Capture the flag simulation in Netlogo using GAIA design methodology using BDI model

Ernesto Torres and Alan Torres

***Abstract*—Multiagent systems (MAS) are composed of several intelligent agents that interact with each other. MAS include problem solving that can be difficult for a single agent, also allowing decentralization. Each agent is at least partially autonomous. There are many MAS coordination techniques, in this paper we present the use of Contract-Net for task assignment. The system is modeled and simulated in NetLogo[6], a powerful multi-agent programmable modeling environment.**

The simulation consists in a capture the flag model that allows two teams to compete with each other. The red team implements contract-net for task assignment such as captain assignment and flag defending. Green team is programmed only with basic behaviors such as patrol and defend. Both teams implement primitive behaviors such as move and attack, also a capture the flag behavior for each captain.

Each agent in the model has beliefs based on their own environment observation, based on their beliefs they produce intentions and select an action to commit to.

For this simulation we used GAIA design methodology[4] because it based in role interaction. We used the BDI model to design a belief structure to allow each agent to decide which action is more appropiate. Agents generate intentions and commit to make one of them each cycle.

I. Introduction

THIS paper is intended to demonstrate the effectiveness of task assignment using contract-net. The domain selected is a capture the flag simulation to make a graphical demonstration of two teams playing, in which only the red team implements coordination and the green teams follows only simple behaviors. Each agent is responsible of keeping its own state of the world and committing to do an action.

Contract-net was selected as the multi-agent coordination technique. This allows an agent to become a contractor and to broadcast the task that needs helps with. Contract-net is used in two cases: when selecting assigning the role of captain and when the red flag is asking for defenders.

GAIA methodology and BDI model are used together to make a more robust multi-agent system. For extending GAIA methodology to use BDI model, extra activities and protocols are implemented for each agent. This will be explained further in the document.

For assigning the role of captain, the red flag broadcast the task and all freeagents send their own bid with their offer. This allows to select the best agent that is nearest the enemy flag to minimize the travel time to enemy base.

For assigning the defenders, a "box" strategy is made. The flag makes a contract for each patch that needs defending and it is assigned to the best bidder consisting in minimum distance.

This paper consist in the following sections: analysis and design, prototype, experiments and results, conclusion and references.

Analysis and design covers the system organization, roles model, protocols used, agent model and acquaintance model.

Prototype section describes the model in Netlogo, including how to run the simulation, user interface description. This information can also be found embedded in the nlogo file in the Information section.

Experiments and results presents the experimental data that is obtained by executing the model. It demonstrate the advantage a coordinated team has against a basic behavior team.

Conclusions show lessons learned and also includes observations.

May 5, 2010

II. GAIA and BDI

GAIA is a way of viewing multi-agent systems as a computational organization, identifying roles and their interaction. Each role has its own set of responsibilities, activities, protocols and permissions. To extend the methodology we used the BDI model to allow each agent to have their own sets of beliefs and intentions. Each agent is capable of updating their own beliefs according to how they see the world. Then they can create intentions according to their beliefs and commit to one intention each execution cycle. To allow this each agent is provided with a protocol named update beliefs in which they ask other agents for information in order to create intentions. Also an activity is provided, update intentions. After their beliefs are updated, each agent decides what intention can be generated according to the environment state. A limited number of intentions are allowed according to their roles.

An example is the role flag defender:
(Update beliefs · Update intentions · ([Change role] | [Move] | [Patrol] | [Defend]))$^{\omega}|Die$

First it updates beliefs, then updates intentions and will decide to either change role, move, patrol or defend. This cycle will repeat until game ends or the agent dies.

III. Analysis and design

GAIA methodology

1) System Organization
   - The environment is discrete, sequential, dynamic, deterministic, fully observable and multi-agent. Meaning that all the agents positions are known at all time. There will be two teams with a defined number of players and there will be one flag agent

for each player. Agents can only communicate with other agents on the same team.

2) Roles Model

- **Name:** Flag
  - Description: Item for each team
  - Protocols:
    - Update beliefs
    - Broadcast task
    - Assign captain
    - Assign defenders
    - Assign bodyguards
  - Activities:
    - Update intentions
    - Win
    - Lose
  - Responsibilities:
    - Liveness: Flag = (Update beliefs · Update intentions · ([Broadcast task] | [Assign captain] | [Assign defenders] | [Assign bodyguards]))$^{\omega}|Win|Lose$
    - Safety:
      - Do not allow moves after Lose | Win
    - Permissions:
      Change
      Status //Indicate if is captured or in base
      Position //Indicates position on the map
- **Name:** Captain
  - Description: Head of the team and in charge of assigning the roles. Capture opponents flag and take it to base.
  - Protocols:
    - Update beliefs
    - Capture flag
  - Activities:
    - Update intentions
    - Move
    - Die
  - Responsibilities:
    - Liveness: Captain = (Update beliefs · Update intentions · ([Move] | [Capture flag]))$^{\omega}|Die$
    - Safety:
      - PlayerPosition must be unique and other player cannot be in the same PlayerPosition.
      - MyPosition can change only 1 step in any direction
      - No valid moves for this player after death.
    - Permissions:
      Reads
      PlayerPosition // gets x,y of desired player
      PlayerLifeLevel // gets life level of desired player

      Changes
      MyPosition(x,y) // Position of the agent in the map
      MyLife // Updates current life level
      OpponentFlagStatus // Update if captain has flag or not
- **Name:** Flag defender
  - Description: Make sure the team's flag is not captured
  - Protocols:
    - Update beliefs
    - Attack
    - Patrol
    - Defend
    - Change role
  - Activities:
    - Update intentions
    - Move
    - Die
  - Responsibilities:
    - Liveness: Flag defender = (Update beliefs · Update intentions · ([Change role] | [Move] | [Patrol] | [Defend]))$^{\omega}|Die$
    - Safety:
      - PlayerPosition must be unique and other player cannot be in the same PlayerPosition.
      - MyPosition can change only 1 step in any direction
      - Attack only one enemy at a time at patch ahead
      - No valid moves for this player after death.
    - Permissions:
      Reads
      PlayerPosition // gets x,y of desired player
      PlayerLifeLevel // gets life level of desired player

      Changes
      MyPosition(x,y) // Position of the agent in the map
      MyLife // Updates current life level
      OpponentFlagStatus // Update if captain has flag or not
- **Name:** Bodyguards
  - Description: Make sure the captain captures opponent's flag and return safely to base.
  - Protocols:
    - Update beliefs
    - Attack
    - Patrol
    - Defend
    - Change role
  - Activities:
    - Update intentions
    - Move
    - Die
  - Responsibilities:

    * Liveness: Bodyguards = (Update beliefs · Update intentions · ([Change Role] | [Move] | [Patrol] | [Defend]))$^{\omega}|Die$
    * Safety:
      - PlayerPosition must be unique and other player cannot be in the same PlayerPosition.
      - MyPosition can change only 1 step in any direction
      - Attack only one enemy at a time at patch ahead
      - No valid moves for this player after death.
    * Permissions:
      Reads
      PlayerPosition // gets x,y of desired player
      PlayerLifeLevel // gets life level of desired player

      Changes
      MyPosition(x,y) // Position of the agent in the map
      MyLife // Updates current life level
      OpponentFlagStatus // Update if captain has flag or not
- **Name:** Freeagent
  - Description: Initial role for non-flag agents
  - Protocols:
    * Bid
    * Change role
  - Activities:
    * Get in random position
  - Responsibilities:
    * Liveness: Freeagent = (Get in random position+ · Bid+ · Change role)
    * Safety:
      - Starting position is random, but will respect only one agent per patch and each team can only sprout in their side of the field
    * Permissions:
      Reads
      Task //Task offered by flag
      Changes
      MyOffer(x,y) // offer for bid

3) Interaction Model, protocol definitions.
   - Protocol: Attack
     - Purpose: damage a player
     - Initiator: Attacker
     - Responder: Victim
     - Inputs: Position (x,y) of the victim
     - Outputs: Victim's new reduced life level
     - Processing: Victim life level - attack
   - Protocol: CaptureFlag
     - Purpose: Capture flag
     - Initiator: Captain
     - Responder: Opponent's Flag
     - Inputs: Position (x,y) of the captain
     - Outputs: Flag status changed
     - Processing: Check that the captains position is the same as the opponent's flag position
   - Protocol: Bid
     - Purpose: Make offer for a task
     - Initiator: Flag
     - Responder: Freeagent
     - Inputs: task
     - Outputs: offer
     - Processing: Make offer according to task requirements
   - Protocol: Change role
     - Purpose: Change to another role (captain, bodyguard or flagdefender)
     - Initiator: Flag
     - Responder: Any agent
     - Inputs: role
     - Outputs: Agent changes role
     - Processing: Flag requirements for roles and if captain dies, a new one needs to be selected
   - Protocol: Update beliefs
     - Purpose: Updates how agent sees the world
     - Initiator: Each agent
     - Responder: Any agent involved in perception
     - Inputs: Percept
     - Outputs: belief updated
     - Processing: Updates belifs according to percepts
   - Protocol: Patrol
     - Purpose: Patrol around captain or flag
     - Initiator: Flags or captains
     - Responder: Flagdefenders or Bodyguards
     - Inputs: Flag or captain position
     - Outputs: Valid patches to move to
     - Processing: Radius to flag or captain
   - Protocol: Defend
     - Purpose: Defends captain or flag
     - Initiator: Flags or captains
     - Responder: Flagdefenders or Bodyguards
     - Inputs: Flag or captain enemies near
     - Outputs: list of enemies near flag or captain
     - Processing: Enemies in a Radius to flag or captain
   - Protocol: Broadcast task
     - Purpose: Offer a task
     - Initiator: Flags
     - Responder: Freeagents or Flagdefenders or Bodyguards
     - Inputs: task
     - Outputs: list of offers
     - Processing: Generate list according to bidders
   - Protocol: Assign captain
     - Purpose: Assign captain
     - Initiator: Flags
     - Responder: Freeagents or Flagdefenders or Bodyguards
     - Inputs: No captain

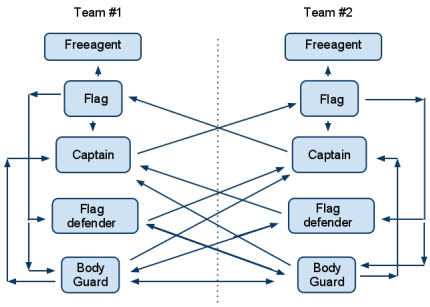

Fig. 1. Acquintance Model

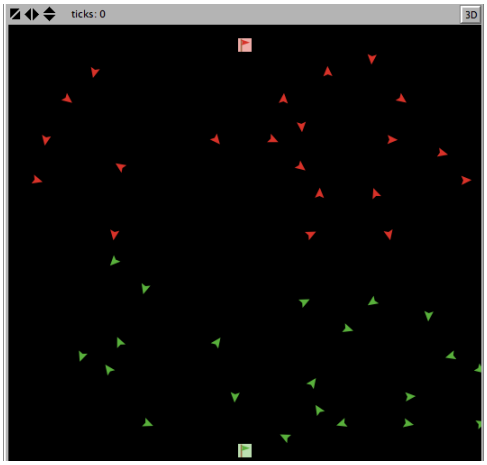

Fig. 2. Model setup

- Outputs: new captain
- Processing: Calculate best agent for task

- Protocol: Assign defenders
  - Purpose: Assign defenders
  - Initiator: Flags
  - Responder: Freeagents
  - Inputs: No defenders
  - Outputs: new defenders
  - Processing: Calculate best agents for task
- Protocol: Assign bodyguards
  - Purpose: Assign bodyguards
  - Initiator: Flags
  - Responder: Freeagents
  - Inputs: No bodyguards
  - Outputs: new bodyguards
  - Processing: Calculate best agents for task

4) Activities description.
   - Activity: Move
     - Description: Change player's current position
   - Activity: Die
     - Description: Remove player when life level reaches zero.
   - Activity: Win
     - Description: End game with positive results
   - Activity: Lose
     - Description: End game with negative results
   - Activity: Get in random position
     - Description: End game with negative results
   - Activity: Update intentions
     - Description: Update internal intentions according to beliefs
5) Agent Model
   - Captain: 1 per team
   - Flag: 1 per team
   - Flag defender: +
   - Bodyguards: +
6) Acquintance Model
   See figure #1

IV. Prototype

A. Description

This is a capture the flag simulation between two teams, red vs green (see figure #2). The objetive of this simulation is to prove the advantajes of having multiagent coordination. Red team implements contract-net to assign roles for captain, bodyguards and for flagdefenders for making a "box" cover for their flag. Green team has only basic behaviors such as patrol and defend. Green also has the advantaje of inmediatelly moving, while Red takes some time for assigning roles.

There are five role types: freeagents, flags, captains, bodyguards and flagdefenders.

- Flags: One flag for each team and has its position fixed at game start. Has status ("captured" or "in-base"). Flags decide what roles should be assigned to the freeagents.
- Freeagents: At the beginning of the game all agents start as freeagents, meaning that any role can be assigned to them.
- Captains: The only agent capable of capturing enemy flag. It Can't attack other agents.
- Bodyguards: Protect captain from enemies by patrolling around captain and attacking any enemy near their captain.
- Flagdefenders: Protect the flag and attack all approaching enemies.

Both teams have basic capabilities such as: move and attack.

B. Agents

42 total agents:

- 21 per team
  - 1 flag
  - 20 freeagents

C. Game rules

- When enemy flag is captured by a captain and returned to their own base the game ends.
- Only captain can capture flag. A new captain is assigned if he dies.

- Only bodyguards and flagdefenders can attack, but both roles can attack and kill captains.
- If a captain carrying a flag is killed, the flag returns to its base automatically.
- It is not necessary to have own flag in base to end the game. This meaning that the only and principal goal is to retrieve enemy flag.

D. How it works

The execution is a loop that includes the following:

- agent-loop-flags : Roles are assigned here. Also contracts are broadcasted by the flags.
- agent-loop-captain : Captain actions for capturing flag and returning to base
- agent-loop-def : Defending, patrolling and formations.
- agent-loop-bg: Patrolling and Captain protection.
- agent-loop-freeagent : Freeagents waiting for role to be assigned

E. Roles assignment

1) Red Team:
   - Captain: Nearest agent to flag with higher life
   - Flagdefenders: 7 freeagents to surround flag
   - Bodyguards: The rest of the freeagents
2) Green Team:
   - Captain: Random agent
   - Flagdefenders : 10 random agents
   - Bodyguards: The rest of the freeagents

F. How to use it

Type the number of times you want to execute the model in to the Loops text-box. Press start and the simulation will begin.

Two monitors are used for displaying the number of agents in each team during execution. This is only useful when executing only one loop.

When executing more than one loop, the plot to the right of the model is recommended, as it shows graphically the agents that remain alive at the end of the cycle.

Game Time is also a plot what displays the ticks spent for each game round.

Victories are shown in monitors to the right of the model. Each green or red win is counted and displayed.

G. Extending the model

At the moment attack coordination was not implemented, only flag defending and roles assignment. Future work may include implementing another type of coordination to allow bodyguard block and attack enemies more efficiently.

H. BDI agents

The major effort was to use the BDI model in Netlogo, for this, some research was made and functions were created based in the documents: Agents with Beliefs and Intentions in Netlogo[1] and Symbolic, intentional and deliberative agents[2].

Functions created include:

- add-intention [new-intention]
- add-belief [new-belief]
- clean-beliefs
- i-have-intention? [intention]
- i-have-belief? [belief]

I. Agent control loop

while true

1) observe the world;
2) update internal world model;
3) deliberate about what intention to achieve next;
4) use means-ends reasoning to get a plan for the intention;
5) execute the plan

end while

J. Contract-net

In this model contract-net is used for task assignment. For this task some functions were created to simulate the exact behavior of contract-net.

This includes: Broadcast, Bidding and Awarding.

Broadcasting sends request in a message package: [task requirement1 reference1 requirement2 reference2 contractor]

Bidders receive task and bid by sending their offers.

Contractor then picks the higher bid and awards it with the contract.

V. Experiments and results

After the model was completed we created a special start that is able to run any number of matches. This allows to view the behavior of both team in a large number of matches. We run 100 matches with red team winning 94 while green winning only 6 matches, this meaning that coordinated team won 94% of the times. Game time was stable at 75-80 ticks approximately with exception of a few matches with peaks around 150. The survivor count of both oscillates around 15 survivors per time, but the red team getting the lowest count per match. We should remember that the main goal of each match is to capture the enemy's flag. Results are displayed in figure #3.

Another set of simulations were executed, this time 1000 matches were played. Red winned 916 and green 84. Red win percentage is still above 90%. Here since a large number of matches were played we can observe that there were some peaks in game time, reaching about 495 ticks. In that case more ticks were required since many agents died when carrying the flag and the needed to return again for the flag. Results are displayed in figure #4.

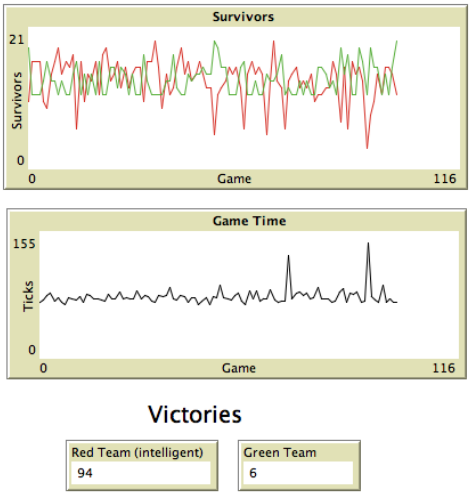

Fig. 3. 100 matches result

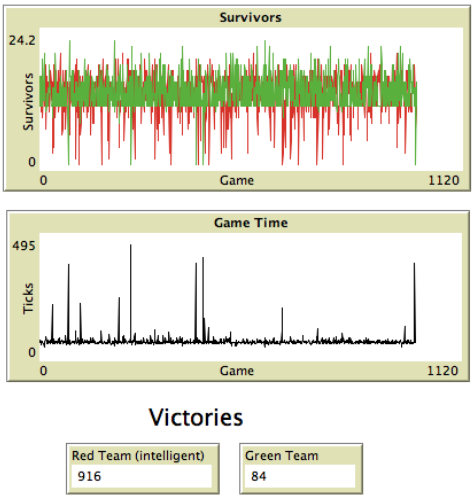

Fig. 4. 1000 matches result

VI. Conclusion

This paper proves that a coordinated team gets better results when competing against a non-coordinated team with basic behaviors. Contract-net was used as task assignment, and has proven effective. Although coordination only was used for role assignment and ”box” defending, it proved it is good enough to win more than 90% of matches. While the simulation results are more than satisfactory a lot of work can still be done. This simulation tried to be simple, but still covers the major roles for a capture the flag game. We are pleased with the results obtained during the programming of this model during the course of Multi-agent systems, unfortunately the game is very complex and a lot of work can still be done. Also the duration of the course is not time enough to take full advantage of the model.

This simulation can be upgraded by adding attack coordination and will surely reach even better results. This may be implemented in future work and can be used as a thesis topic to make the simulation more complex. Considering variables such as variable damage, obstacles and fog of war can really make the game more realistic and challenging.

References

[1] Ilias Sakellariou. *Agents with Beliefs and Intentions in Netlogo*. March 2004, Updated March 2008
[2] Symbolic, intentional and deliberative agents PDF presentation in Multi-agent systems course
[3] Carmen Fernndez Chamizo, Jorge Gmez Sanz, Juan Pavn Mestras. *Metodologas de desarrollo de sistemas multi-agente*. http://grasia.fdi.ucm.es
[4] Michael Wooldridge. *The Gaia Methodology for Agent-Oriented Analysis and Design*
[5] Agent Methodologies.Agent-based systems course. With material from:Adina Magda Florea
[6] NetLogo. http://ccl.northwestern.edu/netlogo/