

COMP30024 – Artificial Intelligence

Project Part A - Searching

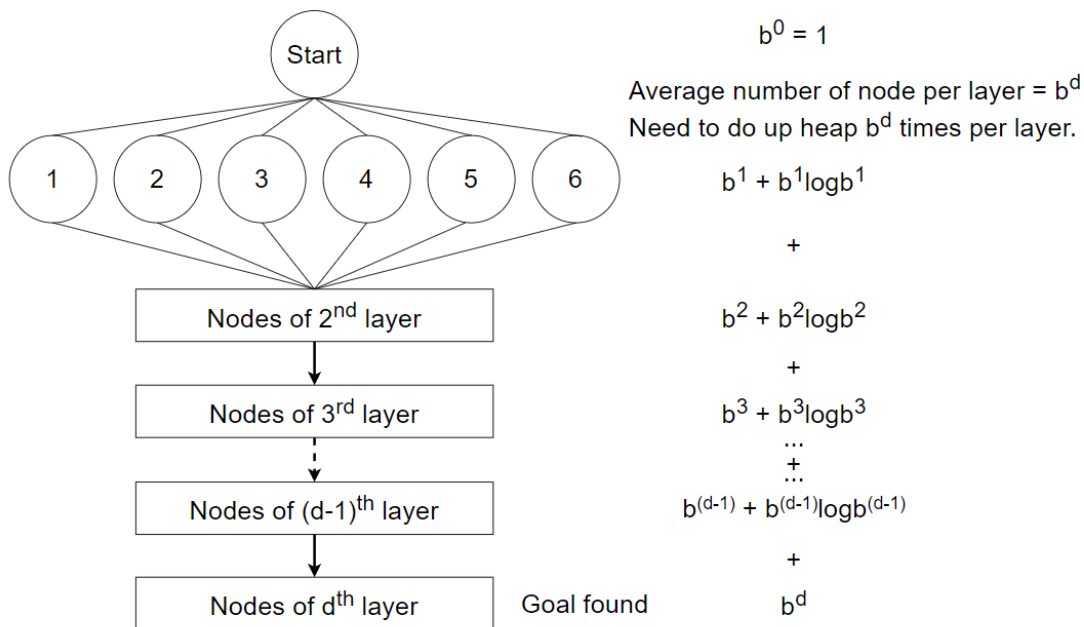
Team: ELDEN_KING
Jiahao Chen, Zhiquan Lai

I. Implementation of A* Search

A* search used an evaluation function so that if the function is monotonic, the solution must have lower costs than all unexpanded nodes. The evaluation function ($f(n)$) consists of two parts, the cost so far to reach the current node ($g(n)$) plus the estimated cost to reach the goal from the current node ($h(n)$). The latter one is an admissible heuristic function, which will be discussed in the next part in detail. The algorithm expands the node with the lowest $f(n)$ each time. To implement node expansion, a priority queue is applied so that the node with the lowest $f(n)$ is guaranteed to be placed at the top. Figure 1 explains our implementation details.

d = depth of the least solution
 b = average number of successors per state < 6
 k = current length of priority queue

Find neighbours and compute $f(n)$
 Push new node to the queue and do up heap: $\log k$
 Expand the node with the least $f(n)$



Goal test is performed upon node expansion.

The queue does not need to be sorted when the goal is found at d^{th} layer so the cost at d^{th} layer is b^d .

← Time complexity of A* = $O(b^0 + b^1 + b^1 \log b^1 + \dots + b^d) = O(b^d)$ →

Space complexity = Number of all nodes = $O(b^d)$

Time/Space complexity of computing $h(n)$ are both $O(1)$ and are omitted above.

Figure 1: Implementation & Analysis of A*

II. Heuristic Function

Manhattan distance is applied as our heuristic function. By definition, a heuristic function $h(n)$ is admissible if $h(n)$ is not greater than the true cost from n . Suppose the shape of the board is quadrilateral, the length of the nearest path between two cells equals their Manhattan distance as it computes the sum of lengths of the projections of the line segment between the cells onto the coordinate axes.

In the case of a hexagonal board, cells in diagonal directions also have distances of 1. We improved the algorithm so that the nearest path between two cells on a hexagonal board equals their Manhattan distance. Therefore, $h(n)$ never overestimates the cost since some cells are occupied, which proves it is not greater than the true cost, namely admissible.

Euclidean distance is also considered, but it is less than Manhattan distance, as figure 2 shows. Since the larger the admissible function is, the better it performs, Manhattan distance becomes the optimal choice. The time and space complexity of $h(n)$ are both $O(1)$. The function only consists of basic math operations and stores the horizontal and vertical distance between two coordinates.

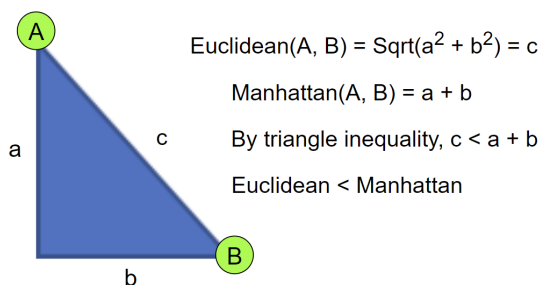


Figure 2: Euclidean < Manhattan

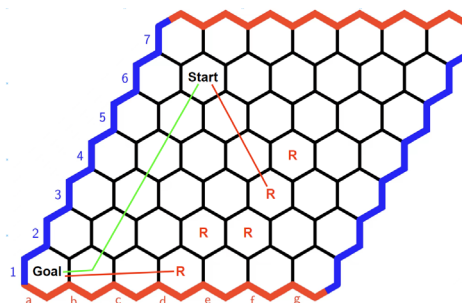


Figure 3: Example of suboptimal path

III. Challenge

In the case that existing pieces can be made use of, Manhattan distance overestimates the true cost because some moves cost 0 so that it is no longer admissible. At first, we intuitively set the cost of moving to existing pieces as 0. However, as figure 3 depicts, this can result in a suboptimal path, namely the green one. The existing pieces are not captured so that the algorithm cannot find the optimal path, which is the red one.

To solve this problem, we designed a new heuristic function $h(n) = \text{Manhattan distance} - \text{the number of existing pieces}$, which is forced to be bounded below by 0. Since it takes the existing pieces into consideration, it cannot be greater than the true value so that it is admissible. When no pieces can be captured, it works the same as the original one. If all the nodes have $h(n)$ equals 0, it will become a uniform-cost search, and the cost of capturing an existing piece is 0. As a result, the algorithm will fully use the pieces to find the optimal path.