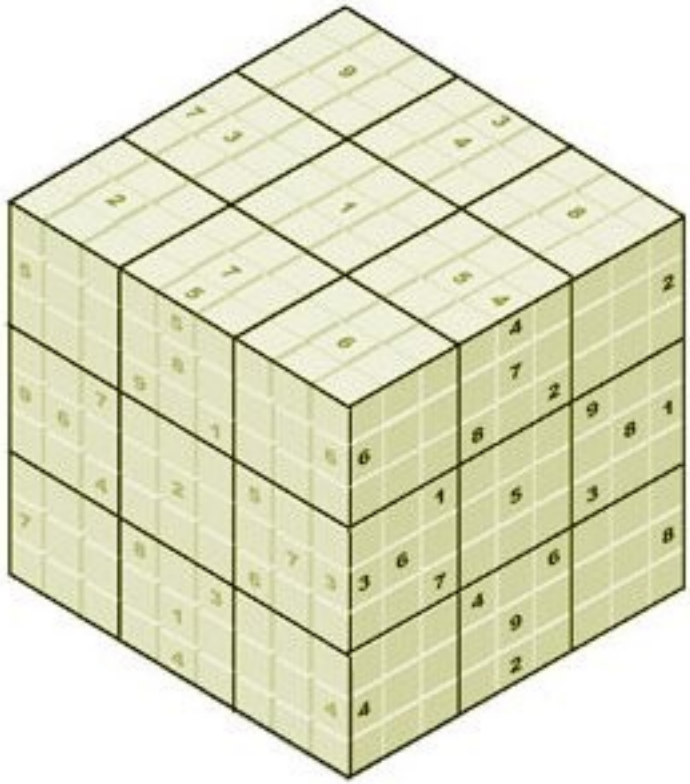
Parallelization of algorithms to solve a three dimensional Sudoku puzzle

Mikahil Mayorov and Paula A. Whitlock

The standard two-dimensional sudoku puzzle has been generalized to be played on the 6 faces of a cube. Each face is a complete two-dimensional puzzle with constraints applied to the edges of each face.

Three faces of a 3 dimensional puzzle



Simulated Annealing applied to solving a 3 D Sudoku puzzle

- Probability distribution of proposed solutions:

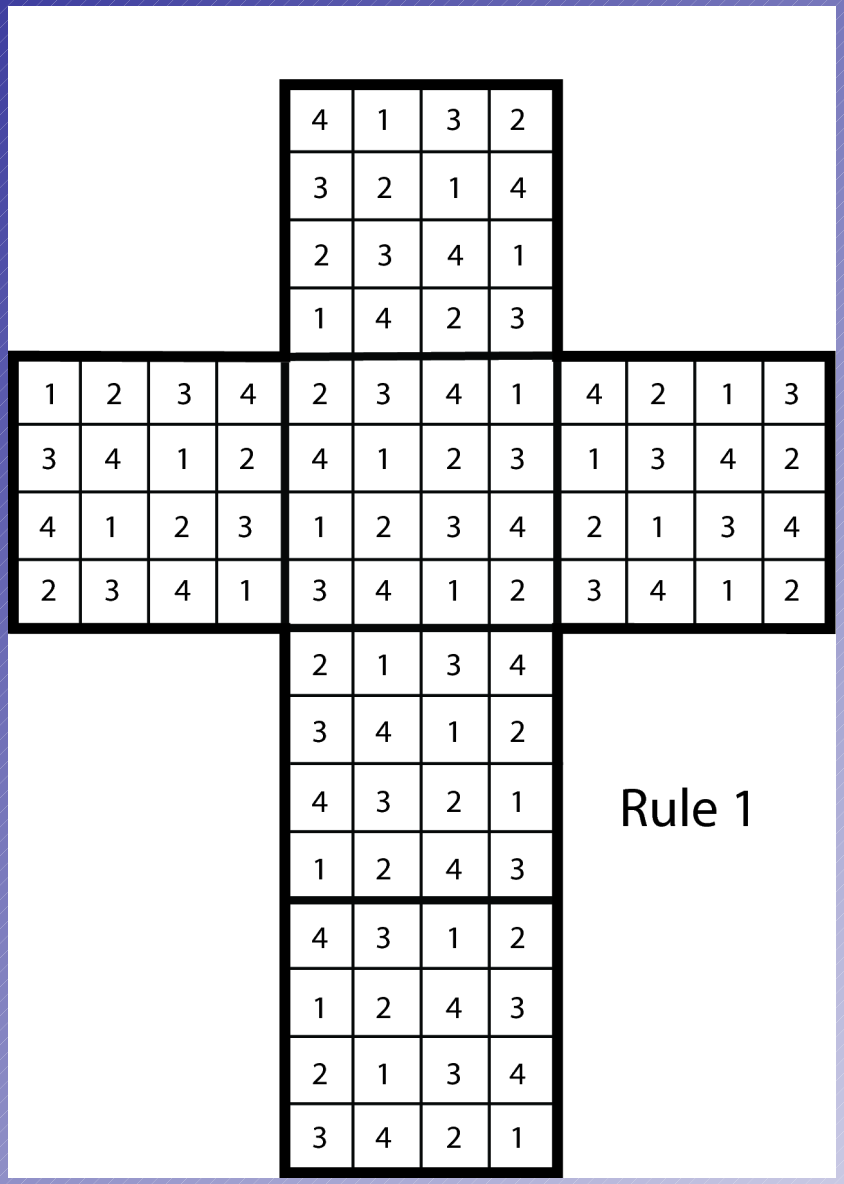
$$F(X) \propto \exp(-t * U(X))$$

  - t is labeled an inverse temperature, U(x) is the cost function.
  - A variation of the Metropolis Monte Carlo method is used to sample F(X).
  - The random walks begin at a high temperature and the temperature is reduced as the system converges to a solution.
  - The rate at which temperatures are reduced is the cooling schedule

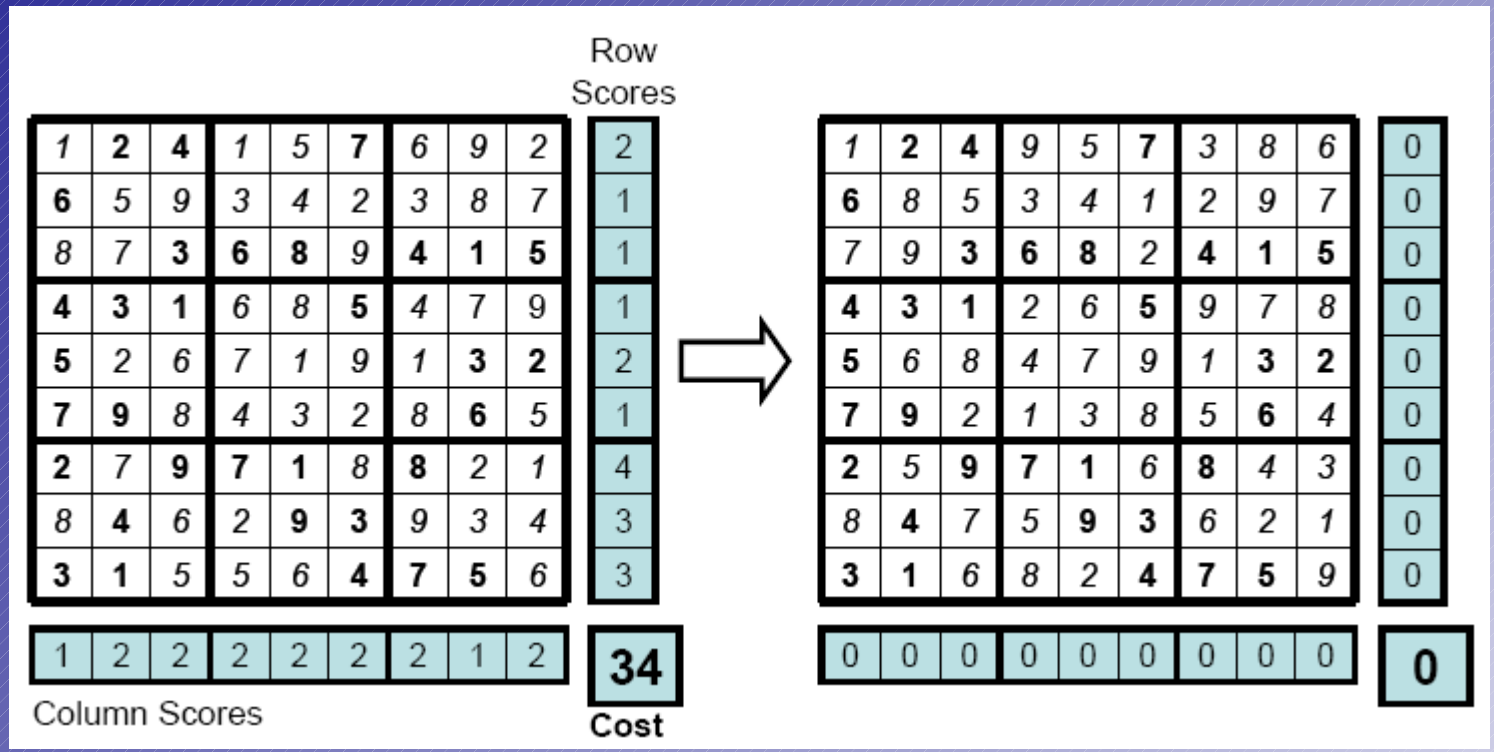
Simulated annealing algorithm

- Initially, the 6 faces are populated with random values in each cell.
- Either the cells on the edges match in value or they are chosen to be non-matching.
- The initial cost function is calculated:
  - Each missing value, $1 – n^2$, in a row or column is added to the cost function.
  - Cells on the edges of the faces that do not follow the appropriate rule add to the cost function.
- The goal is a cost of 0.

Edges of the faces do not match



Contributions to the cost function from one face of the puzzle



The neighborhood function

- The random walk proceeds by repeatedly applying a neighborhood function:
  - Two non-fixed cells within a block are chosen at random.
  - A possible move, x', that exchanges the cells is proposed.
  - The cost function is recalculated. If the exchange decreased the cost function, it is accepted
  - If the exchange increases the cost function, it is accepted with probability proportional to $\exp(-t*(U(x)-U(x'))$

Summary of algorithm

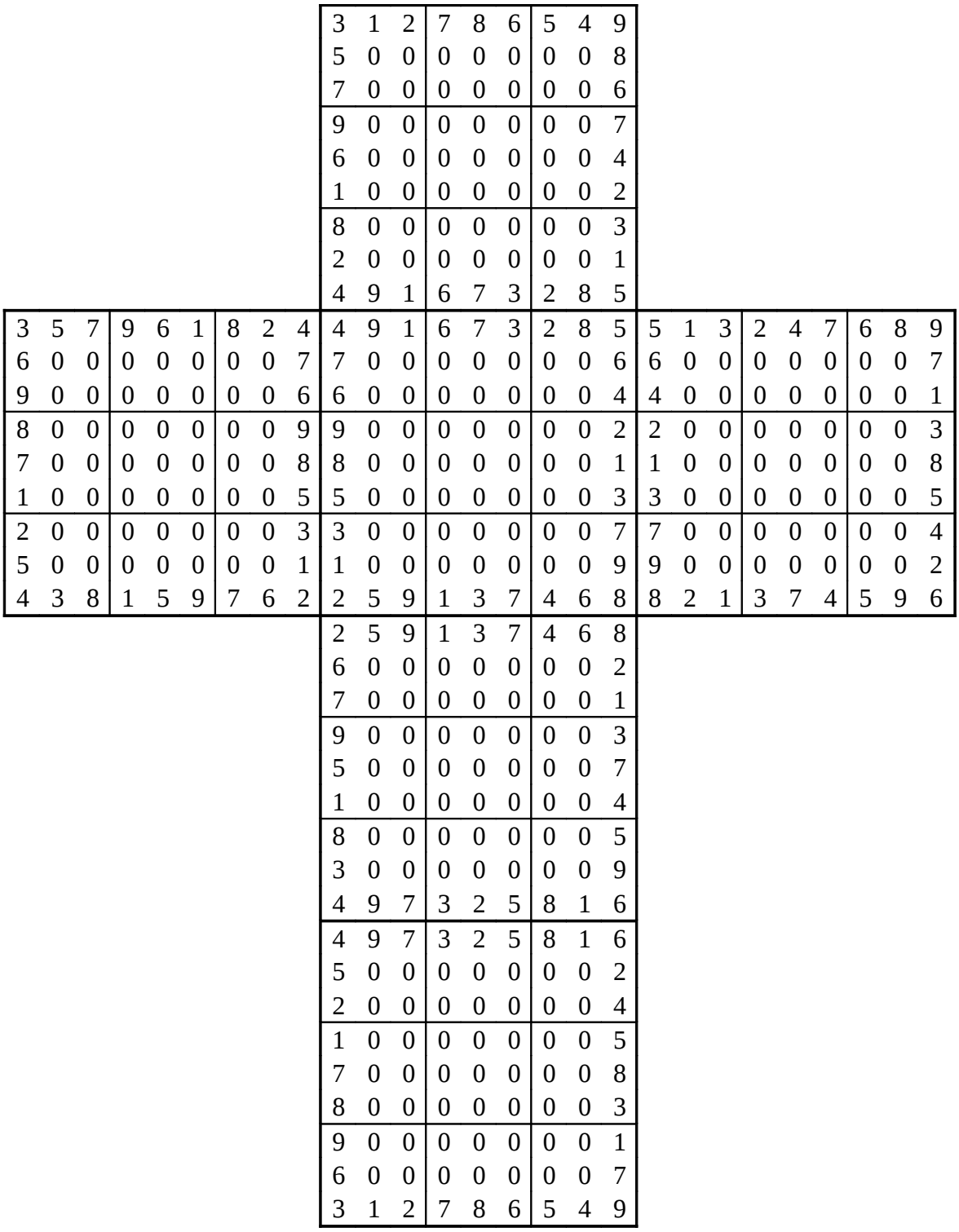
- A sequence of Markov chains, one per temperature visited, is formed.
- A Markov chain is composed of a series of proposed exchanges of non-fixed cells on a face.
- The Markov chain needs enough exchanges for the system to equilibrate at that temperature.
- When the cost function becomes 0, the simulation is terminated.
- If the temperature becomes too low, the simulation is restarted at a high temperature.

Independent processes on multiple computers solve the puzzle – method 1

- The edge cells are randomly and consistently populated.
- The internal cells of each face are randomly assigned values.
- Each process tries to solve face #0 by exchanging internal cells only.
- If no solution can be found after several reheats, two edge cells are exchanged on face #0. All other faces are changed consistently.

Parallelization method 1 (cont.)

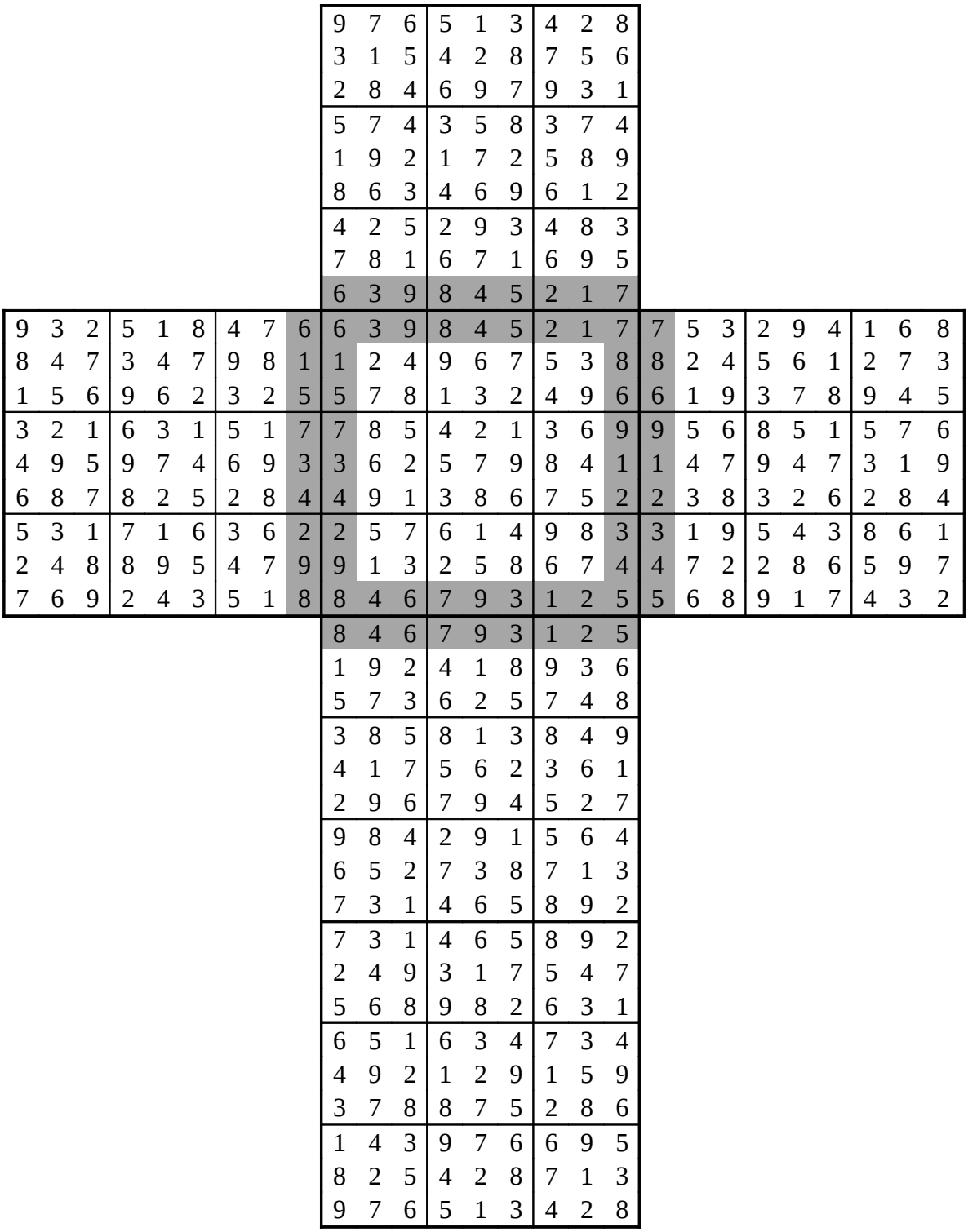
- Once face #0 is solved, all faces that border it have their edge cells fixed.
- The process continues the algorithm on another face which leads to more edge cells fixed.
- The last face has all its edge cells fixed and is solved very quickly.
- The first process to solve the puzzle sends a termination request to all the other processes.



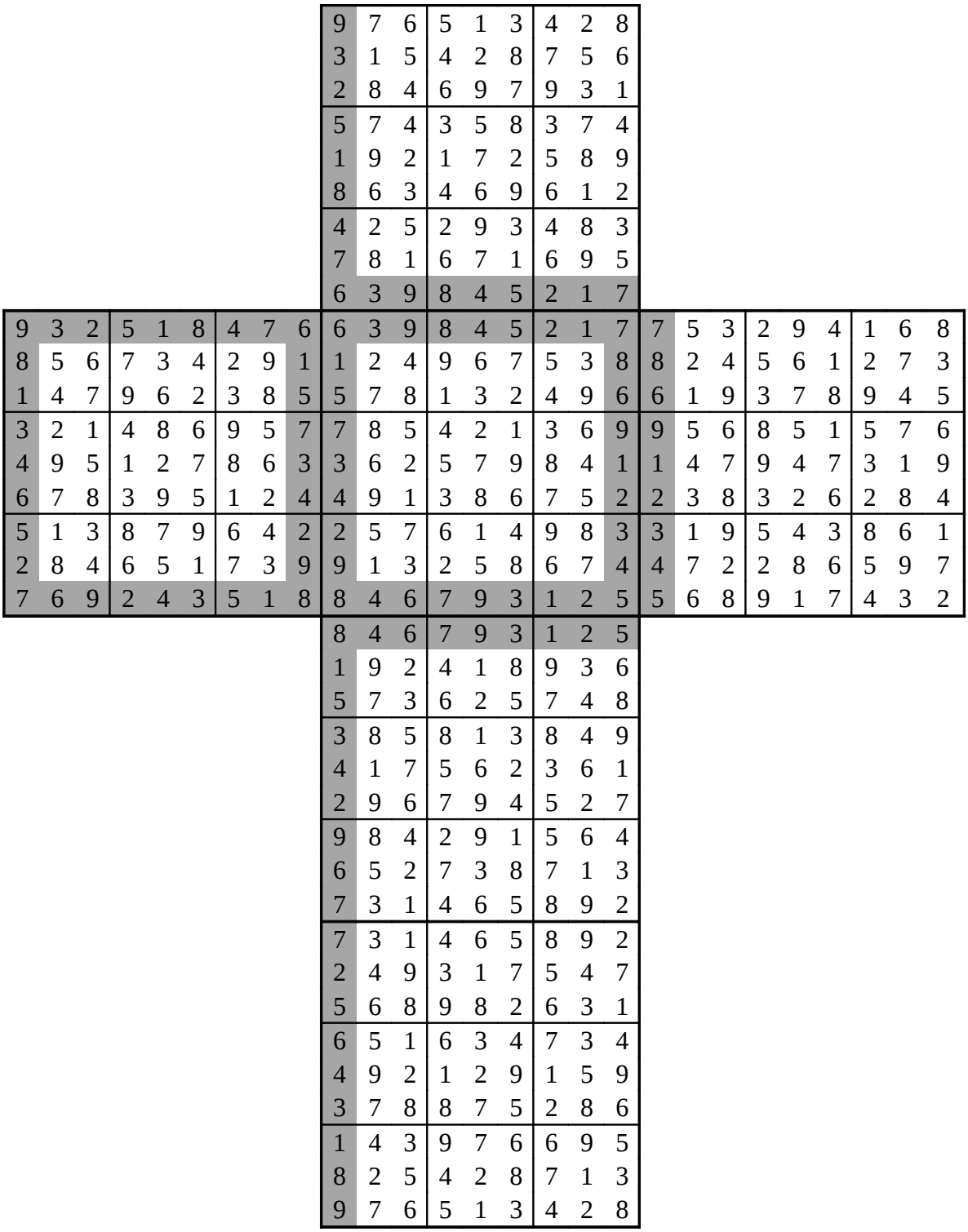
The edge cells are assigned values in a consistent manner. In this case, all edge cells match.

								9 7 3	5 4 6	1 2 8														
								6 1 5	1 2 8	7 5 3														
								2 8 4	3 9 7	9 6 4														
								5 7 4	3 5 8	3 7 1														
								1 9 2	1 7 2	5 8 9														
								8 6 3	4 6 9	6 4 2														
								4 2 5	8 4 3	9 2 6														
								7 1 9	6 7 1	3 8 5														
								3 6 8	2 9 5	4 1 7														
9 6 2	5 1 8	4 7 3	3 6 8	2 9 5	4 1 7	7 5 6	2 9 1	4 3 8																
8 1 7	3 4 7	1 8 9	9 1 7	3 1 7	6 9 8	8 1 4	5 6 4	2 7 6																
4 5 3	9 6 2	6 2 5	5 4 2	6 4 8	2 5 3	3 2 9	3 7 8	9 1 5																
6 2 4	6 3 1	5 4 7	7 4 3	6 7 9	7 8 2	2 5 6	8 5 1	5 7 3																
1 9 5	9 7 4	3 9 6	6 9 8	3 1 2	5 6 9	9 1 7	9 4 7	6 9 4																
3 8 7	8 2 5	2 8 1	1 2 5	4 5 8	1 3 4	4 3 8	3 2 6	2 8 1																
5 6 9	7 1 3	3 6 2	2 3 7	5 8 6	4 7 6	6 8 9	5 4 3	8 3 9																
2 4 1	8 4 5	9 7 8	8 5 1	9 4 2	9 3 1	1 7 2	2 1 6	5 4 7																
7 3 8	2 9 6	5 1 4	4 9 6	7 1 3	8 2 5	5 3 4	9 8 7	1 6 2																
								4 9 6	7 1 3	8 2 5														
								1 8 2	9 4 8	9 6 3														
								5 7 3	6 2 5	7 1 4														
								6 8 5	8 1 3	1 4 9														
								9 1 7	5 6 2	3 6 8														
								2 4 3	7 9 4	5 2 7														
								8 1 4	2 9 4	5 3 1														
								3 5 2	7 6 8	7 9 6														
								7 6 9	1 3 5	4 8 2														
								7 6 9	1 3 5	4 8 2														
								2 4 1	6 4 7	5 1 7														
								5 3 8	9 8 2	6 3 9														
								3 5 4	6 3 4	7 6 1														
								1 9 2	1 2 9	9 5 4														
								6 7 8	8 7 5	2 8 3														
								4 1 6	9 7 3	3 9 5														
								8 2 5	1 2 8	7 4 6														
								9 7 3	5 4 6	1 2 8														

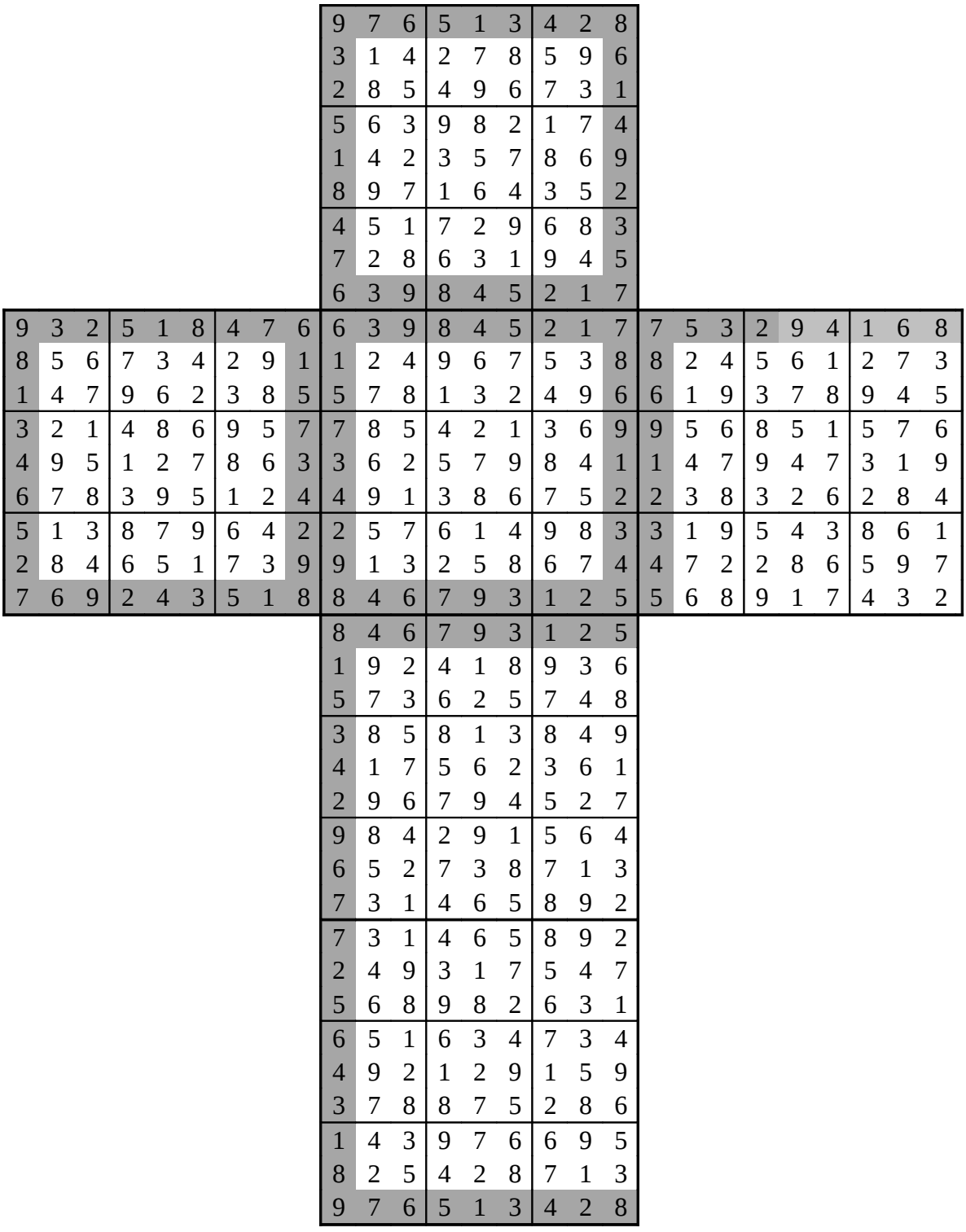
The internal cells are assigned values randomly but consistently.



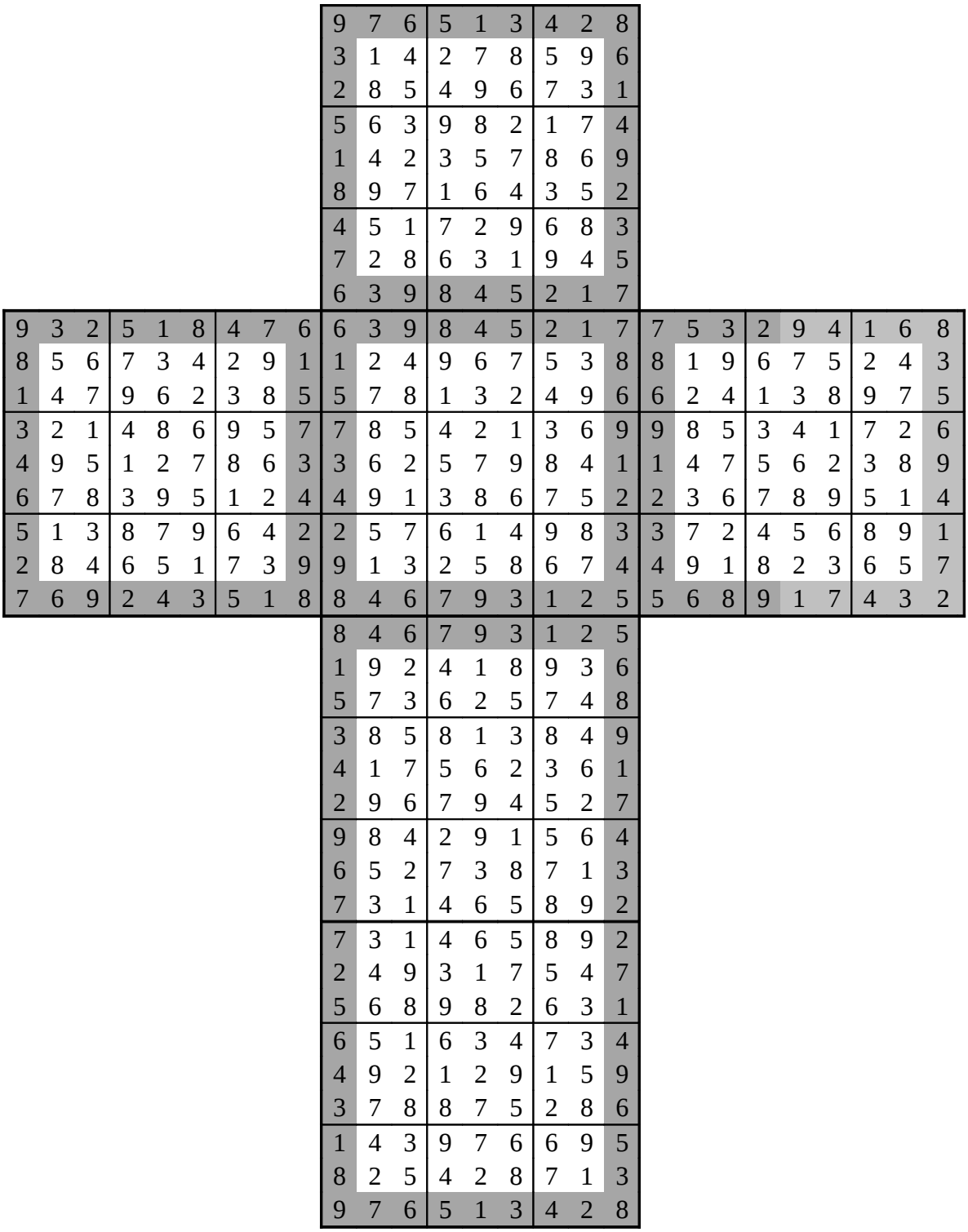
Face #0 is solved first. All edge cells of bordering faces are fixed to match the face #0 edge cells.



Face #1 has been solved and all edge cells bordering it have been fixed.



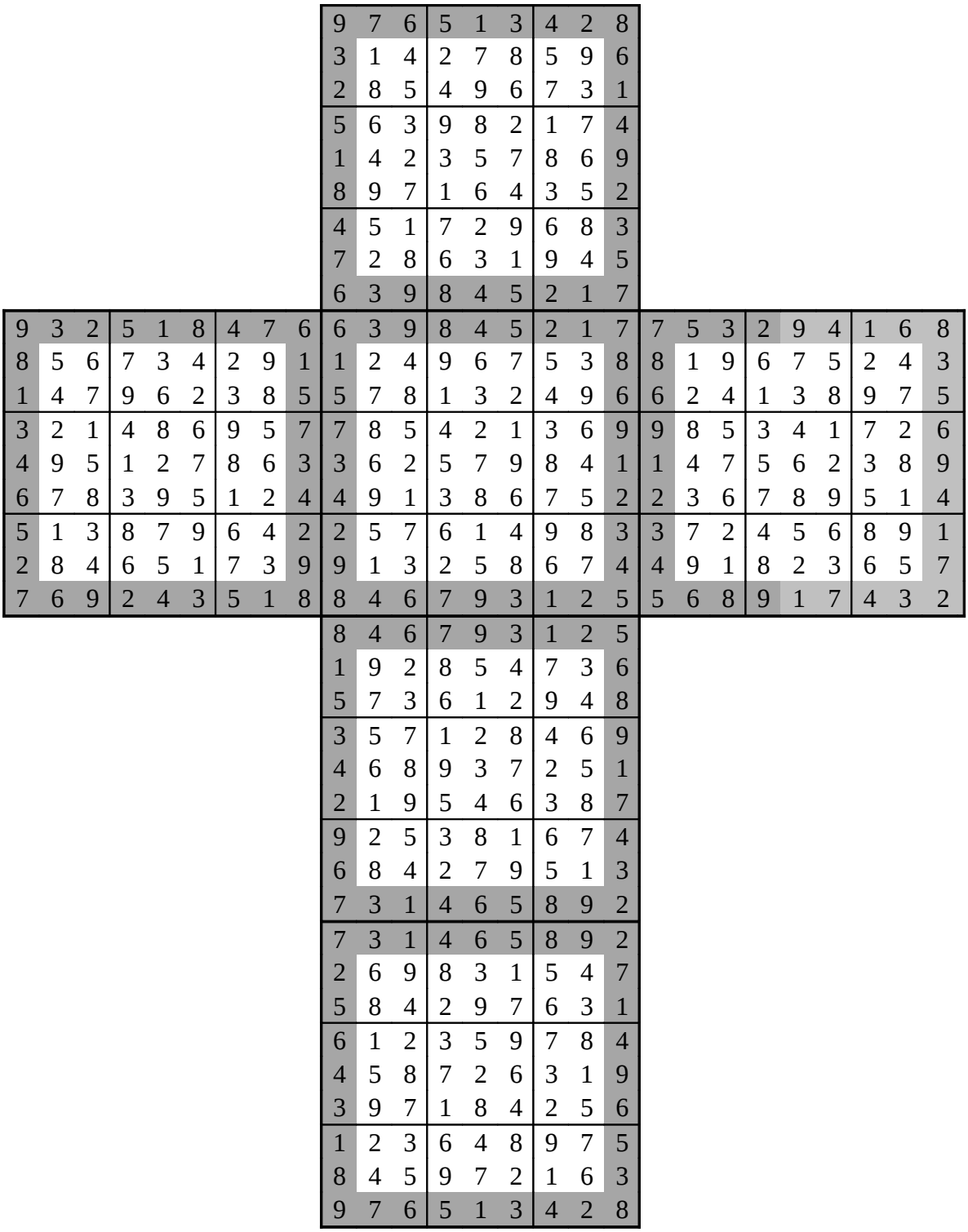
Face #2 is solved and edge cells on faces bordering it are fixed.



Face #3 has been solved. Note that all edge cells are now fixed cells.

								9	7	6	5	1	3	4	2	8								
								3	1	4	2	7	8	5	9	6								
								2	8	5	4	9	6	7	3	1								
								5	6	3	9	8	2	1	7	4								
								1	4	2	3	5	7	8	6	9								
								8	9	7	1	6	4	3	5	2								
								4	5	1	7	2	9	6	8	3								
								7	2	8	6	3	1	9	4	5								
								6	3	9	8	4	5	2	1	7								
9	3	2	5	1	8	4	7	6	6	3	9	8	4	5	2	1	7	7	5	3	2	9	4	1	6	8
8	5	6	7	3	4	2	9	1	1	2	4	9	6	7	5	3	8	8	1	9	6	7	5	2	4	3
1	4	7	9	6	2	3	8	5	5	7	8	1	3	2	4	9	6	6	2	4	1	3	8	9	7	5
3	2	1	4	8	6	9	5	7	7	8	5	4	2	1	3	6	9	9	8	5	3	4	1	7	2	6
4	9	5	1	2	7	8	6	3	3	6	2	5	7	9	8	4	1	1	4	7	5	6	2	3	8	9
6	7	8	3	9	5	1	2	4	4	9	1	3	8	6	7	5	2	2	3	6	7	8	9	5	1	4
5	1	3	8	7	9	6	4	2	2	5	7	6	1	4	9	8	3	3	7	2	4	5	6	8	9	1
2	8	4	6	5	1	7	3	9	9	1	3	2	5	8	6	7	4	4	9	1	8	2	3	6	5	7
7	6	9	2	4	3	5	1	8	8	4	6	7	9	3	1	2	5	5	6	8	9	1	7	4	3	2
								8	4	6	7	9	3	1	2	5								
								1	9	2	8	5	4	7	3	6								
								5	7	3	6	1	2	9	4	8								
								3	5	7	1	2	8	4	6	9								
								4	6	8	9	3	7	2	5	1								
								2	1	9	5	4	6	3	8	7								
								9	2	5	3	8	1	6	7	4								
								6	8	4	2	7	9	5	1	3								
								7	3	1	4	6	5	8	9	2								
								7	3	1	4	6	5	8	9	2								
								2	4	9	3	1	7	5	4	7								
								5	6	8	9	8	2	6	3	1								
								6	5	1	6	3	4	7	3	4								
								4	9	2	1	2	9	1	5	9								
								3	7	8	8	7	5	2	8	6								
								1	4	3	9	7	6	6	9	5								
								8	2	5	4	2	8	7	1	3								
								9	7	6	5	1	3	4	2	8								

Face #4 is solved.



Face #5 has been solved and the puzzle is complete.

Sample results for method 1

Network: 86 nodes, each node has 2 Intel 2.80 GHz Woodcrest dual-core processors

Puzzles dimension: 6 faces, each 9x9
Initial number of fixed cells: 0
Initial temperature: 0.1
Cooling rate: 0.99
Number of CPUs/run: 10
Number of runs: 10
Edge cells: match

average time (seconds): 0.444
standard deviation: 0.09

Master/worker distributed code – method 2

- The master process, process 0,
  - Assigns values to the edge cells
  - Assigns values to the internal cells of each face
  - Distributes face #0 to all participating processes
- The first process to solve face #0, returns it to the master process.
  - The master process fixes the edge cells for all faces based on the results from face #0.
  - The master process distributes the five remaining faces to the other processes.
- When a process solves a face, it returns it to process 0

Method 2 (cont.)

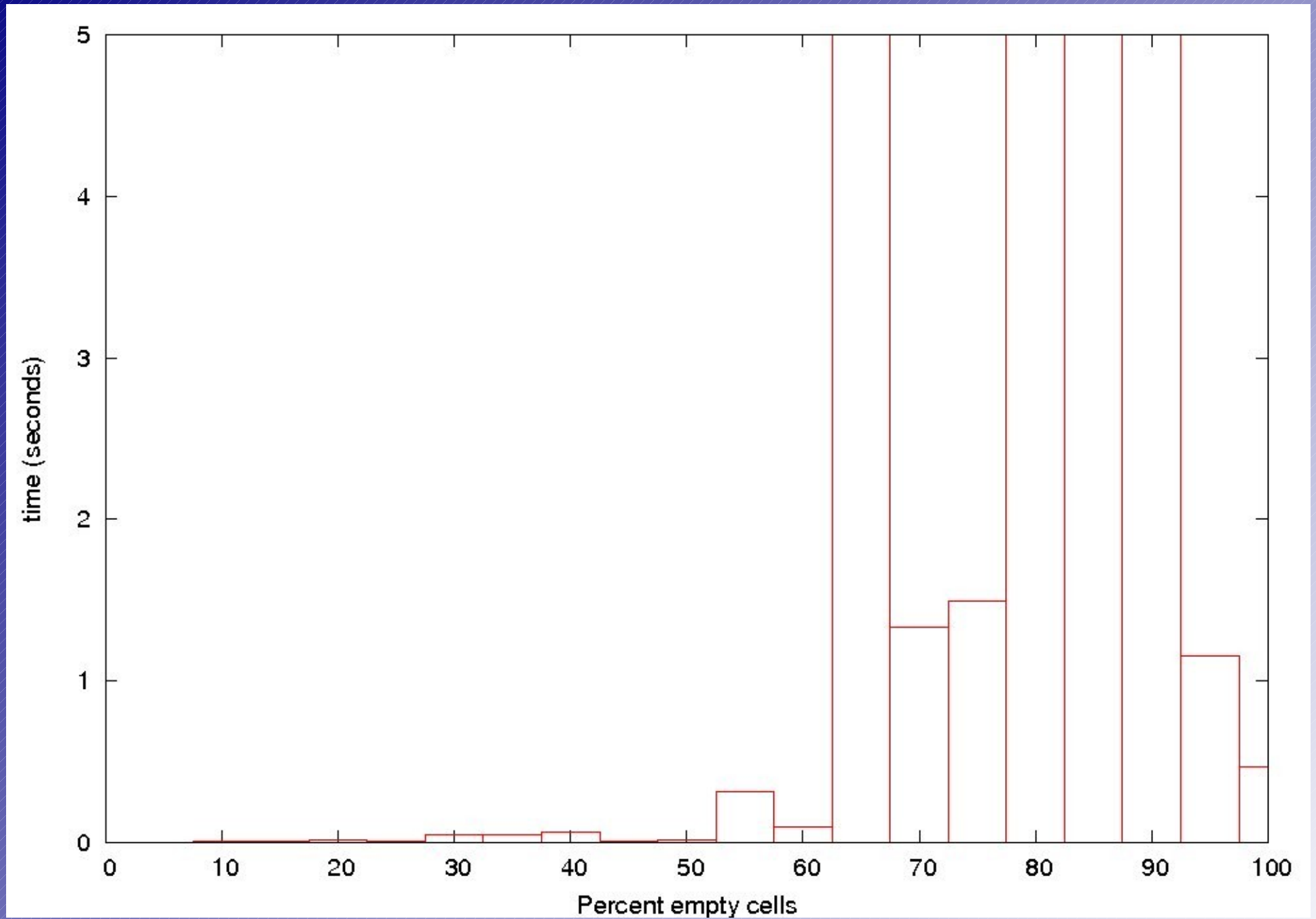
- When all five faces have been returned, the puzzle is solved and the process terminates.

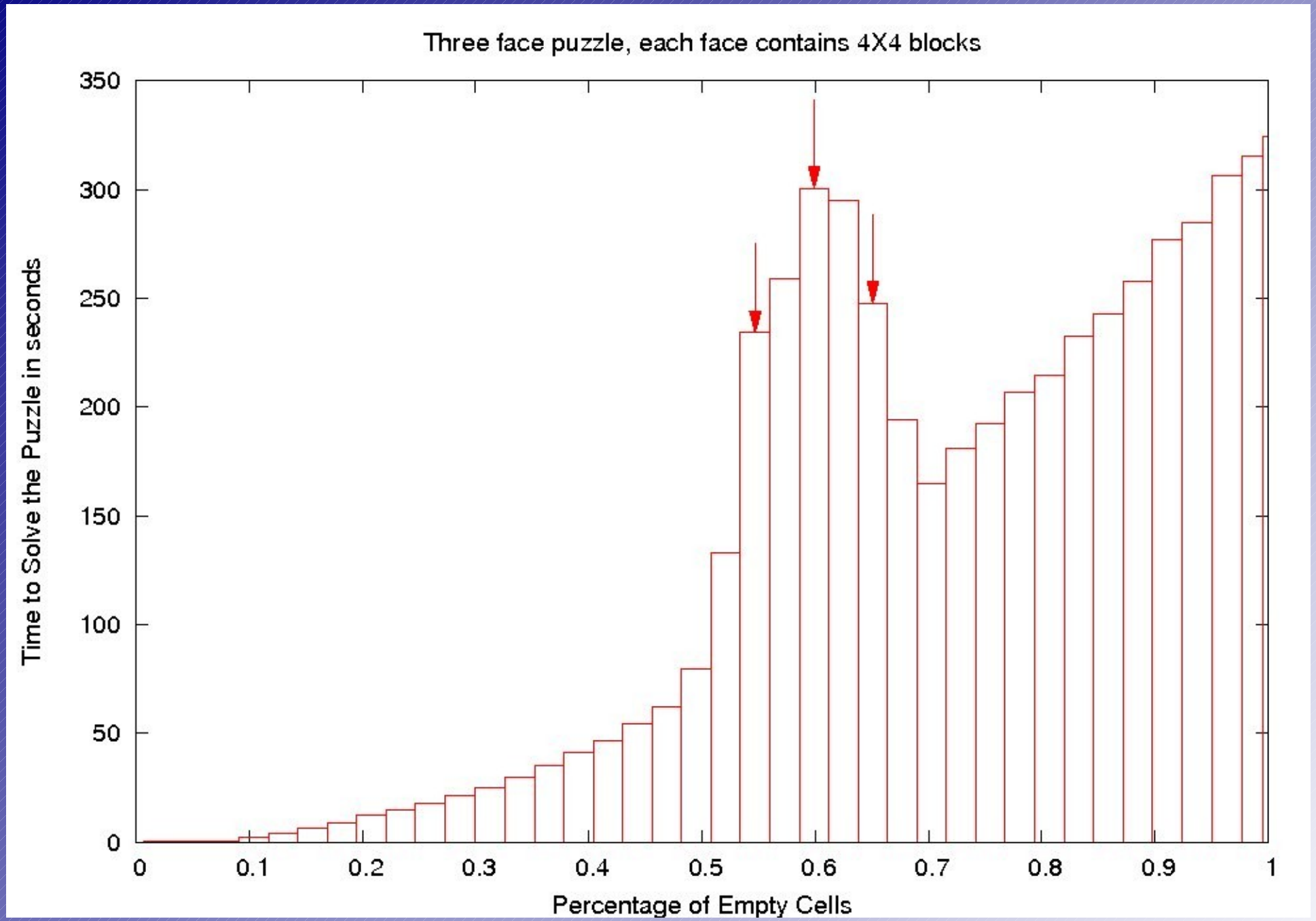
Results:
Puzzles dimension: 6 faces, each 9x9
Initial number of fixed cells: 0
Initial temperature: 0.1
Cooling rate: 0.99
Number of CPUs/run: 10
Number of runs: 10
Edge cells: match

average time (seconds): 0.11
standard deviation: 0.02

Characteristics of Distributed Simulated Annealing Method Code

- The case where the edge cells must match is solved faster than the case where the edge cells do not match
- As the number of fixed cells in the puzzle decreases, the time to solve the puzzle increases.
- The distributed code does not scale well
  - Using 20 processors does not run any faster than using 10 or 6 processors.
  - The fastest timings are achieved on this network using 2 cores on each of three processors.


time (seconds)
0
1
2
3
4
5
0
10
20
30
40
50
60
70
80
90
100
Percent empty cells


Three face puzzle, each face contains 4X4 blocks
Time to Solve the Puzzle in seconds
Percentage of Empty Cells
0
50
100
150
200
250
300
350
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

How the six face puzzle behaves as the number of fixed cells changes

Percent of empty cells in a Sudoku puzzle			
Constraint:s	For a unique puzzle	For degenerate solutions	No solution found
Edge cells must match	0 - 49%	50% - 100%	65 %- 95%
Edge cells do not match	0 - 29%	30% - 100%	35% - 90%

What is needed for a three-dimensional Sudoku puzzle to be playable

- The puzzle should have a unique solution
- The solving of the puzzle should use a chain of reasoning to reach the unique solution.
- The player should be able to geometrically envision the game – an on-line version where the player can rotate the cube may be easier to play than a flat representation.
- Easier to solve puzzles will have edge cells that match.