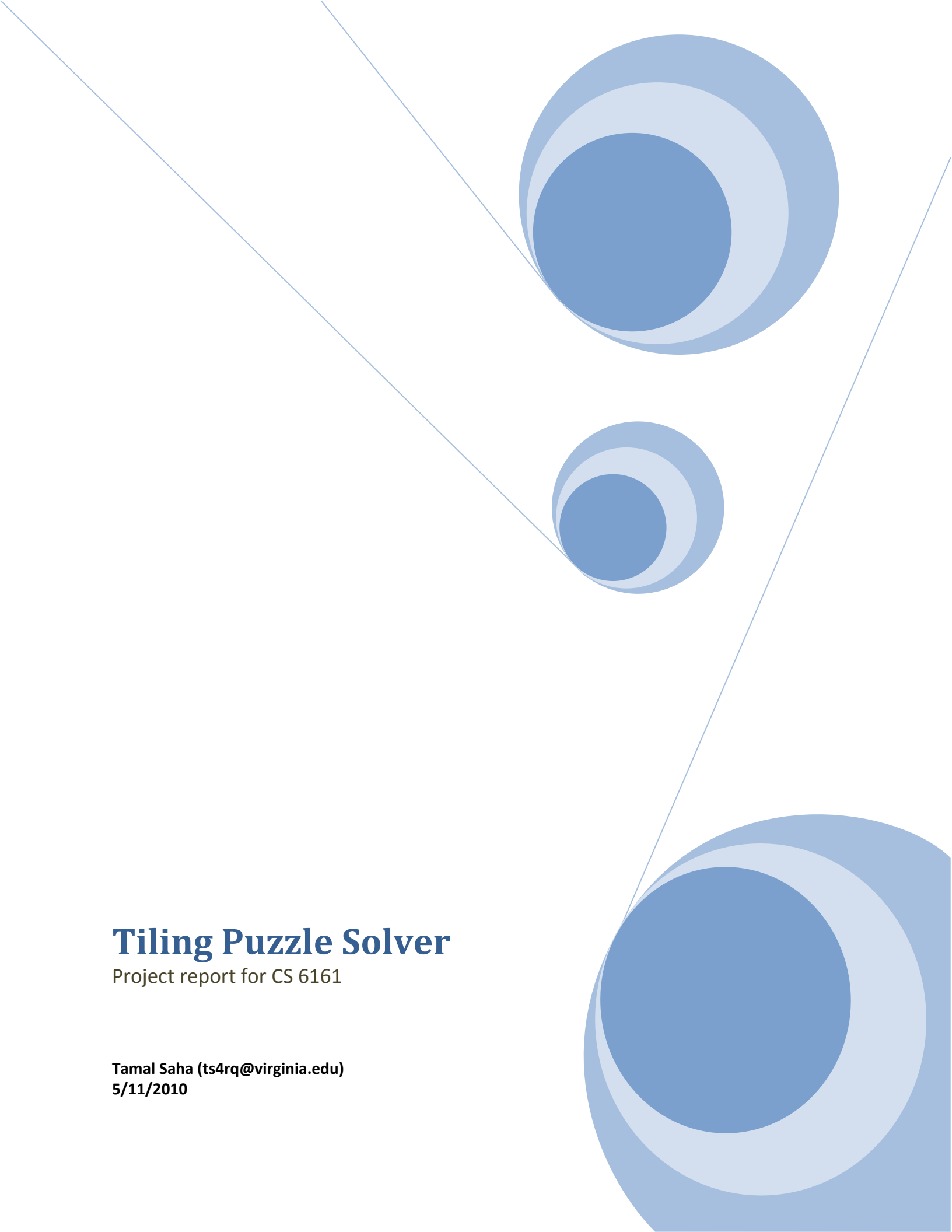


Tiling Puzzle Solver

Project report for CS 6161

Tamal Saha (ts4rq@virginia.edu)
5/11/2010

Introduction:

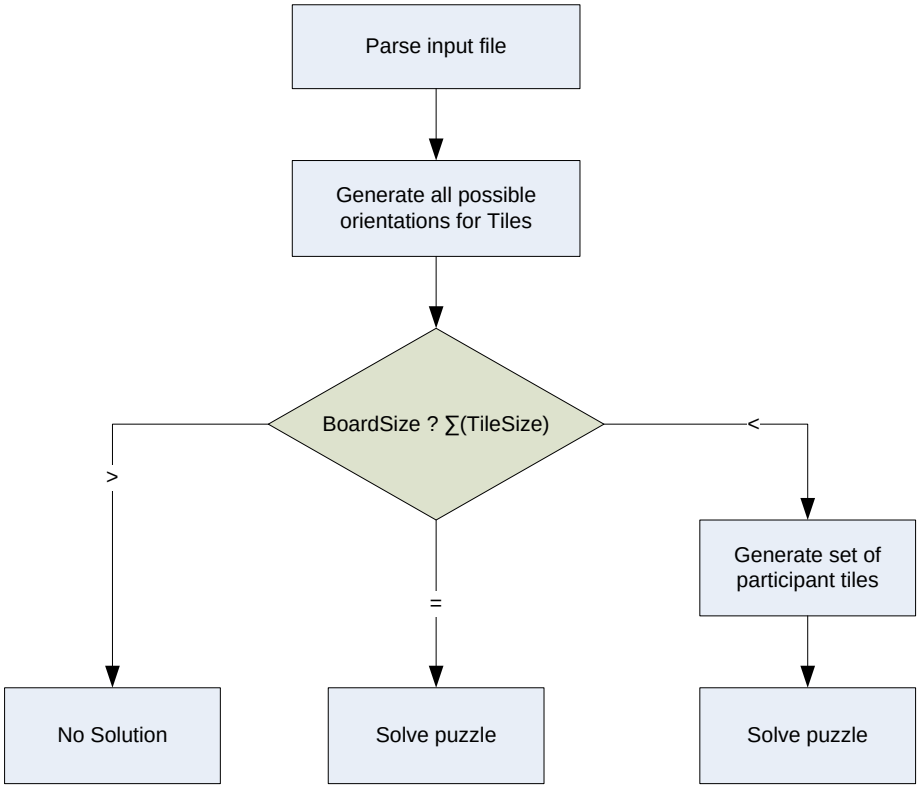
This report documents the flowchart of the algorithm and bounding conditions used for the tiling puzzle solver, some screenshots of the program and a detailed execution time statistics for the given puzzles and details of the system where this program was developed.

Algorithm:

The puzzle solver finds solutions for the puzzle using branch and bound algorithm, which in other words means backtracking algorithm.

Flowchart:

The following diagram shows the high level flowchart for the puzzle solver.



Bounding conditions:

This algorithm uses different bounding condition based on the puzzle.

- If all the tiles of the puzzle are of same size, then if during any stage of the branching process the unfilled holes in the board are not multiple of a tile size, then a dead end is detected.

- If all the tiles of the puzzle are not of same size, then dead end is detected as follows:

    - The smallest unfilled hole in the board in smaller than the remaining unused tile.

    - The smallest unfilled hole is of same size as the smallest remaining unused tile but the number of such holes in greater than the number of remaining tiles. As a result, all the unfilled holes can't be filled by any unused tiles.

    - The largest unfilled hole in the board is smaller than the largest remaining unused tile.

    - The largest unfilled hole in the board is of same size as the largest remaining unused tile but the number of tiles is greater than the number of such holes. As a result, some of the tiles can't be placed anywhere in the board.

Implementation Details:

- Processor: Intel Dual Core 1.6 GHz
- RAM: 2.5 GB
- Programming Language: Java
- JDK version: 1.6.0_20
- IDE: Eclipse 3.5
- GUI Toolkit: SWT

Screenshots:

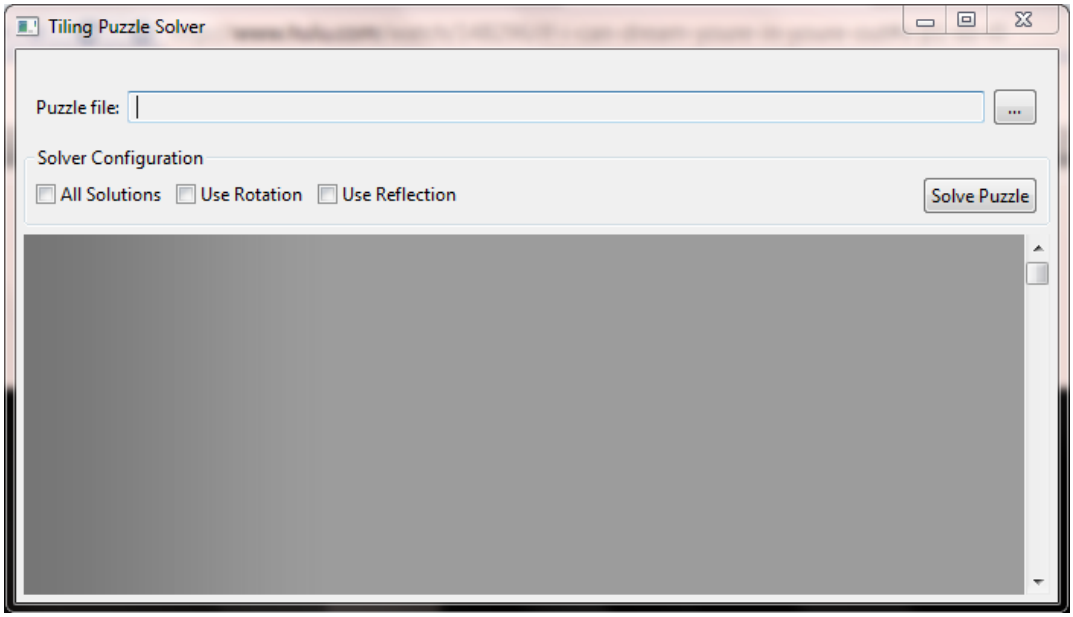


Figure: GUI after start up

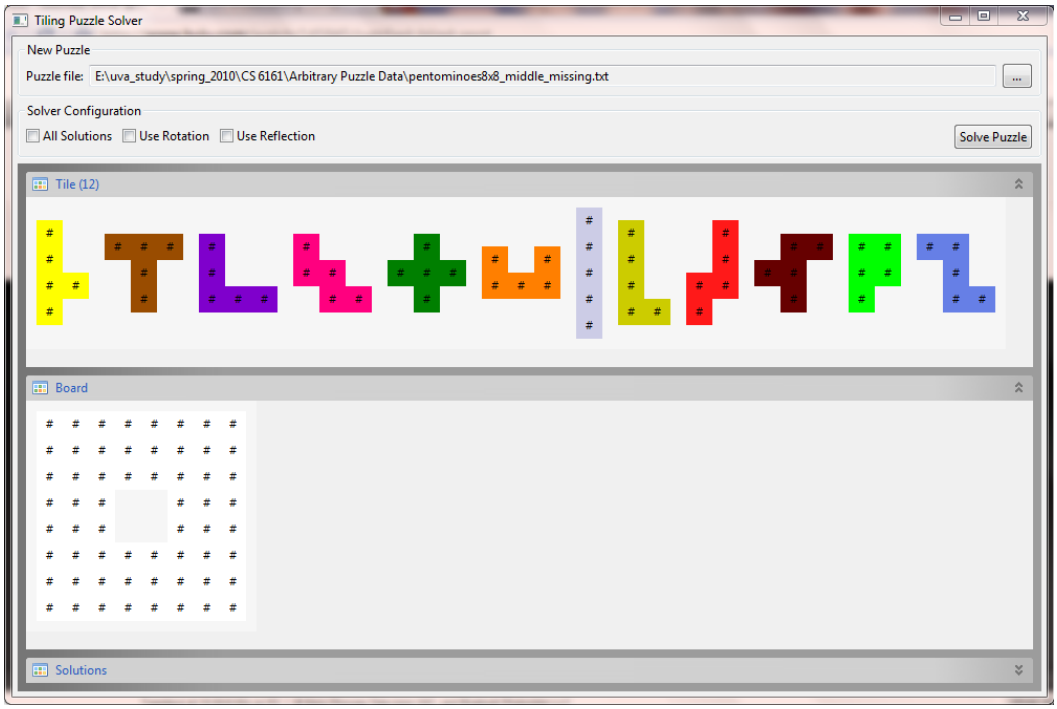


Figure: A puzzle is loaded from file

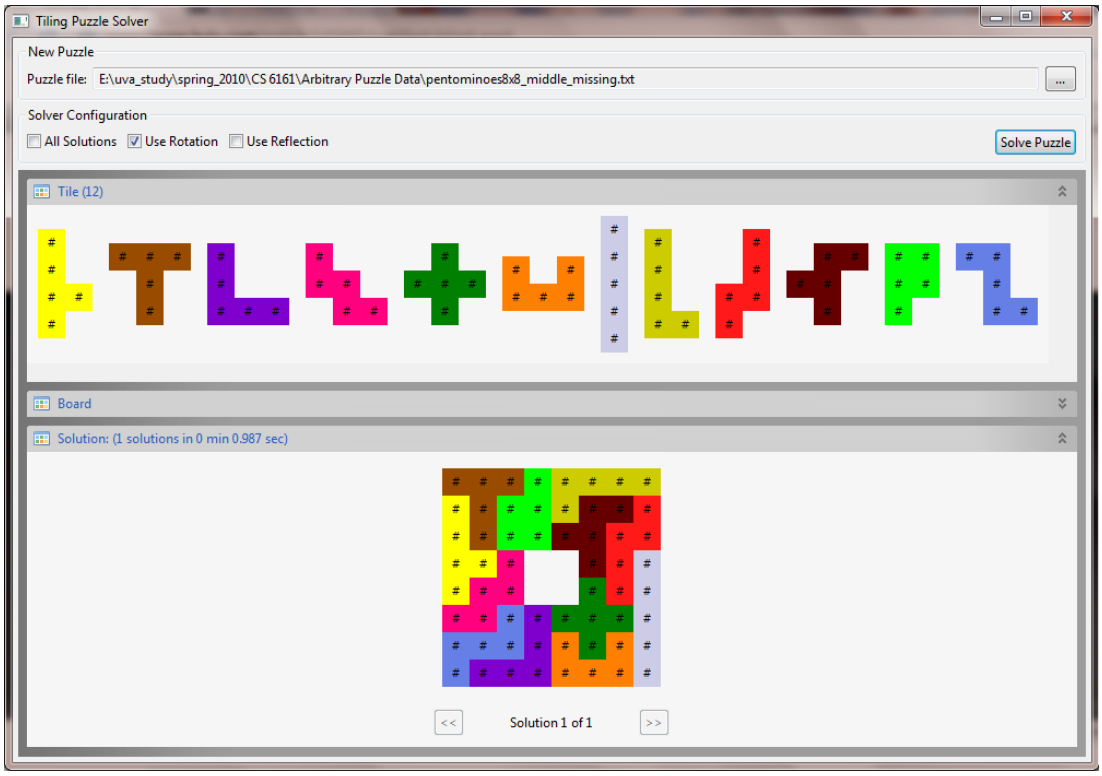


Figure: After puzzle solved

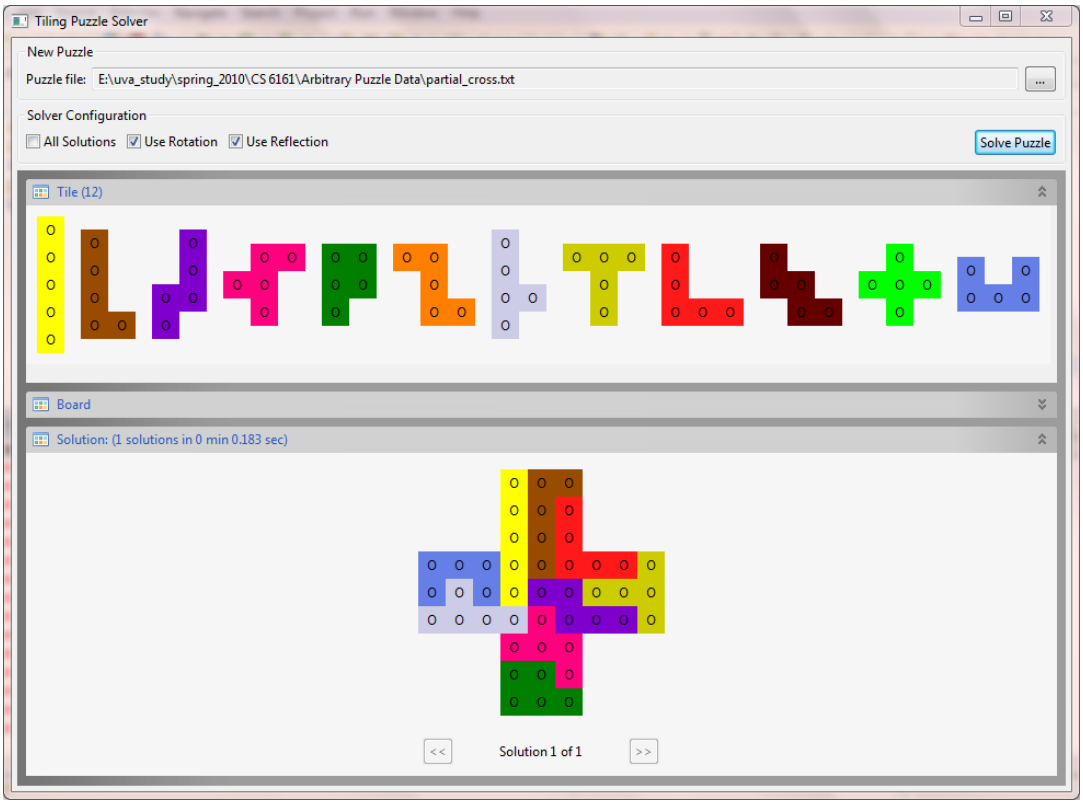


Figure: Another puzzle solved

Execution time statistics:

Without Rotation and Reflection:

Name	Single Solution		All Solutions			
			Before†		After*	
	#	t (ms)	#	t (ms)	#	t (ms)
Checkerboard	1	30	1	17	1	17
IQ_creator	1	0	1	1	1	1
partial_cross	0	0	0	0	0	1
pentominoes3x20	0	0	0	1	0	1
pentominoes4x15	0	4	0	5	0	5
pentominoes5x12	0	24	0	24	0	24
pentominoes6x10	0	15	0	15	0	15
pentominoes8x8_corner_missing	0	10	0	11	0	11
pentominoes8x8_four_missing_corners	0	33	0	42	0	42
pentominoes8x8_four_missing_diagonal	0	4	0	3	0	3
pentominoes8x8_four_missing_near_corners	0	4	0	12	0	12
pentominoes8x8_four_missing_near_middle	0	1	0	2	0	2
pentominoes8x8_four_missing_offset_near_corners	0	6	0	6	0	6
pentominoes8x8_four_missing_offset_near_middle	0	3	0	2	0	2
pentominoes8x8_middle_missing	0	20	0	6	0	6
pentominoes8x8_side_missing	0	1	0	1	0	1
pentominoes8x8_side_offset_missing	0	5	0	5	0	5
thirteen_holes	0	0	0	1	0	1
trivial	0	0	0	0	0	0

Before† : Before symmetric solutions are removed

After* : After symmetric solutions are removed

With Rotation and without Reflection:

Name	Single Solution		All Solutions			
			Before†		After*	
	#	t (ms)	#	t (ms)	#	t (ms)
Checkerboard	1	30	1	17	1	17
IQ_creator	1	0	8	69	2	69
partial_cross	0	37	0	35	0	35
pentominoes3x20	0	9553	0	9469	0	9469
pentominoes4x15	1	1753	16	105032	8	105032
pentominoes5x12	1	15294	46	132111	23	132111
pentominoes6x10	1	418	106	67809	53	67811
pentominoes8x8_corner_missing	1	393	130	19060	130	19065
pentominoes8x8_four_missing_corners	1	302	192	34392	48	34400
pentominoes8x8_four_missing_diagonal	0	1920	0	1785	0	1785
pentominoes8x8_four_missing_near_corners	1	826	20	12809	5	12809
pentominoes8x8_four_missing_near_middle	0	2555	0	2696	0	2696
pentominoes8x8_four_missing_offset_near_corners	0	3901	0	2966	0	2966
pentominoes8x8_four_missing_offset_near_middle	1	802	4	2904	1	2904
pentominoes8x8_middle_missing	1	529	4	4914	1	4914
pentominoes8x8_side_missing	1	244	27	7500	27	7500
pentominoes8x8_side_offset_missing	1	435	52	5459	52	5460
thirteen_holes	0	82	0	80	0	80
trivial	1	0	1	0	1	0

Before† : Before symmetric solutions are removed

After* : After symmetric solutions are removed

With Rotation and Reflection:

Name	Single Solution		All Solutions			
			Before†		After*	
	#	t (ms)	#	t (ms)	#	t (ms)
Checkerboard	1	30	1	17	1	17
IQ_creator	1	0	48	294	6	294
partial_cross	1	3	160	197	20	201
pentominoes3x20	1	26101	8	133327	2	133327
pentominoes4x15	1	11533	1472	1837736	368	1837847
pentominoes5x12	1	3631	4040	2551105	1010	2552015
pentominoes6x10	1	1853	9356	1685426	2339	1691408
pentominoes8x8_corner_missing	1	88	10054	670753	5027	716294
pentominoes8x8_four_missing_corners	1	98	17360	970408	2170	1025150
pentominoes8x8_four_missing_diagonal	1	96	296	20797	74	20807
pentominoes8x8_four_missing_near_corners	1	381	1504	257987	188	258229
pentominoes8x8_four_missing_near_middle	1	54	168	50888	21	50890
pentominoes8x8_four_missing_offset_near_corners	1	535	216	77757	54	77763
pentominoes8x8_four_missing_offset_near_middle	1	377	504	44449	126	44487
pentominoes8x8_middle_missing	1	150	520	87124	65	87137
pentominoes8x8_side_missing	1	52	2576	170206	1288	171577
pentominoes8x8_side_offset_missing	1	7	1839	127903	1839	129278
thirteen_holes	1	281	8	719	2	719
trivial	1	0	1	0	1	0

Before† : Before symmetric solutions are removed

After* : After symmetric solutions are removed

Website:

All source code and report for this project is hosted on http://code.google.com/p/fasttiles/