Исследование операций

Лабораторная работа №3
Транспортная задача

Отчет

Студент: Корепанов Кирилл

Группа: ИУ7-83

Вариант: 4

2013

Постановка транспортной задачи

Содержательная

Имеется m производителей с мощностями $S_i \geq 0$ и n потребителей с мощностями $D_j \geq 0$. Предполагается, что $\sum_{i=1}^{m} S_i = \sum_{j=1}^{n} D_j$. Стоимость перевозки единицы продукции от i-го производителя к j-му потребителю обозначается $c_{ij} \geq 0$. Требуется:

1) Перевезти всю продукцию от производителей к потребителям с учетом ограничений по мощности
2) Общая стоимость перевозок должна быть минимальной

Математическая

$$\begin{cases} 1) \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \to min \\ 2) \sum_{j=1}^{n} x_{ij} = S_i, i = 1..m \\ 3) \sum_{i=1}^{m} x_{ij} = D_j, j = 1..n \\ 4) x_{ij} \geq 0, i = 1..m, j = 1..n \end{cases}$$

Описание метода потенциалов

Вход: вектор мощностей производителей S, вектор мощностей потребителей D, матрица стоимостей C
Выход: матрица X

Строим первоначальную БДР (метод СЗ угла)
Для базисных клеток составляем СЛАУ ui+vj=cij
Решаем СЛАУ, придавая произвольным неизвестным произвольное значение (0)
Для небазисных клеток
    dij = cij-ui-vj
Все dij >= 0
    Вывести текущее БДР
Выбрать min dij – вводим его в базис
Строим цикл транспортной задачи, начиная с введенной в базис клетки
В цикле расставляем +w, -w, …, -w начиная с введенной в базис клетки
Среди клеток с –w находим минимальную xij и исключаем ее из базиса
Производим перекачку по циклу (w)

Текст программы

```
private static List<PStatus> log;
private static int[] initialSources;
private static int[] initialDestinations;
private static int[,] prices;
private static int nS, nD;
private static int[,] counts;
private static int[] sources;
private static int[] destinations;
private static List<Coordinate> basis = new List<Coordinate>();
private static List<Coordinate> path = new List<Coordinate>();
private static Random rand = new Random(System.DateTime.Now.Millisecond);
private static string[] iterations = new string[] {
  "Еще один оборот\nКолесо Судьбы совершило\nПора за курсач...",
  "Ночь, улица, фонарь, аптека\nБессмысленный и тусклый свет\nЖиви еще хоть четверть века\nВсе будет так, исхода нет",
  "Идешь вперед, преодолевая пространство и время\nВ попытках отыскать смысл путис\nКонец один, ты знаешь его",
  "We do what we must\nBecause we can\nFor the good of all of us\nExcept the ones who are dead",
  "We just keep on trying\nTill we ran out of cake",
  "И днём и ночью Кот ученый\nВсе ходит по цепи кругом\nБедный кот...",
  "You spin me round round~",
  "А вы знали, что код цикла записан как\ndo { ... } while (true);\nи выхода нет? =("
};
public static List<PStatus> solve(int[] source, int[] destination,
  int[,] price) {
  log = new List<PStatus>();
  initialSources = Util.copeOne<int>(source);
  sources = Util.copeOne<int>(source);
  nS = sources.Length;
  initialDestinations = Util.copeOne<int>(destination);
  destinations = Util.copeOne<int>(destination);
  nD = destinations.Length;
  prices = Util.copy<int>(price);
  counts = new int[nS, nD];
  basis = new List<Coordinate>();
  path = new List<Coordinate>();
  writeLog("В начале было слово...");
  getNordWestPath(out counts, out basis);
  writeLog("Путь на северо-запад начинается\nс одной итерации");
  do {
    int[] uI, vJ;
    solveSlau(out uI, out vJ);
    int[,] dIJ = computeD(uI, vJ);
    Coordinate min = getMinElement(dIJ);
    if (dIJ[min.row, min.col] >= 0) {
      path.Clear();
      basis = basis.Where(p => (counts[p.row, p.col] > 0)).ToList();
      // Решение найдено
      writeLog("На сим закончена история моя");
      return log;
    }
    // Вводим min(dIJ) в базис
    basis.Add(min);
    path = getWPath(min);
    Coordinate fromBasis = getMinCount(path);
    int w = counts[fromBasis.row, fromBasis.col];
    for (int i = 0; i < path.Count; i++) {
      counts[path[i].row, path[i].col] += (i % 2 == 0 ? 1 : -1) * w;
    }
    removeFromBasis(fromBasis);
    writeLog(iterations[rand.Next(iterations.Length)]);
  } while (log.Count < 1e3);
  return log; // errora =(
}
private static Coordinate getMinCount(List<Coordinate> path) {
  int min = int.MaxValue;
  Coordinate coord = new Coordinate(-1, -1);
  for (int i = 1; i < path.Count; i += 2) {
    int value = counts[path[i].row, path[i].col];
    if (min > value) {
      min = value;
      coord.row = path[i].row;
      coord.col = path[i].col;
    }
  }
  return coord;
}
private static List<Coordinate> getWPath(Coordinate start) {
  List<Coordinate> startPath = new List<Coordinate>();
  startPath.Add(start);
  return pathRec(startPath, true);
  List<Coordinate> path = new List<Coordinate>();
  path.Add(start);
  int row = start.row, col = start.col;
```

```
  do {
    // По столбцу
    for (int i = 0; i < nS; i++) {
      // Базисная необработанная ячейка
      if (inBasis(i, col) && !inList(i, col, path)) {
        bool isGood = false;
        // В этой строке есть другие элементы
        for (int j = 0; j < nD; j++) {
          if (j != col && inBasis(i, j)) {
            isGood = true;
            break;
          }
        }
        if (isGood) {
          path.Add(new Coordinate(i, col));
          row = i;
          break;
        }
      }
    }
    // Можем вернуться на старт - закончили цикл
    if (row == start.row)
      break;
    // По (новой) строке
    for (int i = 0; i < nD; i++) {
      // Базисная необработанная ячейка
      if (inBasis(row, i) && !inList(row, i, path)) {
        bool isGood = false;
        // В этой столбце есть другие элементы
        for (int j = 0; j < nS; j++) {
          if (j != row &&
            (inBasis(j, i) || (j == start.row && i == start.col))) {
            isGood = true;
            break;
          }
        }
        if (isGood) {
          path.Add(new Coordinate(row, i));
          col = i;
          break;
        }
      }
    }
  }
  while (true);
  return path;
}
private static List<Coordinate> pathRec(List<Coordinate> currentPath, bool doRow) {
  List<Coordinate> newPath = new List<Coordinate>(currentPath);
  Coordinate start = currentPath[0];
  Coordinate current = currentPath[currentPath.Count - 1];
  Coordinate newPoint = new Coordinate(-1, -1);
  newPath.Add(newPoint);
  if (doRow) {
    // По строке
    for (int i = 0; i < nD; i++) {
      // Базисная необработанная ячейка
      if (inBasis(current.row, i) && !inList(current.row, i, newPath)) {
        // Добавляем ячейку в путь и проверяем ее ркурсивно
        newPoint.row = current.row;
        newPoint.col = i;
        List<Coordinate> recursivePath = pathRec(newPath, false);
        if (recursivePath.Count != 0) {
          return recursivePath;
        }
      }
    }
    // Ошибочный вариант пути
    return new List<Coordinate>();
  }
  else {
    // По столбцу
    for (int i = 0; i < nS; i++) {
      // вернулись на старт
      if (start.row == i && start.col == current.col) {
        newPath.RemoveAt(newPath.Count - 1); // Удаляем последнюю точку (-1, -1)
        return newPath;
      }
      // Базисная необработанная ячейка
      if (inBasis(i, current.col) && !inList(i, current.col, newPath)) {
        // Добавляем ячейку в путь и проверяем ее ркурсивно
        newPoint.row = i;
        newPoint.col = current.col;
        List<Coordinate> recursivePath = pathRec(newPath, true);
        if (recursivePath.Count != 0) {
          return recursivePath;
        }
      }
```

```
    }
    // Ошибочный вариант пути
    return new List<Coordinate>();
  }
}
private static Coordinate getMinElement(int[,] matrix) {
  Coordinate coord = new Coordinate(-1, -1);
  int min = int.MaxValue;
  for (int i = 0; i < nS; i++)
    for (int j = 0; j < nD; j++)
      if (min > matrix[i, j] && !inBasis(i, j)) {
        min = matrix[i, j];
        coord.row = i;
        coord.col = j;
      }
  return coord;
}
private static int[,] computeD(int[] uI, int[] vJ) {
  int[,] d = new int[nS, nD];
  for (int i = 0; i < nS; i++)
    for (int j = 0; j < nD; j++)
      if (!inBasis(i, j))
        d[i, j] = prices[i, j] - uI[i] - vJ[j];
  return d;
}
private static void solveSlau(out int[] uI, out int[] vJ) {
  uI = new int[nS];
  vJ = new int[nD];
  // Принимаем последнее u за 0
  // Формат: v1+...+vn+u1+...+um=cij
  int n = nS + nD - 1;
  int nEq = 0;
  double[,] x = new double[n, n];
  double[] y = new double[n];
  foreach (Coordinate coord in basis) {
    // Добавляем уравнение
    x[nEq, coord.col] = 1;    // vj
    if (coord.row != nS - 1)
      x[nEq, nD + coord.row] = 1; // ui
    y[nEq] = prices[coord.row, coord.col];
    nEq++;
  }
  alglib.densesolverreport report;
  int info;
  double[] result;
  alglib.rmatrixsolve(x, n, y, out info, out report, out result);
  // Копируем обратно
  for (int i = 0; i < nD; i++)
    vJ[i] = (int)result[i];
  for (int i = 0; i < nS - 1; i++)
    uI[i] = (int)result[nD + i];
  uI[nS - 1] = 0;
}
private static bool inBasis(int row, int col) {
  return inList(row, col, basis);
}
private static bool inList(int row, int col, List<Coordinate> list) {
  foreach (Coordinate coord in list)
    if (coord.row == row && coord.col == col)
      return true;
  return false;
}
private static void removeFromBasis(Coordinate el) {
  foreach (Coordinate coord in basis) {
    if (coord.row == el.row && coord.col == el.col) {
      basis.Remove(coord);
      return;
    }
  }
}
private static void writeLog(string comment = "") {
  int f = 0;
  for(int i = 0; i < nS; i++)
    for (int j = 0; j < nD; j++) {
      f += prices[i, j] * counts[i, j];
    }
  log.Add(new PStatus(sources, destinations, prices, counts, basis, path, f, comment));
}
public static void getNordWestPath(out int[,] counts, out List<Coordinate> basis) {
  int nS = sources.Length;
  int nD = destinations.Length;
  counts = new int[nS, nD];
  basis = new List<Coordinate>(nS + nD - 1);
  int sum = 0;
  foreach (int s in sources) {
    sum += s;
  }
  foreach (int d in destinations) {
```

```
    sum -= d;
  }
  if (sum != 0) {
    return;
  }
  int i = 0;
  int j = 0;
  while (destinations[nD - 1] != 0) {
    basis.Add(new Coordinate(i, j));
    int send = Math.Min(sources[i], destinations[j]);
    counts[i, j] += send;
    sources[i] -= send;
    destinations[j] -= send;
    if (sources[i] == 0)
      i++;
    else //if (destinations[j] == 0)
      j++;
  };
}
```

Исходные данные

Вариант №4

	75	80	60	85
100	0 | 6	0 | 7	0 | 3	0 | 5
150	0 | 1	0 | 2	0 | 5	0 | 6
50	0 | 8	0 | 10	0 | 20	0 | 1

Транспортная задача

	1	1	1	1	1
1	0 | 3	0 | 5	0 | 2	0 | 4	0 | 8
1	0 | 10	0 | 10	0 | 4	0 | 3	0 | 6
1	0 | 5	0 | 6	0 | 9	0 | 8	0 | 3
1	0 | 6	0 | 2	0 | 5	0 | 8	0 | 4
1	0 | 5	0 | 4	0 | 8	0 | 9	0 | 3

Задача о назначениях (из ЛР №1)

Результат работы

	75	80	60	85
100	5 | 6	0 | 7	60 | 3	35 | 5
150	70 | 1	80 | 2	0 | 5	0 | 6
50	0 | 8	0 | 10	0 | 20	50 | 1

f = 665

Транспортная задача

	1	1	1	1	1
1	0 | 3	0 | 5	1 | 2	0 | 4	0 | 8
1	0 | 10	0 | 10	0 | 4	1 | 3	0 | 6
1	1 | 5	0 | 6	0 | 9	0 | 8	0 | 3
1	0 | 6	1 | 2	0 | 5	0 | 8	0 | 4
1	0 | 5	0 | 4	0 | 8	0 | 9	1 | 3

f = 15 (аналогично ответу в ЛР №1)

Задача о назначениях (из ЛР №1)