Министерство образования Республики Беларусь

Учреждение образования «Белорусский Государственный Университет Информатики и Радиоэлектроники»

Кафедра «Информатика»

Лабораторная работа №5

«Задача квадратичного программирования»

**Выполнил:**

Волчецкий А.М., гр. 253504

**Проверила:**

Саттарова П.С.

Минск 2015

**Задание:**

Решить задачу квадратичного программирования.

**Листинг:**

public bool Solve(out DenseVector sol, out string report)

{

Report = "";

sol = null;

bool is2StepMustBeSkiped = false;

RecalculateBasisAndStarVariables();

bool isSolved = false;

for (var i = 0; i < IterCount; i++)

{

Report += string.Format("\n{0} итерация", i);

if (!is2StepMustBeSkiped)

{

Step1BuildPotencialsAndEstimations();

if (Step2CheckForOptimumAndCalculateJ0())

{

sol = \_x;

Report += string.Format("\n\nОценки:\n{0}", \_estimation.Aggregate("", (acc, x) => $"{acc}det{x.Key} = {x.Value}\n"));

isSolved = true;

break;

}

}

Report += string.Format("\n\nОценки:\n{0}", \_estimation.Aggregate("", (acc, x) => $"{acc}det{x.Key} = {x.Value}\n"));

Step3BuildLDirection();

Report += string.Format("\nВектор l:\n{0}", Helper.ToPrint(\_l));

Step4CalculateTet0();

Report += string.Format("\nTeta{0} = {1}\n", \_tet0.Key, \_tet0.Value);

if (!IsTargetFunctionHasBottomBondary())

{

isSolved = false;

break;

}

Step5BuildNewPlan();

Report += string.Format("\nНовый базисный план:\n{0}", Helper.ToPrint(\_x));

is2StepMustBeSkiped = Step6RecalculateJbAndJStar();

Report += string.Format("\nJop:\n[{0}]", \_jBasis.Aggregate("", (a, x) => a + x.ToString() + ", "));

Report += string.Format("\n\nJ\*:\n[{0}]", \_jStar.Aggregate("", (a, x) => a + x.ToString() + ", "));

RecalculateBasisAndStarVariables();

}

report = Report;

return isSolved;

}

#endregion

#region Private methods

private static bool IsZero(double d)

{

return -Eps < d && d < Eps;

}

private static bool IsGreaterZero(double d)

{

return d > Eps;

}

private static bool IsLessZero(double d)

{

return d < -Eps;

}

private static bool IsGreaterOrEqualZero(double d)

{

return IsZero(d) || IsGreaterZero(d);

}

private void RecalculateBasisAndStarVariables()

{

int m = \_a.RowCount;

int n = \_a.ColumnCount;

\_aBasis = new DenseMatrix(m, \_jBasis.Count);

\_aStar = new DenseMatrix(m, \_jStar.Count);

\_cNewBasis = new DenseVector(\_jBasis.Count);

\_dBasis = new DenseMatrix(\_jBasis.Count, \_jBasis.Count);

\_dStar = new DenseMatrix(\_jStar.Count, \_jStar.Count);

\_xBasis = new DenseVector(\_jBasis.Count);

\_jNotBasis = new List<int>();

\_cNew = \_c + \_d \* \_x;

Report += string.Format("\n\nВектор c:\n{0}", Helper.ToPrint(\_cNew));

for (var i = 0; i < m; i++)

{

for (var j = 0; j < \_jStar.Count; j++)

{

\_aStar[i, j] = \_a[i, \_jStar[j]];

}

}

for (var i = 0; i < m; i++)

{

for (var j = 0; j < \_jBasis.Count; j++)

{

\_aBasis[i, j] = \_a[i, \_jBasis[j]];

}

}

for (var i = 0; i < \_jBasis.Count; i++)

{

for (var j = 0; j < \_jBasis.Count; j++)

{

\_dBasis[i, j] = \_d[\_jBasis[i], \_jBasis[j]];

}

}

for (var i = 0; i < \_jStar.Count; i++)

{

for (var j = 0; j < \_jStar.Count; j++)

{

\_dStar[i, j] = \_d[\_jStar[i], \_jStar[j]];

}

}

for (var i = 0; i < \_jBasis.Count; i++)

{

\_cNewBasis[i] = \_cNew[\_jBasis[i]];

\_xBasis[i] = \_x[\_jBasis[i]];

}

for (var i = 0; i < n; i++)

{

if (!\_jStar.Contains(i))

{

\_jNotBasis.Add(i);

}

}

}

private void Step1BuildPotencialsAndEstimations()

{

var potencialsT = (DenseVector)((-1) \* \_cNewBasis \* \_aBasis.Inverse());

Report += string.Format("\n\nПотенциалы:{0}", Helper.ToPrint(potencialsT));

\_estimation = new Dictionary<int, double>();

foreach (var j in \_jNotBasis)

{

double delta = potencialsT \* \_a.Column(j) + \_cNew[j];

\_estimation.Add(j, delta);

}

}

private bool Step2CheckForOptimumAndCalculateJ0()

{

\_j0 = -1;

foreach (KeyValuePair<int, double> pair in \_estimation.Where(pair => IsLessZero(pair.Value)))

{

\_j0 = pair.Key;

Report += string.Format("\nj0 = {0}", \_j0);

return false;

}

return true;

}

private void BuildHStarMatrix()

{

int m = \_a.RowCount;

int k = \_jStar.Count;

\_hStar = new DenseMatrix(k + m, k + m);

//D\*

for (var i = 0; i < k; i++)

{

for (var j = 0; j < k; j++)

{

\_hStar[i, j] = \_dStar[i, j];

}

}

Report += string.Format("\nD\*:\n{0}", Helper.ToPrint(\_hStar));

//A\*

for (var i = 0; i < m; i++)

{

for (var j = 0; j < k; j++)

{

\_hStar[i + k, j] = \_aStar[i, j];

}

}

Report += string.Format("\nA\*:\n{0}", Helper.ToPrint(\_hStar));

//A\*'

var aStarT = (DenseMatrix)\_aStar.Transpose();

for (var i = 0; i < k; i++)

{

for (var j = 0; j < m; j++)

{

\_hStar[i, j + k] = aStarT[i, j];

}

}

Report += string.Format("\nAT\*:\n{0}", Helper.ToPrint(\_hStar));

}

private void Step3BuildLDirection()

{

BuildHStarMatrix();

int k = \_jStar.Count;

int m = \_a.RowCount;

DenseVector hJ0 = new DenseVector(k + m);

//at D\* indexes

int j0 = \_jNotBasis.IndexOf(\_j0);

if (j0 < 0)

{

throw new Exception("j0 not in JNotBasis");

}

//D\*j0

for (var i = 0; i < k; i++)

{

hJ0[i] = \_d[\_jStar[i], \_j0];

}

for (var i = 0; i < m; i++)

{

hJ0[i + k] = \_a[i, \_j0];

}

var lStarY = (-1) \* \_hStar.Inverse() \* hJ0;

\_l = new DenseVector(\_a.ColumnCount);

for (var i = 0; i < k; i++)

{

\_l[\_jStar[i]] = lStarY[i];

}

\_l[\_j0] = 1;

}

private void Step4CalculateTet0()

{

Dictionary<int, double> tetS = new Dictionary<int, double>();

foreach (var j in \_jStar)

{

if (IsGreaterOrEqualZero(\_l[j]))

{

tetS.Add(j, double.MaxValue);

}

else

{

tetS.Add(j, -\_x[j] / \_l[j]);

}

}

\_sigma = \_l \* \_d \* \_l;//??? \_l.Copy().Transpose().Multiply(\_d).Multiply(\_l)[0, 0]; what is (\_l)[0, 0]

if (!\_jStar.Contains(\_j0))

{

if (IsZero(\_sigma))

{

tetS.Add(\_j0, double.MaxValue);

}

else if (IsGreaterZero(\_sigma))

{

tetS.Add(\_j0, Math.Abs(\_estimation[\_j0]) / \_sigma);

}

else

{

//

//throw new Exception("Can't determine condition path in Step 4");

}

}

\_tet0 = new KeyValuePair<int, double>(-1, double.MaxValue);

foreach (var pair in tetS.Where(pair => pair.Value < \_tet0.Value))

{

\_tet0 = pair;

}

}

private bool IsTargetFunctionHasBottomBondary()

{

return \_tet0.Value != double.MaxValue;

}

private void Step5BuildNewPlan()

{

\_x = \_x + \_l\* \_tet0.Value;

}

//return true if first 2 steps must be skipped

private bool Step6RecalculateJbAndJStar()

{

if (\_tet0.Key == \_j0)

{

\_jStar.Add(\_j0);

//\_jStar.Sort();

return false;

}

if (\_jStar.Contains(\_tet0.Key) && !\_jBasis.Contains(\_tet0.Key))

{

\_jStar.Remove(\_tet0.Key);

//\_jStar.Sort();

\_estimation[\_j0] = \_estimation[\_j0] + \_tet0.Value \* \_sigma;

return true;

}

bool is3DStep = false;

int jPlus = -1;

if (\_jBasis.Contains(\_tet0.Key))

{

DenseMatrix aBasisInv = (DenseMatrix)\_aBasis.Inverse();

for (var i = 0; i < \_jBasis.Count; i++)

{

if (\_jStar.Contains(i) && !\_jBasis.Contains(i))

{

DenseVector eS = new DenseVector(\_aBasis.RowCount) {[\_jBasis.IndexOf(\_tet0.Key)] = 1};

if (!IsZero(eS \* aBasisInv \* \_a.Column(i)))

{

jPlus = i;

is3DStep = true;

break;

}

}

}

}

if (is3DStep)

{

\_jBasis.Remove(\_tet0.Key);

\_jBasis.Add(jPlus);

//\_jBasis.Sort();

\_jStar.Remove(\_tet0.Key);

//\_jStar.Sort();

\_estimation[\_j0] = \_estimation[\_j0] + \_tet0.Value \* \_sigma;

return true;

}

\_jBasis.Remove(\_tet0.Key);

\_jBasis.Add(\_j0);

//\_jBasis.Sort();

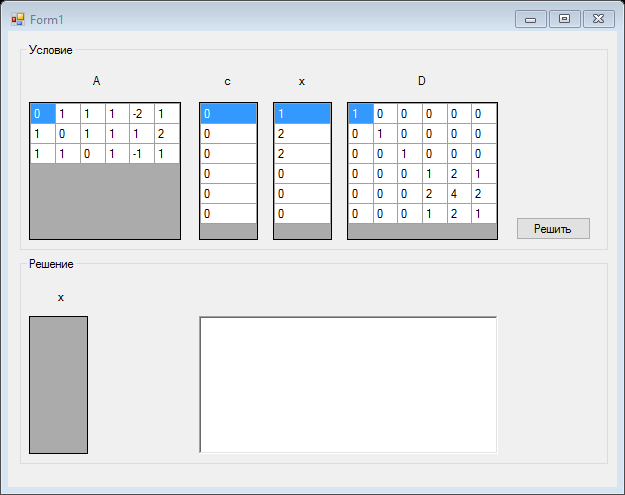
\_jStar.Remove(\_tet0.Key);

\_jStar.Add(\_j0);

//\_jStar.Sort();

return false; }

**Пример:**

****