**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **КНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 3

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

**На тему:** *“Розв’язування задачі цілочислового програмування за допомогою методу Гоморі. Геометрична інтерпретація розв’язку. ”*

**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:** Розв’язування задачі цілочислового програмування за допомогою методу Гоморі. Геометрична інтерпретація розв’язку.

**Мета роботи:** Ознайомитись на практиці із основними алгоритмами

розв’язування цілочислових задач математичного програмування.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

3. Які є типи задач ЦЛП?

- Повністю цілочисельно.

- Змішана або частково цілочисельна (лише деякі параметри цілі).

6. Назвіть основні кроки першого алгоритму Гоморі.

1. Знайти оптимальний розв’язок будь-яким методом без врахування цілочисельності. Якщо немає розв’язків, то Кінець. Якщо всі змінні цілочисельні, то Кінець.
2. Знаходим значення з найбільшою дробовою частиною і укладаєм нове обмеження.
3. За допомогою двоїстого методу знаходимо новий оптимальний розв’язок.
4. Якщо для отриманого розв’язку виконуються умови цілочиселовості, то кінець – знайдений оптимальний розв’язок задачі. Якщо розв’язок задачі

відсутній – кінець. В іншому випадку переходимо до кроку 2

9. Яка геометрична інтерпретація розв'язків цілочислової задачі на площині?

Цілочисельна сітка – точки у який значення змінних цілі і вони знаходяться в області розв’язків.

**ЗАВДАННЯ**

1. Отримати індивідуальний варіант завдання.

2. Написати програму розв’язування задачі ЦЛП методом Гоморі та знайти розв’язок (максимальне значення функції та значення цілочислових змінних, при якому воно досягається) згідно з варіантом Додатку 1 до лабораторної роботи № 3.

3. Розв’язати задачу з Додатка 1 до лабораторної роботи № 3 графічним

методом (знайти максимальне значення функції та значення змінних,

при якому воно досягається).

4. Оформити звіт про виконану роботу.

5. Продемонструвати викладачеві результати, відповісти на запитання

стосовно виконання роботи.

**ХІД ВИКОНАННЯ**

1. Індивідуальне завдання з додатка №1.

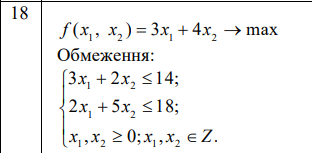


Рис. 1. Задача з додатка №1

1. Код програми:

bool SyntexSolver::SolveWithGomory() {

return m\_GomorySolve;

}

void SyntexSolver::GomorySolve() {

if (!CheckIfCanSolve()) {

std::cout << "\nCan't solve start problem with syntex method!\n";

SolveWithDoubleMethod();

m\_conditions = m\_conditionsDoubleMethod;

}

else {

Solve();

}

GomorySolveAfterInitialResult();

}

bool SyntexSolver::CheckGomoryResults() {

int isIntegerCount = 0;

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

auto basisVectorIndicesIterator =

std::find(basisVectorIndices.begin(), basisVectorIndices.end(), m\_conditions.m\_parameters[i].GetCoeficients()[0].first);

if (basisVectorIndicesIterator != basisVectorIndices.end()) {

int index = std::distance(basisVectorIndices.begin(), basisVectorIndicesIterator);

if ((m\_syntexTable[index][0] - static\_cast<int>(m\_syntexTable[index][0])) > 0.05) {

std::cout << "x[" << m\_conditions.m\_parameters[i].GetCoeficients()[0].first + 1 << "] is not integer\n";

}

else {

m\_syntexTable[index][0] = static\_cast<int>(m\_syntexTable[index][0]);

++isIntegerCount;

}

m\_resultsValueWithBasisVectorIndex[i] = std::pair<double, int>(m\_syntexTable[index][0], index);

}

}

if (isIntegerCount == startParamNumber) {

return false;

}

return true;

}

void SyntexSolver::GomorySolveAfterInitialResult()

{

while (CheckGomoryResults()) {

std::cout << "\nResult not integer. Next iteration: \n";

double maxValue = 0;

int maxIndex = -1;

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

double fractionPart;

if (m\_resultsValueWithBasisVectorIndex[i].first >= 0) {

fractionPart = m\_resultsValueWithBasisVectorIndex[i].first - static\_cast<int>(m\_resultsValueWithBasisVectorIndex[i].first);

}

else {

fractionPart = m\_resultsValueWithBasisVectorIndex[i].first - static\_cast<int>(m\_resultsValueWithBasisVectorIndex[i].first - 1);

}

if (fractionPart > maxValue) {

maxValue = fractionPart;

maxIndex = m\_resultsValueWithBasisVectorIndex[i].second;

}

}

if (maxValue == 0 || maxIndex == -1) {

std::cout << "Error! Fraction part = 0.";

return;

}

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

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

if (col <= startParamNumber) {

double fractionPart;

if (m\_syntexTable[maxIndex][col] >= 0) {

fractionPart = m\_syntexTable[maxIndex][col] - static\_cast<int>(m\_syntexTable[maxIndex][col]);

}

else {

fractionPart = m\_syntexTable[maxIndex][col] - static\_cast<int>(m\_syntexTable[maxIndex][col] - 1);

}

if (fractionPart <= maxValue) {

coef.push\_back(std::pair<int, double>(col, -fractionPart \* 1 / maxValue));

}

else {

double a = (maxValue / (1 - maxValue)) \* (1 - fractionPart);

coef.push\_back(std::pair<int, double>(col, -a \* 1 / maxValue));

}

}

else {

if (m\_syntexTable[maxIndex][col] >= 0) {

coef.push\_back(std::pair<int, double>(col, -m\_syntexTable[maxIndex][col] \* 1 / maxValue));

}

else {

double a = (maxValue / (1 - maxValue)) \* abs(m\_syntexTable[maxIndex][col]);

coef.push\_back(std::pair<int, double>(col, -a \* 1 / maxValue));

}

}

}

coef.push\_back(std::pair<int, double>(m\_syntexTable[0].size(), 1 \* 1 / maxValue));

auto newEquation = Equation::Create(coef, EquationSign::Equal, -maxValue \* 1 / maxValue);

m\_conditions.m\_equations.push\_back(newEquation);

m\_conditions.m\_parameters.push\_back(Equation::Create(std::vector<std::pair<int, double>>(1, std::pair<int, double>(m\_syntexTable[0].size() - 1, 1)), EquationSign::GreaterThan, 0));

AddAllParams(m\_conditions);

ShowConditions(m\_conditions, "\nAfter adding new equation\n");

m\_q.push\_back(0);

m\_lastColumn.push\_back(0);

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

basisVectorIndices.push\_back(m\_syntexTable[0].size() - 1);

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

m\_syntexTable[i].push\_back(0);

}

std::vector<double> addedRow;

addedRow.push\_back(newEquation.GetB());

for (int i = 0; i < coef.size(); ++i) {

addedRow.push\_back(coef[i].second);

}

m\_syntexTable.push\_back(addedRow);

m\_conditionsDoubleMethod = m\_conditions;

CalculateQForDoubleMethod();

ShowSyntexTableWithDoubleMethod();

while (!FoundOptimalDoubleMethod()) {

RecalculateTable();

CalculateQForDoubleMethod();

ShowSyntexTableWithDoubleMethod();

}

std::cout << "\n Optimal parameters and value:\n";

std::cout << "Q = " << -m\_q[0] << "\n";

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

auto basisVectorIndicesIterator =

std::find(basisVectorIndices.begin(), basisVectorIndices.end(), m\_conditionsDoubleMethod.m\_parameters[i].GetCoeficients()[0].first);

if (basisVectorIndicesIterator != basisVectorIndices.end()) {

int index = std::distance(basisVectorIndices.begin(), basisVectorIndicesIterator);

if (m\_conditionsDoubleMethod.m\_problem.GetCoeficients()[\*basisVectorIndicesIterator].second != 0) {

std::cout << "x[" << m\_conditionsDoubleMethod.m\_parameters[i].GetCoeficients()[0].first + 1 << "] = " << m\_syntexTable[index][0] << "\n";

}

else {

std::cout << "x[" << m\_conditionsDoubleMethod.m\_parameters[i].GetCoeficients()[0].first + 1 << "] = 0" << "\n";

}

}

else {

std::cout << "x[" << m\_conditionsDoubleMethod.m\_parameters[i].GetCoeficients()[0].first + 1 << "] = 0" << "\n";

}

}

}

std::cout << "\n\nResults: \nQ = " << (int)m\_q[0] << ";\n";

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

std::cout << "x[" << i+1 << "] = " << m\_resultsValueWithBasisVectorIndex[i].first << ";\n";

}

std::cout << "\nEnd!!";

}

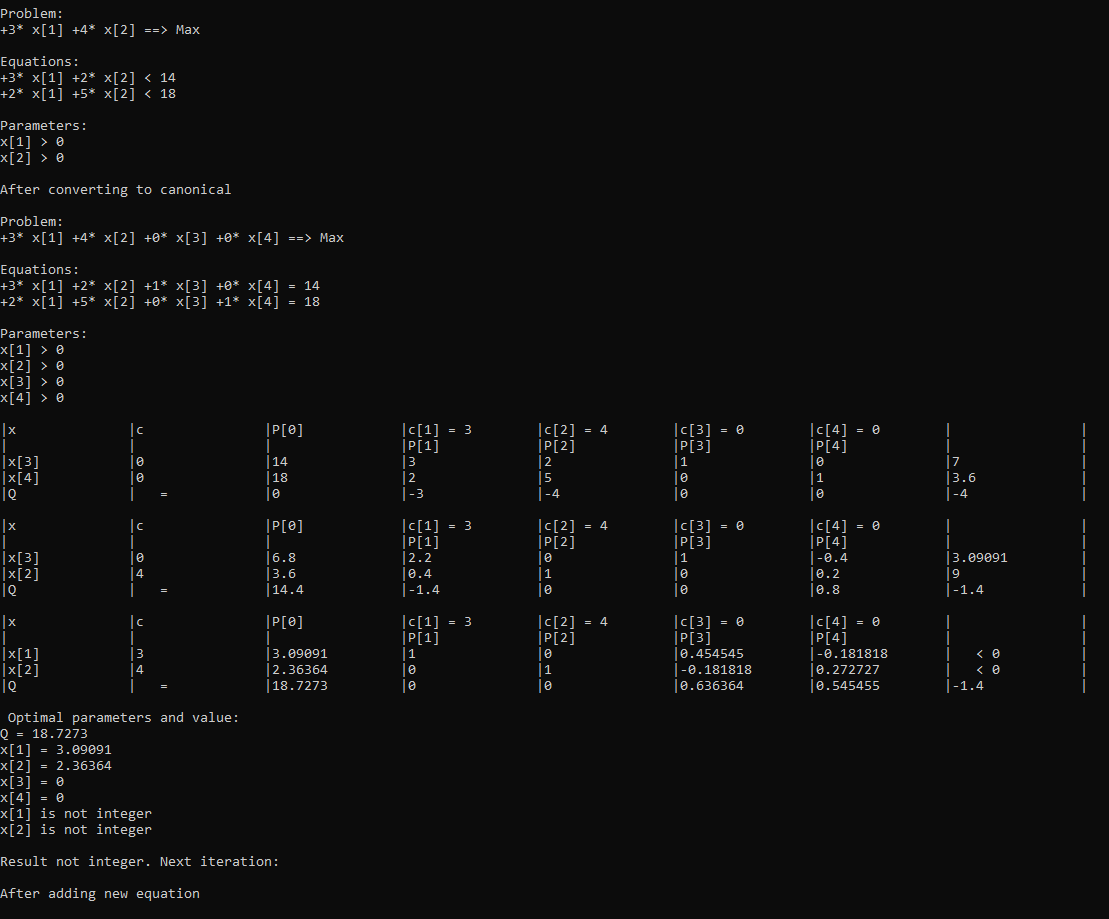


Рис. 2. Результати програми

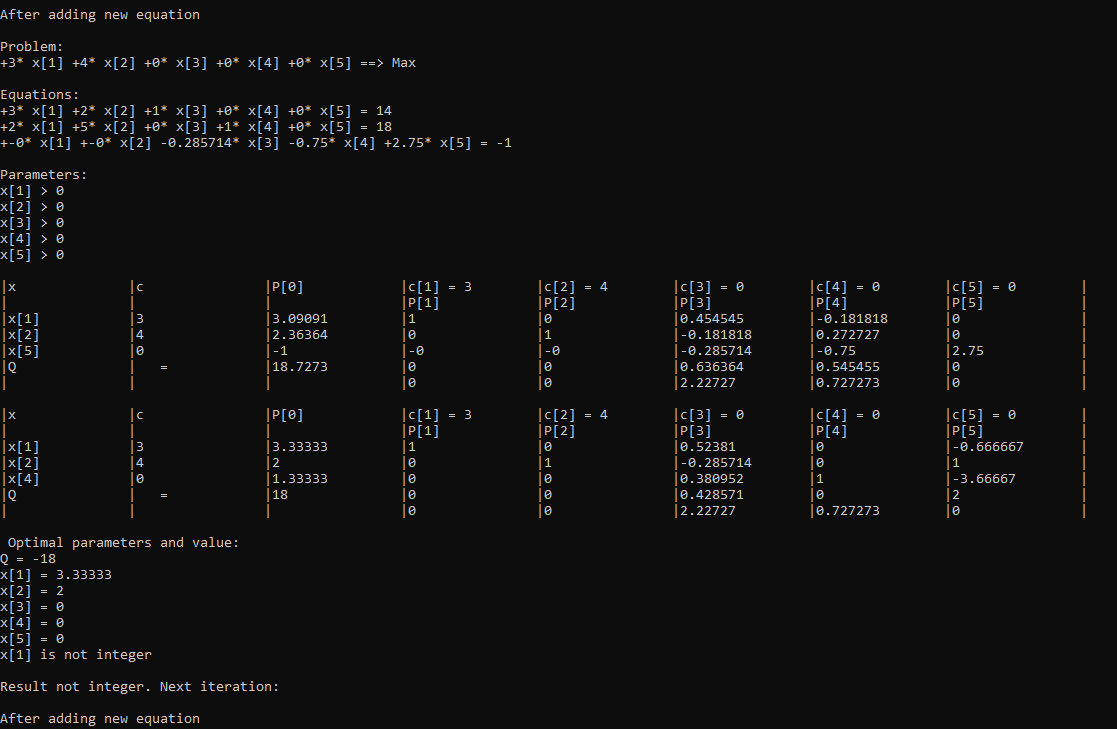


Рис. 3. Результати програми

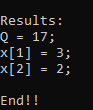
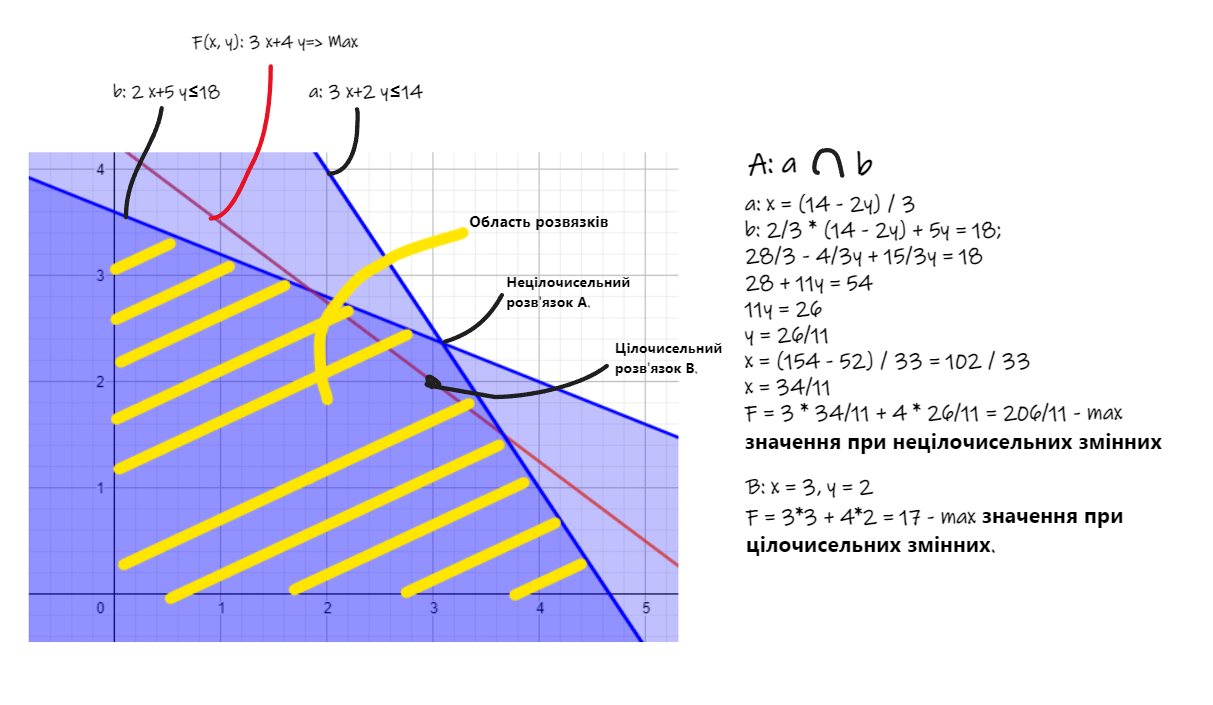


Рис. 4. Результати програми

1. Розв’язання графічним методом:



**ВИСНОВКИ**

На цій лабораторній роботі я ознайомився з одним із методів вирішення задачі цілочисельного ЛП. Реалізував програму з використанням методу Гоморі для частково цілочисельнихх задач, яка зводить дані до канонічного вигляду та шукає оптимальне цілочисельне рішення задачі.