**Введение**

В современном мире существует огромное количество предприятий, использующих для производства своей продукции машинное оборудование. При внедрении данного оборудования на предприятии, составляется оптимальный план использования и замены(ремонта).

Для составления оптимального плана необходимо применять методы динамического программирования, так как это позволяет максимизировать прибыль или минимизировать затраты.

Цель этого курсового проекта <!------------------ >

Задача о замене оборудования состоит в определении оптимальных сроков профилактических работ. Невыполнение данных сроков может привести к снижению качества выпускаемой продукции, к частым поломкам оборудования. Критерием для оптимизации возьмём сумму квадратов отклонений трудоёмкости в n-м месяце от средней трудоёмкости, умноженной на коэффициент корректировки.

Обозначения в тексте

M – количество типов различных устройств, подлежащих профилактической замене в календарном планировании;

k – тип устройства (k = 1, … , M);

Ck – трудоемкость обслуживания одной единицы k-го типа;

rn – коэффициент корректировки трудоёмкости (1 ≥ ? ? ≥ 0);

N – количество единиц планового периода;

n – единица планового периода (n = 1, …, N), в дальнейшем условно будем называть месяцем для удобства;

Vn – трудоёмкость в n-м интервале планового периода;

Vср – средняя трудоемкость за весь плановый период;

µik – время эксплуатации i-го устройства k-го типа на начало планового

периода;

bikn – плановая замена i-го устройства k-го типа в n-й интервал обслуживания;

lk – нормативный срок эксплуатации устройства k-го типа;

t1k – предельный допуск на уменьшение срока замены;

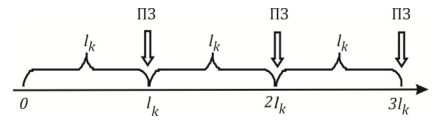
t2k – предельный допуск на увеличение срока замены;

mk – число устройств k-го типа;

i = i1, (△i), i2 – означает, что индекс i меняется с шагом △i от i1 до i2.

**Постановка задачи**

Задача профилактической замены (ремонта, обслуживания) устройств k-го типа оборудования заключается в плановой замене (ПЗ) через каждые интервалы времени Lk (рис. 1), которые задаются нормативно. Жёстко заданные сроки не позволяют равномерно распределить трудозатраты на весь планируемый период, что приводит к авралам в отдельные периоды времени и простоям в другие.



**Рис. 1.** Замена оборудования через интервалы равной длительности.

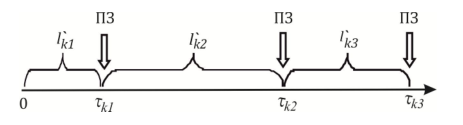
Поэтому изначально полезно учитывать интервал, в котором можно варьировать сроками ПЗ. Допускается начать ПЗ раньше на величину t1k или позже на величину t2k , то есть начало ПЗ можно выбрать в интервале

(1)

Где - срок эксплуатации до i - ой замены k-го устройства по новому плану.

В этом случае возможна оптимизация календарного плана, которая заключается в выборе величин ki , определяющих срок замены от начала планового периода:

(2)

Тогда ПЗ(рис. 2) производится через выбранные интервалы , которые оптимизируют процесс замены по заданным критериям.

**Рис. 2.** Замена оборудования через интервалы разной длительности

**Математическая модель задачи**

Пусть

(3)

Первоначальные план ( до оптимизации ) по замене оборудования

(4)

означает, что i – е устройство k – го типа должно быть заменено в единицу времени n. Однако этот план не является оптимальным, так как трудоёмкости в каждом месяце, которые рассчитываются по формуле:

(5)

могут значительно отличаться от средней величины :

(6)

Задача оптимизации заключается в составлении нового плана

(7)

Трудоемкость в каждый период времени , в соответствии с новым ПЗ вычисляется по формуле:

(8)

В (7) элементы имеют значение:

(9)

Теперь введём переменную булевого типа:

(10)

Если переменная по значению равна единице, то замена i-го устройства k-го переносится с n-й единицы планового периода на j-й. При этом должно выполняться соотношение:

(11)

где

i = 1, … , ; k = 1, … , M;

*;*

*;*

*;*

Соотношение (11) получено из следующих соображений . Значения элементов и совпадают. Если замена переносится с *p*- го месяца на *q* – й, то следует предусмотреть перенос замены с каждого на

месяц. Если n > , то учитываем возможность замены устройства в месяце ( n - ). То есть если в n-м месяце уже запланирована замена, то можно её передвинуть на более ранний или поздний срок. Фактически перенос замены возможен только в том случае, когда правая часть в уравнении (11) равна единице.

Если переменные (10) определены и удовлетворяют условиям (11), то элементы матрицы (7) можно определить как

(12)

то есть учесть все переносы на месяц n, где

*;*

*.*

В качестве критерия для оптимизации возьмем сумму квадратов отклонений трудоёмкости в n-м месяце (8) от средней трудоёмкости (6), умноженной на коэффициент корректировки:

(13)

Подставляя в (13) вместо по формуле (12) получаем:

(14)

В (14) X – это матрица переменных рассматриваемой задачи:

*(15)*

*.*

Таким образом, { ( 11 ) , ( 14 ) , ( 15 ) } – это задача целочисленного программирования с булевыми переменными и квадратичной функцией цели.

**Алгоритм решения**

1. Вычислить трудоёмкость замены устройств в каждом месяце планового периода и среднюю трудоёмкость :

.

Присвоить элементам матрицы (7) первоначальные значения :

, а величинам (10) значения

1. Выполнить п.3, меняя индекс k от 1 до М. В конце перейти к п.11.
2. Выполнить п.4, меняя индекс n от 1 до (N - 1).
3. Вычислить величины s и s’ :

Здесь плановая средняя трудоемкость в n- м месяце. Далее генерируется случайное число α по равномерному закону в интервале

(0,1); вычисляется величина

для определения разницы между средней планируемой трудоемкостью в месяце n и текущей трудоёмкостью:

Перейти к =

1. Выполнить п.6, меняя
2. Выполнить п.7, меняя индекс
3. Если

Меняя

Меняя

Вычислить Если , иначе выполнить .

1. Выполнить п.9 меняя индекс
2. Выполнить п.10 меняя индекс
3. Если , то перейти к п.8, иначе { если , то перейти к п.9, иначе

Меняя

Меняя

Если , то перейти к п.3, иначе выполнить .

1. Выход

**Заключение**

**Литература**

1. Барабаш П.И. Комбинаторный алгоритм решения задачи календарного планирования профилактических работ в здравоохранении // Матер. науч.-практ. конф. "Актуальные проблемы медицинской информатики на Дальнем Востоке". Хабаровск: ДВГМУ, 2011. С. 272-273.

2. Жардин Э. Техническое обслуживание оборудования // В кн. "Исследование операций". Т. 2. Модели и применение / Под ред. Дж. Моудера, С. Элмаграби. М.: Мир, 1981.С. 344-363

3. Корбут А., Финкельштейн Ю. Дискретное программирование. М.: Наука, 1969. 368 с.

4. Кофман А., Анри-Лабордер А. Методы и модели исследования операций. Целочисленное программирование. М.: Мир, 1977. 251 с.

**Приложение**

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication5

{

class Program

{

public static double F\_min(double t\_k, double N, double n)

{

if (t\_k < N - n)

{

return t\_k;

}

else

{

return N - n;

}

}

static void Main(string[] args)

{

int N;

Console.WriteLine("Введите кол-во единиц периодов: ");

N = Int32.Parse(Console.ReadLine());

int M;

Console.WriteLine("Введите кол-во типов различных устройств: ");

M = Int32.Parse(Console.ReadLine());

List<int> Mk = new List<int>();

List<int> L\_k = new List<int>();

List<double> C\_k = new List<double>();

List<double> R\_n = new List<double>();

List<double> t\_1k = new List<double>();

List<double> t\_2k = new List<double>();

List<double> V\_n = new List<double>();

string strC\_k;

string strL\_k;

string strt1\_k;

string strt2\_k;

string strMk;

string strR\_n;

string strMU\_ik;

string[] parametrC\_k;

string[] parametrL\_k;

string[] parametrt1\_k;

string[] parametrt2\_k;

string[] parametrMk;

string[] parametrR\_n;

string[] parametrMU\_ik;

double V\_sr;

Random rand = new Random();

double alpha;

double s, ss;

double w;

double V\_alpha;

double d;

bool logic\_p3 = false;

bool logic\_p3\_2 = false;

Console.WriteLine("Введите кол-во устройств k-го типа: ");

strMk = Console.ReadLine();

parametrMk = strMk.Split(' ', ';');

int sumApparatus = 0;

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

{

Mk.Add(Int32.Parse(parametrMk[i]));

sumApparatus += Mk[i];

}

List<double> MU\_ik = new List<double>();

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

{

Console.WriteLine("Введите время эксплуатации i-го устройства " + (i + 1) + "-го типа:");

strMU\_ik = Console.ReadLine();

parametrMU\_ik = strMU\_ik.Split(' ', ';');

for (int j = 0; j < Mk[i]; j++)

{

MU\_ik.Add(Double.Parse(parametrMU\_ik[j]));

}

}

Console.WriteLine("Введите нормативный срок эксплуатации устройства k-го типа: ");

strL\_k = Console.ReadLine();

parametrL\_k = strL\_k.Split(' ', ';');

Console.WriteLine("Введите трудоемкость k-го типа: ");

strC\_k = Console.ReadLine();

parametrC\_k = strC\_k.Split(' ', ';');

Console.WriteLine("Введите предельный допуск на уменьшение срока замены k-го типа: ");

strt1\_k = Console.ReadLine();

parametrt1\_k = strt1\_k.Split(' ', ';');

Console.WriteLine("Введите предельный допуск на увеличение срока замены k-го типа: ");

strt2\_k = Console.ReadLine();

parametrt2\_k = strt2\_k.Split(' ', ';');

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

{

L\_k.Add(Int32.Parse(parametrL\_k[i]));

C\_k.Add(Double.Parse(parametrC\_k[i]));

t\_1k.Add(Double.Parse(parametrt1\_k[i]));

t\_2k.Add(Double.Parse(parametrt2\_k[i]));

}

Console.WriteLine("Введите коэффициент корректировки трудоёмкости: ");

strR\_n = Console.ReadLine();

parametrR\_n = strR\_n.Split(' ', ';');

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

{

R\_n.Add(Double.Parse(parametrR\_n[i]));

}

int[,] Aikn = new int[sumApparatus, N + 1];

int k = 1, count = 0, tempCount = 0;

for (int j = 0; j < M; j++)

{

tempCount = 0;

for (int i = 0; i < Mk[j]; i++)

{

Aikn[i + count, 0] = k;

tempCount++;

}

k++;

count += tempCount;

}

/\* Aikn for (int j = 0; j < Aikn.GetLength(0); j++)

{

for (int i = 0; i < Aikn.GetLength(1); i++)

{

Console.Write(Aikn[j, i]);

}

Console.WriteLine();

}\*/

for (int ki = 0; ki < Aikn.GetLength(0); ki++)

{

int tempAik = 0;

for (int month = 1; month <= N; month++)

{

if ((month + MU\_ik[ki] - 1) % L\_k[Aikn[ki, 0] - 1] == 0)

{

tempAik = 1;

}

else

{

tempAik = 0;

}

Aikn[ki, month] = tempAik;

}

}

int[,] Bikn = new int[Aikn.GetLength(0), Aikn.GetLength(1)];

Console.WriteLine("Aikn = ");

for (int i = 0; i < Aikn.GetLength(0); i++) // bikn = aikn;

{

for (int j = 0; j < Aikn.GetLength(1); j++)

{

Console.Write(Aikn[i, j] + " ");

Bikn[i, j] = Aikn[i, j];

}

Console.WriteLine();

}

/\* Vn \*/

double tempSumAikn = 0;

for (int i = 1; i < Aikn.GetLength(1); i++)

{

for (int j = 0; j < Aikn.GetLength(0); j++)

{

tempSumAikn += C\_k[Aikn[j,0] - 1]\*Aikn[j, i];

}

V\_n.Add(tempSumAikn);

tempSumAikn = 0;

Console.WriteLine(V\_n[i - 1] + " ");

}

V\_sr = (1.0 / N) \* (V\_n.Sum());

Console.WriteLine("V\_sr: " + V\_sr);

/\* Xiknp \*/

int[,,] Xiknp = new int[sumApparatus, N + 1, N + 1];

for ( int i = 0; i < Xiknp.GetLength(0); i++)

{

for( int n = 1; n < Xiknp.GetLength(1); n++)

{

for(int p = 1; p <Xiknp.GetLength(2); p++)

{

if ( p == n)

{

Xiknp[i, n, p] = Aikn[i, n];

}

else

{

Xiknp[i, n, p] = 0;

}

Xiknp[i, 0, 0] = Aikn[i, 0];

}

}

}

for (int i = 0; i < Xiknp.GetLength(0); i++)

{

for (int n = 0; n < Xiknp.GetLength(1); n++)

{

for (int p = 0; p < Xiknp.GetLength(2); p++)

{

Console.Write(Xiknp[i, n, p] +" ");

}

Console.WriteLine();

}

Console.WriteLine();

}

Console.WriteLine("-------------------------------");

count = 0;

int count2 = 0;

/\* п. 2 \*/

for (k = 1; k <= M; k++)

{

/\* п.3 \*/

for (int n = 1; n <= N - 1; n++)

{

logic\_p3\_2 = false;

logic\_p3 = false;

s = V\_sr \* (R\_n[n - 1] / C\_k[k - 1]);

ss = (V\_sr \* R\_n[n - 1]) % C\_k[k - 1];

alpha = rand.NextDouble();

Console.WriteLine("alpha: " + alpha);

if (alpha > s - ss)

{

V\_alpha = ss \* C\_k[k - 1];

}

else

{

V\_alpha = (ss + 1) \* C\_k[k - 1];

}

w = V\_alpha \* R\_n[n - 1] - V\_n[n - 1];

if (Math.Abs(w) < C\_k[k - 1])

{

// =>>> p3; ++++

}

else if (w > 0)

{

// =>>> p5; ++++

for (int j = 1; j <= F\_min(t\_1k[k - 1], N, n); j++)

{

tempCount = 0;

/\* п.6 \*/

for (int i = 1; i <= Mk[k - 1]; i++)

{

tempCount++;

/\* п. 7 \*/

if (Bikn[i + count - 1, n] == 1 || Bikn[i - 1, n + j] == 0)

{

}

else

{

for (int lamda = n; lamda < N; lamda += L\_k[k - 1])

{

Bikn[i + count - 1, lamda] = 1;

V\_n[lamda - 1] += C\_k[k - 1];

}

for (int lamda = n + j; lamda < N; lamda += L\_k[k - 1])

{

Xiknp[i + count - 1, lamda, lamda] = 0;//? index

Xiknp[i + count - 1, lamda, lamda - j] = 1;//? index

Bikn[i + count - 1, lamda] = 0;

V\_n[lamda - 1] -= C\_k[k - 1];

}

d = w - C\_k[k - 1];

if (d < C\_k[k - 1])

{

// =>>> p.3;

logic\_p3 = true;

break;

}

else

{

w = d;

Console.WriteLine("finish");

Environment.Exit(0);

}

}

}

count += Mk[k - 1]+1;

if (logic\_p3)

{

break;

}

}

}

else if (w <= 0)

{

// =>>> p8;

for (int i = 1; i <= Mk[k - 1]; i++)

{

tempCount = 0;

/\* п.9 \*/

for (int j = 1; j <= F\_min(t\_2k[k - 1], N, n); j++)

{

tempCount++;

if (Bikn[i + count2 - 1, n] == 0)

{

// k p.8++++

break;

}

else

{

if (Xiknp[i + count2 - 1, n, n + j - 1] == 0)

{

// k p.9 +++

}

else

{

d = w + C\_k[k - 1];

for (int lamda = n + 1; lamda < N; lamda += L\_k[k - 1])

{

Xiknp[i + count2 - 1, lamda, lamda + j] = 1;

Bikn[i + count2 - 1, lamda] = 1;

V\_n[lamda - 1] += C\_k[k - 1];

}

for (int lamda = n; lamda < N; lamda += L\_k[k - 1])

{

Xiknp[i + count2 - 1, lamda, lamda - 1] = 0;

Bikn[i + count2 - 1, lamda] = 0;

V\_n[lamda - 1] -= C\_k[k - 1];

}

if (d < C\_k[k - 1])

{

// k p.3

logic\_p3\_2 = true;

break;

}

else

{

w = d;

Console.WriteLine("finish");

Environment.Exit(0);

}

}

}

}

if(logic\_p3\_2)

{

break;

}

}

}

}

count2 += Mk[k - 1];

}

for (int i = 0; i < Xiknp.GetLength(0); i++)

{

for (int n = 0; n < Xiknp.GetLength(1); n++)

{

for (int p = 0; p < Xiknp.GetLength(2); p++)

{

Console.Write(Xiknp[i, n, p] + " ");

}

Console.WriteLine();

}

Console.WriteLine();

}

Console.WriteLine("------------------------");

Console.WriteLine("Bikn = ");

for (int i = 0; i < Bikn.GetLength(0); i++)

{

for (int j = 0; j < Bikn.GetLength(1); j++)

{

Console.Write(Bikn[i, j] + " ");

}

Console.WriteLine();

}

}

}

}