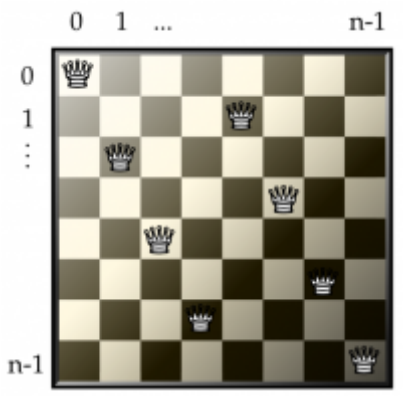
Assignment #3 - DCSP

Assignment



Implement Java agents that will cooperatively solve the **n-queens puzzle [http://en.wikipedia.org/wiki/Eight_queens_puzzle]** using the *Asynchronous Backtracking* (ABT) algorithm. There are N agents indexed 0,..,n-1 representing N queens on an NxN chessboard. Agent *i* controls the queen at the *i*-th row of the chessboard. The task is to find a corresponding column for each agent/queen such that no two agents/queens attack each other according to the classical rules of chess. The rows and columns are indexed as indicated on the picture at the left. The solution must be found in a decentralized manner using ABT (or possibly another complete asynchronous decentralized CSP algorithm). I.e., the agents must be able to detect that a) a valid assignment has been found by all agents (if it exists) and b) that the NxN chessboard does not admit a valid solution.

A good starting point to learn ABT is the book Fundamentals of Multiagent Systems [http://jmvidal.cse.sc.edu/papers/mas.pdf] by J. M. Vidal (chapter 2.1.3 on page 24).

Deadline

This assignment is due **8. 1. 2018 4:00 AM**.

Submission

The solution is to be submitted through the CW Upload system [https://cw.felk.cvut.cz/upload/]. You are expected to submit one zip file that must contain one directory "`student`" with your source code and file `report.pdf` with your report. The `student` directory is expected to include at least file `MyQueenAgent.java` with the implementation of your agents and optionally any other custom classes.

If you cannot access the upload system, send your solution to `karel.horak (zavinac) agents.fel.cvut.cz`.

Assessment

The maximum number of points for this assignment is **12 pts**.

Report

One page report should be submitted together with your source code. The report should contain:

- How is the n-queens problem modeled as a DCSP? (variables, domains, constraints, agents)
- How is the ABT algorithm customized for the n-queens problem?
- How do you determine priorities between agents?
- How do you detect that the search has terminated?

Assessment specification

The total of 12 points can be earned as follows.

- Agents find a valid solution or detect non-existence of a valid solution on a problem with 3 queens: **3 points**

- Agents find a valid solution or detect non-existence of a valid solution on a problem with 4 queens: **2 points**
- Agents find a valid solution or detect non-existence of a valid solution on a problem with 8 queens: **2 points**
- Agents find a valid solution or detect non-existence of a valid solution on a problem with 12 queens: **3 points**

(each will be run 10 times to test the reliability of your algorithm)

- Guaranteed termination detection is used (e.g. not one based on a timeout): **1 point**
- Reasonable report is included: **1 point**

Exceptions

- Non-general, hard-coded, centralized or other solution that does not satisfy the specification: **0 pts**
- Solution handed in after the deadline: **0 pts**

Packages to download

Agents Java template

Agents

An example implementation of one queen agent is in `MyQueenAgent` class in `student` package. The whole team of agents can be started using `StartAgents` class, use the NAGENTS constant within the class to specify the number of agents to start.

Make sure that lib/alite-SNAPSHOT.jar is added to your build path.

F.A.Q.

Q: Can I use the synchronous backtracking approach?

A: No, synchronous decentralized approach (where only one agent assigns a value to a partial solution at a time) does not allow for parallelization of the search. Your solution must be implemented in an asynchronous manner.

—

Q: My code does not compile in the Upload System.

A: The automatic evaluation uses Java 7 and thus it does not support new features from Java 8.

—

Q: Do I need to use hyper-resolution to generate no-goods?

A: No, since the focus of the assignment is on the asynchronous decentralized search and not on the hyper-resolution, you can use any simpler method you like to generate no-goods.

—

Q: Can I use global knowledge to detect termination?

A: No, termination must be detected only using local knowledge, i.e. each agent can check only the constraints in which it is involved.

courses/be4m36mas/assignment3-dcsp.txt · Last modified: 2017/12/11 14:52 by horakka5