Predicting Subgoals in Ricochet Robots with a Graph Neural Network

HYNER Petr[1,2], MRÓGALA Jan[1,2], ADAMCZYK David[1]and HŮLA Jan[1,3]

[1]*Institute for Research and Application of Fuzzy Modeling, University of Ostrava*
Ostrava, Czech Republic

[2]*Department of Informatics and Computers, University of Ostrava*
Ostrava, Czech Republic

[3]*Czech Technical University in Prague*
Prague, Czech Republic

E-mail: {petr.hyner, jan.mrogala, david.adamczyk}@osu.cz, jan.hula@cvut.cz

1 Introduction

This contribution describes a preliminary work focused on subgoal prediction in the puzzle game called *Ricochet Robots*. It is motivated by a broader goal of designing an agent which learns to explore complex state-spaces by predicting subgoals and trying to reach them. In the ideal case, such an agent would discover useful subgoals without any supervision but in this work we focus on a much simpler task of predicting subgoals proposed by a manually designed heuristic. These tasks serve as a sanity check that confirms that a graph neural network is capable of expressing the computation required for the subgoal prediction. The text of this contribution is structured as follows: In Section 2 we describe the mechanics of the game, in Section 3 we describe the heuristic which we use to score potential subgoals, in Section 4 we describe the machine learning task and Sections 5 and 6 contain description of experimental setup and preliminary results.

2 Problem Statement

Ricochet Robots is a puzzle board game designed by A. Randolph. The game board consists of a square grid of tiles on which movable game pieces called robots of different colors and a target tile are placed. Walls may be positioned between adjacent tiles to block robot movement. Robots can move only vertically or horizontally, and once they start moving, they stop only when they hit a wall or another robot. The goal is to cover the target tile with a robot of the matching color using the minimum number of moves. In our case, we focus on games with a single target tile.

The game is NP-complete with k robots [1], making it computationally challenging.

Each move in the game consists of selecting a robot and a direction. Before executing the next move, the previous move must be fully completed. To reach the target tile, a robot may

Acknowledgement This contribution is from the project "Research of Excellence on Digital Technologies and Wellbeing CZ.02.01.01/00/22_008/0004583" which is co-financed by the European Union. Model training and evaluation was supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90254).

https://www.doi.org/10.15452/978-80-7599-515-5.2026.10

need assistance from other robots – these serve as stoppers, allowing the robot to halt at tiles from which it can reach the goal. Helper robots may themselves depend on assistance from other robots, making the reachability problem nontrivial. For a more detailed analysis of the game mechanics, see Masseport et al. [3].

3 A Heuristic for Scoring Subgoals

Our approach uses the hierarchical search heuristic from Hula et al. [2] for scoring potential subgoals. This heuristic decomposes the problem by identifying subgoals through graph analysis.

The game state is represented as a weighted directed graph $G = (V, E)$ where nodes are grid tiles and edges are feasible robot movements. Edges receive weights: $w(e) = 1$ for moves executable without assistance, $w(e) = 10$ for dependent moves requiring a helper robot as a blocker.

The final component $\mathcal{M}$ consists of tiles from which the target robot can reach the goal using only non-dependent edges. These are computed via reverse reachability: starting from the target tile, traverse backwards through weight-1 edges only. This identifies all positions where the target robot can complete its path independently of other robots once such a position is reached.

Transition tiles $\mathcal{S}(m)$ for each tile in the final component m are adjacent tiles outside $\mathcal{M}$ from which the target robot could approach m but requires helper assistance.

The heuristic scores each tile m in the final component $\mathcal{M}$ by computing the cost of reaching the target tile from the tile m and estimating the cost of reaching m from the current state through a tile $s \in \mathcal{S}(m)$. The estimated cost of reaching $s \in \mathcal{S}(m)$ denoted by $\text{cost}(s)$ contains the cost of reaching the tile s by the target robot but also the cost of reaching the required supporting tile by the nearest helper robot. These two costs are computed as lengths of shortest paths (from the position of the target robot to a state s and form the position of a helper robot to the respective supporting tile) which allow to cross dependent edges with $w(e) = 10$. Therefore, these costs are only estimates.

Scoring: Each target tile in the final component is scored as:

$$\text{score}(m) = \text{path\_cost}(m \to \text{target}) + \min_{s \in \mathcal{S}(m)} \{\text{cost}(s)\}$$

Tiles m in the final component with $\mathcal{S}(m) = \emptyset$ receive score ∞ (unreachable). Using this scoring, we can identify the most promising subgoals: the tile with the minimum score. The complete heuristic described by Hula et al. [2] recursively expands subgoals in the order defined by the scoring above and improves the estimates of the costs as it tries to reach the given subgoal.

4 Predicting Subgoals

Using the heuristic described above, we formulate two prediction tasks:

Task 1 (Final Component Classification): Binary classification identifying whether each node belongs to $\mathcal{M}$. This serves as a sanity check, as the task should be learnable given sufficient message passing to propagate information from the target tile backwards through non-dependent edges.

Task 2 (Best Subgoal Classification): Binary classification identifying nodes with minimum score among all tiles in the final component; the most promising subgoals the target robot should reach.

Both tasks are formulated as node-level predictions, where the model must classify every tile in the grid (node in the graph) simultaneously.

5 Experimental Setup

In this section, we will describe the task representation, model architecture, training, and evaluation.

5.1 Graph Representation

Each game instance is represented as a directed graph where nodes correspond to grid tiles and edges represent feasible robot movements. Node features are 4-dimensional binary vectors encoding: [empty, helper robot, target robot, target tile]. Edge features are 4-dimensional vectors indicating directionality and dependency: [forward non-dependent, backward non-dependent, forward dependent, backward dependent], where dependent edges require helper robot assistance.

5.2 Model Architecture

We use a recurrent Graph Attention Network (GATv2) [4] with residual connections. The architecture consists of:

- An input layer: a 4-head GATv2Conv projecting 4D node features to hidden space
- A recurrent hidden layer: single 4-head GATv2Conv applied iteratively
- An output layer: a single-head GATv2Conv followed by a task specific classifier

All layers use edge features and dropout. This recurrent design enables iterative message passing across the graph structure while maintaining parameter efficiency.

We conducted hyperparameter search varying hidden dimensions $\{64, 128\}$, recurrent iterations $[5, 10]$, learning rates $[10^{-4}, 5 \times 10^{-2}]$, dropout $[0.0, 0.5]$, and weight decay $[10^{-5}, 10^{-3}]$, with fixed batch size (128) and 4 attention heads. We report averaged performance of the top 10 models per task based on test exact match metric.

5.3 Training and Evaluation

Training uses 15,000 randomly generated game instances of 16x16 grid size, from which we use 128 instances for test and 128 instances for validation sets. We train for 100 epochs with batch size 128 using AdamW optimizer and cosine annealing with 20-epoch warmup.

We evaluate using node-level metrics (accuracy, precision, recall, F1) and graph-level **exact match ratio**: the proportion of graphs where all node predictions are correct.

6 Preliminary Results

In Task 1 (Final Component Classification), we reach near-perfect performance, where the best model obtains 100% exact match on both validation and test sets using 64 hidden channels, 7 recurrent iterations, and 0.005 learning rate. Task 2 (Best Component Classification) proves more challenging, with the best model achieving 65.6% validation and 69.5% test exact match,

requiring larger capacity (128 hidden channels) and higher learning rate (0.02). Both tasks benefit from deeper recurrence (7 iterations) with 4 attention heads. The difference in model performance between tasks confirms that while identifying reachable positions is straightforward, predicting optimal subgoals requires more complex reasoning.

Table 1: Performance metrics averaged over 10 runs. Values show mean ± standard deviation.

Node-Level Metrics

Task	**Accuracy**	**Precision**	**Recall**	**F1**
Final Component	0.998 ± 0.002	0.982 ± 0.024	0.944 ± 0.069	0.961 ± 0.043
Best Component	0.998 ± 0.000	0.822 ± 0.018	0.747 ± 0.019	0.782 ± 0.013

Graph-Level Metrics (Exact Match)

Task	**Validation**	**Test**
Final Component	0.952 ± 0.042	0.961 ± 0.033
Best Component	0.663 ± 0.016	0.680 ± 0.009

This work demonstrates that graph neural networks can learn to imitate a heuristic to predict subgoals in the computationally challenging domain of Ricochet Robots, an NP-complete puzzle game. Our experiments confirm that a recurrent Graph Attention Network architecture can capture complex spatial reasoning required for subgoal identification.

We validated our architectural choices, particularly the use of recurrent message passing to propagate reachability information through the graph structure.

In future work, we plan to extend these experiments to regression tasks for direct state value prediction and compare the graph-based approach with sequence models such as autoregressive transformers.

References

[1] Birgit Engels and Tom Kamphans. Randolph's robot game is np-complete. In *Proceedings of the Twenty-second European Workshop on Computational Geometry (EWCG'06)*, pages 157–160, 2006.

[2] Jan Hůla, David Adamczyk, and Mikoláš Janota. Fast heuristic for ricochet robots. In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART*, pages 71–79. INSTICC, SciTePress, 2023.

[3] Samuel Masseport, Benoit Darties, Rodolphe Giroudeau, and Jorick Lartigau. Ricochet robots game: complexity analysis technical report. 2019.

[4] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.