COMP 472

Rani Rafid[1], Gabriel Haoois[2], and Manpreet Tahim[3]

[1] 26975852 `rani.rafid@concordia.ca`
[2] 40058821 `your@email2.com`
[3] 26592006 `your@email3.com`

1 Introduction

2   Heuristics

# 3   Challenges

3.1   Relational Modeling

A large portion of time leading to the completion of the project was attributed to understanding how subjects should be linked. This process was judged to be difficult since the domain requirements can be ambiguous at times. To resolve this issue, an experiment was conducted using the three main subjects of the domain: agents (search algorithms), puzzles (graphs), and nodes (board state).

(i) Agents use a puzzles and nodes.

(ii) Puzzles use agents and nodes.

(iii) Each agent uses puzzles and nodes.

(iv) Each puzzle uses agents and nodes.

When the models were compared on the same data set, it was observed that the average run-time for models (i) to (iv) was 2 minutes, 66 ms, 102 ms, and 18 ms respectively. Moreover, results of the experiment led to two important findings that helped us classify the domain requirements of the project.

The first finding resolves the concern of whether or not to use agent or puzzle as the primary subject of the system. For example, should (i/iii) be chosen over (i/iv)? If agent was the primary subject, then each agent must have a reference to a puzzle, and each agent would rely on the puzzle to provide it with a new state for every step of a traversal. The issue with this is that each agent had similar implementations for traversing the puzzle. This affected the overall performance of our system because there were redundancies for the traversal behaviour. To resolve this issue, we took away the responsibility of traversing nodes from the agent class and assigned it to the puzzle class. As a result of this change, the agent's responsibility was now limited to computing heuristic values. Overall, this finding suggests against using (i) and (iii) as a schema for modelling the system.

The first finding also suggests that agents should not have direct access to the node class. Yet, having access to this object is important for the system to function properly since it stores the values of each heuristic. A first approach to this problem was to include an accesssor method in the puzzle class. However, this increased the time complexity for state searching by a linear factor, and also contradicts the schema proposed in (ii) and (iv). Therefore, the puzzle needs to reference the agent.
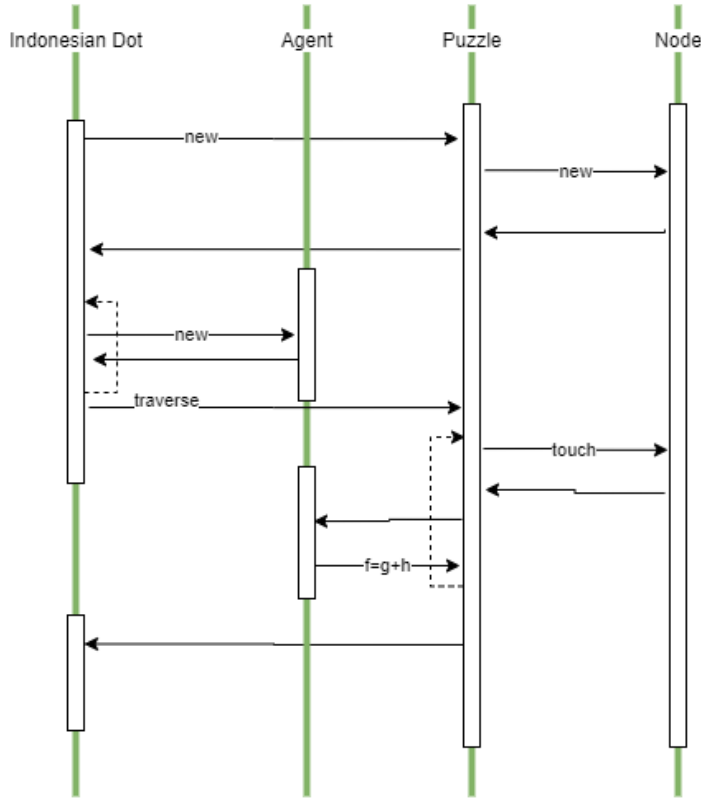
Fig. 1. Sequence Diagram of Indonesian Dot Solver

This leads to the next and last finding: A puzzle and an agent cannot have a many-to-many relational type because agent no longer has a reference to a puzzle. Moreover, many nodes are used by one puzzle. Therefore, the combination of both the agent and the node identifies the puzzle that it belongs to. This makes sense since each node points to it's predecessor, with the root node uniquely linked to the puzzle it belongs to (assuming there are no duplicate lines in the test file). Therefore, the second finding suggests that the logical architecture should be modelled after statement (iv).

Overall, by performing this experimentation and discovering why the logical architecture performed well under different scenarios, we were able to create a classification plan for classifying each domain requirement into one that belongs to agent, puzzles, or nodes. This experimentation also helped us maximize the performance of the overall system.
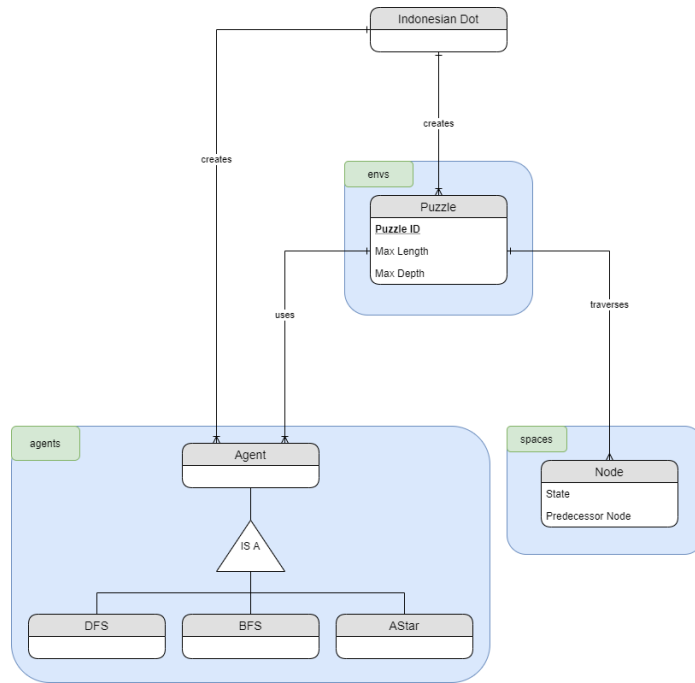
Fig. 2. Entity Relationship Diagram of Indonesian Dot Solver

3.2   Touch Redundancy

Every state of the Indonesian Dot puzzle has actions equivalent to the size of the board. For example, if the board size 25 (or 5x5), then there are 25 actions to choose from. Whenever a traversal was made, it used to be that the puzzle object would touch every action on a node and if it was visited, then it would ignore the action. Yet, checking is an expensive operation especially when the board size increases in size. The touch redundancy issue refers to the issue of having to check for 2^{2n} actions in the worst case, where n is the size of the board. This is called the touch redundancy issue and can be seen in the following Figure:

```
Agent dfs average time is 1.21e+02 ms.
Agent astar average time is 2.76e+02 ms.
Agent bfs average time is 2.8e+02 ms.
```

Fig. 3. Average Run Time of DFS, BFS, and A* on 7 Puzzles with Touch Redundancy Issue

To resolve this issue, we discovered that the order of actions leading to the goal state was not important. More formally, the solution path (in actions) can be any combination of the solution set. This also implies that the system needs only to check 2^n actions in the worst case. This discovery greatly helped reduce the run time of the system, as can be seen below:

```
Agent astar average time is 1.09 ms.
Agent bfs average time is 1.19 ms.
Agent dfs average time is 31.3 ms.
```

Fig. 4. Average Run Time of DFS, BFS, and A* on 7 Puzzles without Touch Redundancy Issue

4   Analysis

5   Contributions

5.1   Rani Rafid

Rani Rafid designed and implemented the logical architecture of the Indonesian Dot puzzle solver system. He assigned responsibilities to his teammates and managed the project during the iteration. He also set deadlines and organized meetings with the team.

5.2   Gabriel Haoois

Gabriel Haoois implemented the main (entry) program for the Indonesian Dot puzzle solver system. He contributed to the development of complimentary features for the system and theorized creative solutions for past issues.

5.3   Manpreet Tahim

Manpreet Tahim implemented the search algorithms of the Indonesian Dot puzzle solver system. He explored and studied several heuristic functions. He also shared his findings with his teammates, and contributed to the development of the logical architecture.

References

1. Author, F.: Article title. Journal **2**(5), 99–110 (2016)
2. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016). https://doi.org/10.10007/1234567890
3. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)
4. Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)
5. LNCS Homepage, http://www.springer.com/lncs. Last accessed 4 Oct 2017