Exercise Session 3

8.5.2014.

- Multi commodity flows (MCFP)
- Programming Exercise

MCFP: Given directed graph $G=(V,A)$, $c: A \to \mathbb{R}_0^+$, $u: A \to \mathbb{R}^+$
pair $(s_i,t_i)_{1\le i\le K}$ with $d_i \in \mathbb{R}^+$
$\in V^2$

solution: flows $x_{i,a}$ with minimal cost $\sum_{i=1}^{K} \sum_{a\in A} c(a) \cdot x_{i,a}$
flow value on arc a for commodity i;

- $\sum_{a\in\delta^-(v)} x_{i,a} - \sum_{a\in\delta^+(v)} x_{i,a} = \begin{cases} -d_i & v=s_i \\ +d_i & v=t_i \\ 0 & \text{else} \end{cases} \quad \forall i\in[K]$

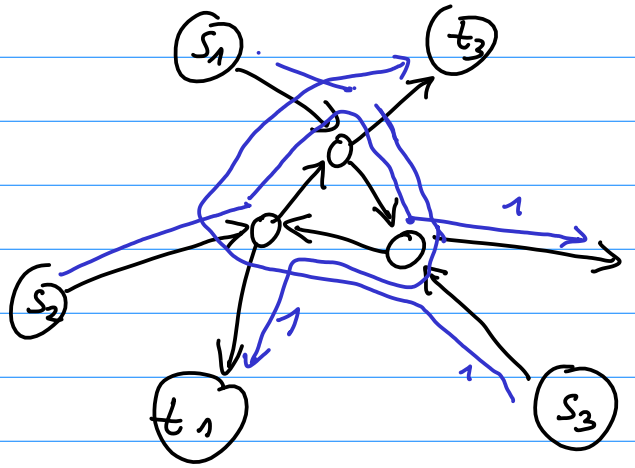
flow is conservative

- $\sum_{i\in[K]} x_{i,a} \le u(a) \quad \forall a\in A$

flow on arcs is below capacity

$x_{i,a} \ge 0 \quad \forall i\in[K], a\in A$

example:

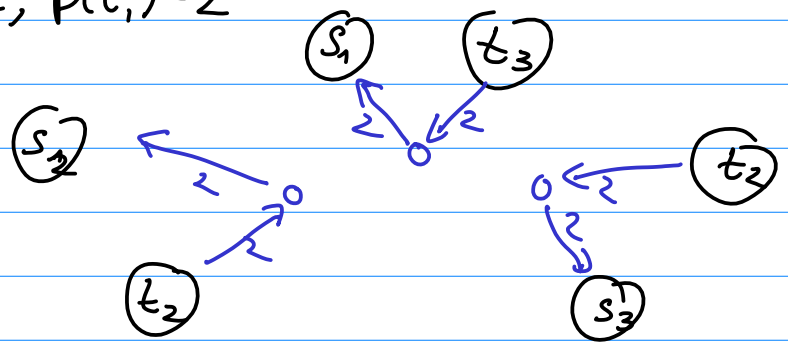


$u(a) = 2 \quad \forall a\in A$
$d_i = 1, \quad c(a) = 1 \quad \forall a\in A$

solution to MCFP

difference to b-Transshipment:

$b(s_i) = -2$, $b(t_i) = 2$



- can not use residual network as in b-Transshipment
  demands are coupled in pairs
- directed and undirected case not similar

LP formulation: arc based

$$\min \sum_{i \in [K]} \sum_{a \in A} c(a) \cdot x_{i,a}$$

- $$\sum_{a \in \delta^-(v)} x_{i,a} - \sum_{a \in \delta^+(v)} x_{i,a} = \begin{cases} -d_i & v = s_i \\ +d_i & v = t_i \\ 0 & \text{else} \end{cases} \quad \forall i \in [K], \forall v \in V$$

  flow is conservative

- $$\sum_{i \in [K]} x_{i,a} \leq u(a) \quad \forall a \in A$$

  flow on arcs is below capacity

$$x_{i,a} \geq 0 \quad \forall i \in [K], a \in A$$

Problems: LP is solvable via simplex, # constr, # variables is polynomial
⇝ possibly HUGE

- example for real life data set: $|V| = 106$ $K = 11.130 \approx |V|^2$
  $|A| = 904$

⇒ $|E| \cdot K \geq 10$ million variables
⇒ $\geq |V| \cdot K \geq 1$ million constraints ⇒ Huge Matrix A

Using Dantzig Wolfe Decomposition:

Path-based LP formulation $\rightsquigarrow$ dualize LP

$$\min \sum_{p \in \mathcal{P}} c_p \cdot x_p$$

$$\sum_{p \in \mathcal{P} : a \in p} x_p \leq u(a) \quad \forall a \in A \qquad \rightsquigarrow y_a \leq 0$$

$$\sum_{p \in \mathcal{P}_i} x_p \geq d_i \quad \forall i \in [K] \qquad \rightsquigarrow z_i \geq 0$$

$$x_p \geq 0 \quad \forall p \in \mathcal{P} \qquad \rightsquigarrow \text{constr. for all } p \in \mathcal{P}$$

dual LP: $\max \sum_{a \in A} u(a) \cdot y_a + \sum_{i \in [K]} d_i \cdot z_i$

$$z_i + \sum_{a \in p} y_a \leq c \qquad \forall p \in \mathcal{P}_i, i \in [K]$$

$y_a \leq 0 \quad \forall a \in A$ // arc prices

$z_i \geq 0 \quad \forall i \in [K]$ // potential for commodities

Idea: expon. many paths s_i-t_i, only a couple of them are interesting
- → start with a few paths variables
- → solve primal LP
- → use dual LP to find variable (path) with neg. red. costs
- → add path to primal LP and repeat

reduced cost of path p: $\bar{c} := c - \underbrace{c_B^T \cdot \bar{B}^{-1}}_{\text{dual variable}} \cdot A$

using: red. cost is negative $\Leftrightarrow$ corresponding dual constr. is violated

$\Rightarrow$ looking for s_i-t_i path p s.t.:

$$z_i + \sum_{a \in p} y_a > c_p \Leftrightarrow z_i > \sum_{a \in p} c(a) - y_a =: \sum_{a \in p} c'(a) = c'_p$$

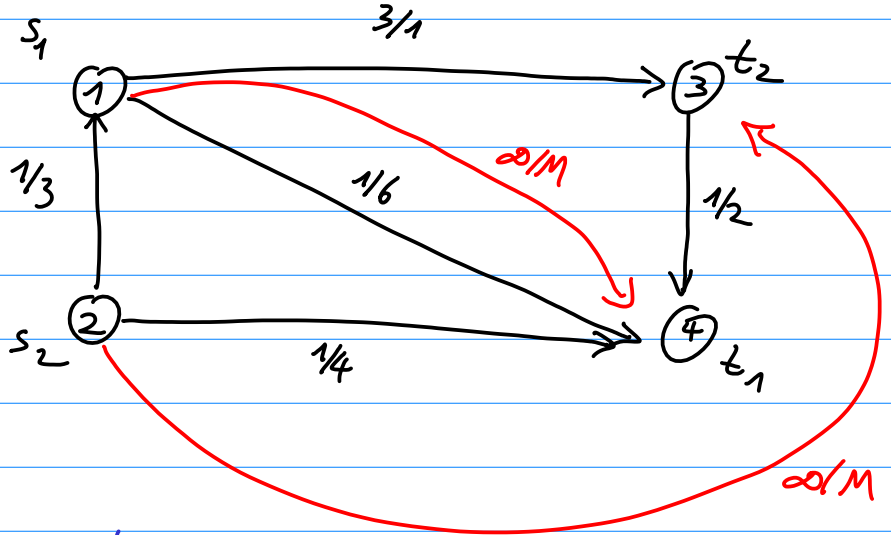
find s_i-t_i path in G with edge cost $< z_i$
$\rightarrow$ can be done by min s_i-t_i path computation

$c' \geq 0$ because $c(a) \geq 0$ and $y_a \leq 0$

$\rightarrow$ Dijkstra algorithm can be used

Procedure:
1. set up LP with paths such that it contains a solution — add s_i-t_i arcs with $c(a) = M$, $u(a) = \infty$
2. solve LP and get dual sol. z_i, y_a
3. for $i = 1$ to k
   - find shortest s_i-t_i path P in G with $c'_a = c_a - y_a$
   - add path variable to LP if $c'_P < z_i$
4. goto 2 until no add. paths were found

Example:



$d_1 = d_2 = 2$

$u(a) / c(a)$

choice for M: big enough $M = \sum_{a \in A} c(a) \cdot \sum_{i=1}^{k} d_i$

Programming exercise:

- read file format network/demands
- build graph
- build primal path based LP
- solve LP via CPLEX and get dual sol.
- search paths with length $< z_i$ ⇝ Dijkstra...
- repeat until no further path found

path calculation:

- Bellman: $O(n^4)$ / $O(n^3 \log n)$
- Floydwarshall: $O(n^3)$
- Dijkstra: $O(n^2)$ · k
  - Listen: $O(m \log n) \cdot k$
  - Fibb. Heap: $O((m+n) \log n) \cdot k$
  - AVL tree

CPLEX:

```java
package examples;

import ilog.concert.*;
import ilog.cplex.*;

/**
 * Example of the usage of the CPLEX API with Java.
 * The code shows how to build an LP for a multi commodity
 * flow problem with four given paths.
 *
 * @author Torsten Gellert
 */

public class SolvingLpWithCplex{

    /** Contant for the cost of artifical arcs for this particular ex
    public static final double M = 1000.0;
    /** Hide constructor, since this class is an simple example. */
    private SolvingLpWithCplex() {
    }
```

```java
/**
* Creates a path based LP formulation of the multi commodity flow problem
* for the instance shown in the exercise session.
* Solves and prints all variables and additional available information.
*
* @param  args The argument list . Not used by the method.
*/

public static void main( String[] args ) {
    try {
        IloCplex cplex = new IloCplex();

        IloNumVar[][] var = new IloNumVar[1][];
        IloRange[][]  rng = new IloRange[1][];

        populateByColumn(cplex, var, rng);

        if ( cplex.solve() ) {
            double[] x     = cplex.getValues(var[0]);
            double[] dj    = cplex.getReducedCosts(var[0]);
            double[] pi    = cplex.getDuals(rng[0]);
            double[] slack = cplex.getSlacks(rng[0]);

            cplex.output().println("Solution status = " + cplex.getStatus());
            cplex.output().println("Solution value  = " + cplex.getObjValue());

            int nvars = x.length;
            for (int j = 0; j < nvars; ++j) {
                cplex.output().println(var[0][j].getName() +
                                       " = " + x[j] +
                                       "\tReduced cost = " + dj[j]);
            }

            int ncons = slack.length;
            for (int i = 0; i < ncons; ++i) {
                cplex.output().println(rng[0][i].getName() +
                                       ":\tSlack = " + slack[i] +
                                       "\tPi = " + pi[i]);
            }
        }
        cplex.end();
    } catch (IloException exc) {
        System.err.println("Concert exception '" + exc + "' caught");
    }
}

/**
* Builds all variables and constraints(ranges). These are
* stored in a arrays and these array have to be placed in the
* first position of the given two dimentional arrays.
*
* @param model The Cplex model for handling with LPs.
* @param var   The array to store all created variables.
* @param rng   The array to store all ranges/constraints.
* @throws IloException if there occurs an error while build the LP model.
*/
```

```
    static void populateByColumn(IloMPModeler model,
                                 IloNumVar[][] var,
                                 IloRange[][] rng) throws IloException {
        IloObjective obj = model.addMinimize();

        //create all constraints
        rng[0] = new IloRange[7];
        //rhs for arcs
        rng[0][0] = model.addRange(-Double.MAX_VALUE, 3.0, "arc_1_3");
        rng[0][1] = model.addRange(-Double.MAX_VALUE, 1.0, "arc_1_4");
        rng[0][2] = model.addRange(-Double.MAX_VALUE, 1.0, "arc_2_1");
        rng[0][3] = model.addRange(-Double.MAX_VALUE, 1.0, "arc_2_4");
        rng[0][4] = model.addRange(-Double.MAX_VALUE, 1.0, "arc_3_4");
        //rhs for commodities
        rng[0][5] = model.addRange(2,Double.MAX_VALUE, "commodity_1");
        rng[0][6] = model.addRange(2,Double.MAX_VALUE, "commodity_2");


        //add variables
        IloRange c1 = rng[0][5];
        IloRange c2 = rng[0][6];

        IloRange arc_1_3 = rng[0][0];
        IloRange arc_1_4 = rng[0][0];
        IloRange arc_2_1 = rng[0][0];
        IloRange arc_2_4 = rng[0][0];
        IloRange arc_3_4 = rng[0][0];

        var[0] = new IloNumVar[4];
        var[0][0] = model.numVar(model.column(obj,  M).and(
                                 model.column(c1,  1.0)),
                                 0.0, Double.MAX_VALUE, "p_1_1");
        var[0][1] = model.numVar(model.column(obj,  M).and(
                                model.column(c2,  1.0)),
                                0.0, Double.MAX_VALUE, "p_2_1");
        var[0][2] = model.numVar(model.column(obj,  4).and(
                                 model.column(arc_1_3, 1.0).and(
                                 model.column(arc_3_4,  1.0).and(
                                 model.column(c1,  1.0)))),
                                 0.0, Double.MAX_VALUE, "p_1_2");
        var[0][3] = model.numVar(model.column(obj,  4.0).and(
                                 model.column(arc_2_1,   1.0).and(
                                 model.column(arc_1_3,   1.0).and(
                                 model.column(c2,   1.0)))),
                                 0.0, Double.MAX_VALUE, "p_2_2");
    }
}
```

```
/*
Output:

LP Presolve eliminated 4 rows and 0 columns.
Reduced LP has 3 rows, 4 columns, and 6 nonzeros.
Presolve time =    0.00 sec.

Iteration log . . .
Iteration:     1   Dual objective     =             8.000000
Solution status = Optimal
Solution value  = 1012.0
p_1_1 = 1.0     Reduced cost = 0.0
p_2_1 = 0.0     Reduced cost = 0.0
p_1_2 = 1.0     Reduced cost = 0.0
p_2_2 = 2.0     Reduced cost = 0.0
arc_1_3:        Slack = 0.0     Pi = -996.0
arc_1_4:        Slack = 1.0     Pi = 0.0
arc_2_1:        Slack = 1.0     Pi = 0.0
arc_2_4:        Slack = 1.0     Pi = 0.0
arc_3_4:        Slack = 1.0     Pi = 0.0
commodity_1:    Slack = 0.0     Pi = 1000.0
commodity_2:    Slack = 0.0     Pi = 1000.0
*/
```

⇝ see web page for code and running instructions

Also: checkstyle compliant code!