Solving Warehouse Location Problems with CMPL and pyCMPL

Guest lecture at the

Ural State University of Railway Transport
Yekaterinburg
February 2014

Prof. Dr. Mike Steglich
Technical University of Applied Sciences Wildau

1 About CMPL

Specialized Optimization Software

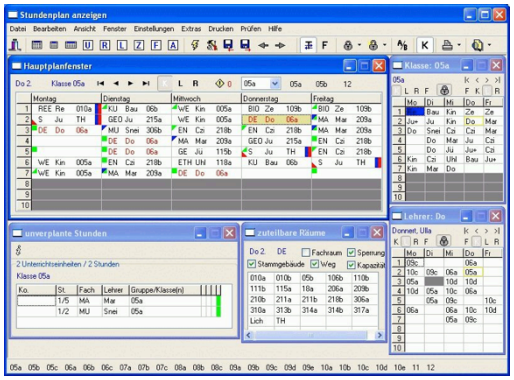
- Software intended to solve a particular decision problem.

Problem orientated Optimization Software

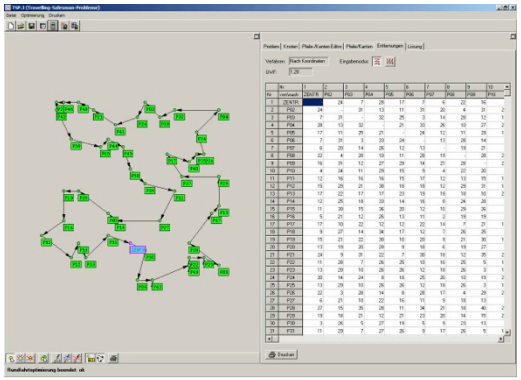
- Software that can be used for a class of decision problems.
- This kind of software works like a template that is to be customized by the decision maker.

Generalized Optimization Software

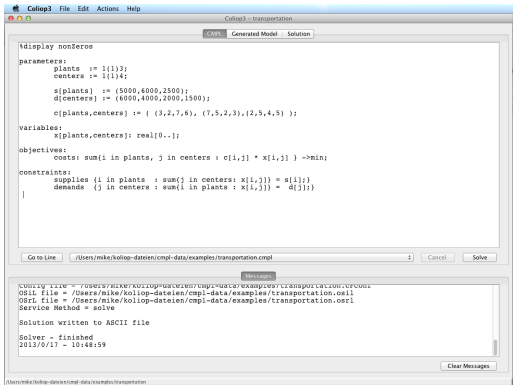
- Software that can be used for a wide range of decision problems.
- e.g.
  - Spread sheet software combined with solvers
  - Mathematical Programming Languages
  - APIs of commercial or Open Source solvers



[http://www.fuxschool.de/]



[TSP by Dieter Feige]



[http://coliop.org]

1 About CMPL

- **CMPL** (<Coliop|Coin> Mathematical Programming Language)
  - is a mathematical programming language and a system for mathematical programming and optimization of linear optimization problems.
  - CMPL executes CBC, GLPK, SCIP, Gurobi and CPLEX directly to solve the generated model instance. Since it is also possible to transform the mathematical problem into MPS, Free-MPS or OSiL files, alternative solvers can be used, as well.

- **pyCMPL** is the CMPL API for Python and an interactive shell.
  - The main idea of this API is:
    - to define sets and parameters within the user application,
    - to start and control the solving process and
    - to read the solution(s) in the application if the problem is feasible.
  - All variables, objective functions and constraints are defined in CMPL.
  - These functionalities can be used with a local CMPL installation or a CMPLServer.

- **CMPL** and **pyCMPL** are
  - COIN-OR projects initiated by the Technical University of Applied Sciences Wildau and the Institute for Operations Research and Business Management at the Martin Luther University Halle-Wittenberg
  - open source projects and available for most of the relevant operating systems (Windows, OS X and Linux)

2 The (single-source) Capacitated Warehouse Location Problem (CWLP)

⇨ Given a set of customers $C=\{1, \dots, m\}$ and a set of potential warehouses $W=\{1, \dots, n\}$ with

- Parameters
  - d_j - demand of costumer $j \in C$
  - s_i - capacities of warehouse $i \in W$
  - c_{ij} - total yearly transportation costs from warehouse i to customer j
  - F_i - fixed costs for establishing and running warehouse i
- Variables
  - $x_{ij} \in \{0,1\}$ - indicator variable whether factory j delivers costumer i (x_{ij}=1) or not (x_{ij}=0)
  - $y_i \in \{0,1\}$ - indicator variable whether warehouse j is established (y_i=1) or not (y_i=0)

⇨ The objective is to minimize the total transportation costs and the total fixed costs of establishing and running warehouses by deciding which warehouses are established and which customer is delivered by which warehouse.

$$\sum_{i \in W} \sum_{j \in C} c_{ij} \cdot x_{ij} + \sum_{i \in W} F_i \cdot y_i \rightarrow \min!$$

$$s.t.$$

$$\sum_{i \in W} x_{ij} = 1 \qquad ; j \in C$$

$$\sum_{j \in C} d_j \cdot x_{ij} \leq s_i \cdot y_i \qquad ; i \in W$$

$$x_{ij} \in \{0,1\} \qquad ; i \in W, j \in C$$

$$y_i \in \{0,1\} \qquad ; i \in W$$

3 Solving a text book CWLP example with CMPL

total yearly transportation costs if warehouse i delivers customer j

Ware-houses	Customers						Capacities	Fixed costs
	1	2	3	4	5	6		
1	19	75	42	11	52	63	40	30
2	46	109	41	8	34	38	80	40
3	47	65	29	14	17	18	100	70
4	71	64	14	24	22	33	60	40
Demands	25	50	15	10	30	20		

3 Solving a text book CWLP example with CMPL

W	C 1	2	3	4	5	6	s	F
1	19	75	42	11	52	63	40	30
2	46	109	41	8	34	38	80	40
3	47	65	29	14	17	18	100	70
4	71	64	14	24	22	33	60	40
d	25	50	15	10	30	20		

Problem.cmpl

```
%display nonZeros

parameters:
    C:= 1..6;
    W:= 1..4;

    c[W,C] := ( (19,75,42,11,52,63),
                (46,109,41,8,34,38),
                (47,65,29,14,17,18),
                (71,64,14,24,22,33) );

    d[C] := (25,50,15,10,30,20);
    s[W] := (40,80,100,60);
    F[W] := (30,40,70,40);

variables:
    x[W,C] : binary;
    y[W]   : binary;

objectives:
    costs:    sum {i in W, j in C: c[i,j] * x[i,j] } +
              sum {i in W: F[i] * y[i] }->min;

constraints:
    sos      {j in C: sum{i in W: x[i,j] } = 1; }
    capacity {i in W: sum {j in C: d[j] * x[i,j] } <= s[i] * y[i]; }
```

$$\sum_{i \in W} \sum_{j \in C} c_{ij} \cdot x_{ij} + \sum_{i \in W} F_i \cdot y_i \rightarrow \min!$$

$$s.t.$$

$$\sum_{i \in W} x_{ij} = 1 \qquad ; j \in C$$

$$\sum_{j \in C} d_j \cdot x_{ij} \leq s_i \cdot y_i \qquad ; i \in W$$

$$x_{ij} \in \{0,1\} \qquad ; i \in W, j \in C$$

$$y_i \in \{0,1\} \qquad ; i \in W$$

3 Solving a text book CWLP example with CMPL

```
cmpl problem.cmpl
```

```
-----------------------------------------------------------------------------------------------------
Problem              Problem1.cmpl
Nr. of variables     28
Nr. of constraints   10
Objective name       costs
Solver name          CBC
Display variables    nonzero variables (all)
Display constraints  nonzero constraints (all)
-----------------------------------------------------------------------------------------------------

Objective status     optimal
Objective value      284 (min!)

Variables
Name                 Type           Activity        Lower bound        Upper bound           Marginal
-----------------------------------------------------------------------------------------------------
x[1,1]                  B                  1                  0                  1                  -
x[1,4]                  B                  1                  0                  1                  -
x[3,2]                  B                  1                  0                  1                  -
x[3,5]                  B                  1                  0                  1                  -
x[3,6]                  B                  1                  0                  1                  -
x[4,3]                  B                  1                  0                  1                  -
y[1]                    B                  1                  0                  1                  -
y[3]                    B                  1                  0                  1                  -
y[4]                    B                  1                  0                  1                  -
-----------------------------------------------------------------------------------------------------

Constraints
Name                 Type           Activity        Lower bound        Upper bound           Marginal
-----------------------------------------------------------------------------------------------------
sos[1]                  E                  1                  1                  1                  -
sos[2]                  E                  1                  1                  1                  -
sos[3]                  E                  1                  1                  1                  -
sos[4]                  E                  1                  1                  1                  -
sos[5]                  E                  1                  1                  1                  -
sos[6]                  E                  1                  1                  1                  -
capacity[1]             L                 -5          -Infinity                  0                  -
capacity[4]             L                -45          -Infinity                  0                  -
-----------------------------------------------------------------------------------------------------
```

4 Solving a text book CWLP example with pyCMPL

Problem1a.cmpl

```
%data : C set, W set, c[W,C], d[C], s[W], F[W]

variables:
    x[W,C]  : binary;
    y[W]    : binary;


objectives:
    costs:  sum {i in W, j in C: c[i,j] * x[i,j] } +
            sum {i in W: F[i] * y[i] }-> min;

constraints:

    sos       {j in C: sum{i in W: x[i,j] } = 1; }
    capacity  {i in W: sum {j in C: d[j] * x[i,j] } <= s[i] * y[i]; }
```

4 Solving a text book CWLP example with pyCMPL

W	C 1	2	3	4	5	6	s	F
1	19	75	42	11	52	63	40	30
2	46	109	41	8	34	38	80	40
3	47	65	29	14	17	18	100	70
4	71	64	14	24	22	33	60	40
d	25	50	15	10	30	20		

Problem1.py

```
#!/usr/bin/python
from pyCmpl import *

try:
    m = Cmpl("Problem1a.cmpl")

    cities=CmplSet("C")
    cities.setValues(1,6)

    centers=CmplSet("W")
    centers.setValues(1,4)

    demandData=[25,50,15,10,30,20]
    demand=CmplParameter("d", cities)
    demand.setValues(demandData)

    costMat= [ [19,75,42,11,52,63],[46,109,41,8,34,38], \
               [47,65,29,14,17,18],[71,64,14,24,22,33] ];
    costs = CmplParameter("c", centers, cities)
    costs.setValues(costMat)

    capData=[40,80,100,60]
    cap= CmplParameter("s",centers)
    cap.setValues(capData)

    fcData=[30,40,70,40]
    fc= CmplParameter("F",centers)
    fc.setValues(fcData)

    m.setSets(centers, cities)
    m.setParameters(costs, cap, demand, fc )

    m.setOption("%display nonZeros")
    m.setOption("%display con capa*”)
```

4 Solving a text book CWLP example with pyCMPL

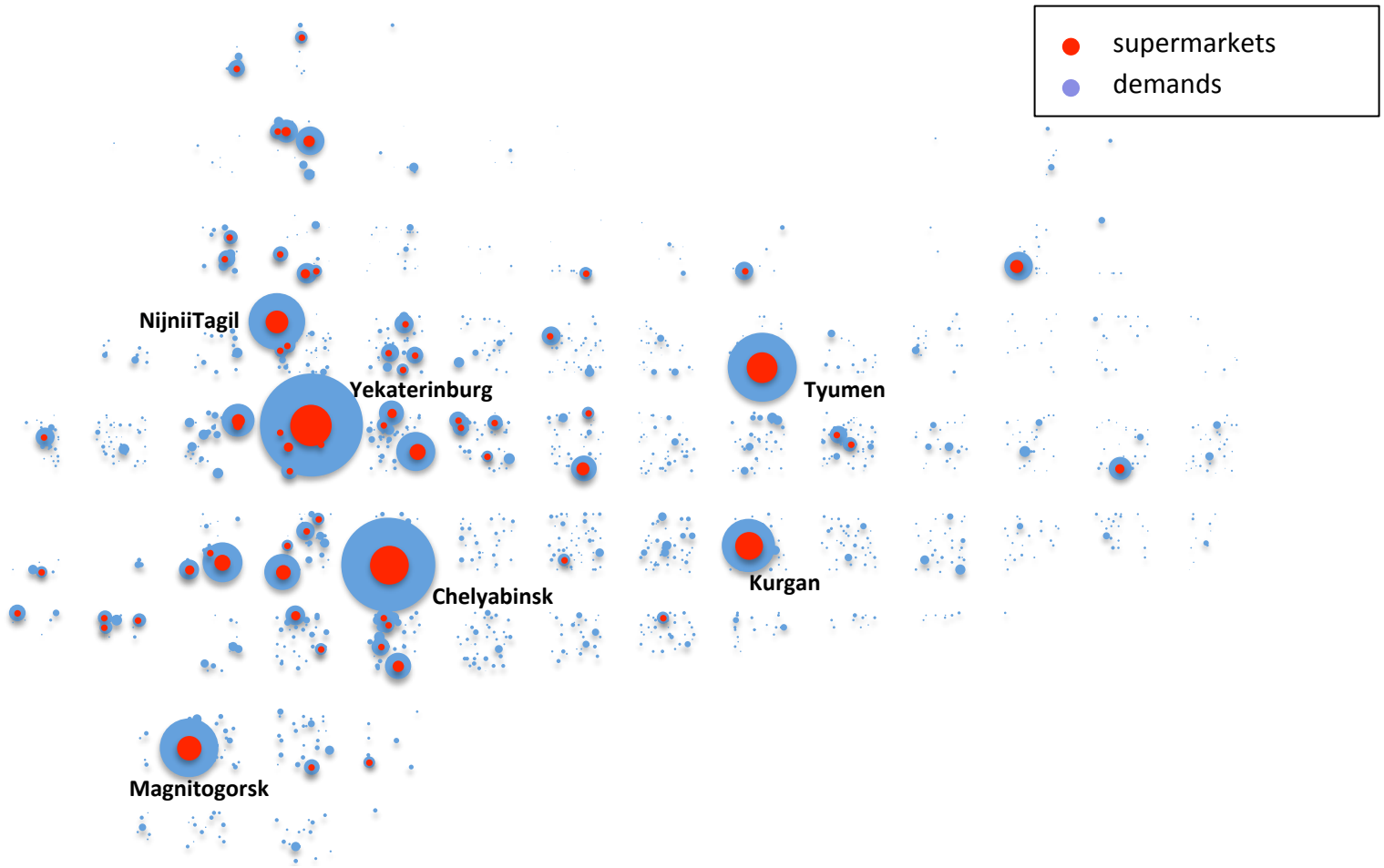
```
    m.solve()

    if m.solverStatus == SOLVER_OK:

       print "Total costs: ", m.solution.value
       for v in m.solution.variables:
           print v.idx,v.name, v.type, v.activity,v.lowerBound, v.upperBound

       for c in m.solution.constraints:
           print c.idx, c.name, c.type, c.activity,c.lowerBound, c.upperBound

    else:
       print "solver failed: "+ m.solver + " " + m.solverMessage

except CmplException, e:
    print e.msg
```

```
pyCmpl problem1.py
```

```
Total costs:  284.0
0 x[1,1] B 1 0.0 1.0
3 x[1,4] B 1 0.0 1.0
13 x[3,2] B 1 0.0 1.0
16 x[3,5] B 1 0.0 1.0
17 x[3,6] B 1 0.0 1.0
20 x[4,3] B 1 0.0 1.0
24 y[1] B 1 0.0 1.0
26 y[3] B 1 0.0 1.0
27 y[4] B 1 0.0 1.0
6 capacity[1] L -5.0 -inf 0.0
9 capacity[4] L -45.0 -inf 0.0
6 capacity[1] L -5.0 -inf 0.0
9 capacity[4] L -45.0 -inf 0.0
```

5 A CWLP model for *Rouble* – A fictitious retail chain company

- ***Rouble*** is a fictitious retail chain company which operates 245 supermarkets in the regions Yekaterinburg, Chelyabinsk, Kurgan and Tyumen.
- The head quarter is based in Yekaterinburg.



5 A CWLP model for *Rouble* – A fictitious retail chain company

⇨ **Assumptions and definitions**

- **General**
  - The warehouse location decision is a strategic planning problem over a timeframe of T years.
  - Therefore all impacts have to be discounted with a given interest rate $wacc$. This can be done as a net present value of ordinary annuities if constant yearly cost figures can be assumed.

- **Transportation costs**
  - The supermarkets are delivered by the logistic centres via trucks with a given weight of load measured in tons.
  - The yearly demand of a city is given by the sum of the necessary weekly deliveries by trucks d_j of all supermarkets located in the city multiplied by 52 weeks.
  - The distance dst_j of a tour from a centre i to a city j is equal to the Euclidian Distance multiplied by a diversion factor df

$$dst_{ij} = \sqrt{\left(xCoord_i - xCoord_j\right)^2 + \left(yCoord_i - yCoord_j\right)^2} \cdot df$$

  - With the given variable transportation costs per kilometre tc, the discounted yearly transportation costs between a centre i and a city j can be calculated as follows:

$$c_{ij} \cdot x_{ij} = tc \cdot 2 \cdot dst_{ij} \cdot 52 \cdot d_j \cdot af \cdot x_{ij} \;\; ; \;\; af = \frac{q^T - 1}{(q-1) \cdot q^T} \; ; \; q = (1 + wacc)$$

annuity present value factor

5 A CWLP model for *Rouble* – A fictitious retail chain company

⇨ **Assumptions and definitions**

- **Logistic centre costs**
  - *Rouble* plans to implement several logistic centres of one type with identical capital expenditures $capex$ to establish one logistic centre with identical yearly operating costs f and an identical capacity s that is defined by the maximal number of weekly deliveries by trucks.
  - The capacity of a logistic centre can be expanded in steps of s. Therefore the variables y_i are changed from a binary to an integer variables.
  - The discounted logistic centre costs of one centre j can be calculated as sum of the capital expenditures and the discounted operating costs:

$$F_i \cdot y_i = \left(capex + f \cdot af\right) \cdot y_i \;\; ; \; y_i \in \{0,1,\ldots\}$$

- **Objective function**

$$\sum_{i \in W} \sum_{j \in C} c_{ij} \cdot x_{ij} + \sum_{i \in W} F_i \cdot y_i \rightarrow \min!$$

$$\sum_{i \in W} \sum_{j \in C} tc \cdot 2 \cdot dst_{ij} \cdot 52 \cdot d_j \cdot af \cdot x_{ij} + \sum_{i \in W} \left(capex + f \cdot af\right) \cdot y_i \rightarrow \min!$$

5 A CWLP model for *Rouble* – A fictitious retail chain company

⇨ **Assumptions and definitions**

- **Constraints**
  - It is necessary to constrain the distance per tour $mDst$ due to the long distances between potential logistic centres and the cities in the regions Yekaterinburg, Chelyabinsk, Kurgan and Tyumen.

$$dst_{ij} \cdot x_{ij} \leq mDst \qquad ; i \in W, j \in C$$

  - All other constraints are not changed.

- The entire **Capacitated Warehouse Location Problem** for *Rouble* can be formulated as follows:

$$\sum_{i \in W} \sum_{j \in C} c_{ij} \cdot x_{ij} + \sum_{i \in W} F_i \cdot y_i \rightarrow \min!$$

$$s.t.$$

$$\sum_{i \in W} x_{ij} = 1 \qquad ; j \in C$$

$$\sum_{j \in C} d_j \cdot x_{ij} \leq s \cdot y_i \qquad ; i \in W$$

$$dst_{ij} \cdot x_{ij} \leq mDst \qquad ; i \in W, j \in C$$

$$x_{ij} \in \{0,1\} \qquad ; i \in W, j \in C$$

$$y_i \in \{0,1,\ldots\} \qquad ; i \in W$$

6 Solving the *Rouble* CWLP with pyCMPL

```
%f <  180164000 >
%capex < 176000000>
%tc <35>
%af < 6.1445671 >
%mDst <350>
```

Rouble.cmpl

```
%data Rouble.cdat : f , capex, tc, af , mDst
%data : C set , W set, d[C], dst[W,C], s

variables:
    x[W,C]: binary;
    y[W]:   integer[0..];

objectives:
    costs:  sum {i in W, j in C: tc*2*dst[i,j]*52*d[j]*af*x[i,j]} +
            sum {i in W: (capex + f*af) * y[i] }-> min;

constraints:
    sos         {j in C: sum {i in W: x[i,j] } = 1; }
    capacity    {i in W: sum {j in C: d[j] * x[i,j] } <= s * y[i] ;}
    maxDist     {i in W, j in C: dst[i,j] * x[i,j]  <= mDst; }
```

$$\sum_{i \in W} \sum_{j \in C} c_{ij} \cdot x_{ij} + \sum_{i \in W} F_i \cdot y_i \rightarrow \min!$$

$$s.t.$$

$$\sum_{i \in W} x_{ij} = 1 \qquad ; j \in C$$

$$\sum_{j \in C} d_j \cdot x_{ij} \leq s \cdot y_i \qquad ; i \in W$$

$$dst_{ij} \cdot x_{ij} \leq mDst \qquad ; i \in W, j \in C$$

$$x_{ij} \in \{0,1\} \qquad ; i \in W, j \in C$$

$$y_i \in \{0,1,\ldots\} \qquad ; i \in W$$

6 Solving the *Rouble* CWLP with pyCMPL

Rouble.py

```
#!/usr/bin/python

from __future__ import division

from pyCmpl import *
import csv
import math
from Tkinter import *

try:
   centerNames = []
   centerXpos = []
   centerYpos = []
   cityNames = []
   cityXpos = []
   cityYpos = []
   cityDemand = []
   df=1.15

   centerData = csv.reader(open("Rouble-xy-centers-data.csv","rU"), delimiter=";")
   for centerName,centerX,centerY in centerData:
      centerNames.append(centerName)
      centerXpos.append(int(centerX))
      centerYpos.append(int(centerY))

   cityData = csv.reader(open("Rouble-xy-cities-data.csv","rU"), delimiter=";")
   for cityName,cityX,cityY, demand in cityData:
      cityNames.append(cityName)
      cityXpos.append(int(cityX))
      cityYpos.append(int(cityY))
      cityDemand.append(int(demand))
```

```
Alapaevsk;6141;5749
Aramil;6050;5640
Artemovskii;6153;5718
Asbest;6127;5659
Berezovskii;6047;5653
Bogdanovich;6202;5645
VerhnyayaSalda;6032;5801
Yekaterinburg;6037;5648
...
```

6 Solving the *Rouble* CWLP with pyCMPL

```
i=0
for center in centerNames:
   j=0
   for city in cityNames:
       if center==city:
           distMatrix[i][j]=5
       else:
           distMatrix[i][j] = math.sqrt( (centerXpos[i]-cityXpos[j])**2 + \
                              (centerYpos[i]-cityYpos[j])**2 ) * df
       j+=1
   i+=1


m = Cmpl("Rouble.cmpl")

cities=CmplSet("C")
cities.setValues(cityNames)

weeklyDeliveries=CmplParameter("d", cities)
weeklyDeliveries.setValues(cityDemand)

centers=CmplSet("W")
centers.setValues(centerNames)

distance = CmplParameter("dst", centers, cities)
distance.setValues(distMatrix)

cap= CmplParameter("s")
cap.setValues(100)

m.setSets(centers, cities)
m.setParameters(weeklyDeliveries, distance, cap)
```

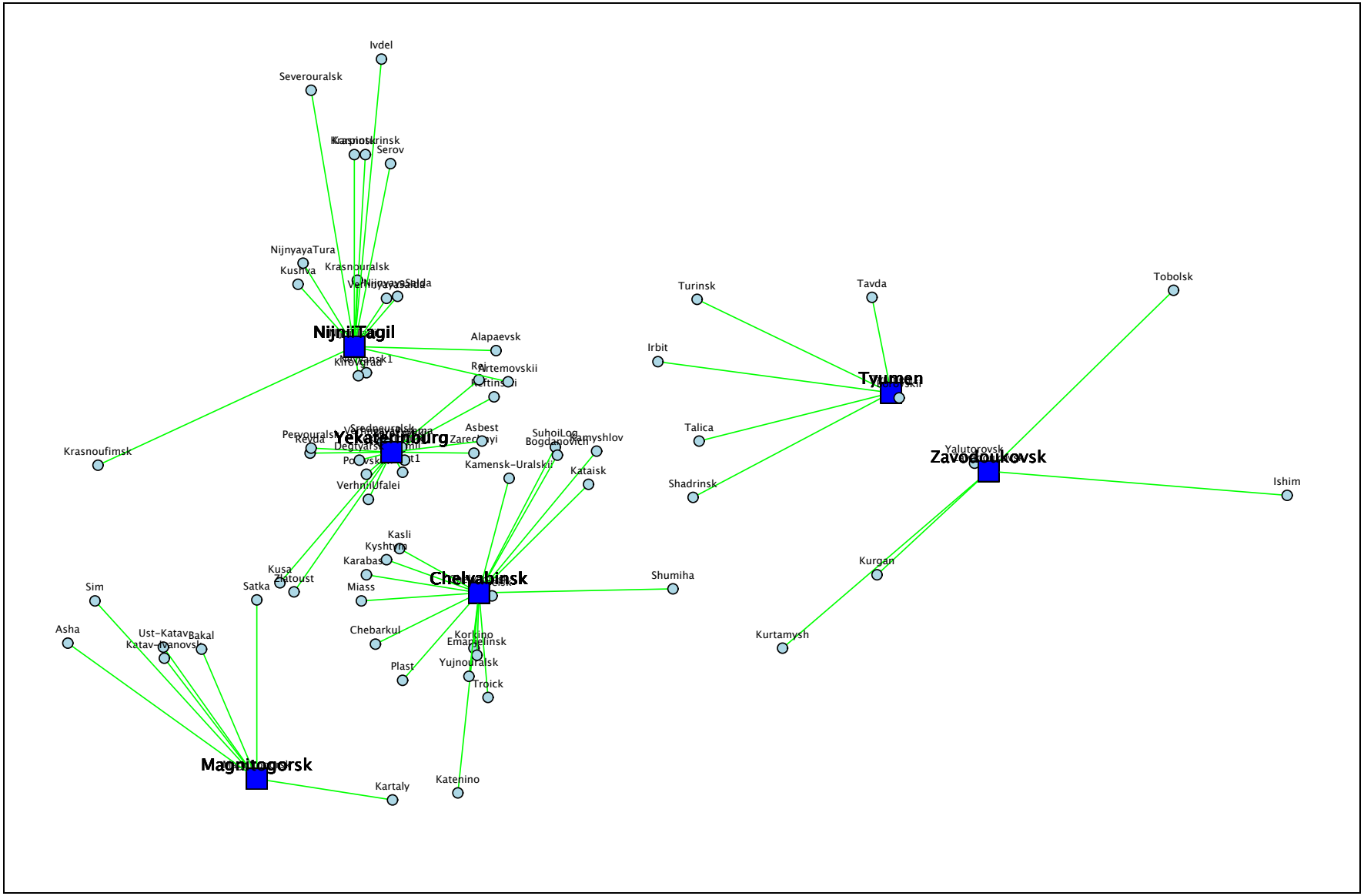
6 Solving the *Rouble* CWLP with pyCMPL

```
    m.setOption("%arg -solver cplex")
    m.setOption("%opt cplex mip/tolerances/mipgap 0.05")

    m.solve()

    if m.solverStatus == SOLVER_OK:
       m.varByName()
       m.conByName()

       print "Total costs: " , m.solution.value
       print "Logistik centers"

       for i in m.y:
           if m.y[i].activity>0:
               print "    %-15s - %3i weekly tours from maximally %3i to:" % \
                     (i , m.y[i].activity * cap.value + m.capacity[i].activity , \
                     m.y[i].activity * cap.value )

               for j in m.x:
                   if m.x[j].activity>0:
                       if i==j[0]:
                           idx=cityNames.index(j[1])
                           print "        %-15s %3i " % (j[1], cityDemand[idx]   )

       drawNetwork()

    else:
       print "solver failed: "+ m.solver + " " + m.solverMessage

except CmplException, e:
    print e.msg
```

6 Solving the *Rouble* CWLP with pyCMPL

```
Total costs:  11515900000.0
Logistik centers
    Chelyabinsk     - 194 weekly tours from maximally 200 to:
        Katenino           2
        Troick             9
        Kopeisk            6
        Emanjelinsk        3
        Kyshtym            4
        Chebarkul          4
        Korkino            4
        Kamensk-Uralskii  18
        Kamyshlov          3
        Yujnouralsk        4
        Karabash           2
        Miass             15
        Kasli              2
        Shumiha            3
        Chelyabinsk      105
        Bogdanovich        3
        Kataisk            2
        SuhoiLog           3
        Plast              2
    Yekaterinburg   - 200 weekly tours from maximally 200 to:
        Asbest             6
        Rej                4
        Sysert1            2
        Berezovskii        4
        Zarechnyi          3
        Kusa               2

...
```

6 Solving the *Rouble* CWLP with pyCMPL



Literature

Feige D./Klaus P.: Modellbasierte Entscheidungsunterstützung in der Logistik, Hamburg 2008.

Steglich, M/Schleiff, Th.: CMPL - <Coliop|Coin> Mathematical Programming Language – Manual, Version 1.8 , 2013.

For more information:

CMPL - http://www.coliop.org

Thank you for your attention.

If you have any questions, please feel free to ask.

Спасибо за ваше внимание