2011 IEEE International Symposium on Intelligent Control (ISIC)
Part of 2011 IEEE Multi-Conference on Systems and Control
Denver, CO, USA. September 28-30, 2011

A Cell Decomposition Approach to Online Evasive Path Planning and the Video Game Ms. Pac-Man

Greg Foderaro, Vikram Raju and Silvia Ferrari

Abstract— **This paper presents an approach for optimizing paths online for a pursuit-evasion problem where an agent must visit several target positions within a region of interest while simultaneously avoiding one or more actively-pursuing adversaries. This is relevant to applications such as robotic path planning, mobile-sensor applications, and path exposure. The methodology described utilizes cell decomposition to construct a modified decision tree to achieve the objective of minimizing the risk of being caught by an adversary and maximizing a reward associated with visiting the target locations. By computing paths online, the algorithm can quickly adapt to unexpected movements by the adversaries or dynamic environments. The approach is illustrated through a modified version of the video game Ms. Pac-Man which is shown to be a benchmark example of the pursuit-evasion problem. The results show that the approach presented in this paper runs in real-time and outperforms several other methods as well as most human players.**

I. Introduction

Although simple in appearance and gameplay, the single player video game Ms. Pac-Man offers a challenging representation of a pursuit-evasion problem that requires extended foresight, quick decisions and a high degree of adaptability. As explained in [1], games often present excellent benchmarks for testing intelligent algorithms because they have simple rules and objectives but offer challenging environments and tasks. The pursuit-evasion family of games describes a predator and prey scenario where the objective of one group is to evade a second group in the environment which has the goal of tracking and catching the first group. This type of game is analogous to several real-world applications such as robotic path planning [2], [3], mobile-sensor applications [4], and path exposure [5], [6].

In Ms. Pac-Man, the player assumes the role of the evader and must navigate a maze to visit several target locations ("dots") while avoiding a team of pursuing adversaries with individualized strategies. There is a pre-determined pattern of dots scattered in each maze which Ms. Pac-Man must eat, and when all of the dots have been eaten, the player advances to the next level which involves a more difficult maze, a new set of dots, and faster adversaries. A screenshot of the game's first maze is shown in Figure 1.

Silvia Ferrari is with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708 USA (email: sferrari@duke.edu). Greg Foderaro is with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708 USA (email: greg.foderaro@duke.edu). Vikram Raju is with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708 USA(email: vikram.raju@duke.edu).

The methodology presented in this paper gives an approach for determining optimal paths online for an evader in a pursuit-evasion problem where an agent must balance the tasks of avoiding a group of pursuers while visiting several target locations within a region of interest. Cell decomposition is used to transform the space in the environment where the agent is allowed to travel into a finite set of convex cells. The decomposition is used to construct a modified decision tree and achieve the objective of minimizing the risk of being caught by an adversary and maximizing a reward associated with visiting the target locations. The approach is illustrated through a modified version of the video game Ms. Pac-Man, which is shown to be a benchmark example of the pursuit-evasion problem. The results show that the approach presented in this paper outperforms several other methods as well as most human players.

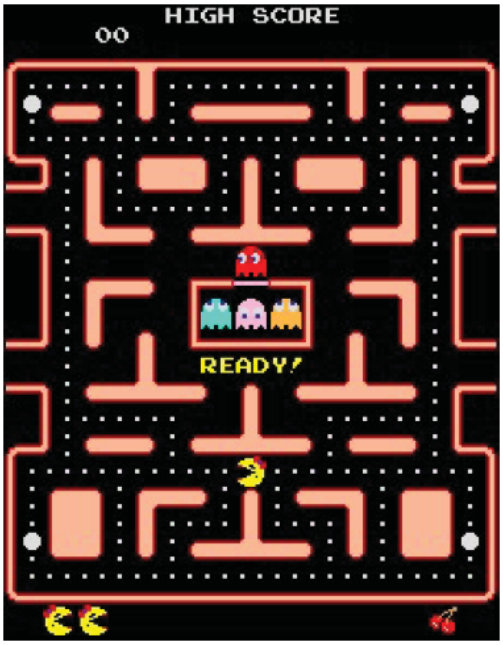

Fig. 1. Screenshot of Level 1 game maze.

II. Problem Formulation and Assumptions

The path planning problem considered in this paper is to find the optimal paths of a single mobile agent with position or state, x_p, that travels in a two-dimensional Euclidian workspace denoted by $\mathcal{W} \subset \Re^2$. The agent must navigate

978-1-4577-1103-9/11/$26.00 ©2011 IEEE

through the workspace and collect a set of distributed objects or visit several points of interest within $\mathcal{W}$ while simultaneously avoiding collisions with a group of N actively-pursuing adversaries with states denoted by x_G^I where I corresponds to an adversary's index, and a collision is defined as any instant where $x_p = x_G^I$ for all I. The optimal paths are then those that minimize the risk of encountering an adversary while maximizing the number of goal positions achieved. The workspace geometry and the positions of the points of interest or distributed objects are assumed to be known *a priori*, and the approach can be extended to higher-dimensional workspaces. The behaviors or control laws of the adversaries are also assumed to be known, and their positions are assumed to be observable in real-time.

This problem can be put in the context of a benchmark problem taken from the video game Ms. Pac-Man, for which the details are provided in Section I. The agent's (Pac-Man's) state and control are represented by the vectors,

$$x_p = \begin{bmatrix} x_{p_x} & x_{p_y} \end{bmatrix}^T \tag{1}$$

$$u_p = \begin{bmatrix} u_{p_x} & u_{p_y} \end{bmatrix}^T \tag{2}$$

where x_{px} and x_{py} are Pac-Man's x and y coordinates in pixels, and Pac-Man's controls, u_{px} and u_{py}, signify the attempted movement in the x and y directions, respectively. The adversaries' (ghosts') states and controls, x_G^I and u_G^I, are defined in an identical manner. The workspace is defined as a maze encountered in the game, as shown in Figure 1. Let $\mathcal{W}$ have an inertial frame of reference, $F_{\mathcal{W}}$, such that all possible xy-coordinates are in the positive orthant and are always greater than zero. The geometries of Pac-Man and the ghosts can be thought of as rectangular in shape such that their widths only allow for bidirectional movement along a straight path length. Therefore, each point within the maze has a set of admissible actions, $U[x(t_k)] \subset \mathcal{U}$ where

$$\mathcal{U} = \{a_1, a_2, a_3, a_4\} \equiv \left\{ \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\} \tag{3}$$

is the space of all possible actions or control values for both Pac-Man and the ghosts, and t_k is a discretized instant in time such that $t_i \leq t_k \leq t_f$. According to the coordinate frame convention, $u_x = +1$ denotes motion to the right, $u_x = -1$ represents motion to the left, $u_y = +1$ corresponds to upward motion, and $u_y = -1$ coincides with downward motion.

In the game, the player's goal is to achieve the highest possible score by eating various objects. However, contrary to the game, the objective of the problem presented in this paper is solely to avoid the ghosts and collect all of the dots within $\mathcal{W}$. Therefore the agent performance is based on the number of dots collected (up to a maximum of the number initially present in the maze) without being caught by a ghost. To make the evasion more difficult for the agent, this paper also neglects the effects of the "power pill" bonuses seen in the game, which temporarily cause the ghosts to retreat and allow Pac-Man to send the ghosts back to their starting locations by eating them. In addition, the ghosts are not slowed when they travel through a "tunnel" which is a wrap-around path that allows characters to quickly move to the opposite side of the maze. This and the other bonus features are not included in the described simulations or performance metrics. However as will be shown in a separate paper, the approach presented can be extended to allow additions that give the agent the capabilities to temporarily assume the role of a pursuer when necessary, as is required to achieve high scores within the game.

The method for computing the optimal paths is based on the cell decomposition and connectivity tree approach described in Section V.

III. Modeling of Adversary Behavior

Let $I_G = \{I | I = r, p, b, o\}$ denote the ghosts' index set where r, p, b and o represent each of the four ghosts in the game. While in active pursuit, each of the ghosts chases Pac-Man in an individualized manner by utilizing different rules for choosing a set of target locations, denoted by x_T^I, which guide their decisions. The positions of the targets are functions of time and Pac-Man's state where Pac-Man's position and control are represented by (1) and (2). The ghosts then share an identical algorithm for moving to their separate targets. The laws employed for determining the ghosts' targets are as follows:

For the red ghost, $I = r$, the target is assigned as the location of Pac-Man. This causes the red ghost to often chase the player from behind.

$$x_T^r(t_k) = x_p(t_k) \tag{4}$$

For the pink ghost, $I = p$, the target is set as the position slightly in front of Pac-Man, and the resulting behavior is an adversary that tries to attack from the front.

$$x_T^p(t_k) = x_p(t_k) + A_i d \text{ for } u_p(t_k) = a_i \tag{5}$$

where d = $\begin{bmatrix} 32 & 32 \end{bmatrix}^T$ in units of pixels and,

$$\begin{aligned} A_1 &= \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}, \\ A_3 &= \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}, \quad A_4 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}. \end{aligned} \tag{6}$$

For the light blue ghost, $I = b$, the target is a reflection of the red ghost's position about the pink ghost's target. This causes the light blue ghost to seem like it attempts to guess Pac-Man's future paths.

$$x_T^b(t_k) = [2 \cdot x_R(t_k) - x_G^r(t_k)] \tag{7}$$

where the reflection point, $x_R(t_k)$ is,

$$x_R(t_k) = x_p(t_k) + A_i e, \quad e = \begin{bmatrix} 16 & 16 \end{bmatrix}^T \tag{8}$$

For the orange ghost, $I = o$, the target is set as the bottom left corner of the maze if Pac-Man is within a defined radius of the ghost. However, if Pac-Man is outside of that radius, the target becomes the location of Pac-Man itself, which is identical to the red ghost's strategy. The orange ghost is the least threatening, as it often keeps its distance from Pac-Man,

but its seemingly unpredictable behavior sometimes causes it to collide with the player's path unexpectedly.

$$x_T^o(t_k) = \begin{Bmatrix} x_B & \text{for } \left\| x_G^o(t_k) - x_p(t_k) \right\| \leq c \\ x_p(t_k) & \text{for } \left\| x_G^o(t_k) - x_p(t_k) \right\| > c \end{Bmatrix} \forall k \quad (9)$$

where $c = 80$ pixels and x_B denotes the position vector of the bottom left corner of the game maze.

After the target positions are calculated, the ghosts all utilize a common rule for moving towards their separate targets:

$$u_G^l(t_k) = \begin{cases} a_i = H\{B\}, & for\ a_i \in U_G^l[x_G^l(t_k)] \\ a_j = H\{C\} \circ sgn\{D\} & for\ a_i \notin U_G^l[x_G^l(t_k)], \\ & \quad a_j \in U_G^l[x_G^l(t_k)] \\ a_k = U_G^l\{1\} & for\ a_i \notin U_G^l[x_G^l(t_k)], \\ & \quad a_j \notin U_G^l[x_G^l(t_k)] \end{cases} \quad (10)$$

where $\circ$ denotes the Schur product, $H\{\cdot\}$ represents the Heaviside function, and,

$$B = \begin{bmatrix} \left| x_{Gx}^I(t_k) - x_{px}(t_k) \right| & \left| x_{Gy}^I(t_k) - x_{py}(t_k) \right| \\ \left| x_{Gy}^I(t_k) - x_{py}(t_k) \right| & \left| x_{Gx}^I(t_k) - x_{px}(t_k) \right| \end{bmatrix} \quad (11)$$

$$C = \begin{bmatrix} \left| x_{Gy}^I(t_k) - x_{py}(t_k) \right| & \left| x_{Gx}^I(t_k) - x_{px}(t_k) \right| \\ \left| x_{Gx}^I(t_k) - x_{px}(t_k) \right| & \left| x_{Gy}^I(t_k) - x_{py}(t_k) \right| \end{bmatrix} \quad (12)$$

$$D = \begin{bmatrix} x_{px}(t_k) - x_{Gx}^I(t_k) \\ x_{py}(t_k) - x_{Gy}^I(t_k) \end{bmatrix} \quad (13)$$

Since it can be seen in (10) that the ghosts will not choose an action opposite to their current action, they will not reverse direction on a path and will only effectively make decisions when they encounter intersections where there are three or more directions in which they can move.

IV. Numerical Verification of Model

By recording the position of Pac-Man and the ghosts during gameplay on an emulated Ms. Pac-Man game found at [7] with a simple screen-capture program, it is possible to verify the models of the ghosts' behaviors by comparing these positions with those generated by the equations described above. This is done by setting the initial positions of Pac-Man and the ghosts in a simulated game to the positions recorded at some arbitrary instant during the real game. The simulated game is then run for a period of time, and the resulting trajectories of the ghosts are compared to those observed from the real game. An example of the comparison is shown in Figure 2.

It can be seen that the simulated ghosts behave very similarly to the ones in the real game. When the initial state of the simulated game was set to match the real game's state at an arbitrary instant, the decisions chosen and the resulting paths were almost always identical. The small number of errors that are present are predicted to be caused by slight imprecisions in the screen-capture approach when extracting the game state. These is due to the inability to know the exact positions referenced by the game compared to the character images it displays. Figure 3 shows how the path comparison of the light blue ghosts from Figure 2.c appears with respect

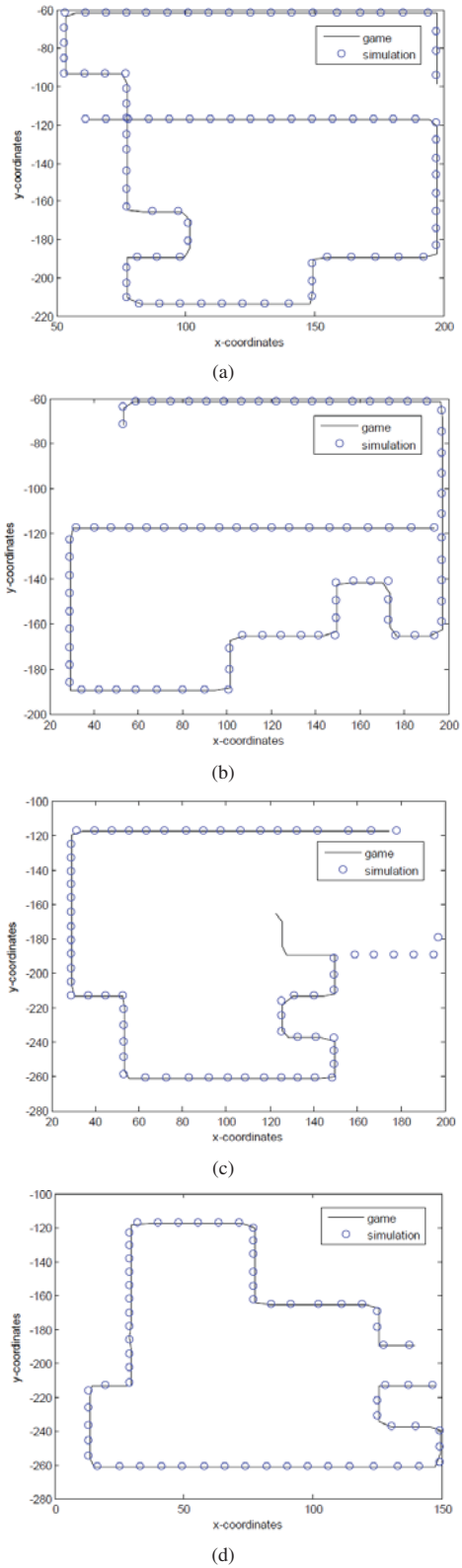

Fig. 2. Comparisons between the ghosts' paths in the Ms. Pacman game and using the derived model (a) Red ghost, (b) Pink ghost, (c) Light blue ghost, (d) Orange ghost

to time, and it can be seen that there is a small amount of error present. To calculate a numerical approximation of the model's accuracy, the simulated game was given an initial state from the real game and run until a ghost's decision from the simulated game differed from the corresponding ghost in the real game. The number of correct decisions was counted, and the process was repeated several times. After 10 runs, the model of the ghost behaviors correctly evaluated 818 decisions out of a total of 829. Therefore, the approximate accuracy of the model is 98.7%.

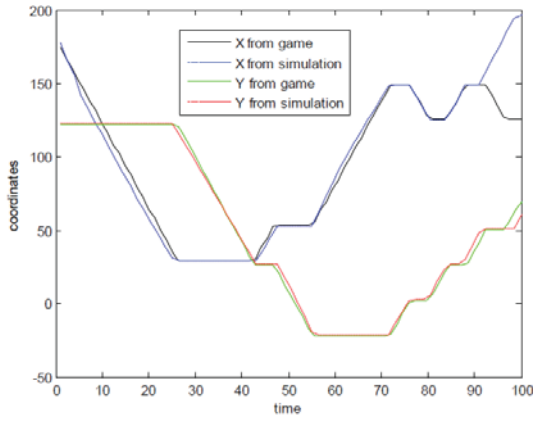

Fig. 3. Comparison between the light blue ghosts' paths in the Ms. Pacman game and using the derived model

V. Methodology

The methodology presented in this paper can be summarized as follows. The workspace $\mathcal{W}$ is decomposed into rectangloids using a line-sweeping approach. From the decomposition, a connectivitiy tree, $\mathcal{T}$, is formed using the adjacency relationships between cells and the agent cell position, κ_p. Each branch in the connectivity tree represents a path extending from κ_p. The instaneous optimal path is chosen by selecting the branch that maximizes a given objective function.

A. Cell Decomposition and Connectivity Tree

Cell decomposition is a well-known robotic path planning method used for obstacle avoidance [8], [9]. The approach decomposes a workspace into a finite set of non-overlapping convex polygons, known as cells, such that each cell represents a subset of the workspace in which the agents and adversaries can move freely without colliding with an obstacle. In classical cell decomposition, this can be obtained by using a line-sweeping algorithm and constructing a one-dimensional representation of the free-space geometry known as a connectivity graph.

The workspace of Ms. Pac-Man, $\mathcal{W}$, is decomposed as shown in Figure 4 using an approach that has an added property compared to classical decomposition. Namely, a unique set of admissible actions from (3) is associated with each cell, and Pac-Man and the ghosts can perform those actions anywhere inside the cell. Let κ_j denote a cell in the decomposition, and I_κ represent the index set of all cells in the decomposition of $\mathcal{W}$, and in the corresponding connectivity graph denoted by $\mathcal{G}$ and illustrated in Figure 5.

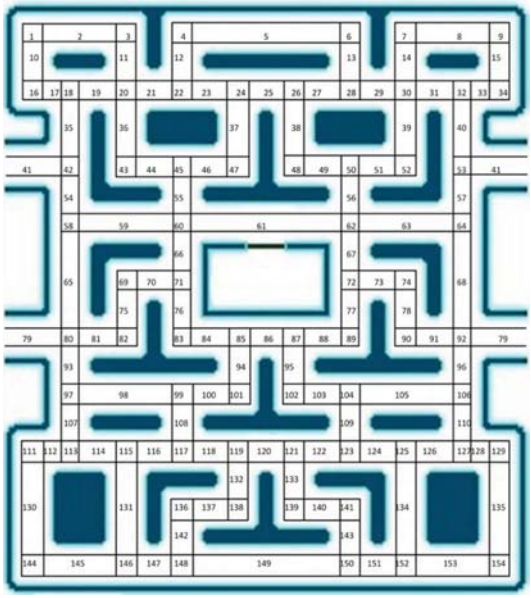

Fig. 4. Cell decomposition of the Level 1 maze

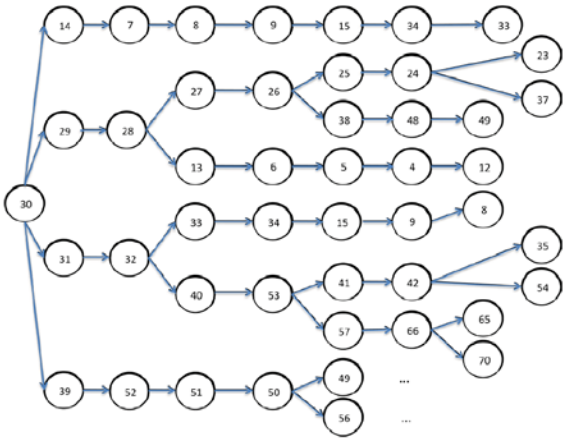

Fig. 5. Connectivity tree of the cell decomposition

Definition 5.1: A connectivity graph, $\mathcal{G}$, is a non-directed graph where the nodes represent rectangloid cells in the cell decomposition, and two nodes κ_i and κ_j in $\mathcal{G}$ are connected by an arc (κ_i, κ_j) if and only if the corresponding cells are adjacent in the decomposition.

Several methods can be used to search $\mathcal{G}$ for sequences of adjacent cells, connecting possible paths for Pac-Man within the maze. Based on a maximum feasible distance, the connectivity graph can be pruned and transformed into a decision tree, which is a graphical representation of all possible paths from κ_p. After visiting a cell, Pac-Man can

only move to an adjacent cell, creating a causal process that can be represented as follows:

Definition 5.2: The connectivity tree $\mathcal{T}$ associated with $\mathcal{G}$ and the agent's cell position, κ_p, is a tree graph with κ_p as the root and branches with length L. The nodes represent cells where travel is possible, and a branch represents a non-reversing path extending from κ_p.

Based on the cell properties, it can be seen that a unique action value is attached to each arc in $\mathcal{T}$. Therefore, the set of admissible actions is a function of x_p and of $u_p(t_k - 1)$, and is denoted by $\mathcal{U}_p[x_p(t_k)] \subset \mathcal{U}$. Since the agent is not allowed to immediately reverse direction in $\mathcal{T}$, the set of admissible actions is given by the complement of $u_p(t_{k-1})$ in the set of arcs attached to the cell occupied at time t_k, in $\mathcal{G}$.

For the example workspace, $\mathcal{W}$ in Figure 4, the connectivity tree is illustrated in Figure 5. Since the size of tree used in this paper is very large, only the first few layers of the tree are shown, where layers are defined by their adjacency position relative to the initial cell. For the example problem, the connectivity tree is limited to prohibit decisions involving reversals in the middle of a cell or actions resulting in no movement. However, these capabilities can be added by incorporating tree reflections or additional repeating nodes.

B. Optimal Agent Strategy

Based on the above problem formulation, it is possible to define and compute Pac-Man's instantaneous reward, $\mathcal{L}$, for every cell $\kappa_j \in \mathcal{G}$, over time. The instantaneous reward is defined as the tradeoff between the value and risk associated with visiting a cell $\kappa_j \in \mathcal{G}$, that is,

$$\begin{aligned} \mathcal{L}[x_p(t_k), u_p(t_k)] &\equiv w_V V[x_p(t_k), u_p(t_k)] \\ &+ w_R R[x_p(t_k), u_p(t_k)] \end{aligned} \tag{14}$$

where w_V and w_R are user-defined weights, and $V[x_p(t_k), u_p(t_k)]$ is the number of dots in the corresponding cell at the time Pac-Man would visit that cell. Note that the reward for a corresponding cell would change once Pac-Man travels through it. The risk is defined as,

$$R[x_p(t_k), u_p(t_k)] = \sum_{\ell \in I_G} [|x_p(t_k) - x_G^\ell| - \rho_0]^2 \tag{15}$$

where $|\cdot|$ is the Manhattan distance and ρ_0 is a user-defined parameter, such that when $[x_p(t_k) - x_G^\ell(t_k)] \to \rho_0, R \to 0$. The ghost states at the time corresponding to the layer of the tree are evaluated using the validated equations from Section III.

We seek an optimal strategy $\sigma*$ defined as a sequence of functions,

$$\sigma* = c_i, ..., c_F \tag{16}$$

where each function c_k maps the state x_p into an admissible decision,

$$u_p(t_k) = c_k[x_p(t_k)], fork = i, ..., F \tag{17}$$

and maximizes the cost-to-go,

$$J_{i,F}[x_p(t_i)] \equiv \sum_{k=i}^{F} \alpha_k \mathcal{L}[x_p(t_k), u_p(t_k)] \tag{18}$$

from the present time, t_i, up to the final time, t_F, over the finite horizon $[t_i, t_F]$, where $\mathcal{L}[\cdot]$ is defined in (14) and α_k is a discount factor that is defined as an exponential function of k, such that future rewards are discounted compared to immediate ones.

Since the connectivity tree effectively amounts to a decision tree, as the tree is grown and the instantaneous rewards are computed and attached to each node, along with $\mathcal{L}$, the cumulative cost,

$$\begin{aligned} J_{i,k}[x_p(t_i)] &\equiv \sum_{j=i}^{k} \alpha_j \mathcal{L}[x_p(t_j), u_p(t_j)] \\ &= J_{i,k-1}[x_p(t_i)] + \alpha_k \mathcal{L}[x_p(t_k), u_p(t_k)] \end{aligned} \tag{19}$$

can also be computed for each node at each time step, t_k in $\mathcal{T}$ iteratively over time, where it can be seen from Figure 5 that each node in the time step t_k denotes one of the possible values of Pac-Man's state $x_p(t_k)$, and each outgoing arc denotes one of the possible values of Pac-Man's control $u_p(t_k)$. Therefore, by the time the tree is completed, the cost-to-go, $J_{i,F}$ for each branch (i.e. each possible sequence of state and decision values) will be attached to the leaf of the branch (i.e. the last node, at t_F) and, thus, the optimal branch can be determined simply by picking the largest value or $J_{i,f}$, namely $J_{i,F}*$. The optimal branch will then determine the optimal strategy $\sigma*$ which will consist of the sequence of arcs in the optimal branch.

VI. Results

To quantify the algorithm's performance under the desired conditions, an accurate simulation of the game was created in C# based on the verfied equations in Section III, the maze map from the first level, and a knowledge of the game mechanics. The simulation differs slightly from the real game in that some features were removed to focus on the objectives discussed and to make them more difficult for the agent. These include power pill bonuses which temporarily cause the ghosts to retreat and allow Pac-Man to send the ghosts back to their starting locations by eating them, and the slower movement the ghosts experience when traveling through a "tunnel", which is a wrap-around path that allows characters to quickly move to the opposite side of the maze. In addition, the ghosts' speeds have been altered manually to be faster than in the game. The objective for the agent is simply to eat as many dots as possible (up to the maximum number of 220 initially present in the maze) before being caught by a ghost.

The approach described in Section V has been implemented using a connectivity tree branch length of 15 cells. The simulation was run 20 times for each of four ghost speed configurations: 90%, 95%, 100% and 105% of Pac-Man's speed. In the game, the ghosts' speeds on the first and

fifth level were measured to be approximately 93% and 96% of Pac-Man's speed, respectively [7]. The results from the simulations and a comparison with novice human players are shown in Table I. Screenshots from the simulation's graphical output is displayed in Figure 6.

TABLE I
PERFORMANCE OF ARTIFICIAL AND HUMAN PAC-MAN PLAYERS OVER 20 PARTIAL GAME SIMULATIONS

Cell Decomposition Approach		
Ghost speed %	Mazes cleared	Average dots eaten
90%	19	217
95%	19	216
100%	14	204
105%	3	148
Human Players		
Ghost speed %	Mazes cleared	Average dots eaten
90%	7	171
95%	4	161
100%	1	105
105%	0	88

The agent performed the evasion and collection objectives well and was able to clear the maze on 95% of its attempts for conditions similar to the real game and 15% of its tries for ghost speeds much faster than the game. The algorithm also runs faster than what is needed for real-time gameplay, so the presented approach can interact with the actual Ms. Pac-man game if a method for inputting the game states is added.

This level of success is greater than what has been accomplished with automated players that are not hand-coded [10]–[13], and the results would be very difficult for most human players to match. However, the approach has flaws that make it weaker than the average human player in some situations. For example, the method does not allow Pac-Man to reverse direction in the middle of a cell or to stop moving. Many of its failures occured when the agent moved itself from a safe situation into dangerous one because it did not have the capability to evaluate other options. However, additions can be made to the connectivity tree (e.g. tree reflections or repeated nodes) that would effectively fix most of the shortfalls.

VII. CONCLUSIONS

This paper presents a methodology for optimizing paths online for a pursuit-evasion problem where an agent must visit several target positions within a region of interest while simultaneously avoiding one or more actively-pursuing adversaries. The methodology described utilizes cell decomposition to construct a modified decision tree to achieve the objective of minimizing the risk of being caught by an adversary and maximizing a reward associated with visiting the target locations. The approach is illustrated through a modified version of the video game Ms. Pac-Man which is shown to be a benchmark example of the pursuit-evasion problem. It was shown that the approach presented in this

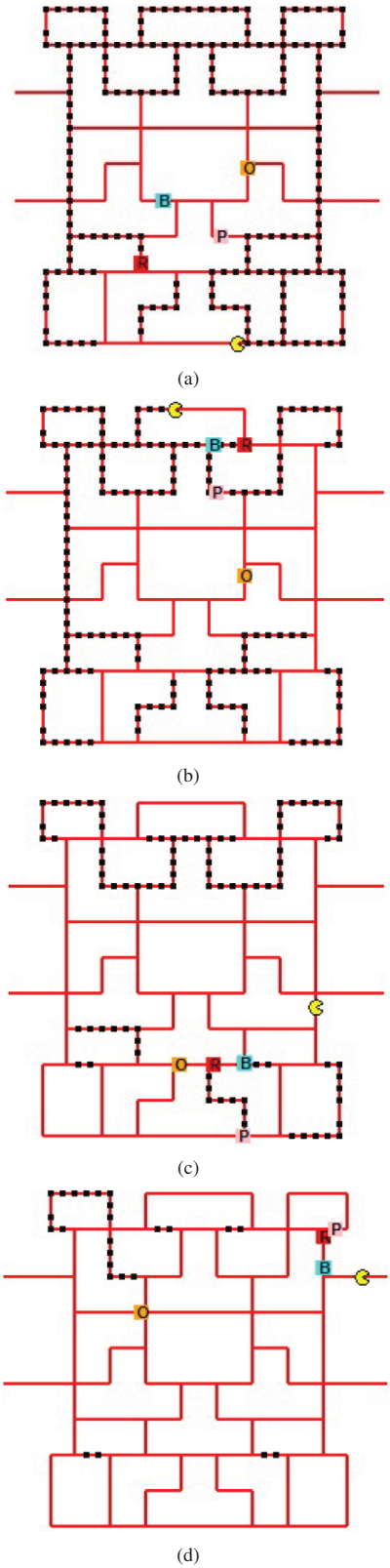

Fig. 6. Screenshots from the simulation's graphical output. The round yellow character represents Pac-man, the squares are the ghosts of their corresponding colors, and the small black squares are the dots that Pac-man must collect.

paper runs in real-time and outperforms several other methods as well as most human players.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grant ECS 0925407.

REFERENCES

[1] S. M. Lucas and G. Kendall, "Evolutionary computation and games," *IEEE Computer Intelligence Magazine*, vol. 1, no. 1, 2006.

[2] J. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1998.

[3] Z. Sun and J. Reif, "On robotic optimal path planning in polygonal regions with pseudo-eucledian metrics," *IEEE Transactions on Systems, Man, and Cybernetics - Part A*, vol. 37, no. 4, pp. 925–936, 2007.

[4] D. E. D. Culler and M. Srivastava, "Overview of sensor networks," *Computer*, vol. 37, no. 8, pp. 41–49, 2004.

[5] S. Megerian, F. Koushanfar, G. Q. amd G. Veltri, and M. Potkonjak, "Exposure in wireless sensor networks: Theory and practical solutions," *Wireless Networks*, vol. 8, pp. 443–454, 2002.

[6] V. Phipatanasuphorn and P. Ramanathan, "Vulnerability of sensor networks to unauthorized traversal and monitoring," *IEEE Transactions on Computers*, vol. 53, no. 3, 2004.

[7] *Ms. Pacman Game*. http://webpacman.com/.

[8] D. Zhu and J. C. Latombe, "New heuristic algorithms for efficient hierarchical path planning," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 1, pp. 9–20, 1991.

[9] K. Kadem and M. Sharir, "An efficient motion planning algorithm for convex polygonal object in 2-dimensional polygonal space," *Courant Institute of Mathematical Science, New York, NY, Tech. Rep. 253*.

[10] S. Lucas, "Evolving a neural network location evaluator to play ms. pac-man," *IEEE Symposium on Computational Intelligence and Games*, 2005.

[11] M. Gallagher and A. Ryan, "Learning to play pac-man: An evolutionary, rule-based approach," *Proc. of the Congress on Evolutionary Computation (CEC)*, 2003.

[12] P. Burrow and S. Lucas, "Evolution versus temporal difference learning for learning to play ms. pac-man," *Proc. on the 5th International Conference on Computational Intelligence and Games*, 2009.

[13] L. DeLooze and W. Viner, "Fuzzy q-learning in a nondeterministic environment: Developing an intelligent ms. pac-man agent," *Proc. on the 5th International Conference on Computational Intelligence and Games*, 2009.