AD (Attacker Defender) game

in the Information Dynamics Framework

(DRAFT)

Andrzej P. Kochut, Ashok K. Agrawala, Ron Larsen, A. Udaya Shankar

{kochut,shankar,agrawala}@cs.umd.edu,
rlarsen@deans.umd.edu

Institute for Advanced Computer Studies
Department of Computer Science
University of Maryland
College Park, Maryland 20742

June 2001

1. Introduction

Information Dynamics is a framework for agent-based systems that gives a central position to the role of information, time, and the value of information. We illustrate system design in the Information Dynamics Framework by developing an intelligence game called AD involving attackers and defenders operating in some space of locations. The goal of the attackers is to converge in an arbitrary location in such a way that the number of defenders at this location at that time is minimized. The goal of defenders is to prevent attackers from achieving their goal.

We developed a simulator for the game that allows users to try different action sets, strategies and value functions. If you want the software, please email us.

2. AD game

The AD game is played on a rectangular board (Figure 1). Each location is identified by x,y coordinates. The board is populated with intelligent agents of two types: attackers and defenders. At any instant an agent is at a location and has a perceived view...
of the world, which includes accurate knowledge of positions of agents of the same type and inaccurate knowledge of positions of agents of the opposite type. The closeness of the estimation to the real world measures the quality of the agent's strategy. The value of a piece of information is how much this information improves the agent's perceived world in the sense of closeness to the real world.

The agent can perform the following kinds of actions:

1. Move from the current position on the board to one of the neighboring positions
2. Scan any given location on the board. The result of the scan operation is a number, called scan value, representing the likelihood that the location is "infested" by agents of opposite type. The scan value is a random from [0,1], with higher values being returned when the location has larger number of agents of opposite type
3. Communicate with the other agents of its type and obtain their locations of

![Figure 1. Attackers (ovals) and defenders (rectangles) on the board.](image)

The goal of the attackers is to converge in an arbitrary location in such a way that the number of defenders at this location is as small as possible. The goal of defenders is to prevent attackers from achieving their goal. The agents execute concurrently and asynchronously.

The architecture of the system follows the rules of the information-centric approach. Each agent maintains its own perceived view of the world. The actions of an agent are based solely on the agent's perceived view, and not that of the real world. The only way for an agent to obtain new information is to scan board locations and communicate. However recall that the results obtained in scans are only an approximation of the real situation.
3. Simulator

In the simulator we assume discrete-time operation. At each tick every agent can do multiple scans, communicate, do one move and update its perceived world. The simulator allows the user to compare the quality of different agent strategies, i.e. the proximity between the achieved perceived world and the real world. The user defines function that takes the real world and the perceived world and computes the value representing the closeness of two worlds (Figure 2). The user can log results to a text file or observe it on the graphical display.

The simulator is written in C++. It's main classes are:

- simulator - encapsulates all the general discrete-event simulation functions and data structures, such as event queue
- realWorld - contains the state of the real world (positions of agents on the board), as well as implements all primitive real world functions (move, scan etc.)
actionSet - agent's interface to the realWorld. Defines the set of actions that an agent can perform. Does not contain any state - it is just a function interface.

agent - represents an agent. Contains state associated with the agent. The most important information stored in the class are:

- agent type - defender/attacker
- unique identifier
- agent's perceived world
- handler to the actionSet (function interface to the realWorld)
- function containing logic of the agent. The logic implements agent's strategy.

perceivedWorld - contains agent's view of the current world. The specific information that appears here depends on the actual agent's logic. In general the agent can store here all information that it obtains during the game. This is the base on which agent's decisions are made.

4. Example strategies

We examine the behavior of the system for one attacker strategy and three different defender strategies. We considered the following attacker strategy:

**Attacker strategy** - Attackers perform random moves for a given period of time. Then in order to converge, attacker looks around for the closest agent of its kind and starts moving in this direction.

We considered the following defender strategies:

**Simple average** - Each agent maintains for each location the history of the performed scans. To estimate the current state of the location, the agent computes the average of the scan values obtained in the previous scans. Whenever this average exceeds a threshold, the agent marks the location as a potential convergence point. Then the agent moves in the direction of a randomly chosen potential convergence point.

**Adaptive strategy** - Each agent maintains for each location the history of the values returned in the performed scans. To estimate the current state of the location, the agent checks the history to figure out whether the scan value at the location is increasing with time. If it is, the location is marked as a potential convergence point.

**Area strategy** - In addition to performing the adaptive strategy, the agent computes the number of neighboring locations for which the scan value exceeds a threshold. Then the agent modifies the result obtained in the adaptive strategy by a function depending upon the scan values from neighboring locations.

A "difference" function $D$ is defined for a defender as the number of locations for
which its perceived world indicates small number of attackers at this location and the real world indicates large number of attackers at this location. An "average difference" function $DA$ is defined as an average $D$ over all defenders.

Figure 3 illustrates the behavior of the system with defender agents executing simple average strategy. Until time around 2000, attackers move randomly. At time 2258 the attackers converge but the defenders' estimation of the real world is inaccurate causing them to not be at the attackers' convergence point. Figure 4 shows the system behavior with defenders executing the adaptive strategy. In this case the defenders' estimation of the real world state is much better. At the time of convergence nearly all defenders are at the attackers' convergence point. Figure 5 illustrates behavior of the system with defenders executing area strategy. It is similar to the adaptive strategy case.

5. Conclusions

<TBD>

6. References

1. Agrawala, A., Larsen, R., Szajda, D., Information Dynamics: An information-centric approach to system design
Figure 3. Defenders executing *simple average strategy*: Snapshots of real world board state (attackers in black and defenders in gray) and average difference function value at different time instants.
Figure 4. Defenders executing *adaptive strategy*: Snapshots of real world board state (attackers in black and defenders in gray) and average difference function value at different time instants.
Figure 5. Defenders executing *area strategy*: Snapshots of real world board state (attackers in black and defenders in gray) and average difference function value at different time instants.