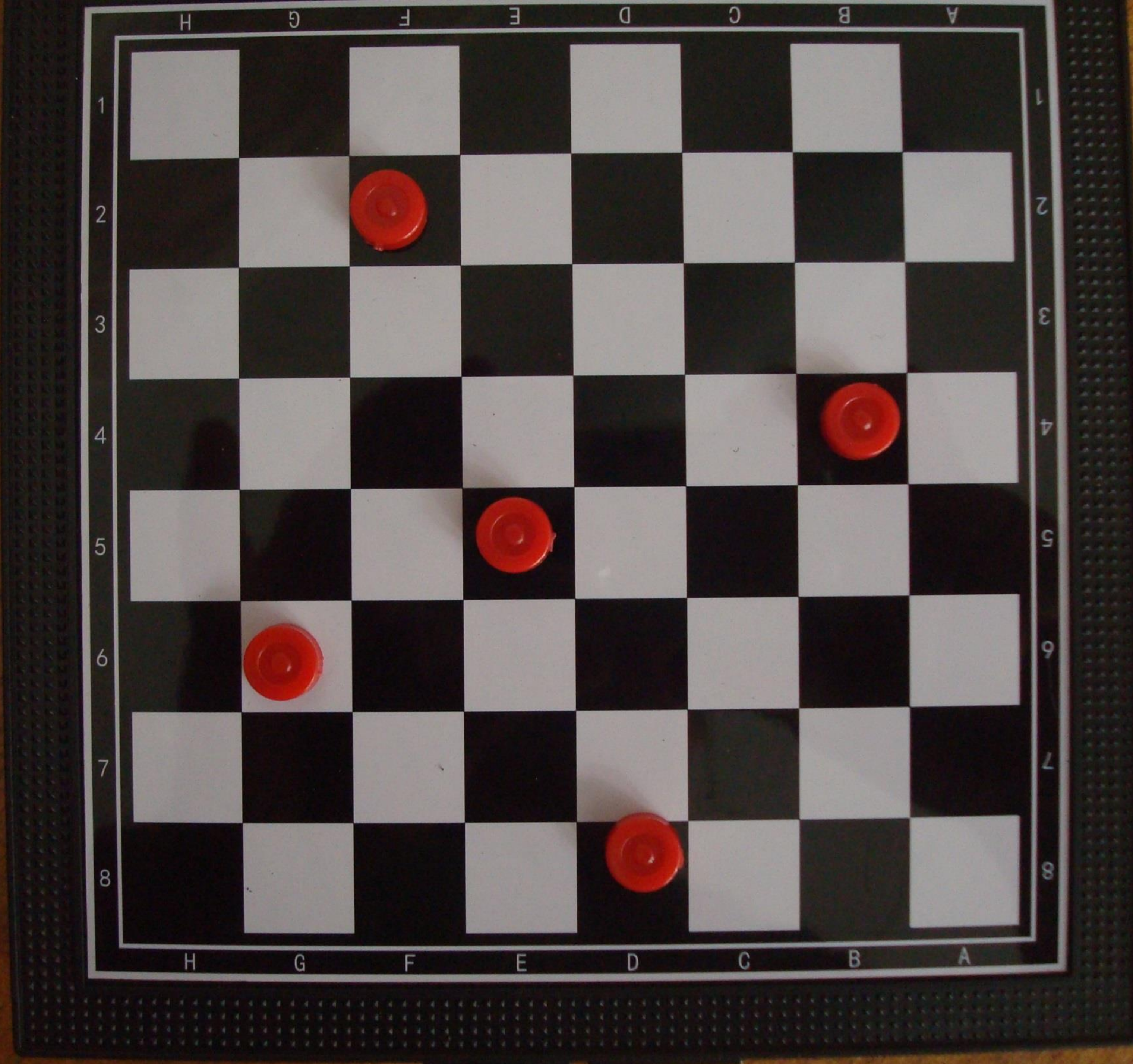
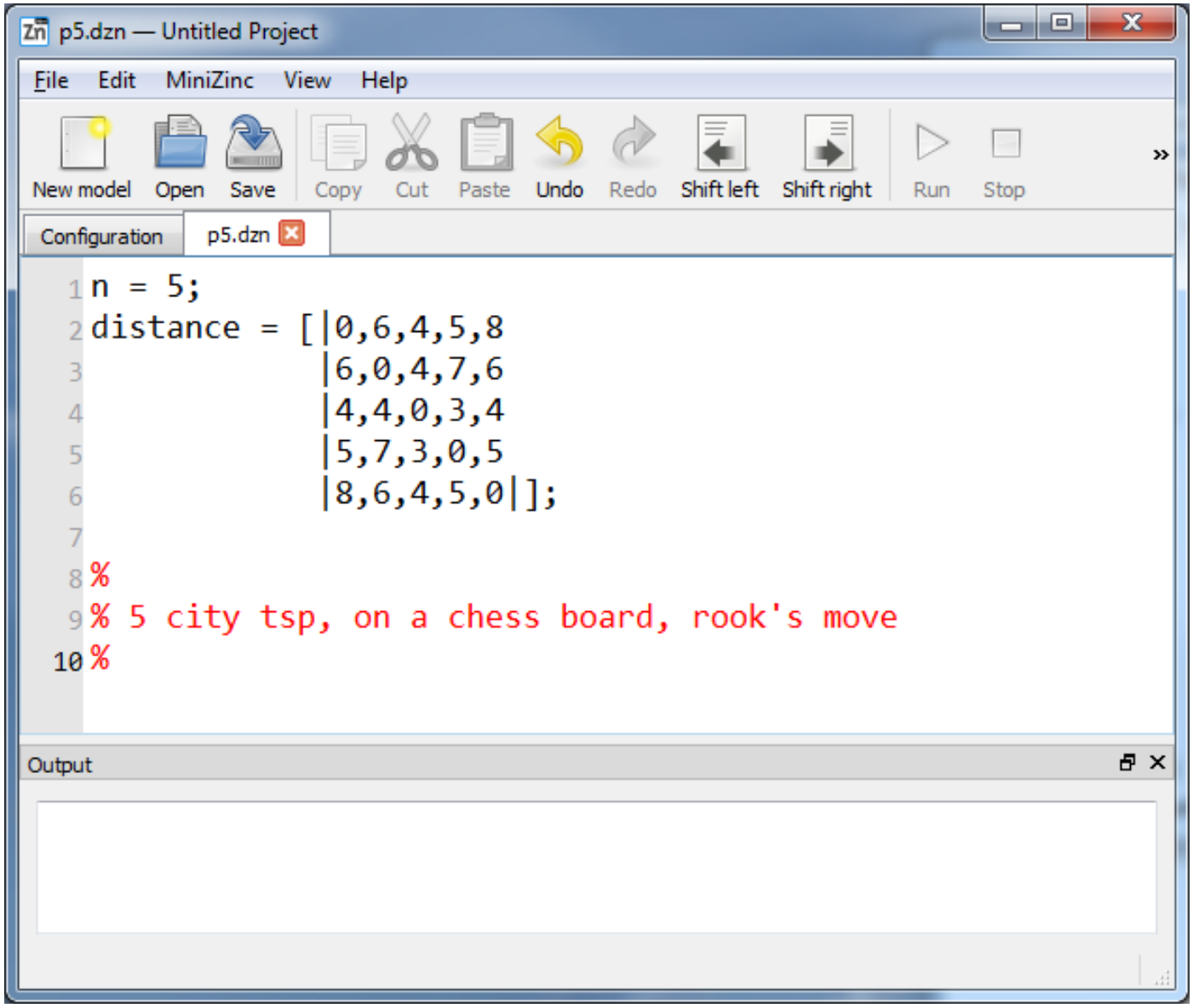
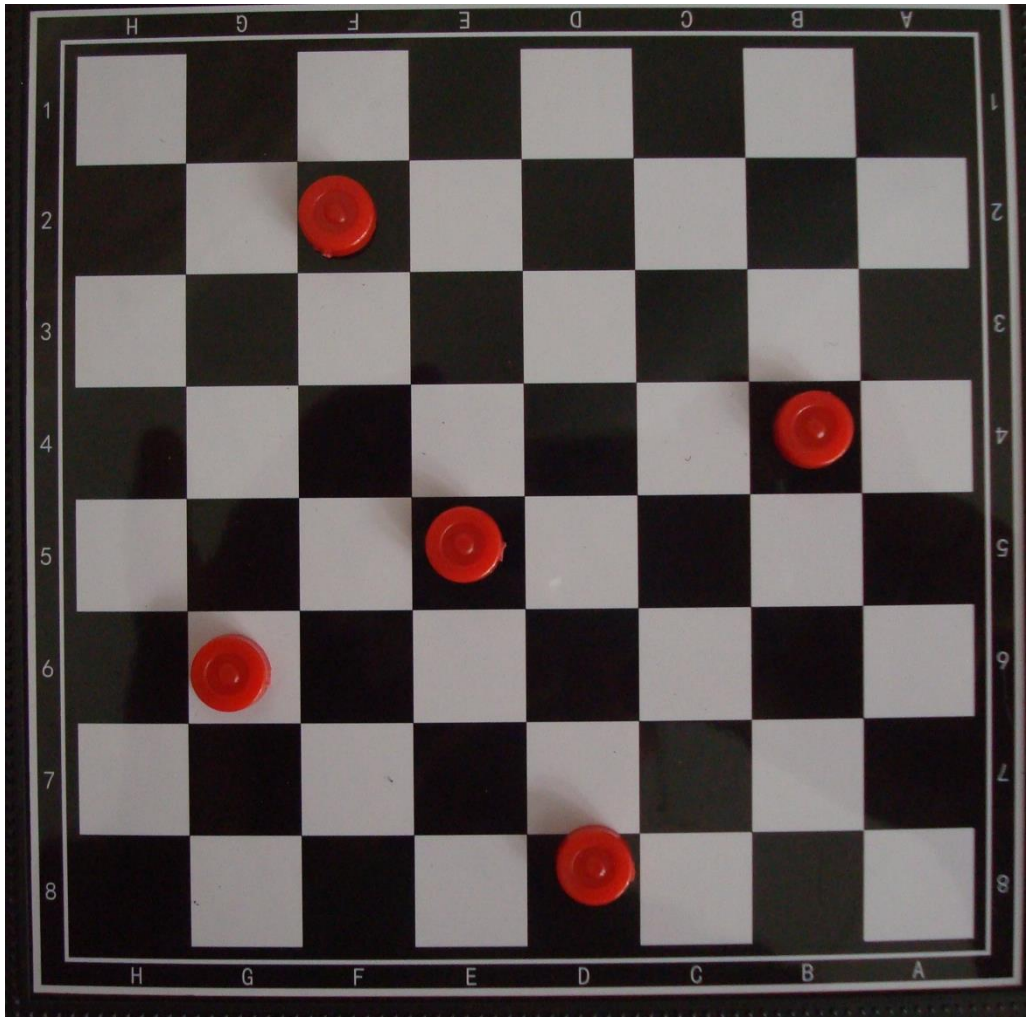
Small TSP


MAGNETIC GAMES
MAGNETIC GAMES
4 Games in 1
2-4
3+
SONY


H G F E D C B A
1 2 3 4 5 6 7 8

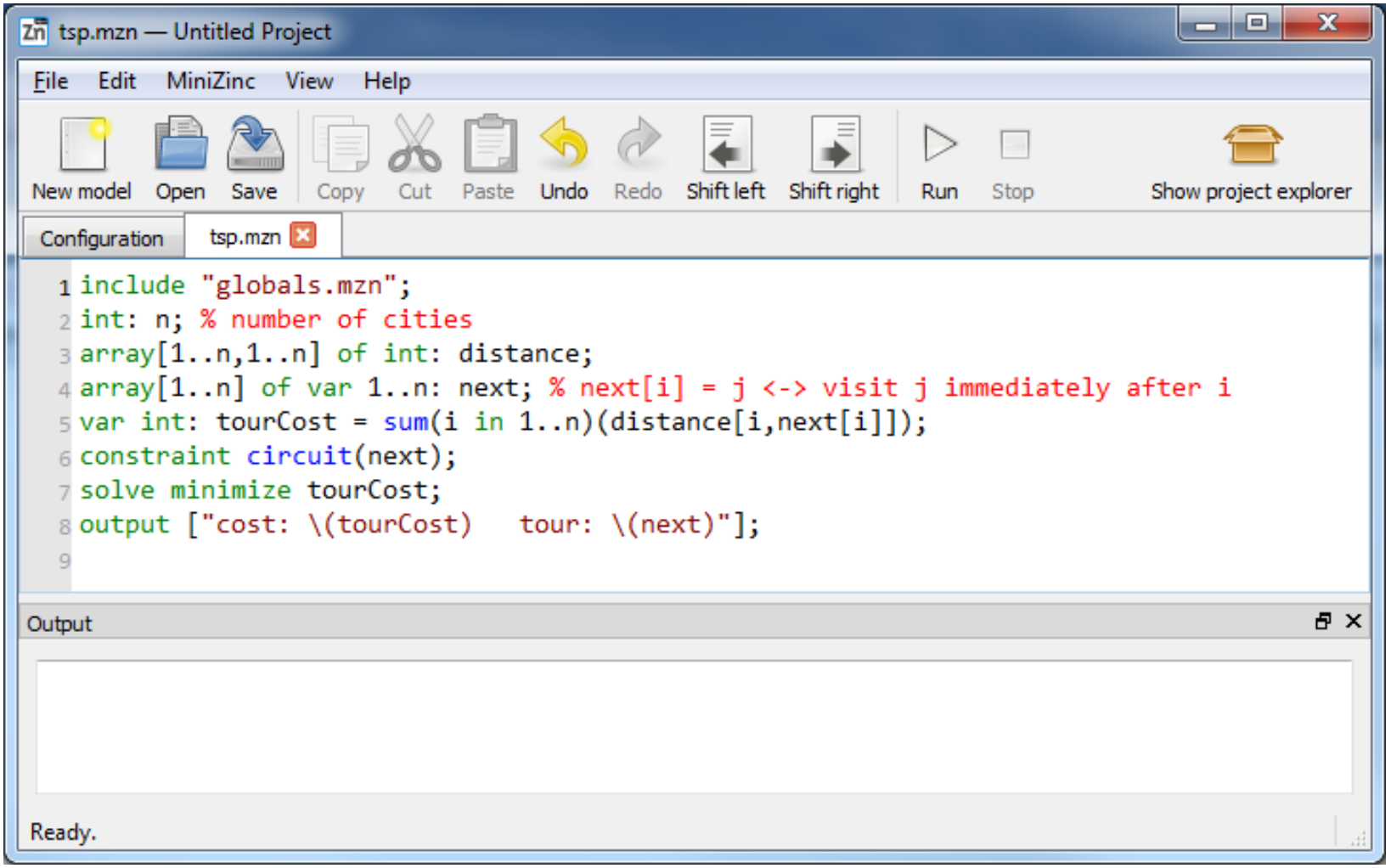

p5.dzn — Untitled Project
File Edit MiniZinc View Help
New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop
Configuration p5.dzn
n = 5;
distance = [|0,6,4,5,8
|6,0,4,7,6
|4,4,0,3,4
|5,7,3,0,5
|8,6,4,5,0|];
%
% 5 city tsp, on a chess board, rook's move
%
Output

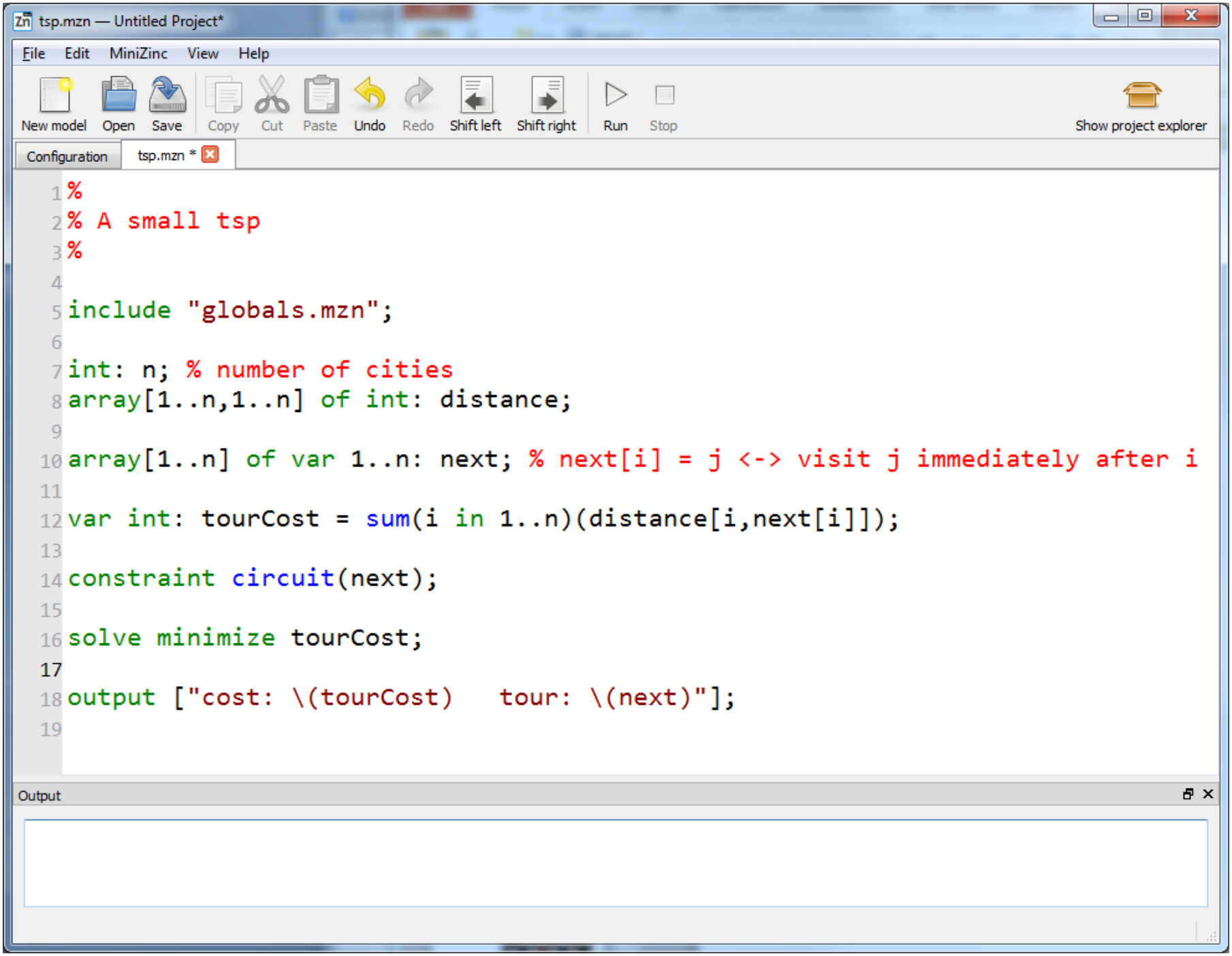


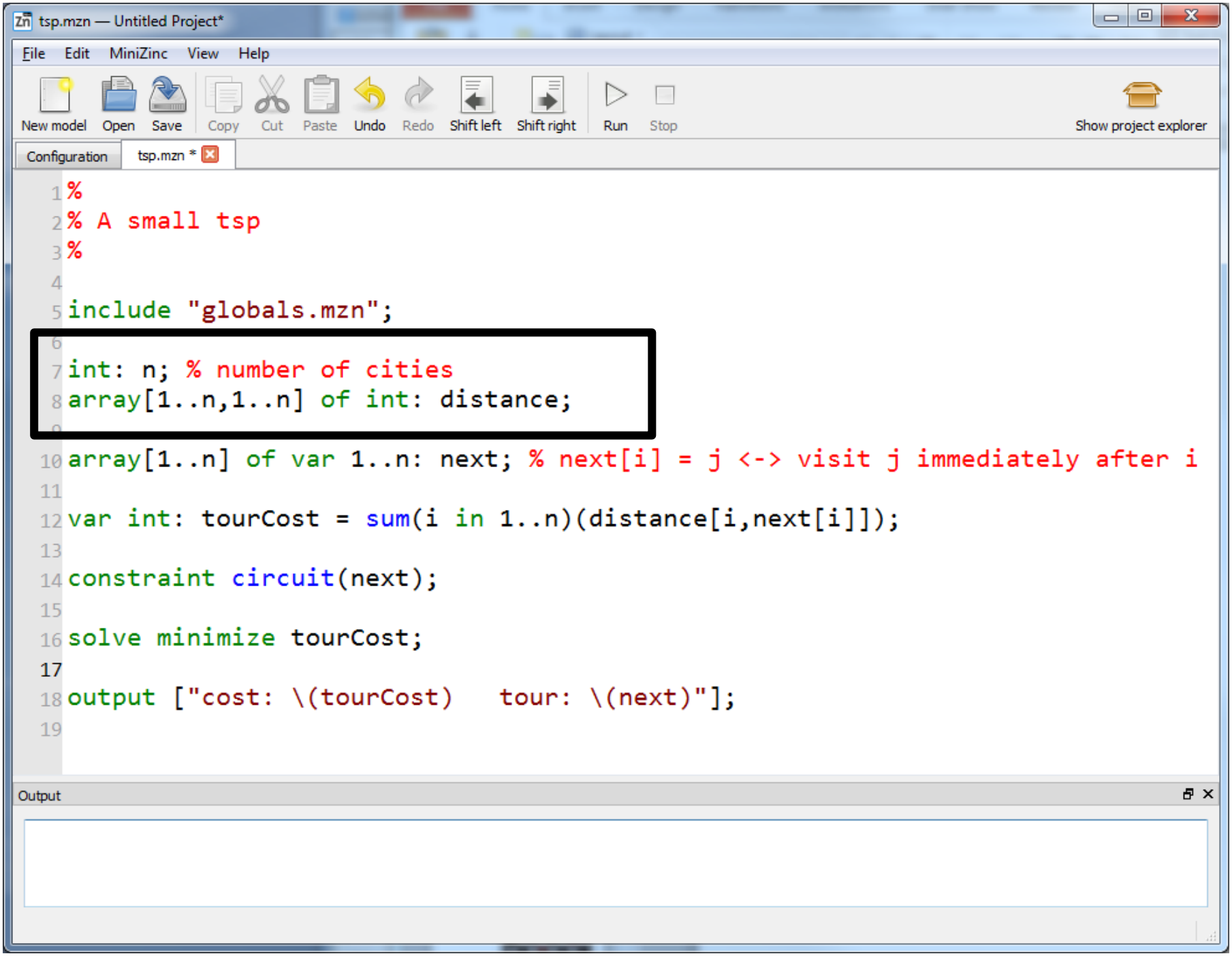

```
n = 5;
distance = [|0,6,4,5,8
            |6,0,4,7,6
            |4,4,0,3,4
            |5,7,3,0,5
            |8,6,4,5,0|];
```

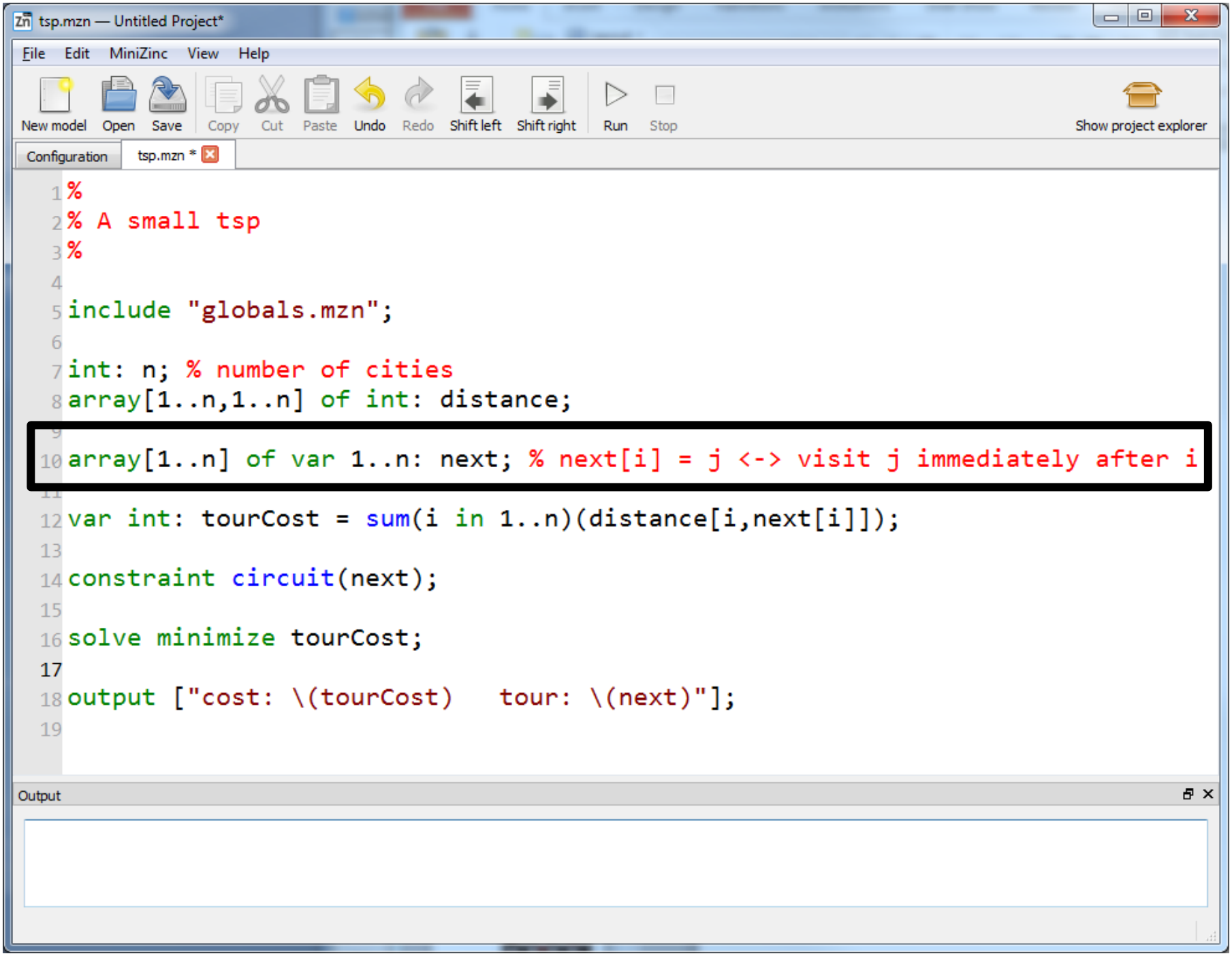
Single successor model

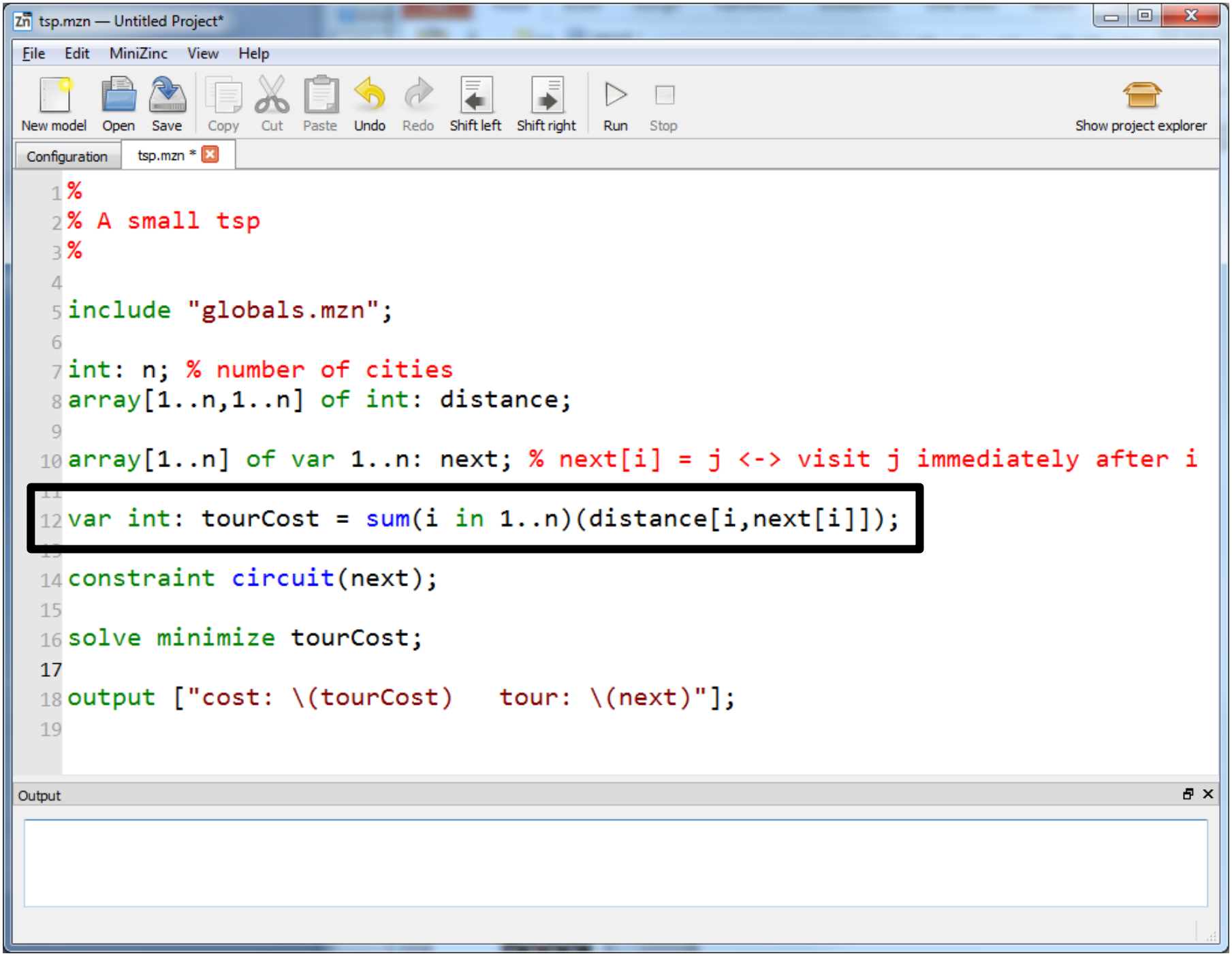
next[i] = j means “visit city j immediately after city i”

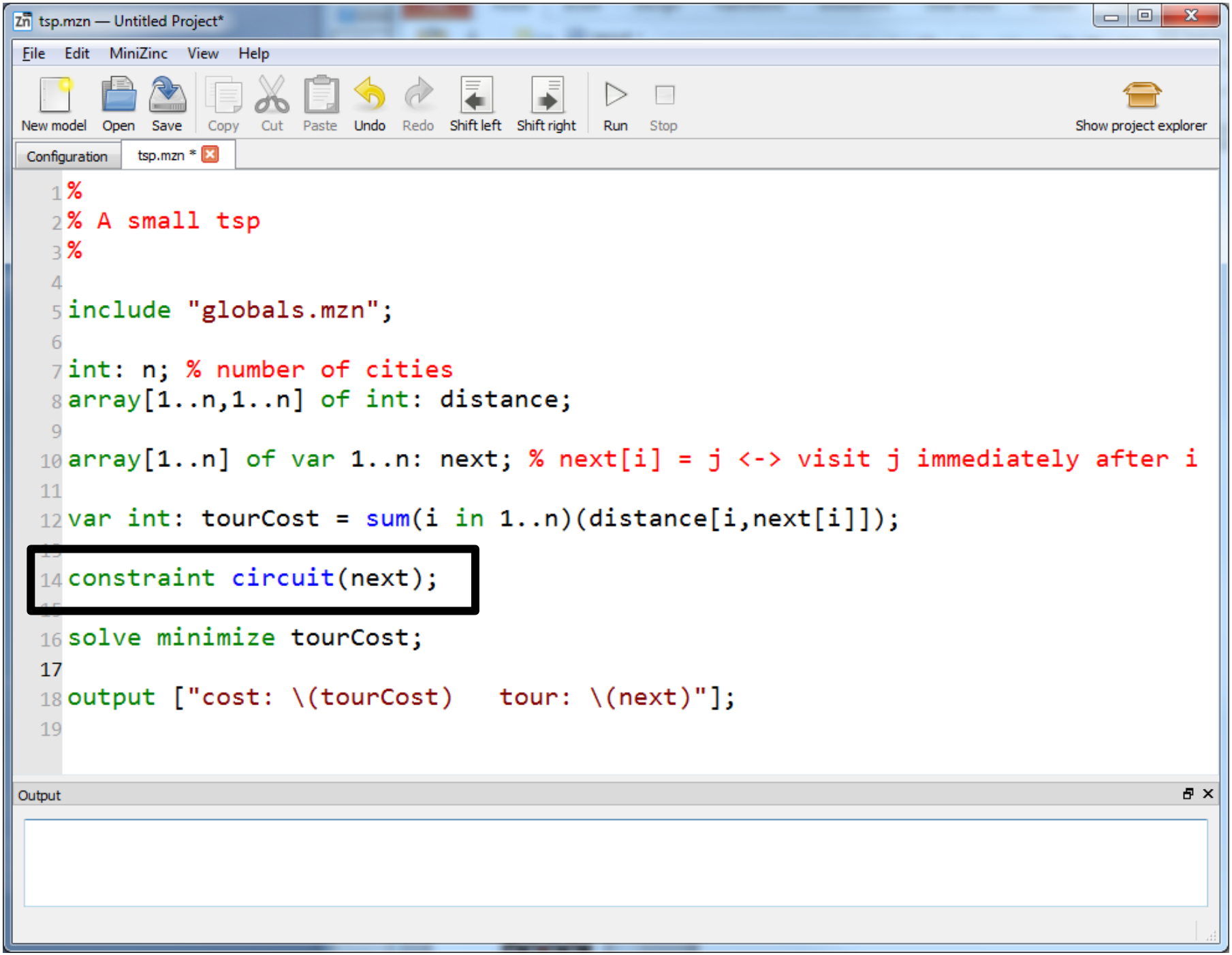

tsp.mzn — Untitled Project
File
Edit
MiniZinc
View
Help
New model
Open
Save
Copy
Cut
Paste
Undo
Redo
Shift left
Shift right
Run
Stop
Show project explorer
Configuration
tsp.mzn
1 include "globals.mzn";
2 int: n; % number of cities
3 array[1..n,1..n] of int: distance;
4 array[1..n] of var 1..n: next; % next[i] = j <-> visit j immediately after i
5 var int: tourCost = sum(i in 1..n)(distance[i,next[i]]);
6 constraint circuit(next);
7 solve minimize tourCost;
8 output ["cost: \(tourCost) tour: \(next)"];
9
Output
Ready.

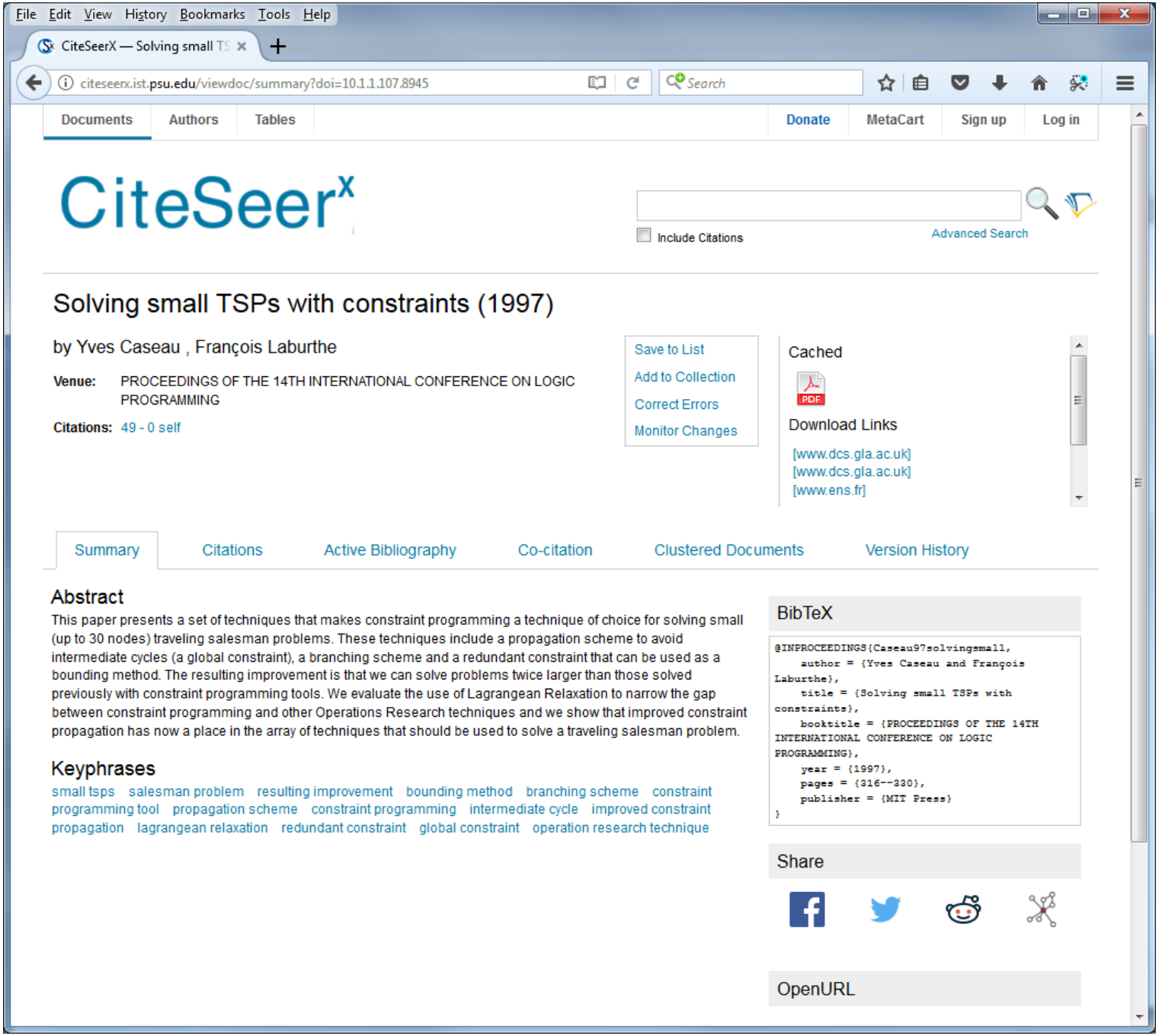

tsp.mzn — Untitled Project*
File Edit MiniZinc View Help
New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop Show project explorer
Configuration tsp.mzn *
1 %
2 % A small tsp
3 %
4
5 include "globals.mzn";
6
7 int: n; % number of cities
8 array[1..n,1..n] of int: distance;
9
10 array[1..n] of var 1..n: next; % next[i] = j <-> visit j immediately after i
11
12 var int: tourCost = sum(i in 1..n)(distance[i,next[i]]);
13
14 constraint circuit(next);
15
16 solve minimize tourCost;
17
18 output ["cost: \(tourCost) tour: \(next)"];
19
Output


tsp.mzn — Untitled Project*
File Edit MiniZinc View Help
New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop Show project explorer
Configuration tsp.mzn *
1 %
2 % A small tsp
3 %
4
5 include "globals.mzn";
6
7 int: n; % number of cities
8 array[1..n,1..n] of int: distance;
9
10 array[1..n] of var 1..n: next; % next[i] = j <-> visit j immediately after i
11
12 var int: tourCost = sum(i in 1..n)(distance[i,next[i]]);
13
14 constraint circuit(next);
15
16 solve minimize tourCost;
17
18 output ["cost: \(tourCost) tour: \(next)"];
19
Output


tsp.mzn — Untitled Project*
File Edit MiniZinc View Help
New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop Show project explorer
Configuration tsp.mzn *
1 %
2 % A small tsp
3 %
4
5 include "globals.mzn";
6
7 int: n; % number of cities
8 array[1..n,1..n] of int: distance;
9
10 array[1..n] of var 1..n: next; % next[i] = j <-> visit j immediately after i
11
12 var int: tourCost = sum(i in 1..n)(distance[i,next[i]]);
13
14 constraint circuit(next);
15
16 solve minimize tourCost;
17
18 output ["cost: \(tourCost) tour: \(next)"];
19
Output


tsp.mzn — Untitled Project*
File Edit MiniZinc View Help
New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop Show project explorer
Configuration tsp.mzn *
1 %
2 % A small tsp
3 %
4
5 include "globals.mzn";
6
7 int: n; % number of cities
8 array[1..n,1..n] of int: distance;
9
10 array[1..n] of var 1..n: next; % next[i] = j <-> visit j immediately after i
12 var int: tourCost = sum(i in 1..n)(distance[i,next[i]]);
14 constraint circuit(next);
15
16 solve minimize tourCost;
17
18 output ["cost: \(tourCost) tour: \(next)"];
19
Output


tsp.mzn — Untitled Project*
File Edit MiniZinc View Help
New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop Show project explorer
Configuration tsp.mzn *
1 %
2 % A small tsp
3 %
4
5 include "globals.mzn";
6
7 int: n; % number of cities
8 array[1..n,1..n] of int: distance;
9
10 array[1..n] of var 1..n: next; % next[i] = j <-> visit j immediately after i
11
12 var int: tourCost = sum(i in 1..n)(distance[i,next[i]]);
14 constraint circuit(next);
16 solve minimize tourCost;
17
18 output ["cost: \(tourCost) tour: \(next)"];
19
Output


File Edit View History Bookmarks Tools Help
CiteSeerX — Solving small TS
citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.107.8945
Search
Documents Authors Tables Donate MetaCart Sign up Log in
CiteSeerx
Include Citations
Advanced Search
Solving small TSPs with constraints (1997)
by Yves Caseau , François Laburthe
Venue: PROCEEDINGS OF THE 14TH INTERNATIONAL CONFERENCE ON LOGIC PROGRAMMING
Citations: 49 - 0 self
Save to List
Add to Collection
Correct Errors
Monitor Changes
Cached
Download Links
[www.dcs.gla.ac.uk]
[www.dcs.gla.ac.uk]
[www.ens.fr]
Summary Citations Active Bibliography Co-citation Clustered Documents Version History
Abstract
This paper presents a set of techniques that makes constraint programming a technique of choice for solving small (up to 30 nodes) traveling salesman problems. These techniques include a propagation scheme to avoid intermediate cycles (a global constraint), a branching scheme and a redundant constraint that can be used as a bounding method. The resulting improvement is that we can solve problems twice larger than those solved previously with constraint programming tools. We evaluate the use of Lagrangean Relaxation to narrow the gap between constraint programming and other Operations Research techniques and we show that improved constraint propagation has now a place in the array of techniques that should be used to solve a traveling salesman problem.
Keyphrases
small tsps salesman problem resulting improvement bounding method branching scheme constraint programming tool propagation scheme constraint programming intermediate cycle improved constraint propagation lagrangean relaxation redundant constraint global constraint operation research technique
BibTeX
@INPROCEEDINGS{Caseau97solvingsmall,
author = {Yves Caseau and François Laburthe},
title = {Solving small TSPs with constraints},
booktitle = {PROCEEDINGS OF THE 14TH INTERNATIONAL CONFERENCE ON LOGIC PROGRAMMING},
year = {1997},
pages = {316--330},
publisher = {MIT Press}
}
Share
OpenURL

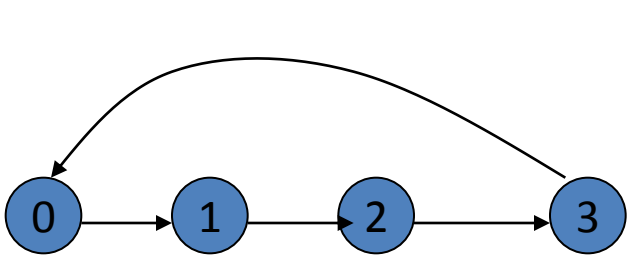
The single successor model

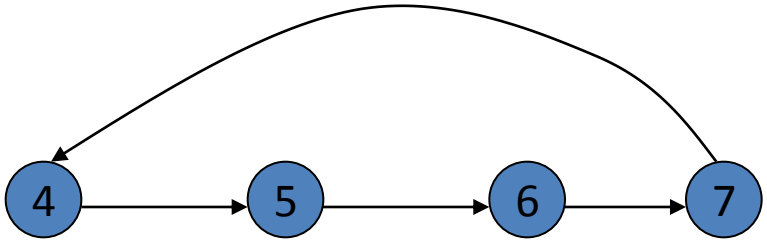
$$next_i \in \{0..n-1\}$$

$$next_i = j \leftrightarrow city_j \text{ immediately follows } city_i$$

NOT A TOUR!

1	2	3	0	5	6	7	4





We need subtour elimination

3.1.3. Subtour elimination

However, propagation of the assignment constraint is not enough, since it accepts assignments that are the union of disjoint tours. In order to propagate the *nocycle* constraint to avoid subtours, we explicitly store the start, end and total number of arcs of the chain (defined by *Next*) going through node *1*, which we respectively denote $start_1$, end_1, $length_1$. Upon each assignment *Next(x):=y* we look if subchains adjacent to *x* or *y* have already been built and call *b* the end of the chain starting from y and *a* the start of the chain ending in *x*, $length_a$ the number of arcs in the chain from *a* to *x* and $length_b$ the number of arcs in the chain from *y* to *b*.

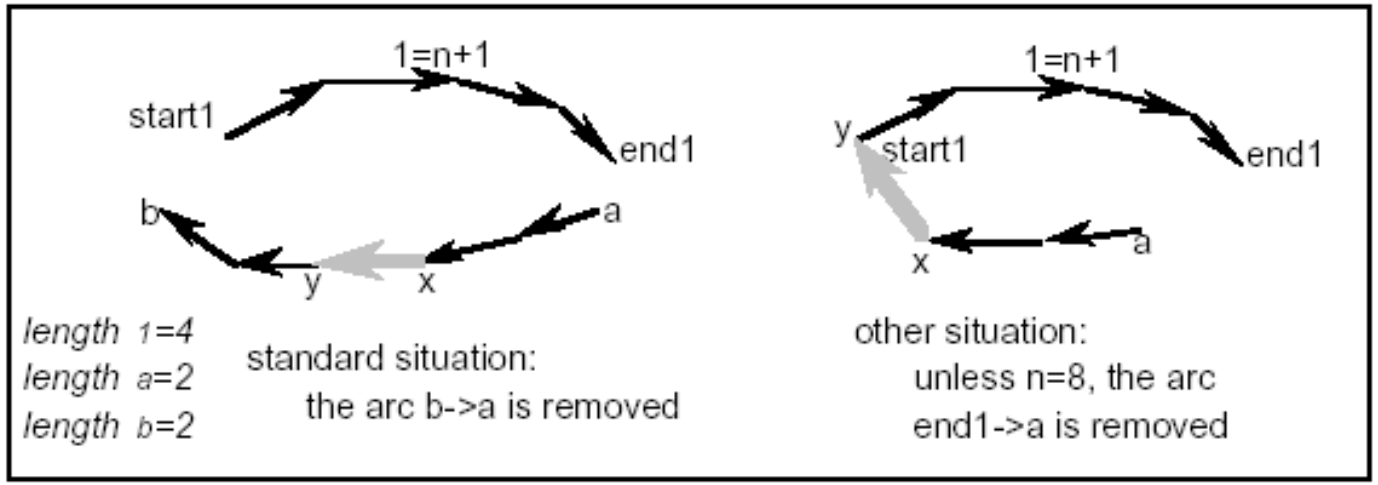
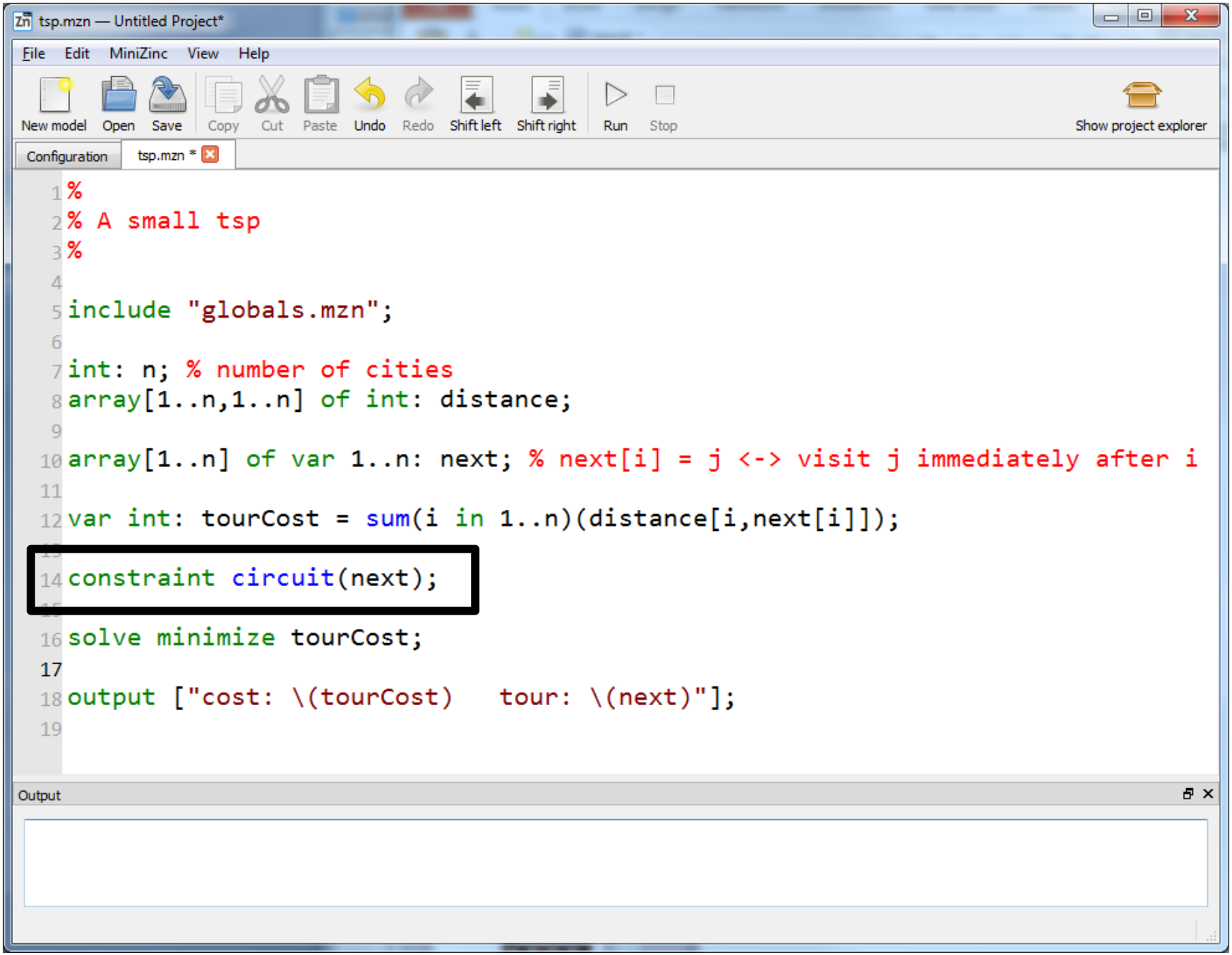
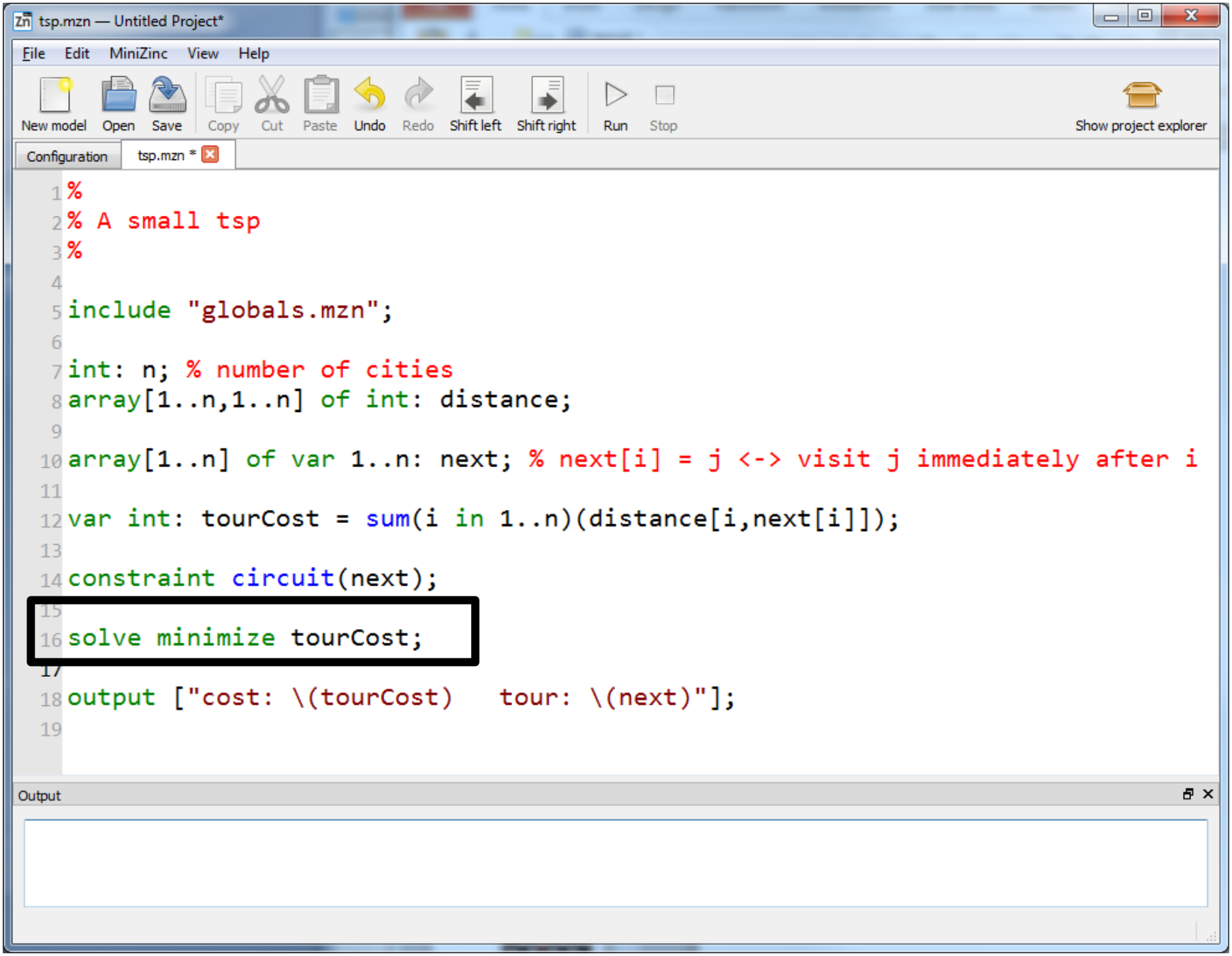
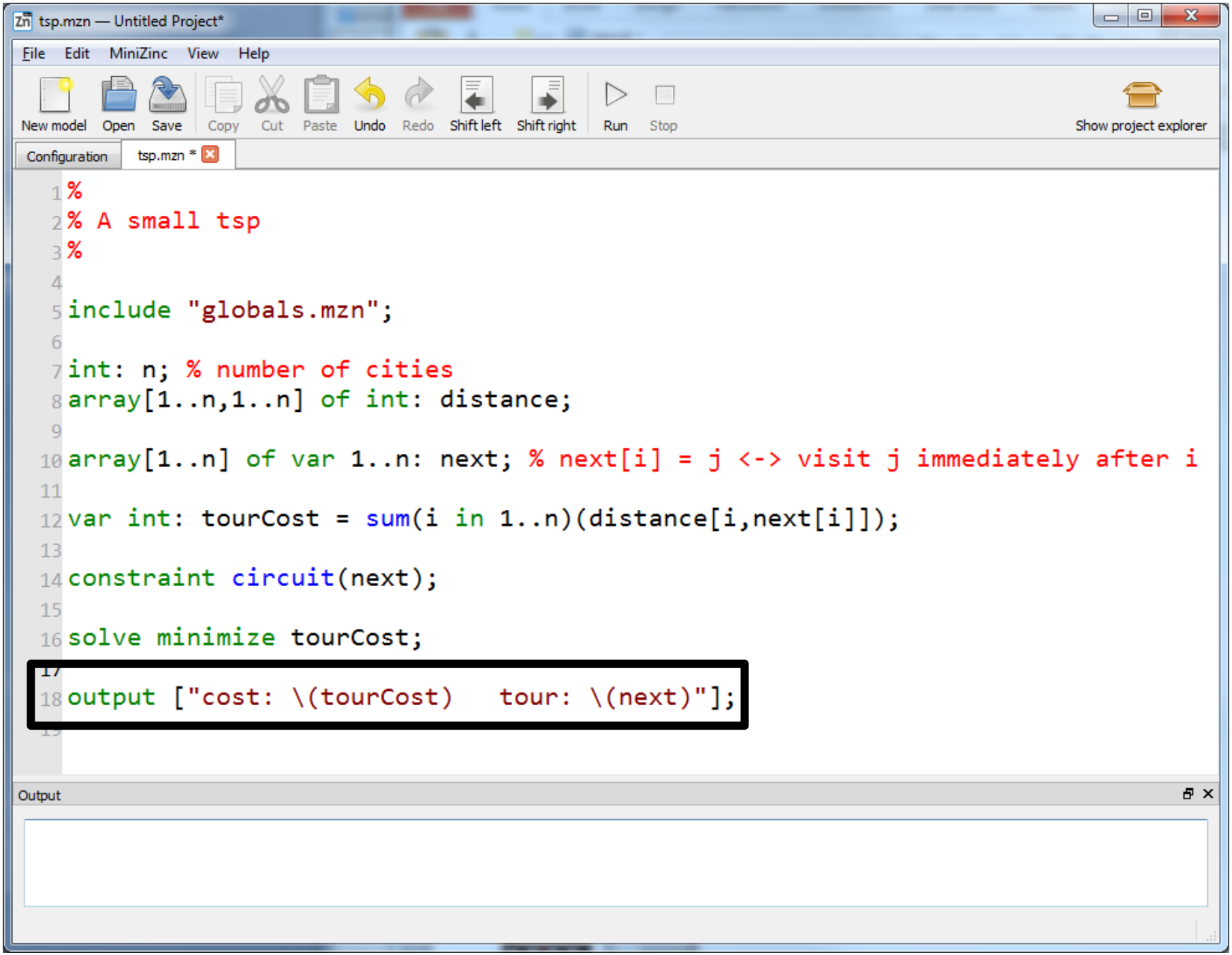


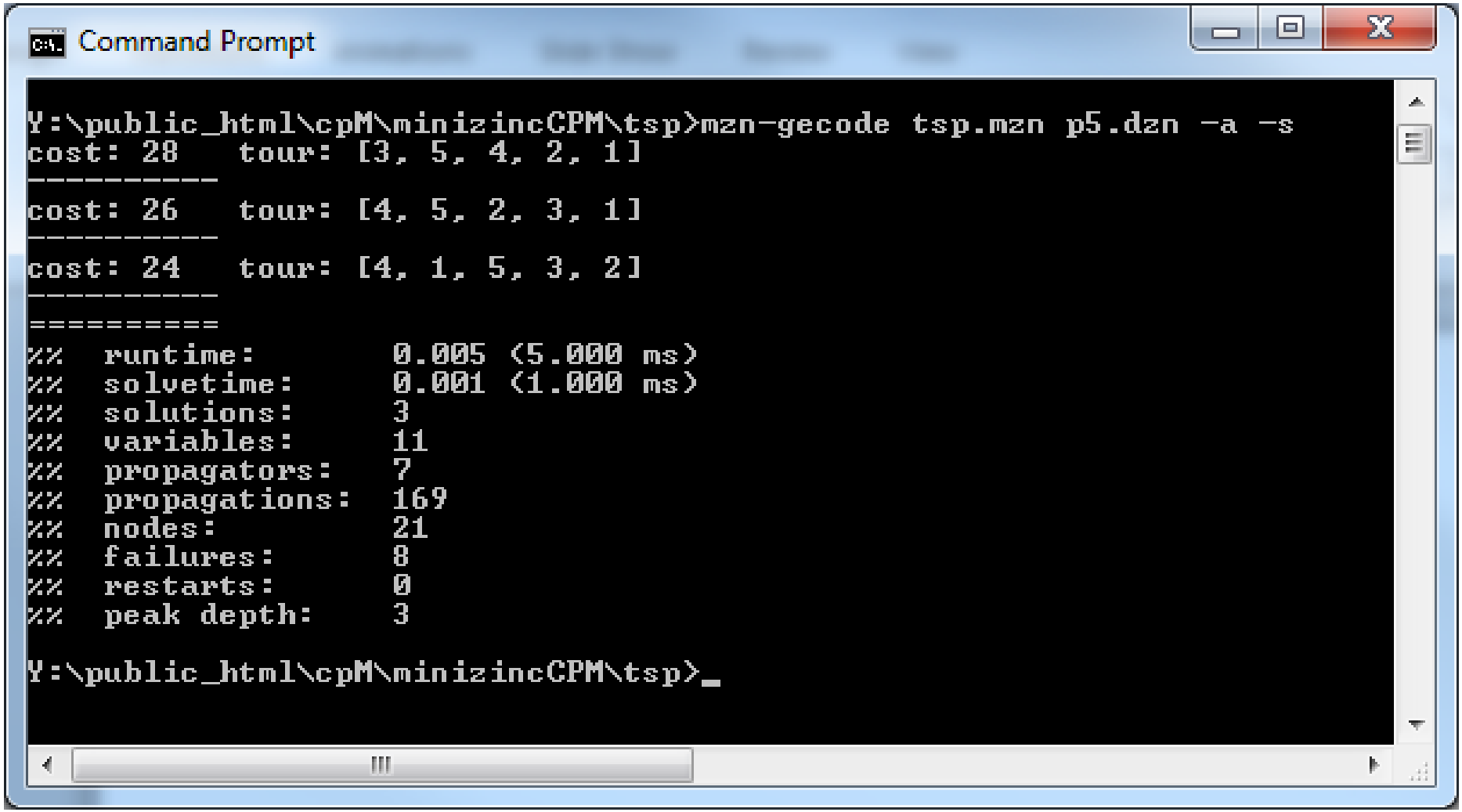
Figure 1: Propagation of the nocycle constraint

- If $x=end_1$ and $length_1+length_b<n-2$ we infer $Next(b) \neq start_1$.
- If $y=start_1$ and $length_1+length_a<n-2$ we infer $Next(end_1) \neq a$
- Otherwise, we infer $Next(b) \neq a$.

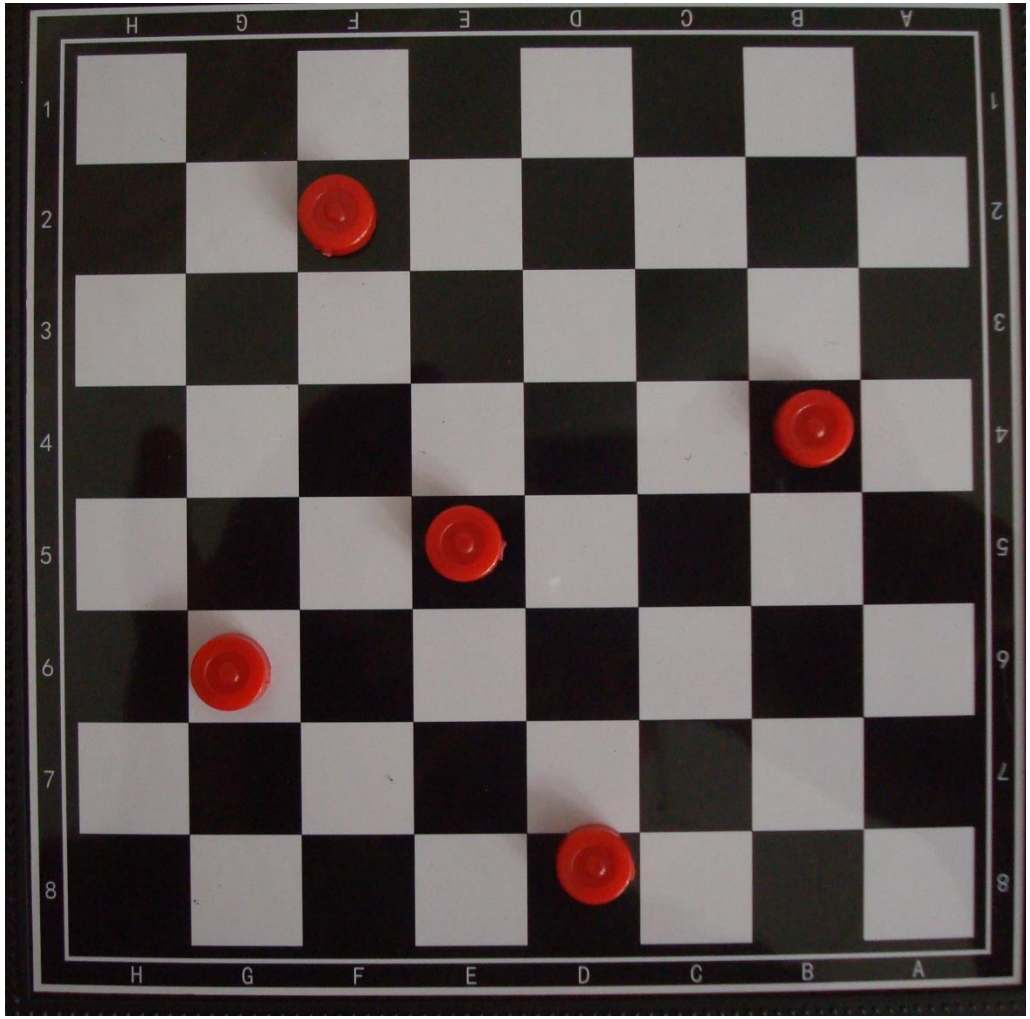

tsp.mzn — Untitled Project*
File Edit MiniZinc View Help
New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop Show project explorer
Configuration tsp.mzn *
1 %
2 % A small tsp
3 %
4
5 include "globals.mzn";
6
7 int: n; % number of cities
8 array[1..n,1..n] of int: distance;
9
10 array[1..n] of var 1..n: next; % next[i] = j <-> visit j immediately after i
11
12 var int: tourCost = sum(i in 1..n)(distance[i,next[i]]);
14 constraint circuit(next);
16 solve minimize tourCost;
17
18 output ["cost: \(tourCost) tour: \(next)"];
19
Output


tsp.mzn — Untitled Project*
File Edit MiniZinc View Help
New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop Show project explorer
Configuration tsp.mzn *
1 %
2 % A small tsp
3 %
4
5 include "globals.mzn";
6
7 int: n; % number of cities
8 array[1..n,1..n] of int: distance;
9
10 array[1..n] of var 1..n: next; % next[i] = j <-> visit j immediately after i
11
12 var int: tourCost = sum(i in 1..n)(distance[i,next[i]]);
13
14 constraint circuit(next);
15
16 solve minimize tourCost;
17
18 output ["cost: \(tourCost) tour: \(next)"];
19
Output


tsp.mzn — Untitled Project*
File Edit MiniZinc View Help
New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop Show project explorer
Configuration tsp.mzn *
1 %
2 % A small tsp
3 %
4
5 include "globals.mzn";
6
7 int: n; % number of cities
8 array[1..n,1..n] of int: distance;
9
10 array[1..n] of var 1..n: next; % next[i] = j <-> visit j immediately after i
11
12 var int: tourCost = sum(i in 1..n)(distance[i,next[i]]);
13
14 constraint circuit(next);
15
16 solve minimize tourCost;
17
18 output ["cost: \(tourCost) tour: \(next)"];
Output


Command Prompt
Y:\public_html\cpM\minizincCPM\tsp>mzn-gecode tsp.mzn p5.dzn -a -s
cost: 28 tour: [3, 5, 4, 2, 1]
----
cost: 26 tour: [4, 5, 2, 3, 1]
----
cost: 24 tour: [4, 1, 5, 3, 2]

==========
%% runtime: 0.005 (5.000 ms)
%% solvetime: 0.001 (1.000 ms)
%% solutions: 3
%% variables: 11
%% propagators: 7
%% propagations: 169
%% nodes: 21
%% failures: 8
%% restarts: 0
%% peak depth: 3
Y:\public_html\cpM\minizincCPM\tsp>

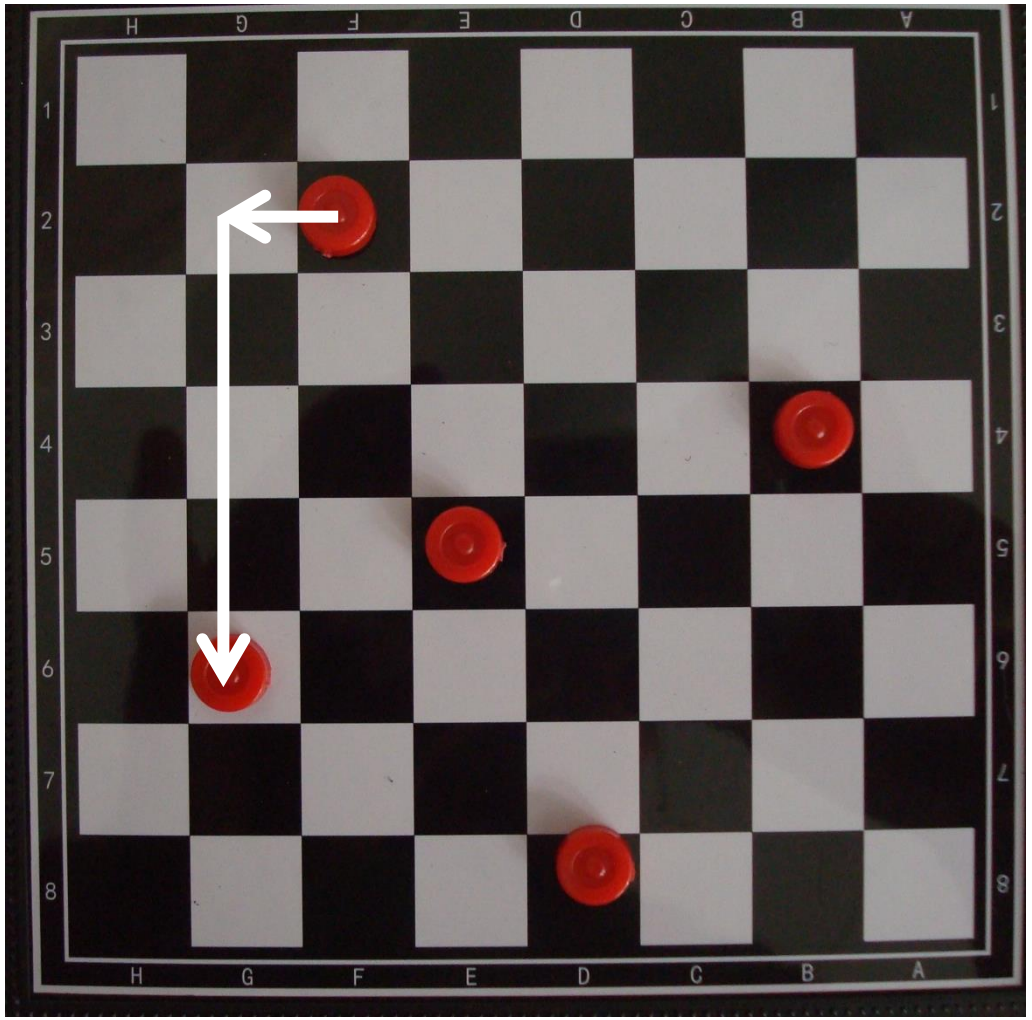


Command Prompt

```
Y:\public_html\cpM\minizincCPM\tsp>mzn-gecode tsp.mzn p5.dzn
cost: 24   tour: [4, 1, 5, 3, 2]
----------
==========

Y:\public_html\cpM\minizincCPM\tsp>_
```

1 goes to 4

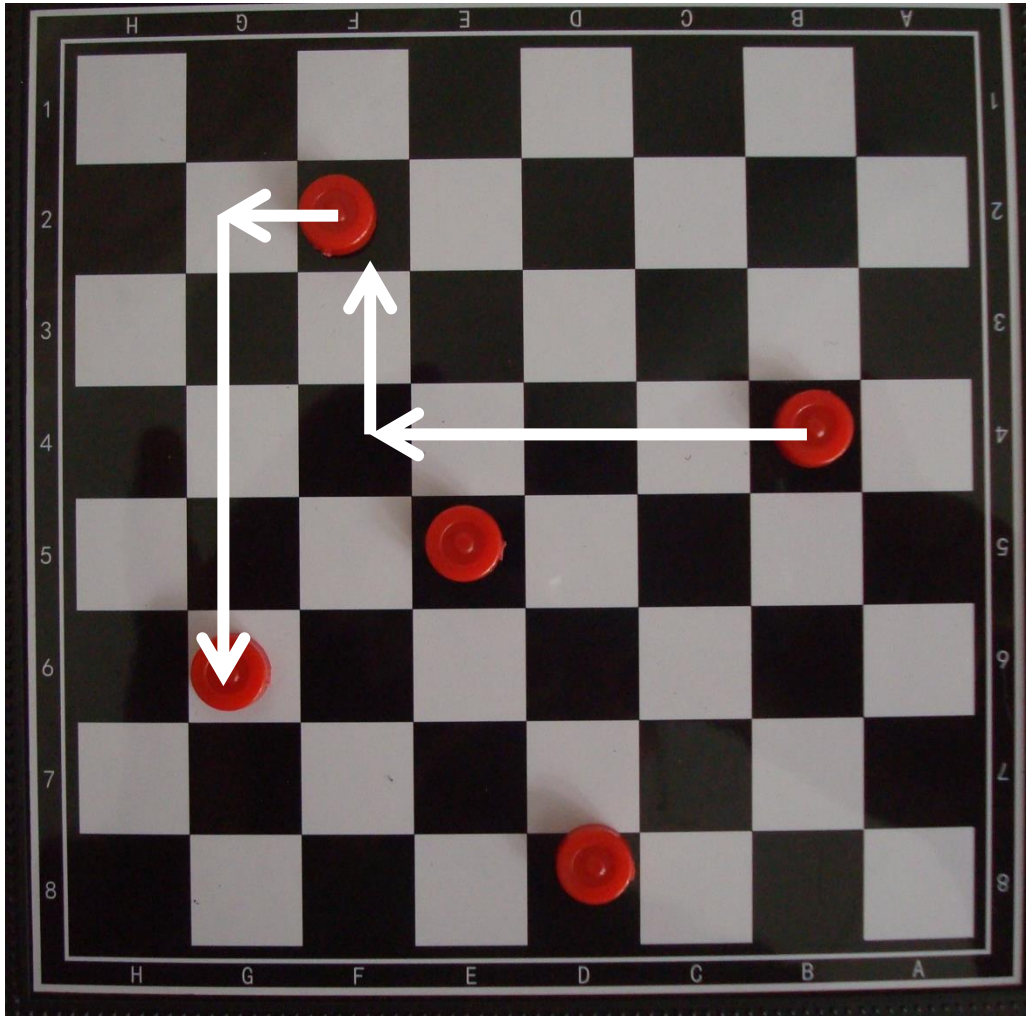


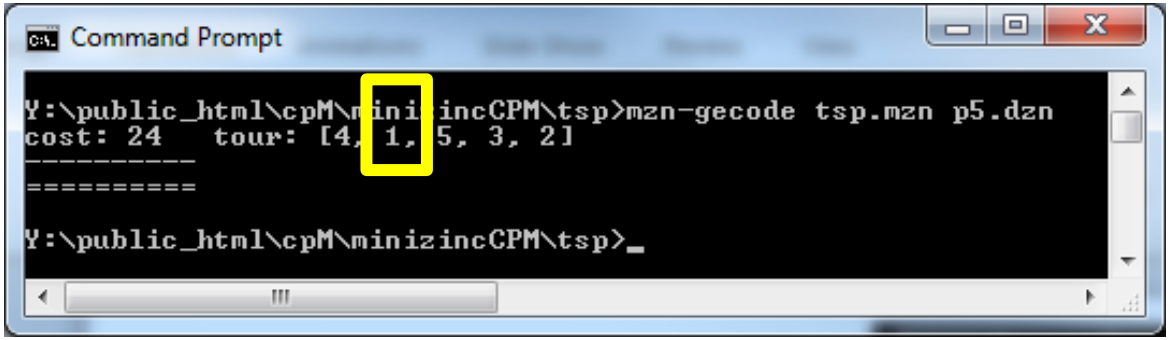
```
Command Prompt

Y:\public_html\cpM\minizincCPM\tsp>mzn-gecode tsp.mzn p5.dzn
cost: 24   tour: [4, 1, 5, 3, 2]
----------
==========

Y:\public_html\cpM\minizincCPM\tsp>_
```

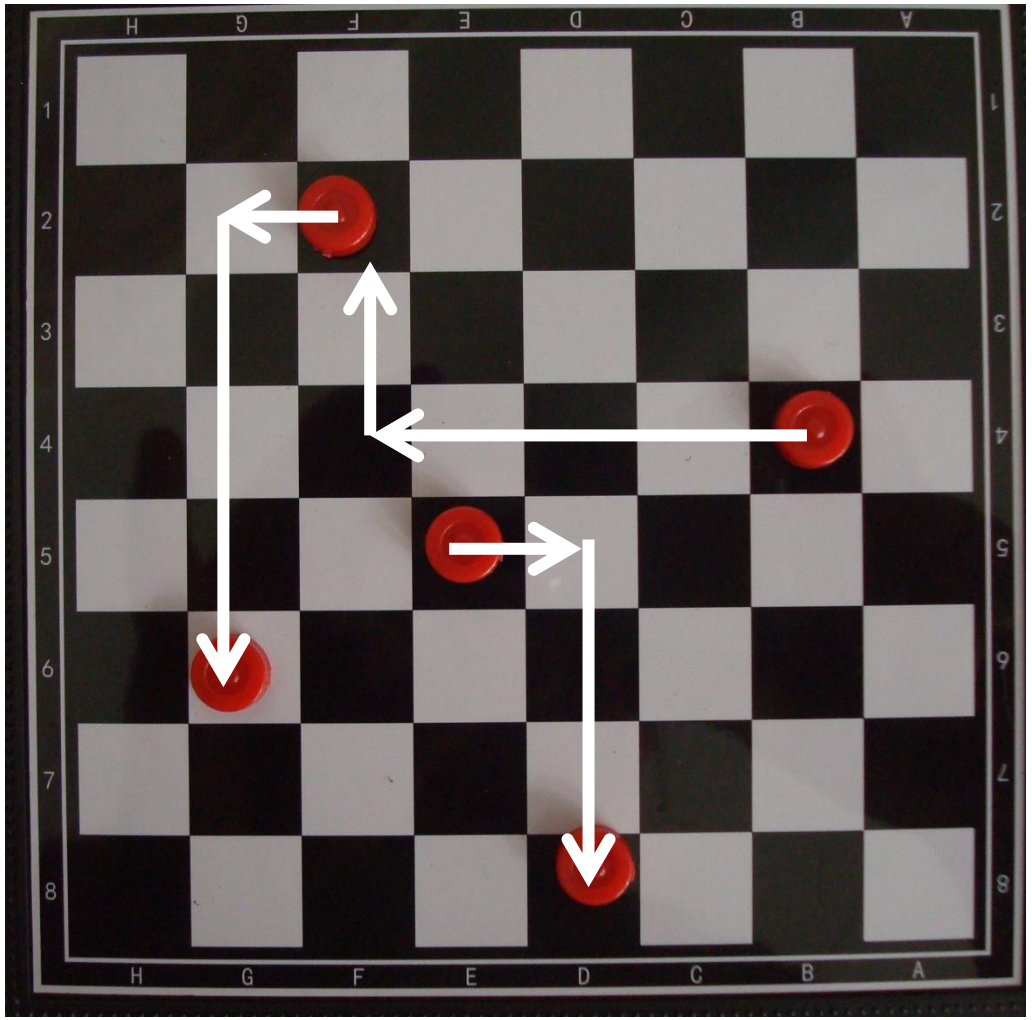
2 goes to 1




Command Prompt
Y:\public_html\cpM\minizincCPM\tsp>mzn-gecode tsp.mzn p5.dzn
cost: 24 tour: [4, 1, 5, 3, 2]

==========
Y:\public_html\cpM\minizincCPM\tsp>

3 goes to 5

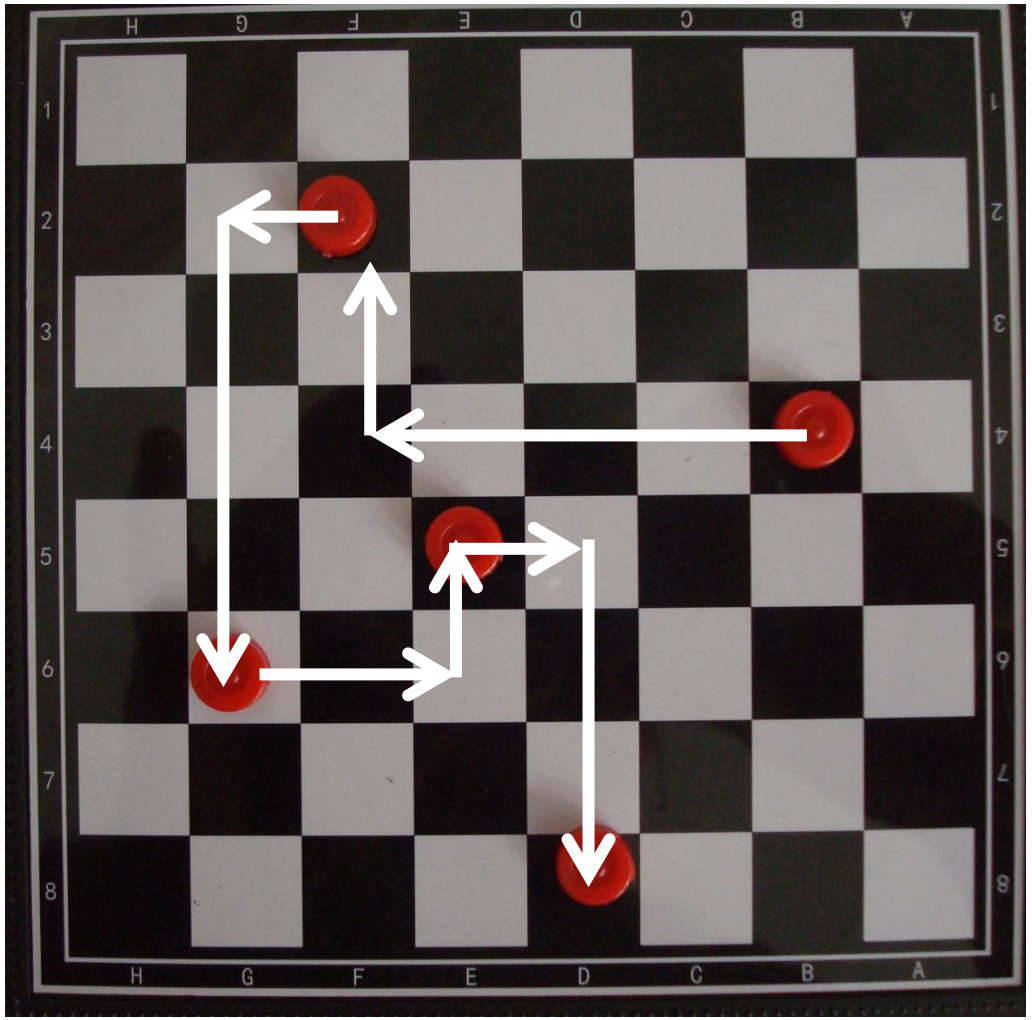


```
Command Prompt

Y:\public_html\cpM\minizincCPM\tsp>mzn-gecode tsp.mzn p5.dzn
cost: 24   tour: [4, 1, 5, 3, 2]
----------
==========

Y:\public_html\cpM\minizincCPM\tsp>_
```

4 goes to 3

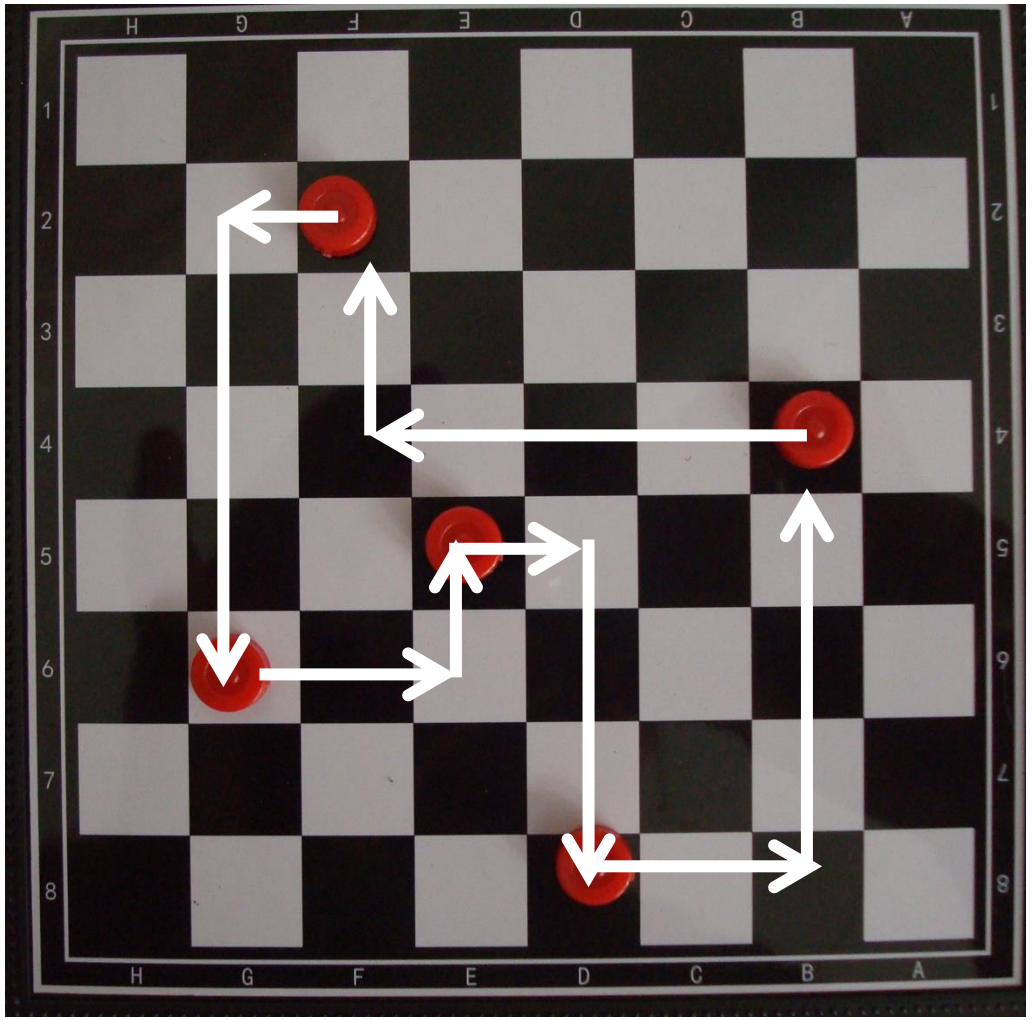


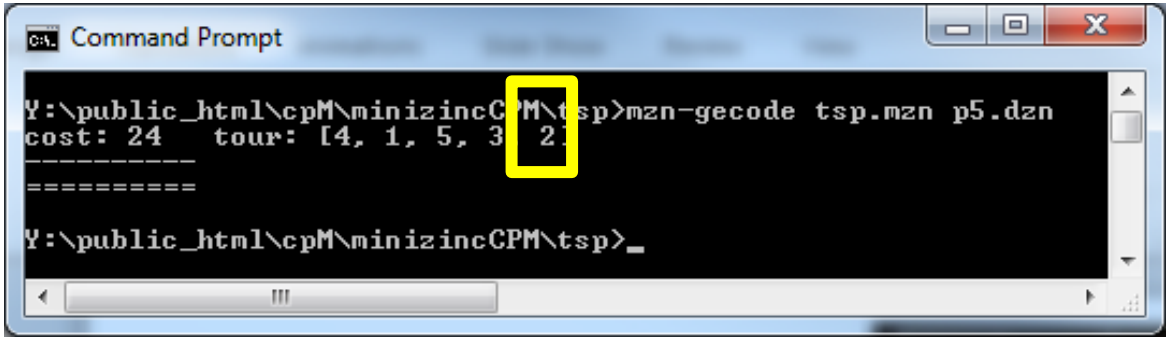
```
Command Prompt

Y:\public_html\cpM\minizincCPM\tsp>mzn-gecode tsp.mzn p5.dzn
cost: 24   tour: [4, 1, 5, 3, 2]
----------
==========

Y:\public_html\cpM\minizincCPM\tsp>_
```

5 goes to 2




Command Prompt
Y:\public_html\cpM\minizincCPM\tsp>mzn-gecode tsp.mzn p5.dzn
cost: 24 tour: [4, 1, 5, 3, 2]

==========
Y:\public_html\cpM\minizincCPM\tsp>


Feic!