**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **КНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 2

**З дисципліни:** *“Дослідження операцій”*

**На тему:** *“Двоїстий симплекс-метод. Зв’язок між розв’язками*

*прямої та двоїстої задач ЛП”*

**Лектор:**

доц. каф. ПЗ

Журавчак Л. М.

**Виконав:**

ст. гр. ПЗ-23

Михалевич П.-І.В.

**Прийняла:**

асист. каф. ПЗ

Івасько Н. М.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_\_ .

Львів – 2022

**Тема роботи:** Двоїстий симплекс-метод. Зв’язок між розв’язками прямої та двоїстої задач ЛП.

**Мета роботи:** Ознайомитись на практиці із двоїстими задачами лінійного програмування та навчитись розв’язувати їх із використанням двоїстого симплекс-методу.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

18. Який розв’язок називають надоптимальним?

Кращий ніж оптимальний, проте недопустимий.

3. Якою є двоїста до двоїстої задачі лінійного програмування?

Пряма.

4. Який алгоритм побудови двоїстої задачі?

Крок 1. Знаходження початкового надоптимального розв'язку.

Крок 2. Перевірка біжучого надоптимального розв'язку на припустимість (якщо в стовпчику P0 відсутні від'ємні значення, то знайдено оптимальний припустимий розв'язок. Кінець)

Крок 3. Обираємо найбільше за модулем від'ємне значення у стовпчику P0 і визначаємо його індекс:



змінну x[k] виключаємо з бази. Визначаємо індекс змінної, яку включаємо до бази:



Отже, провідний елемент a[k][r].

Крок 4. Переходимо до наступного надоптимального розв’язку так само, як і в звичайному СМ, використовуючи схему Гауса-Жордана (правило трикутника). Перехід до кроку 2.

**ЗАВДАННЯ**

1. Отримати індивідуальний варіант завдання.

2. Записати математичну модель задачі ЛП (згідно з варіантом з Додатка № 1 до лабораторної роботи № 1) і побудувати до неї двоїсту задачу. Зазначити економічний зміст цільової функції і системи обмежень обидвох задач.

3. Написати програму розв’язування задачі ЛП двоїстим симплексметодом та за її допомогою знайти розв’язок (максимальне (мінімальне) значення функції та значення змінних, при якому воно досягається) побудованої двоїстої задачі.

4. Знайти оптимальний план прямої задачі, використовуючи зв’язок між розв’язками прямої та двоїстої задачі. Порівняйте отриманий розв’язок з відповідними результатами лабораторної роботи №1.

5. Дати економічне тлумачення основним і додатковим змінним вихідної та двоїстої задач.

6. Оформити звіт про виконану роботу.

7. Продемонструвати викладачеві результати, відповісти на запитання стосовно виконання роботи.

**ХІД ВИКОНАННЯ**

1. Індивідуальне завдання з додатка №1.

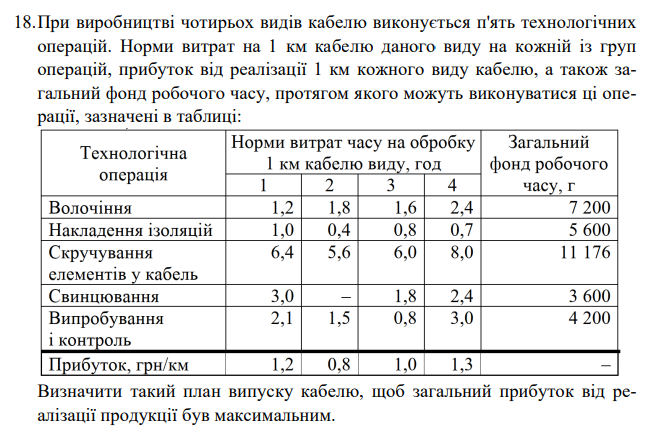


Рис. 1. Задача з додатка №1

1. Математична модель задачі ЛП:

F = 1.2 + 0.8 + + 1.3 🡺 Max;

Максимізація прибутку.

x1 – кількість кілометрів кабеля першого виду.

x2 – кількість кілометрів кабеля другого виду.

x3 – кількість кілометрів кабеля третього виду.

x4 – кількість кілометрів кабеля четвертого виду.

Обмеження по загальному фонду робочого часу для кожного виду технологічної операції:

1.2 + 1.8 + 1.6 + 2.4 <= 7200

+ 0.4 + 0.8 + 0.7 <= 5600

6.4 + 5.6 + 6 + 8 <= 11176

3 + 1.8 + 2.4 <= 3600

2.1 + 1.5 + 0.8 + 3 <= 4200

Змінні:

x1 >= 0

x2 >= 0

x3 >= 0

x4 >= 0

1. Математична модель двоїстої задачі ЛП:

F = 7200 + 5600 +11176 + 3600 + 4200🡺 Min;

Мінімізація витрат по кожному виду технологічної операції.

– міра дефіцинтності волочіння.

- міра дефіцинтності накладання ізоляції.

- міра дефіцинтності скручування елементів у кабель.

- міра дефіцинтності свинцюванння.

- міра дефіцинтності випробування.

Обмеження по кількості витраченого часу на кожен вид технологічної операції:

1.2 + 1 + 6.4 + 3 + 2.1 >= 1.2

1.8 + 0.4 + 5.6 + 1.5 >= 0.8

1.6 + 0.8 + 6 + 1.8 + 0.8 >= 1

2.4 + 0.7 + 8 + 2.4 + 3 >= 1.3

Змінні:

>= 0

>= 0

>= 0

>= 0

>= 0

1. Код програми:

bool SyntexSolver::FoundOptimalDoubleMethod() {

return m\_mainRowIndex == -1;

}

void SyntexSolver::CreateTableForDoubleMethodFromSyntex() {

auto paramSize = m\_conditions.m\_equations.size();

auto coefSize = m\_conditions.m\_equations[0].GetCoeficients().size();

auto prevProblemCoefs = m\_conditions.m\_problem.GetCoeficients();

std::vector<std::pair<int, double>> problemCoefs;

std::vector<std::vector<std::pair<int, double>>> equationsCoefs(coefSize);

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

auto coefs = m\_conditions.m\_equations[i].GetCoeficients();

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

equationsCoefs[j].push\_back({i+1, coefs[j].second});

}

problemCoefs.push\_back({ i + 1, m\_conditions.m\_equations[i].GetB() });

m\_conditionsDoubleMethod.m\_parameters.push\_back(Equation::Create({ {i+1, 1} }, EquationSign::GreaterThan, 0));

}

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

int count1 = 0;

int count0 = 0;

for (int j = 0; j < equationsCoefs[i].size(); ++j) {

if (equationsCoefs[i][j].second == 0) {

count0++;

}

else if (equationsCoefs[i][j].second == 1) {

count1++;

}

}

if (!(((count1 + count0) == equationsCoefs[i].size()) && (prevProblemCoefs[i].second == 0))) {

m\_conditionsDoubleMethod.m\_equations.push\_back(Equation::Create(equationsCoefs[i], EquationSign::GreaterThan, prevProblemCoefs[i].second));

}

}

m\_conditionsDoubleMethod.m\_problem = Equation::Create(problemCoefs, EquationSign::None, 0);

AddAllParams(m\_conditionsDoubleMethod);

ShowConditions(m\_conditionsDoubleMethod, "\nSolve with double method.\n");

ConvertToCanonical(m\_conditionsDoubleMethod);

for (auto& equation : m\_conditionsDoubleMethod.m\_equations) {

for (auto coef : equation.GetCoeficients()) {

if (coef.second == 0) {

continue;

}

if (coef.first <= paramSize - 1) {

continue;

}

if (coef.second == -1) {

for (auto& changeCoef : equation.GetCoeficients()) {

changeCoef.second = -changeCoef.second;

}

equation.SetB(-(equation.GetB()));

}

}

}

basisVectorIndices.resize(m\_conditionsDoubleMethod.m\_parameters.size() - paramSize, 0);

std::iota(basisVectorIndices.begin(), basisVectorIndices.end(), paramSize);

m\_syntexTable.resize(basisVectorIndices.size(), std::vector<double>(m\_conditionsDoubleMethod.m\_parameters.size() + 1, 0));

m\_q.resize(m\_conditionsDoubleMethod.m\_parameters.size() + 1, 0);

m\_lastRow.resize(m\_q.size() - 1, 0);

for (int i = 0; i < m\_syntexTable.size(); ++i) {

const auto equationCoef = m\_conditionsDoubleMethod.m\_equations[i].GetCoeficients();

m\_syntexTable[i][0] = m\_conditionsDoubleMethod.m\_equations[i].GetB();

for (int j = 1; j < m\_syntexTable[0].size(); ++j) {

m\_syntexTable[i][j] = equationCoef[j - 1].second;

}

}

ShowConditions(m\_conditionsDoubleMethod, "\nAfter converting to canonical\n");

}

void SyntexSolver::CreateTableForDoubleMethod() {

m\_conditionsDoubleMethod = m\_conditions;

for (auto& equation : m\_conditionsDoubleMethod.m\_equations) {

for (auto coef : equation.GetCoeficients()) {

if (coef.second == 0) {

continue;

}

if (coef.first <= startParamNumber - 1) {

continue;

}

if (coef.second == -1) {

for (auto& changeCoef : equation.GetCoeficients()) {

changeCoef.second = -changeCoef.second;

}

equation.SetB(-(equation.GetB()));

}

}

}

basisVectorIndices.resize(m\_conditionsDoubleMethod.m\_parameters.size() - startParamNumber, 0);

std::iota(basisVectorIndices.begin(), basisVectorIndices.end(), startParamNumber);

m\_syntexTable.resize(basisVectorIndices.size(), std::vector<double>(m\_conditionsDoubleMethod.m\_parameters.size() + 1, 0));

m\_q.resize(m\_conditionsDoubleMethod.m\_parameters.size() + 1, 0);

m\_lastRow.resize(m\_q.size() - 1, 0);

for (int i = 0; i < m\_syntexTable.size(); ++i) {

const auto equationCoef = m\_conditionsDoubleMethod.m\_equations[i].GetCoeficients();

m\_syntexTable[i][0] = m\_conditionsDoubleMethod.m\_equations[i].GetB();

for (int j = 1; j < m\_syntexTable[0].size(); ++j) {

m\_syntexTable[i][j] = equationCoef[j - 1].second;

}

}

ShowConditions(m\_conditionsDoubleMethod, "\nSolve with double method.\n");

}

void SyntexSolver::SolveWithDoubleMethod(bool fromSyntex) {

if (fromSyntex) {

CreateTableForDoubleMethodFromSyntex();

}

else {

CreateTableForDoubleMethod();

}

CalculateQForDoubleMethod();

ShowSyntexTableWithDoubleMethod();

while (!FoundOptimalDoubleMethod()) {

RecalculateTable();

CalculateQForDoubleMethod();

ShowSyntexTableWithDoubleMethod();

}

}

void SyntexSolver::CalculateQForDoubleMethod() {

for (int j = 0; j < m\_syntexTable[0].size(); ++j) {

m\_q[j] = 0;

}

double min\_row = 0;

int min\_index\_row = -1;

double min\_col = std::numeric\_limits<double>::max();

int min\_index\_col = -1;

const auto& problemCoef = m\_conditionsDoubleMethod.m\_problem.GetCoeficients();

for (int j = 0; j < m\_syntexTable.size(); ++j) {

m\_q[0] += m\_syntexTable[j][0] \* problemCoef[basisVectorIndices[j]].second;

}

for (int i = 1; i < m\_syntexTable[0].size(); ++i) {

for (int j = 0; j < m\_syntexTable.size(); ++j) {

m\_q[i] += (m\_syntexTable[j][i] \* problemCoef[basisVectorIndices[j]].second);

}

m\_q[i] -= problemCoef[i - 1].second;

for (int n = 0; n < m\_syntexTable.size(); ++n) {

if (m\_syntexTable[n][0] < 0 && m\_syntexTable[n][0] < min\_row) {

min\_row = m\_syntexTable[n][0];

min\_index\_row = n;

}

}

}

m\_mainRowIndex = min\_index\_row;

if (m\_mainRowIndex == -1) {

return;

}

for (int j = 1; j < m\_q.size(); ++j) {

if (m\_q[j] < 0) {

m\_mainRowIndex = -1;

return;

}

}

for (int j = 1; j < m\_q.size(); ++j) {

if (m\_syntexTable[m\_mainRowIndex][j] >= 0) {

continue;

}

else {

auto delta = (-m\_q[j]) / m\_syntexTable[m\_mainRowIndex][j];

m\_lastRow[j - 1] = delta;

if (delta < min\_col) {

min\_col = delta;

min\_index\_col = j;

}

}

}

m\_mainColIndex = min\_index\_col;

}

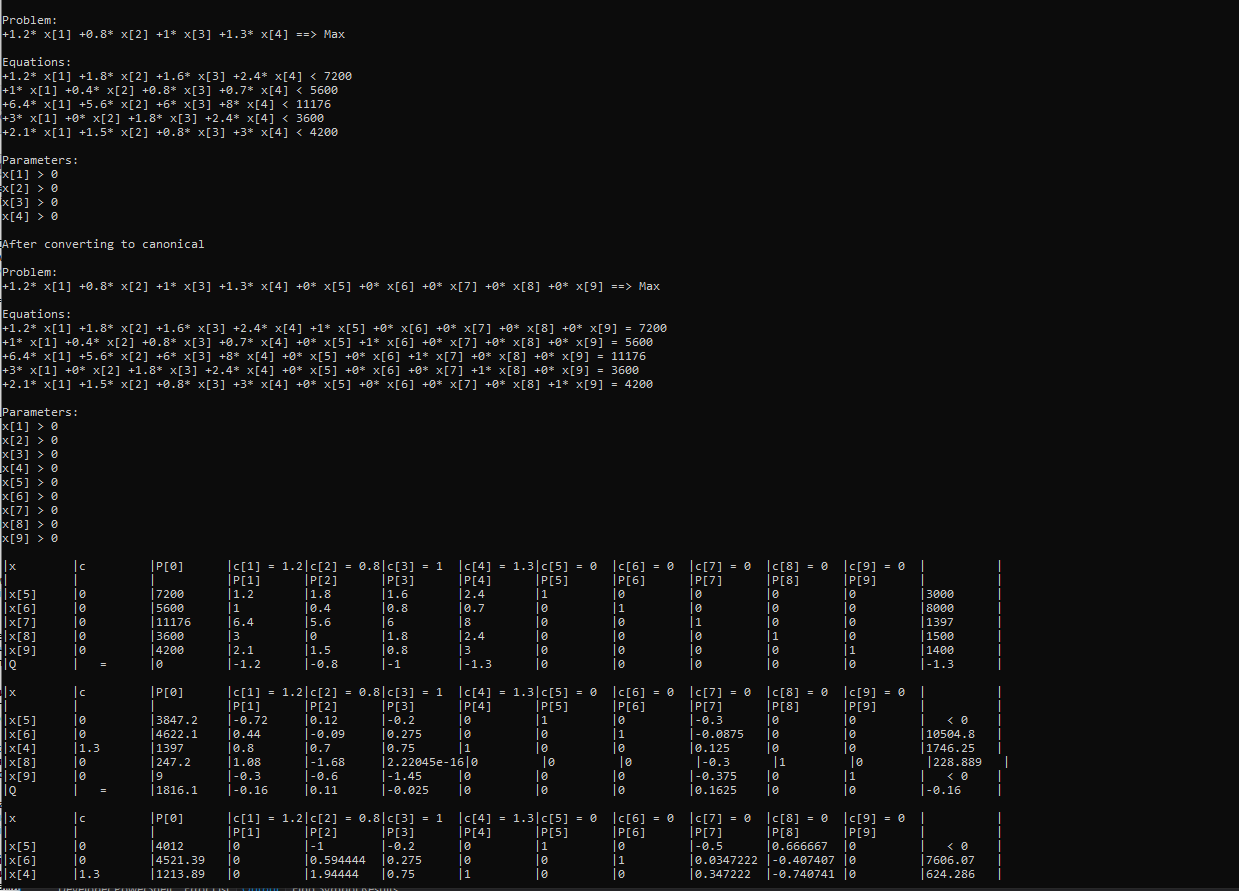


Рис. 2. Результати програми

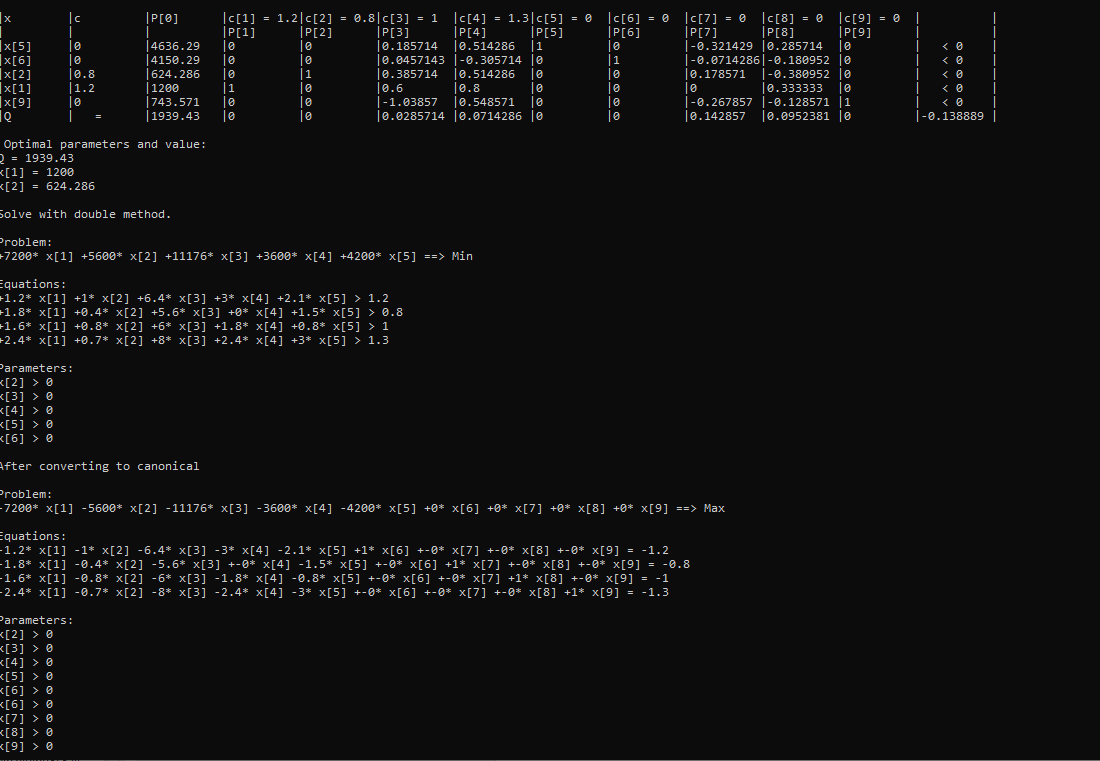


Рис. 3. Результати програми

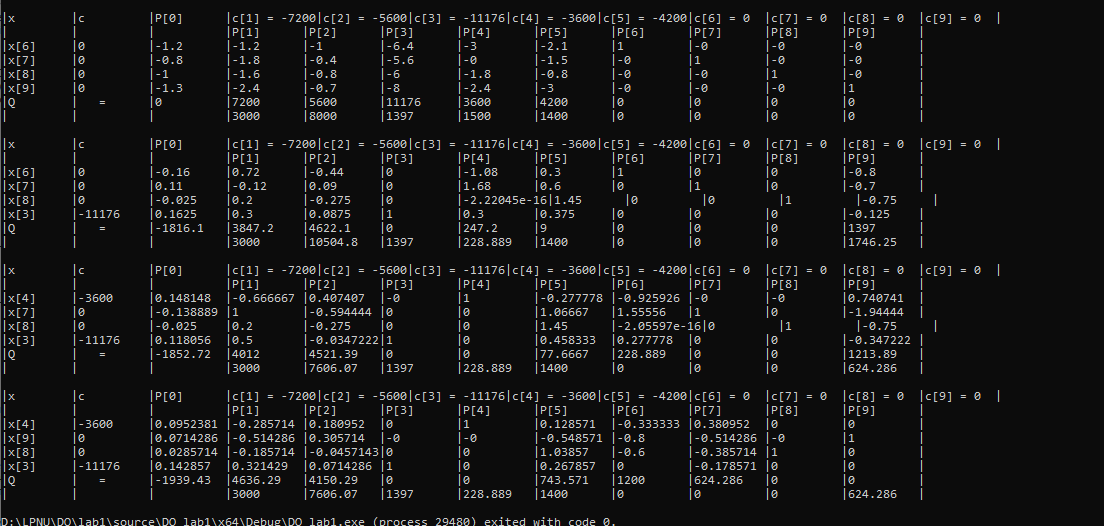


Рис. 4. Результати програми

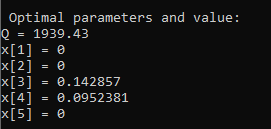


Рис. 5. Результати програми

Це означає що, волочіння, накладання ізоляції та випробування і контроль мають надлишок у фонді загального часу, а скручування елементів у кабель і свинцюванння використовуються у повному обсязі.

**ВИСНОВКИ**

На цій лабораторній роботі я ознайомився з одним із методів вирішення задачі ЛП. Реалізував програму з використанням двоїстого методу, яка зводить дані до канонічного вигляду та шукає оптимальне рішення задачі. Також ознайомився із зв’язком між звичайним і двоїстим симплекс-методом.