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:

Search type	*k* / *n* param	Heuristic type	Board size	*Iterations count	Failed iterations	Average duration	Average nodes count
Beam search	8	Global threats	8 x 8	2586	1586	37ms	1702
Beam search	8	Local threats	8 x 8	2626	1626	212ms	1701
Beam	16	Global	8 x 8	1115	115	71ms	3196

search		threats					
Beam search	16	Local threats	8 x 8	1081	81	411ms	3216
Beam search	32	Global threats	8 x 8	1000	0	123ms	5498
Beam search	32	Local threats	8 x 8	1000	0	740ms	5731
Beam search	32	Global threats	16 x 16	105	5	1127ms	49632
Beam search	64	Global threats	16 x 16	100	0	2033ms	92784
Beam search	128	Global threats	16 x 16	100	0	3634ms	172154
Tabu search	8	Global threats	8 x 8	1000	0	110ms	2578
Tabu search	8	Local threats	8 x 8	1000	0	325ms	2800
Tabu search	16	Global threats	8 x 8	1000	0	97ms	2255
Tabu search	16	Local threats	8 x 8	1000	0	288ms	2406
Tabu search	32	GLobal threats	8 x 8	1000	0	96ms	2221
Tabu search	32	Local threats	8 x 8	1000	0	293ms	2297
Tabu search	32	Global threats	16 x 16	100	0	1562ms	23797
Tabu search	64	Global threats	16 x 16	100	0	1268ms	20454
Tabu search	128	Global threats	16 x 16	100	0	1097ms	18391
Tabu search	256	Global threats	16 x 16	100	0	1101ms	18465

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

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)