Formulations for the Traveling Salesman Problem

Drive 1 car via *n* cities, minimising travel.
History: Dantzig, Fulkerson, Johnson paper: B&B, cutting planes.
Cook, Karp, Levin: complexity theory.

Some formulations for the TSP:
Assignment.
Assignment + tight subtours (DFJ).
Assignment + loose subtours (MTZ).
Symmetric.
Flow (Svestka).
Steps (Dantzig).

The Travelling Salesman Contest

Here are the distances between 9 cities. Find the shortest tour.

	1	2	3	4	5	6	7	8	9
1		41.0	53.0	62.9	106.3	86.2	78.2	74.5	47.7
2			25.8	17.1	69.6	60.8	48.2	46.6	36.4
3				31.9	67.3	72.3	57.9	58.6	59.3
4					53.9	44.2	31.2	29.9	28.8
5						37.0	32.5	37.6	67.0
6							14.5	14.3	39.5
7								5.1	35.0
8									30.2

This is all the data you need to solve the problem exactly.
The problem is easy to describe, but *very* hard to solve exactly.

Early history of the TSP

1930s-50s: Publicised in the math & OR community by Merrill Flood.

1948: Popularised at RAND Corp., a U.S. military thinktank, which helped developed OR.

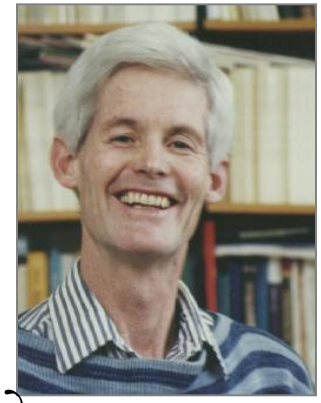
1954: "Solution of a large-scale traveling-salesman problem," Dantzig, Fulkerson & Johnson, *J. of Ops Research of America*. At RAND, they solved a 49-city TSP to optimality.

DF&J thought a nearly optimal tour could be improved, and then optimality could be guaranteed, by adding just a few cuts.

DF&J were the first to use cuts and B&B for integer programming.

Why is the TSP important?[4]

1971 Cook, 1972 Karp, 1973 Levin: *computational complexity*.
How hard is a problem to solve, as a function of the input data?
If Problem 1 *converts* easily to Problem 2 & vice versa,
then Problem 1 is as easy or hard as Problem 2.


Stephen Cook

Some problems are
easy: shortest path, min spanning tree, assignment.
impossible: given a computer program & input, will it stop?
hard: TSP – because no "good" algorithms have been found.

Many hard problems, such as job shop scheduling,
can be converted algebraically to & from the TSP.
A good TSP algorithm will be good for other hard problems.

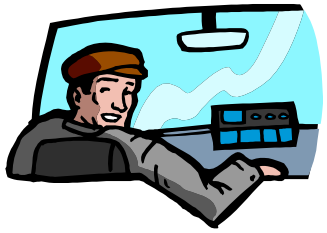
The TSP is easy to state, takes no math background to understand,
and no great talent to find feasible solutions.
It's fun, and invites recreational problem solvers.

2 general strategies for the TSP

Given that the TSP is hard, what can we do?

1) Approximate. Use heuristics!
   Nearest neighbour, nearest merger,
   Love & Norback (angles), Lin-Kernighan (3-opt),
   min spanning tree + shortcuts (Christofides),
   and many others.

2) Spend a lot of time at it. Enumeration!
   Cutting planes,
   branch & bound,
   dynamic programming.



We'll do (2) here.
Next: formulations for the TSP.

Assignment problem - easy, naturally integer.

Indices: i = teacher, j = course.

Parameters: c_{ij} = value if teacher i is assigned to course j.

Variables: x_{ij} = 1 if teacher i is assigned to course j, else 0.

Model AP: 1) Max $\sum_i \sum_j c_{ij} x_{ij}$ subject to

2) $\sum_j x_{ij} = 1$, for all i,

3) $\sum_i x_{ij} = 1$, for all j,

4) $x_{ij} \in \{0,1\}$, for all i,j.

Explanation: 1) Maximise value of assignments.

2) Assign each teacher i to one course.

3) Assign each course j to one teacher.

Almost the TSP. Is AP a possible formulation for the TSP?

Indices: i, j = city.

Parameter: c_{ij} = cost to go from city i to city j.

Variables: x_{ij} = 1 if we drive from city i to city j, else 0.

The AP formulation won't work.

J32 = =WB(SUM(J23:J31),"=",1)

	A	B	C	D	E	F	G	H	I	J	K	L
1						To					Formula	
2		1	2	3	4	5	6	7	8	9		
13	km 1	0	56.6	51.6	53.1	92.3	62.4	82.5	101.9	47.4		
14	2	56.6	0	12.2	3.6	67.2	62.3	25.9	56.6	24.4		
15	3	51.6	12.2	0	10	78.4	69.7	34.7	68.7	31.6		
16	From 4	53.1	3.6	10	0	68.4	61.5	29.5	59.5	23.3		
17	5	92.3	67.2	78.4	68.4	0	37	66.4	35.7	51.1		
18	6	62.4	62.3	69.7	61.5	37	0	75.2	64.1	38.2		
19	7	82.5	25.9	34.7	29.5	66.4	75.2	0	42.4	43		
20	8	101.9	56.6	68.7	59.5	35.7	64.1	42.4	0	54.5		
21	9	47.4	24.4	31.6	23.3	51.1	38.2	43	54.5	0		
22											Flow out = 1	
23	Go? 1	1	0	0	0	0	0	0	0	0	0	
24	2	0	1	0	0	0	0	0	0	0	0	
25	3	0	0	1	0	0	0	0	0	0	0	
26	4	0	0	0	1	0	0	0	0	0	0	
27	5	0	0	0	0	1	0	0	0	0	0	
28	6	0	0	0	0	0	1	0	0	0	0	
29	7	0	0	0	0	0	0	1	0	0	0	
30	8	0	0	0	0	0	0	0	1	0	0	
31	9	0	0	0	0	0	0	0	0	1	0	
32	Flow in = 1	0	0									
33												
34												
35	Total distance	0										

What'sBest formulation. What's wrong?
At least we have a lower bound of 0.

So just leave out variables x_{ii}.

J32 = =WB(SUM(J23:J31),"=",1)

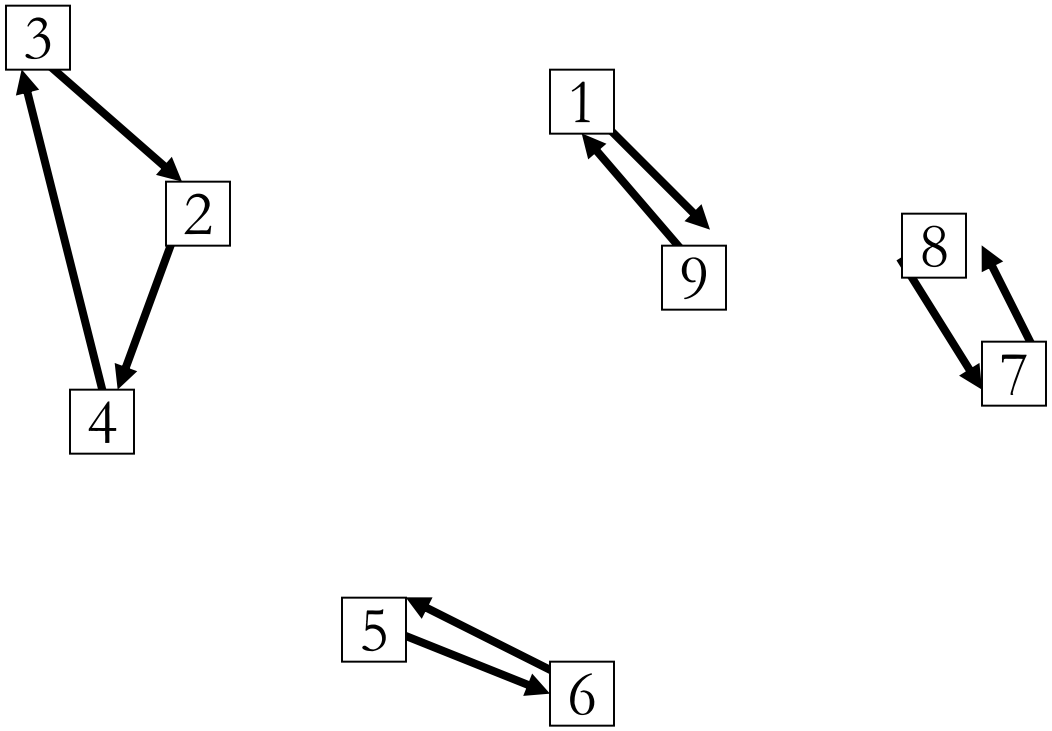
	A	B	C	D	E	F	G	H	I	J	K	L
1						To					Formula	
2		1	2	3	4	5	6	7	8	9		
13	km 1	0	56.6	51.6	53.1	92.3	62.4	82.5	101.9	47.4		
14	2	56.6	0	12.2	3.6	67.2						
15	3	51.6	12.2	0	10	78.4						
16	From 4	53.1	3.6	10	0	68.4						
17	5	92.3	67.2	78.4	68.4	0						
18	6	62.4	62.3	69.7	61.5	37						
19	7	82.5	25.9	34.7	29.5	66.4						
20	8	101.9	56.6	68.7	59.5	35.7						
21	9	47.4	24.4	31.6	23.3	51.1						
22											Flow out = 1	
23	Go? 1	0	0	0	0	0	0	0	0	1	0	
24	2	0	0	0	1	0	0	0	0	0	0	
25	3	0	1	0	0	0	0	0	0	0	0	
26	4	0	0	1	0	0	0	0	0	0	0	
27	5	0	0	0	0	0	1	0	0	0	0	
28	6	0	0	0	0	1	0	0	0	0	0	
29	7	0	0	0	0	0	0	0	1	0	0	
30	8	0	0	0	0	0	0	1	0	0	0	
31	9	1	0	0	0	0	0	0	0	0	0	
32	Flow in = 1	0	0	0	0	0	0	0	0	0		
33												
34												
35	Total distance	279.4										

I reset the variables on the diagonal so they are fixed to 0.

Looks good. Or does it?

A better lower bound of 279.4.

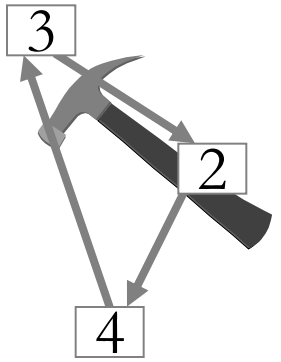
We have subtours.



Oops. How do we get rid of these?

1. Can you write code to print the tour from an optimal solution?
2. Can you write code to print a subtour of an infeasible solution?
3. Do we need an integer solution to find subtours?

Ways to break subtours: 2^n subtour constraints.



The Dantzig, Fulkerson & Johnson (DFJ) model.

Indices, parameters, & decision variables as before.

Minimise total cost: $\min \sum_i \sum_j c_{ij} x_{ij}$,

Enter each city once: $\sum_i x_{ij} = 1$ for all j.

Leave each city once: $\sum_j x_{ij} = 1$ for all i.

Subtour breaking constraints: $\sum_{i,j \in S} x_{ij} \leq |S| - 1$, for every subset S.

Binary integrality: $x_{ij} \in \{0, 1\}$ for all i, j.

For the subtour shown, add: $x_{3,2} + x_{2,4} + x_{4,3} \leq 2$. What are the others?

After solving again with the new constraints, more subtours appear.

For a large TSP, we may need many subtour breaking constraints.

In the worst case, we may need 2^n subtour breaking constraints.

Next week, we will see a way to generate these constraints.

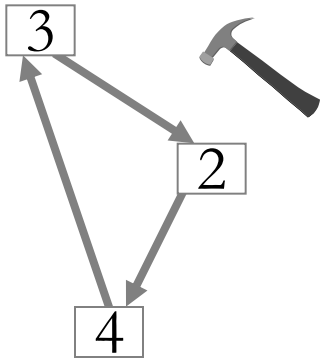
The solution becomes fractional, so we also need to do B&B.

However, every solution gives a lower bound on the optimum.

Ways to break subtours: MTZ model

Indices & parameters as before.

Variables: $x_{ij} = 1$ if we drive from city i to city j, else 0.
u_i = number of cities visited at city i.



Minimise total cost: $\min \sum_i \sum_j c_{ij} x_{ij}$,

Enter each city once: $\sum_i x_{ij} = 1$ for all j.

Leave each city once: $\sum_j x_{ij} = 1$ for all i.

Subtour breaking: $u_i + 1 \le u_j + n(1 - x_{ij})$, for $i = 2, \ldots, n$, $i \ne j$, $j = 2, \ldots, n$,

$x_{ij} \in \{0, 1\}$ for all i, j; $u_i \ge 0$ for all i.

Fewer constraints, but harder to solve! The LP relaxation is not as tight.

- Okay for small problems, but is bad for large ones.
- Related variations are a bit tighter.

Ref: C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulations and traveling salesman problems," *J. ACM*, 7 (1960), pp. 326–329.

We'll use this data for our AMPL models.

```
param N := 12;

param C := [*,*] :
1  2    3    4    5    6    7    8    9    10   11   12:=
1  .    26   40   34   35   48   30   23   48   17   20
29
2  26   .    64   61   59   78   54   47   50   29   28
33
3  40   64   .    3    5    8    10   15   24   30   33
39
4  34   61   3    .    3    10   8    13   26   28   30
35
5  35   59   5    3    .    13   5    10   29   25   28
33
6  48   78   8    10   13   .    18   23   16   38   39
33
7  30   54   10   8    5    18   .    5    34   20   23
28
8  23   47   15   13   10   23   5    .    38   15   18
```

MTZ formulation in AMPL, ~6 secs.[13]

```
param N integer > 2; # Number of nodes
set Nodes ordered := {1..N};
set Arcs := {i in Nodes, j in Nodes: i <> j};
param C{(i,j) in Arcs};
var x {(i,j) in Arcs} binary;
var u {Nodes} >= 0;

minimize Tourlength: sum {(i,j) in Arcs} C[i,j]*x[i,j];
subject to Degree1 {i in Nodes}: sum{(i,j) in Arcs} x[i,j]=1;
subject to Degree2 {i in Nodes}: sum{(j,i) in Arcs} x[j,i]=1;

subject to NoSubtour1 {(i,j) in Arcs: i<>j and i>=2 and j>=2}:
u[i] - u[j] + N*x[i,j] <= N - 1;
subject to NoSubtour2 {i in Nodes: i >= 2}:
u[i] <= N - 1 - (N - 2)*x[1,i];
subject to NoSubtour3 {i in Nodes: i >= 2}:
u[i] >= 1  +  (N - 2)*x[i,1];
```

AMPL solution

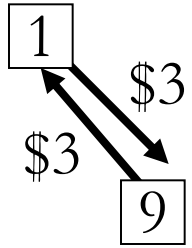
```
u[1]    u[4] u[12]u[6] u[2] u[11]u[9] u[3] u[7] u[10]u[5]
u[8]
0  1    2    3    4    5    6    7    8    9    10   11
1,4     4,12 12,6 6,2  2,11 11,9 9,3  3,7  7,10 10,5 5,8
```

```
min 500 x1,1 + 70.9 x1,2 + 41.6 x1,3 + 29.1 x1,4
  + 17.1 x1,5 + 56.1 x1,6 + 55.6 x1,7 + 7.3 x1,8
  ...
  + 104 x12,10 + 39.4 x12,11 + 500 x12,12
subject to
3] x2,1 + x3,1 + x4,1 + x5,1 + x6,1 + x7,1
   + x8,1 + x9,1 + x10,1 + x11,1 + x12,1 = 1
4] x1,2 + x1,3 + x1,4 + x1,5 + x1,6 + x1,7
   + x1,8 + x1,9 + x1,10 + x1,11 + x1,12 = 1
...
25] x1,12 + x2,12 + x3,12 + x4,12 + x5,12 + x6,12
    + x7,12 + x8,12 + x9,12 + x10,12 + x11,12 = 1
26] x12,1 + x12,2 + x12,3 + x12,4 + x12,5 + x12,6
    + x12,7 + x12,8 + x12,9 + x12,10 + x12,11 = 1
27] 12 x2,2 <= 11
28] 12 x2,3 + u2 - u3 <= 11
29] 12 x2,4 + u2 - u4 <= 11
30] 12 x2,5 + u2 - u5 <= 11
...
143] 12 x12,8 - u8 + u12 <= 11
144] 12 x12,9 - u9 + u12 <= 11
145] 12 x12,10 - u10 + u12 <= 11
146] 12 x12,11 - u11 + u12 <= 11
147] 12 x12,12 <= 11
end
inte x1,1
inte x1,2
...
```

If we go from city 2 to city 5, then u2 + 1= u5.

Parts of the MTZ model in Lindo

The symmetric TSP



Symmetric TSP: $c_{ij} = c_{ji}$.

Indices: i, j = city.

Parameter: c_{ij} = cost to go from city i to city j.

Variables: $x_{ij} = 1$ if we drive from city i to city j, else 0, defined only for $i<j$. Half as many variables as the asymmetric!

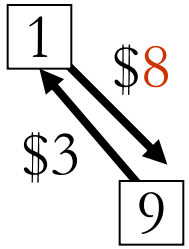
Minimise total cost: $\min \sum_i \sum_{j>i} c_{ij} x_{ij}$,

Enter each city once: $\sum_{j<i} x_{ji} + \sum_{j>i} x_{ij} = 2$ for all i.

Subtour breaking: $\sum_{i,j \in S} x_{ij} \leq |S| - 1$, for each subset S.

Binary integrality: $x_{ij} \in \{0, 1\}$ for all i, j.

The homework is a symmetric TSP.



The asymmetric TSP, $c_{ij} \neq c_{ji}$, is more realistic. Why?

How does the row 2 summation work?

Model:

1. Min $\sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{i,j} x_{i,j}$,
2. $\sum_{i=1}^{j-1} x_{i,j} + \sum_{i=j+1}^{n} x_{j,i} = 2$, for all j.
3. $\sum_{i,j \in S} x_{i,j} \leq |S| - 1$, for every subset S,
4. $x_{i,j} \in \{0,1\}$ for all i,and j: $j>i$.

The variables *into* city 5 are: x15, x25, x35, x45, x65, x75, x85, x95.
The variables *out of* city 5 are: x51, x52, x53, x54, x56, x57, x58, x59.

Since costs are symmetric, $c_{ij} = c_{ji}$, let's drop half the variables.

For x_{ij}, require $i < j$. Allow only the variables going out.

We need only variables x15, x25, x35, x45, x56, x57, x58, x59.

The meaning is not "Go in" or "come out", but "use this arc".

The summation makes sure that we cover only the variables we need.

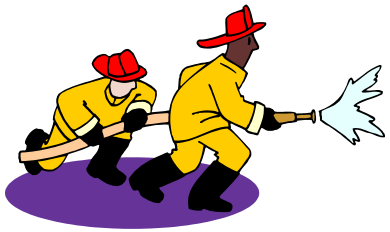
x15 + x25 + x35 + x45 + x56 + x57 + x58 + x59 = 2.

A flow with gain model of the TSP

Parameter: f = gain in flow from city i to city j.

Decision variables: $x_{ij} = 1$ if we drive from city i to city j, else 0.

y_{ij} = flow from city i to city j.



1. Min total cost: $\min \sum_i \sum_j c_{ij} x_{ij}$,
2. Arrive in each city: $\sum_{j \neq i} y_{ji} \geq 1$, for $i = 2,\ldots,n$,
3. Flow with gain of f: $\sum_{j \neq i} y_{ij} - \sum_{j \neq i} y_{ji} = f$, for $i = 1,\ldots,n$,

4. Only n positive vars: $\sum_i \sum_j x_{ij} \leq n$, cardinality constraint,
5. Force the y_{ij} vars: $y_{ij} \leq (1 + n \cdot f) x_{ij}$ for all i, j,
6. Nonnegative, binary: $y_{ij} \geq 0$ for all i, j, $x_{ij} \in \{0, 1\}$ for all i, j.

This formulation flows f "material" from city to city.

As f approaches 0, the solution approaches the optimum.

Hard to solve because of (4), the cardinality constraint.

Other rows can be added to tighten the model.

Ref: JA Svestka, "A continuous variable representation of the TSP," *Math Prog*, v15, 1978, pp211-213.

TSP gain model in AMPL.

About 7 secs.

```
# TSP, Svestka formulation. Ref: JA Svestka, "A continuous variable
# representation of the TSP," Math Prog, v15, 1978, pp211-213.
param N integer > 2; # Number of nodes
set Nodes ordered := {1..N};
set Arcs := {i in Nodes, j in Nodes: i <> j};

param C{(i,j) in Arcs};
param F := 0.01;
var x {(i,j) in Arcs} binary;          var y {(i,j) in Arcs} >= 0;

minimize Tourlength: sum {(i,j) in Arcs} C[i,j] * x[i,j];

subject to Demandy {i in Nodes: i>=2}: sum{(i,j) in Arcs} y[j,i] >= 1;
subject to Flowy {i in Nodes: i>=2}:
sum {(i,j) in Arcs} y[i,j] - sum {(i,j) in Arcs} y[j,i] = F;
subject to Totalx: sum {(i,j) in Arcs} x[i,j] <= N;
subject to Arcgain {(i,j) in Arcs}: y[i,j] <= (1 + N*F)*x[i,j];

# These rows tighten the model.
subject to Demandx {i in Nodes: i>=2}: sum{(i,j) in Arcs} x[j,i] = 1;
subject to Flowx {i in Nodes: i >= 2}:
sum {(i,j) in Arcs} x[i,j] = sum {(i,j) in Arcs} x[j,i];
subject to Startx: sum {i in Nodes: i >= 2} x[1,i] = 1;
subject to Starty: sum {i in Nodes: i >= 2} y[1,i] = 1;
```

AMPL solution

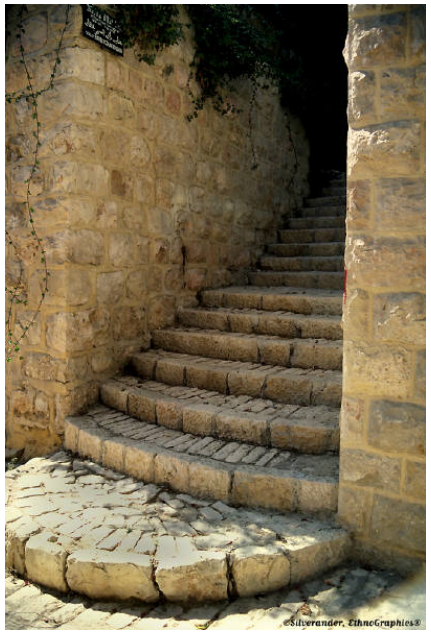
```
y1,8     y8,7  y7,5  y5,4  y4,3  y3,6  y6,9  y9,12 y12,11      y11,10
y10,2    y2,1
1.00     1.01  1.02  1.03  1.04  1.05  1.06  1.07  1.08  1.09  1.1
1.11
x1,8     x8,7  x7,5  x5,4  x4,3  x3,6  x6,9  x9,12 x12,11      x11,10
```

A formulation of steps for the TSP, Dantzig

Indices: i, j, k = city, t = step.
Variables: $x_{ijt} = 1$ if we drive from city i to city j at step t, else 0.

Minimise total cost: $\min \sum_i \sum_j \sum_t c_{ij} x_{ijt}$,
Flow in = flow out: $\sum_i x_{i,j,t} - \sum_k x_{j,k,t+1} = 0$ for all j, and $t=1,\ldots,n$.
Go to each city once: $\sum_j \sum_t x_{ijt} = 1$, for $i=1,\ldots,n$.
Binary integrality: $x_{ijt} \in \{0, 1\}$ for all i, j, t.

This is tighter than the earlier formulations.
Unfortunately this requires n^3 variables.
A 50-city problem would require 125,000 variables!



Next: solution methods.

- Assignment + cuts.
- Assignment + B&B.
- Min spanning tree.

TSP step model in AMPL. About 7 secs.

```
# TSP steps. Ref Linear Progg & Extensions, Dantzig, 1963;
param N integer > 2; # Number of nodes
set Nodes ordered := {1..N};
set Arcs := {i in Nodes, j in Nodes, k in Nodes: i <> j};

param C{i in Nodes, j in Nodes: i <> j};
var x {(i,j,k) in Arcs} binary;

minimize Tourlength: sum{(i,j,k) in Arcs: i<>j} C[i,j]*x[i,j,k];

subject to Demand {i in Nodes}: sum{(i,j,k) in Arcs} x[i,j,k] = 1;
subject to End {i in Nodes}:
     sum {j in Nodes: j <> i} x[j,i,N]
   = sum {k in Nodes: k <> i} x[i,k,1];

subject to Step {t in Nodes, j in Nodes: t <= N - 1}:
     sum {i in Nodes: i <> j} x[i,j,t]
   = sum {k in Nodes: k <> j} x[j,k,t+1];
```

AMPL solution

8,1,1 1,2,2 2,10,3 10,11,4 11,12,5 12,9,6 9,6,7 6,3,8 3,4,9 4,5,10 5,7,11 7,8,12