# Introduction to Intelligent Systems
# Project1: Rolling Die Mazes

Joseph Fuchs, Damien Cremilleux

October 8, 2014

## 1 Problem definition

The objective of this project is to solve rolling-die mazes. A die start from a location on the maze, and can roll along its edges through a grid, until a goal location is found. The mazes can contain obstacles, and there are restrictions on which numbers may face up on the die. The die will always start with the 1 facing up ('visible'), 2 facing up/north, and 3 to the right/east. Moreover, the number 6 should never be face up on the die, and the number 1 must be on top of the die when the goal location is reached.

An example of maze is given below (S is the start location, G is the goal location and * are obstacles).

```
. G * . S .
. . . * . .
. * . . . *
. . . . . .
. . . * . .
```

This problem can be formulated as follow:

- States: The state is determined by both the die position and its orientation. So, a states description specifies the location (ie the coordinates) and the orientation (which number is facing up, north and east) of the dice on the maze.
- Initial state: The die is on the start location, with the 1 facing up.
- Actions: Rolling north, south, east or west. Different subsets of these are possible depending on the orientation of the dice (the number 6 should never be face up).
- Transitions model: Given a state and action, this return the resulting state.
- Goal test: The die is on the goal location, with the number 1 on the top.

- Path cost: Each step costs 1, so the path cost is the number of steps in the path.

# 2 Heuristics

We use A* algorithm (discussed in the class) to solve this search problem. A* includes as a component of its evaluation function a heuristic function, $h(n)$. In the following sections, we describe the three heuristics chosen to solve this search problem.

## 2.1 Uniform-cost search

For the first heuristic, $h_1$, we chose a very simple solution: $h(n) = 0$. This means that the evaluation function is now $f(n) = g(n)$ where $g(n)$ is the cost to reach the node. With this heuristic, we have an uniform-cost search algorithm. Uniform-cost search first visits the node with the shortest path costs from the root node.

This heuristic is admissible. It will never overestimate the cost to reach the goal because the heuristic is equal to 0, and we are dealing with positive step costs.

Moreover this heuristic is consistent, which is a stricter requirement than admissibility. According to the textbook, a heuristic is consistent if, for every node $n$ and every successor $n$ of n generated by any action $a$, the estimated cost of reaching the goal from $n$ is no greater than the step cost of getting to $n$ plus the estimated cost of reaching the goal from $n$:

$h(n) \leq c(n, a, n) + h(n, )$

$0 \leq c(n, a, n)$

This is true because we are dealing with positive step costs.

## 2.2 Manhattan distance without obstacles

The second heuristic, $h_2$, is based on the city block distance, or Manhattan distance, which is the sum of the absolute differences of Cartesian coordinates. To find this heuristic, we take the original problem with fewer restrictions: no obstacles and we do not take into account what is the orientation of the die. We illustrate this on the following maze. The die is on the cell marked D. In this case, the heuristic will be $h(n = D) = 4$.

```
S . . . .
D . * * *
. * . . G
. . . . .
* . . . .
```

So this is a relaxed problem of the first one. We know that the cost of an optimal solution to a relaxed problem is an admissible heuristic for the original

problem. Moreover, this heuristic is also consistent, because it is the exact cost for the relaxed problem so it must obey the triangle inequality. Finally, this heuristic gives bigger values than h1. This means that we can expect better results with this heuristics, compare to the previous one.

### 2.3 Manhattan distance with orientation

For the last heuristic, $h_3$, we also generate an heuristic from a relaxed problem. We know only ignore the restriction about the obstacles. Contrary to the heuristic two, we take the orientation of the die into consideration. Given the position and the orientation of the dice, it is possible to find a minimal number of rotations to do to solve the maze. For instance, if the die with the 1 on top is on the same row or column as the goal position, the maze can be solve in $mahattan\_distance + 2$.

So we can generate a set of rules giving the position and orientation. We observe that each time the rule has the form $mahattan\_distance + cst$, the value given by $h_3$ will be at least as big as $h_2$ for all nodes. This heuristic is likely to be more effective than the previous one. We illustrate this heuristic with the following maze. Now the heuristic will be $h_3(n = D) = 4 + 2 = 6$.

```
S . . . .
. . * * *
D * . . G
. . . . .
* . . . .
```

We again used a relaxed problem of the original one. For the same reasons like $h_2$, this heuristic is admissible and consistent.

## 3 Performance metrics

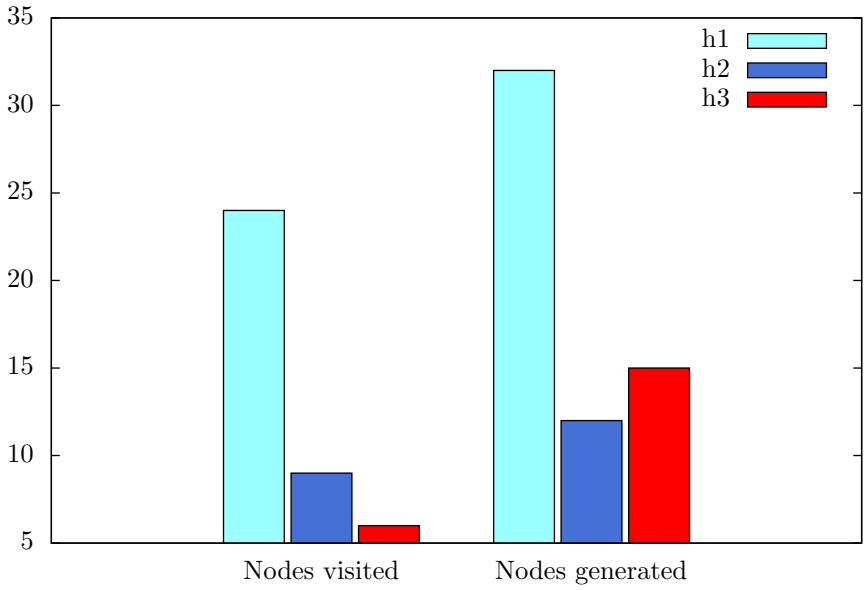
For each heuristics, we ran the five puzzles given in the wording. The results are in the following tables.

| | Puzzle 1 | | | Puzzle 2 | | | Puzzle 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| | $h_1$ | $h_2$ | $h_3$ | $h_1$ | $h_2$ | $h_3$ | $h_1$ | $h_2$ | $h_3$ |
| Number of nodes generated | 32 | 12 | | 95 | 67 | | 2 | 2 | |
| Number of nodes visited | 24 | 9 | | 84 | 50 | | 3 | 3 | |

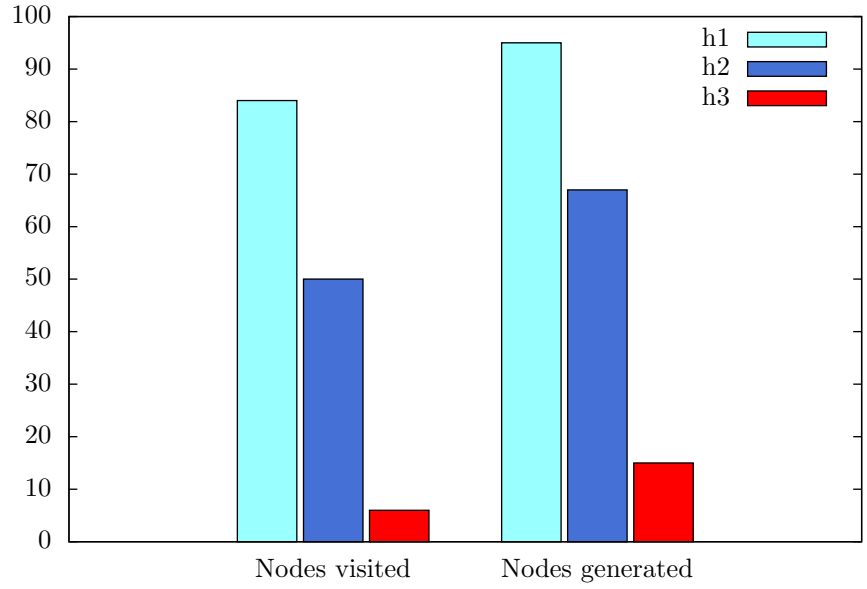
| | Puzzle 4 | | | Puzzle 5 | | |
|---|---|---|---|---|---|---|
| | $h_1$ | $h_2$ | $h_3$ | $h_1$ | $h_2$ | $h_3$ |
| Number of nodes generated | 161 | 101 | | 1271 | 1080 | |
| Number of nodes visited | 149 | 82 | | 1260 | 737 | |

To show the difference, the following bar graphs give the number of nodes generated and visited according to the heuristic, for each puzzle.

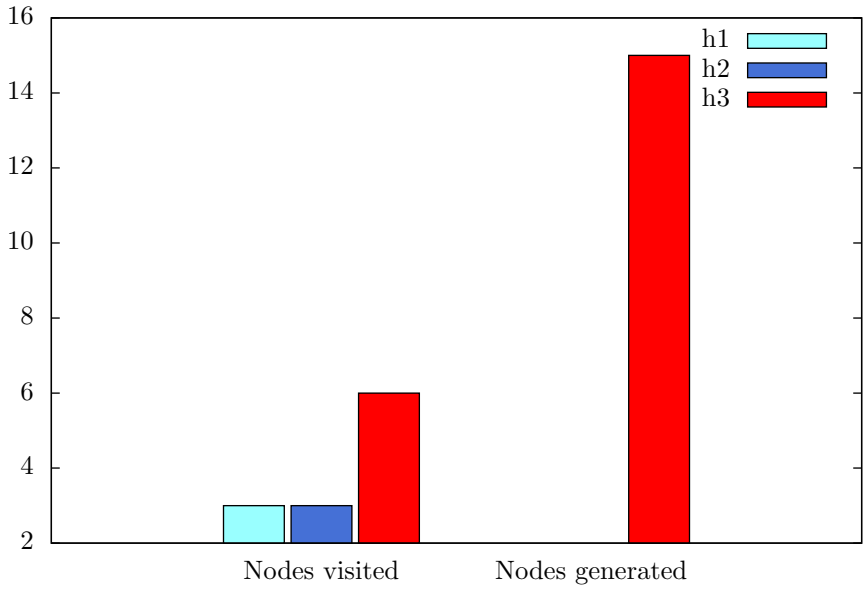
Results for puzzle 1



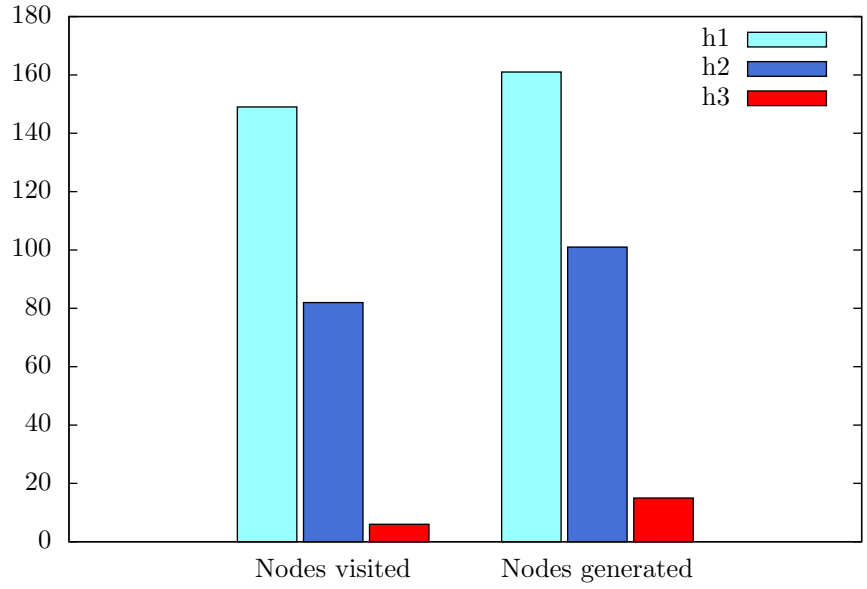

Results for puzzle 2

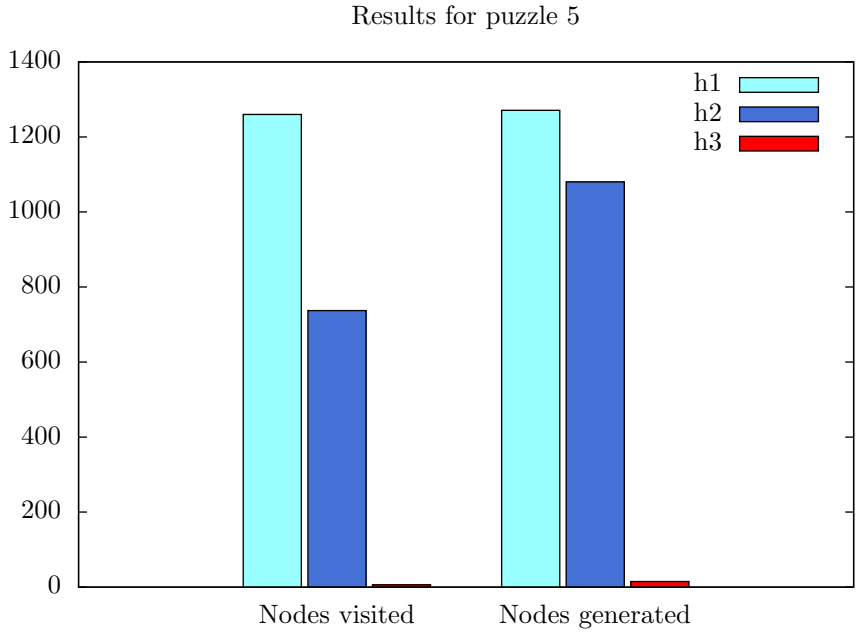


Results for puzzle 3




Results for puzzle 4






# 4 Discussions

The first observation we can make is that the first heuristic generates a huge amount of nodes, more than 1200 for the puzzle 5. The second heuristic is better, however the number of nodes generated is still huge. According to our expectations, the third heuristic has the best results. With this heuristic, the number of nodes generated is only .

For the puzzle 3, we see that the number of nodes generated is less than the number of nodes visited, for all the heuristics. The explaination is that the initial node is visited, and not expanded.

As a conclusion, this project shows use the importance of the choice of heuristic. The quality of the heuristic has a huge effect on the performance of the algorithm.