Combining Cooperative and Adversarial Coevolution in the Context of Pac-Man

by Alexander Dockhorn and Rudolf Kruse

Institute for Intelligent Cooperating Systems
Department for Computer Science, Otto von Guericke University Magdeburg
Universitaetsplatz 2, 39106 Magdeburg, Germany

Email: {alexander.dockhorn, rudolf.kruse}@ovgu.de

Contents

What is Pac-Man?

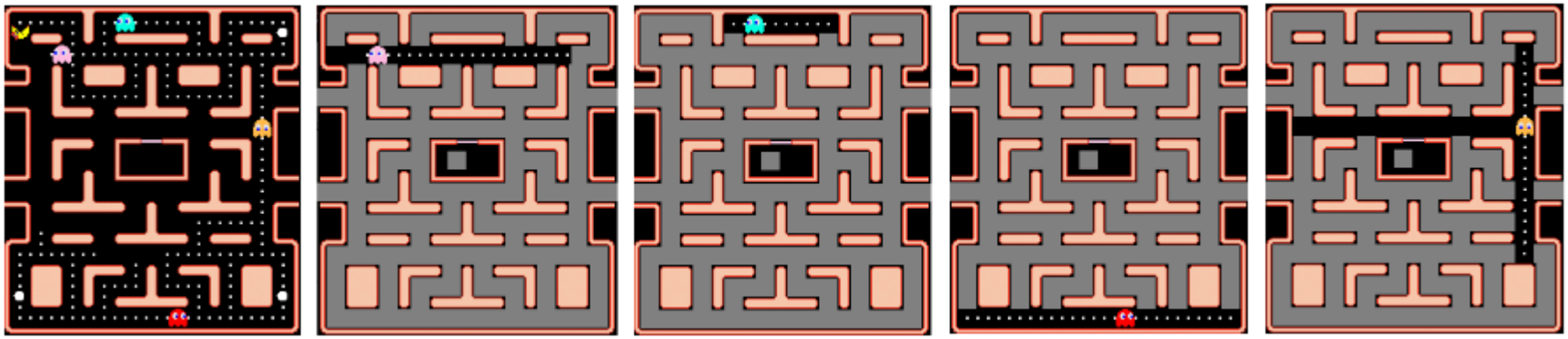
- Pac-Man is an arcade video game released by Konami in 1980.
- It yielded the second highest cross revenue of all arcade games (approx. 7.27 billion dollar).
- Pac-Man is the best known video character among American customers [source].

Pac-Man's Goals

- Pac-Man's task is to traverse a maze and eat all the pills.
- Four ghosts will hunt and try to stop him.
- Eating one of the our power pills will allow Pac-Man to eat ghosts for a short duration.
- Each of those actions scores Pac-Man points.
- After all pills were eaten, the next level starts.
- The game ends when no continues remain, after Pac-Man was eaten by a ghost.

Mrs. Pac-Man vs. Ghost Team Competition

- Since 2007 the Mrs. Pac-Man vs. Ghost Team Competitions.
- This work is part of this years competition, which features partial observation.
- The competition allows to program agents for Mrs. Pac-Man and the Ghost Team.
- In contrast to previous installments, agents will only receive information about objects in line of sight or general information about the map.



Related Work

- Previous Competition installments included agents based on:
  - State Machines [Gallagher and Ryan]
  - MCTS [Robles, Tong, Nguyen]
  - Neural Networks [Gallagher and Ledwich]
  - Ant Colony Algorithms
  - Genetic Programming [Alhejali, Brandstetter]

- It is not clear how well those solutions translate to the partial observation scenario!

Genetic Programming

- The behavior of each individual is encoded by a tree.
- The tree includes simple control structures using input by the game and points to an appropriate output.
- Evolutionary Algorithms are used to create a diverse set of trees while trying to improve the fitness of applied trees over time.
- Mutation and Crossover operators are used to modify parts of the trees.

Genetic Programming for Ghost Agents

- Implemented nodes should give access to all capabilities of the API, while being as general as possible.
- We differentiate function nodes, data terminal and action terminals.
- **Function Nodes:** include basic control functions (e.g. If…Then…Else…-nodes), and Boolean or Numeric operators
- **Data Terminals:** queries the API and the internal memory
- **Action Terminals:** perform a basic action, which is provided by the API

Mrs. Pac-Man Data and Action Terminals

Data Terminals:
- IsPowerPillStillAvailable
- AmICloseToPower
- AmIEmpowered
- IsGhostClose
- SeeingGhost
- DistanceToGhostNr<1,2,3,4>
- EmpoweredTime

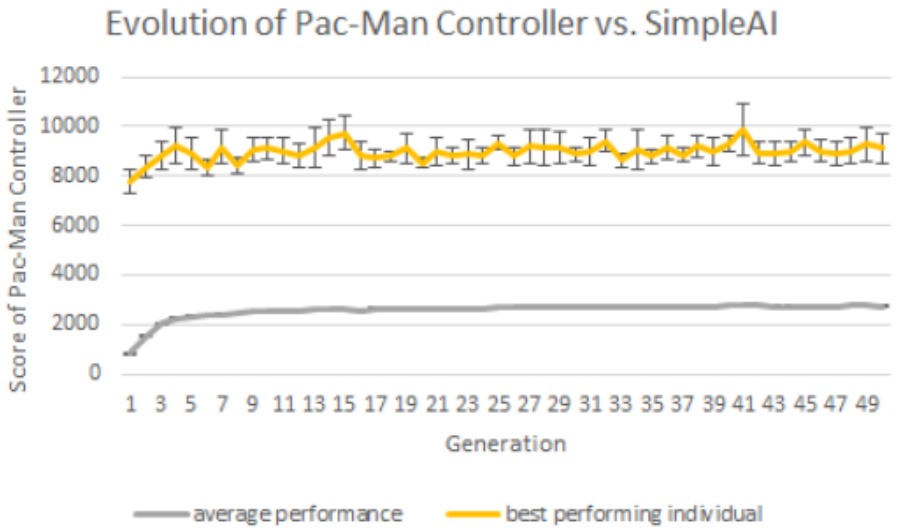
Action Terminals:
- FromClosestGhost
- ToClosestEdibleGhost
- ToClosestPowerPill
- ToClosestPill

This approach was adapted by previous competition submissions!

Due to partial observation restrictions we extended most Data Terminals with a short term memory:
- Remembers the last seen position of a ghost
- and simulates its behavior for a few ticks
- after a tick threshold is reached, the memory is cleared

Evaluation in a Partial Observation Scenario

- We first validated if the Genetic Programming works with partial observation.
- A ghost team of simple state machine agents were used as contrahent for evolved Pac-Man agents.
- The average performance as well as the performance of the best Pac-Man improved only slightly over time.



Ghost Team Data and Action Terminals

Data Terminals:

- SeeingPacMan
- IsPacManClose
- IsPacManCloseToPower
- IsEdible
- IsPowerPillStillAvailable
- DistanceToOtherGhosts
- EstimatedDistance

Action Terminals:

- ToPacMan
- FromPacMan
- FromClosestPowerPill
- ToClosestPowerPill
- Split
- Group

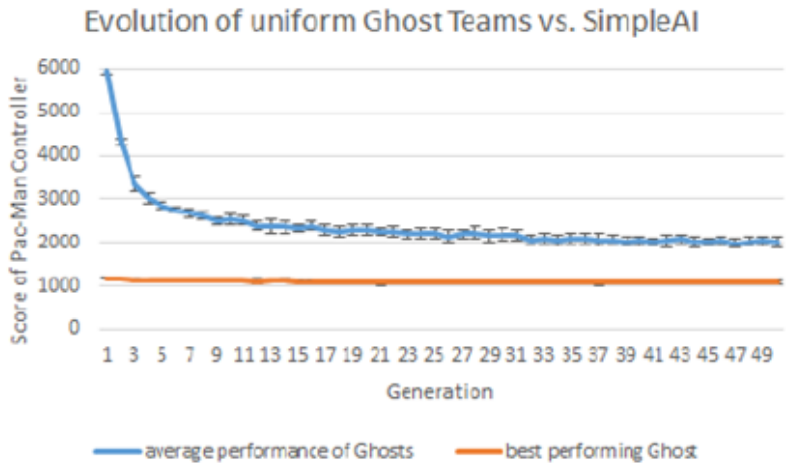
Evaluating Genetic Programming for Ghost Teams

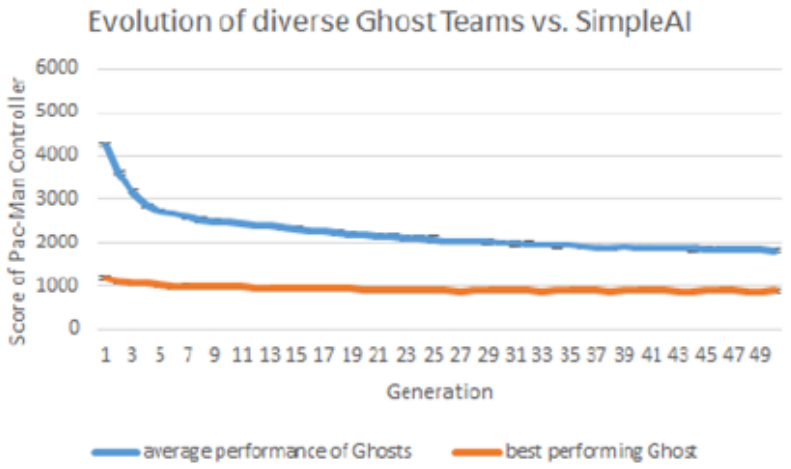
- Two Pac-Man agents were used as contrahents for evolved ghost teams.
  - SimpleAI = state machine agent
  - MCTSAI = Monte Carlo Tree Search agent

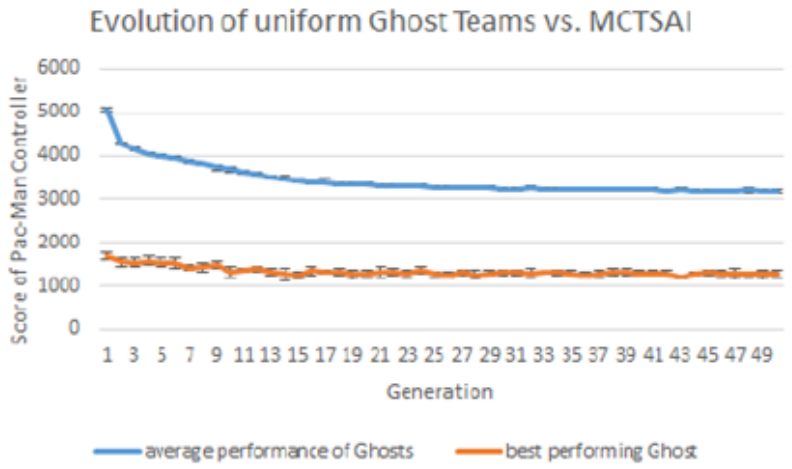
- Two approaches were compared:
  - uniform:
    - Ghost tTeams are made of four instances of the same individual
    - all individuals share the same population
    - ➢ single evolution

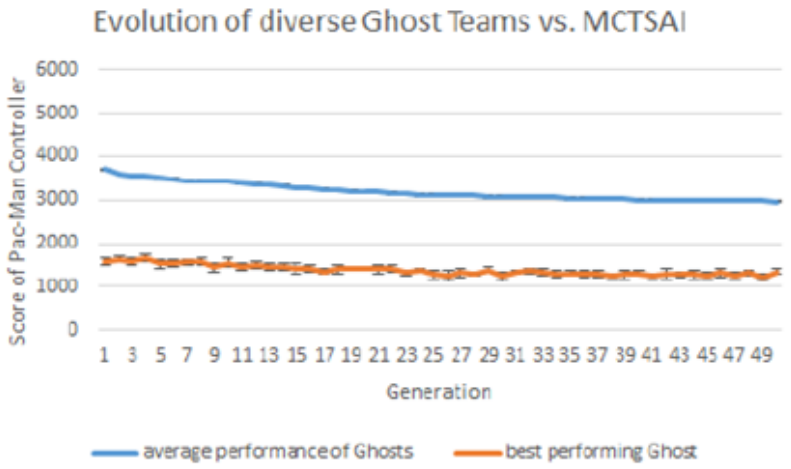
  - diverse:
    - Ghost Teams are made of four instances of different individuals
    - each individual is of one from four populations
    - ➢ cooperative coevolution

Single Evolution vs. Cooperative Coevolution









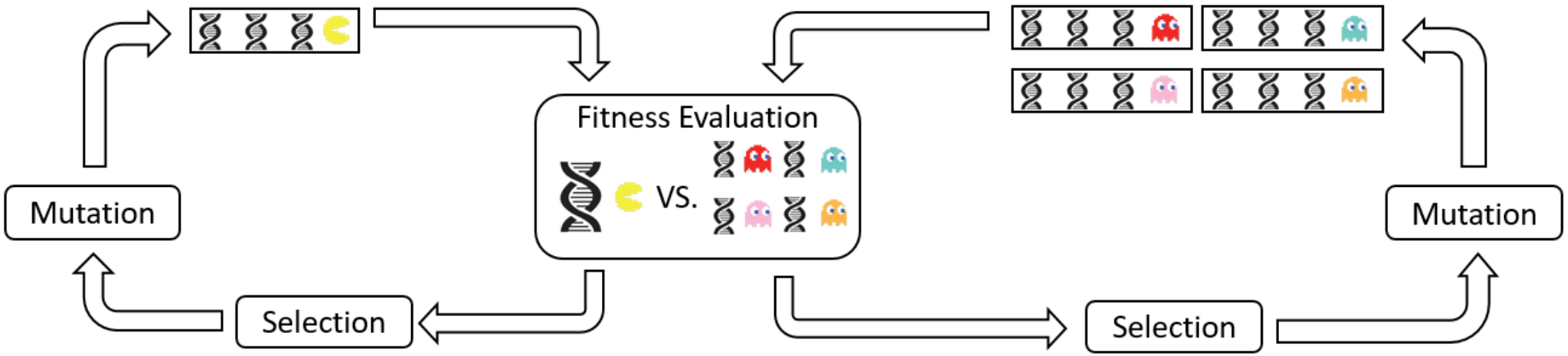
Genetic Programming Summary

- Agents for both agent parties can be learned using genetic programming.
- However, we need a suitable contrahent to assist the generation of complex behavior.

- Contrahents need to be hand-coded in the current framework.
  - Time consuming
  - Can miss possible strategies
  - Can be limited in the play-strength

- How can we combine both genetic programming procedures to get suitable Pac-Man agents **AND** Ghost Team agents?

Combined Coevolution Framework

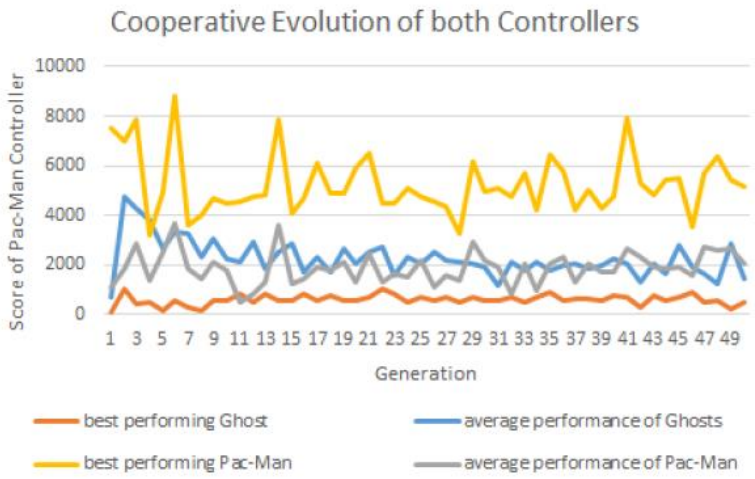


- Mrs. Pac-Man agents have one population

- Ghosts are split into 4 populations
- Each population exhibits its own strategy
- The best individuals per population will survive

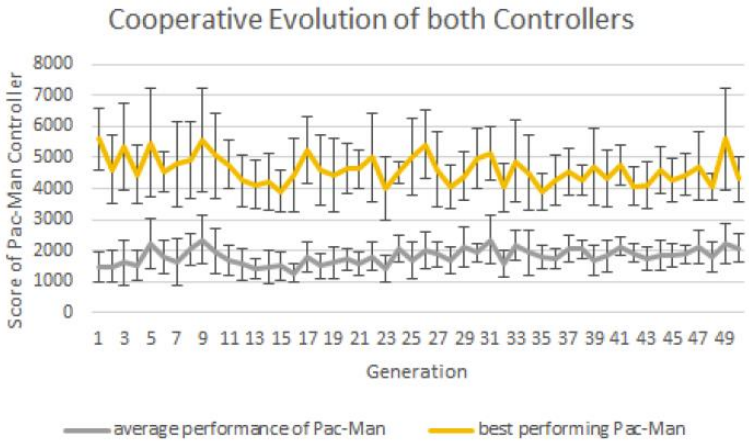
Combined Coevolution Framework

- The general idea:
  - ➢ When one agent type becomes stronger, their opponents need to react
- From our evaluation we can see bumps in Pac-Mans fitness values, which degrade over time
  - Those correspond to faster strategy changes in the beginning
  - And higher complexity in the end of the evolutionary process

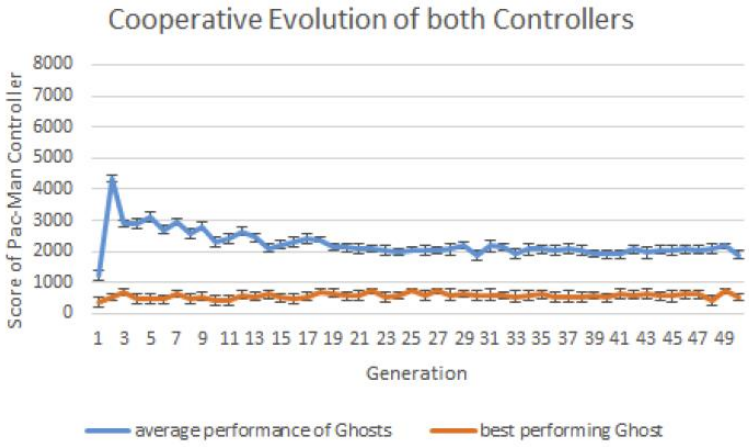


Combined Coevolution Framework

- We repeated the learning process 10 times to get insights in the general behavior of this learning process
  - Average points of Pac-Man and the Ghost Team converge over time
  - Best individuals per population quickly foster new strategies in the next generations
  - Overall complexity increases very slowly



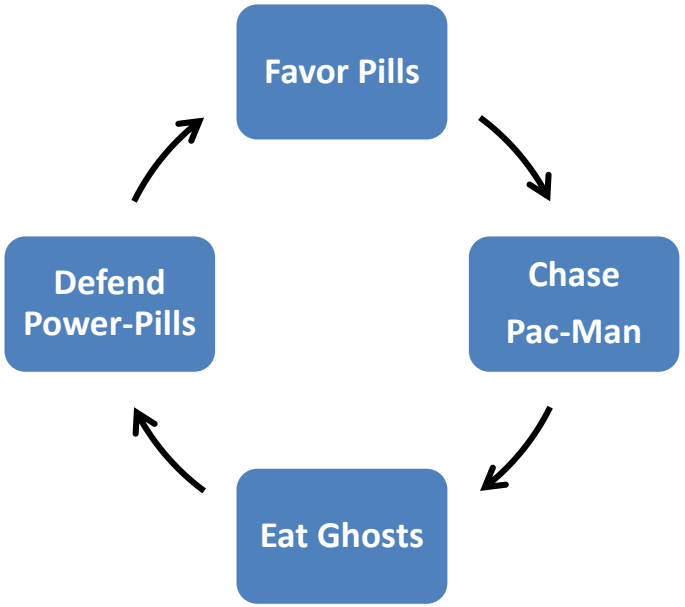
(a) Evolution of Pac-Man Controllers



(b) Evolution of Ghost Controllers

Insights

- The combined genetic programming reaches the same levels of complexity compared to single evolutionary processes.
  - …, but is increadible slow in doing so.
- Why does the complexity increase so slow?
  - Due to the scoring of the game, few basic strategies have a high return
  - This cycle dominates the first generations
- Open Question:
  - How can we promote complexity?



Conclusions

- Genetic Programming proved to be capable of generating simple and complex behavior in agents.

- Using four diverse ghost controllers was better and converged faster than using only one kind of ghosts
  - Either it is generally better to have mixed ghost teams
  - … or individuals from the single population need more time to built up comparable complexity

- Combining both genetic programming procedures potentially removes the need of creating suitable opponents.

Limitations and Open Research Questions

- Strategy loops hinder the combined framework in creating more complex strategies
  - Can those loops be detected during the evolutionary process?
  - Can we promote more complex solutions?

- Local maximas hinder the process
  - Exchange game induced scoring
  - Use a dynamic scoring function, which takes current strategies into account?

- How can other agent types be included, e.g. learning a multi-objective MCTS score function?

Thank you for your attention!

Check on Updates on our project at:
http://fuzzy.cs.ovgu.de/wiki/pmwiki.php/Mitarbeiter/Dockhorn
(Download of our project files will be made available soon)

by Alexander Dockhorn and Rudolf Kruse

Institute for Intelligent Cooperating Systems
Department for Computer Science, Otto von Guericke University Magdeburg
Universitaetsplatz 2, 39106 Magdeburg, Germany

Email: {alexander.dockhorn, rudolf.kruse}@ovgu.de