AD (Attacker Defender) Game *

Andrzej Kochut[1], Ashok K. Agrawala[1], Ronald L.Larsen[2], A. Udaya Shankar[1]

[1]Department of Computer Science,
and Institute for Advanced Computer Studies
University of Maryland
College Park, Maryland 20742
{kochut, agrawala, shankar}@cs.umd.edu

[2]Maryland Applied Information
Technology Initiative
University of Maryland
College Park, Maryland 20742
rlarsen@deans.umd.edu

CS-TR-4265
UMIACS-TR-2001-45

January 31, 2002

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, defenders and targets operating in some space of locations. The goal of the attackers is to destroy all targets. Target destruction takes place when the number of attackers in the target's neighborhood exceeds the number of defenders in this neighborhood by a value WINNING_DIFFERENCE. The goal of defenders is to prevent attackers from achieving their goal.

The model that we present has attributes that, when appropriately used, can generate either statically or dynamically features such as:

- hierarchy of agents (e.g., information collectors and controllers)

*This work is supported partly by DARPA/ Rome Labs, Department of the Air Force, under contract F306020020578 to the Department of Computer Science at the University of Maryland. The views, opinions, and/or findings contained in this report are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the Department of Air Force, DARPA, DOD, or the U.S. Government.

- information fusion
- restricted communication models

2 Model

2.1 Space

The game is played on a rectangular board of size XMAX by YMAX. Each location L is identified by its (x,y) coordinate, $0 \leq x < \text{XMAX}$, $0 \leq y < \text{YMAX}$, and can be in one of two states, accessible or inaccessible. Accessible locations can be occupied by one or more entities involved in the game. Inaccessible locations can not be occupied by any entity. For a given location L we define neighborhood of L as:

$$\text{neighborhood(L)} = \{\text{all accessible locations X for which } |\text{X.x} - \text{L.x}| \leq 1 \text{ and } |\text{X.y} - \text{L.y}| \leq 1 \text{ and not } (\text{X.x} = \text{L.x and X.y} = \text{L.y})\}$$

2.2 Entities involved in the game

The board is populated with following entities:

- attacker agents
- defender agents
- targets

Each agent has a unique identifier. The parameters ENTITY_IDS and INITIAL_POSITIONS describe, respectively, the identifiers and initial positions of the entities on the board. At each instant an entity occupies one accessible location of the board. An accessible location can have one of the following configurations of entities at any instant:

- empty
- one or more attacker
- one or more defender
- a target

2.3 State of the agent

The state maintained at each agent contains a set of variables encoding the information that the agent gathered, for example, the time-stamped history of scan results, received messages, and results of move operations. Moreover the agent's state may contain any information that agent derived from its state.

2.4 Allowed actions for attackers and defenders

An agent at location L can perform following actions:

- Move(loc): to a location *loc* in its neighborhood. The operation returns boolean value of true when the move succeeded and false when the operation failed (due to the location being inaccessible, occupied by a target, or occupied by agents of different type).
- Scan(loc): a neighboring location. The result of the scan operation consists of following data:
  - occupancy type - specifies the type of entities at the location. Possible values are:
    - 0 - empty location
    - 1 - location occupied by the agents of the same type
    - 2 - location occupied by the agents of the opposite type
    - 3 - location occupied by a target
    - 4 - inaccessible location
  - expected number of entities at the location (used for occupancy type 2)
  - maximal deviation from the expected value (used for occupancy type 2)

  There is a function SCAN_RESULT_FUNCTION that defines distribution of values.
- Chat(agent_id(s)): communicate with other agents. An agent can send one message to CHAT_DEGREE other agents of its kind. The message may contain part or all of the perceived world information of the agent. The message is sent using the destination agent's identifier as the receiver address. An agent can receive multiple messages from other agents of its kind.

2.5 Strategy of the agent

Each agent defines a STRATEGY_FUNCTION that implements its logic of execution. The function may make use of the agent's state and perform actions allowed for the agent.

2.6 Allowed target actions

At each instant a target is located at a board location. The target can move to a neighboring empty location. Each target has a distribution function TARGET_MOVE_DIST defining the probability of target to move to one of neighboring locations or to stay at the current location.

Notes: The TARGET_MOVE_DIST function may depend on location or its neighborhood. For example, the function may define equal probabilities for 8 neighboring locations and currently occupied location. On the other hand we can have the distribution preferring moves in a given direction.

2.7 Goal of the attackers

The goal of the attackers is to destroy all targets. The destruction of a target takes place when the number of attackers in the target's neighborhood exceeds the number of defenders in this neighborhood by WINNING_DIFFERENCE.

2.8 Goal of the defenders

The goal of the defenders is to prevent attackers from destroying targets.

3 Sample strategies

3.1 Attacker's strategy

Each of attacker agents executes following strategy:

- move randomly searching for a target
- once the target is sensed decide whether to chase. To do so agent performs random selection between two choices: follow the target or ignore the target.
- if the attacker decides to follow the target it starts moving in direction of board position where the target was last sensed. Moreover the attacker sends messages to other attackers giving them coordinate of the current target's location.
- each agent that is not currently chasing a target and receives message from other agent starts moving towards the direction specified in the message
- the chase ends once either the chased target is destroyed or a specified timeout expires

3.2 Defender's strategy I

Each of defender agents executes following strategy:

- move randomly searching for a target
- once the target is sensed the defender remains near the target and sends messages to other defenders which include endangered target's location
- each defender that is not currently protecting any targets and receives the message starts moving towards the location specified in the message
- the defense stops only if the target is destroyed. In this case defender starts searching for new target.

3.3 Defender's strategy II

Each of defender agents executes following strategy:

- move randomly searching for a target
- once the target is sensed and the number of other defenders in the neighborhood is smaller than specified value the defender remains near the target

- if the defender finds out that the number of attackers exceeds the number of defenders in the target's neighborhood, it sends messages to other defenders which includes endangered target's location
- each defender that is not currently protecting any targets and receives the message starts moving towards the location specified in the message
- the defense stops once the number of attackers in the target's neighborhood becomes smaller than the number of defenders in this area

4 The simulator

We developed a software package that may be used to perform simulation of agents' behavior in the environment described above. The user specifies strategy functions for each entity taking part in the game and the system performs discrete simulation. Figures 1 and 2 are examples of the board situations as seen in the user's interface. Figure 1 shows execution with defenders using strategy I, while Figure 2 shows execution with defenders using strategy II.

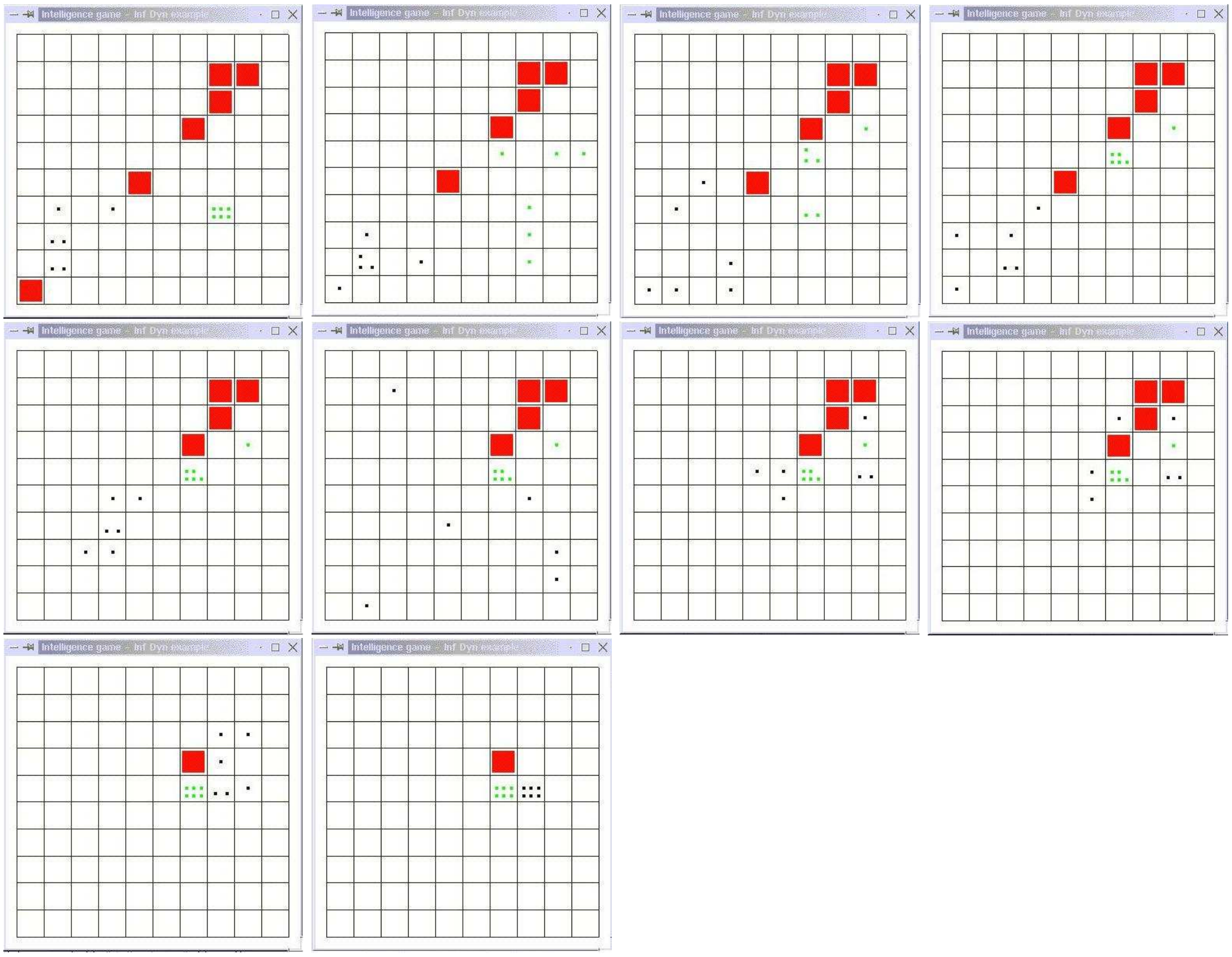

Figure 1: Sample game with defenders executing the strategy I. Targets are depicted as red squares, attackers as black dots, and defenders as green dots.

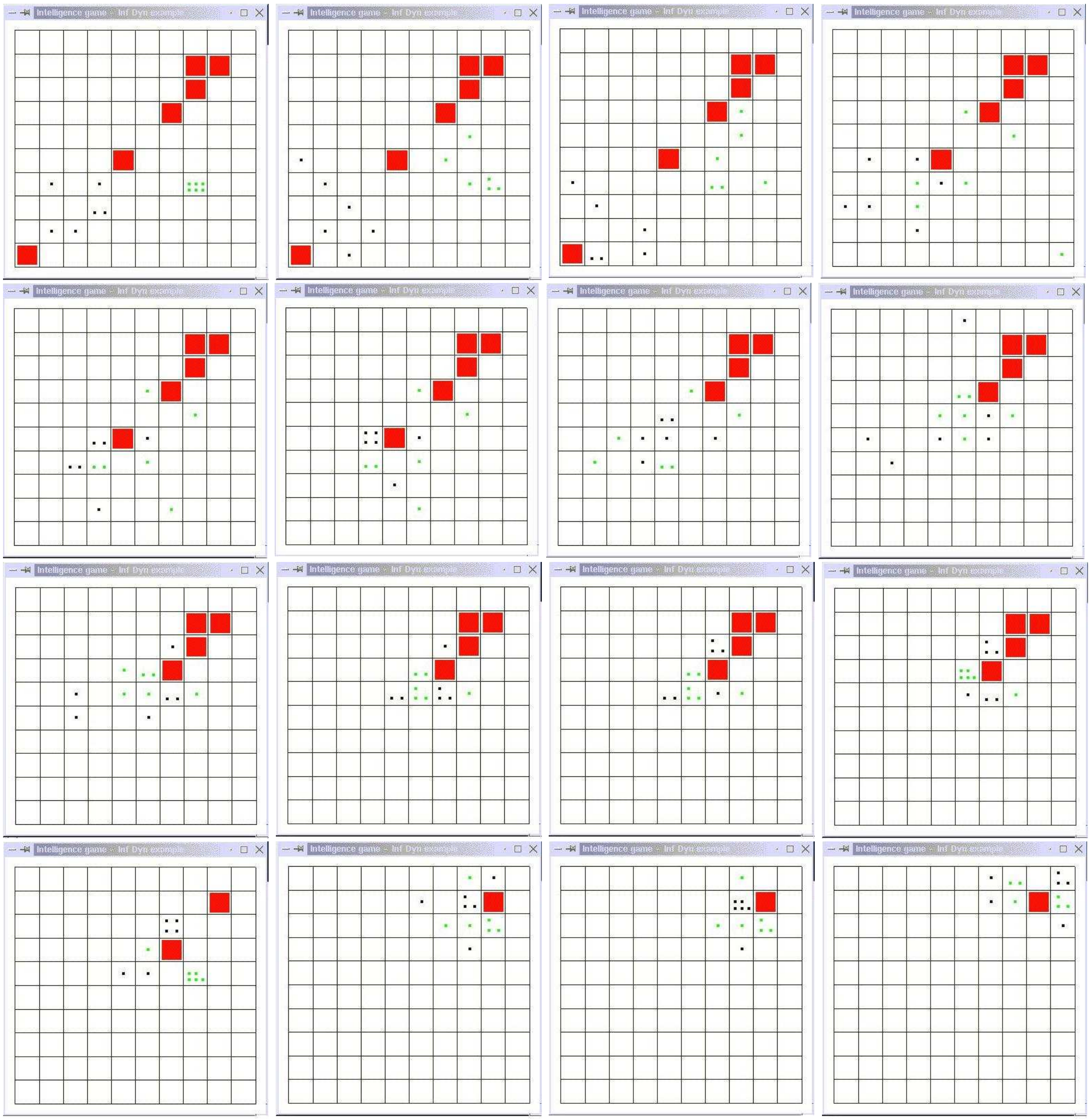

Figure 2: Sample game with defenders executing the strategy II. Targets are depicted as red squares, attackers as black dots, and defenders as green dots.