Multi-Agent Systems

B. Nebel, R. Bergdoll, T. Engesser
Winter Semester 2019/20

University of Freiburg
Department of Computer Science

Exercise Sheet 6

Due: December 6, 2019

Exercise 6.1 (CA* Implementation, 12 points)

The objective of this exercise is to implement the Cooperative A* algorithm as presented in the lecture. To test your implementations, you should use grid world multi-agent path-finding benchmark problems as provided by Nathan Sturtevant's Moving AI Lab [1]. Such instances consist of two text files: a map file (encoding the map's terrain using ASCII characters) and a scenario file (specifying the start and goal coordinates as well as assigning a bucket number for each agent). Make yourself accustomed to the input format at `https://movingai.com/benchmarks/formats.html`. To keep the exercise simple, we adjust the original interpretation as follows:

- Terrain types `.`, `G`, `S` and `W` are traversable unconditionally, all other types are impassable.
- At each point in time, each agent may move one step in either x or y direction (but not diagonally), or wait. All actions have unit costs.

Your goal is to implement a CA* solver that takes a scenario file and a bucket number as input, constructs a MAPF task containing all the agents from that bucket, and solves this task on the respective map, if possible. Your program should **not** try to optimize the order of the agents but just compute the paths according to the agent order in the scenario file.
We provide a C++ code base, which handles input/output parsing and provides some suitable helper classes. You can download it from `http://gki.informatik.uni-freiburg.de/teaching/ws1920/multiagent-systems/mapf.zip`. However, you may also use any other programming language, as long as the resulting program adheres to our specifications. The implementation of the actual algorithm should consist of:

(a) A modified version of A*, which respects a given reservation table and provides a wait action.

(b) A CA* module, which successively calls A* for each agent and updates the reservation table. Remember that agents do not just disappear at their goal, but stay there in perpetuity.

We also provide a set of example instances, on which you can test your code. Some of those are not solvable by CA*, in which case your program should terminate with an appropriate message. Optionally, you can try to visualize your output using GraphRec[2]. If you use our code base and adhere to its proposed output format, then you can directly use the output files as input for GraphRec (make sure that the output files have an `.xml` ending). Be aware that GraphRec is no longer getting updated and it is a hassle to get it to run on newer systems (try Ubuntu 16.04).

Submit your program to `bergdolr@cs.uni-freiburg.de` and provide compilation/execution instructions if necessary.

[1] `https://movingai.com/benchmarks/mapf.html`
[2] `http://koupy.net/graphrec.php`