COMP250 Artificial Intelligence

8: Procedural Content Generation

FALMOUTH
UNIVERSITY

What is PCG?

What is procedural content generation (PCG)?

- **Generation**: creating stuff

What is procedural content generation (PCG)?

- **Content**: levels, maps, art, animations, stories, items, quests, music, weapons, vehicles, characters, ...
- **Generation**: creating stuff

What is procedural content generation (PCG)?

- **Procedural**: by computer program or algorithm, with little or no direct input from designer or user
- **Content**: levels, maps, art, animations, stories, items, quests, music, weapons, vehicles, characters, ...
- **Generation**: creating stuff

Types of PCG

Types of PCG

- **Online**

Types of PCG

- **Online**
  - Generate content at run-time

Types of PCG

- **Online**
  - Generate content at run-time
  - Part of the game

Types of PCG

- **Online**
  - Generate content at run-time
  - Part of the game
- **Offline**

Types of PCG

- **Online**
  - Generate content at run-time
  - Part of the game
- **Offline**
  - Generate content at design-time

Types of PCG

- **Online**
  - Generate content at run-time
  - Part of the game
- **Offline**
  - Generate content at design-time
  - Tool for developers

PCG $\neq$ randomness

PCG $\neq$ randomness

- Many PCG systems use random numbers, but randomness in itself is not PCG

PCG $\neq$ randomness

- Many PCG systems use random numbers, but randomness in itself is not PCG
- Can have PCG without randomness, e.g. based on fractals or simulations

PCG $\neq$ randomness

- Many PCG systems use random numbers, but randomness in itself is not PCG
- Can have PCG without randomness, e.g. based on fractals or simulations
- Randomness in PCG is generally **constrained** to produce desired content

PCG ≠ randomness

- Many PCG systems use random numbers, but randomness in itself is not PCG
- Can have PCG without randomness, e.g. based on fractals or simulations
- Randomness in PCG is generally **constrained** to produce desired content
- Shuffling a deck of cards for a game of Solitaire is **not** PCG!

Not to be confused with...

Not to be confused with...

- **Procedural Rhetoric** (Bogost)

Not to be confused with...

- **Procedural Rhetoric** (Bogost)
- "the art of persuasion through rule-based representations and interactions, rather than the spoken word, writing, images, or moving pictures"

Not to be confused with...

- **Procedural Rhetoric** (Bogost)
- "the art of persuasion through rule-based representations and interactions, rather than the spoken word, writing, images, or moving pictures"
- There: "procedural" = "rule-based"

Not to be confused with...

- **Procedural Rhetoric** (Bogost)
- "the art of persuasion through rule-based representations and interactions, rather than the spoken word, writing, images, or moving pictures"
- There: "procedural" = "rule-based"
- Here: "procedural" = "algorithmic"

Why PCG?

Why PCG?

- More content for less development effort

Why PCG?

- More content for less development effort
- Decrease development costs

Why PCG?

- More content for less development effort
- Decrease development costs
- Increase replayability

Why PCG?

- More content for less development effort
- Decrease development costs
- Increase replayability
- Decrease storage requirements

Why PCG?

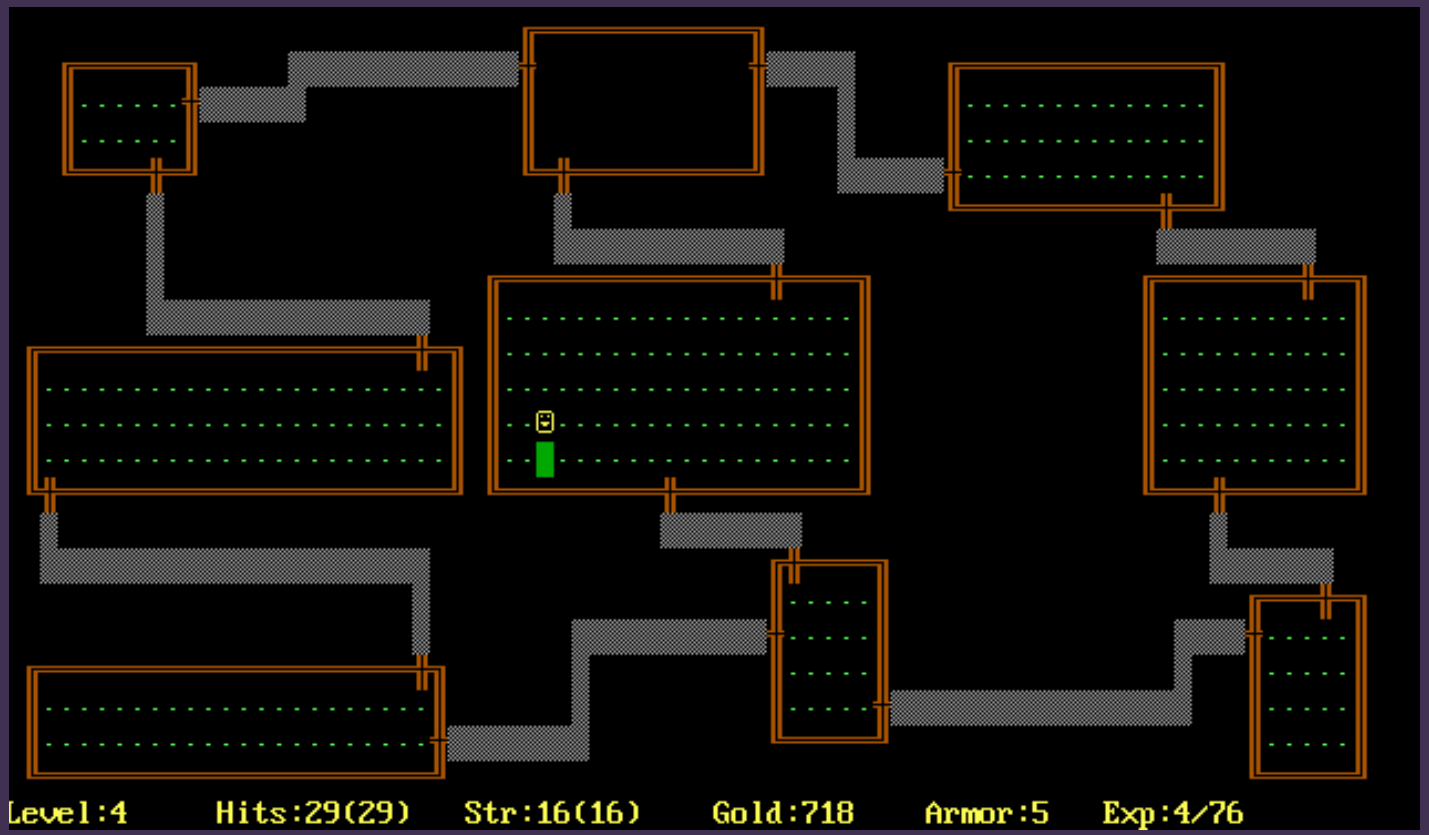
- More content for less development effort
- Decrease development costs
- Increase replayability
- Decrease storage requirements
- Allow game mechanics based on unseen content

FALMOUTH
UNIVERSITY

A brief history of PCG

Dungeons & Dragons (1974)

Rogue (1980)



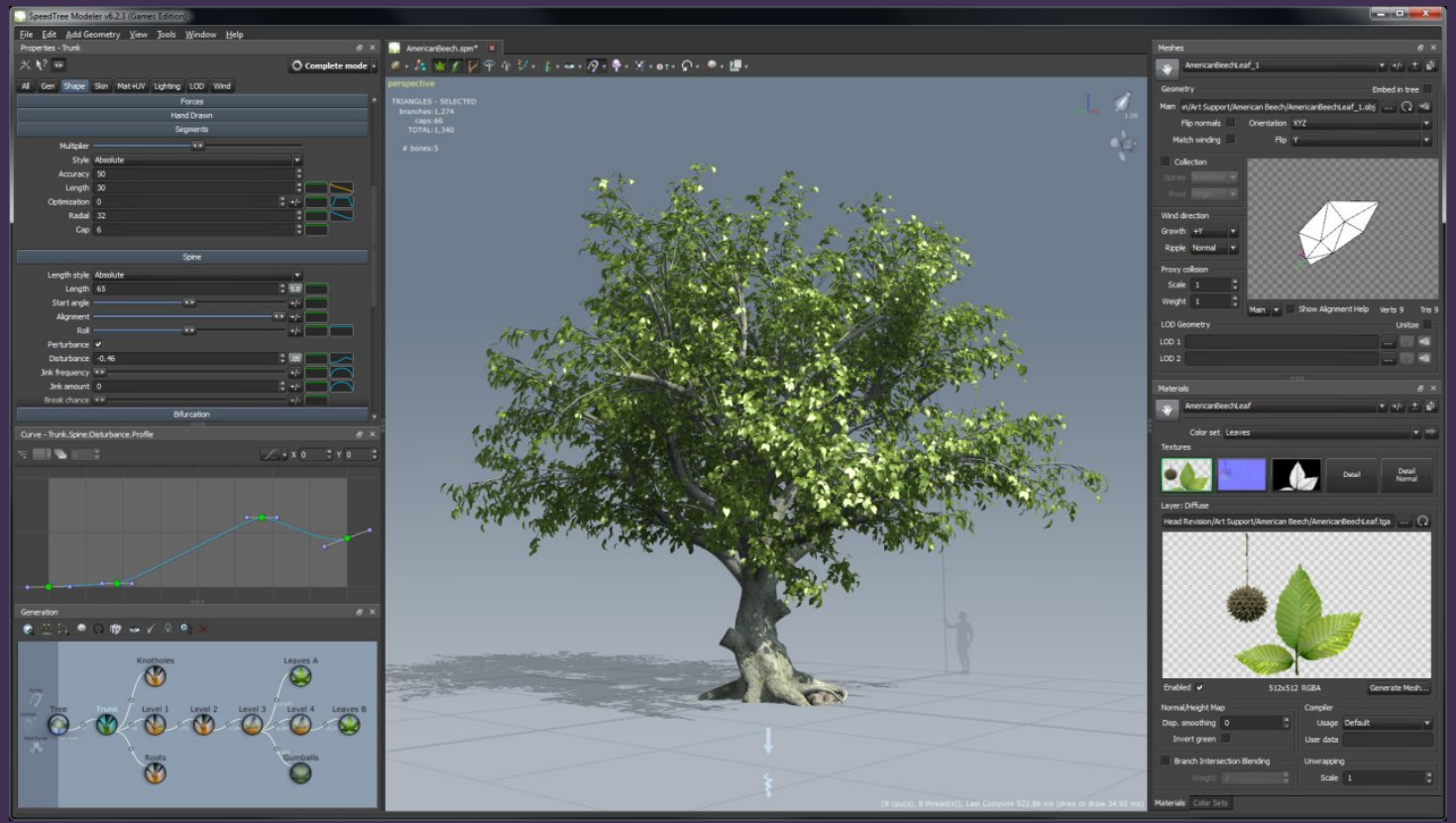
Elite (1984)

Sid Meier's Civilization (1991)

Frontier: Elite II (1993)

The Elder Scrolls II: Daggerfall (1996)

SpeedTree (2002)



.kkrieger (2004)

Dwarf Fortress (2006)

Spelunky (2008)

Spore (2008)

Left 4 Dead (2008)

Borderlands (2009)

Minecraft (2011)

The Binding of Isaac (2011)

Ultima Ratio Regum (2012)

To That Sect (2013)

Elite: Dangerous (2014)

No Man's Sky (2016)

FALMOUTH
UNIVERSITY

The role of PCG in games

Lessons from No Man's Sky

FALMOUTH
UNIVERSITY

Lessons from No Man's Sky

Lessons from No Man's Sky

- If you overscope, pray that you don't have to cut any features that you announced on stage at E3...

Lessons from No Man's Sky

- If you overscope, pray that you don't have to cut any features that you announced on stage at E3...
- PCG is not a substitute for gameplay

Lessons from No Man's Sky

- If you overscope, pray that you don't have to cut any features that you announced on stage at E3...
- PCG is not a substitute for gameplay
- PCG is not magic — it doesn't (by itself) let an indie-sized team produce a AAA game

Lessons from No Man's Sky

- If you overscope, pray that you don't have to cut any features that you announced on stage at E3...
- PCG is not a substitute for gameplay
- PCG is not magic — it doesn't (by itself) let an indie-sized team produce a AAA game
- When talking about scale and PCG, it's easy to set unrealistic expectations

Big numbers

Big numbers

- “Over 18 quintillion planets”

Big numbers

- "Over 18 quintillion planets"
- $2^{64} = 18\,446\,744\,073\,709\,551\,616$

Big numbers

- "Over 18 quintillion planets"
- $2^{64} = 18\,446\,744\,073\,709\,551\,616$
- What does this number even **mean**?

Big numbers

- "Over 18 quintillion planets"
- $2^{64} = 18\,446\,744\,073\,709\,551\,616$
- What does this number even **mean**?
- What it **really** means: "our random number generator uses a 64-bit seed"

Big numbers

- "Over 18 quintillion planets"
- $2^{64} = 18\,446\,744\,073\,709\,551\,616$
- What does this number even **mean**?
- What it **really** means: "our random number generator uses a 64-bit seed"
- They could have said "a near infinite number of planets"

Big numbers

- "Over 18 quintillion planets"
- $2^{64} = 18\,446\,744\,073\,709\,551\,616$
- What does this number even **mean**?
- What it **really** means: "our random number generator uses a 64-bit seed"
- They could have said "a near infinite number of planets"
- They could easily have made it "over 340 undecillion" planets ($2^{128} =$ 340 282 366 920 938 463 463 374 607 431 768 211 456)

Even bigger numbers

Even bigger numbers

- There are

$$52! = 80\,658\,175\,170\,943\,878\,571\,660\,636\,856\,403\,766$$
$$975\,289\,505\,440\,883\,277\,824\,000\,000\,000\,000$$

ways of shuffling a deck of playing cards

Even bigger numbers

- There are

$$52! = 80\,658\,175\,170\,943\,878\,571\,660\,636\,856\,403\,766$$
$$975\,289\,505\,440\,883\,277\,824\,000\,000\,000\,000$$

  ways of shuffling a deck of playing cards
- When you shuffle a deck, it is almost certain that **no deck of cards in human history** has ever existed in that order

Even bigger numbers

- There are

$$52! = 80\,658\,175\,170\,943\,878\,571\,660\,636\,856\,403\,766$$
$$975\,289\,505\,440\,883\,277\,824\,000\,000\,000\,000$$

  ways of shuffling a deck of playing cards
- When you shuffle a deck, it is almost certain that **no deck of cards in human history** has ever existed in that order
- But how **interesting** is that particular shuffled deck?

Even bigger numbers

- There are

$$52! = 80\,658\,175\,170\,943\,878\,571\,660\,636\,856\,403\,766$$
$$975\,289\,505\,440\,883\,277\,824\,000\,000\,000\,000$$

  ways of shuffling a deck of playing cards
- When you shuffle a deck, it is almost certain that **no deck of cards in human history** has ever existed in that order
- But how **interesting** is that particular shuffled deck?
- How **different** from another shuffled deck?

Uniqueness

"I can easily generate 10,000 bowls of plain oatmeal, with each oat being in a different position and different orientation, and *mathematically speaking* they will all be completely unique. But the user will likely just see *a lot of oatmeal*."
— Kate Compton

http://galaxykate0.tumblr.com/post/139774965871/so-you-want-to-build-a-generator

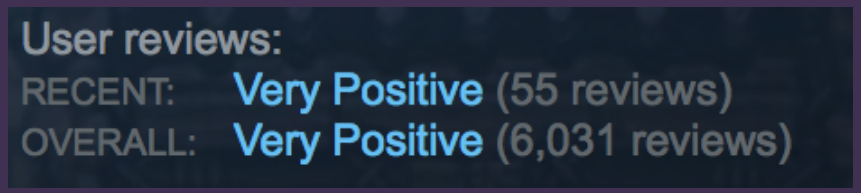
Uniqueness

" `Every Planet Unique' might mean that each planet has a complex sci-fi backstory rich enough to fill a two-part Star Trek episode. It might also mean that, mathematically speaking, there's a rock somewhere on the planet that doesn't look like any other rock in the universe."
— Michael Cook

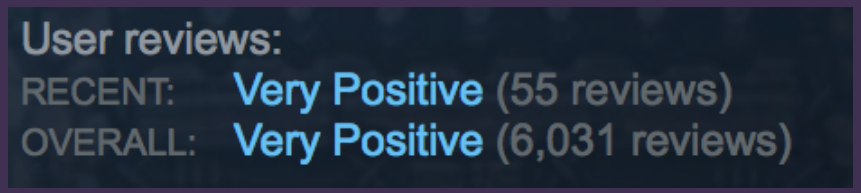
http://www.gamesbyangelina.org/2016/08/procedurallanguage/

Lessons from Spelunky

Lessons from Spelunky

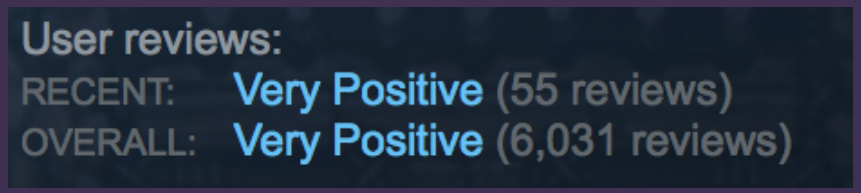


Lessons from Spelunky



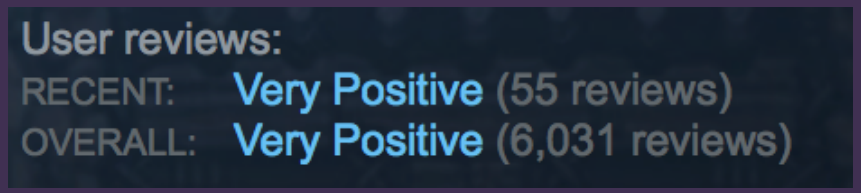
- PCG can complement solid game mechanics

Lessons from Spelunky



- PCG can complement solid game mechanics
- PCG can **enable** new (discovery-based) game mechanics

Lessons from Spelunky



- PCG can complement solid game mechanics
- PCG can **enable** new (discovery-based) game mechanics
- No need to dazzle the audience with big numbers

Curation

Curation

Curation

- Human creators constantly ask themselves: **is this any good?**

Curation

- Human creators constantly ask themselves: **is this any good?**
- Smart PCG should not **merely generate**: it should also **evaluate**

Authorship

Authorship

- In a game with **emergent narrative**, who is the author? Is it the developer, the player, or both?

Authorship

- In a game with **emergent narrative**, who is the author? Is it the developer, the player, or both?
- In a game with **procedurally-generated content**, who (or what) is the author? Is it the developer, the player, the system, or all three?

Authorship

"(We) create the systems (including some fixed content), and the choices made at that stage are influenced by our preferences, worldviews, talents and flaws, and then the system creates the content. The players are exposed to the content and can manipulate it using the tools we (and others) create for them. How they use the tools is up to them, and how the content reacts is up to our systems."
— Tarn Adams

http://www.nullpointer.co.uk/content/interview-dwarf-fortress/

FALMOUTH
UNIVERSITY

The future of PCG

"You are playing an "open world" game, something like Grand Theft Auto or Skyrim. Instead of going straight to the next mission objective in the city you are in, you decide to drive (or ride) five hours in some randomly chosen direction. The game makes up the landscape as you go along, and you end up in a new city that no human player has visited before. In this city, you can enter any house (though you might have to pick a few locks), talk to everyone you meet, and involve yourself in a completely new set of intrigues and carry out new missions. If you would have gone in a different direction, you would have reached a different city with different architecture, different people and different missions. Or a huge forest with realistic animals and eremites, or a secret research lab, or whatever the game engine comes up with."
— Julian Togelius

http://togelius.blogspot.co.uk/2015/10/what-if-videogames-had-actual-ai.html

Whole game generation

Whole game generation

Whole game generation

- E.g. ANGELINA (Michael Cook)

Whole game generation

- E.g. ANGELINA (Michael Cook)
- Generate **entire games** from scratch, possibly using ideas or themes provided by the user

Whole game generation

- E.g. ANGELINA (Michael Cook)
- Generate **entire games** from scratch, possibly using ideas or themes provided by the user
- **Democratise** game design — create games in **collaboration** with a non-skilled user

Whole game generation

- E.g. ANGELINA (Michael Cook)
- Generate **entire games** from scratch, possibly using ideas or themes provided by the user
- **Democratise** game design — create games in **collaboration** with a non-skilled user
  - (i.e. make it so that you don't need to do a degree to learn how to make games...)

Computational creativity

Computational creativity

Computational creativity

Computational creativity

- Open question: can an AI system be **creative**?

Computational creativity

- Open question: can an AI system be **creative**?
- Beyond **mere generation**

Computational creativity

- Open question: can an AI system be **creative**?
- Beyond **mere generation**
- Beyond generating **content** to generating **ideas**

FALMOUTH
UNIVERSITY

Optimisation

Optimisation

Optimisation

- Define a **fitness function** $f(x)$

Optimisation

- Define a **fitness function** $f(x)$
- $f(x)$ **evaluates** a piece of content x, assigning it a **numerical score**

Optimisation

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

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

Running example

- `08_pathfinding` in `bsc_live_coding` repository

Running example

- `08_pathfinding` in `bsc_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

Running example

- `08_pathfinding` in `bsc_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)

Hillclimbing (a.k.a. gradient ascent)

- Start with an element x

Hillclimbing (a.k.a. gradient ascent)

- Start with an element x
- Create an element x' by making a **small change** to x

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

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

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 $x = x'$

Hillclimbing (a.k.a. gradient ascent)

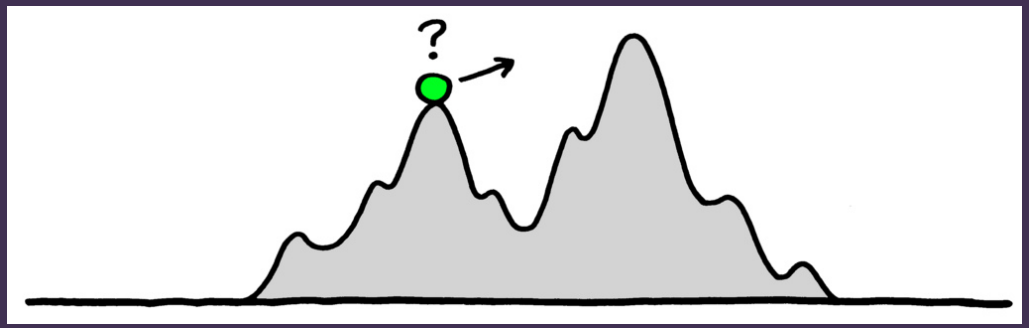
- Start with an element x
- Create an element x' by making a **small change** to x
  - May choose the small change at random
  - Or may try every possible change
- If $f(x') > f(x)$, set $x = x'$
- Otherwise, throw x' away and keep x as it is

Hillclimbing (a.k.a. gradient ascent)

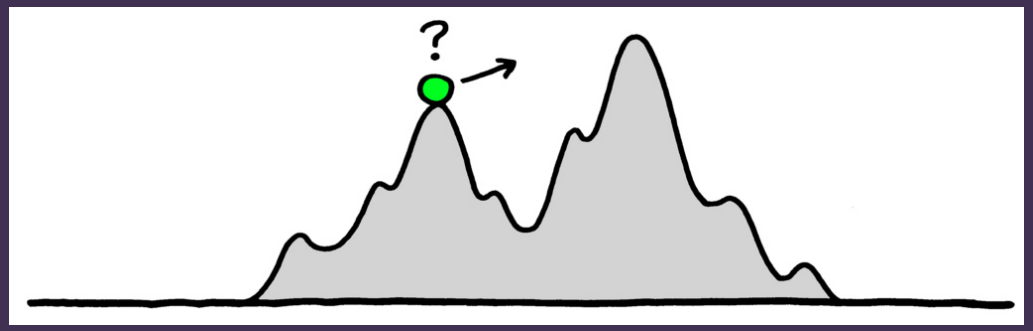
- Start with an element x
- Create an element x' by making a **small change** to x
  - May choose the small change at random
  - Or may try every possible change
- If $f(x') > f(x)$, set $x = x'$
- Otherwise, throw x' away and keep x as it is
- Repeat

Local optima

Local optima

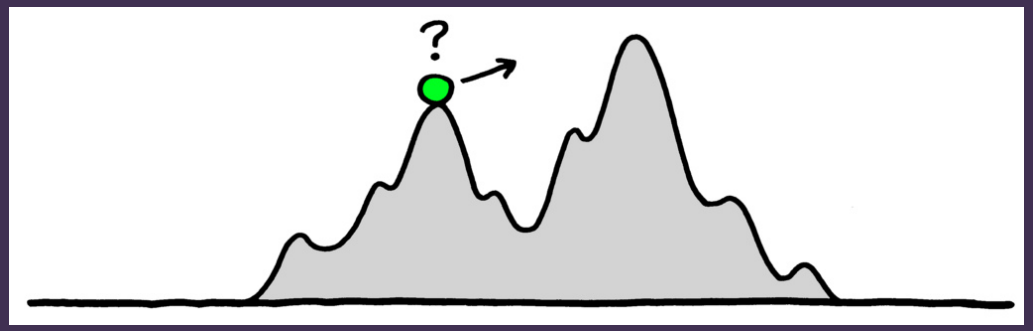


Local optima



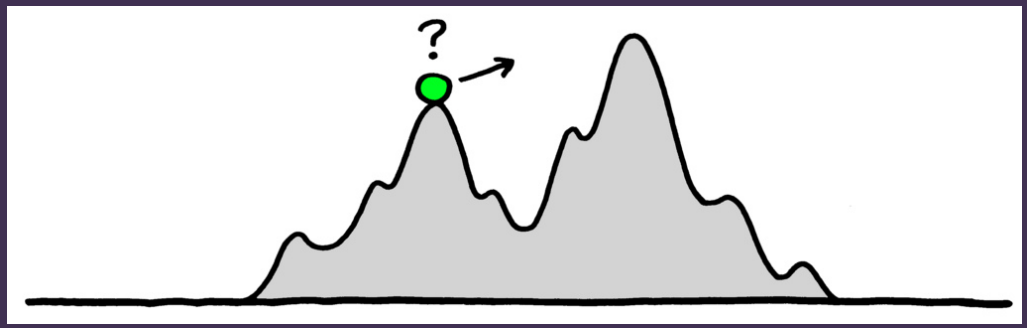
- Hillclimbing tends to get stuck at a **local optimum**

Local optima



- Hillclimbing tends to get stuck at a **local optimum**
- This may be much worse than the **global optimum**

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

Escaping the local optimum

- Shotgun search (a.k.a. random restart)

Escaping the local optimum

- Shotgun search (a.k.a. random restart)
  - Do several runs of hillclimbing from different starting positions

Escaping the local optimum

- Shotgun search (a.k.a. random restart)
  - Do several runs of hillclimbing from different starting positions
- Simulated annealing

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

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

FALMOUTH
UNIVERSITY

Evolutionary algorithms

Evolutionary algorithms (EAs)

Evolutionary algorithms (EAs)

- Optimisation technique inspired by **biological evolution**

Evolutionary algorithms (EAs)

- Optimisation technique inspired by **biological evolution**
- We have a **population** of N solutions

Evolutionary algorithms (EAs)

- Optimisation technique inspired by **biological evolution**
- We have a **population** of N solutions
- Generation 0: choose N solutions at random

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

Selecting the fittest

- **All** individuals should have a **chance** of being selected

Selecting the fittest

- **All** individuals should have a **chance** of being selected
- But **fitter** individuals should be selected **more often**

Selecting the fittest

- **All** individuals should have a **chance** of being selected
- But **fitter** individuals should be selected **more often**
- Simple method: **tournament selection**

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

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

Mutation

- Select an individual

Mutation

- Select an individual
- Make a small change to it

Mutation

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

Crossover

Crossover

- Select two individuals

Crossover

- Select two individuals
- Combine them somehow (take “half” of one and “half” of the other)

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

Elitism

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