COMP250 Artificial Intelligence

8: Evolutionary Algorithms

Research journal

Research wiki check-in

Research journal submission

- The deadline is rapidly approaching!
- **Everybody** must submit a copy of the wiki via LearningSpace
  - Clone the wiki using Git: `https://github.com/Falmouth-Games-Academy/comp250-wiki.wiki.git`
  - Make sure you are cloning the correct repo! It should have all of the wiki content in `.md` files
  - Zip your cloned repo and upload it

Optimisation

Optimisation

- Define a **fitness function** $f(x)$
- $f(x)$ **evaluates** a piece of content x, assigning it a **numerical score**
- **Higher** scores are **better**
- We are exploring a **fitness landscape**

Running example

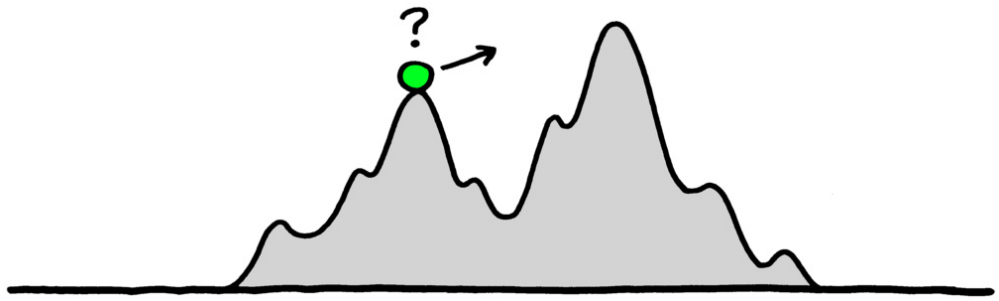
- `08_evolution` in `COMP250_live_coding` repository
- Want to generate a map where there is a path from start to goal, and that path is as long as possible
- Fitness measure:

$$f(x) = \begin{cases} \text{path length} & \text{if a path exists} \\ 0 & \text{otherwise} \end{cases}$$

Hillclimbing (a.k.a. gradient ascent)

- Start with an element x
- Create an element x' by making a **small change** to x
  - May choose the small change at random
  - Or may try every possible change
- If $f(x') > f(x)$, set $x = x'$
- Otherwise, throw x' away and keep x as it is
- Repeat

Local optima



- Hillclimbing tends to get stuck at a **local optimum**
- This may be much worse than the **global optimum**
- Have to let the solution get worse before it gets better — hillclimbing doesn't allow this

Escaping the local optimum

- Shotgun search (a.k.a. random restart)
  - Do several runs of hillclimbing from different starting positions
- Simulated annealing
  - Probability of allowing the search to keep a worse solution
  - This probability decreases as search progresses

Evolutionary algorithms

Evolutionary algorithms (EAs)

- Optimisation technique inspired by **biological evolution**
- We have a **population** of N solutions
- Generation 0: choose N solutions at random
- Generation $i + 1$: choose N new solutions based on the **fittest** individuals from generation i

Selecting the fittest

- **All** individuals should have a **chance** of being selected
- But **fitter** individuals should be selected **more often**
- Simple method: **tournament selection**
  - Randomly choose t individuals
  - Select the fittest out of those t

Mutation

- Select an individual
- Make a small change to it
- Add the changed individual to the new population

Crossover

- Select two individuals
- Combine them somehow (take “half” of one and “half” of the other)
- Add the resulting individual to the new population

Elitism

- Take the top x% of generation i, and pass it straight through to generation $i + 1$

Not just for PCG

- Common use for optimisation: **parameter tuning**
- Suppose we have several simple heuristic evaluation functions $h_1, h_2, \ldots, h_n$ which we want to combine into a single heuristic
- **Linear combination**:

$$w_1 h_1 + w_2 h_2 + \cdots + w_n h_n$$

  where $w_1, w_2, \ldots, w_n$ are constants: **weights**
- What value to choose for the weights? This is an optimisation problem!

MicroRTS

MicroRTS competition

- This should be your main focus (for COMP250) from now on
- Fork the repository at `https://github.com/falmouth-games-academy/comp250-bot` and follow the instructions in the YouTube video
- Look at the example bots in the `microrts` project
  - Start with the "rush" bots in `ai.abstraction`
  - Move on to search-based AI: `ai.minimax.RTMiniMax.RTMiniMax`, `ai.mcts.naivemcts.NaiveMCTS`, ...
  - Use these samples as a basis to create your own AI