8 Queens Puzzle

This project is a visualization program of the *8 Queens Puzzle* (https://en.wikipedia.org/wiki/Eight_queens_puzzle). It is a faculty project at Slovak University of Technology Faculty of Informatics and Information Technologies (STU FIIT) Bratislava.

Acknowledgement

David Barr's `PixelGameEngine` (https://github.com/OneLoneCoder/olcPixelGameEngine) was used for rendering. Icons (https://devilsworkshop.itch.io/pixel-art-chess-asset-pack) are by Ajay Karat | Devil's Work.shop http://devilswork.shop/.

About- 8 Queens Puzzle

8 Queens Puzzle consists of **N x N** board, where each column is occupied by a single Queen (chess figure). Goal is to place figures so there is no endangerment between them. Canonically, this problem is solved with various search algorithms. This specific implementation includes solutions with Beam search (https://en.wikipedia.org/wiki/Beam_search) and Tabu search (https://en.wikipedia.org/wiki/Tabu_search) algorithms.

Platform	Language	Compiler
Windows	C++	Visual Studio Compiler

Search algorithms

Beam search

Beam Search is a modified version of the Breadth-first search (https://en.wikipedia.org/wiki/Breadth-first_search). In the Beam search, each iteration instead of expanding every node of the traversal tree, only predetermined ***k*** nodes are expanded, where each node is selected based on some heuristic. Before the traversal begins, queue is populated with randomly generated ***k*** nodes. The ***k*** factor is the key (*pun intended*) of the

algorithm. Well balanced value of ***k*** will produce the most optimal result, whereas too small or too big value is more likely to result in a failure of the algorithm, since it is not guaranteed to find a correct solution.

Tabu search

Tabu search is a local heuristic search algorithm that attempts to optimize its state by keeping track of states that lead to dead-ends. Similarly to the Beam search, Tabu search has a ***n*** parameter, which is the maximum size of the tabu list. Each iteration algorithm selects the best fitting candidate from the available set of states. If candidate leads to a dead-end, then it is added to the tabu list and the next best candidate is selected to continue the search. If the maximum size of the tabu list is reached, then the oldest state is released from the list. Maximum size parameter is a critical factor, since it determines how long it takes for the algorithm to get out of the local extreme situations.

Heuristics

Global threats

Global threats heuristic is a total amount of threats in the current state. Threats are counted from left to right, ignoring already counted ones.

Local threats

Local threats heuristic is an amount of threats for the given figure on the board. This heuristic is less informative then the Global threats, thus it proved to be less efficient way of determining the fitness of the candidates for the future states.

Node structure

Each state node is represented as a structure:

```
struct SearchState {
  std::vector<olc::vi2d> figuresPositions;
  uint32_t heuristicValue;
}
```

- `figuresPositions` - array of 2D position vectors of individual figures on the board;
- `heuristicValue` - heuristic value (fitness) of the state.

Results

Below is a table with some testing results:

Beam search

Beam width	Heuristic type	Board size	*Iterations count	Failed iterations	Avg. duration	Avg. nodes count
8	Local threats	8 x 8	10000	3405	3096 μs	1665
8	Global threats	8 x 8	10000	3563	1105 μs	1670

16	Local threats	8 x 8	10000	709	5681 *µs*	3126
16	Global threats	8 x 8	10000	674	1952 *µs*	3137
32	Local threats	8 x 8	10000	21	10145 *µs*	5500
32	Global threats	8 x 8	10000	26	3277 *µs*	5487
48	Local threats	8 x 8	10000	0	14915 *µs*	7622
48	Global threats	8 x 8	10000	1	4454 *µs*	7611
48	Local threats	16 x 16	1000	1	234110 *µs*	71603
48	Global threats	16 x 16	1000	3	83167 *µs*	71503
64	Local threats	16 x 16	1000	0	305245 *µs*	92421
64	Global threats	16 x 16	1000	0	106990 *µs*	91374

Tabu search

Tabu list max size	Heuristic type	Board size	*Iterations count	Failed iterations	Avg. duration	Avg. nodes count
8	Local threats	8 x 8	10000	0	6481 *µs*	2618
8	Global threats	8 x 8	10000	0	3734 *µs*	2587
16	Local threats	8 x 8	10000	1	5418 *µs*	2302
16	Global threats	8 x 8	10000	0	3585 *µs*	2363
32	Local threats	8 x 8	10000	0	5158 *µs*	2159
32	Global threats	8 x 8	10000	0	3018 *µs*	2094
64	Local threats	8 x 8	10000	0	5001 *µs*	2090
64	Global threats	8 x 8	10000	0	3139 *µs*	2058
64	Local threats	16 x 16	1000	0	92554 *µs*	21470
64	Global threats	16 x 16	1000	0	63238 *µs*	21238
128	Local threats	16 x 16	1000	0	76764 *µs*	18351
128	Global threats	16 x 16	1000	0	56112 *µs*	20323
256	Local threats	16 x 16	1000	0	82747 *µs*	20310
256	Global threats	16 x 16	1000	0	57262 *µs*	19491

*Iterations count is the amount of ran tests for the given configuration.

Tests were conducted on a laptop with these parameters:

Processor	RAM	OS
Intel Core i7-7700HQ 2.80GHz	16 GB	Windows 10 Home (64bit)

Conclusion

C++ is fun.

References

- 8 Queens Puzzle, wikipedia.org (https://en.wikipedia.org/wiki/Eight_queens_puzzle)
- Tabu search, wikipedia.org (https://en.wikipedia.org/wiki/Tabu_search)
- Beam search, wikipedia.org (https://en.wikipedia.org/wiki/Beam_search)
- Breadth-first search, wikipedia.org (https://en.wikipedia.org/wiki/Breadth-first_search)
- PixelGameEngine, github.com (https://github.com/OneLoneCoder/olcPixelGameEngine)
- Pixel Art Chess Asset Pack, devilsworkshop.itch.io (https://devilsworkshop.itch.io/pixel-art-chess-asset-pack)