Heuristic Function (h2) Explanation

Enver Eren

19.10.2024

Heuristic Function (h2) Explanation

In the chessboard problem where the objective is to capture all pawns using agents (knight, rook, and bishop), the heuristic function (h2) estimates the minimum cost required to reach each pawn, either from an agent or another pawn. The steps and logic used in h2 are as follows:

1. **Knight Cost Calculation**:
   - Calculate the horizontal and vertical distances:
     $$dx = |i - k_x| \quad \text{and} \quad dy = |j - k_y|$$
     where (i, j) is the position of the pawn and (k_x, k_y) is the knight's position.
   - The number of moves is determined as:
     $$n\_moves = \max\left(\left\lceil \frac{dx + dy}{3} \right\rceil, \left\lceil \frac{\max(dx, dy)}{2} \right\rceil\right)$$
   - The total knight cost is then:
     $$\text{knight\_cost} = 6 \times n\_moves$$
2. **Rook Cost Calculation**:
   - If the rook and pawn are on the same row or column without obstacles, the cost is 8.
   - If there is an obstacle between them on the same row or column, the cost is 24.
   - If they are not on the same row or column, the cost is 16.
   - If the pawn is restricted by obstacles and inaccessible by the rook, the cost is set to infinity to denote infeasibility.
3. **Bishop Cost Calculation**:
   - If the Manhattan distance between the bishop and pawn is even (meaning the pawn is accessible) and they are on the same diagonal without obstacles, the cost is 10.
   - If there is an obstacle on the diagonal, the cost is 30.
   - If the Manhattan distance is even but they are not on the same diagonal, the cost is 20.
   - If the Manhattan distance is odd (indicating the pawn is unreachable by the bishop), the cost is set to infinity.
   - If the pawn is restricted by obstacles and inaccessible by the bishop, the cost is set to infinity to denote infeasibility.
4. **Pawn-to-Pawn Cost Calculation**:
   - For each pawn, movement options from every other pawns are evaluated using all available agent movement patterns (knight, rook, bishop) from other pawn to the specific pawn.
   - If a pawn is surrounded by obstacles and inaccessible by rook or bishop, the knight's movement pattern is enforced. If no knight is present and a pawn is restricted, the cost is set to infinity to denote infeasibility.

5. **Cost Minimization Strategy**:
    - Eventually, for each pawn we have agents (knight, bishop, rook) to pawn costs and pawn-to-pawn cost which are calculated using movement patterns of existing agents.
    - For each pawn, the minimum cost is determined by considering all options (agents and pawns) and choose the minimum one for each pawn.
    - If all costs involve pawn-to-pawn costs, one of them is adjusted using an agent-to-pawn cost, ensuring the minimum increase in total cost.

This comprehensive approach ensures that the heuristic evaluates the minimal feasible costs efficiently.

Why the Heuristic (h2) is Admissible

A heuristic is admissible if it never overestimates the true cost of reaching the goal (capturing all pawns). Here's why h2 meets this criterion:

- **Accurate Agent Movement Costs**: The costs calculated for each agent (knight, rook, bishop) are based on the minimum number of moves and associated action costs. These values are realistic and account for obstacles and movement patterns specific to each agent, ensuring no overestimation.
- **Minimum Path Selection**: By evaluating all possible paths (agent-to-pawn and pawn-to-pawn) and choosing the minimum, since the cost calculations are indicating best costs the heuristic ensures that only the lowest possible costs are considered, maintaining the admissibility requirement.
- **Fallback to Infeasibility**: When a pawn is inaccessible due to the absence of a required agent (e.g., a knight when needed), the heuristic assigns a cost of infinity, accurately reflecting that reaching that pawn is impossible within the given configuration. This does not overestimate but correctly represents the game's constraints.

Heuristic Adjustment for Consistency

To ensure the consistency of the heuristic, I manually adjust the heuristic values using the following equation:

$$\text{former heuristic cost} \leq \text{move cost} + \text{new heuristic cost}$$

The consistency adjustment operates as follows:

- For each state transition, I calculate the **new heuristic cost** based on the current configuration.
- If the calculated **new heuristic cost** satisfies the equation above, meaning it does not violate consistency, the **new heuristic cost** is used.
- If the **new heuristic cost** does not satisfy the equation, I adjust the heuristic by setting it to **former heuristic cost - move cost** to ensure that the consistency condition is maintained.

By implementing this adjustment, I guarantee that the heuristic remains consistent throughout the algorithm's execution.

Why the Heuristic (h2) is Complete

A heuristic is complete if it guarantees finding a solution if one exists. The h2 heuristic achieves this by:

- **Loop Prevention**: Graph search algorithm used in the project is designed to check whether the node is expanded before or not. If it is not expanded, then expand it and store it in a set. By doing so, the algorithm prevents loops. Therefore, the algorithm can visit every possible configuration if the board size is finite. Eventually, if there exist a solution, the algorithm will visit it.