CS5011: Assignment 2 - Logical Agents - Daggers and Gold Sweeper

Assignment: A2 - Assignment 2

Deadline: 5th of November 2018

Weighting: 25% of module mark

Please note that MMS is the definitive source for deadline and credit details. You are expected to have read and understood all the information in this specification and any accompanying documents at least a week before the deadline. You must contact the lecturer regarding any queries well in advance of the deadline.

---

1 Objective

This practical aims to implement an AI logical agent that is able to play and solve a Daggers and Gold sweeper game.

2 Competencies

- Design and implement a logical agent that is able to perceive certain facts about the world it occupies, and act accordingly
- Use of logical reasoning to solve a puzzle

3 Practical Requirements

3.1 Introduction

The Daggers and Gold (D&G) sweeper game is inspired by the well-known Mineswepeer computer game[1]. The D&G sweeper world is a square grid of NxN squares with D daggers and G gold mines scattered among them. In addition the agent has a number of lives L. Any square can be probed by an agent in any order, but the opening move is to be made on the top left-hand corner. If the agent probes a square that contains a Dagger, the agent will loose a life. If the agent probes a square that contains a Gold mine, the agent will gain a life. Otherwise, a clue appears on the square indicating the number of Daggers in the 8 adjacent neighbours of the probed square. The agent will start with one life only, and its aim would be of clearing all the cells, however, when it runs out of lives, the game is over. The rules of the D&G sweeper game are similar to those of Minesweeper, but the D&G worlds used for this practical are a subset of those of Minesweeper. In this practical, we

[1] `http://www.minesweeper.info/wiki/Main_Page`

will develop a logical algorithm that can be used by the agent to clear the cells reducing the likelihood of loosing the game.

3.2 The task

For this practical, a Java file `World.java` is provided, defined as Enum Type[2]. This file contains the D&G worlds to be used, 30 worlds in total, named according to different levels of difficulty, *EASY, MEDIUM, HARD*. Each world is represented as Java array filled with Strings, where each String represents a cell with the following code:

- `"d"` means that the cell contains a Dagger
- `"0"`– `"8"` is the number of Daggers in the 8 adjacent neighbours
- `"g"` means that the cell contains a Gold Mine

You are free to use this input class as-is or you may modify it, however, please ensure that you do not change the worlds to be tested.

Each level has 10 examples of D&G worlds to be used in this assignment with the following characteristics:

- Easy level: in which N=5, D=4, G=1 indicating a board of 5x5 with 4 Daggers and 1 Gold mine
- Medium level: in which N=8, D=10, G=4 indicating a board of 8x8 with 10 Daggers and 4 Gold mines
- Hard level: in which N=12, D=25, G=3 indicating a board of 12x12 with 25 Daggers and 3 Gold mines

The rules of the game are as follows:

1. The total number of Daggers is known by the agent in advance.
2. The agent starts with one life $L = 1$.
3. At the initial step the cells are all covered and the agent has no other information other than the total number of Daggers. The first probe is to be made at the top left-hand corner at position $[0, 0]$.
4. At each step the agent probes a covered cell, and receives the information about the content of the cell. The cell will then be marked as uncovered.
5. If the probed cell is a Dagger, the agent loses a life (-1 life)
6. If the cell contains a value 0 meaning that no adjacent cells contain Daggers, all the adjacent cells will also be uncovered.

[2] `http://docs.oracle.com/javase/tutorial/java/javaOO/enum.html`

7. If the probed cell is a Gold Mine, the agent gains a life. In this case, the cell should also be considered as a cell with value 0, meaning that no other cells around the gold mine contain daggers.

8. If the content of the cell has a value > 0, this is a clue about the number of Daggers existing in the 8 adjacent cells. The agent should make inferences about its view of the world and decide which cell should be probed next.

9. When the total number of lives is 0, the agent dies and the game is over.

10. If all but D cells are uncovered without dying, the agent wins the game.

11. The agent may use stone-flags to signal the presence of a Dagger in covered cells.

There are a number of possible reasoning strategies that the agent can employ, the agent goal is to employ one that limits the amount of random probing needed to solve the game, in order to increase the chances of winning. For more details on the strategies, please refer to the lecture slides.

3.3 Part 1: Core D&G Sweeper Agent

For this part of the practical, you are required to implement an agent that can play the D&G sweeper game using a Random Probing or a Single Point Reasoning strategy. The following requirements should be fulfilled for this part:

1. Describe the PEAS agent model of this task in your report

2. Write a program that takes any of the D&G worlds provided and permits an agent to play the game by probing the cells and count the agent lives.

3. The agent should hold a knowledge base indicating the cells that are yet to be probed, and the information about the cells already uncovered. The agent should be able to probe a cell and receive the information contained in that cell.

4. Implement the Random Probing and the Single Point reasoning strategies:

   - **Random probing strategy RPS**: As minimum requirement for this step, the agent should be able to randomly select a cell to probe and identify whether a Dagger, a Gold mine or a clue is present in the selected cell.
   - **Single point strategy SPS**: Additionally provide the agent with the ability of storing the cells that are stone-flagged as Daggers in the knowledge base. The next cell to be probed and the Dagger flagging approach should take into consideration the knowledge acquired so far. Use the single point reasoning strategy to develop the reasoning approach for this part to uncover/flag covered cells. Resort to random probing when no other move can be made.

5. At each step, a statement should be printed out indicating the action of the agent or the state of the game; e.g. "reveal x y" for uncovering a cell in [x,y] coordinates, "stone-in x y" for marking the presence of a Dagger in [x,y], "goldmine-in x y, new life

count l" for uncovering a gold mine, "dagger-in x y, new life count l" for uncovering a dagger, "game won" or "game lost", etc ...

6. Describe the implementation of the game infrastructure in your report.
7. Describe the implementation of this agent and its strategy in your report.
8. Evaluate the agent performance on the given D&G worlds. Comment on the amount of random probing required to solve each of the D&G worlds.

3.4 Part 2: Logical D&G Sweeper Agent

For this part of the practical, the aim is to improve the agent strategy of Part 1 to be able to solve the games by modelling the world as a logic sentence, and use satisfiability results to select the next move. A solver should be used to prove that a given cell does or does not contain a Dagger.

1. Use the implementation of Part 1 for both agent and game logic.
2. As in part 1, the agent should be able to probe a cell and receive the information contained in that cell as above.
3. Run all instances of SPS; when no other moves are available switch to ATS.
4. Implement the **SAtisfiability Test Strategy ATS** for this step:
   - The next cell to be probed and the Dagger marking approach should take into consideration the knowledge acquired so far. These steps should be identified using satisfiability information obtained with a common SAT solver. For this part we will use two existing libraries[3]: SAT4J Core[4], a java library for solving boolean satisfaction and optimisation problems; and the next generation logic library[5] Logic^{NG}. The following steps should be taken:

   (a) Write a boolean formula for the cell under inspection.
   (b) Transform this formula in CNF form using the library Logic^{NG}.
   (c) Convert the CNF form into a DIMACS form compatible with SAT4J input.
   (d) Use SAT4J to determine the satisfiability of this formula
   (e) Use the result to decide what cell should be probed next, then start again from point 3 (SPS).
   (f) When no other solution is possible, resort to random probing (RPS).

5. Describe the implementation of this agent and its strategy in your report.
6. Evaluate the agent performance on the given D&G worlds. Comment on the amount of random probing required to solve each of the D&G worlds. How do these results compare to the results of the previous agents?

[3]Please refer to the lectures for details on how to use the following libraries:
[4]`http://www.sat4j.org/products.php#core`
[5]`http://github.com/logic-ng/LogicNG`

3.5 Part 3: Extensions

It is strongly recommended that you ensure you have completed the Requirements in part 1 and part 2 before attempting any of these additional requirements. You may consider one or more of the following additional tasks:

- Implement an agent that plays the D&G sweeper game with or against a user through the terminal in a turn-taking fashion (text interface is sufficient).
- Include a board generator that checks the validity of a board.
- Use additional clues for Gold Mines, such that the 4 adjacent cells of a Gold Mine (North-South-West-East) include a clue `"m"` that concatenates with the number of adjacent daggers. For example, if we probe a cell which is adjacent to 1 Dagger and a Gold Mine, the cell will return `"1m"`. Use the clues to guide the reasoning strategy.
- Suggest an additional strategy to improve the performance of Part 2 in case the agent has to resort to random probing.
- Allow a group of agents to play the D&G sweeper game against each other.
- Include additional items in the game, write the rules for these items and extend the implementation of the game to include these new rules (some inspiration may be drawn from the Wumpus game).
- Consider hexagonal boards instead of square boards.
- Suggest your own additional requirements: for a significant extension, you may increase the complexity of the game and extend your reasoning strategies to guide the agent to solve the problem, or identify additional strategies for the current game that will improve the agent performance.

4 Code and Report Specifications

4.1 Code Submission and Running

The program must be written in Java and your implementation must compile and run on the School lab Machines. **Please do not use libraries that implement minesweeper algorithms.** Please make use of at least a class to handle the game and a class to handle the agent functionalities.

A jar file is to be submitted for each part of the three parts attempted.
Name the jar file for Part 1 as "Logic1.jar" and ensure it runs using:

```
java -jar Logic1.jar [any param]
```

Name the jar file for Part 2 as "Logic2.jar" and ensure it runs using:

```
java -jar Logic2.jar [any param]
```

Name the jar file for Part 3 as "Logic3.jar" and ensure it runs using:

```
java -jar Logic3.jar [any param]
```

4.2 Report

You are required to submit a report describing your submission, with a limit of 4500 words excluding references. The report should include:

1. A list of the parts implemented and any extension attempted.
2. If any of the functionalities is only partially working, ensure that this is discussed in your report.
3. Literature Review: A short literature survey on minesweeper or similar games and algorithms.
4. Design: A description and justification for the mechanisms you have implemented as part of your solution.
5. Examples and Testing: Examples of the main functionalities and your approach to testing the system.
6. Evaluation: A critical analysis of the functionalities of your system and what can be improved.
7. Running: Include clear instructions on how to run your system
8. Bibliography: List all the references you cite in your report and code.

More details on points to be discussed under sections 4–6 are listed in the description of each part of this assignment. Please include a word count in your report and an estimate of the time taken to complete the assignment in hours.

5 Deliverables

A single ZIP file must be submitted electronically via MMS by the deadline. Submissions in any other format will be rejected.

Your ZIP file should contain:

1. A PDF report as discussed in Section 4.2
2. One, two, or three jars depending on the tasks achieved as discussed in Section 4.1
3. The source code of your implementation and any non-standard libraries

6 Assessment Criteria

Marking will follow the guidelines given in the school student handbook (see link in the next section).

The following issues will be considered:

- Achieved requirements
- Quality of the solution provided
- Code quality
- Examples
- Insights and analysis demonstrated in the report

Some guidelines are as follows. For a mark up to the band 11-13, the game infrastructure and a random probing strategy should be achieved in Part 1. For a mark up to the band 14-16 you must complete Part 1 and make an attempt to Part 2. For a mark up to 17 you must complete Part 1 and Part 2. To obtain marks above 17, in addition to Part 1 and 2, at least one extension should be completed as those suggested in Part 3. All parts are to be accompanied by a good insightful report covering all points and good design and implementation quality.

7 Policies and Guidelines

7.1 Marking

See the standard mark descriptors in the School Student Handbook

`https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_-Descriptors`

7.2 Lateness Penalty

The standard penalty for late submission applies (Scheme B: 1 mark per 8 hour period, or part thereof):

`https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#latenesspenalties`

7.3 Good Academic Practice

The University policy on Good Academic Practice applies:

`https://www.st-andrews.ac.uk/students/rules/academicpractice/`

Alice Toniolo
(a.toniolo@st-andrews.ac.uk)
October 10, 2018