# Отчёт к лабораторной работе №3

Выполнил: Фроловский А. ИУ7-82

Вариант 6

Содержательная и математическая постановки транспортной задачи

1. Содержательная постановка

Имеется производителей (поставщиков) некоторой однородной продукции. Мощность производителя обозначается . Имеется потребителей этой продукции, мощность потребителя обозначим .

Стоимость перевозки единицы продукции от поставщика к потребителю составляет единиц.

Необходимо составить план перевозок продукции таким образом, чтобы вся продукция была перевезена с учётом ограничений на мощности (в этой постановке предполагается, что – сбалансированная задача) и общая стоимость всех перевозок была бы минимальной.

1. Математическая постановка

где

– объём продукции, перевозимой от поставщика к потребителю,

*-* стоимость перевозки единицы продукции от поставщика к потребителю,

- мощность производителя,

- мощность потребителя.

Содержательная и математическая постановки задачи о назначениях.

1. Содержательная постановка

В распоряжении работодателя имеется n работ (заданий) и n исполнителей. Затраты на выполнение i-м работником j-й работы составляют условных единиц. Каждый работник может выполнить только 1 задание, при этом все задания должны быть выполнены. Необходимо составить план распределения работ между исполнителями, чтобы общие затраты были минимальны.

1. Математическая постановка

где

– матрица стоимостей,

- матрица назначений, в которой

Краткое описание метода потенциалов

Блок схема метода потенциалов представлена на рис.1.

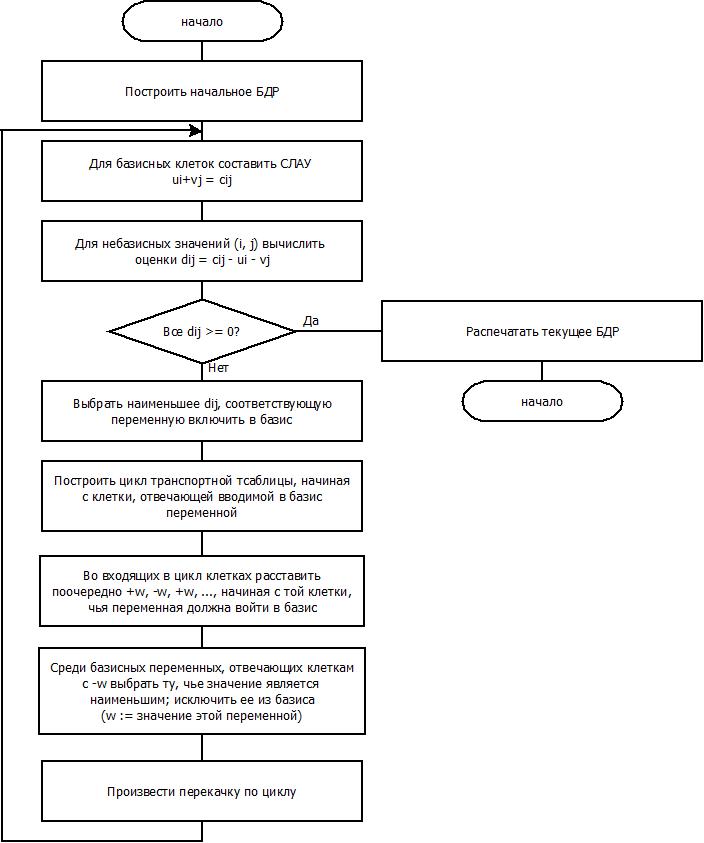


Рис. 1. Блок-схема метода потенциалов

Текст программы

package lab1.pkg1;

public class PotentialsMethod

{

public PotentialsMethod(InitialBASBuilder BASbuilder, TransportTable table)

{

this.BASbuilder = BASbuilder;

this.table = table;

}

public String getOutput()

{

return output;

}

public Matrix execute() throws Exception

{

initializeVariables();

buildBasisAccessibleSolution();

return executeMainActions();

}

private void initializeVariables()

{

this.output = "";

}

private void buildBasisAccessibleSolution()

{

BASbuilder.execute(table);

output += "\nНачальное БДР: \n" + table.toString() + "\n";

}

private Matrix executeMainActions() throws Exception

{

calculateEstimates();

while (isExistedNegativeEstimates())

{

numberOfIteration++;

improveSolution();

calculateEstimates();

}

return table.getVolumes();

}

private void improveSolution() throws Exception

{

output += "\n" + numberOfIteration + " итерация\n";

SolutionImprover improver = new SolutionImprover(estimates, table);

table = improver.execute();

output += improver.toString();

}

private void calculateEstimates() throws Exception

{

estimates = new EstimatesHolder(table);

this.output += "\nОценки: \n" + estimates.toString() + "\n";

}

private boolean isExistedNegativeEstimates()

{

return estimates.isExistedNegativeEstimates();

}

public double getCost()

{

return table.getSummaryCost();

}

private EstimatesHolder estimates;

private TransportTable table;

private InitialBASBuilder BASbuilder;

private String output;

private int numberOfIteration = 0;

}

package lab1.pkg1;

import java.util.logging.Level;

import java.util.logging.Logger;

public class NorthWesternCornerMethod extends InitialBASBuilder

{

@Override

public void execute(TransportTable table)

{

try

{

this.table = table;

initializeVariable();

buildBAS();

}

catch (Exception ex)

{

Logger.getLogger(NorthWesternCornerMethod.class.getName()).log(Level.SEVERE, null, ex);

}

}

private void initializeVariable() throws Exception

{

this.curIndexOfRow = 0;

this.curIndexOfColumn = 0;

this.sourcePowerBalances = table.getSourcePowers();

this.numberOfSources = table.getNumberOfSources();

this.flowPowerBalances = table.getFlowPowers();

this.numberOfFlows = table.getNumberOfFlows();

excludeAllTablElementsFromBasis();

}

private void excludeAllTablElementsFromBasis() throws Exception

{

for (int i = 0; i < this.numberOfSources; i++)

for (int j = 0; j < this.numberOfFlows; j++)

table.excludeFromBasis(i, j);

}

private void buildBAS() throws Exception

{

while (isNotSatisfiedStopConditions())

addElementToBasis();

}

private boolean isNotSatisfiedStopConditions()

{

return (curIndexOfRow < numberOfSources) && (curIndexOfColumn < numberOfFlows);

}

private void addElementToBasis() throws Exception

{

double transferredValue = getTransferredValue();

changeCurrentPowerBalances(transferredValue);

includeCurrentElementInBasis(transferredValue);

changeIndexes();

}

private double getTransferredValue()

{

currentSourcePower = this.sourcePowerBalances[this.curIndexOfRow];

currentFlowPower = this.flowPowerBalances[this.curIndexOfColumn];

return Math.min(currentSourcePower, currentFlowPower);

}

private void changeCurrentPowerBalances(double min)

{

this.sourcePowerBalances[this.curIndexOfRow] = currentSourcePower - min;

this.flowPowerBalances[this.curIndexOfColumn] = currentFlowPower - min;

}

private void includeCurrentElementInBasis(double min) throws Exception

{

table.includeInBasis(this.curIndexOfRow, this.curIndexOfColumn);

table.setVolume(this.curIndexOfRow, this.curIndexOfColumn, min);

}

private void changeIndexes()

{

if (currentSourcePower > currentFlowPower)

this.curIndexOfColumn++;

else

this.curIndexOfRow++;

}

private int numberOfSources;

private int numberOfFlows;

private double[] sourcePowerBalances;

private double[] flowPowerBalances;

private double currentSourcePower;

private double currentFlowPower;

private int curIndexOfRow;

private int curIndexOfColumn;

private TransportTable table;

}

package lab1.pkg1;

public class SolutionImprover

{

public SolutionImprover(EstimatesHolder estimates, TransportTable table)

{

this.estimates = estimates;

this.table = table;

}

public TransportTable execute() throws Exception

{

initializeVariables();

return executeMainActions();

}

private void initializeVariables()

{

this.output = "";

}

private TransportTable executeMainActions() throws Exception

{

includeVariableInBasis();

buildCycle();

exceptVariableFromBasis();

swapping();

output += "\nПолученная транспортная таблица:\n" + table.toString();

return table;

}

private void includeVariableInBasis() throws Exception

{

findMinEstimate();

includeElementWithMinEstimateInBasis();

CycleElement element = new CycleElement(minEstimate.getIndexOfRow(), minEstimate.getIndexOfColumn());

output += "\nВ базис включена переменная " + element.toString() + "\n";

}

private void findMinEstimate()

{

minEstimate = estimates.getMinEstimate();

}

private void includeElementWithMinEstimateInBasis() throws Exception

{

table.includeInBasis(minEstimate.getIndexOfRow(), minEstimate.getIndexOfColumn());

}

private void buildCycle() throws Exception

{

CycleBuilder builder = new CycleBuilder(table);

int startElementRowIndex = minEstimate.getIndexOfRow();

int startElementColumnIndex = minEstimate.getIndexOfColumn();

cycle = builder.execute(new CycleElement(startElementRowIndex, startElementColumnIndex));

output += "\nПостроенный цикл: " + cycle.toString() + "\n";

}

private void exceptVariableFromBasis() throws Exception

{

findVariableForException();

exceptFindedVariable();

output += "\nИз базиса исключена переменная " + exceptedVariable.toString() + "\n";

}

private void findVariableForException() throws Exception

{

CycleElement minElement = cycle.getElement(1);

for (int i = 3; i < cycle.getNumberOfElements(); i += 2)

{

CycleElement curElement = cycle.getElement(i);

if (isCurElementLessMinElement(curElement, minElement))

minElement = curElement;

}

exceptedVariable = minElement;

}

private boolean isCurElementLessMinElement(CycleElement curElement, CycleElement minElement) throws Exception

{

double curValue = table.getVolume(curElement.getIndexOfRow(), curElement.getIndexOfColumn());

double minValue = table.getVolume(minElement.getIndexOfRow(), minElement.getIndexOfColumn());

return curValue < minValue;

}

private void exceptFindedVariable() throws Exception

{

table.excludeFromBasis(exceptedVariable.getIndexOfRow(), exceptedVariable.getIndexOfColumn());

}

private void swapping() throws Exception

{

double omega = getOmega();

output += "\nТранспортная таблица до перекачки:\n" + table.toString();

output += "\nомега = " + omega;

int sizeOfCycle = cycle.getNumberOfElements();

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

changeOmegaFromCycleElementVolume(i, omega);

}

private double getOmega() throws Exception

{

return table.getVolume(exceptedVariable.getIndexOfRow(), exceptedVariable.getIndexOfColumn());

}

private void changeOmegaFromCycleElementVolume(int indexOfCycleElement, double omega) throws Exception

{

CycleElement element = cycle.getElement(indexOfCycleElement);

int indexOfRow = element.getIndexOfRow();

int indexOfColumn = element.getIndexOfColumn();

if (isOddIndex(indexOfCycleElement))

subValueFromElementVolume(indexOfRow, indexOfColumn, omega);

else

addValueToElementVolume(indexOfRow, indexOfColumn, omega);

}

private boolean isOddIndex(int index)

{

return index % 2 == 1;

}

private void subValueFromElementVolume(int indexOfRow, int indexOfColumn, double subtractedValue) throws Exception

{

double oldValue = table.getVolume(indexOfRow, indexOfColumn);

double newValue = oldValue - subtractedValue;

table.setVolume(indexOfRow, indexOfColumn, newValue);

}

private void addValueToElementVolume(int indexOfRow, int indexOfColumn, double subtractedValue) throws Exception

{

double oldValue = table.getVolume(indexOfRow, indexOfColumn);

double newValue = oldValue + subtractedValue;

table.setVolume(indexOfRow, indexOfColumn, newValue);

}

@Override

public String toString()

{

return output;

}

private Cycle cycle;

private Estimate minEstimate;

private EstimatesHolder estimates;

private TransportTable table;

private CycleElement exceptedVariable;

private String output;

}

package lab1.pkg1;

import java.util.ArrayList;

import java.util.List;

public class CycleBuilder

{

public CycleBuilder(TransportTable table)

{

this.table = table;

}

public Cycle execute(CycleElement startElement) throws Exception

{

cycle = new Cycle();

cycle.add(startElement);

Cycle newCycle = executeStep(startElement, cycle, TypeStep.HorizontalStep);

if (newCycle == null)

newCycle = executeStep(startElement, cycle, TypeStep.VerticalStep);

return newCycle;

}

private Cycle executeStep(CycleElement currentElement, Cycle cycle, TypeStep typeStep) throws Exception

{

List<CycleElement> nextElementsList = getNextElementsList(cycle, currentElement,

typeStep.getPerpendicularStepDiraction());

if (executingStopConditions(cycle, nextElementsList))

return cycle;

return addElements(cycle, nextElementsList, typeStep);

}

private Cycle addElements(Cycle currentCycle, List<CycleElement> nextElementsList, TypeStep typeStep) throws Exception

{

Cycle newCycle = currentCycle.clone();

for (CycleElement element: nextElementsList)

{

Cycle workList = currentCycle.clone();

workList.add(element);

Cycle resultList = executeStep(element, workList, typeStep.getPerpendicularStepDiraction());

if (resultList != null)

return resultList;

}

return null;

}

private boolean executingStopConditions(Cycle cycle, List<CycleElement> nextElementsList) throws Exception

{

CycleElement startElement = cycle.getStartElement();

if (cycle.getNumberOfElements() > 1 && nextElementsList.contains(startElement))

return true;

return false;

}

private List<CycleElement> getNextElementsList(Cycle currentCycle, CycleElement currentElement, TypeStep typeStep) throws Exception

{

ArrayList<CycleElement> elementsList = new ArrayList<CycleElement>();

switch (typeStep)

{

case HorizontalStep:

fillNextHorizontalElementsList(elementsList, currentElement, currentCycle);

break;

case VerticalStep:

fillNextVerticalElementsList(elementsList, currentElement, currentCycle);

break;

}

return elementsList;

}

private void fillNextVerticalElementsList(List<CycleElement> elementsList, CycleElement startElement, Cycle currentCycle) throws Exception

{

for (int i = 0; i < table.getNumberOfSources(); i++)

{

CycleElement element = new CycleElement(i, startElement.getIndexOfColumn());

if (isSuitableElement(currentCycle, element))

elementsList.add(element);

}

}

private boolean isSuitableElement(Cycle currentCycle, CycleElement element) throws Exception

{

CycleElement startElement = currentCycle.getStartElement();

return (element.isEquals(startElement) && currentCycle.getNumberOfElements() > 1 ) ||

( !currentCycle.containElement(element) &&

table.isIncludedInBasis(element.getIndexOfRow(), element.getIndexOfColumn()));

}

private void fillNextHorizontalElementsList(List<CycleElement> elementsList, CycleElement startElement, Cycle currentCycle) throws Exception

{

for (int j = 0; j < table.getNumberOfFlows(); j++)

{

CycleElement element = new CycleElement(startElement.getIndexOfRow(), j);

if (isSuitableElement(currentCycle, element))

elementsList.add(element);

}

}

private enum TypeStep

{

HorizontalStep, VerticalStep;

public TypeStep getPerpendicularStepDiraction()

{

if (this.equals(HorizontalStep))

return VerticalStep;

return HorizontalStep;

}

};

private Cycle cycle;

private TransportTable table;

}

package lab1.pkg1;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Iterator;

import java.util.List;

public class EstimatesHolder

{

public EstimatesHolder(TransportTable table) throws Exception

{

estimates = new ArrayList<Estimate>();

this.table = table;

calculateEstimates();

}

private void calculateEstimates() throws Exception

{

initializeSLAE();

buildSLAE();

solveSLAE();

buildEstimates();

}

private void initializeSLAE()

{

numberOfSources = table.getNumberOfSources();

numberOfFlows = table.getNumberOfFlows();

numberOfBasisVariables = numberOfSources + numberOfFlows - 1;

a = new double[numberOfBasisVariables][numberOfBasisVariables];

b = new double[numberOfBasisVariables];

}

private void buildSLAE() throws Exception

{

int numberOfRows = table.getNumberOfSources();

int numberOfColumns = table.getNumberOfFlows();

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

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

addEquationIfElementIncludingInBasis(i, j);

}

private void addEquationIfElementIncludingInBasis(int indexOfRow, int indexOfColumn) throws Exception

{

if (isBasisElement(indexOfRow, indexOfColumn))

addEquation(indexOfRow, indexOfColumn);

}

private boolean isBasisElement(int indexOfRow, int indexOfColumn) throws Exception

{

return table.isIncludedInBasis(indexOfRow, indexOfColumn);

}

private void addEquation(int indexOfRow, int indexOfColumn) throws Exception

{

zeroingExtendedMatrixRow();

addUi(indexOfRow);

addVj(indexOfColumn);

addRightPartOfEquation(indexOfRow, indexOfColumn);

indexOfAddedEquations++;

}

private void zeroingExtendedMatrixRow()

{

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

a[indexOfAddedEquations][j] = 0;

b[indexOfAddedEquations] = 0;

}

private void addUi(int indexOfRow)

{

a[indexOfAddedEquations][indexOfRow] = 1.0;

}

private void addVj(int indexOfColumn)

{

if (isUnzeroingElement(indexOfColumn))

a[indexOfAddedEquations][getIndexOfVj(indexOfColumn)] = 1.0;

}

private boolean isUnzeroingElement(int indexOfColumn)

{

return getIndexOfVj(indexOfColumn) < (numberOfBasisVariables);

}

private int getIndexOfVj(int indexOfColumn)

{

return numberOfSources + indexOfColumn;

}

private void addRightPartOfEquation(int indexOfRow, int indexOfColumn) throws Exception

{

b[indexOfAddedEquations] = table.getCost(indexOfRow, indexOfColumn);

}

private void solveSLAE()

{

GaussMethod solver = new GaussMethod();

x = solver.execute(a, b);

}

private void buildEstimates() throws Exception

{

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

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

addEstimateIfElementNotIncludingInBasis(i, j);

}

private void addEstimateIfElementNotIncludingInBasis(int indexOfRow, int indexOfColumn) throws Exception

{

if (!isBasisElement(indexOfRow, indexOfColumn))

addEstimate(indexOfRow, indexOfColumn);

}

private void addEstimate(int indexOfRow, int indexOfColumn) throws Exception

{

double cij = table.getCost(indexOfRow, indexOfColumn);

double ui = getValueOfUi(indexOfRow);

double vj = getValueOfVj(indexOfColumn);

double dij = cij - ui - vj;

estimates.add(new Estimate(indexOfRow, indexOfColumn, dij));

}

private double getValueOfUi(int indexOfRow)

{

return x[indexOfRow];

}

private double getValueOfVj(int indexOfColumn)

{

if (isUnzeroingElement(indexOfColumn))

return x[getIndexOfVj(indexOfColumn)];

return 0.0;

}

public boolean isExistedNegativeEstimates()

{

Iterator iter = getIterator();

while (iter.hasNext())

{

Estimate estimate = (Estimate) iter.next();

if (estimate.isNegative())

return true;

}

return false;

}

public Iterator getIterator()

{

return estimates.iterator();

}

public Estimate getMinEstimate()

{

return Collections.min(estimates);

}

public String toString()

{

String output = "";

Iterator iter = estimates.iterator();

while (iter.hasNext())

{

Estimate estimate = (Estimate) iter.next();

output += estimate.toString() + "\n";

}

return output;

}

private List<Estimate> estimates;

private TransportTable table;

private int indexOfAddedEquations = 0;

private int numberOfBasisVariables;

private int numberOfSources;

private int numberOfFlows;

private double[][] a;

private double[] b;

private double[] x;

}

package lab1.pkg1;

public abstract class InitialBASBuilder

{

abstract void execute(TransportTable table);

}

Результаты расчётов для задач из индивидуальных вариантов

Вариант 6

Решение транспортной задачи

Исходная транспотрная таблица:

{D[0]= 180} {D[1]= 110} {D[2]= 60} {D[3]= 40}

{S[ 0] = 170} [ | 9 ] [ | 7 ] [ | 5 ] [ | 3 ]

{S[ 1] = 125} [ | 1 ] [ | 2 ] [ | 4 ] [ | 6 ]

{S[ 2] = 95} [ | 8 ] [ | 10 ] [ | 12 ] [ | 1 ]

Начальное БДР:

{D[0]= 180} {D[1]= 110} {D[2]= 60} {D[3]= 40}

{S[ 0] = 170} [ 170 | 9 ] [ | 7 ] [ | 5 ] [ | 3 ]

{S[ 1] = 125} [ 10 | 1 ] [ 110 | 2 ] [ 5 | 4 ] [ | 6 ]

{S[ 2] = 95} [ | 8 ] [ | 10 ] [ 55 | 12 ] [ 40 | 1 ]

Оценки:

d(0,1) = -3.0

d(0,2) = -7.0

d(0,3) = 2.0

d(1,3) = 13.0

d(2,0) = -1.0

d(2,1) = 0.0

1 итерация

В базис включена переменная (0,2)

Построенный цикл: (0,2)(1,2)(1,0)(0,0)

Из базиса исключена переменная (1,2)

Транспортная таблица до перекачки:

{D[0]= 180} {D[1]= 110} {D[2]= 60} {D[3]= 40}

{S[ 0] = 170}[ 170 | 9 ] [ | 7 ] [ 0 | 5 ] [ | 3 ]

{S[ 1] = 125} [ 10 | 1 ] [ 110 | 2 ] [ | 4 ] [ | 6 ]

{S[ 2] = 95} [ | 8 ] [ | 10 ] [ 55 | 12 ] [ 40 | 1 ]

омега = 5.0

Полученная транспортная таблица:

{D[0]= 180} {D[1]= 110} {D[2]= 60} {D[3]= 40}

{S[ 0] = 170} [ 165 | 9 ] [ | 7 ] [ 5 | 5 ] [ | 3 ]

{S[ 1] = 125} [ 15 | 1 ] [ 110 | 2 ] [ | 4 ] [ | 6 ]

{S[ 2] = 95} [ | 8 ] [ | 10 ] [ 55 | 12 ] [ 40 | 1 ]

Оценки:

d(0,1) = -3.0

d(0,3) = 9.0

d(1,2) = 7.0

d(1,3) = 20.0

d(2,0) = -8.0

d(2,1) = -7.0

2 итерация

В базис включена переменная (2,0)

Построенный цикл: (2,0)(0,0)(0,2)(2,2)

Из базиса исключена переменная (2,2)

Транспортная таблица до перекачки:

{D[0]= 180} {D[1]= 110} {D[2]= 60} {D[3]= 40}

{S[ 0] = 170} [ 165 | 9 ] [ | 7 ] [ 5 | 5 ] [ | 3 ]

{S[ 1] = 125} [ 15 | 1 ] [ 110 | 2 ] [ | 4 ] [ | 6 ]

{S[ 2] = 95} [ 0 | 8 ] [ | 10 ] [ | 12 ] [ 40 | 1 ]

омега = 55.0

Полученная транспортная таблица:

{D[0]= 180} {D[1]= 110} {D[2]= 60} {D[3]= 40}

{S[ 0] = 170} [ 110 | 9 ] [ | 7 ] [ 60 | 5 ] [ | 3 ]

{S[ 1] = 125} [ 15 | 1 ] [ 110 | 2 ] [ | 4 ] [ | 6 ]

{S[ 2] = 95} [ 55 | 8 ] [ | 10 ] [ | 12 ] [ 40 | 1 ]

Оценки:

d(0,1) = -3.0

d(0,3) = 1.0

d(1,2) = 7.0

d(1,3) = 12.0

d(2,1) = 1.0

d(2,2) = 8.0

3 итерация

В базис включена переменная (0,1)

Построенный цикл: (0,1)(1,1)(1,0)(0,0)

Из базиса исключена переменная (1,1)

Транспортная таблица до перекачки:

{D[0]= 180} {D[1]= 110} {D[2]= 60} {D[3]= 40}

{S[ 0] = 170} [ 110 | 9 ] [ 0 | 7 ] [ 60 | 5 ] [ | 3 ]

{S[ 1] = 125} [ 15 | 1 ] [ | 2 ] [ | 4 ] [ | 6 ]

{S[ 2] = 95} [ 55 | 8 ] [ | 10 ] [ | 12 ] [ 40 | 1 ]

омега = 110.0

Полученная транспортная таблица:

{D[0]= 180} {D[1]= 110} {D[2]= 60} {D[3]= 40}

{S[ 0] = 170} [ 0 | 9 ] [ 110 | 7 ] [ 60 | 5 ] [ | 3 ]

{S[ 1] = 125} [ 125 | 1 ] [ | 2 ] [ | 4 ] [ | 6 ]

{S[ 2] = 95} [ 55 | 8 ] [ | 10 ] [ | 12 ] [ 40 | 1 ]

Оценки:

d(0,3) = 1.0

d(1,1) = 3.0

d(1,2) = 7.0

d(1,3) = 12.0

d(2,1) = 4.0

d(2,2) = 8.0

Ответ:

0,0 110,0 60,0 0,0

125,0 0,0 0,0 0,0

55,0 0,0 0,0 40,0

Стоимость перевозок: 1675.0

Решение задачи о назначениях

Исходная транспотрная таблица:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ | 10 ] [ | 8 ] [ | 6 ] [ | 4 ] [ | 9 ]

{S[ 1] = 1} [ | 11 ] [ | 9 ] [ | 10 ] [ | 5 ] [ | 6 ]

{S[ 2] = 1} [ | 5 ] [ | 10 ] [ | 8 ] [ | 6 ] [ | 4 ]

{S[ 3] = 1} [ | 3 ] [ | 11 ] [ | 9 ] [ | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ | 10 ] [ | 11 ] [ | 8 ] [ | 7 ]

Начальное БДР:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ 1 | 10 ] [ | 8 ] [ | 6 ] [ | 4 ] [ | 9 ]

{S[ 1] = 1} [ 0 | 11 ] [ 1 | 9 ] [ | 10 ] [ | 5 ] [ | 6 ]

{S[ 2] = 1} [ | 5 ] [ 0 | 10 ] [ 1 | 8 ] [ | 6 ] [ | 4 ]

{S[ 3] = 1} [ | 3 ] [ | 11 ] [ 0 | 9 ] [ 1 | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ | 10 ] [ | 11 ] [ 0 | 8 ] [ 1 | 7 ]

Оценки:

d(0,1) = 0.0

d(0,2) = 0.0

d(0,3) = 1.0

d(0,4) = 7.0

d(1,2) = 3.0

d(1,3) = 1.0

d(1,4) = 3.0

d(2,0) = -7.0

d(2,3) = 1.0

d(2,4) = 0.0

d(3,0) = -10.0

d(3,1) = 0.0

d(3,4) = 1.0

d(4,0) = -7.0

d(4,1) = -3.0

d(4,2) = 0.0

1 итерация

В базис включена переменная (3,0)

Построенный цикл: (3,0)(1,0)(1,1)(2,1)(2,2)(3,2)

Из базиса исключена переменная (1,0)

Транспортная таблица до перекачки:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ 1 | 10 ] [ | 8 ] [ | 6 ] [ | 4 ] [ | 9 ]

{S[ 1] = 1} [ | 11 ] [ 1 | 9 ] [ | 10 ] [ | 5 ] [ | 6 ]

{S[ 2] = 1} [ | 5 ] [ 0 | 10 ] [ 1 | 8 ] [ | 6 ] [ | 4 ]

{S[ 3] = 1} [ 0 | 3 ] [ | 11 ] [ 0 | 9 ] [ 1 | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ | 10 ] [ | 11 ] [ 0 | 8 ] [ 1 | 7 ]

омега = 0.0

Полученная транспортная таблица:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ 1 | 10 ] [ | 8 ] [ | 6 ] [ | 4 ] [ | 9 ]

{S[ 1] = 1} [ | 11 ] [ 1 | 9 ] [ | 10 ] [ | 5 ] [ | 6 ]

{S[ 2] = 1} [ | 5 ] [ 0 | 10 ] [ 1 | 8 ] [ | 6 ] [ | 4 ]

{S[ 3] = 1} [ 0 | 3 ] [ | 11 ] [ 0 | 9 ] [ 1 | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ | 10 ] [ | 11 ] [ 0 | 8 ] [ 1 | 7 ]

Оценки:

d(0,1) = -10.0

d(0,2) = -10.0

d(0,3) = -9.0

d(0,4) = -3.0

d(1,0) = 10.0

d(1,2) = 3.0

d(1,3) = 1.0

d(1,4) = 3.0

d(2,0) = 3.0

d(2,3) = 1.0

d(2,4) = 0.0

d(3,1) = 0.0

d(3,4) = 1.0

d(4,0) = 3.0

d(4,1) = -3.0

d(4,2) = 0.0

2 итерация

В базис включена переменная (0,1)

Построенный цикл: (0,1)(2,1)(2,2)(3,2)(3,0)(0,0)

Из базиса исключена переменная (2,1)

Транспортная таблица до перекачки:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ 1 | 10 ] [ 0 | 8 ] [ | 6 ] [ | 4 ] [ | 9 ]

{S[ 1] = 1} [ | 11 ] [ 1 | 9 ] [ | 10 ] [ | 5 ] [ | 6 ]

{S[ 2] = 1} [ | 5 ] [ | 10 ] [ 1 | 8 ] [ | 6 ] [ | 4 ]

{S[ 3] = 1} [ 0 | 3 ] [ | 11 ] [ 0 | 9 ] [ 1 | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ | 10 ] [ | 11 ] [ 0 | 8 ] [ 1 | 7 ]

омега = 0.0

Полученная транспортная таблица:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ 1 | 10 ] [ 0 | 8 ] [ | 6 ] [ | 4 ] [ | 9 ]

{S[ 1] = 1} [ | 11 ] [ 1 | 9 ] [ | 10 ] [ | 5 ] [ | 6 ]

{S[ 2] = 1} [ | 5 ] [ | 10 ] [ 1 | 8 ] [ | 6 ] [ | 4 ]

{S[ 3] = 1} [ 0 | 3 ] [ | 11 ] [ 0 | 9 ] [ 1 | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ | 10 ] [ | 11 ] [ 0 | 8 ] [ 1 | 7 ]

Оценки:

d(0,2) = -10.0

d(0,3) = -9.0

d(0,4) = -3.0

d(1,0) = 0.0

d(1,2) = -7.0

d(1,3) = -9.0

d(1,4) = -7.0

d(2,0) = 3.0

d(2,1) = 10.0

d(2,3) = 1.0

d(2,4) = 0.0

d(3,1) = 10.0

d(3,4) = 1.0

d(4,0) = 3.0

d(4,1) = 7.0

d(4,2) = 0.0

3 итерация

В базис включена переменная (0,2)

Построенный цикл: (0,2)(3,2)(3,0)(0,0)

Из базиса исключена переменная (3,2)

Транспортная таблица до перекачки:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ 1 | 10 ] [ 0 | 8 ] [ 0 | 6 ] [ | 4 ] [ | 9 ]

{S[ 1] = 1} [ | 11 ] [ 1 | 9 ] [ | 10 ] [ | 5 ] [ | 6 ]

{S[ 2] = 1} [ | 5 ] [ | 10 ] [ 1 | 8 ] [ | 6 ] [ | 4 ]

{S[ 3] = 1} [ 0 | 3 ] [ | 11 ] [ | 9 ] [ 1 | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ | 10 ] [ | 11 ] [ 0 | 8 ] [ 1 | 7 ]

омега = 0.0

Полученная транспортная таблица:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ 1 | 10 ] [ 0 | 8 ] [ 0 | 6 ] [ | 4 ] [ | 9 ]

{S[ 1] = 1} [ | 11 ] [ 1 | 9 ] [ | 10 ] [ | 5 ] [ | 6 ]

{S[ 2] = 1} [ | 5 ] [ | 10 ] [ 1 | 8 ] [ | 6 ] [ | 4 ]

{S[ 3] = 1} [ 0 | 3 ] [ | 11 ] [ | 9 ] [ 1 | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ | 10 ] [ | 11 ] [ 0 | 8 ] [ 1 | 7 ]

Оценки:

d(0,3) = -9.0

d(0,4) = -3.0

d(1,0) = 0.0

d(1,2) = 3.0

d(1,3) = -9.0

d(1,4) = -7.0

d(2,0) = -7.0

d(2,1) = 0.0

d(2,3) = -9.0

d(2,4) = -10.0

d(3,1) = 10.0

d(3,2) = 10.0

d(3,4) = 1.0

d(4,0) = 3.0

d(4,1) = 7.0

d(4,2) = 10.0

4 итерация

В базис включена переменная (2,4)

Построенный цикл: (2,4)(4,4)(4,3)(3,3)(3,0)(0,0)(0,2)(2,2)

Из базиса исключена переменная (4,4)

Транспортная таблица до перекачки:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ 1 | 10 ] [ 0 | 8 ] [ 0 | 6 ] [ | 4 ] [ | 9 ]

{S[ 1] = 1} [ | 11 ] [ 1 | 9 ] [ | 10 ] [ | 5 ] [ | 6 ]

{S[ 2] = 1} [ | 5 ] [ | 10 ] [ 1 | 8 ] [ | 6 ] [ 0 | 4 ]

{S[ 3] = 1} [ 0 | 3 ] [ | 11 ] [ | 9 ] [ 1 | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ | 10 ] [ | 11 ] [ 0 | 8 ] [ | 7 ]

омега = 1.0

Полученная транспортная таблица:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ 0 | 10 ] [ 0 | 8 ] [ 1 | 6 ] [ | 4 ] [ | 9 ]

{S[ 1] = 1} [ | 11 ] [ 1 | 9 ] [ | 10 ] [ | 5 ] [ | 6 ]

{S[ 2] = 1} [ | 5 ] [ | 10 ] [ 0 | 8 ] [ | 6 ] [ 1 | 4 ]

{S[ 3] = 1} [ 1 | 3 ] [ | 11 ] [ | 9 ] [ 0 | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ | 10 ] [ | 11 ] [ 1 | 8 ] [ | 7 ]

Оценки:

d(0,3) = -9.0

d(0,4) = 7.0

d(1,0) = 0.0

d(1,2) = 3.0

d(1,3) = -9.0

d(1,4) = 3.0

d(2,0) = -7.0

d(2,1) = 0.0

d(2,3) = -9.0

d(3,1) = 10.0

d(3,2) = 10.0

d(3,4) = 11.0

d(4,0) = 3.0

d(4,1) = 7.0

d(4,2) = 10.0

d(4,4) = 10.0

5 итерация

В базис включена переменная (0,3)

Построенный цикл: (0,3)(3,3)(3,0)(0,0)

Из базиса исключена переменная (3,3)

Транспортная таблица до перекачки:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ 0 | 10 ] [ 0 | 8 ] [ 1 | 6 ] [ 0 | 4 ] [ | 9 ]

{S[ 1] = 1} [ | 11 ] [ 1 | 9 ] [ | 10 ] [ | 5 ] [ | 6 ]

{S[ 2] = 1} [ | 5 ] [ | 10 ] [ 0 | 8 ] [ | 6 ] [ 1 | 4 ]

{S[ 3] = 1} [ 1 | 3 ] [ | 11 ] [ | 9 ] [ | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ | 10 ] [ | 11 ] [ 1 | 8 ] [ | 7 ]

омега = 0.0

Полученная транспортная таблица:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ 0 | 10 ] [ 0 | 8 ] [ 1 | 6 ] [ 0 | 4 ] [ | 9 ]

{S[ 1] = 1} [ | 11 ] [ 1 | 9 ] [ | 10 ] [ | 5 ] [ | 6 ]

{S[ 2] = 1} [ | 5 ] [ | 10 ] [ 0 | 8 ] [ | 6 ] [ 1 | 4 ]

{S[ 3] = 1} [ 1 | 3 ] [ | 11 ] [ | 9 ] [ | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ | 10 ] [ | 11 ] [ 1 | 8 ] [ | 7 ]

Оценки:

d(0,4) = 7.0

d(1,0) = 0.0

d(1,2) = 3.0

d(1,3) = 0.0

d(1,4) = 3.0

d(2,0) = -7.0

d(2,1) = 0.0

d(2,3) = 0.0

d(3,1) = 10.0

d(3,2) = 10.0

d(3,3) = 9.0

d(3,4) = 11.0

d(4,0) = -6.0

d(4,1) = -2.0

d(4,2) = 1.0

d(4,4) = 1.0

6 итерация

В базис включена переменная (2,0)

Построенный цикл: (2,0)(0,0)(0,2)(2,2)

Из базиса исключена переменная (0,0)

Транспортная таблица до перекачки:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ | 10 ] [ 0 | 8 ] [ 1 | 6 ] [ 0 | 4 ] [ | 9 ]

{S[ 1] = 1} [ | 11 ] [ 1 | 9 ] [ | 10 ] [ | 5 ] [ | 6 ]

{S[ 2] = 1} [ 0 | 5 ] [ | 10 ] [ 0 | 8 ] [ | 6 ] [ 1 | 4 ]

{S[ 3] = 1} [ 1 | 3 ] [ | 11 ] [ | 9 ] [ | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ | 10 ] [ | 11 ] [ 1 | 8 ] [ | 7 ]

омега = 0.0

Полученная транспортная таблица:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ | 10 ] [ 0 | 8 ] [ 1 | 6 ] [ 0 | 4 ] [ | 9 ]

{S[ 1] = 1} [ | 11 ] [ 1 | 9 ] [ | 10 ] [ | 5 ] [ | 6 ]

{S[ 2] = 1} [ 0 | 5 ] [ | 10 ] [ 0 | 8 ] [ | 6 ] [ 1 | 4 ]

{S[ 3] = 1} [ 1 | 3 ] [ | 11 ] [ | 9 ] [ | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ | 10 ] [ | 11 ] [ 1 | 8 ] [ | 7 ]

Оценки:

d(0,0) = 7.0

d(0,4) = 7.0

d(1,0) = 7.0

d(1,2) = 3.0

d(1,3) = 0.0

d(1,4) = 3.0

d(2,1) = 0.0

d(2,3) = 0.0

d(3,1) = 3.0

d(3,2) = 3.0

d(3,3) = 2.0

d(3,4) = 4.0

d(4,0) = 1.0

d(4,1) = -2.0

d(4,2) = 1.0

d(4,4) = 1.0

7 итерация

В базис включена переменная (4,1)

Построенный цикл: (4,1)(0,1)(0,3)(4,3)

Из базиса исключена переменная (0,1)

Транспортная таблица до перекачки:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ | 10 ] [ | 8 ] [ 1 | 6 ] [ 0 | 4 ] [ | 9 ]

{S[ 1] = 1} [ | 11 ] [ 1 | 9 ] [ | 10 ] [ | 5 ] [ | 6 ]

{S[ 2] = 1} [ 0 | 5 ] [ | 10 ] [ 0 | 8 ] [ | 6 ] [ 1 | 4 ]

{S[ 3] = 1} [ 1 | 3 ] [ | 11 ] [ | 9 ] [ | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ 0 | 10 ] [ | 11 ] [ 1 | 8 ] [ | 7 ]

омега = 0.0

Полученная транспортная таблица:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ | 10 ] [ | 8 ] [ 1 | 6 ] [ 0 | 4 ] [ | 9 ]

{S[ 1] = 1} [ | 11 ] [ 1 | 9 ] [ | 10 ] [ | 5 ] [ | 6 ]

{S[ 2] = 1} [ 0 | 5 ] [ | 10 ] [ 0 | 8 ] [ | 6 ] [ 1 | 4 ]

{S[ 3] = 1} [ 1 | 3 ] [ | 11 ] [ | 9 ] [ | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ 0 | 10 ] [ | 11 ] [ 1 | 8 ] [ | 7 ]

Оценки:

d(0,0) = 7.0

d(0,1) = 2.0

d(0,4) = 7.0

d(1,0) = 5.0

d(1,2) = 1.0

d(1,3) = -2.0

d(1,4) = 1.0

d(2,1) = 2.0

d(2,3) = 0.0

d(3,1) = 5.0

d(3,2) = 3.0

d(3,3) = 2.0

d(3,4) = 4.0

d(4,0) = 1.0

d(4,2) = 1.0

d(4,4) = 1.0

8 итерация

В базис включена переменная (1,3)

Построенный цикл: (1,3)(4,3)(4,1)(1,1)

Из базиса исключена переменная (4,3)

Транспортная таблица до перекачки:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ | 10 ] [ | 8 ] [ 1 | 6 ] [ 0 | 4 ] [ | 9 ]

{S[ 1] = 1} [ | 11 ] [ 1 | 9 ] [ | 10 ] [ 0 | 5 ] [ | 6 ]

{S[ 2] = 1} [ 0 | 5 ] [ | 10 ] [ 0 | 8 ] [ | 6 ] [ 1 | 4 ]

{S[ 3] = 1} [ 1 | 3 ] [ | 11 ] [ | 9 ] [ | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ 0 | 10 ] [ | 11 ] [ | 8 ] [ | 7 ]

омега = 1.0

Полученная транспортная таблица:

{D[0]= 1} {D[1]= 1} {D[2]= 1} {D[3]= 1} {D[4]= 1}

{S[ 0] = 1} [ | 10 ] [ | 8 ] [ 1 | 6 ] [ 0 | 4 ] [ | 9 ]

{S[ 1] = 1} [ | 11 ] [ 0 | 9 ] [ | 10 ] [ 1 | 5 ] [ | 6 ]

{S[ 2] = 1} [ 0 | 5 ] [ | 10 ] [ 0 | 8 ] [ | 6 ] [ 1 | 4 ]

{S[ 3] = 1} [ 1 | 3 ] [ | 11 ] [ | 9 ] [ | 6 ] [ | 6 ]

{S[ 4] = 1} [ | 8 ] [ 1 | 10 ] [ | 11 ] [ | 8 ] [ | 7 ]

Оценки:

d(0,0) = 7.0

d(0,1) = 0.0

d(0,4) = 7.0

d(1,0) = 7.0

d(1,2) = 3.0

d(1,4) = 3.0

d(2,1) = 0.0

d(2,3) = 0.0

d(3,1) = 3.0

d(3,2) = 3.0

d(3,3) = 2.0

d(3,4) = 4.0

d(4,0) = 3.0

d(4,2) = 3.0

d(4,3) = 2.0

d(4,4) = 3.0

Ответ:

0,0 0,0 1,0 0,0 0,0

0,0 0,0 0,0 1,0 0,0

0,0 0,0 0,0 0,0 1,0

1,0 0,0 0,0 0,0 0,0

0,0 1,0 0,0 0,0 0,0

Стоимость перевозок: 28.0