Solution Summary

1. Overview

This PDDL based problem solving is what has been utilized in solving Wumpus World Escape. The main objective is to create an agent who can go through a complicated world with several obstacles and dangers in order to find the way out.

2. Important Ideas and Approach

2.1. PDDL Domain Modeling

- To start with, Wumpus World was modeled into a PDDL domain by defining many predicates of world states like agent position, object positions and contents of the agent’s inventory.
- Careful action design was necessary because there are numerous things that an agent can do in Wumpus World: walking, pushing objects, shooting arrows unlocking doors and jumping over barriers.
- Some complex situations were handled by conditional effects such as automatically picking items whenever agents reached their locations.

2.2 Problem Generation

Below is a description of how it was done.

- generate_problem.py which is a Python script was created that took textually represented maps and automatically transformed them into PDDL problem files.
- Therefore, one can easily scale them for numerous maps and multiple configurations of worlds.

2.3 Planning with Fast Downward

- We employed the Fast Downward planner, a top classical planning system, to solve generated PDDL issues.
- The planner uses lazy greedy search with heuristics to achieve a good mix of solution quality and computational performance.

2.4 Plan Execution and Verification

- ‘execute_plan.py’ post-processes the raw plans generated by Fast Downward and converts them into the required action format.
- Another script, verify.py, was written to check the correctness of the generated solutions against the original map.

2.5 Batch Processing

- The main script (main.py) drives the whole processing and provides options to handle batch processing of multiple maps,
- Testing can be very easily done in this model and validations for various scenarios are also easy to handle.

3. Problems Encountered and Solutions

3.1 Complex Action Modeling

- Challenge: how to model actions which have several effects and preconditions, the pushing action and jump action are very challenging.
- Solution: use of conditional effects in PDDL and design the predicates delicately for every action.

3.2 Efficient Problem Encoding

- Challenge: Generating PDDL problem files in a way that is accurate but, at the same time, efficient for the planner to process.
- Solution: Improved problem generation script to output compact, succinct PDDL representations with no superfluous information.

3.3 Performance of the Planner

- Challenge: In some complicated maps, planning was taking too much time or causing memory issues.
- Solution: Ran various configurations and heuristics in Fast Downward to get the right balance between solution quality and computational efficiency.

3.4 Unsolvable Maps

- Challenge: How to discern between genuinely unsolvable maps and those that were simply taking more time to compute.
- Solution: Timed out maps and prepared empty solution files for them; we would revisit them later either for analysis or with re-runs with different planner settings.

3.5 Edge Cases in Map Layouts

- Challenge: How to deal with special map configurations that exposed unique edge cases in our domain modeling.
- Solution: Repeat PDDL domain and problem generation in order to handle more scenarios, including the most complex ones involving trampolines and crates.

4. Conclusion

The PDDL-based solution approach gave satisfactory results on various map configurations of Wumpus World Escape problem. Dividing the problem into a domain modeling, problem generation, planning, and solution verification allows us to present a system highly robust and scalable. The main challenges were the correct modelling of complex world dynamics in PDDL and optimization of the efficiency of the planning process. In the future, further optimizations of the planner configuration can be done for more complex maps, and alternative ways of planning can be studied for cases where classical planning is in difficulty.