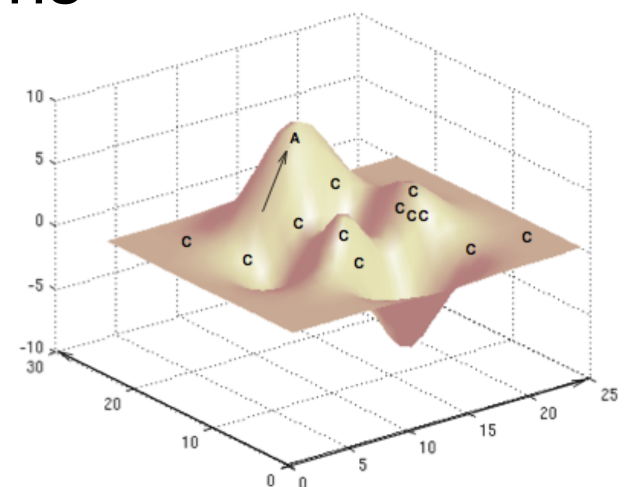
Competitive Multi-Agent Search

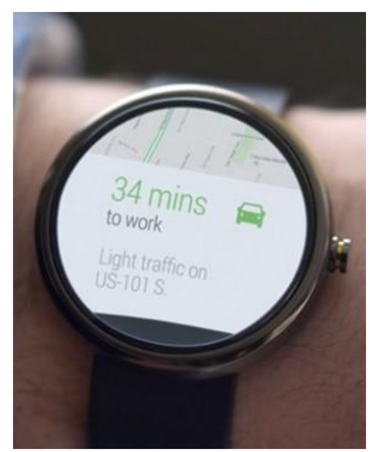
Erkin Bahçeci

November 26, 2014

Motivation

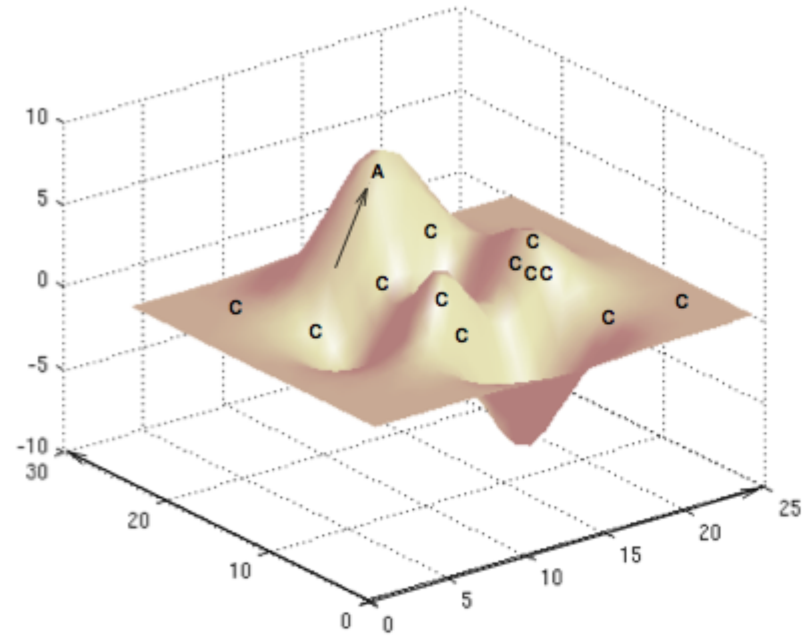
- Example: Innovation search
  - Firms (agents) looking for products (solutions)
  - Agents not in isolation, competing with other agents
- Past work: Aggregate NK simulations
- Scientific goals
  - New domain formalization
  - Method to optimize strategies
  - Help humans/companies do better





Competitive Multi-agent Search

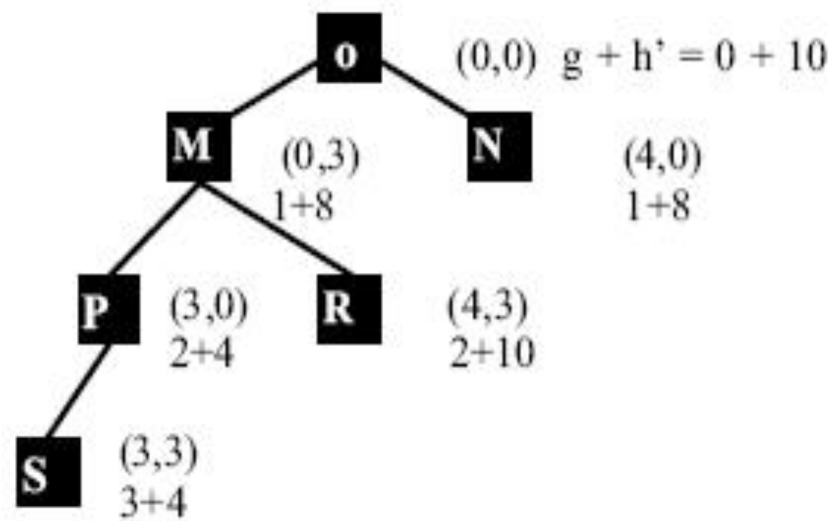
- Multiple agents
- Simultaneous search
- Competing for the same solutions (i.e. peaks)
- Hypothesis:
  - We can use the CMAS formalization to understand such domains, and use evolution to discover good strategies.



Single-Agent Search Methods

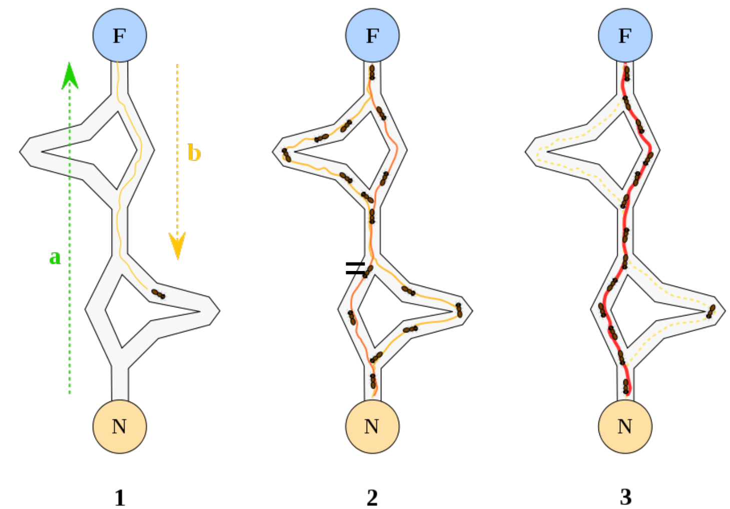
- e.g.: A* and IDA* algorithms
- Optimal solutions guaranteed at small scale but doesn't scale up
- Competition: e.g. 2 (or more) player games (minimax)

- Does not support
  - Multiple agents
  - Dynamic fitness landscape (altered by agents)
  - Large search space

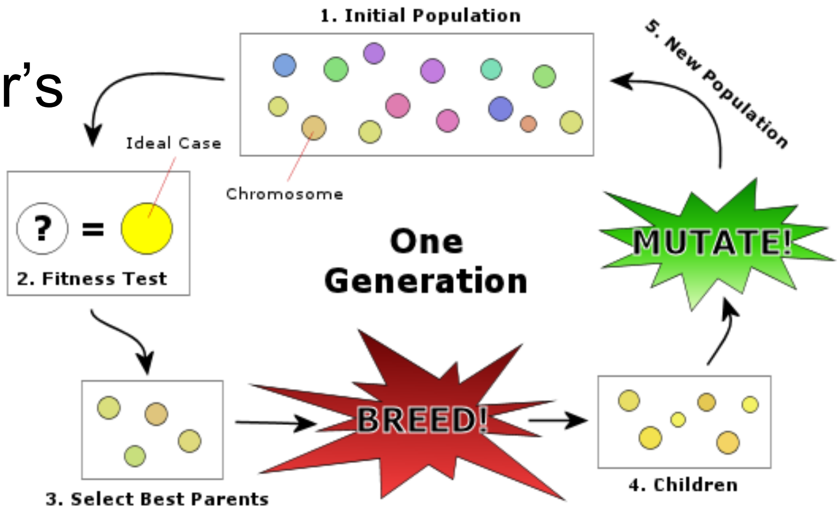


Team Search Methods

- e.g.
  - Particle Swarm Optimization [1]
  - Ant-colony Optimization [2]
  - Multi-agent Real Time A* [3]
  - Evolutionary methods

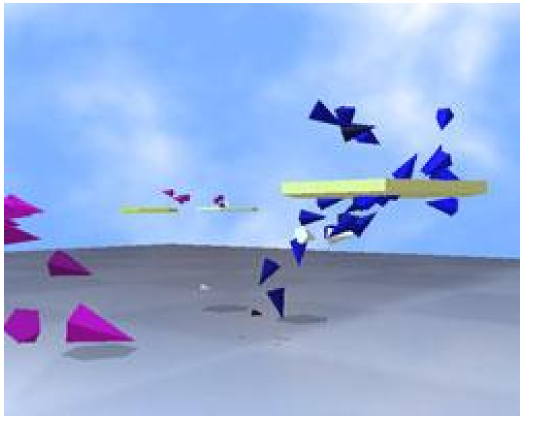


- Does not support dynamic fitness landscape
- Agents do not influence each other's search



[1] Kennedy, J., Eberhart, R., et al. (1995). Particle swarm optimization
[2] Dorigo, M., and Stützle, T. (2004). Ant colony optimization
[3] Knight, K. (1993). Are many reactive agents better than a few deliberative ones?

Agent-based Modeling

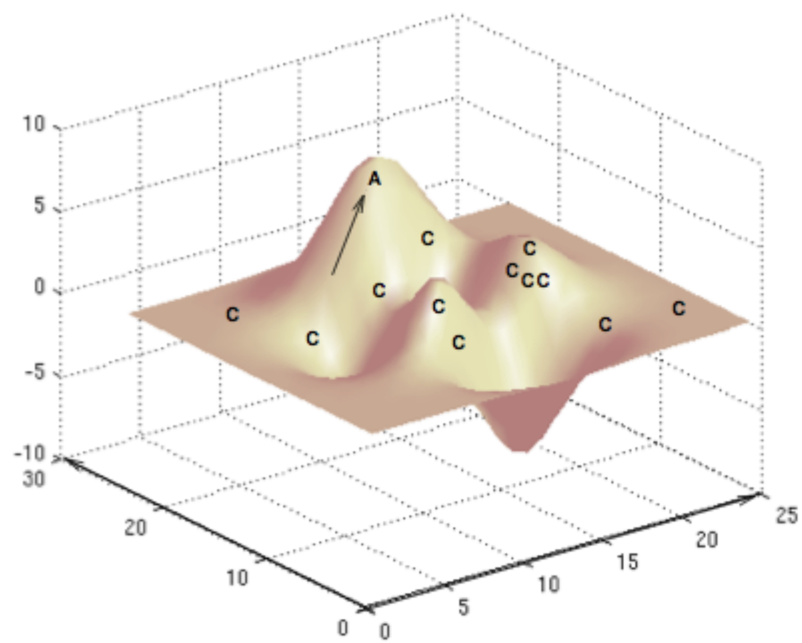


- Can explain emergence of higher order patterns
- Artificial Life
  - Rule-based: e.g. cellular automata, boids (flocking)
  - Neural net-based: e.g. Creatures
- Political science, economics, sociology [1,2,3]
  - e.g. seasonal migrations, pollution, sexual reproduction, combat, ethnocentrism, and transmission of disease and culture

[1] Axelrod, R. (1997). The complexity of cooperation: Agent-based models of competition and collaboration
[2] Epstein, J. (1999). Agent-based computational models and generative social science
[3] Tesfatsion, L. (2002). Agent-based computational economics: Growing economies from the bottom up

Formalization of CMAS

- Multiple agents searching for the same peaks on the same landscape
- Agent actions
  - Moving to a new point in the space
- Direct interactions
  - Knowledge/memory
- Indirect interactions
  - Landscape changes



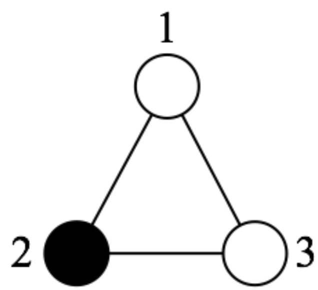
Outline

- Abstract domain
  - Simulation on NK landscapes
  - Experiments to characterize various effects
  - Evolving strategies in various environments against extreme opponents
- Concrete human game domain
  - Simulation of multi-player game
  - Modeling of human subjects
  - Evolving strategies against human subject models

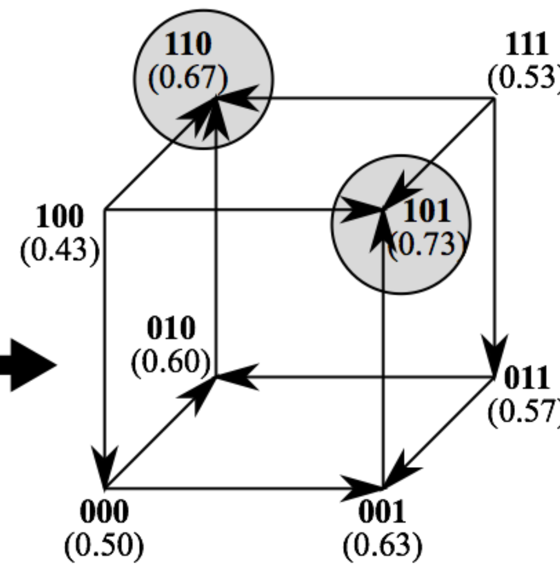
NK Model

- Boolean space (e.g. point or solution: 0100100111)
- N: Number of dimensions
- K: Ruggedness (number of correlated dimensions)

N=3, K=2

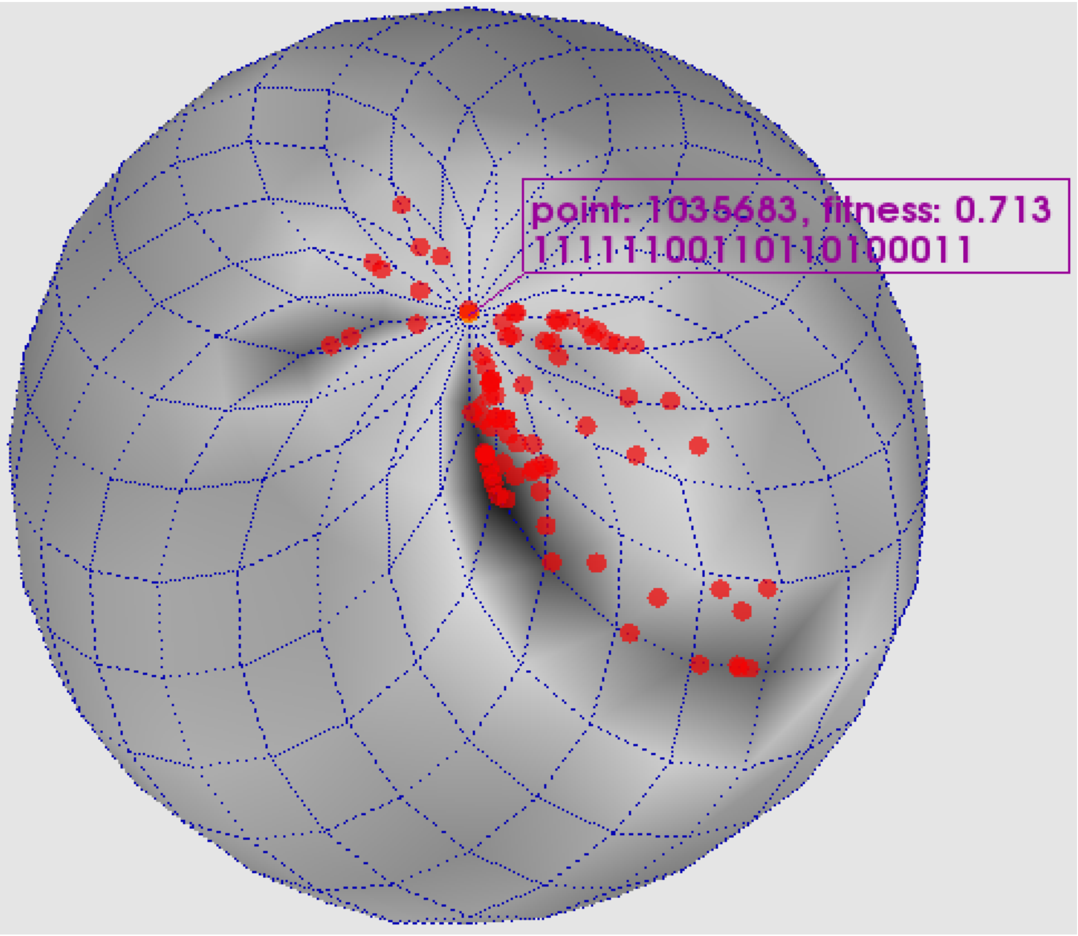


Element			Fitness contribution			Overall fitness
1	2	3	F_1	F_2	F_3	$F = \frac{1}{N}\sum_{i=1}^{N} F_i$
0	0	0	0.6	0.7	0.2	0.50
0	0	1	0.8	0.3	0.8	0.63
0	1	0	0.7	0.4	0.7	0.60
0	1	1	0.4	0.6	0.7	0.57
1	0	0	0.3	0.8	0.2	0.43
1	0	1	0.5	0.9	0.8	0.73
1	1	0	0.5	0.7	0.8	0.67
1	1	1	0.5	0.6	0.5	0.53



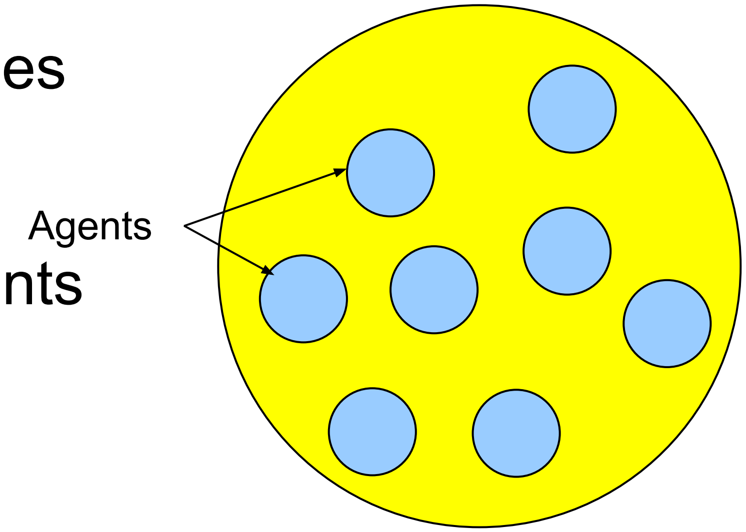
Shperical NK Visualization

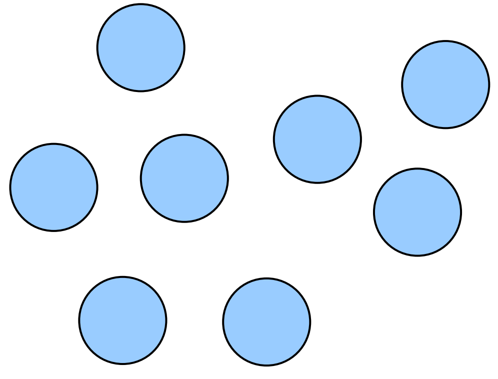
- High resolution and continuous near a specific point
- Low resolution farther away
- Points shown within closest diamond-shaped region



Knowledge About Others

- Memory
  - List of points and their fitness values
- Public memory
  - Common
  - Sharing knowledge among all agents
  - e.g. patents
- Private memory
  - Unique to each agent
  - e.g. trade secrets

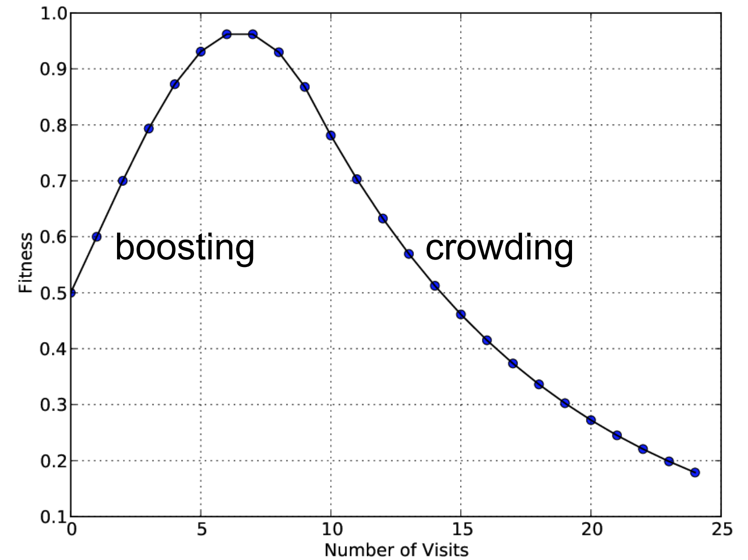


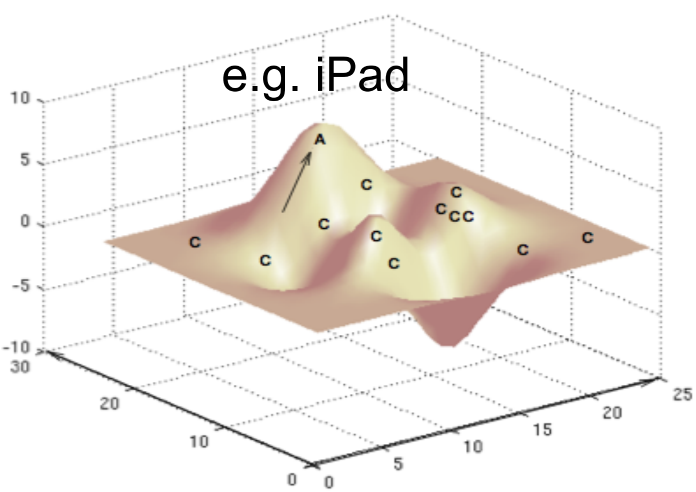


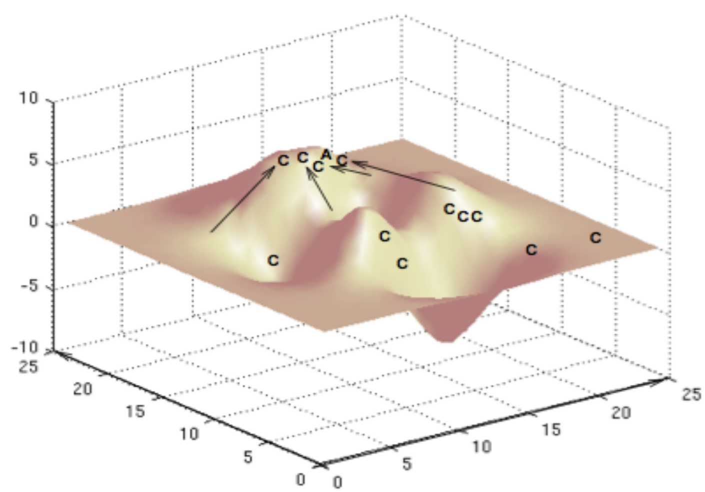
Changing Landscape

- Parameters
  - Flocking intensity
    - multiplied with fitness
  - Flocking radius
    - specifies size of affected area

- Decaying flocking
  (First boosting, then crowding)







Strategy

- S1: Select search method and starting point (memory) probabilistically, e.g:

Action: / State:	Exploit with public memory	Exploit with private memory	Explore with public memory	Explore with private memory	
Public: low fitness Private: low fitness	0	0	0.8	0.2	← Sum = 1.0
Public: low fitness Private: high fitness	0	0	0.8	0.2	← Sum = 1.0
Public: high fitness Private: low fitness	0.8	0.2	0	0	← Sum = 1.0
Public: high fitness Private: high fitness	0.8	0.2	0	0	← Sum = 1.0

- S2: Decide where to put new point probabilistically, e.g:

Action: / State:	Place in public memory	Place in private memory	
Point fitness: Low	0.7	0.3	← Sum = 1.0
Point fitness: High	0.0	1.0	← Sum = 1.0

Strategy Steps

1. Pick a search method and source memory probabilistically using S1 strategy component.
2. Perform one search step starting with the best point in the source memory.
3. **if** found a better point than the last one, **then**
   a. Pick a destination memory probabilistically using S2 strategy component.
   b. Place the new point in the destination memory.
4. **end if**

S1 strategy component

Action: / State:	Exploit with public memory	Exploit with private memory	Explore with public memory	Explore with private memory
Public: low fitness Private: low fitness	0	0	0.8	0.2
Public: low fitness Private: high fitness	0	0	0.8	0.2
Public: high fitness Private: low fitness	0.8	0.2	0	0
Public: high fitness Private: high fitness	0.8	0.2	0	0

Agent's current S1 state

Strategy Steps

1. Pick a search method and source memory probabilistically using S1 strategy component.
2. Perform one search step starting with the best point in the source memory.
3. **if** found a better point than the last one, **then**
    a. Pick a destination memory probabilistically using S2 strategy component.
    b. Place the new point in the destination memory.
4. **end if**

S1 strategy component

Chosen S1 action

State: \ Action:	Exploit with public memory	Exploit with private memory	Explore with public memory	Explore with private memory
Public: low fitness Private: low fitness	0	0	0.8	0.2
Public: low fitness Private: high fitness	0	0	0.8	0.2
Public: high fitness Private: low fitness	0.8	0.2	0	0
Public: high fitness Private: high fitness	0.8	0.2	0	0

Agent's current S1 state

Strategy Steps

1. Pick a search method and source memory probabilistically using S1 strategy component.
2. Perform one search step starting with the best point in the source memory.
3. **if** found a better point than the last one, **then**
   a. Pick a destination memory probabilistically using S2 strategy component.
   b. Place the new point in the destination memory.
4. **end if**

Agent's chosen S1 action:
Exploit private memory

Agent picks best point in private memory.

Strategy Steps

1. Pick a search method and source memory probabilistically using S1 strategy component.
2. Perform one search step starting with the best point in the source memory.
3. **if** found a better point than the last one, **then**
   a. Pick a destination memory probabilistically using S2 strategy component.
   b. Place the new point in the destination memory.
4. **end if**

Agent's chosen S1 action:
Exploit private memory

Agent exploits (flips one bit of the current point) to get new points.

Strategy Steps

1. Pick a search method and source memory probabilistically using S1 strategy component.
2. Perform one search step starting with the best point in the source memory.
3. **if** found a better point than the last one, **then**
   a. Pick a destination memory probabilistically using S2 strategy component.
   b. Place the new point in the destination memory.
4. **end if**

Agent's chosen S1 action:
Exploit private memory

Action: / State:	Place in public memory	Place in private memory
Point fitness: Low	0.7	0.3
$\boxed{\text{Point fitness: High}}$	0.0	1.0

Agent's current S2 state

Strategy Steps

1. Pick a search method and source memory probabilistically using S1 strategy component.
2. Perform one search step starting with the best point in the source memory.
3. **if** found a better point than the last one, **then**
   a. Pick a destination memory probabilistically using S2 strategy component.
   b. Place the new point in the destination memory.
4. **end if**

Agent's chosen S1 action:
Exploit private memory

Chosen S2 action

Action: / State:	Place in public memory	Place in private memory
Point fitness: Low	0.7	0.3
Point fitness: High	0.0	1.0

Agent's current S2 state

Outline

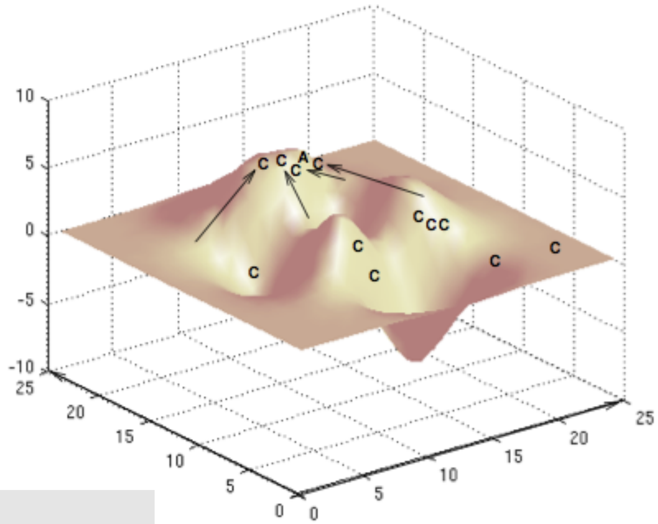
- Abstract domain
  - Simulation on NK landscapes
  - Experiments to characterize various effects
  - Evolving strategies in various environments against extreme opponents
- Concrete human game domain
  - Simulation of multi-player game
  - Modeling of human subjects
  - Evolving strategies against human subject models

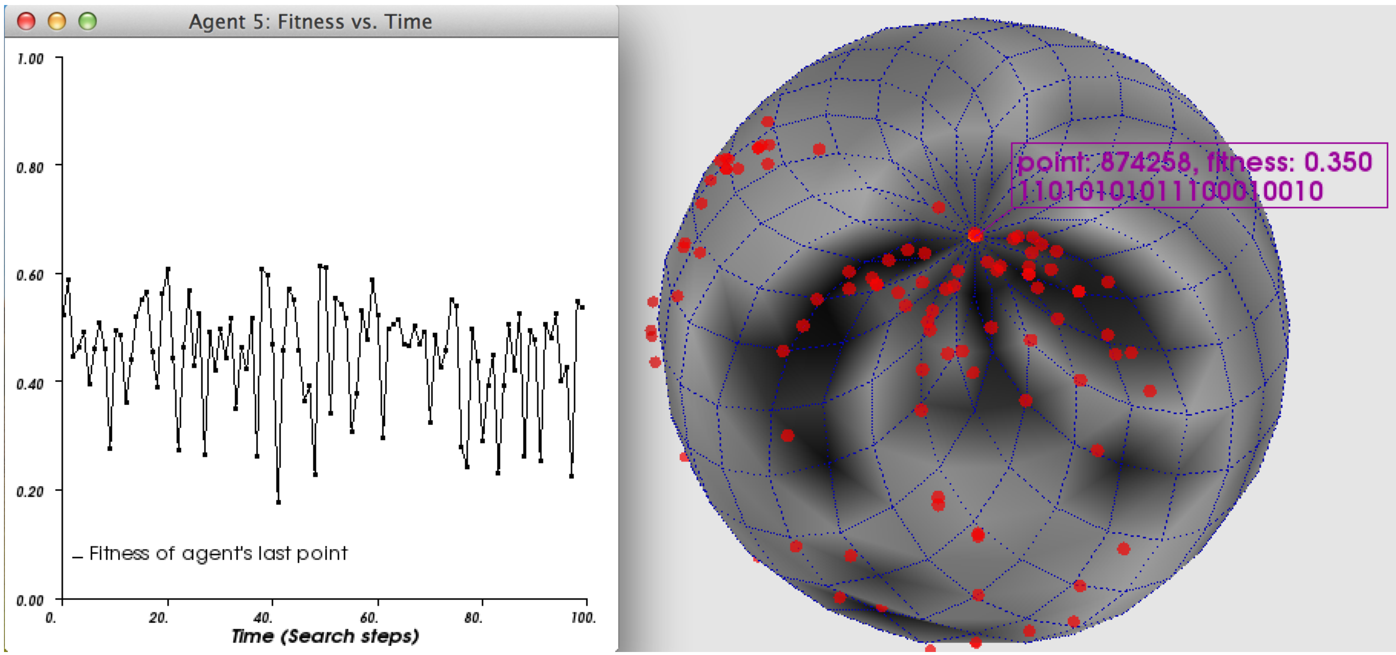
Experiments

- Systematically characterize effects of
  1. **Memory**
  2. Search method
  3. An intuitive strategy
  4. Exploration focus
  5. Environments with (extreme) opponents that exploit or explore with public memory, private memory, or both

Effect of Public vs. Private Memory

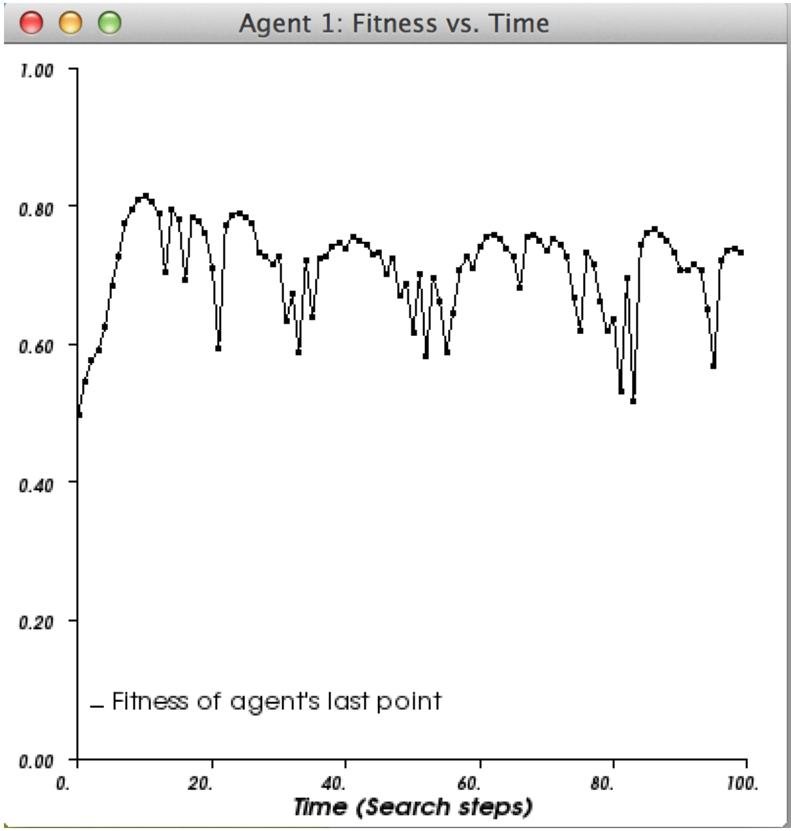
- Using only public memory leads to
  - Low diversity
  - "Twitter effect"
  - Low performance

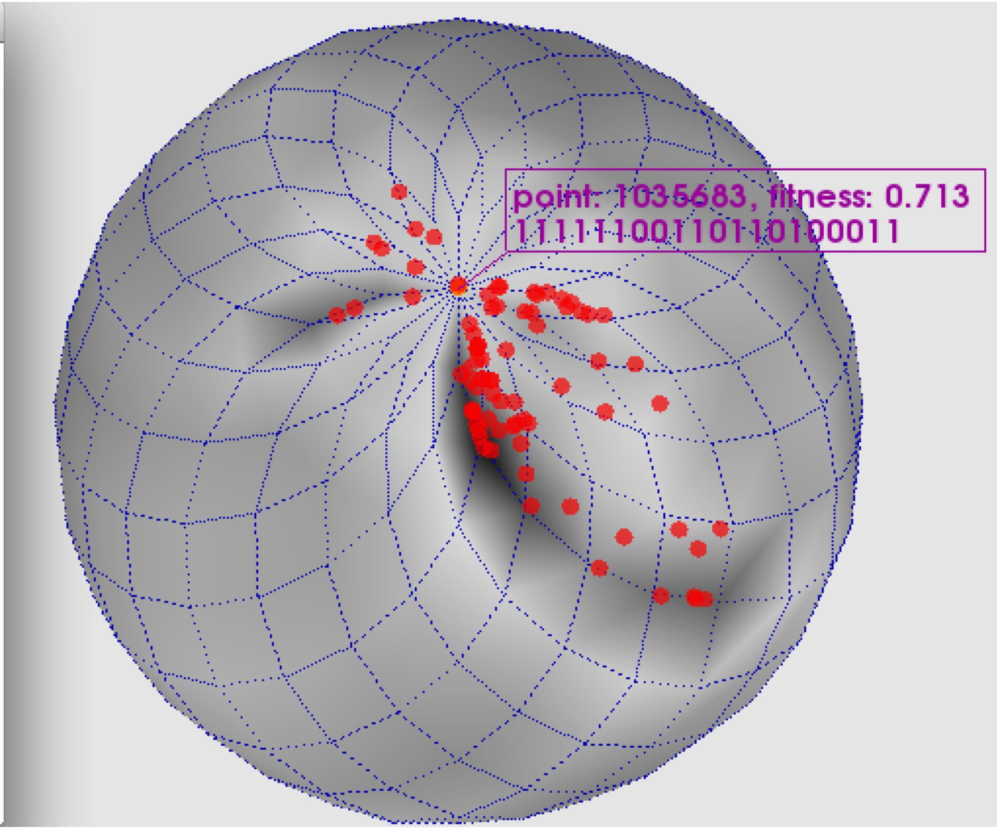




Wave-Riding Behavior

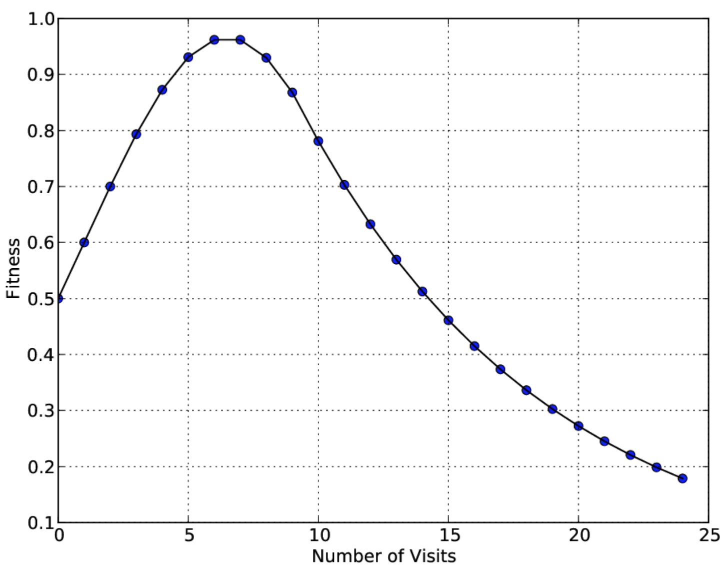
- Exploiting private memory in landscapes with low agent density leads to
  - riding on top of a wave of boosted fitness

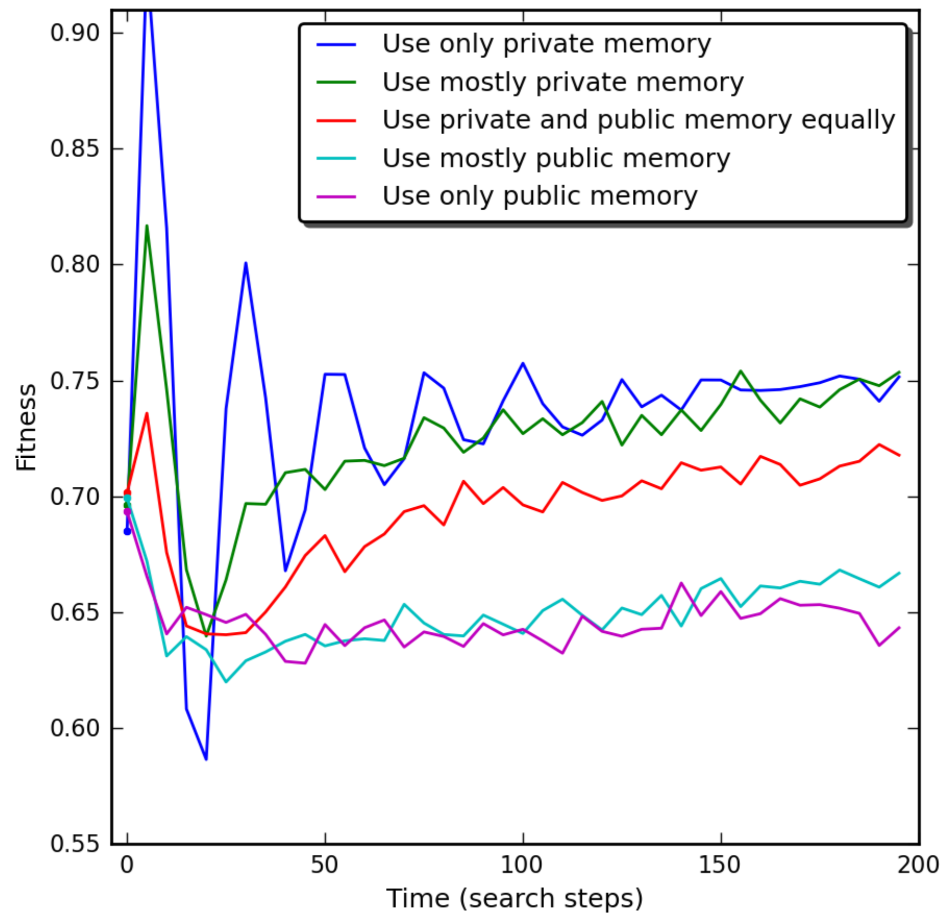




Oscillations

- Caused by landscape changes: boosting & crowding



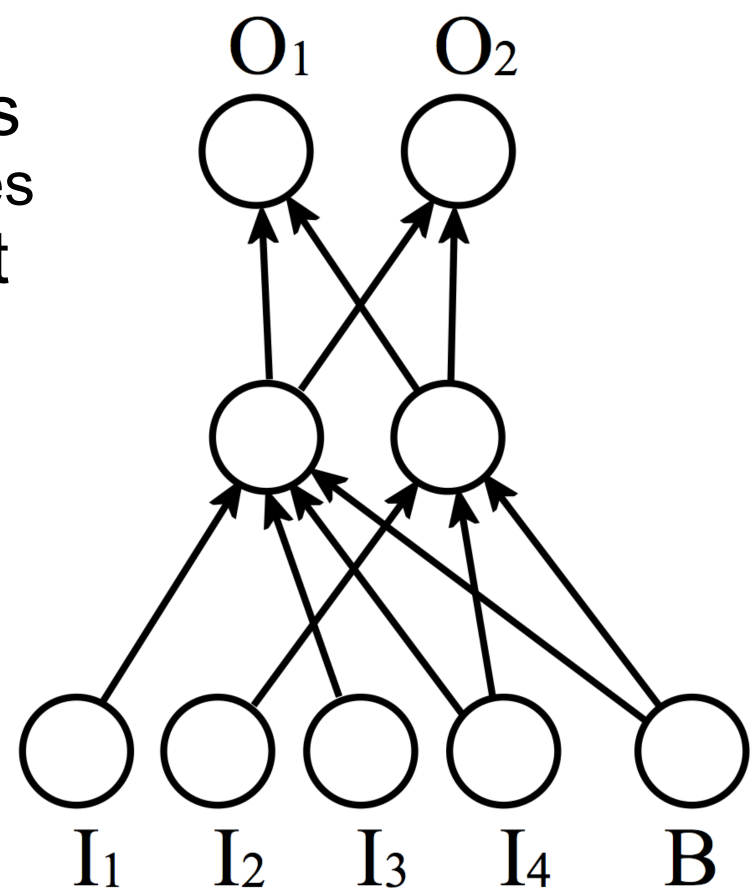


Outline

- Abstract domain
  - Simulation on NK landscapes
  - Experiments to characterize various effects
  - Evolving strategies in various environments against extreme opponents
- Concrete human game domain
  - Simulation of multi-player game
  - Modeling of human subjects
  - Evolving strategies against human subject models

Evolution Experiments

- 8 agents
- Evolve CPPNs with NEAT
- Environments
  - Homogeneous environments
    - Opponents: extreme strategies
  - Heterogeneous environment
  - Multiple homogeneous environments
- 64 evolutionary runs
- 500 generations
- Population size: 100

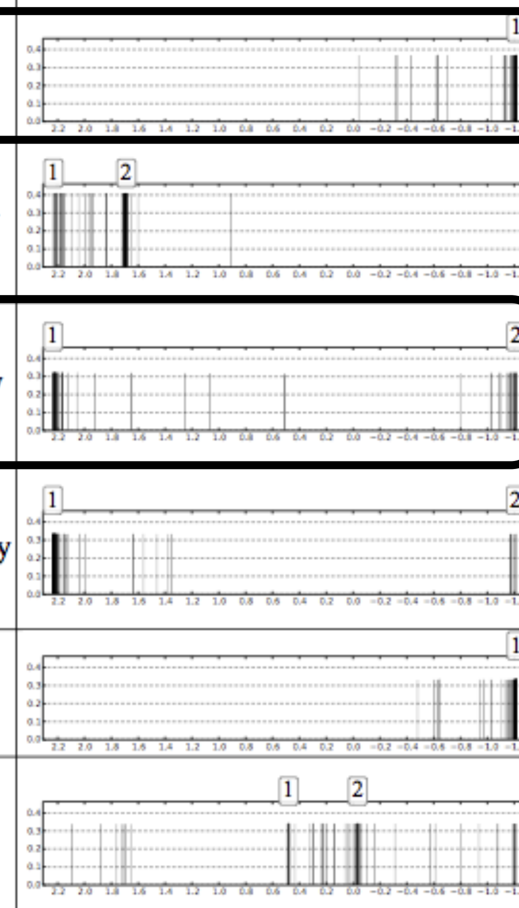


Evolved Strategies

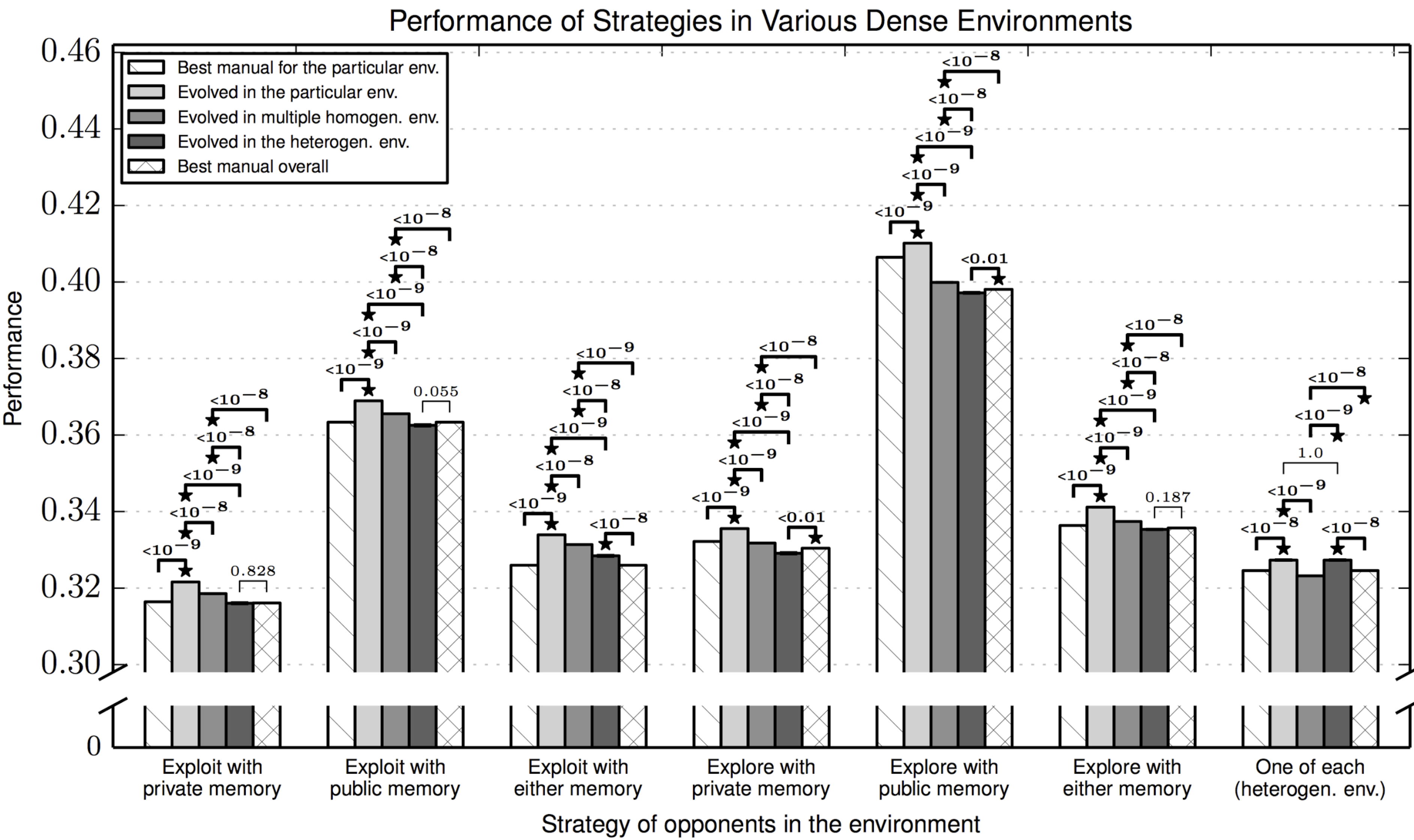
- Environment 1 (Opponents exploiting only public memory)
  - Avoid public memory

- Environment 3 (Opponents exploiting only their private memory)
  - Bimodal behavior

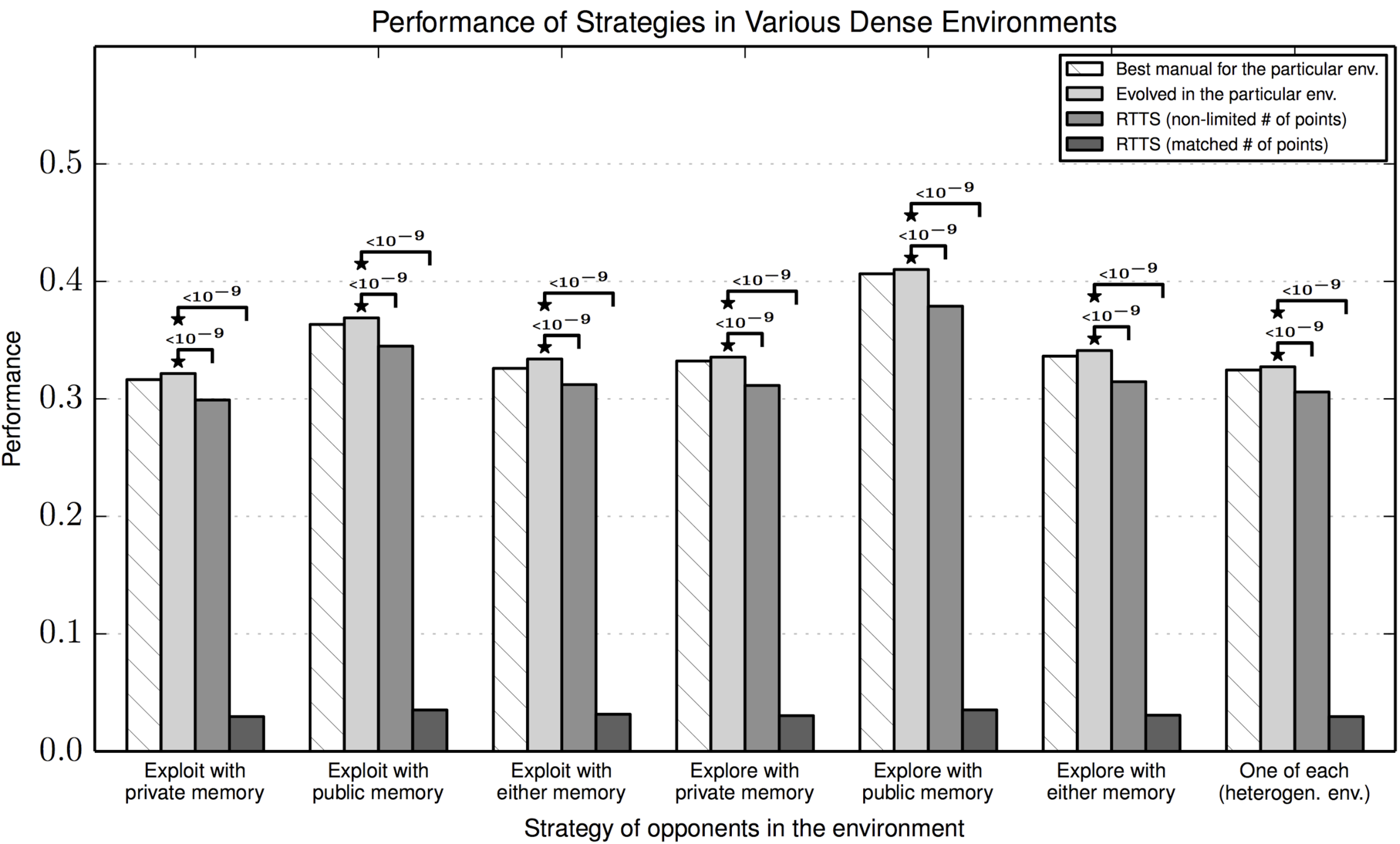
Dense Environment		PCA for All Evolved Strategies
Id	**Opponents**	
		⟵ Public Private ⟶
1	**Exploit** with **public** memory	
2	**Explore** with **public** memory	
3	**Exploit** with **private** memory	
4	**Explore** with **private** memory	
5	**Exploit** with **either** memory	
6	**Explore** with **either** memory	



Evolution Results



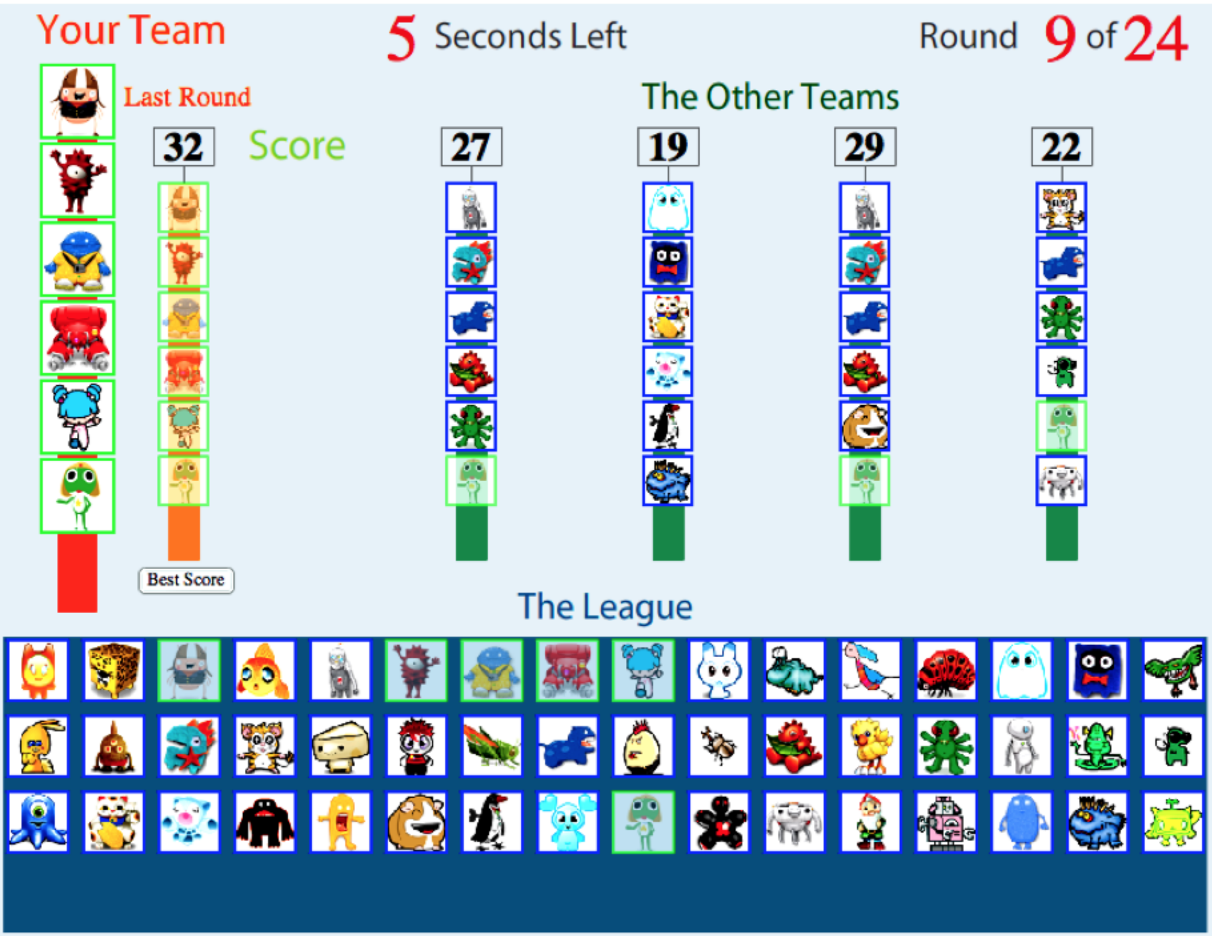
Comparison with Tree Search



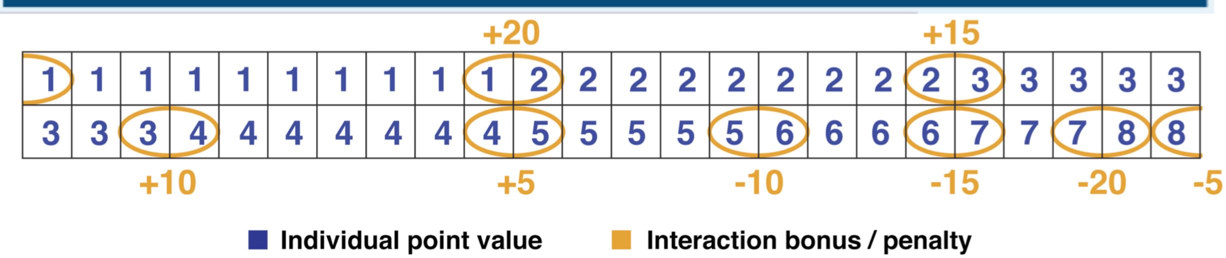
Outline

- Abstract domain
  - Simulation on NK landscapes
  - Experiments to characterize various effects
  - Evolving strategies in various environments against extreme opponents
- Concrete human game domain
  - Simulation of multi-player game
  - Modeling of human subjects
  - Evolving strategies against human subject models

Concrete Domain:
Social Innovation Game

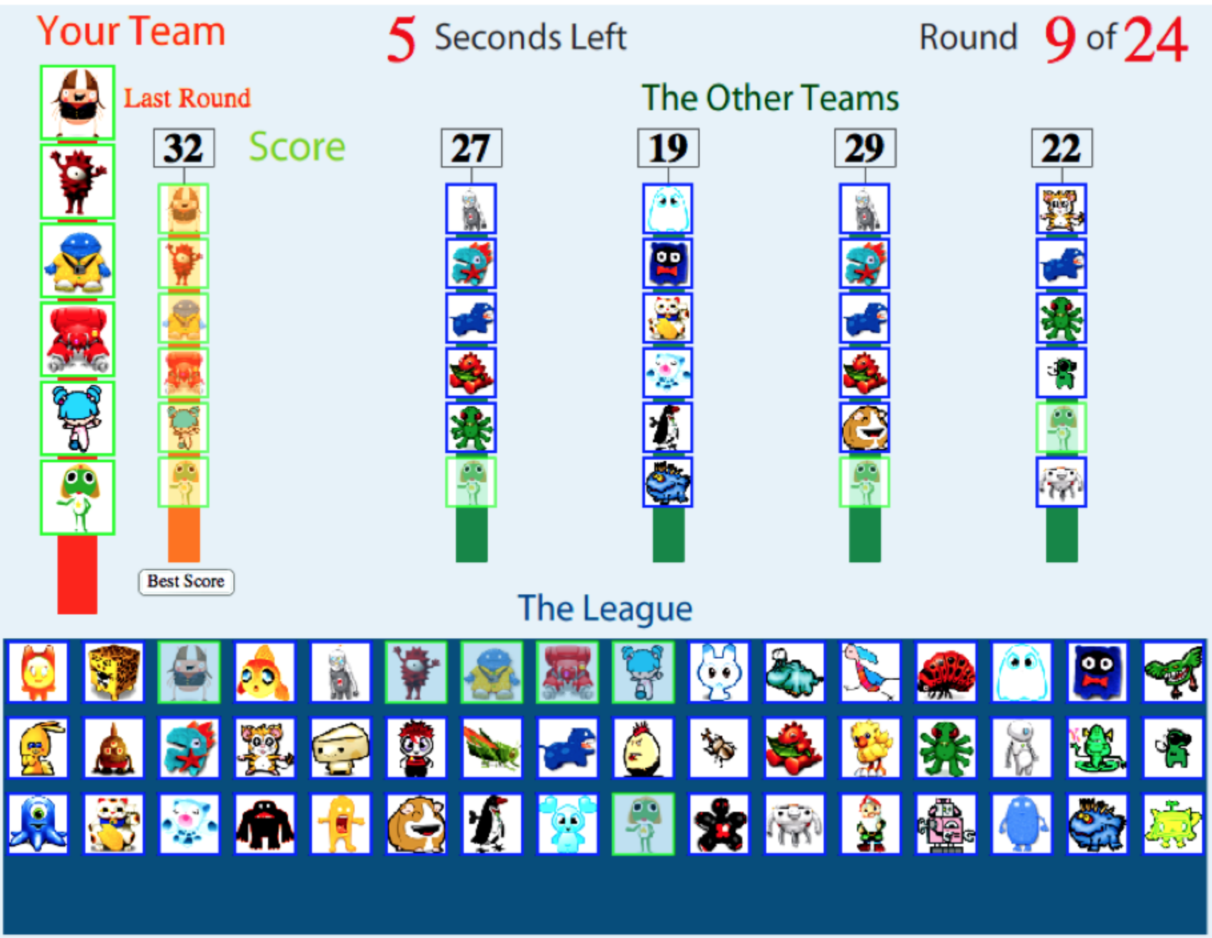


- Goal:
  Build a team with high score
- Score
  - Calculation:
    Icon & pair contributions (unknown by players)
  - Shown at the end of each round
  - Opponents' last teams & scores visible



[1] Wisdom, T. N., X. Song, and R. L. Goldstone (2013). Social learning strategies in networked groups. Cognitive Science 37, 1383–1425

Concrete Domain: Social Innovation Game



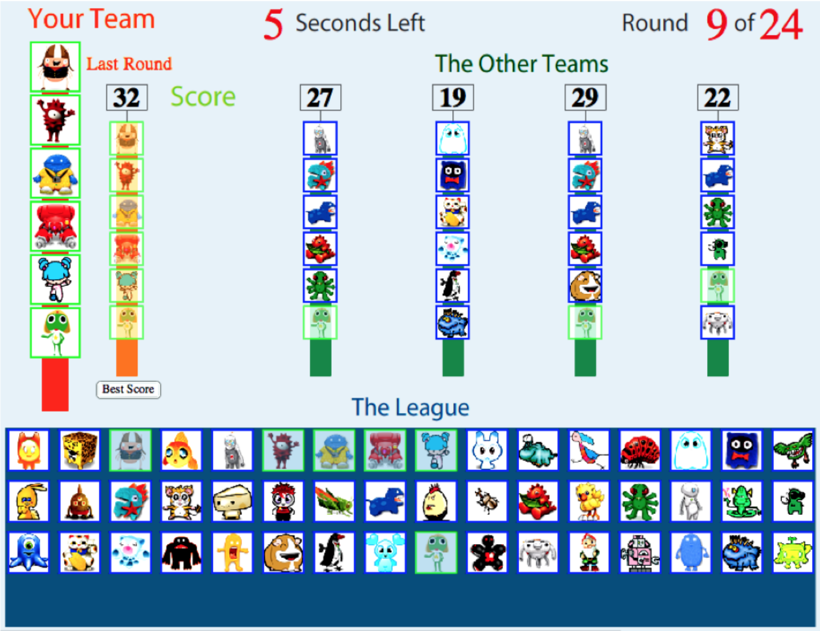
- Source (action) for each icon
  - "Innovate" from league
  - "Imitate" from an opponent
  - "Retrieve" from the best scoring team so far
  - "Retain" from previous round

- 39 sessions
- 1-9 players
- 8 games
- 24 rounds (10 s)
- 5 or 6 icons

[1] Wisdom, T. N., X. Song, and R. L. Goldstone (2013). Social learning strategies in networked groups. Cognitive Science 37, 1383–1425

Concrete Domain:
Social Innovation Game

- Real world application of CMAS
- Game played by human subjects
- Solution: 48-bit number with six 1 bits
- Difference: Static fitness landscape



[1] Wisdom, T. N., X. Song, and R. L. Goldstone (2013). Social learning strategies in networked groups. Cognitive Science 37, 1383–1425

Outline

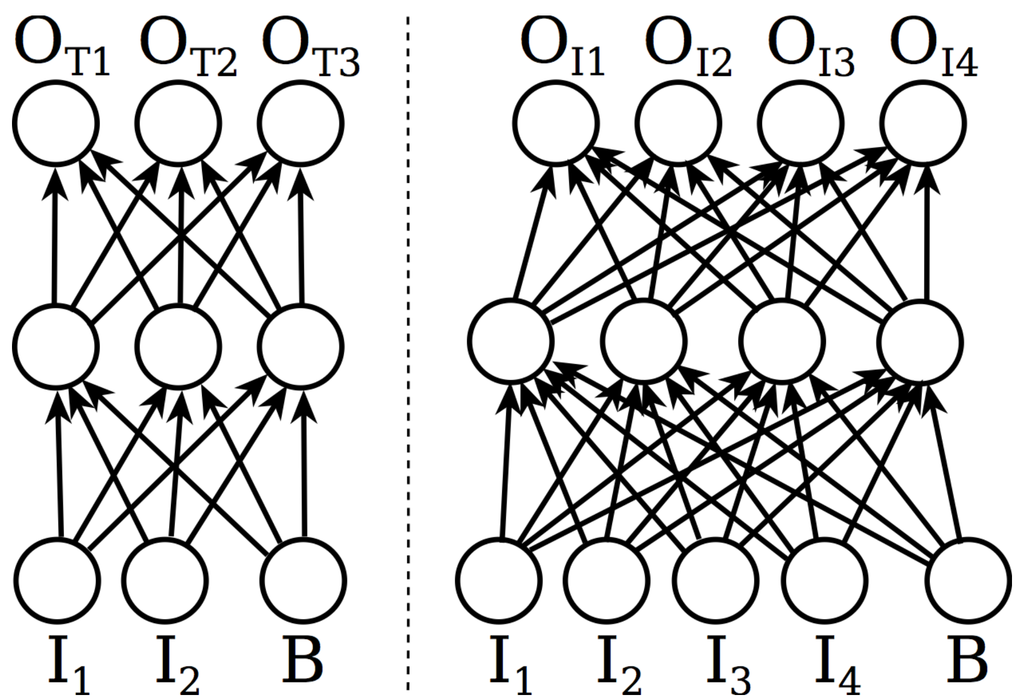
- Abstract domain
  - Simulation on NK landscapes
  - Experiments to characterize various effects
  - Evolving strategies in various environments against extreme opponents
- Concrete human game domain
  - Simulation of multi-player game
  - Modeling of human subjects
  - Evolving strategies against human subject models

Modeling Human Behavior

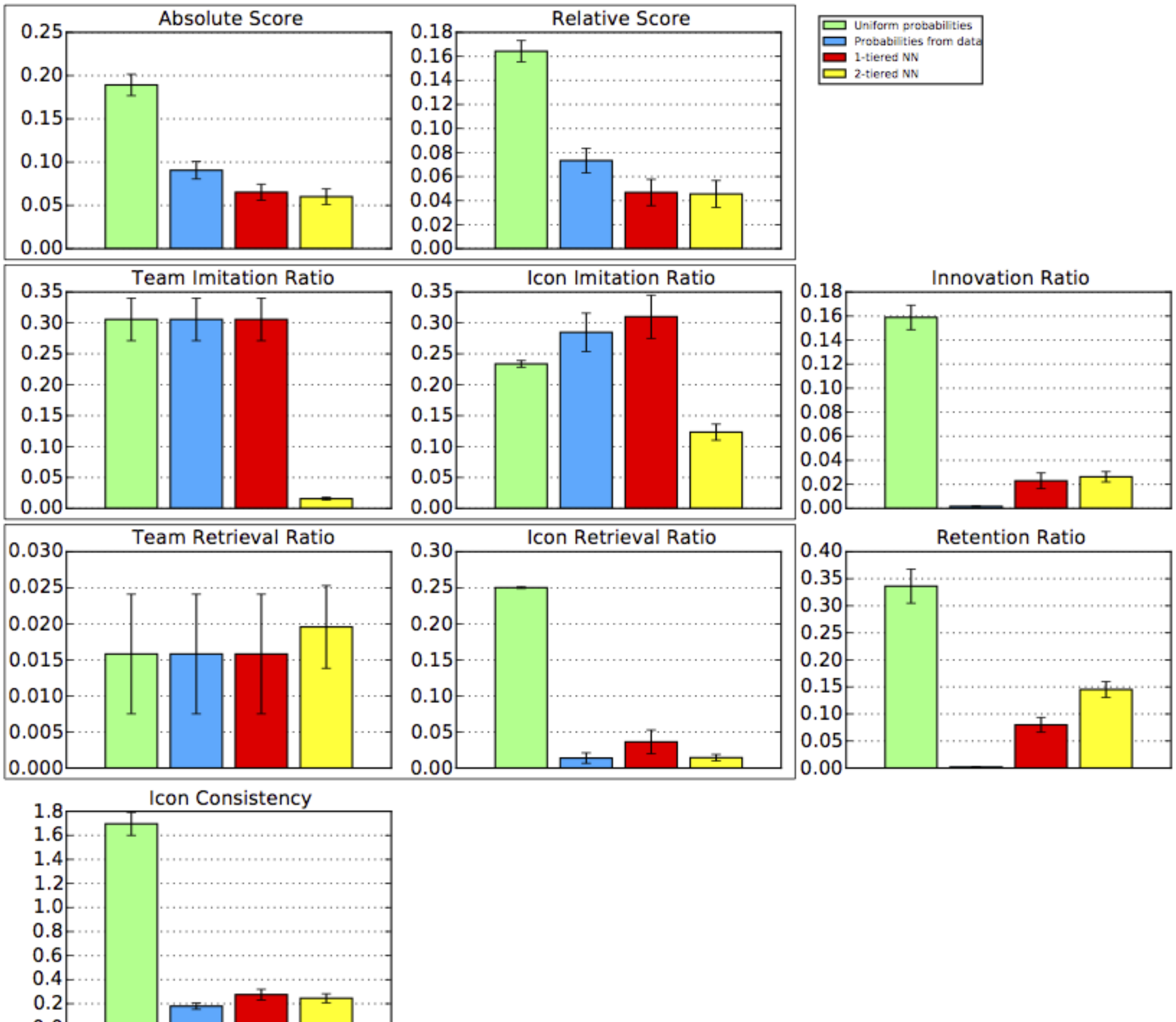
- Goal: Behave “like” the human subject(s)
  - Understand how people do CMAS
  - Create environments for optimization
- Choice of target to model (subset of dataset)
- Supervised learning via Backpropagation
- Distance objectives
  - Absolute and relative score
  - Team and icon action ratios
  - Icon consistency

Two-Tiered Neural Network Model

- Team-level actions (drag&drop whole team)
  - Inputs: current round, relative score
  - Outputs: team-imitate, team-retrieve, none
- Icon-level actions (drag&drop one icon)
  - Inputs: current round, relative score, icon age, icon popularity
  - Outputs: innovate, imitate, retrieve, retain



Modeling Results



2-tiered NN

- best in team & icon imitation
- worse than 1-tiered NN in retention
- tie with 1-t. NN in score & innovation, icon consistency

Fixed prob. model

- Best in innovation & retention

Team & icon retrieval are rare actions.

Outline

- Abstract domain
  - Simulation on NK landscapes
  - Experiments to characterize various effects
  - Evolving strategies in various environments against extreme opponents
- Concrete human game domain
  - Simulation of multi-player game
  - Modeling of human subjects
  - Evolving strategies against human subject models

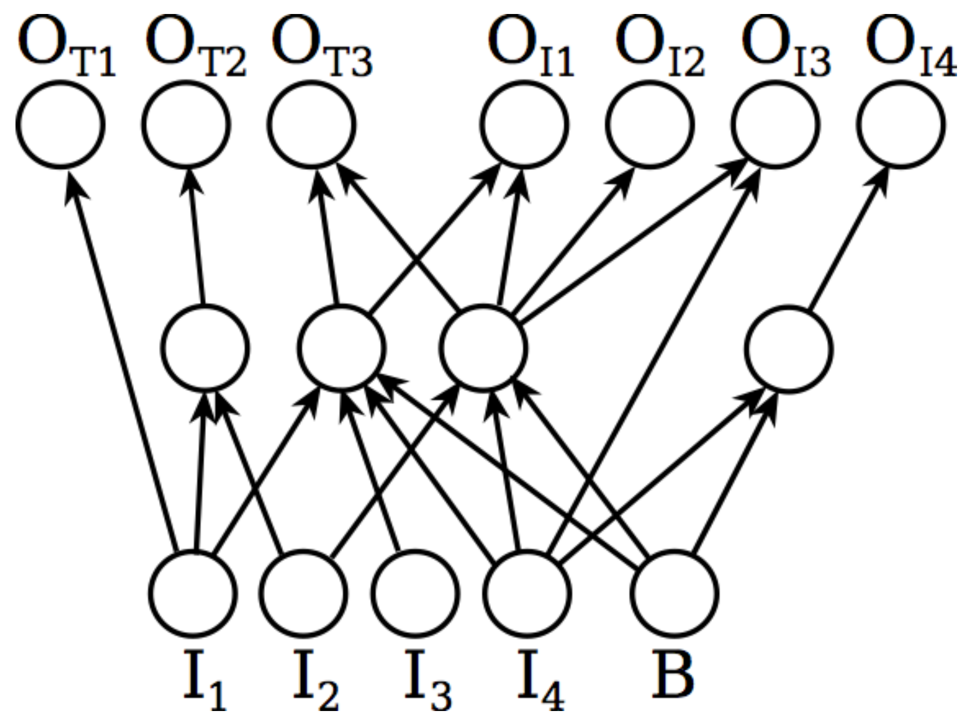
Evolution Experiments

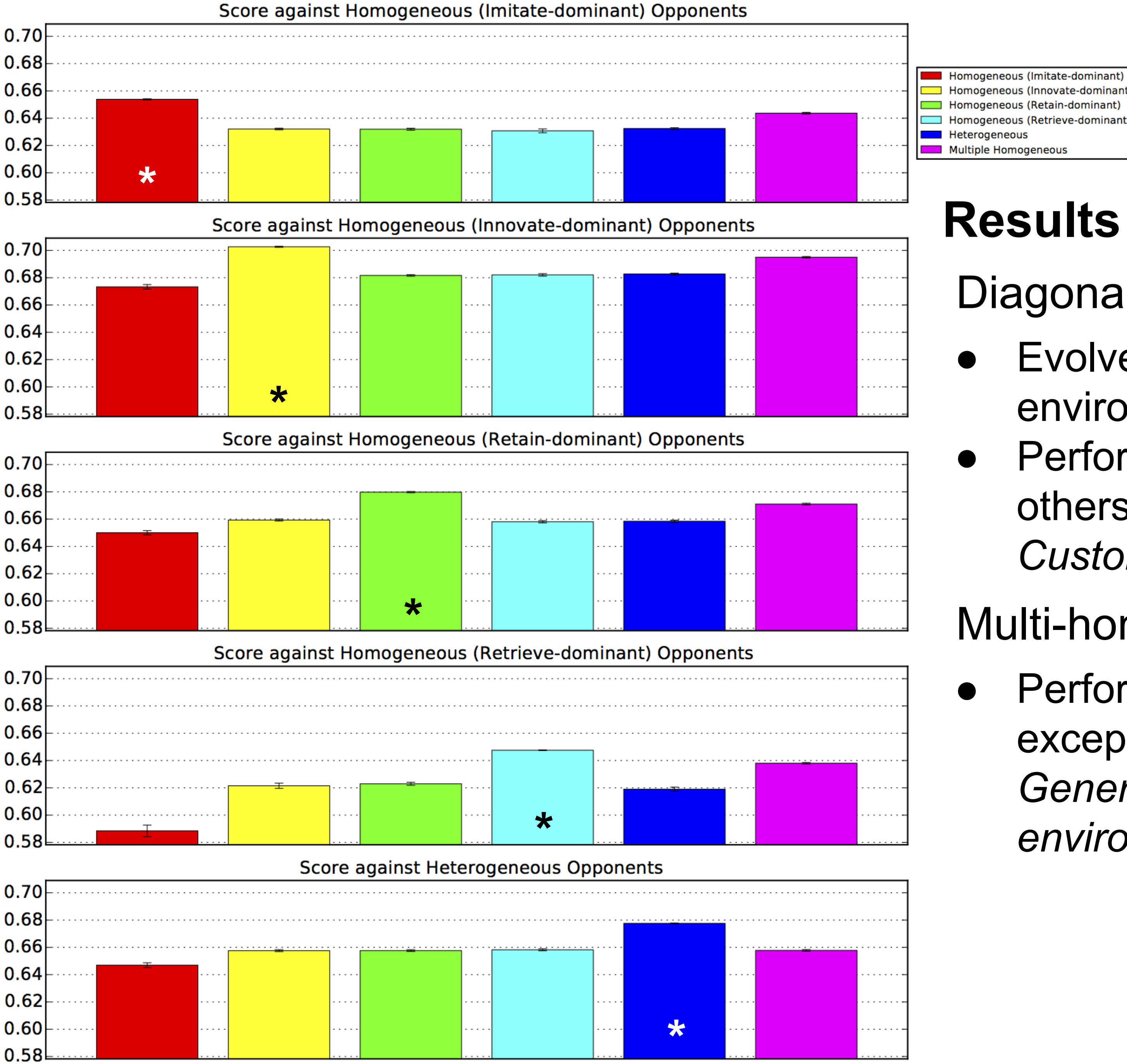
- Homogeneous environments
  Imitate-, innovate-, retain-, retrieve-dominant opponents
  - Customized strategies
- Heterogeneous environment
- Multiple homogeneous environments
  - General strategies
- Complex environments (human groups)
  - Group 1: 9-player env. x9
  - Group 2: 8-player env. x8
  - Group 3: 8-player env. x8

Evolution Experiments (contd.)

- Evolve combined team+icon networks with NEAT

- 8 or 9 agents
- 64 evolutionary runs
- 500 generations
- Population size: 100





Results

Diagonal

- Evolved in the same environment
- Performs better than all others:
  Customized for the env.

Multi-homogeneous

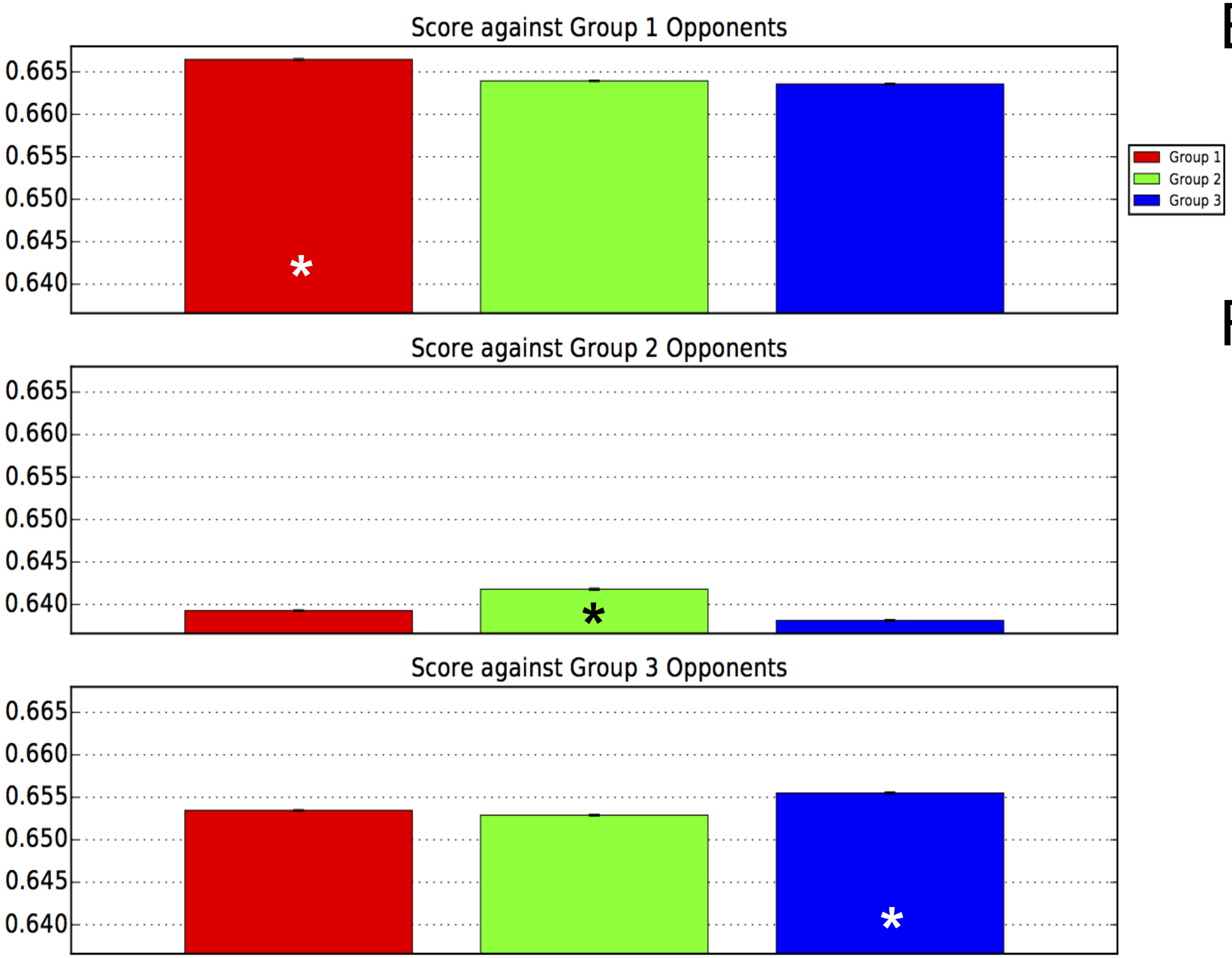
- Performs better than all except diag.:
  Generalized for multiple environments

Insights from Evolution

Strategies evolved in this environment, compared to those evolved in other environments:

- Imitation-dominant opponents
  - More innovation
  - Less imitation
- Innovation-dominant opponents
  - More team imitation
- Retrieve-dominant opponents
  - Less innovation
  - More icon imitation

Evolution Results with Complex Subject Model Groups



Environments

- G1: 9-player env. x9
- G2: 8-player env. x8
- G3: 8-player env. x8

Results

Diagonal

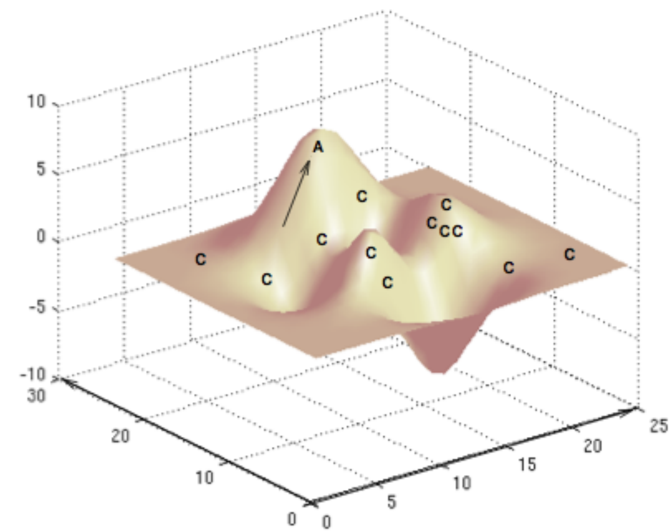
- Evolved in the same environment
- Performs better than others:
  Customized for the env.
- Subtle adaptations make a significant difference
- Score difference smaller due to higher diversity of environments

Evolved Strategies vs. Human Models

- Evolved strategies perform significantly better
- More
  - Imitation
- Less
  - Innovation
  - Retention

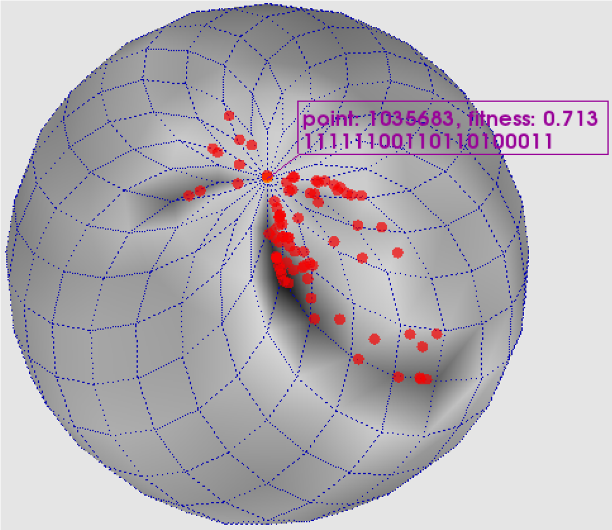
Future Work

- Strategy representation
  - Use another network output to choose which icon to copy
- Optimization method
  - Evolve multimodal strategies
  - Multi-objective optimization
- Human model applications
  - Replace human with model
- Evidence-based simulation
  - e.g. based on patent data from industry
- Theory



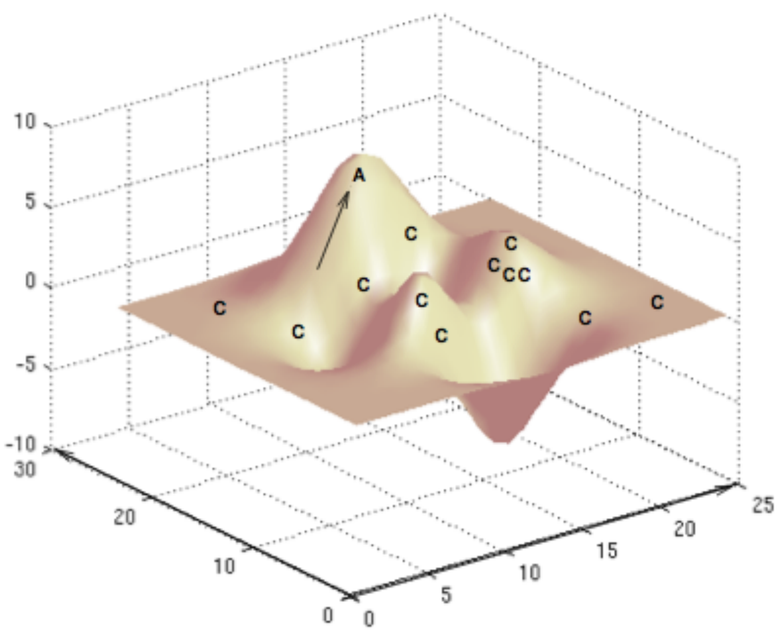
Contributions

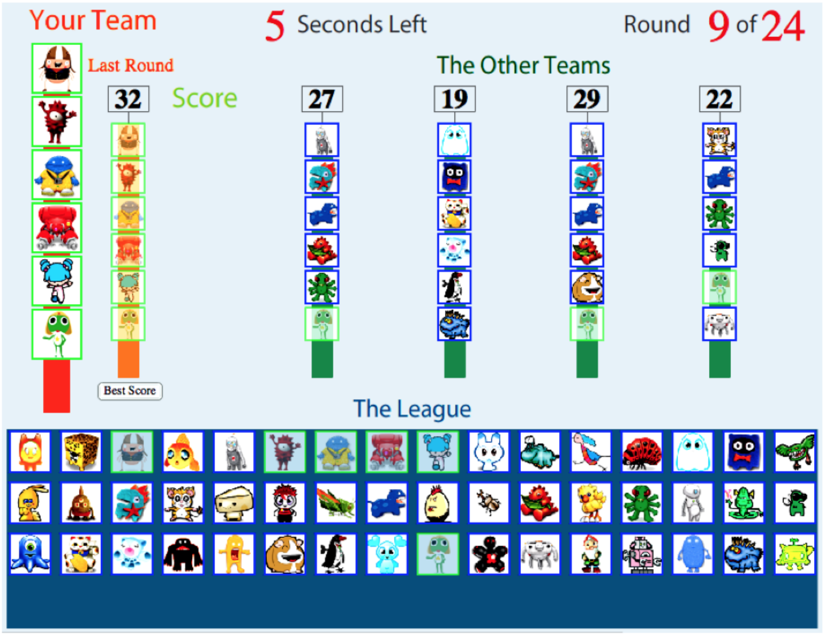
- Formalize & characterize experimentally in an abstract simulation
- Apply it to understand real-world search with concrete human domain
- Evolve
  - customized strategies for specific environments
  - general strategies
  - strategies better than humans
- Spherical visualization of NK landscapes



Conclusion

- Real world agents search in competitive multi-agent environments
- Useful way to study problem solving in the real world
- Potential to improve competitive search in other domains





Questions?