Lecture 12 – Revision and Common Themes

Optimisation CT5141

James McDermott

University of Galway

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Overview

1. **Some things we didn't study**
2. A big picture of optimisation algorithms
3. Common themes of the module

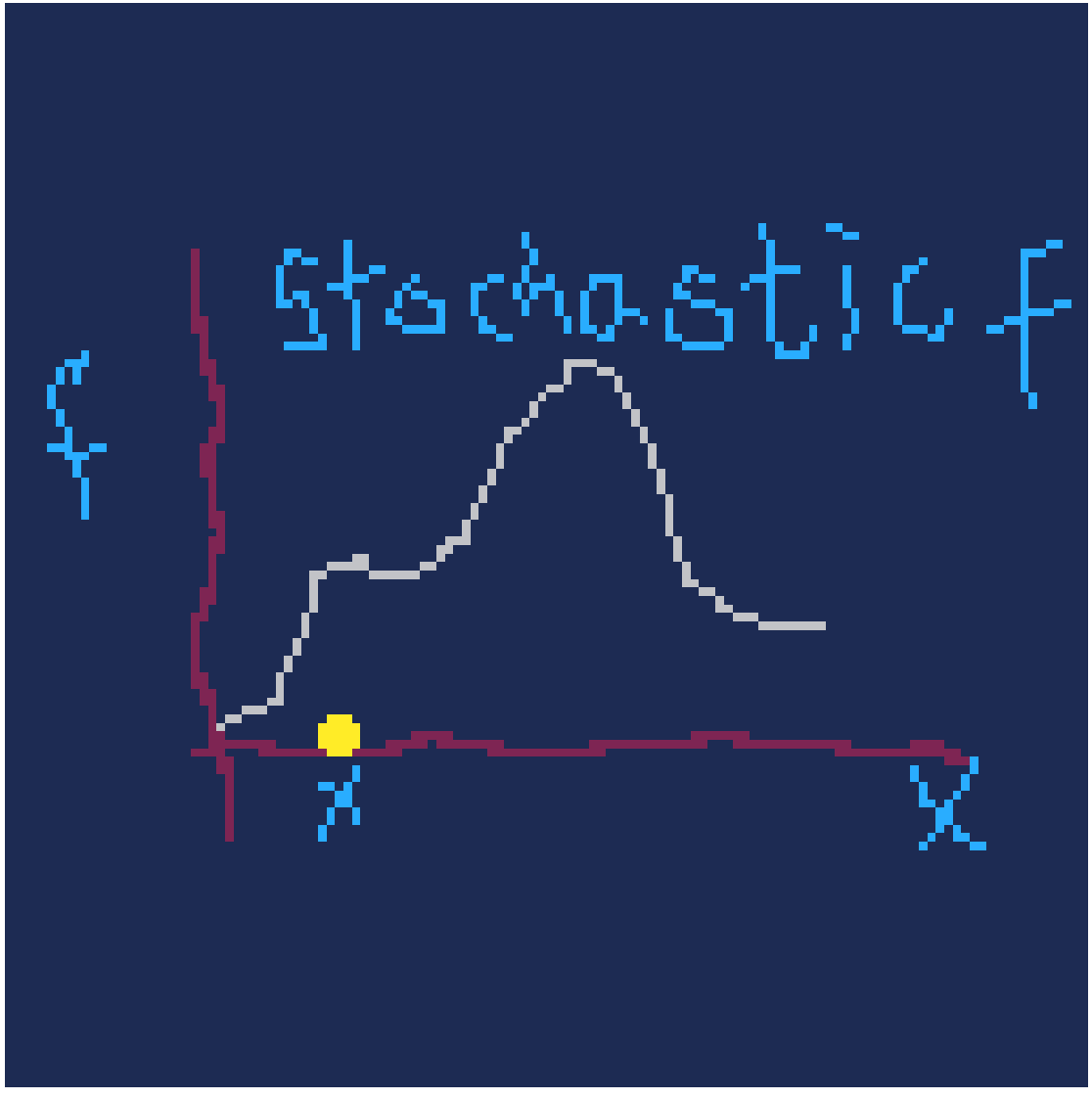
Some things we didn't study

- Stochastic optimisation
- Searching for diversity
- No free lunch
- Before and after optimisation
- Gaming the system

Stochastic Optimisation

Sometimes our objective is **stochastic**. That means that $f(x)$ is not the same every time for fixed x, but is in effect sampled from some (perhaps unknown) distribution which depends on x.

Stochastic Optimisation



Made with https://www.lexaloffle.com/bbs/?tid=40058

Trade-off

Suppose we have enough CPU time for 1,000,000 calls to our stochastic f. We could do:

- Run (e.g.) 1,000,000 steps of hill-climbing using f, OR
- Define $F(x)$ = mean over 1000 samples of $f(x)$, and run only 1000 steps of hill-climbing using F

Trade-off: more samples gives more accurate estimate of the true quality of each x, but fewer steps of hill-climbing.

Best trade-off **may depend on variance, distribution, etc.**

A real-world application

- Rogers, Carroll and McDermott, 2019
- Creating an electricity price structure (i.e. how price changes at weekends, night, etc.)
- Using a GA
- Fitness is the outcome of a large simulation of the choices of many customers in response to the price structure
- Experiments showed the number of runs of the simulation per call of f should be around 80 – not too large, not too small.

Searching for diversity

Sometimes we don't just need to optimise an objective, but rather want to **explore all the possibilities**. This relates to multi-objective optimisation but is not quite the same. Some of the algorithms people use:

- NSGA2
- Novelty Search
- MAP-Elites

See e.g. https://www.frontiersin.org/articles/10.3389/frobt.2016.00040/full

No Free Lunch

Could there be **one algorithm to rule them all**?

No Free Lunch

IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 1, NO. 1, APRIL 1997

No Free Lunch Theorems for Optimization

David H. Wolpert and William G. Macready

***Abstract*—A framework is developed to explore the connection between effective optimization algorithms and the problems they are solving. A number of "no free lunch" (NFL) theorems are presented which establish that for any algorithm, any elevated performance over one class of problems is offset by performance over another class. These theorems result in a geometric interpretation of what it means for an algorithm to be well suited to an optimization problem. Applications of the NFL theorems to information-theoretic aspects of optimization and benchmark measures of performance are also presented. Other issues addressed include time-varying optimization problems and *a priori* "head-to-head" minimax distinctions between optimization algorithms, distinctions that result despite the NFL theorems' enforcing of a type of uniformity over all algorithms.**

***Index Terms*— Evolutionary algorithms, information theory, optimization.**

information theory and Bayesian analysis contribute to an understanding of these issues? How *a priori* generalizable are the performance results of a certain algorithm on a certain class of problems to its performance on other classes of problems? How should we even measure such generalization? How should we assess the performance of algorithms on problems so that we may programmatically compare those algorithms?

Broadly speaking, we take two approaches to these questions. First, we investigate what *a priori* restrictions there are on the performance of one or more algorithms as one runs over the set of all optimization problems. Our second approach is to instead focus on a particular problem and consider the effects of running over all algorithms. In the current paper

Wolpert & Macready

The **no free lunch** theorem: all algorithms' performance is **equal**, averaged across all possible problems.

Doesn't prevent one algorithm being the best on **all problems we care about**.

No Free Lunch

- But still, researchers agree that there probably isn't one algorithm that is best on all problems we care about, either.
- Instead, different algorithms win by being **suited** to different problems.
- For example, an algorithm that uses mutation makes the assumption that neighbouring points have similar objective values. If this is true for a problem then the algorithm is suited to the problem.
  - See McDermott, When and Why Metaheuristics Researchers can Ignore "No Free Lunch" Theorems.

Before and after optimisation

Before optimisation comes problem framing

- Some projects fail by **using wrong optimisation techniques**
- **More** projects fail by **solving the wrong problem**

Before optimisation comes problem framing

- Some projects fail by **using wrong optimisation techniques**
- **More** projects fail by **solving the wrong problem**

"Far better an approximate answer to the right question, which is often vague, than an exact answer to the wrong question, which can always be made precise." – John Tukey

After optimisation comes decision-making

- Even after solving a simple problem, we may need to:
  - revisit assumptions and data
  - think about what the solution is telling us to do
  - decide whether to actually do it
- Multi-objective algorithms also leave us with a choice to make.

The broader field of **decision theory** is about how we **should** make decisions, and how we **actually** make decisions.

Gaming the system

Suppose we want to measure programmer productivity. We could measure lines of code written per day. What behaviour will this incentivise? Will it be a good metric?

Gaming the system

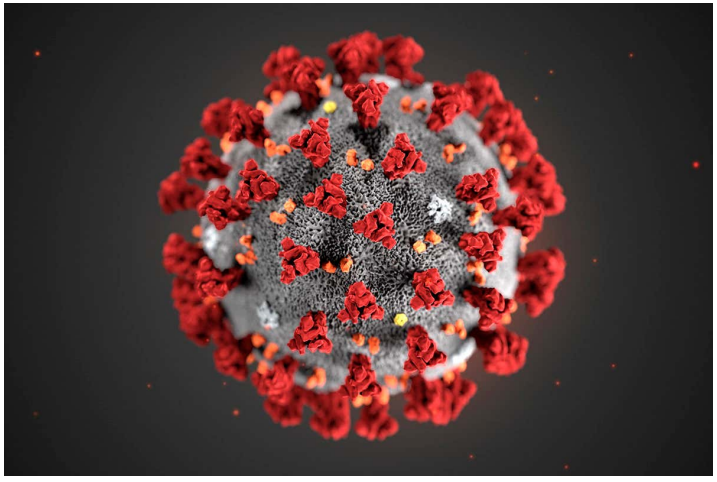
Suppose we want to measure programmer productivity. We could measure lines of code written per day. What behaviour will this incentivise? Will it be a good metric?

When we define a metric in human society, people are incentivised to exploit it ("gaming the system", "juking the stats" or "teaching to the test", aka Goodhart's Law or Campbell's Law).

The Wire

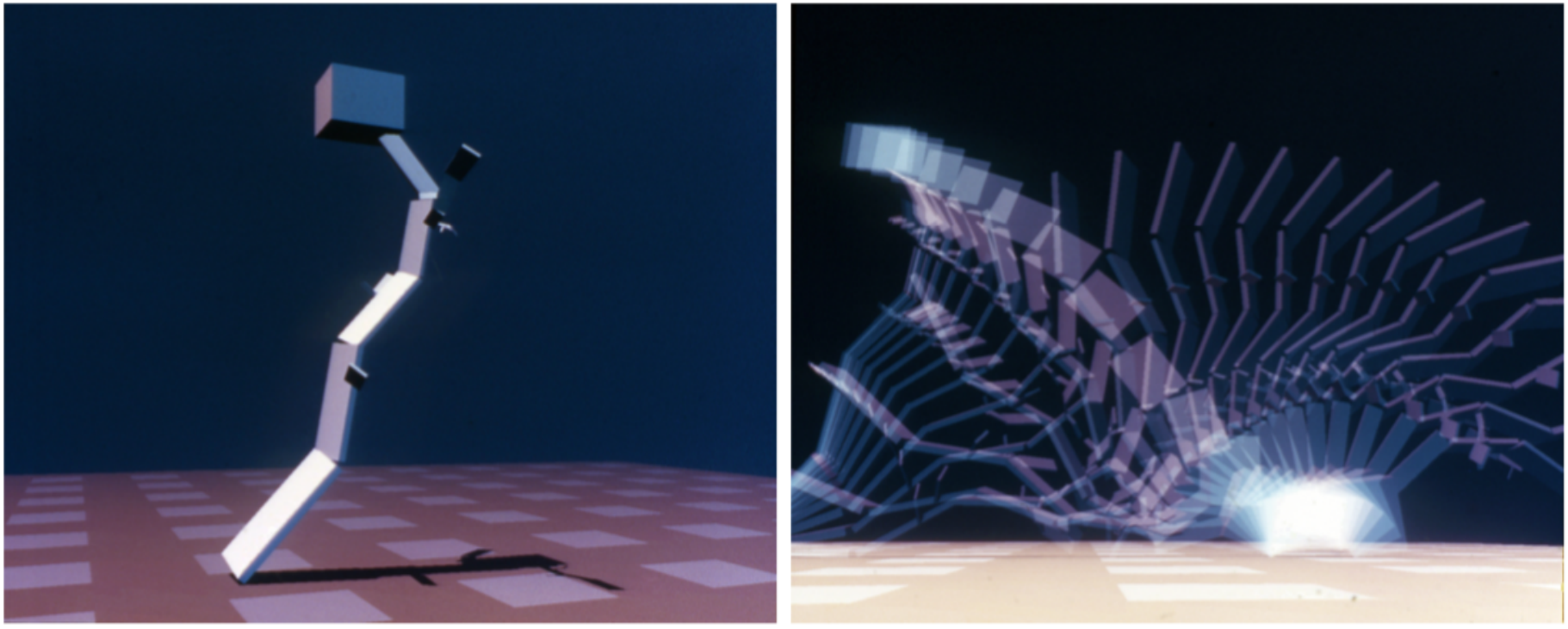
“An alien god”

Evolution is “an alien god”. In biology there is **no objective function** – just replication success. A virus is a good example: it has **no goal**.



EHAI

“An alien god”



Sims: a simulated robot falling instead of walking

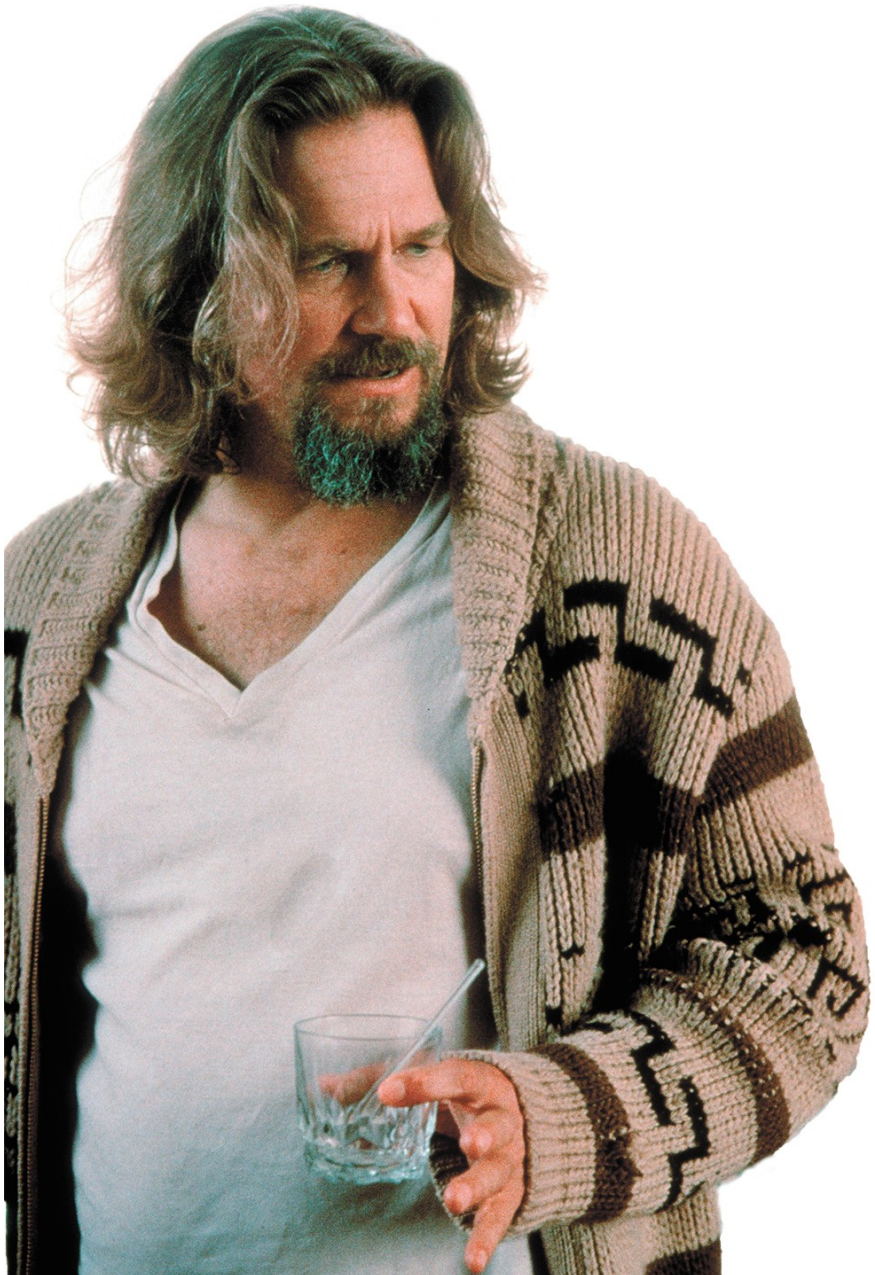
Optimisation algorithms are alien in the same way: the algorithm optimises for the objective function **without applying common sense**.

Optimisation algorithms gaming the system

OpenAI: boat catching fire and collecting Turbo

The RL agent finds an isolated lagoon where it can turn in a large circle and repeatedly knock over three targets, timing its movement so as to always knock over the targets just as they repopulate. Despite repeatedly catching on fire, crashing into other boats, and going the wrong way on the track, our agent manages to achieve a higher score using this strategy than is possible by completing the course in the normal way.

AI gaming the system



The Lebowski Theorem: No super intelligent AI is going to bother with a task that is harder than hacking its reward function – Joscha Bach.

Some things we didn't study

We didn't study them, so they are not examinable.

Overview

1. Some things we didn't study
2. **A big picture of optimisation algorithms**
3. Common themes of the module

Types of search spaces

- Vectors of binary, integer, real, or constrained-real decision variables
- Combinatorial spaces like permutations, trees, graphs

A spectrum of algorithms

Ordered by **strength of assumptions**:

- Random search and exhaustive search
- Black-box search
  - Some correlation between neighbours' objective values
- Constructive heuristics
  - Objective is a sum of parts
- Gradient descent
  - Objective may be a sum or mean of a loss function across dataset
    - With or without back-propagation
  - Or not!
  - Objective is convex, or a local optimum is good enough
- Linear/integer programming

Iterative search/optimisation versus construction

Iterative search/optimisation

- HC, SA, LAHC, GA, PSO, CMA, NSGA2, GD, LP (?)

Construction:

- Deterministic heuristics, GRCH, good old differentiation ($df/dx = 0, d^2f/dx^2 < 0$)

Hybrid:

- GRASP

Single-point search versus multi-point

Single-point:

- HC, SA, LAHC, GD, LP

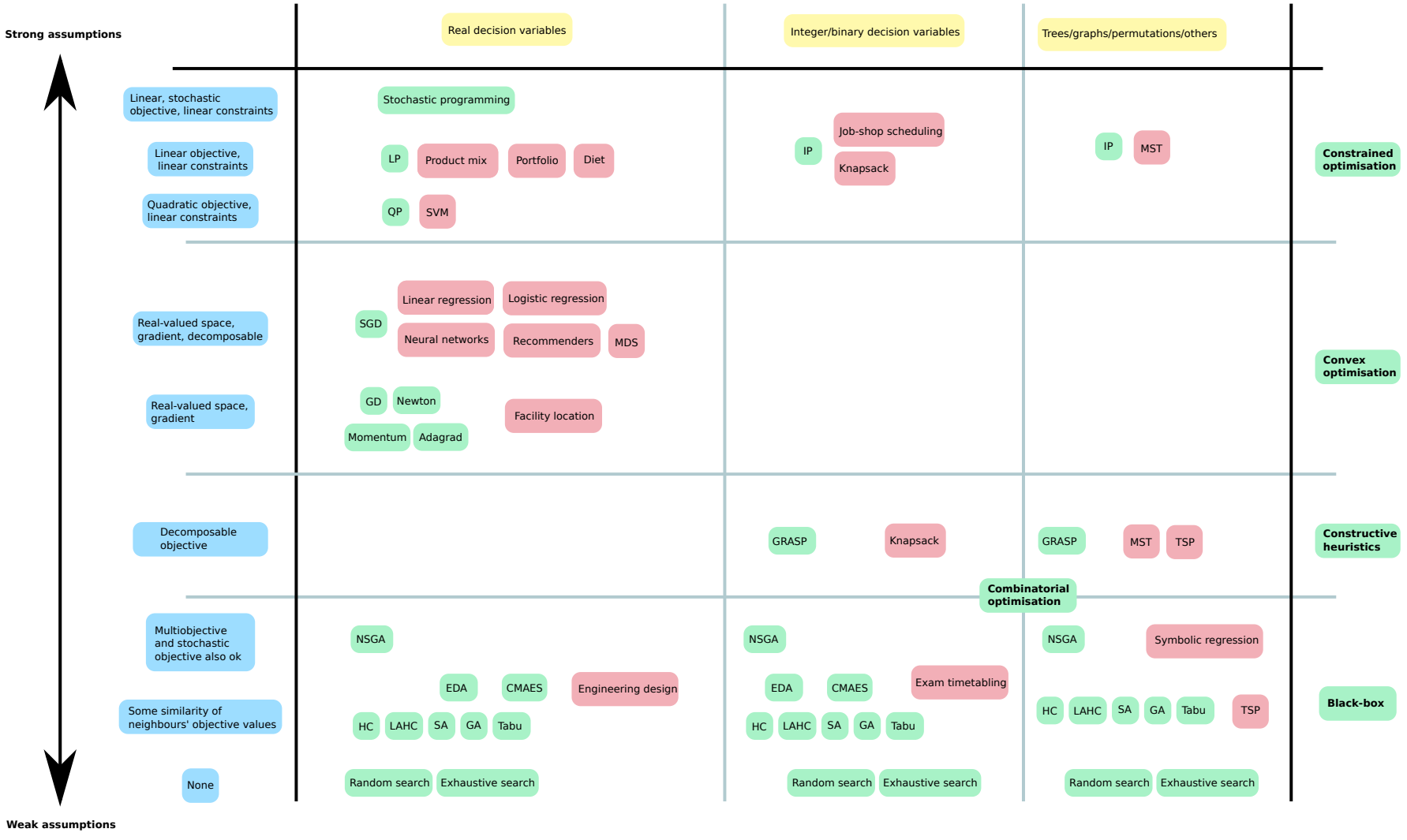
Multi-point:

- GA, NSGA2, PSO, CMA

Optimisation Landscape
James McDermott
MSc Artificial Intelligence
National University of Ireland, Galway

Legend | Assumptions | Search space type | Applications | Techniques

OÉ Gaillimh
NUI Galway



Overview

1. Some things we didn't study
2. A big picture of optimisation algorithms
3. **Common themes of the module**

Common themes

- Scalability
- Landscapes: local optima, problem difficulty
- Exploration and exploitation
- Hyperparameters
- Metaphors
- How much information does the objective give us?
- Choice of algorithm for the problem

Scalability

We often study small problems (small search spaces, e.g. few decision variables):

- to keep things simple
- allows us to draw pictures

But they are misleading:

- e.g. could be solved by enumeration.

Researchers often study **scalability** of algorithms, i.e. performance trends for problems of increasing size.

Landscapes

The landscape affects the difficulty of the problem:

- Easy problems have smooth, unimodal landscapes
- Hard problems have many local optima: e.g. hill-climbing gets "stuck"
- Black-box search: several methods of escaping local optima
- Gradient descent: some problems are convex
- LP: no local optima by definition

Landscapes

The landscape affects the difficulty of the problem:

- Easy problems have smooth, unimodal landscapes
- Hard problems have many local optima: e.g. hill-climbing gets "stuck"
- Black-box search: several methods of escaping local optima
- Gradient descent: some problems are convex
- LP: no local optima by definition

Landscape concepts don't apply to constructive heuristics *per se*.

Exploration and exploitation

(Applicable to black-box search and gradient descent, not LP/IP)

Choice of algorithm and hyperparameters allows control of **exploration-exploitation balance**

- Over-exploitation: get stuck in local optima, or stay too near starting point
- Over-exploration: no stronger than random search.

Hyperparameters

A problem throughout optimisation is that algorithms have hyper-parameters, so we may need to tune them.

- Black-box search, constructive heuristics, and gradient descent: many hyperparameters with strong influence on performance :(
- LAHC: tries to make the hyperparameter easier to tune
- CMA, Adam (and some others): robust self-tuning
- LP/IP: no hyperparameters to worry about :)

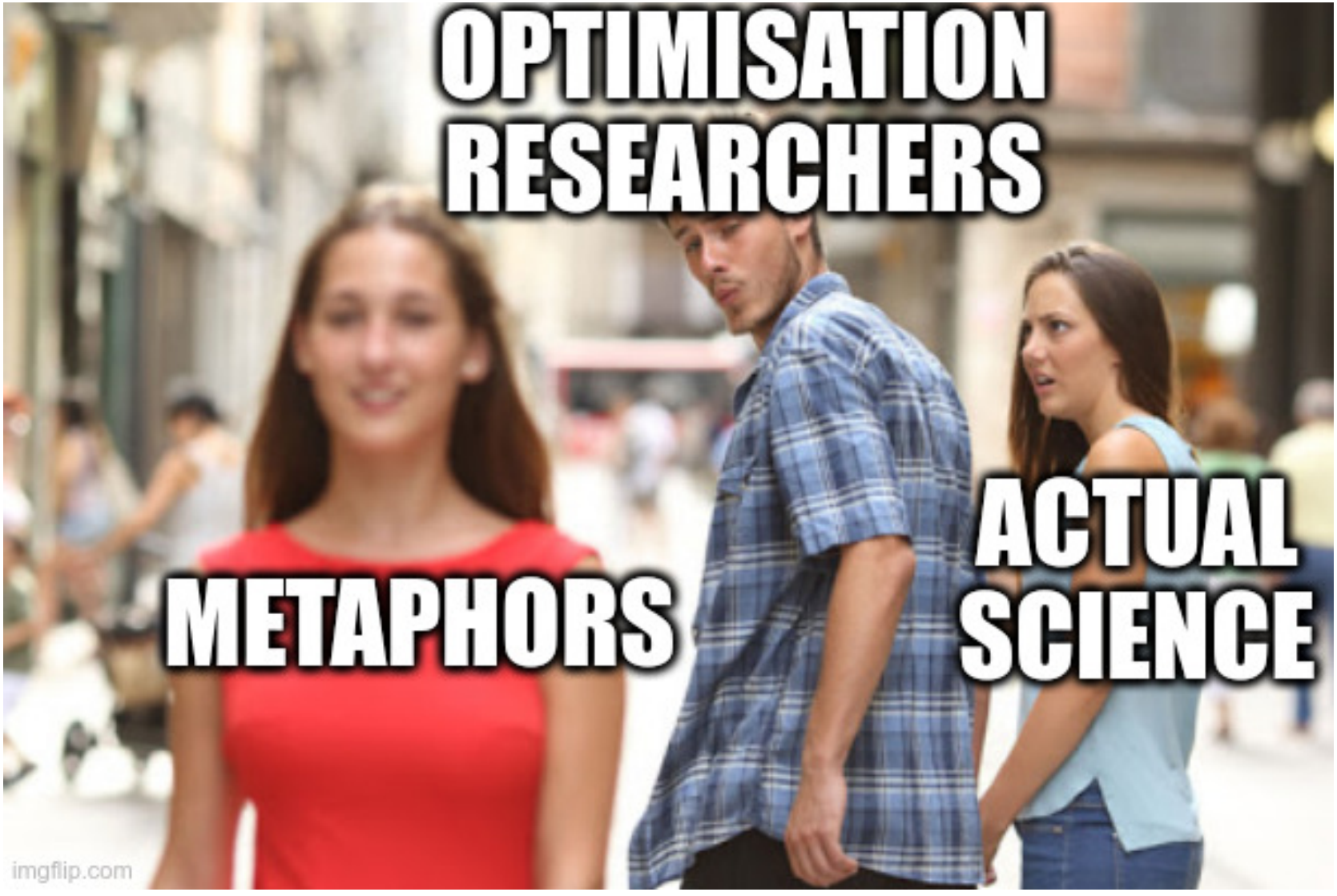
Metaphors

Many great ideas/algorithms are inspired by metaphors:

- **Biological**, e.g. GA, PSO
- **Energy-based**, e.g. graph layout, gradient descent, simulated annealing
- **Mountain landscapes**, e.g. fitness landscape, hill-climbing.

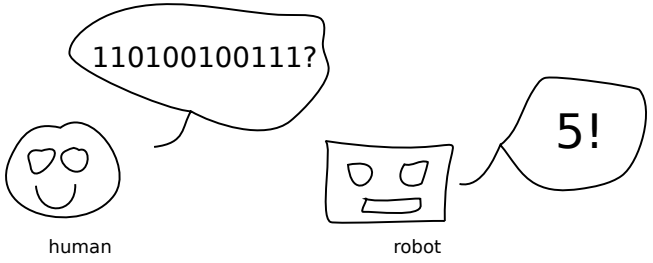
Many bad ones too! See e.g. Sörensen, Metaheuristics—the Metaphor Exposed (in Bb).

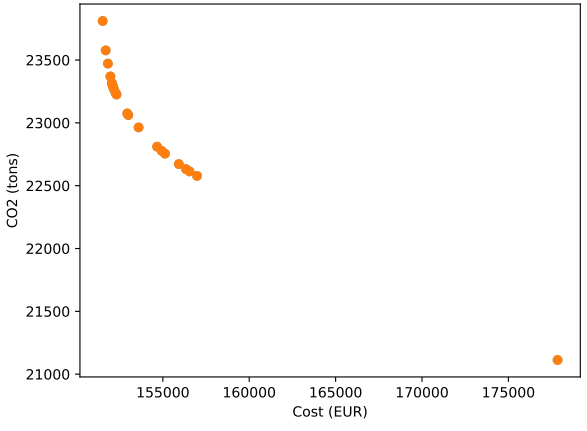
Metaphors


OPTIMISATION RESEARCHERS
METAPHORS
ACTUAL SCIENCE
imgflip.com

How much information does the objective give us?

- Recall the binary guessing game: if the robot tells us **how many** bits are correct, we solve it immediately
- More information is better
- What range is the objective in?
- If the objective is **smooth**, then one value tells us something about neighbouring values
- If the search space or objective is discrete, we may have many **ties**
- Could the objective be stochastic?
- In multiobjective, we no longer have an unambiguous ordering





Choosing algorithms for problems

The right algorithm depends on **what we know about the problem**.

- Are there constraints?
- Is the objective function linear?
- Is there a gradient?
- Do we know how parts of a solution contribute to the whole?

Overview

1. Some things we didn't study
2. A big picture of optimisation algorithms
3. Common themes of the module