Survival of the Fastest: Using Cultural Algorithms to Optimize the Design of a Controller for a 3D Racing Game

Robert G. Reynolds
aa0057@wayne.edu
Professor, Department of Computer
Wayne State University
Detroit, MI 48202
And

Leonard Kinnaird-Heether
Wayne State University
Acknowledgements

Work reported is an extension of game programming projects in courses:

CSC 6430 (and 6431 (Lab))
Game Programming and Design II

And

CSC 7800
AI in Game Programming

These were offered in the Winter, of 2008.

Dr. Reynolds and Mr. Kinnaird-Heether were supported through UM-WSU IGERT Grant # DGE-0654014
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction to the Car Racing Competition</td>
</tr>
<tr>
<td>2</td>
<td>Design of the Driver Controller State Machine</td>
</tr>
<tr>
<td>3</td>
<td>Cultural Algorithms Background</td>
</tr>
<tr>
<td>4</td>
<td>Training the Controller with Cultural Algorithms</td>
</tr>
<tr>
<td>5</td>
<td>The Competition</td>
</tr>
<tr>
<td>6</td>
<td>Do Networks Matter?</td>
</tr>
<tr>
<td>7</td>
<td>Track Complexity and Network Learning</td>
</tr>
<tr>
<td>8</td>
<td>Conclusions</td>
</tr>
<tr>
<td>9</td>
<td>Future Work</td>
</tr>
</tbody>
</table>
Develop a controller for the TORCS open source racing game.

One of several competitions held at the 2008 World Congress in Hong Kong, China, June 1-6, 2008.

Rationale for the competition

1) To provide benchmarks for the comparison of learning algorithms which capture the complexity of real-world problems.

2) Competition provides software, resources, interfaces and scoring procedures to fairly and independently evaluate various algorithms and development methodologies.

3) Motivates researchers in the application of their approach to a new area, since entry is not restricted to those with computational intelligence-based approaches. Encourages comparison of seemingly heterogeneous approaches.

4) Opportunity to convince the game industry that CI algorithms can handle “real” games and not just academically posed ones.
The Open Source Car Simulator is a state-of-the-art open source car racing simulator. 

http://torcs.sourceforge.com

Midway between an advanced simulator and a 3D car racing game.

Features are:

1). A sophisticated physics engine
   (e.g. aerodynamics, traction, fuel consumption, and collisions)
2). A 3D graphics engine for race visualization.
3). Designed to support customizable car controllers for plug and play. Implemented as separable software modules.
4). A large amount of game content, many different tracks and surfaces.

Therefore, a large number of different racing surfaces can be produced.
A specific software interface was developed using a client server architecture.

Controllers run as external programs and communicate with a customized version of TORCS through UDP connections.

Controllers perceive the racing environment through sensor readings.

Sensors reflect the surrounding environment and basic game state including the tracks and opponents. They included: distance of nearby opponents, engine speed (RPM), fuel level (petrol), wheel spin velocity, a vector of 19 range finder sensors, and many others.

Controllers can invoke effectors that relate to basic driving commands. They include: accelerate, brake, gear shift, steering, and meta which requests a race restart.

Controllers have to act quickly before the data becomes obsolete.

Clients were available to support simple APIs along with programmed controllers in Java and C++ as well as for Windows, MAC, and Linux operating systems.
The Architecture of the API developed especially for the competition.
# Sensor Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle</td>
<td>Angle between the car direction and the direction of the track axis.</td>
</tr>
<tr>
<td>curLapTime</td>
<td>Time elapsed during current lap.</td>
</tr>
<tr>
<td>damage</td>
<td>Current damage of the car (the higher is the value the higher is the damage).</td>
</tr>
<tr>
<td>distFromStartLine</td>
<td>Distance of the car from the start line along the track line.</td>
</tr>
<tr>
<td>distRaced</td>
<td>Distance covered by the car from the beginning of the race</td>
</tr>
<tr>
<td>fuel</td>
<td>Current fuel level.</td>
</tr>
<tr>
<td>gear</td>
<td>Current gear: -1 is reverse, 0 is neutral and the gear from 1 to 6.</td>
</tr>
<tr>
<td>lastLapTime</td>
<td>Time to complete the last lap</td>
</tr>
<tr>
<td>opponents</td>
<td>Vector of 18 sensors that detects the opponent distance in meters (range is [0,100]) within a specific 10 degrees sector; each sensor covers 10 degrees, from (-\pi/2) to (+\pi/2) in front of the car.</td>
</tr>
<tr>
<td>racePos</td>
<td>Position in the race with respect to other cars.</td>
</tr>
<tr>
<td>rpm</td>
<td>Number of rotation per minute of the car engine.</td>
</tr>
<tr>
<td>speedX</td>
<td>Speed of the car along the longitudinal axis of the car.</td>
</tr>
<tr>
<td>speedY</td>
<td>Speed of the car along the transverse axis of the car.</td>
</tr>
<tr>
<td>track</td>
<td>Vector of 19 range finder sensors; each sensors represents the distance between the track edge and the car. Sensors are oriented every 10 degrees from (-\pi/2) and (+\pi/2) in front of the car. When the car is outside of the track (i.e., pos is less than -1 or greater than 1), these values are not reliable!</td>
</tr>
<tr>
<td>trackPos</td>
<td>Distance between the car and the track axis. The value is normalized w.r.t to the track width: it is 0 when car is on the axis, -1 when the car is on the left edge of the track and +1 when it is on the right edge of the car. Values greater than 1 or smaller than -1 means that the car is outside of the track.</td>
</tr>
<tr>
<td>wheelSpinVel</td>
<td>Vector of 4 sensors representing the rotation speed of the wheels.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>accel</td>
<td>Virtual gas pedal (0 means no gas, 1 full gas).</td>
</tr>
<tr>
<td>brake</td>
<td>Virtual brake pedal (0 means no brake, 1 full brake).</td>
</tr>
<tr>
<td>gear</td>
<td>Gear value.</td>
</tr>
<tr>
<td>steering</td>
<td>Steering value: -1 and +1 means respectively full left and right, that corresponds to an angle of 0.785398 rad.</td>
</tr>
<tr>
<td>meta</td>
<td>This is meta-control command: 0 do nothing, 1 ask competition server to restart the race.</td>
</tr>
</tbody>
</table>
Competition Rules

- Any controller that satisfied the communication protocol was accepted. It could be an Ai-design or a collection of rules authored by a human.

- **Two stages:**
  1. Each entrant did a number of qualifying runs for 1000 game tics, or 200 seconds of real time. The accumulated meters was the score for each run. Performance was compared against a pair of baseline controllers supplied by organizers.

  2. Those contestants who performed better than the baseline controllers were eligible for the final head to head races.

- Competitors had no idea what tracks would be selected for the time trials and the competition. Tracks varied in shape from ovals to city streets over a variety of different surfaces, dirt, concrete etc. Black ice?
2. Designing the State Machine Controller

- Design objectives:
  1). Get to the final head to head phase.
  2). We pragmatically biased our learning to optimize performance in the time trials.
  3). However, we used a socially motivated learning approach, cultural algorithms along with certain kinematic functions that supported basic flocking or swarming activities, **collision avoidance** and **orientation** among “boids”. It was hoped that this learned behavior would generalize over to the multi-car scenario even though we had not trained it with other cars.
  4). As a result while a solo driver might process all of the raw sensors, we tried to design for the “pack”. That is, information about track edges would be obstructed by others, so we wanted to induce information about what we can’t see from the position of nearby competitors.
  5). As a by product of this is that we will use fewer sensors, infer more based upon the position of other drivers, and hopefully reduce processing speed relative to those who use them all.
The basic design of the controller was a state machine.

We employed a state selector configuration in order to expedite reactions to changes in state which in head to head can happen quickly.

Overview of the basic controller states:

1) Idle was the selector state that determined the current state. “bridging”.
2. On_track
3. Off_track
4. Off_target
5. Stuck
Since we assume that we will always be either in or near the pack we reduced the number of sensors needed from over 30 to 6. We used all of the effectors.

- **Sensors**: trackpos, rpm, gear, angle, speedX
- **Effectors**: accel, brake, gear, steering

We derived two other variables from the raw sensor data:

- **turnAngle**: the angle with the furthest distance to edge of the track (in the trackPos variable).
- **maxSpeed**: determined by turnAngle, or an error correction state.
Kinematic Rule Functions

- These functions are used across all of the four states. Each function was hardcoded into the controller.

- **steering**(turnAngle)
  - if turnAngle = 0
    - steering = 0
  - else
    - steering = ((turnAngle/10)9)

- **shifting**(gear, rpm)
  - if gear = 0 or (gear = 2 and rpm < 3000)
    - gear = 1
  - else if gear > 0 and rpm > 7500
    - gear = gear + 1
  - else if gear > 2 and rpm < 4000
    - gear = gear -1

- **acceleration**(speedX, maxSpeed)
  - if(speedX < maxSpeed)
    - accel = 1
  - else
    - accel = 0
Utility Functions Continued

- **braking**(speedX, maxSpeed)
- if(speedX > maxSpeed + 30)
  - brake = 1
- else
  - brake = 0
Our goal was to learn to optimize the parameter values for the kinematic utility functions used in all of the states so as to maximize the distance travelled over a run.

The key parameters are Turn Angle and Maximum speed since they are used in all of the functions.

For each 10 degree increment in turn angle we wish to learn the maximum speed that we can go in that angle. Assume symmetry in turn angles (+ and -).

We will use a Cultural Algorithm with a Genetic Algorithm as the population component, and collect aggregate performance knowledge in the Belief Space. That knowledge will be used to guide the application of the genetic operators.
Holland [1975] developed a formal framework for any generic adaptive systems that is able to alter its structure and/or behavior based on the experience in some set of performance environments.

Scale of Social Interaction: The Emergent Properties Depend Upon the Scale at which the Interaction Takes Place
Cultural Algorithms are computational models of Social Evolution

- Major components

```
Begin
  t = 0;
  Initialize Population POP(t);
  Initialize Belief Space BLF(t);
repeat
  Evaluate Population POP(t);
  Adjust(BLF(t), Accept(POP(t)));
  Adjust(BLF(t));
  Variation(POP(t) from POP(t- 1));
  until termination condition achieved
End
```

www.ai.cs.wayne.edu
○ Regulates the movement of knowledge from the Population Space -> Belief Space

○ A subset of the population is chosen to impact the Belief Space. In an optimization problem it is generally a percentage of the top performers
Integrating the Knowledge Sources in the Influence Function

- Normative Knowledge
- Situational Knowledge
- Domain Knowledge
- Historical Knowledge
- Topographical Knowledge

accept() - update() - influence()
Hierarchical Integration of the Knowledge Sources

- Normative Knowledge
- Domain Knowledge
- History Knowledge
- Situational Knowledge
- Topographical Knowledge

www.ai.cs.wayne.edu
Roulette Wheel

KS1

KS2

KS3

KS4

KS5

www.ai.cs.wayne.edu
The Social Fabric Approach

- Individual are represented as nodes in the landscape.

- Number of connections or hops over which an individual can transmit information to its neighbors will correspond to its influence, by a maximum hop distance and will be limited.

- Number of hops can be either 0 or \(d\) meaning either no connections or \(d\) connections at a time.
Many possible topologies can be used:

- **Fixed social network based on a predetermined topology of interaction**
  - *lBest*
  - *Square*
  - *gBest*

Topologies used in the PSO approach (a) *lbest* ring topology. (b) *gbest* topology.
In case of a tie, tie breaking rules are used as follows:

- Direct Influence
- Least Frequently Used (LFU)
- Most Frequently Used (MFU)
- Random Influence.
Using the SF to Optimize the Cones World Problem

Population Swarm and Population Plot at Generation 10
Using the SF to Optimize the Cones World Problem

Population Swarm and Population Plot at Generation 81
4. Training the Controller

The Population Component

- Genetic Algorithms: evaluate pop => reproduce => modify
- A population of behavioral chromosomes:
  - Each gene corresponds to a turn angle (0, to $\pm 90$). Notice that we assume symmetry in the angles to simplify representation. The value for the gene is the maximum speed that the system can possess when in that turn angle.
  - The goal is to learn these governor relations in order to optimize the application of the kinematic functions within each state.
- An example chromosome:
  
  0  | $\pm 10$  | $\pm 20$  | $\pm 30$  | $\pm 40$  | $\pm 50$  | $\pm 60$  | $\pm 70$  | $\pm 80$  | $\pm 90$
  209 160 130 114 115 107  35   86   27   145

Performance function: Distance travelled in meters over a 200 sec run.
Chromosomes networked to two others in order to produce an lbest configuration.
Chromosome is completely replaced each time step by a new one.
The Acceptance Function

- This function determines how to aggregate the individual behaviors.

- Here, information about the performance of all 100 individuals in the population, their performance, and their performance environment is given to the belief space.
The Cultural Knowledge Component

- Domain Knowledge: Specific track information (surface etc) and indexing. Track knowledge taxonomies.
- Normative: Ranges of parameter values indexed by track.
- Situational: Best, worst, and average performance for indexed tracks.
- History: Statistics on track types raced on. E.G. If an unknown type then select most frequently used track type to start.
- Topographic: Not explicitly used here. Implicitly present in relation between Max_speed and turn_angle. In the future explicit terrain maps can be useful in directing strategy.

- Each knowledge source vies to control the generation of a new individual using the Social Fabric Influence Function
Cultural Algorithm Parameters

- **Social Fabric Influence Function:**

- Fitness proportional application of a knowledge source to individuals in the population.

- The social fabric used here is the *lbest* topology.

- When an individual is selected to be directly influenced by a Knowledge Source, it broadcasts that “*bid*” to its two neighbors (one hop). After each member of the entire population has distributed its direct bid to its neighbors then the total number of bids for each individual are summed up. For each individual, the knowledge source with the *majority* of bids wins.

- In case of a tie we use “*direct influence*” bidder as the winner.
Initial CAT Training

- Population size: 100
- Number of Generations: 100
- Topology: Lbest
- Hops: 1
- Tie breaking: Direct
- Track: Oval
- Run: 10000 game turns (game tics). 1000 per 200 seconds.
Parameter Learning

- Lbest configuration learning

![Graph showing distance traveled over generation number](chart.png)
Incremental Learning

- A succession of **bridging** actions, allows the system to build behaviors on behaviors.

- First increment (Generation 1 through 3): Values allow car to stay on track.

- Second increment (Generation 7 through 17): Learns to accelerate.

- Third increment: Track segment specialization (Generation 18 through 26): Learns to focus on specific track segments, e.g. straight, curved.
5.0 The Competition

- **Time trials:**
  - Three tracks were selected.
  - Each controller was run 10 times on each track (200 secs of game time each).
  - Performance was the median score over all 10 runs.
  - Two baseline controllers were used to screen entries. Those entries that cannot do better than the baseline controllers were not allowed to go on.

- **Head to head:**
  - Five controllers were selected. Three exhibited strong consistent behavior across the tracks.
  - 10 points to the winner, three laps around, 8 second etc.
  - Competitors randomly positioned on the tracks in each of 10 races on each of three tracks.
  - Median score for each contestant awarded for that track.
The D-Speedway

A simple oval track.
The Street1 Track

A more complex street-like track with sharp corners.
Oval-like but with a concave turn. Not easy.
Start Your Engines
<table>
<thead>
<tr>
<th>Entry</th>
<th>Ruudskogen</th>
<th>Street-1</th>
<th>Speedway</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinnaird-Heether et al.</td>
<td>6716.7</td>
<td>3692.9</td>
<td>14406.9</td>
</tr>
<tr>
<td>Lucas</td>
<td>4134.2</td>
<td>5502.8</td>
<td>12664.5</td>
</tr>
<tr>
<td>Simmerson</td>
<td>5934.0</td>
<td>6477.8</td>
<td>12523.3</td>
</tr>
<tr>
<td>Perez et al.</td>
<td>3786.9</td>
<td>2984.8</td>
<td>-317.3</td>
</tr>
<tr>
<td>Tan et al.</td>
<td>3443.5</td>
<td>2998-5</td>
<td>10648.2</td>
</tr>
<tr>
<td>C++ Sample Controller</td>
<td>4465.1</td>
<td>4928.8</td>
<td>7464.5</td>
</tr>
<tr>
<td>Java Sample Controller</td>
<td>5593.8</td>
<td>2963.2</td>
<td>5689.9</td>
</tr>
</tbody>
</table>
## Head to Head

<table>
<thead>
<tr>
<th>Entry</th>
<th>Ruudskogen</th>
<th>Street-1</th>
<th>Speedway</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simmerson</td>
<td>10</td>
<td>10</td>
<td>6</td>
<td>26</td>
</tr>
<tr>
<td>Kinnaird-Heether et al.</td>
<td>4</td>
<td>8</td>
<td>10</td>
<td>22</td>
</tr>
<tr>
<td>Lucas</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>Tan et al.</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>Perez et al.</td>
<td>5.5</td>
<td>4.5</td>
<td>5</td>
<td>14</td>
</tr>
</tbody>
</table>
Youtube Race Video

http://www.youtube.com/v/ruHzCF3CHIA&hl=en
Assessment

- **Positives**
  - Highest rated evolutionary computational entry.
  - Fast. We outperformed the other on the oval and Ruudskogen during time trials. Number one on oval track head to head.
  - Stayed out of logjams. Other controllers used same info and ended up in the same place.
  - Symmetry of turn angle decisions worked well for flat tracks.
  - Did better with the pack on the Street_1 track than alone. E.G. relied on pack information not present when training.
  - Learning took advantage of bridging behavior.

- **Negatives**
  - Failed to account for the aggressive behavior of the rest of the pack. Learning was biased towards a pack environment.
  - The hills and topography of the Ruudskogen was a problem with multiple cars.
  - Topographic information will be helpful with the latter.
  - Need 180 degree behavior for some tracks.
6.0 Do Networks Matter?

What was the impact of the population network used? The “right” network structure appeared to “focus” search more effectively in other “but simpler” problem solving situations.

So for each of the three tracks we compared:

Lbest
Gbest
Square

Does network structure matter as an optimization problem solving tool?
Performance of the three configurations on the D-Speedway
A More Complex Example

- Learning the Street Track

![Graph showing fitness over time for different scenarios.](http://www.themegallery.com)
A More Complex Track

- Learning the Ruudskogen

![Graph showing fitness over time with three lines representing different tracks: Libest, Square, and Ghost.](image-url)
Summary

- Adjustment of the network structure produces improved performance.
- All three performed well on the oval track.
- However, in more complex tracks when new improved speed angle combinations are found they don’t spread as quickly through the population with lbest.
- Gbest outperforms Square on the Oval track but not on the more complex ones.
- Too many new pairs, no time to establish a response.
- Square distributes but does not overload the system.

The key is that distribution of influence can help establish bridging behavior and allow incremental acquisition of skills in a complex environment. However, too much information does not allow performance to stabilize sufficiently to allow new learning to occur. Less distribution makes the system less responsive to new opportunities, bridging can start later.
Conclusions

- A Knowledge-based evolutionary learner was able to compete in a complex game.
- The Optimization process was based upon a socially networked population.
- The structure of the network was a factor effecting the learning activity.
- The network distributed new influences faster.
- Too few connections produced fewer “bridging stages on which to build new behaviors. (lbest).
- Too many connections provided less opportunity for bridging behaviors to be established. (gbest)
- Intermediate level structures provide more of a balance between bridging and dissemination. (square)
Future Work

- Use other knowledge sources more explicitly such as topographic knowledge.
- This will support the spatial structuring of experience (curves vs straight-aways).
- Allow the system to self-organize its social network to boost performance in an unknown system.
- Improve on the use of history knowledge to make predictions about an unknown track as the race goes on.
- Replace simple tie-breaking with n-person games. That is a good fit with the application.
- N-person games as an optimization search tool?
Related publications


www.ai.cs.wayne.edu
Related publications


- A Report on Cultural Algorithms Tutorial System (CAT), A report for the IEEE supported by W.J. Karplus Summer Research Grant
Thank You!