2002

Pursuit Evasion: The Herding Noncooperative Dynamic Game-The Stochastic Model

Pushkin Kachroo
University of Nevada, Las Vegas, pushkin@unlv.edu

Samy A. Shedied
Virginia Polytechnic Institute and State University

H. Vanlandingham
Virginia Polytechnic Institute and State University

GAs have been analyzed from two viewpoints. We studied the best solution found by the system, to observe its ability to obtain a local or global optimum. The second viewpoint is the diversity within the population of GAs; to examine this, the average fitness was calculated. For the first viewpoint, the most important factors were selection operator, type of mutation, the population size, and the number of generations. It is noteworthy that the type of crossover factor (one point/two points) produces practically identical results, although the application probability (p_c) does present statistically significant differences in the evolution of the GA from the perspective of Best Fitness. Regarding the diversity of the population in the final generations, analysis of the average fitness revealed that the most important factors are the selection and mutation operators and the mutation probability.

ACKNOWLEDGMENT

The authors appreciate the comments from the anonymous referees.

REFERENCES

[1] L. Davis, *The Handbook of Genetic Algorithms*. New York: Van Nostrand, 1991.
[2] R. A. Fisher, "Theory of statistical estimation," in *Proc. Cambridge Philos. Soc.*, vol. 22, 1925, pp. 700–725.
[3] D. E. Goldberg, K. Deb, and J. H. Clark, "Genetic algorithms, noise, and the sizing of populations," *Complex Syst.*, vol. 6, pp. 333–362, 1992.
[4] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-16, pp. 122–128, Jan./Feb. 1986.
[5] F. Herrera and M. Lozano, "2-loop real-coded genetic algorithms with adaptive-control of mutation step sizes," *Appl. Intell.*, vol. 13, no. 3, pp. 187–204, 2000.
[6] I. Jagielska, C. Matthews, and T. Whitfort, "An investigation into the application of neural networks, fuzzy logic, genetic algorithms, and rough sets to automated knowledge acquisition for classification problems," *Neurocomputing*, vol. 24, pp. 37–54, 1999.
[7] R. Mead, *The Design of Experiments. Statistical Principles for Practical Application*. Cambridge, U.K.: Cambridge Univ. Press, 1988.
[8] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. New York: Springer-Verlag, 1999.
[9] H. Mühlenbein, "How genetic algorithms really work—Part I: Mutation and hillclimbing," in *Proc. 2nd Conf. Parallel Problem Solving form Nature*, R. Männer and B. Manderick, Eds. Amsterdam, The Netherlands, 1992, pp. 15–25.
[10] M. O. Odetayo, "Relationship between replacement strategy and population size," in *Proc. MENDEL*, P. Osmera, Ed., 1996, pp. 91–96.
[11] M. Ryynänen, "The optimal population size of genetic algorithm in magnetic field refinement," in *Proc. 2nd Nordic Workshop Genetic Algorithms and Their Applications*, J. T. Alander, Ed., 1996, pp. 281–282.
[12] R. Smith, "Population size," in *Handbook of Evolutionary Computation*, T. Bäck, D. Fogel, and Z. Michalewicz, Eds. New York: IOP, Bristol, U.K., and Oxford Univ. Press, 1997, pp. E1.1:1–5.

Pursuit Evasion: The Herding Noncooperative Dynamic Game—The Stochastic Model

Pushkin Kachroo, Samy A. Shedied, and Hugh Vanlandingham

***Abstract*—This correspondence proposes a solution to the herding problem, a class of pursuit evasion problem, in stochastic framework. The problem involves a "pursuer" agent trying to herd a stochastically moving "evader" agent to a pen. The problem is stated in terms of allowable sequential actions of the two agents. The solution is obtained by applying the principles of stochastic dynamic programming. Three algorithms for solution are presented with their accompanying results.**

***Index Terms*—Admissible policy search stochastic shortest path, policy iteration, value function, value iteration.**

I. INTRODUCTION

This correspondence presents the herding problem as a class of pursuit evasion problems. However, in pursuit evasion problems, the terminal state satisfies the spatial coordinates of the pursuer and the evader to be the same [1]–[3]. Meanwhile, the terminal state in the herding problem relates to the evader having reached and satisfied at the same time fixed spatial coordinate point. In another paper [4], we have studied the herding problem in a deterministic setting where the evader is passive. This correspondence studies the stochastic version of the problem where the evader dynamics involves randomness. A classic pursuit evasion game in a stochastic framework was studied [5], but with different terminal state than that of the problem studied here.

This problem can be viewed as a modified version of stochastic shortest path problems. Despite the fact that shortest path techniques, like label correcting algorithms [6] and auction algorithms [7], provide a solution to shortest path problems, these techniques fail to deal with situations like the one we study in this correspondence.

The correspondence is organized as follows. In Section II, we give a detailed description of the system dynamics since it represents the backbone of our proposed solution technique. Based on these dynamics, some characteristic properties of the system are derived in Section III. In Section IV, we introduce a mathematical statement for the system model. Finally, the proposed solution techniques with simulation results and graphs are given in Sections V and VI, respectively.

II. AN $N \times N$ STOCHASTIC PURSUER–EVADER PROBLEM

In this section, we introduce the pursuer–evader problem in an $N \times N$ grid and present the dynamics. The pursuer can occupy one of the $N \times N$ positions, as may the evader. However, they cannot both have the same initial position. The objective of the pursuer is to drive the evader to the pen, (0, 0) position, in minimum expected time.

The state vector at time k, $\boldsymbol{x}(k)$, is determined by the position of the evader and the pursuer, i.e.

$$\boldsymbol{x}(k) = [x_e(k) \quad y_e(k) \quad x_p(k) \quad y_p(k)]$$

Manuscript received February 12, 2001; revised December 27, 2001. This work was supported by the Office of Naval Research under Project N00014-98-1-0779. This paper was recommended by Associate Editor M. Shahidehpour.

The authors are with the Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute of Technology and State University, Blacksburg, VA 24061 USA (e-mail: pushkin@vt.edu).

Publisher Item Identifier S 1094-6977(02)04679-5.

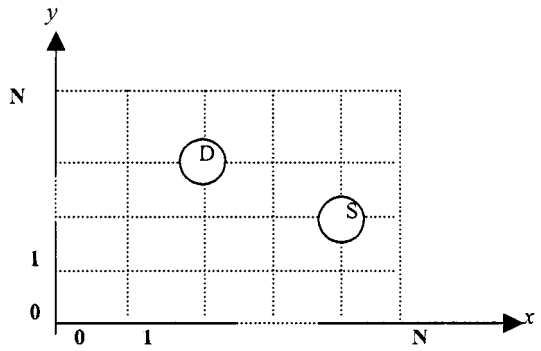

Fig. 1. $N \times N$ pursuer–evader problem grid.

where

$x_e(k)$ x coordinate of the evader at time k;
$y_e(k)$ y coordinate of the evader at time k;
$x_p(k)$ x coordinate of the pursuer at time k;
$y_p(k)$ y coordinate of the pursuer at time k.

Therefore, at any time k, we have $x_p \in \{0, 1, 2 \cdots N\}$, $y_p \in \{0, 1, 2 \cdots N\}$, $x_e \in \{0, 1, 2, \ldots N\}$, and $y_e \in \{0, 1, 2, \ldots N\}$. However, based on the dynamics, as will be illustrated later, if the pursuer and the evader are not in the same initial position, they will never be in the same location in the future. A cost of one unit is assigned for each step (horizontal, vertical, or diagonal) for the pursuer as well as the evader. Fig. 1 illustrates the $N \times N$ spatial grid of the pursuer–evader problem.

The following are some definitions we use to help simplify the description of the system.

Definition 1: Positive Successor Function: Positive successor function is given by

$$PS(Z(k)) = \begin{cases} Z(k), & \text{if } Z(k) = N \\ Z(k)+1, & \text{if } 0 \le Z(k) < N \end{cases} \quad (1)$$

where $Z(k)$ is the x or y coordinate of either the pursuer or the evader.

Thus, $PS(\,.\,)$: $Z = \{0, 1, 2, \ldots N\} \rightarrow Z - \{0\}$.

Definition 2: Negative Successor Function: Negative successor function is given by

$$NS(Z(k)) = \begin{cases} Z(k), & \text{if } Z(k) = 0 \\ Z(k)-1, & \text{if } 0 < Z(k) \le N \end{cases}. \quad (2)$$

Thus, $NS(\,.\,)$: $Z = \{0, 1, 2, \ldots N\} \rightarrow Z - \{N\}$.

Definition 3: Equilibrium State of the Evader: The evader is said to be in an equilibrium state when given a time instant T the following condition is satisfied:

$$\forall k \ge T$$

if

$$x_p(k) = x_p(T) \quad \text{and} \quad y_p(k) = y_p(T)$$

then

$$x_e(k) = x_e(T) \quad \text{and} \quad y_e(k) = y_e(T).$$

Definition 4: Final Equilibrium State of the Evader: The evader is in final equilibrium state at time instant T, when the following condition is satisfied:

$$\forall k > T$$

if

$$x_p(k) = x_p(T) \quad \text{and} \quad y_p(k) = y_p(T)$$

then

$$x_e(k) = 0 \quad \text{and} \quad y_e(k) = 0.$$

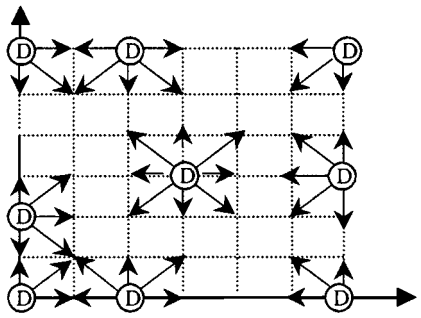

Fig. 2. Pursuer movements with the system in equilibrium state, case 1.

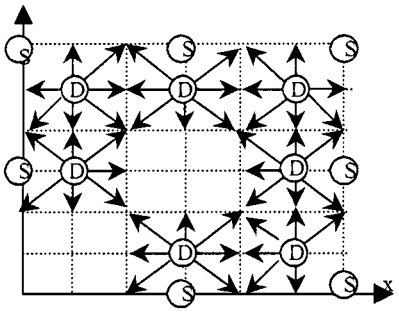

Fig. 3. Pursuer movements with the system in equilibrium state, case 2.

The following rules generate the pursuer-controlling movements and assign probabilities to the evader transitions based on the evader location with respect to the pursuer:

i) $\forall k$, $x_p(k)$, $y_p(k)$, $x_e(k)$, and $y_e(k) \in \{0, 1, 2, \ldots N\}$.

ii) The pursuer moves when the system is at equilibrium state only excluding the final equilibrium.

iii) The pursuer can move one step at a time, depending on its position in the grid, and its relative location with respect to the evader position. In terms of distance between the evader and the pursuer, we have two cases where the system is in equilibrium state. In case 1, the distance between the pursuer (D) and the evader (S) is greater than or equal two steps, as shown in Fig. 2.

Fig. 3 illustrates those equilibrium states (case 2) where the distance between the pursuer and the evader equals one. Here, besides the physical boundary conditions limitations, the current location of the evader disallows the pursuer from taking one of the actions that cause their locations to be identical.

iv) The evader transition probabilities depend on its position with respect to that of the pursuer. These transition probabilities are compactly represented in the following if–then statements.

a) *Far Condition:*

If

$$x_e(k) < NS(x_p(k)) \text{ or } x_e(k) > PS(x_p(k))$$

or

$$y_e(k) < NS(y_p(k)) \text{ or } y_e(k) > PS(y_p(k))$$

then

$$x_e(k+1) = x_p(k) \;\&\; y_e(k+1) = y_p(k).$$

b) *Top-Left Corner Pursuer Right Condition:*

If

$$x_e(k) = 0 \;\&\; x_p(k) = PS(x_e(k)) \;\&\; y_e(k) = y_p(k) = N$$

then

$$P\{x_e(k+1) = x_e(k)\ \&\ y_e(k+1) = NS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = p$$
$$P\{x_e(k+1) = PS(x_e(k))\ \&\ y_e(k+1) = NS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p).$$

c) *Top-Left Corner Pursuer Down Condition:*

If

$$x_e(k) = x_p(k) = 0\ \&\ y_p(k) = NS(y_e(k))\ \&\ y_e(k) = N$$

then

$$P\{x_e(k+1) = PS(x_e(k))\ \&\ y_e(k+1) = y_e(k)|x_p(k)\ \&\ y_p(k)\} = p$$
$$P\{x_e(k+1) = NS(x_e(k))\ \&\ y_e(k+1) = NS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p).$$

d) *Top-Right Corner Pursuer Left Condition:*

If

$$x_e(k) = N\ \&\ x_p(k) = NS(x_e(k))\ \&\ y_e(k) = y_p(k) = N$$

then

$$P\{x_e(k+1) = x_e(k)\ \&\ y_e(k+1) = NS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = p$$
$$P\{x_e(k+1) = NS(x_e(k))\ \&\ y_e(k+1) = NS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p).$$

e) *Top-Right Corner Pursuer Down Condition:*

If

$$x_e(k) = x_p(k) = N\ \&\ y_p(k) = NS(y_e(k))$$

then

$$P\{x_e(k+1) = NS(x_e(k))\ \&\ y_e(k+1) = y_e(k)|x_p(k)\ \&\ y_p(k)\} = p$$
$$P\{x_e(k+1) = NS(x_e(k))\ \&\ y_e(k+1) = NS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p).$$

f) *Bottom-Left Corner Pursuer Right Condition:*

If

$$x_e(k) = 0\ \&\ x_p(k) = PS(x_e(k))\ \&\ y_p(k) = y_e(k) = 0$$

then

$$P\{x_e(k+1) = x_e(k)\ \&\ y_e(k+1) = PS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = p$$
$$P\{x_e(k+1) = PS(x_e(k))\ \&\ y_e(k+1) = PS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p).$$

g) *Bottom-Left Corner Pursuer Up Condition:*

If

$$x_e(k) = x_p(k) = 0\ \&\ y_p(k) = PS(y_e(k))$$

then

$$P\{x_e(k+1) = PS(x_e(k))\ \&\ y_e(k+1) = y_e(k)|x_p(k)\ \&\ y_p(k)\} = p$$
$$P\{x_e(k+1) = PS(x_e(k))\ \&\ y_e(k+1) = NS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p).$$

h) *Bottom-Right Corner Pursuer Left Condition:*

If

$$x_e(k) = N\ \&\ x_p(k) = NS(x_e(k))\ \&\ y_p(k) = y_e(k) = 0$$

then

$$P\{x_e(k+1) = x_e(k)\ \&\ y_e(k+1) = PS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = p$$
$$P\{x_e(k+1) = NS(x_e(k))\ \&\ y_e(k+1) = PS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p).$$

i) *Bottom-Right Corner Pursuer Up Condition:*

If

$$x_e(k) = x_p(k) = N\ \&\ y_e(k) = 0\ \&\ y_p(k) = PS(y_e(k))$$

then

$$P\{x_e(k+1) = NS(x_e(k))\ \&\ y_e(k+1) = y_e(k)|x_p(k)\ \&\ y_p(k)\} = p$$
$$P\{x_e(k+1) = NS(x_e(k))\ \&\ y_e(k+1) = PS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p).$$

There are some conditions that were not represented in the above rules. These are given below under "other conditions" category. These include situations such as the following.

Other Conditions: If a) to i) are not satisfied and;

i) $x_p(k) = NS(x_e(k))$ & $y_p(k) = PS(y_e(k))$ then

$$P\{x_e(k+1) = PS(x_e(k))\ \&\ y_e(k+1) = NS(y_e(k)|x_p(k)\ \&\ y_p(k)\} = p$$
$$P\{x_e(k+1) = x_e(k)\ \&\ y_e(k+1) = NS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p)/2$$
$$P\{x_e(k+1) = PS(x_e(k))\ \&\ y_e(k+1) = y_e(k)|x_p(k)\ \&\ y_p(k)\} = (1-p)/2.$$

ii) $x_p(k) = x_e(k)$ & $y_p(k) = PS(y_e(k))$ then

$$P\{x_e(k+1) = x_e(k)\ \&\ y_e(k+1) = NS(y_e(k)|x_p(k)\ \&\ y_p(k)\} = p$$
$$P\{x_e(k+1) = NS(x_e(k))\ \&\ y_e(k+1) = NS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p)/2$$
$$P\{x_e(k+1) = PS(x_e(k))\ \&\ y_e(k+1) = NS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p)/2.$$

iii) $x_p(k) = PS(x_e(k))$ & $y_p(k) = PS(y_e(k))$ then

$$P\{x_e(k+1) = NS(x_e(k))\ \&\ y_e(k+1) = NS(y_e(k)|x_p(k)\ \&\ y_p(k)\} = p$$
$$P\{x_e(k+1) = x_e(k)\ \&\ y_e(k+1) = NS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p)/2$$
$$P\{x_e(k+1) = NS(x_e(k))\ \&\ y_e(k+1) = y_e(k)|x_p(k)\ \&\ y_p(k)\} = (1-p)/2.$$

iv) $x_p(k) = PS(x_e(k))$ & $y_p(k) = y_e(k)$ then

$$P\{x_e(k+1) = NS(x_e(k))\ \&\ y_e(k+1) = y_e(k)|x_p(k)\ \&\ y_p(k)\} = p$$
$$P\{x_e(k+1) = NS(x_e(k))\ \&\ y_e(k+1) = NS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p)/2$$
$$P\{x_e(k+1) = NS(x_e(k))\ \&\ y_e(k+1) = PS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p)/2.$$

v) $x_p(k) = PS(x_e(k))$ & $y_p(k) = NS(y_e(k))$ then

$$P\{x_e(k+1) = NS(x_e(k))\ \&\ y_e(k+1) = PS(y_e(k)|x_p(k)\ \&\ y_p(k)\} = p$$

$$P\{x_e(k+1) = x_e(k)\ \&\ y_e(k+1) = PS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p)/2$$
$$P\{x_e(k+1) = NS(x_e(k))\ \&\ y_e(k+1) = y_e(k)|x_p(k)\ \&\ y_p(k)\} = (1-p)/2.$$

vi) $x_p(k) = x_e(k)$ & $y_p(k) = NS(y_e(k))$ then

$$P\{x_e(k+1) = x_e(k))\ \&\ y_e(k+1) = PS(y_e(k)|x_p(k)\ \&\ y_p(k)\} = p$$
$$P\{x_e(k+1) = NS(x_e(k))\ \&\ y_e(k+1) = PS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p)/2$$
$$P\{x_e(k+1) = PS(x_e(k))\ \&\ y_e(k+1) = PS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p)/2.$$

vii) $x_p(k) = NS(x_e(k))$ & $y_p(k) = NS(y_e(k))$ then

$$P\{x_e(k+1) = PS(x_e(k))\ \&\ y_e(k+1) = PS(y_e(k)|x_p(k)\ \&\ y_p(k)\} = p$$
$$P\{x_e(k+1) = x_e(k)\ \&\ y_e(k+1) = PS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p)/2$$
$$P\{x_e(k+1) = PS(x_e(k))\ \&\ y_e(k+1) = y_e(k)|x_p(k)\ \&\ y_p(k)\} = (1-p)/2.$$

viii) $x_p(k) = x_e(k)$ & $y_p(k) = NS(y_e(k))$ then

$$P\{x_e(k+1) = PS(x_e(k))\ \&\ y_e(k+1) = y_e(k)|x_p(k)\ \&\ y_p(k)\} = p$$
$$P\{x_e(k+1) = PS(x_e(k))\ \&\ y_e(k+1) = NS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p)/2$$
$$P\{x_e(k+1) = PS(x_e(k))\ \&\ y_e(k+1) = PS(y_e(k))|x_p(k)\ \&\ y_p(k)\} = (1-p)/2.$$

III. Properties of the Stochastic Digraph Associated With the Pursuer–Evader Problem

The following paragraph presents the characteristic properties of the stochastic digraph associated with the pursuer–evader problem.

a) The number of states of the system is finite.
b) The system is stationary, which means that the probability distribution of transition between states, the instantaneous cost, and the system dynamics are independent of the states.
c) There are $(N \times N - 3)$ final equilibrium states of the $N \times N$ stochastic pursuer–evader problem.
d) The cost value of the final equilibrium states is zero.
e) At any time instance t, $(x_e, y_e) \neq (x_p, y_p)$ if at $t = t_0$ $(x_e, y_e) \neq (x_p, y_p)$.
f) The di-graph representing the stochastic pursuer–evader model is pseudostochastic, which means that the transition between states is stochastic when the evader has to move. However, when the pursuer has to move, the transition between states is deterministic since the pursuer is allowed to go to certain locations based on the dynamics. In other words, the probability of pursuer transitions is always 1.
g) The estimated cost associated with each edge in the pseudostochastic pursuer–evader di-graph is equal to 1 [4].

IV. Problem Statement

The problem statement we propose can be solved in terms of a value function. The value function gives a performance measure, the optimization of which provides us with desired solution. The following is the definition of the value function.

Given:

1) A finite state space $S = \{1, 2 \cdots N\}$ with transition probability between states, $p_{ij}(u)$, given by
$$p_{ij}(u) = P(s_{k+1} = j | s_k = i,\ u_k = u)$$
where $u_k = u \in U(i)$, the control set, is finite at each state i.
2) The instantaneous cost $c(i,\ u)$ of a state i incurred when the control $u \in U(i)$ is selected to be
$$c(i,\ u) = \sum_{j=1}^{n} p_{ij}(u)\tilde{c}(i,\ u,\ j) \tag{3}$$
where $\tilde{c}(i,\ u,\ j)$ is the estimated cost to move from state i using control u to go to state j, then, the value function of each state is given by

$$J(i) = \left[c(i,\ u) + \sum_{j=1}^{n} p_{ij}(u)J(j)\right] \qquad i = 1,\ 2,\ \ldots,\ n. \tag{4}$$

Our objective is to find the set of optimal control policy $\mathbf{U}^*$ that gives minimum expected cost value for the pursuer to drive the evader to the pen, i.e.

$$J^*(i) = \min_{u \in U} E\left\{g(i,\ u) + \sum_{j=1}^{n} p_{ij}(u)J(j)\right\}, \qquad i = 1,\ \ldots,\ n. \tag{5}$$

V. Methods of Solution

Solving the problem outlined above mainly depends on calculating the cost function values. Although it may look like that the problem simplifies down to solving a set of $(N \times N)^2$ linear algebraic equations, in fact this is not the case since the policy is based on the cost function values that are yet to be determined.[1] Once this policy is determined, then the problem may be considered as that of solving a system of linear equations possibly even of order less that $(N \times N)^2$. The following describe three techniques for solving the cost value function while searching for the optimum control policy.

A. Admissible Policy Search Technique

Let us denote $\mathbf{J}^*$ as a vector of the optimal cost values of associated with all states of the system. Also, we use $\mathbf{P}$ to denote the *probability state transition matrix (PSTM)*. Solving for $\mathbf{J}^*$ mainly depends on the characteristics of the PSTM, $\mathbf{P}$. Close examination of the PSTM shows that if the transition between states results from pursuer movements, the entities of the corresponding row for the current state are either 0, where there is no path to go to the corresponding state, or 1, which corresponds to the next state, the pursuer can drive the system to. However, if the transition results from the evader's movement, the corresponding entry of the PSTM is either equal to 0, or the probability of the system to go the corresponding state due to evader movement. It can be noticed that there is no control action for the evader's movements and therefore, the minimization process is over only a single value. On the

[1]The power of $N \times N$ is 2 here because the number of players is 2; otherwise the power should be replaced by the number of players.

other hand, there are multiple possible control actions of the pursuer for those states in which the pursuer is allowed to move. For those states, the minimization is done over all the possible actions. In other words, for each action of the pursuer, at each state, (5) can be written as

$$\mathbf{J}^* = \left(\mathbf{I} - \tilde{\mathbf{P}}\right)^{-1} \mathbf{SC} \tag{6}$$

where $\mathbf{S}$ is the $N \times N$ state transition matrix; $\mathbf{C}$ is the $N \times 1$ instantaneous cost matrix; and $\tilde{\mathbf{P}}$ is the $N \times N$ modified PSTM. This matrix $\tilde{\mathbf{P}}$ is obtained from $\mathbf{P}$ by picking the only "1" that corresponds to the state that results in the minimum cost value and replacing all the other ones by zeros. Therefore, the problem now condenses to finding $\tilde{\mathbf{P}}$.

This can be accomplished by searching over all the possible combinations of ones in the PSTM till we obtain the pattern that results in the minimum value of $\mathbf{J}$.

Despite its accuracy in calculating the values of the cost function, this technique is very time consuming since it searches over the entire space of admissible control policies.

B. Value Iteration Technique

Value iteration technique converges to the solution if certain condition on the problem is satisfied [8], [9]. Our system satisfies that condition and therefore we expect the convergence of the solution.

Based on that assumption, an iterative technique can be used to calculate the cost value of each state beginning form any initial values $J_0(1) \cdots J_0(n)$. This iterative algorithm is given by

$$J_{k+1}(i) = \min_{u \in U(i)} \left[c(i,\ u) + \sum_{j=1}^{n} p_{ij}(u) J_k(j) \right]. \tag{7}$$

Equation (7) is referred to as the stochastic form of the Bellman's equation [10].

The sequence $J_{k+1}(i)$ will converge to the optimal cost, $J^*_{k+1}(i)$ given by Bellman's equation, after finite number of iterations.

C. Policy Iteration Technique

This technique depends on searching the admissible policy subspace in a steepest decent way, beginning from any initial admissible policy. This algorithm involves four steps.

Step 1) *Initialization Step:* Start with one of the admissible policies, u^0.

Step 2) *Policy Evaluation Step:* Solve the linear system of equations given by (6) to get the cost values for this policy, $J_{u^0}(i)$, $i = 1, 2, \ldots n$. However, it might be nontrivial to solve this set of linear equations in many situations. For example, when the matrix given by $(\mathbf{I} - \tilde{\mathbf{P}})$ is singular. In such cases, the value iteration algorithm can be used to calculate the corresponding $\mathbf{J}$.

Step 3) *Policy Improvement Step:* In this step, we compute a new policy u^{k+1} that minimizes the expected cost calculated in Step 2), i.e.

$$u^{k+1}(i) = \arg \min_{u \in U(i)} \left[c(i,\ u) + \sum_{j=1}^{n} p_{ij}(u) J_{u^k}(j) \right]. \tag{8}$$

Step 4) If $J_{u^{k+1}}(i) = J_{u^k}(i)$, $i = 1, 2, \ldots n$, terminates or set $J_{u^{k+1}}(i) = J_{u^0}(i)$ and go to Step 1).

Notice that this algorithm terminates in a finite number of steps simply because there are finite number of control policies. However, since value iteration may be used to solve for $\mathbf{J}$ in some cases, this may lead to infinite number of iterations.

TABLE I
COST FUNCTION VALUES

S	V_v	V_p	S	V_v	V_p	S	V_v	V_p	S	V_v	V_p
1	9.00	9.00	22	3.50	3.50	43	3.88	3.88	64	0.00	0.00
2	6.22	6.22	23	5.81	5.81	44	5.60	5.60	65	2.50	2.50
3	4.88	4.88	24	7.21	7.21	45	6.36	6.36	66	3.88	3.88
4	6.22	6.22	25	4.88	4.88	46	0.00	0.00	67	1.50	1.50
5	8.21	8.21	26	5.60	5.60	47	1.50	1.50	68	3.39	3.39
6	6.60	6.60	27	6.36	6.36	48	2.88	2.88	69	4.60	4.60
7	4.88	4.88	28	7.14	7.14	49	2.50	2.50	70	2.88	2.88
8	6.60	6.60	29	6.02	6.02	50	3.39	3.39	71	9.00	9.00
9	7.36	7.36	30	4.88	4.88	51	9.00	9.00	72	5.36	5.36
10	7.14	7.14	31	9.00	9.00	52	3.88	3.88	73	0.00	0.00
11	9.00	9.00	32	6.61	6.61	53	4.60	4.60	74	2.50	2.50
12	8.09	8.09	33	5.60	5.60	54	5.36	5.36	75	3.88	3.88
13	6.02	6.02	34	8.09	8.09	55	0.00	0.00	76	2.50	2.50
14	6.61	6.61	35	7.21	7.21	56	3.50	3.50	77	1.50	1.50
15	7.21	7.21	36	6.36	6.36	57	4.88	4.88	78	4.60	4.60
16	4.88	4.88	37	0.00	0.00	58	1.70	1.70	79	3.88	3.88
17	5.60	5.60	38	2.50	2.50	59	5.81	5.81	80	4.60	4.60
18	6.36	6.36	39	3.88	3.88	60	5.60	5.60	81	9.00	9.00
19	0.00	0.00	40	2.50	2.50	61	9.00	9.00			
20	1.70	1.70	41	9.00	9.00	62	7.21	7.21			
21	9.00	9.00	42	5.60	5.60	63	6.36	6.36			

VI. SIMULATION RESULTS

Simulating our system begins by computing the value of the cost function of each state using one of the techniques mentioned above. Table I, shows the values of the cost function obtained by using value iteration, V_v, and policy iteration, V_p, techniques for a 3×3 grid, and interstate transition probability of 0.8.

As shown from the results, both techniques converge to exactly the same cost function value. The state number corresponds to every possible combination of the x and y coordinates of the pursuer and evader positions. The evader movements are controlled by random number generators according to which, the evader can move randomly according to the dynamics defined in . The pursuer makes its transitions based on the cost function value of the adjacent states to the current state of the system.

The pursuer moves to the state of the lowest cost of all adjacent states, then it checks whether the system is at equilibrium or not to make its next move. This process continues till the system reaches one of the final equilibrium states defined in .

A graphical user interface, designed using MATLAB 5.3, is used to simulate the system where the user is to choose the grid size N from a drop box. The initial position of the pursuer and the evader is supplied to the simulation program using an edit box. The probability p of transition is set using a slider. Finally, the technique used to calculate the cost of each state is chosen using a check box and then simulation starts by pressing the *START* button.

Figs. 4 and 5 show the used graphical user interface provided to simulate the system. Fig. 4 shows a single run with initial position of the pursuer at (0, 0), the initial position of the evader at (4, 4) and the probability of transition is 0.8 where the cost values are calculated using value iteration technique. Meanwhile, Fig. 5 illustrates another simulation with the same initial condition and the same probability of transition system, but the cost value is calculated using policy iteration technique. It should be noticed that the differences between the two systems, although having the same initial states, comes from the stochastic motion of the evader. It can be also noticed that the pursuer's movements are the same from the initial state till the state where the

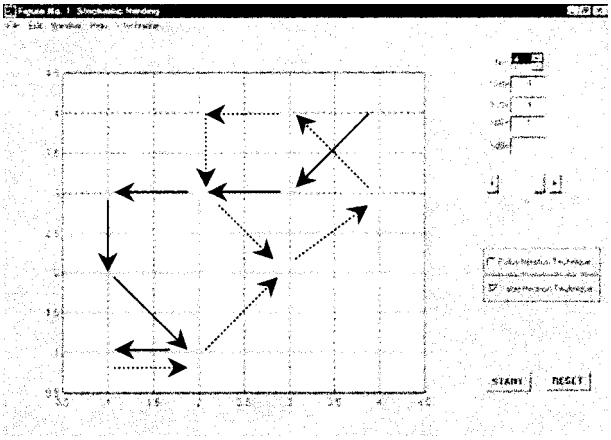

Fig. 4. Graphical user interface model with cost values calculated by value iteration technique.

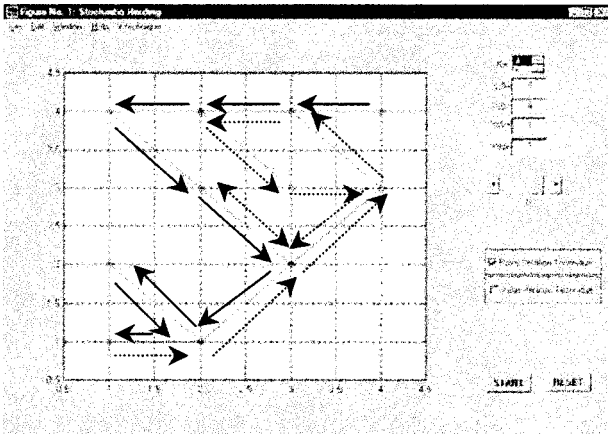

Fig. 5. Graphical user interface model with cost values calculated by policy iteration technique.

evader makes its first move. This is due to the deterministic nature of the pursuer's motion.

VII. Conclusion

In this correspondence, we studied a stochastic class of pursuit evasion problems that is different than the traditional problems in that the aim of the pursuer is to force the evader into a pen. We introduced three techniques for optimal solution and showed one of them to be so time consuming compared to the other two. On the other hand, the other two iterative techniques converged to the same optimal cost value functions, which is used as our performance index. We also presented a graphical user interface package that has been developed to experiment with the problem.

References

[1] Y. Yavin and M. Pachter, Eds., *Pursuit–Evasion Differential Games*. New York: Pergamon, 1987.
[2] R. Isaacs, *Differential Games: A Mathematical Theory With Applications to Warfare and Pursuit, Control and Optimization*. New York: Dover, 1965.
[3] Basar, Tamer, Olsder, and G. Jan, *Dynamic Noncooperative Game Theory*, 2nd ed. New York: Academic, 1982, ch. 8.
[4] P. Kachroo *et al.*, "Dynamic programming solution for a class of pursuit evasion problems: The herding problem," *IEEE Trans. Syst., Man, Cybern. C*, vol. 31, pp. 35–41, Feb. 2001.
[5] P. Bernhard, A. L. Colomb, and G. P. Papavassilopoloulos, "Rabbit and hunter game: Two discrete stochastic formulations," *Comput. Math. Applicat.*, vol. 13, no. 1–3, pp. 205–225, 1987.
[6] D. P. Bertesekas, *Linear Network Optimization: Algorithms and Codes*. Cambridge, MA: MIT Press, 1991.
[7] ——, "The auction algorithms for shortest paths," *SIAM J. Optim.*, vol. 1, pp. 425–447, 1991.
[8] ——, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena, 1995, vol. 2.
[9] ——, *Dynamic Programming*. Englewood Cliffs, NJ: Prentice-Hall, 1987.
[10] R. E. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1969.

Spectral Fuzzy Classification: An Application

Ana del Amo, Javier Montero, Angeles Fernández, Marina López, José Manuel Tordesillas, and Greg Biging

***Abstract*—Geographical information (including remotely sensed data) is usually imprecise, meaning that the boundaries between different phenomena are fuzzy. In fact, many classes in nature show internal gradual differences in species, health, age, moisture, as well other factors. If our classification model does not acknowledge that those classes are heterogeneous, and crisp classes are artificially imposed, a final careful analysis should always search for the consequences of such an unrealistic assumption. In this correspondence, we consider the unsupervised algorithm presented in [3], and its application to a real image in Sevilla province (south Spain). Results are compared with those obtained from the *ERDAS* ISODATA classification program on the same image, showing the accuracy of our fuzzy approach. As a conclusion, it is pointed out that whenever real classes are natural fuzzy classes, with gradual transition between classes, then its fuzzy representation will be more easily understood—and therefore accepted—by users.**

***Index Terms*—Fuzzy classification, outranking models, remote sensing.**

I. Introduction

Classification of land cover by means of remote sensing implies a search for a formal definition for class. From a traditional remote sensing point of view, our theoretical aim is a partition of the image into *homogeneous* sectors, each one of them hopefully corresponding to a class. As long as our precision increases, we can continue partitioning the image, and therefore new classes should be defined. In fact, the number of classes should be as big as possible, provided we can interpret them. However, quite often we realize that an image is based upon a few *natural* classes, with the picture full of transition zones.

Manuscript received December 1, 2000; revised January 22, 2002. This work was supported by Grant PB98-0825 from the Government of Spain, and a Del Amo bilateral program between Complutense University of Madrid, Madrid, Spain, and the University of California at Berkeley. This paper was recommended by Associate Editor A. Kandel.

A. del Amo was with the Faculty of Mathematics, Complutense University of Madrid, Madrid 28040, Spain. She is now with Smiths Aerospace Electronic Systems, Grand Rapids, MI 49512-1991 USA.

J. Montero is with the Faculty of Mathematics, Complutense University of Madrid, Madrid 28040, Spain.

A. Fernández, M. López, and J. M. Tordesillas are with the National Geographic Institute, Madrid 28040, Spain.

G. Biging is with the Department of Environmental Science Policy and Management, University of California, Berkeley, CA 94720 USA.

Publisher Item Identifier S 1094-6977(02)04675-8.