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

Greg Foderaro, Vikram Raju, Silvia Ferrari
Laboratory for Intelligent Systems and Controls (LISC)
Department of Mechanical Engineering and Materials Science
Duke University

MSC 2011
Denver, Colorado
9/28/11
Introduction and Motivation

- Ms. Pac-Man is a challenging benchmark problem in the pursuit-evasion family of games.

- Algorithms are relevant to real-world applications such as robotic path planning, mobile sensor networks, and path exposure.

- The best approaches for solving this type of problem online perform poorly compared to a human.
The player’s main goal in Ms. Pac-Man is to achieve the highest possible score by earning points for eating (traveling over) “dots” and other objects.

Pac-man must navigate through a maze to reach all dots while evading four pursuing ghost adversaries.

A level is cleared when all dots have been eaten. The game continues in a new, more difficult maze with faster ghosts.

When a ghost is able to catch Pac-man, the player loses one of three lives. The game ends when the player runs out of lives.

The focus of this research so far has been to develop an artificial player that is capable of planning optimal trajectories for Pac-man to evade the ghosts and eat dots.
Summary of Methodology

- Construct accurate model of game
- Decompose workspace into cells
- Use cell map to construct connectivity graph
- Utilize connectivity graph as decision tree
- Evaluate values associated with branches
- Choose the decision corresponding to the branch with highest value
Game Model

- Pac-Man’s state and control are represented by the $2 \times 1$ vectors,

$$x_p = \begin{bmatrix} x_{px} & x_{py} \end{bmatrix}^T, \quad u_p = \begin{bmatrix} u_{px} & u_{py} \end{bmatrix}^T$$

where $x_{px}$ and $x_{py}$ are Pac-Man’s x and y coordinates in pixels. Pac-Man’s controls, $u_{px}$ and $u_{py}$, signify the attempted movement in the x and y directions, respectively.

- The ghosts’ states and controls, $x^I_G$ and $u^I_G$, are defined identically, where $I_G = \{I \mid I = r, p, b, o\}$ denote the ghosts’ index set.

- Pac-Man and the ghosts are limited to bidirectional movement along straight paths, so a set of admissible actions, $U[x(t_k)] \subset \mathbb{U}$, is defined where

$$\mathbb{U} = [a_1, a_2, a_3, a_4] = \left\{ \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\}$$

with

- up
- left
- down
- right
Ghost Behavior Models

At any time during the game, each ghost has a target position:

- Red ghost – targets Pac-man
  \[ x'^R(t_k) = x^R_p(t_k) \]

- Pink ghost – targets in front of Pac-man
  \[ x'^P(t_k) = x^P_p(t_k) + A_i d \quad \text{for} \quad u^P_p(t_k) = a_i \]  
  \[ A_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}, \quad A_4 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad d = [32 \ 32]^T \]

- Green ghost – targets reflection of red ghost across Pac-man
  \[ x'^G(t_k) = 2 \cdot x^G_R(t_k) - x'^G(t_k) \]  
  \[ x^G_R(t_k) = x^G_p(t_k) + A_t e, \quad e = [16 \ 16]^T \]

- Orange ghost – targets Pac-man when far away, bottom left corner when close
  \[ x'^O(t_k) = \begin{cases} x^O_B, & \text{for} \quad \left\| x^O(t_k) - x^O_p(t_k) \right\| \leq c, \\
  x^O_p(t_k), & \text{for} \quad \left\| x^O(t_k) - x^O_p(t_k) \right\| > c \end{cases} \quad \forall k \]
Red ghost – targets Pac-man

\[ x^r_T(t_k) = x_p(t_k) \]

*Images courtesy of: Jamey Pittman*
Ghost Behavior Models

**Pink ghost** – targets in front of Pac-man

\[ x^{pT}(t_k) = x_p(t_k) + A_i d \quad \text{for} \quad u_p(t_k) = a_i \]

where,

\[
A_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}, \\
A_3 = \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}, \quad A_4 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix},
\]

\[ d = [32 \quad 32]^T \]

*Images courtesy of: Jamey Pittman*
Ghost Behavior Models

**Light blue ghost** – targets reflection of red ghost across Pac-man

\[ x^b_T(t_k) = \left[2 \cdot x_R(t_k) - x^r_G(t_k)\right], \]

where,

\[ x_R(t_k) = x_p(t_k) + A_i e, \]

\[ A_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}, \]

\[ A_3 = \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}, \quad A_4 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \]

\[ e = \begin{bmatrix} 16 & 16 \end{bmatrix}^T \]

*Images courtesy of: Jamey Pittman*
Ghost Behavior Models

Orange ghost –

Targets Pac-man when Euclidean distance to Pac-man is above a threshold and targets bottom-left corner of maze otherwise

\[ x^o_T(t_k) = \begin{cases} 
  x_B, & \text{for } \| x^o_G(t_k) - x_p(t_k) \| \leq c \\
  x_p(t_k), & \text{for } \| x^o_G(t_k) - x_p(t_k) \| > c
\end{cases} \quad \forall k \]

*Images courtesy of: Jamey Pittman*
All ghosts use the same algorithm to move to their target locations:

• Looks at horizontal and vertical distances from the ghost to its target.

• Tries to choose action that will reduce the larger of the two.

• If not possible, tries to reduce the smaller distance.

• If that is not possible, chooses first possible action from an ordered list of admissible actions.
Ghost Behavior Models

All ghosts use the same algorithm to move to their target locations:

\[
u^G(t_k) = \begin{cases} 
  a_i = H\{B\} \circ \text{sgn}\{D\} & \text{for } a_i \in U^G[x^G(t_k)] \\
  a_j = H\{C\} \circ \text{sgn}\{D\} & \text{for } a_i \not\in U^G[x^G(t_k)], a_j \in U^G[x^G(t_k)] \\
  a_k = U^G\{1\} & \text{for } a_i \not\in U^G[x^G(t_k)], a_j \not\in U^G[x^G(t_k)]
\end{cases}
\]

Where,

\[
B = \begin{bmatrix}
  x^I_{Gx}(t_k) - x_{Px}(t_k) & x^I_{Gy}(t_k) - x_{Py}(t_k) \\
  x^I_{Gy}(t_k) - x_{Py}(t_k) & x^I_{Gx}(t_k) - x_{Px}(t_k)
\end{bmatrix} \\
C = \begin{bmatrix}
  x^I_{Gy}(t_k) - x_{Py}(t_k) & x^I_{Gx}(t_k) - x_{Px}(t_k) \\
  x^I_{Gx}(t_k) - x_{Px}(t_k) & x^I_{Gy}(t_k) - x_{Py}(t_k)
\end{bmatrix} \\
D = \begin{bmatrix}
  x_{Px}(t_k) - x^I_{Gx}(t_k) \\
  x_{Py}(t_k) - x^I_{Gy}(t_k)
\end{bmatrix}
\]
Model Verification

Comparison of Trajectories of Simulated ghosts and ghosts from real game
Cell Decomposition

The workspace was decomposed into cells such that a set of admissible actions is associated with each cell.
The cells are mapped to create a connectivity graph which is then used to generate a decision tree with Pac-man’s current cell as the root.
Objective Function and Control Law

Control Law:
At each timestep, choose the action corresponding to branch with the highest value,

$$J_{i,F}[x_{p}(t_{i})] \equiv \sum_{k=i}^{F} \alpha_{k} L[x_{p}(t_{k}), u_{p}(t_{k})]$$

Where,

$$L[x_{p}(t_{k}), u_{p}(t_{k})] \equiv w_{V} V[x_{p}(t_{k}), u_{p}(t_{k})] + w_{R} R[x_{p}(t_{k}), u_{p}(t_{k})]$$

$$R[x_{p}(t_{k}), u_{p}(t_{k})] = \sum_{\ell \in I_{G}} \left[ \| x_{p}(t_{k}) - x_{G}^{\ell} \| - \rho_{0} \right]^{2}$$

\(V\): number of dots in corresponding cell when Pac-man will visit it
\(w_{V}, w_{R}\): weighting constants
\(\alpha\): discount factor
\(|\cdot|\): Manhattan norm
Simulations

- A partial reproduction of the game was constructed in C# using the maze map from the first level, derived ghost models, and known game mechanics.

- Some features, such as “power pills” and fruit, were omitted to focus on the objectives of evading the ghosts and eating dots.

- The ghost speeds were set as percentages of Pac-man’s speed, ranging from 90% to 105%.

- Each run begins with 220 dots to be eaten, and ends when either Pac-man has been caught by the ghosts or all of the dots are eaten.

- The performance was compared to that of two novice human players using a keyboard input to the modified game.
Results

- The simulation was run 20 times for each ghost speed configuration.
- In the real game, the ghost speeds on the 1st and 5th maze are approximately 93% and 96% of Pac-man’s speed respectively.

<table>
<thead>
<tr>
<th>Ghost speed %</th>
<th>Mazes cleared</th>
<th>Average dots eaten</th>
</tr>
</thead>
<tbody>
<tr>
<td>90%</td>
<td>19</td>
<td>217</td>
</tr>
<tr>
<td>95%</td>
<td>19</td>
<td>216</td>
</tr>
<tr>
<td>100%</td>
<td>14</td>
<td>204</td>
</tr>
<tr>
<td>105%</td>
<td>3</td>
<td>148</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ghost speed %</th>
<th>Mazes cleared</th>
<th>Average dots eaten</th>
</tr>
</thead>
<tbody>
<tr>
<td>90%</td>
<td>7</td>
<td>171</td>
</tr>
<tr>
<td>95%</td>
<td>4</td>
<td>161</td>
</tr>
<tr>
<td>100%</td>
<td>1</td>
<td>105</td>
</tr>
<tr>
<td>105%</td>
<td>0</td>
<td>88</td>
</tr>
</tbody>
</table>
Results
Conclusions and Future Work

- Developed an approach for optimizing paths online for the pursuit-evasion problem seen in the game Ms. Pac-man.
  - Constructed accurate model of game and adversary behavior.
  - Decomposed workspace into cells and constructed decision tree.
  - Evaluate values associated with branches and choose optimal decisions corresponding to the branches with the highest values.

- The presented method outperformed human players in a simplified reproduction of the game.

Future Work

- Complete interface with real game.
- Incorporate pursuit of ghosts.
Questions?