Lecture 23: Reinforcement Learning through Neuroevolution

Artificial Intelligence

Julian Togelius

Neuroevolution

- Training neural networks using evolutionary computation
- Representation: the weights of the neural network
- Fitness function: how well does the network perform its task?
- Works really well - when the network is small…

A tale from Ancient History

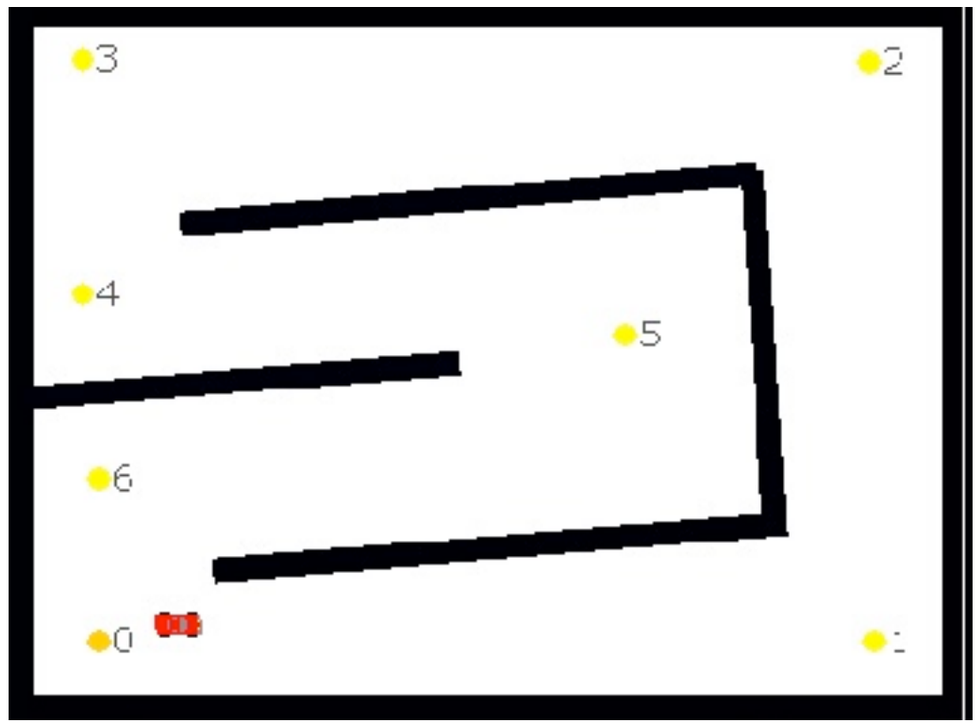
Togelius and Lucas (2006): Evolving Robust and Specialized Car Racing Skills. IEEE CEC.

The challenge: car racing

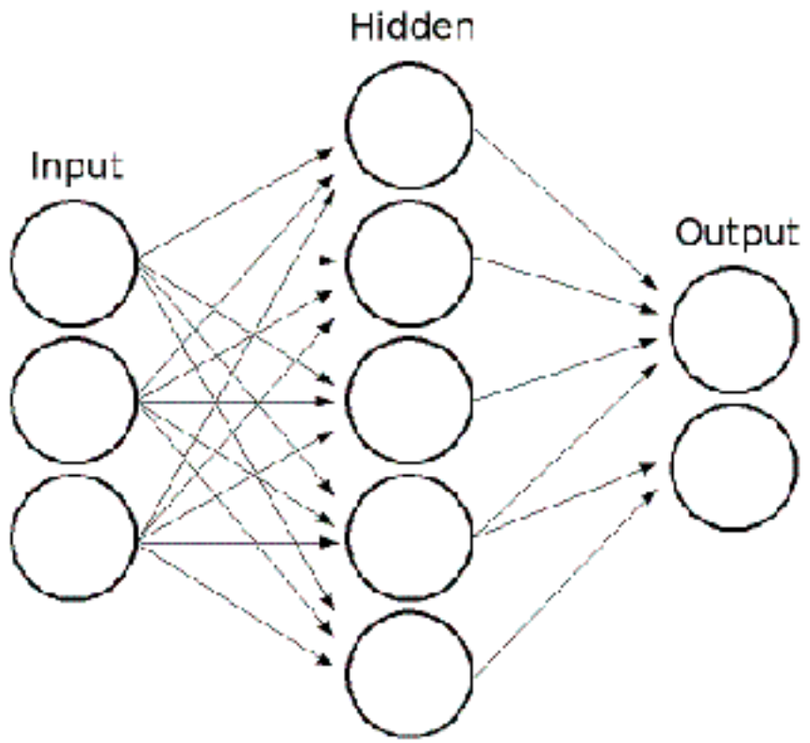
- Driving a car fast requires fine motor control (in both senses)
- Optimizing lap times requires planning
- Overtaking requires adversarial planning

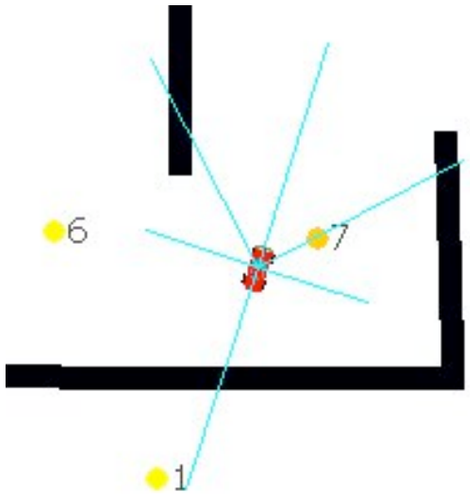
A simple car game

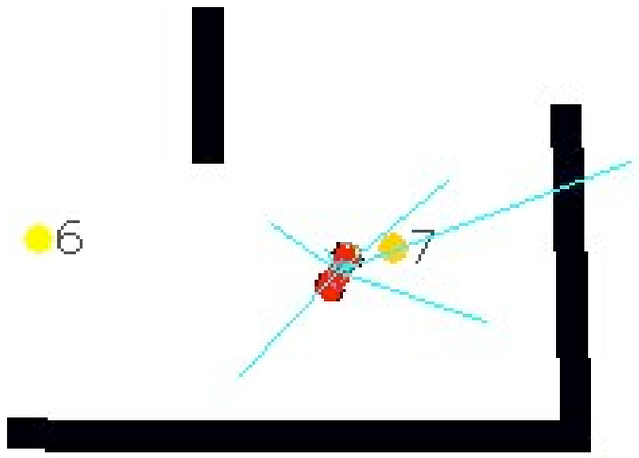
- Walls are solid
- Waypoints must be passed in order
- Fitness: continuous approximation of waypoints passed in 700 time steps

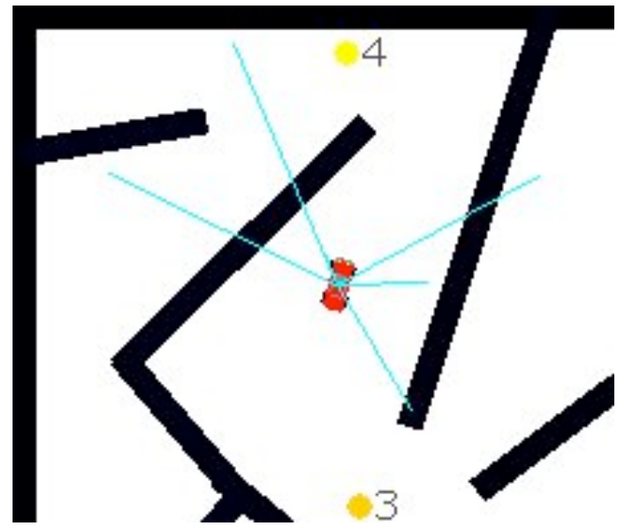


- Inputs
  - Six range-finder sensors (evolvable pos.)
  - Waypoint sensor, Speed, Bias
- Networks
  - Standard multi-layer perceptron, 9:6:2
  - Outputs interpreted as thrust/steering




6
7
1


6
7


4
3

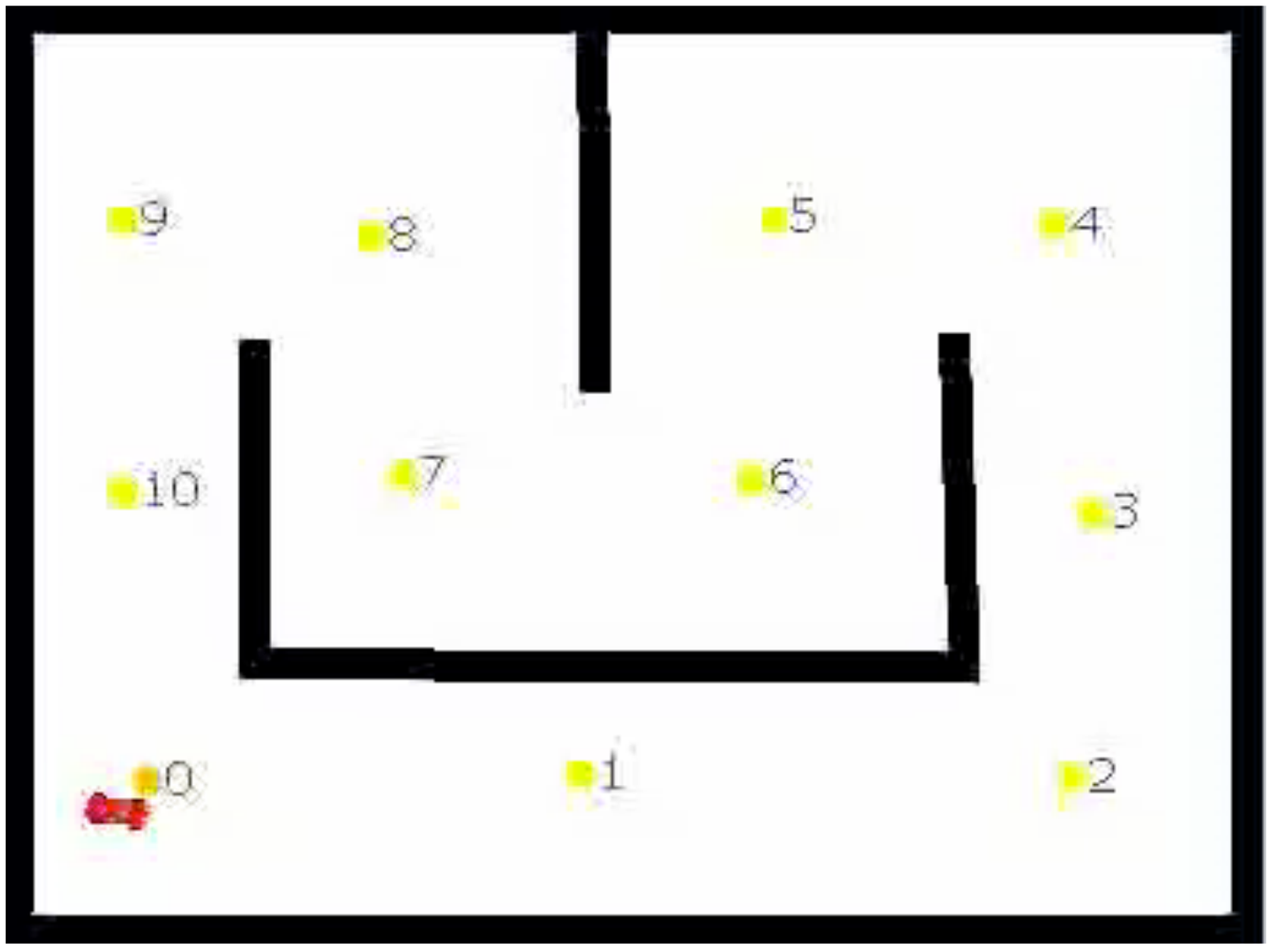
Algorithm 2: Evolution Strategy(μ,λ,n)

```
1  INITIALIZE (Population, μ + λ individuals)
2  for i=1 to n do
3      for j=1 to (μ + λ) do
4          EVALUATE (Population[j])
5      end
6      PERMUTE (Population)
7      SORTONFITNESS (Population)
8      for j=μ to (μ + λ) do
9          Population[j] ← COPY (Population[j-λ])
10         WEIGHTMUTATE (Population[j])
11     end
12 end
```

Mutation: add Gaussian noise with sd 1 to each connection

Fitness: progress around the track

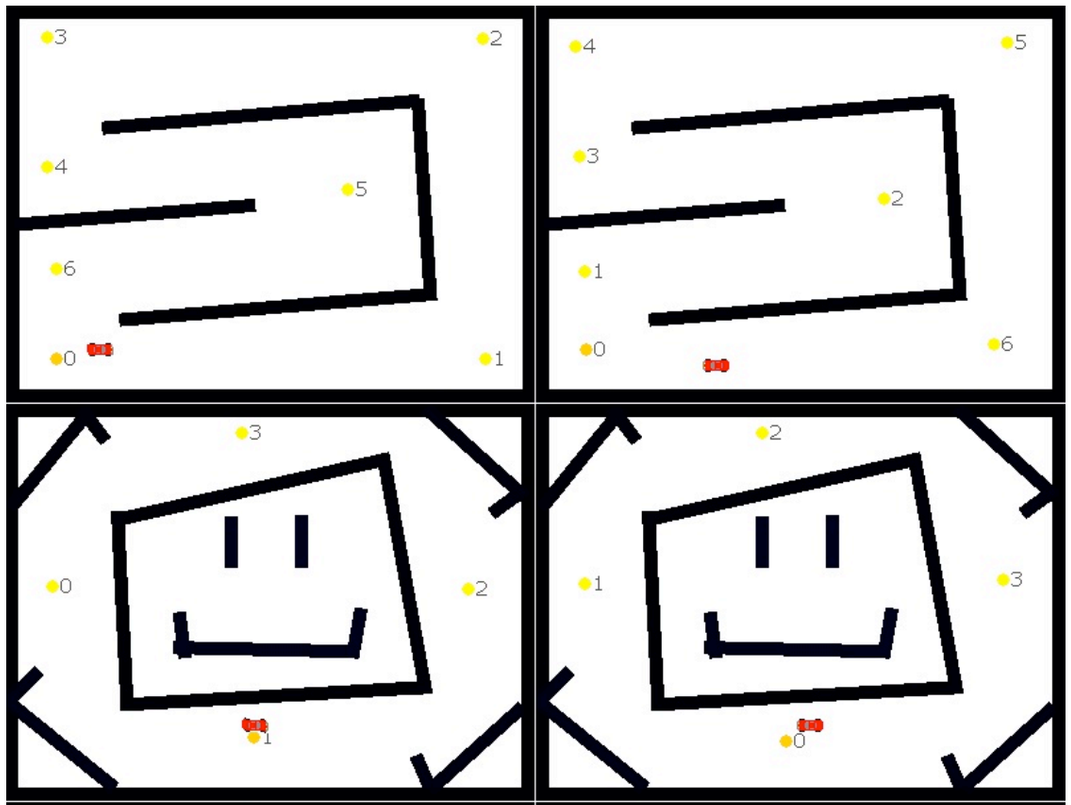
Example video

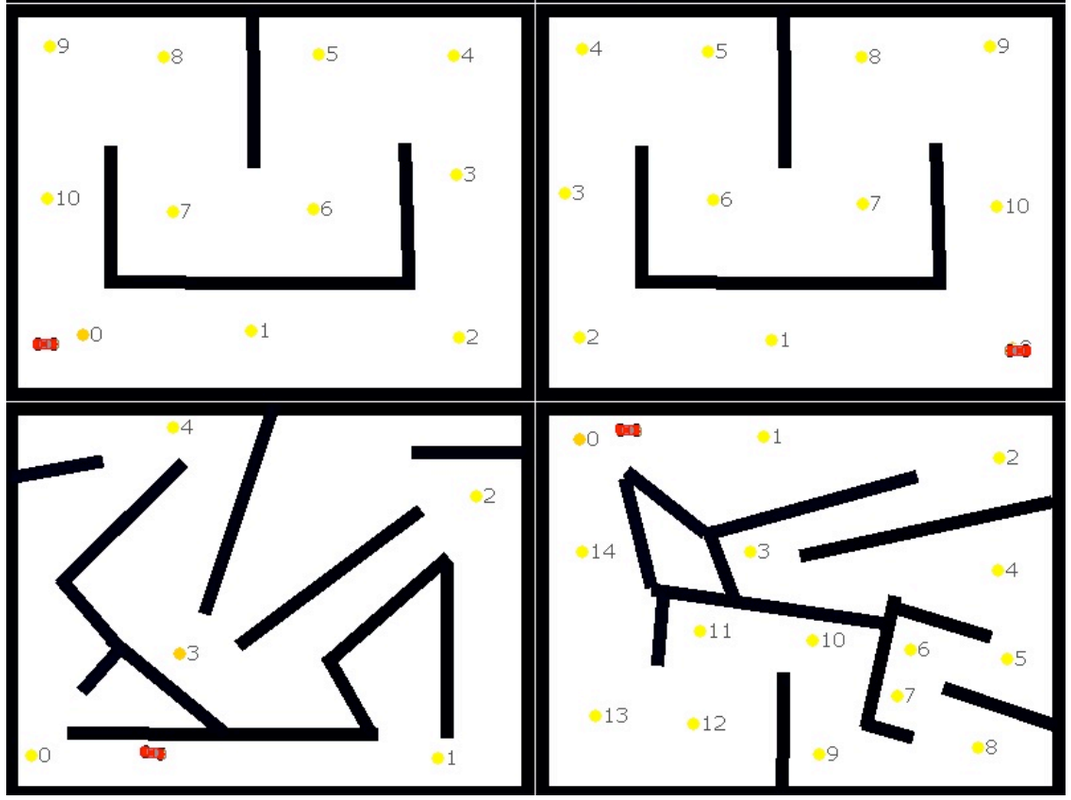


Evolved with 50+50 ES, 100 Generations

Generalization and specialization

- A controller evolved for one track does not necessarily perform well on other tracks
    - In fact, a controller evolved to drive a track clockwise will not be able to drive *the same track* counterclockwise
- How do we achieve more general game-playing skills?
    - Is there a tradeoff between generality and performance?

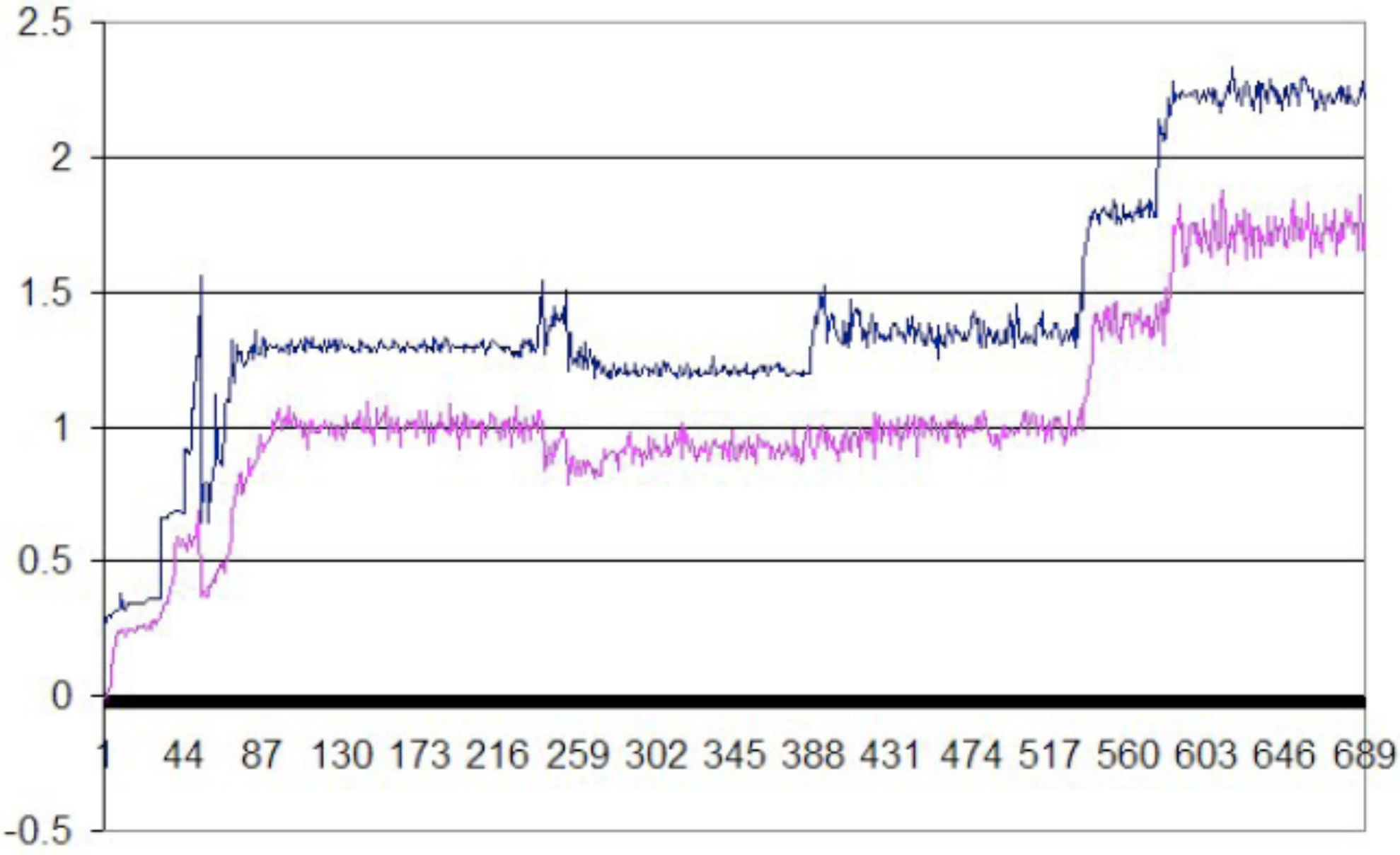

3
2
4
5
6
0
1
4
5
3
2
1
0
6
3
0
2
1
2
1
3
0


9
8
5
4
3
10
7
6
0
1
2
4
5
8
9
3
6
7
10
2
1
6
4
2
3
0
1
0
1
2
14
3
4
11
10
6
5
7
13
12
9
8

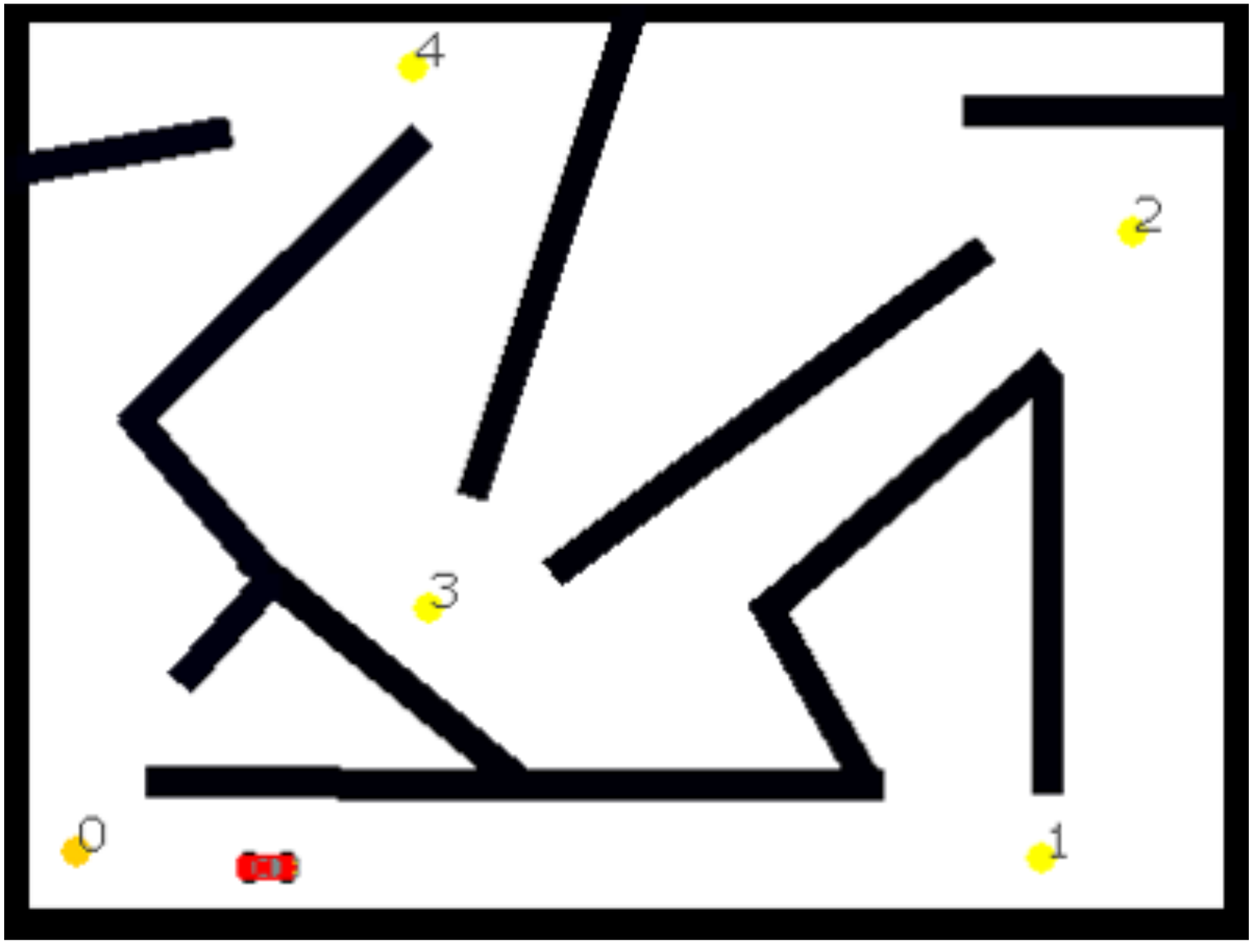
Incremental evolution

- Introduced by Gomez & Mikkulainen (1997)
  - Later also called curriculum learning
    - Change the fitness function *f* (to make it more demanding) as soon as a certain fitness is achieved
    - In this case, add new tracks to *f* as soon as the controller can drive 1.5 rounds on all tracks currently in *f*

Incremental evolution



Video: navigating a complex track



Observations

- Controllers evolved for specific tracks perform poorly on other tracks

- General controllers, that can drive almost any track, can be incrementally evolved

- Starting from a general controller, a controller can be further evolved for *specialization* on a particular track

  - drive faster than the general controller

  - works even when evolution from scratch did not work!

Two cars on a track

- Two car with solo-evolved controllers on one track: disaster
  - they don't even see each other!
- How do we train controllers that take other drivers into account? (avoiding collisions or using them to their advantage)
- Solution: car sensors (rangefinders, like the wall sensors) and *competitive coevolution*

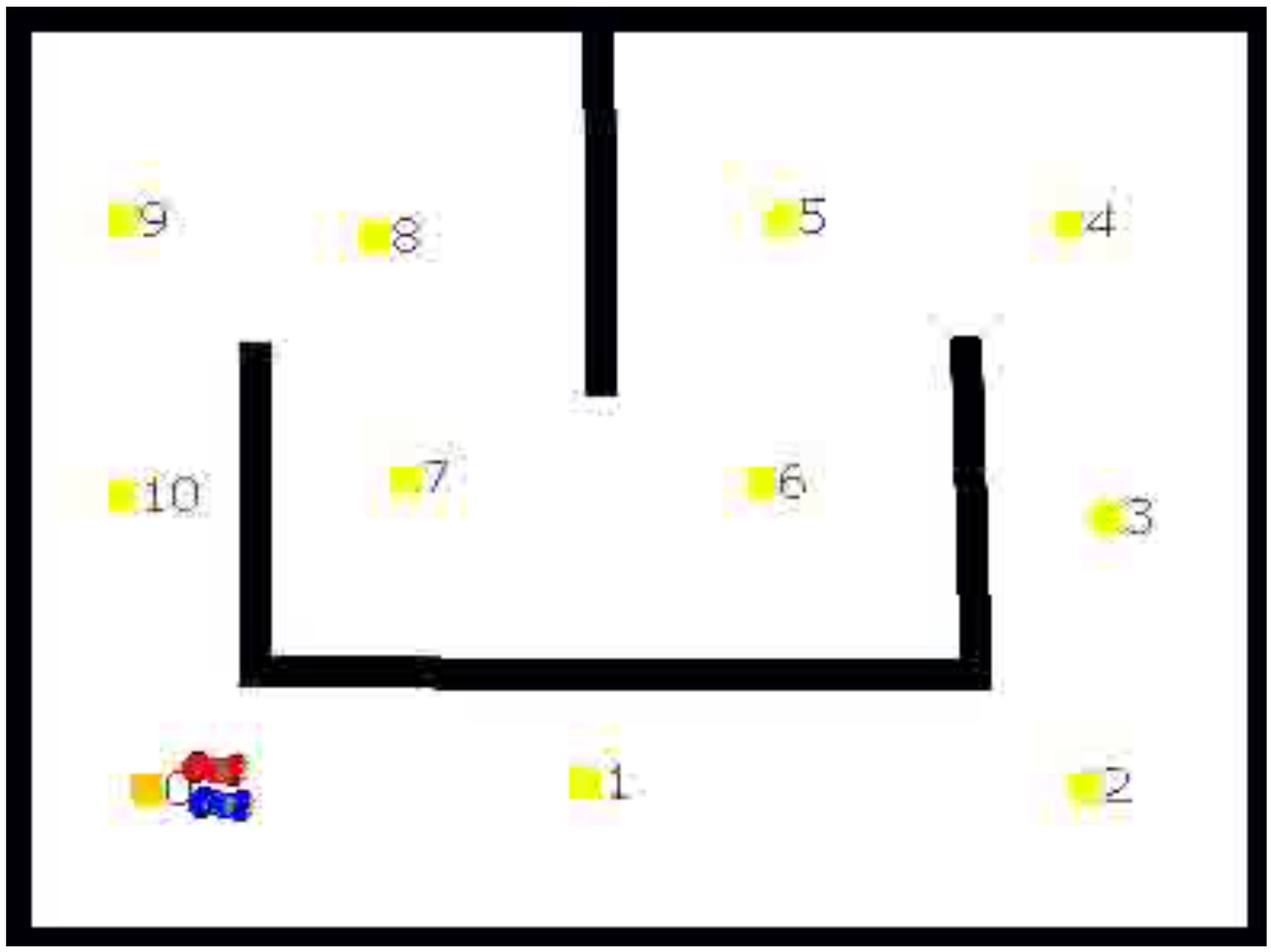
Competitive coevolution

- The fitness function evaluates at least two individuals
- One individual's success is *adversely* affected by the other's (directly or indirectly)
- Very potent, but seldom straightforward; e.g. Hillis (1991), Rosin and Belew (1996)

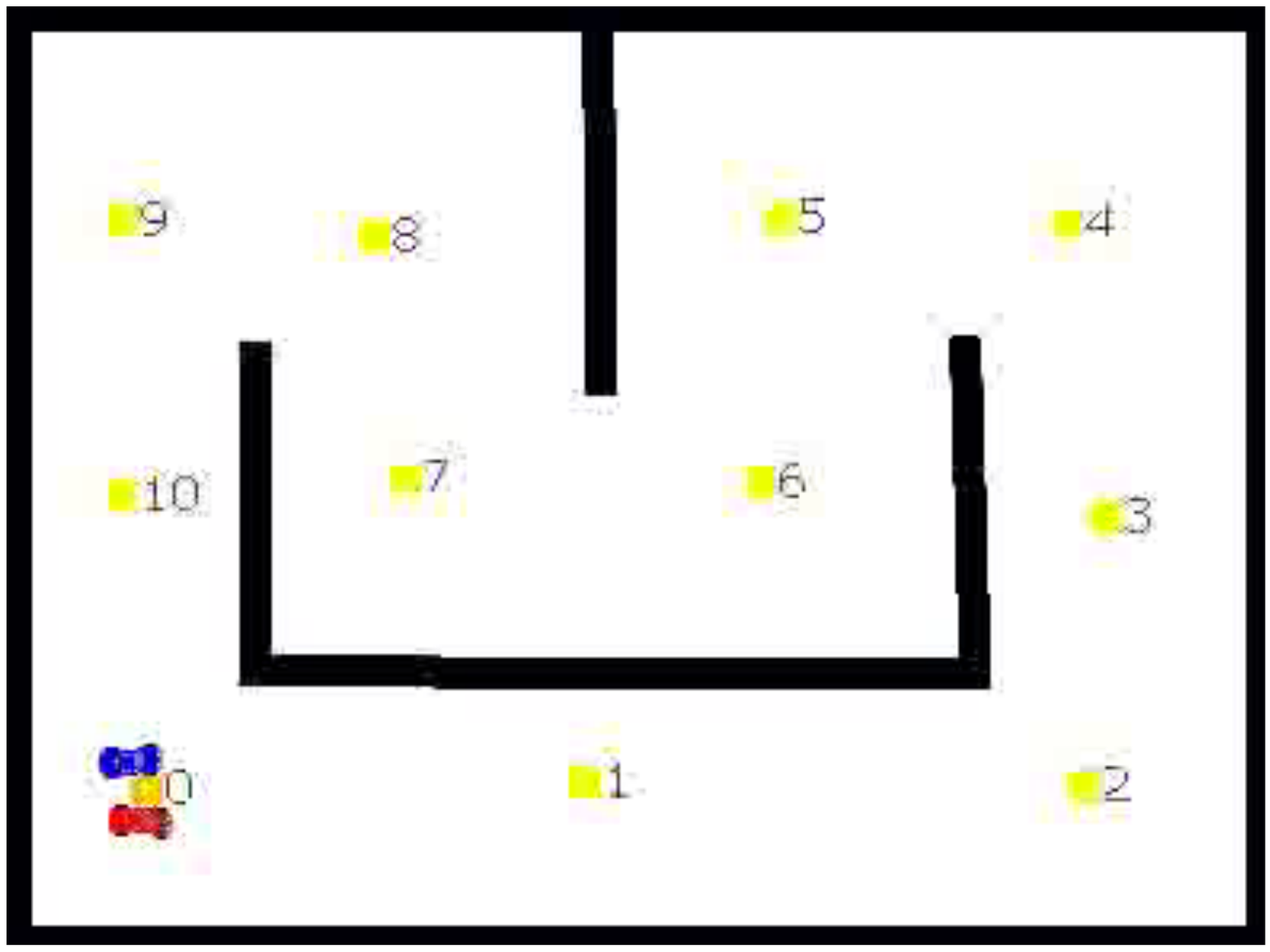
Competitive coevolution

- Standard 15+15 ES; each individual is evaluated through testing against the current best individual in the population
- Fitness function a mix of...
  - Absolute fitness: progress in *n* time steps
  - Relative fitness: distance ahead of or behind the other car after *n* time steps

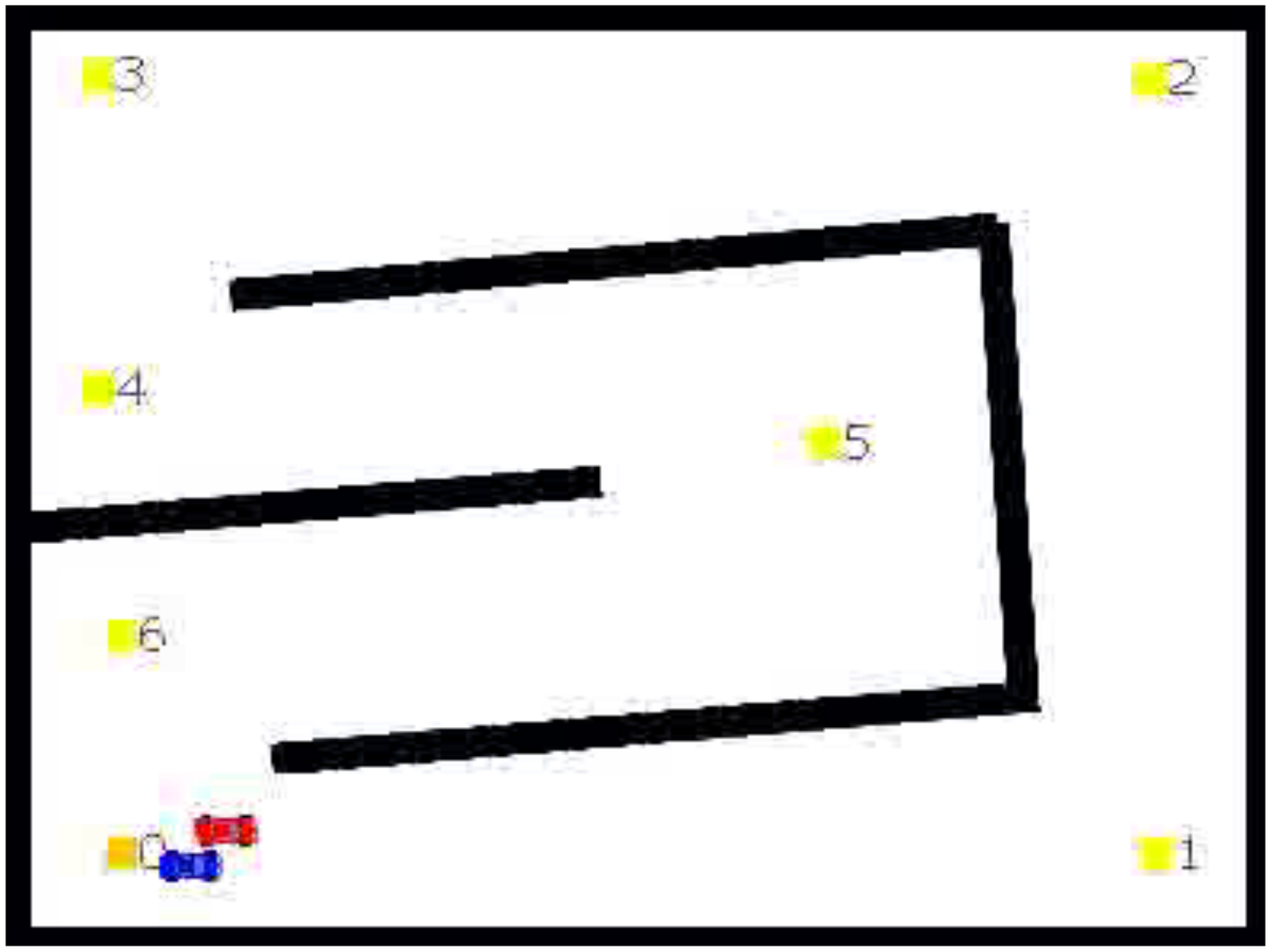
Video: absolute fitness



Video: 50/50 fitness



Video: relative fitness



Is neuroevolution a good way of doing RL?

- In theory, it should be less efficient
- In practice, it is often more robust
- Plus, you can do multiobjective evolution, quality-diversity illumination, etc.