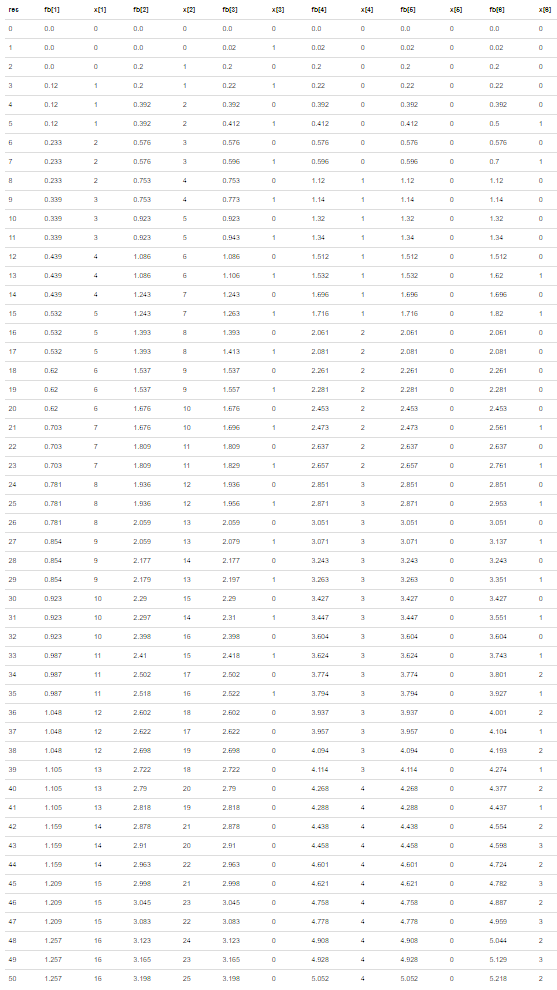
1-ый набор:

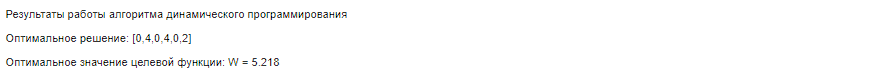




Результаты совпадают.

2-й набор:





Результаты совпадают.

1. Вывод

Оптимальное решение состоит из вектора x[j], который определяет планируемое количество резервных элементов для j-го вида. Параметр  влияет на целевую функцию через степень . При уменьшении  значения степени становятся меньше, что приводит к уменьшению вклада этого слагаемого в целевую функцию. Изменение оптимального решения, таким образом, связано с тем, что уменьшение  приводит к уменьшению влияния компонентов целевой функции. В данном случае, при сильном уменьшении  оптимальным становится использование большего количества резервных элементов, что соответствует менее экономичному расходованию ресурсов. Этот эффект может быть связан с тем, что при уменьшении  понижается общая вероятность безотказной работы системы.

1. Листинг

Файл DynamicProgramm.h:

#pragma once

#include "pch.h"

//--------------------------------------------------------------------------------||

static class DynamicProgramm {

public:

static void Calc(const std::string);

private:

//--------------------------------------------------------------------------------||

#pragma region --- Переменные ---

static int nmax;

static double bmax;

#pragma region --- Выходные переменные ---

/// <summary>

/// Строка с функцией

/// </summary>

static std::string function;

/// <summary>

/// Количество дефицитного ресурса

/// </summary>

/// <param name=""></param>

static int MaxResource;

/// <summary>

/// Количество видов производимой продукции

/// </summary>

static int CountProduct;

/// <summary>

/// Список ограничивающих ограничений

/// </summary>

static std::vector<int> Limitations;

/// <summary>

/// Коэф-ты потребления (проц)

/// </summary>

static std::vector <double> Kj;

/// <summary>

/// Прибыль от реализации продукции

/// </summary>

static std::vector <double> Aj;

/// <summary>

/// Объем выпускаа продукции

/// </summary>

#pragma endregion

/// <summary>

/// Таблица оптимальных управлений.

/// </summary>

static std::vector<std::vector<int>> xj;

/// <summary>

/// Таблица условно оптимальных управлений.

/// </summary>

static std::vector<std::vector<int>> noc;

/// <summary>

/// Таблица значений функции Беллмана.

/// </summary>

static std::vector<std::vector<double>> bellmanFunction;

#pragma endregion

//--------------------------------------------------------------------------------||

/// <summary>

/// Расчет функции Беллмана

/// </summary>

static void FuncBellman();

/// <summary>

/// Расчет значения функции

/// </summary>

/// <param name="nper">Номер переменной</param>

/// <param name="res">Выделенный ресурс</param>

/// <returns></returns>

static double Func(int, int);

/// <summary>

/// Возвращает максимальное значение в столбце

/// </summary>

static void getMax(std::vector<double>, int);

//--------------------------------------------------------------------------------||

/// <summary>

/// Считывает входной файл в формате json

/// </summary>

static void ReadInput(const std::string);

/// <summary>

/// Считывает входной файл в формате json

/// </summary>

static void PrintBellman();

//--------------------------------------------------------------------------------||

};

Файл DynamicProgramm.cpp:

#include "pch.h"

#include "DynamicProgramm.h";

//--------------------------------------------------------------------------------||

std::string DynamicProgramm::function = "";

int DynamicProgramm::MaxResource = 0;

int DynamicProgramm::CountProduct = 0;

int DynamicProgramm::nmax = 0;

double DynamicProgramm::bmax = 0;

std::vector<int> DynamicProgramm::Limitations;

std::vector <double> DynamicProgramm::Kj;

std::vector <double> DynamicProgramm::Aj;

std::vector<std::vector<int>> DynamicProgramm::xj;

std::vector<std::vector<int>> DynamicProgramm::noc;

std::vector<std::vector<double>> DynamicProgramm::bellmanFunction;

//--------------------------------------------------------------------------------||

void DynamicProgramm::Calc(const std::string file) {

ReadInput(file);

CountProduct = Limitations.size();

for (int i = 0; i <= MaxResource; i++) {

bellmanFunction.push\_back(std::vector<double>());

noc.push\_back(std::vector <int>());

for (int j = 0; j < CountProduct; j++) {

noc[i].push\_back(0);

bellmanFunction[i].push\_back(0);

}

}

FuncBellman();

PrintBellman();

std::vector <std::vector<int>> ksi;

for (int i = 0; i <= MaxResource; i++) {

ksi.push\_back(std::vector<int>());

xj.push\_back(std::vector<int>());

for (int j = 0; j < CountProduct; j++) {

ksi[i].push\_back(0);

xj[i].push\_back(0);

}

}

for (int i = 0; i <= MaxResource; i++) {

xj[i][CountProduct - 1] = noc[i][CountProduct - 1];

ksi[i][CountProduct - 1] = i+1;

for (int j = CountProduct - 1; j >= 1; j--) {

ksi[i][j - 1] = ksi[i][j] - Limitations[j] \* xj[i][j];

xj[i][j - 1] = noc[ksi[i][j - 1] - 1][j - 1];

}

}

std::string optX = "[";

for (auto num : xj[xj.size() - 1]) {

optX += std::to\_string(num) + "], [";

}

optX = optX.substr(0, optX.size() - 3);

std::cout << "\n Оптимальное решение:\n " + optX + "\n Оптимальное значение целевой функции: W = " << bellmanFunction[bellmanFunction.size() - 1][CountProduct - 1];

}

//--------------------------------------------------------------------------------||

void DynamicProgramm::FuncBellman() {

int lh = 0;

std::vector<double> wr;

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

wr.push\_back(0);

for (int i = 1; i <= MaxResource + 1; i++) {

lh = (i - 1) / Limitations[0] + 1;

if (lh != 0) {

for (int j = 0; j < lh; j++) {

wr[j] = Func(0, j);

}

getMax(wr, lh);

bellmanFunction[i - 1][0] = bmax;

noc[i - 1][0] = nmax - 1;

}

}

for (int nper = 1; nper < CountProduct; nper++) {

for (int i = 0; i <= MaxResource; i++) {

lh = i / Limitations[nper] + 1;

for (int j = 0; j < lh; j++) {

int tmp = i + 1 - (j + 1) \* Limitations[nper] + Limitations[nper];

wr[j] = Func(nper, j) + bellmanFunction[tmp - 1][nper - 1];

}

getMax(wr, lh);

bellmanFunction[i][nper] = bmax;

noc[i][nper] = nmax - 1;

}

}

}

double DynamicProgramm::Func(int nper, int res) {

return Aj[nper] \* (1 - pow(1 - Kj[nper], res));

}

void DynamicProgramm::getMax(std::vector<double> wr, int lh) {

nmax = 1;

bmax = wr[0];

if (lh == 1) return;

for (int i = 1; i < lh; i++) {

if (wr[i] > bmax) {

nmax = i + 1;

bmax = wr[i];

}

}

}

//--------------------------------------------------------------------------------||

void DynamicProgramm::ReadInput(const std::string path) {

std::ifstream ifile(path);

if (ifile.is\_open()) {

std::string buff;

std::cout << "\n Считывание входного файла...";

while (ifile.eof(), ifile >> buff) {

// Считывание ограничений

if (buff == "\"a\":") {

ifile >> buff;

buff = buff.substr(1, buff.size());

while (buff.rfind(']') == -1) {

buff = buff.substr(0, buff.size() - 1);

Limitations.push\_back(std::stoi(buff));

ifile >> buff;

}

// Добавление последнего элемента

buff = buff.substr(0, buff.size() - 2);

Limitations.push\_back(std::stoi(buff));

}

// Считывание прибыли от реализации

else if (buff == "\"A\":") {

ifile >> buff;

buff = buff.substr(1, buff.size());

while (buff.rfind(']') == -1) {

buff = buff.substr(0, buff.size() - 1);

Aj.push\_back(std::stoi(buff));

ifile >> buff;

}

// Добавление последнего элемента

buff = buff.substr(0, buff.size() - 2);

Aj.push\_back(std::stoi(buff));

}

// Считывание ограничений

else if (buff == "\"K\":") {

ifile >> buff;

buff = buff.substr(1, buff.size());

while (buff.rfind(']') == -1) {

buff = buff.substr(0, buff.size() - 1);

Kj.push\_back(std::stod(buff));

ifile >> buff;

}

// Добавление последнего элемента

buff = buff.substr(0, buff.size() - 2);

Kj.push\_back(std::stod(buff));

}

// Считывание кол-ва ресурсов

else if (buff == "\"max\_res\":") {

ifile >> buff;

MaxResource = std::stoi(buff.substr(0, buff.size() - 1));

}

// Считывание функции

else if (buff == "\"F\":") {

ifile >> buff;

function = buff.substr(1, buff.size() - 1);

}

}

std::cout << "\n Считывание входного файла завершено...";

}

else {

std::cout << "\n Не удалось открыть входной файл: " << path;

}

}

void DynamicProgramm::PrintBellman() {

std::cout << "\n\tξ\t\t fb[1]\t\t x[1]\t\t fb[2]\t\t x[2]\t\t fb[3]\t\t x[3]";

std::cout << "\t\t fb[4]\t\t x[4]\t\t fb[5]\t\t x[5]\t\t fb[6]\t\t x[6]";

for (int row = 0; row <= MaxResource; row++) {

std::cout << "\n\t" << row + 1;

for (int prod = 0; prod < CountProduct; prod++) {

std::cout << "\t| " << std::setw(10) << bellmanFunction[row][prod];

std::cout << std::setw(10) << " | " << noc[row][prod] << std::setw(10);

}

}

}

//--------------------------------------------------------------------------------||