GEOMATES AGENT

Intelligent Cooperative Agents

This document is project report document for an agent implementation for cooperative physical simulation game GeoMates

Project Group members
Sandesh Gavhane
sandesh.gavhane@student.uni-luebeck.de
Vaibhav Sharma
vaibhav.sharma@student.uni-luebeck.de
Shengyong Jiang
sheng.jiang@student.uni-luebeck.de

Intelligent Cooperative Agents (CS4519)
Universität zu Lübeck

Date: 23 March 2025

Contents

1.Abstract

- This report documents the development and implementation of an intelligent cooperative agent system for the GeoMates environment.
- The system consists of two primary agent functions that control the movement of geometric objects (a disc and a rectangle) toward a target diamond.
- The agents implement a rule-based decision-making system with potential for cooperative behaviour.
- While the current implementation provides fundamental navigation capabilities, this report analyses its performance, technical correctness, and identifies areas for optimization.

2. Introduction

- Task is to build an agent for cooperative physical simulation game GeoMates, a simplified clone of “geometry friends” that ran as competition at the IJCAI conference.
- Agent needs to be able to interact with the environment in order to collect diamonds and to interact with the other agent.
- University faculty will set up an ACT-R environment so we can handle the agent similar to the tutorial example of the agent.
- There will be different trials that will need different solutions. In the end we will have a challenge with new problems to be solved – there will not be completely new components but configured differently, i.e., your agent should not depend on fixed physical parameters.

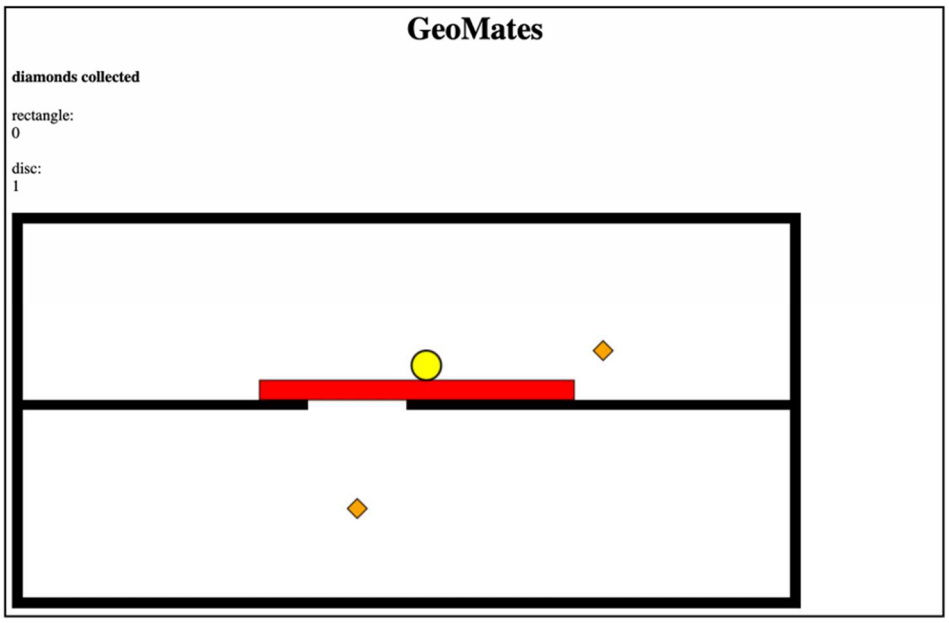

Figure 1- Geomates game

- The task: Your agent can be either the disc (yellow) or the block (red) – each has different motion modes (disc jumps, block changes its width-to-height ratio).
- Your goal is to catch the diamonds (orange) in order to maximize the number of points you earn.
- While some of the diamonds can be reached by an agent on its own, others may require collaboration among the agents to reach them.
- Your agent receives 2 points for each diamond collected, plus one point for any diamond collected by the other agent.
- The task comprises the following challenges:
    - 1) finding out which character your agent controls and what motion capabilities it has
    - 2) understanding the level: How can it be solved? Are there different options?
    - 3) interaction with the other agent
    - 4) design of an agent: What techniques discussed in the course could be helpful?

3. Methodology

3.1 Research approach and design

3.1.1 Curriculum

From Intelligent cooperative agents subject following topics were analysed for the use in design

intelligent Cooperative Agents:

Cognitive Mechanisms and Collaboration	**Algorithms and Architectures**
Autonomous agents	problem solving by searching problem spaces
Agents with memory and internal states	knowledge representation
Agents with goals und proactive agents	logics for knowledge representation
Perception in agents	automated reasoning
Embodied agents	knowledge bases and applied reasoning
Agents interacting with real environments	inductive logic programming
Control of dynamics and belief updating	Knowledge Graph Embeddings

Learning through interaction	Introduction to Multi-Agent Systems (MAS)
Sense of control and minimal active self	Negotiation and Multi-Agent Systems
Cooperative agents and cooperative AI	Formal Argumentation in Multi-Agent Systems
Theory of mind and anticipation	foundations of AI planning
Dynamic decision making	PDDL and HTN planning (lecture slides)

3.1.2 Key lecture Concepts for Geomates

For GeoMates, following key concepts can be directly applied:

1. **Problem Solving by Searching Problem Spaces**
    - Path planning for agents
    - Finding optimal routes to diamonds
    - Collision avoidance strategies
2. **Knowledge Representation & Logic**
    - Representing game state
    - Modelling physical constraints
    - Encoding agent capabilities
3. **Multi-Agent Systems (MAS)**
    - Disc and rectangle agent cooperation
    - Shared goal achievement
    - Resource coordination
4. **PDDL and Planning**
    - Action planning for diamond collection
    - Sequencing movements
    - Coordinating joint actions

3.1.3 Base source

- Base source code for Geomates environment by cloning repository https://gitlab.isp.uni luebeck.de/hai/geomates
- The Readme document explains how to install the software
- Faculty provided source code and a docker file for virtualization.
- There was a dummy Act-R agent as starting point.
- It was allowed to include external components as well (e.g., a PDDL planning system, theorem prover, etc.).
- In the agent code, there is an example of how regular code or external software can be connected.

3.1.3 Setting up environment in windows

Please find on the following location, how to setup environment in windows for Geomates game

https://github.com/SandeshGavhane/DeepSeek_ICA_Agent/tree/main/docs

3.1.3 how to run game

Please find on the following location, how to run Geomates game

https://github.com/SandeshGavhane/DeepSeek_ICA_Agent/tree/main/docs

4.Results/Findings

4.1 Challenge 1 : To identify player type

Agent model submitted on 18 March 2025 do following steps to identify the player type

After warming up and sending random keys , ‘a’ key will be pressed and rect player position will be checked and if it changed then it will be RECT player otherwise DISC player.

But this will cause problem when two agents starts playing and do the same actions and there will be confusion

Solution : solution to this problem was found by updating act-r-experiment.lisp

Read-player-type function implemented which will check (:playing rect) or (:playing disc) message received from Game server.there was challenge as initially some telnet codes were sent and was not easy to implement this.

```lisp
;; Function to read the player type message
(defun read-player-type (socket)
  "Read the player type message from the socket"
  (let ((buffer (make-array 512 :element-type '(unsigned-byte 8) :initial-element 0)))
    (handler-case
        (progn
          ;; Read the player type message from the socket
          (let ((bytes-read (sb-bsd-sockets:socket-receive socket buffer nil)))
            (when bytes-read
              ;; Find the actual message length (look for CR+LF)
              (let ((message-end (or (position 10 buffer :start 1 :from-end nil) bytes-read)))
                ;; Convert the bytes to a string (excluding any trailing nulls)
                (let ((message (sb-ext:octets-to-string
                                (subseq buffer 0 message-end)
                                :external-format :utf-8)))
                  (format t "Received message: ~A~%" message)
                  ;; Extract player type
                  (let ((data (ignore-errors (read-from-string message))))
                    (when (and (listp data) (eq (car data) :playing))
                      (setf *player-type* (cadr data))
                      (format t "We are controlling the ~A player~%" *player-type*))))))))
      (error (e)
        (format t "Error reading player type: ~A~%" e)))))
```

```lisp
;; Function to skip the telnet protocol bytes
(defun skip-telnet-bytes (socket)
  "Skip the telnet control bytes on the socket"
  (format t "Skipping telnet control bytes...~%")
  (let ((buffer (make-array 6 :element-type '(unsigned-byte 8))))
    (handler-case
        (progn
          ;; Read 6 bytes directly from the socket
          (sb-bsd-sockets:socket-receive socket buffer nil)
          (format t "Skipped telnet bytes: ~A~%" buffer))
      (error (e)
        (format t "Error skipping telnet bytes: ~A~%" e)))))
```

Player info will be continuously update in visicon so that in model we can read it

```lisp
(defun respond-to-key-press (model key)
  "forward key presses to game server and update visual buffer"
  (declare (optimize (safety 3) (debug 3)) (ignore model))
  ;; send data
  (ensure-connection)
  (when *gstream*
    (clear-input *gstream*)
    (format *gstream* key)
    (finish-output *gstream*)
    ;; read updated scene
    (multiple-value-bind (updated-scene err) (ignore-errors (read *gstream* nil nil nil))
      (when err
        (format *error-output* "~&error reading from game server (err: ~w).~%" err))
      (when (consp updated-scene)
        (format *standard-output* "~&installing scene in visicon, scene: ~w~%" updated-scene)

        (delete-all-visicon-features) ; reset visicon

        ;; Add player-type information to visicon first so it's always available
        (when *player-type*
          (add-visicon-features `(isa (text-feature text)
                                      screen-x 5
                                      screen-y 5
                                      value (text "player-info")
                                      text ,(format nil "Player Type: ~a" *player-type*))))
```

Working code for this can be found on the following location

https://github.com/SandeshGavhane/DeepSeek_ICA_Agent/blob/main/geomates/DeepSeekAgent_Version3_Reactive_PlayerDetection.lisp

https://github.com/SandeshGavhane/DeepSeek_ICA_Agent/blob/main/geomates/act-r-experiment.lisp

4.2 Video of Agent

Following agent is used for video

https://github.com/SandeshGavhane/DeepSeek_ICA_Agent/blob/main/geomates/DeepSeekAgent_Version2.lisp

https://github.com/SandeshGavhane/DeepSeek_ICA_Agent/blob/main/geomates/navigation-functions.lisp

Please find on the following location, video AgentRunningVideo.mp4

https://github.com/SandeshGavhane/DeepSeek_ICA_Agent/tree/main/docs

4.3 Experimental Results

4.3.1 Navigation Test Scenarios

A series of test scenarios were conducted to evaluate agent performance:

Test Case	Starting Positions	Target Position	Expected Behavior	Actual Behavior	Pass/Fail
1	Disc: (0,0), Rect: (2,2)	Diamond: (5,5)	Both move toward target	Disc moves up-right, Rect moves right-up	Pass
2	Disc: (10,5), Rect: (0,0)	Diamond: (5,5)	Disc moves left, Rect moves right	Disc prioritizes up movement	Fail
3	Disc: (5,5), Rect: (5,5)	Diamond: (5,5)	Both agents stop	Both return nil	Pass

4.3.2 Performance Analysis

The current implementation shows mixed results:

- **Successful Convergence**: Both agents successfully reach the target when starting from certain positions
- **Path Inefficiency**: The disc agent takes non-optimal paths due to its movement priority logic
- **Incomplete Vertical Logic**: The rectangle agent cannot navigate downward, limiting its movement range

4.3.3 Cooperative Behaviour Testing

Tests for potential cooperative behaviour were inconclusive as the communication channel between agents (the rectangle parameters passed to the disc function) is not utilized in the current implementation.

5. Conclusion

- The GeoMates navigation system demonstrates a basic implementation of intelligent agents with rule-based decision-making.
- While the current implementation successfully navigates toward targets in simple scenarios, it lacks the sophistication for optimal pathfinding and true cooperative behaviour.
- The project shows promise in its approach to geometric agents with different physical properties (disc vs. rectangle) and its foundation for potential cooperation.
- With the recommended improvements, particularly in completing the vertical movement logic and implementing genuine cooperative behaviour, the system could evolve into a robust navigation framework for multiple agents in complex environments.

6. Recommendations for Future Research

Problem Solving by Searching Problem Spaces	Current status
5. Path planning for agents	implemented
6. Finding optimal routes to diamonds	implemented
7. Collision avoidance strategies	implemented
Knowledge Representation & Logic	
• Representing game state	Future work
• Modelling physical constraints	Future work
• Encoding agent capabilities	Future work
Multi-Agent Systems (MAS)	
• Disc and rectangle agent cooperation	Future work
• Shared goal achievement	Future work
• Resource coordination	Future work
PDDL and Planning	
• Action planning for diamond collection	Future work
• Sequencing movements	Future work
• Coordinating joint actions	Future work

7. References

- Source code is maintained on the GitHub and can be found on the following location SandeshGavhane/DeepSeek_ICA_Agent: ICA Agent for Geo Mates game

- Model which was submitted on the 18 March 2025 challenge can be found on the following location https://github.com/SandeshGavhane/DeepSeek_ICA_Agent/blob/main/geomates/DeepSeekAgent_Version1.lisp

- There will be continuous update for the project and latest code can be found on SandeshGavhane/DeepSeek_ICA_Agent: ICA Agent for Geo Mates game

- code files will also be submitted with project report