NIHR | Applied Research Collaboration South West Peninsula

@penCHORD_UoE
@peninsula_ARC

Module 5 : Geographic Modelling & Visualisation
Session 5B : Combinatorial Optimisation for Geospatial Problems
Elliott Coyne

"ahiti Nui Mare'are'a"

HSMA 5

#hsma5isalive

NIHR | Applied Research Collaboration South West Peninsula

@penCHORD_UoE
@peninsula_ARC

Module 5 : Geographic Modelling & Visualisation
Session 5B : Combinatorial Optimisation for Geospatial Problems
Elliott Coyne

"ahiti Nui Mare'are'a"

HSMA 5

#hsma5isalive

Facility Location Problems

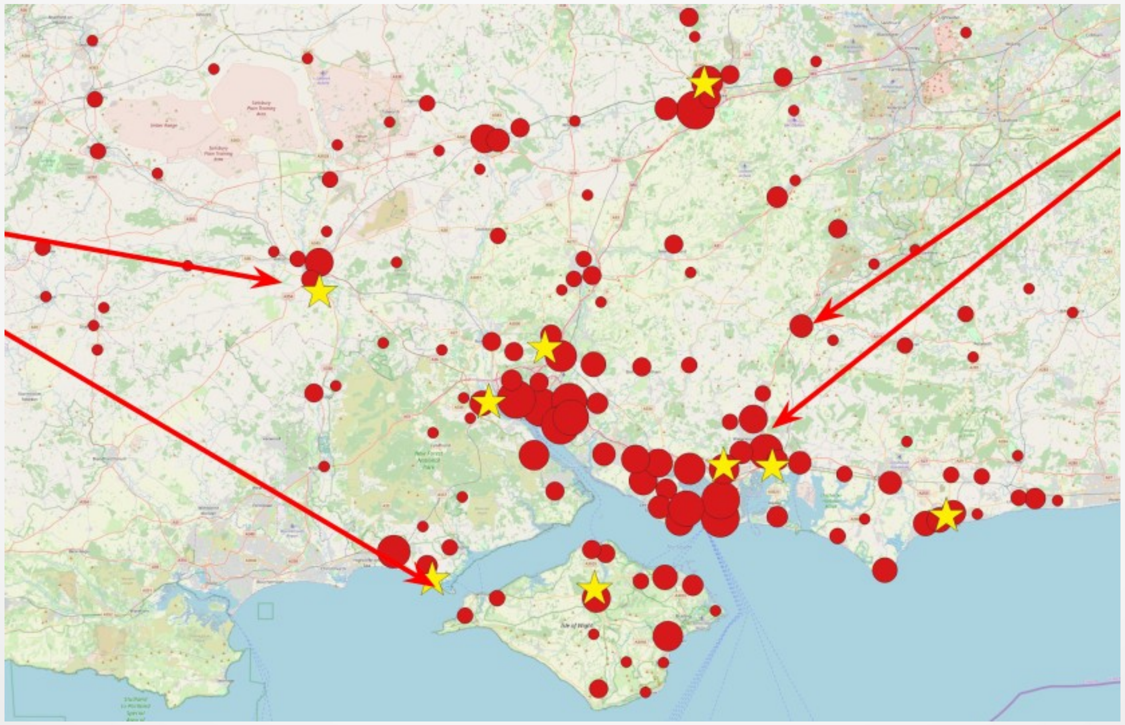
One job of budding Data Scientists is to help their organisations in making decisions when it comes to deciding about the locations for services.

Real life healthcare examples may include reviewing (and rationalising) existing service locations.

Understanding how to approach these problems will then given you data to use with maps. Great for Modules **5C - QGIS** and **5D – GeoPandas**.

Example...

★ Location of Dialysis treatment centres (i.e., hospitals)



Home 'location' of patients using dialysis service (size represents density)

Real Work Examples

Any examples to share with the group?

Feel free to put them in the chat...

Location Modelling

Other health examples include...

- A city hospital providing ophthalmology services needs support to decide where to locate 5 mobile clinics most equitably for users.
- A public health team and sexual health service provider organisation need support in reducing the number of locations for outpatient clinics in a region whilst still offering equitable access.
- NHS England want to support in centralising hyper-acute stroke care across England, to maximise equity, quality and patient outcomes.

When Size Matter

Small problems in health

- best supported using using a combination of simple mapping and analysis

Medium/ large scale applications in health

- interwoven with mathematical and computational optimisation and are possibly *multi-objective* in nature
- You should view this as decision support not just optimisation.
  - Provide quality information to be incorporated into the decision making process.

Data Requirements

For these types of problems, you’ll usually encounter…

Travel times/ distances – From different areas/ locations to existing/ proposed services or facilities (Postcodes, LSOAs)

Demand (aggregated) – Total numbers of service users originating from different locations

Demand (granular) – Details of individual service users and their origin (i.e., home address); these are required to calculate Max or 95th percentile travel times/ distances

Facility details – generally including their name, location and possibly other details.

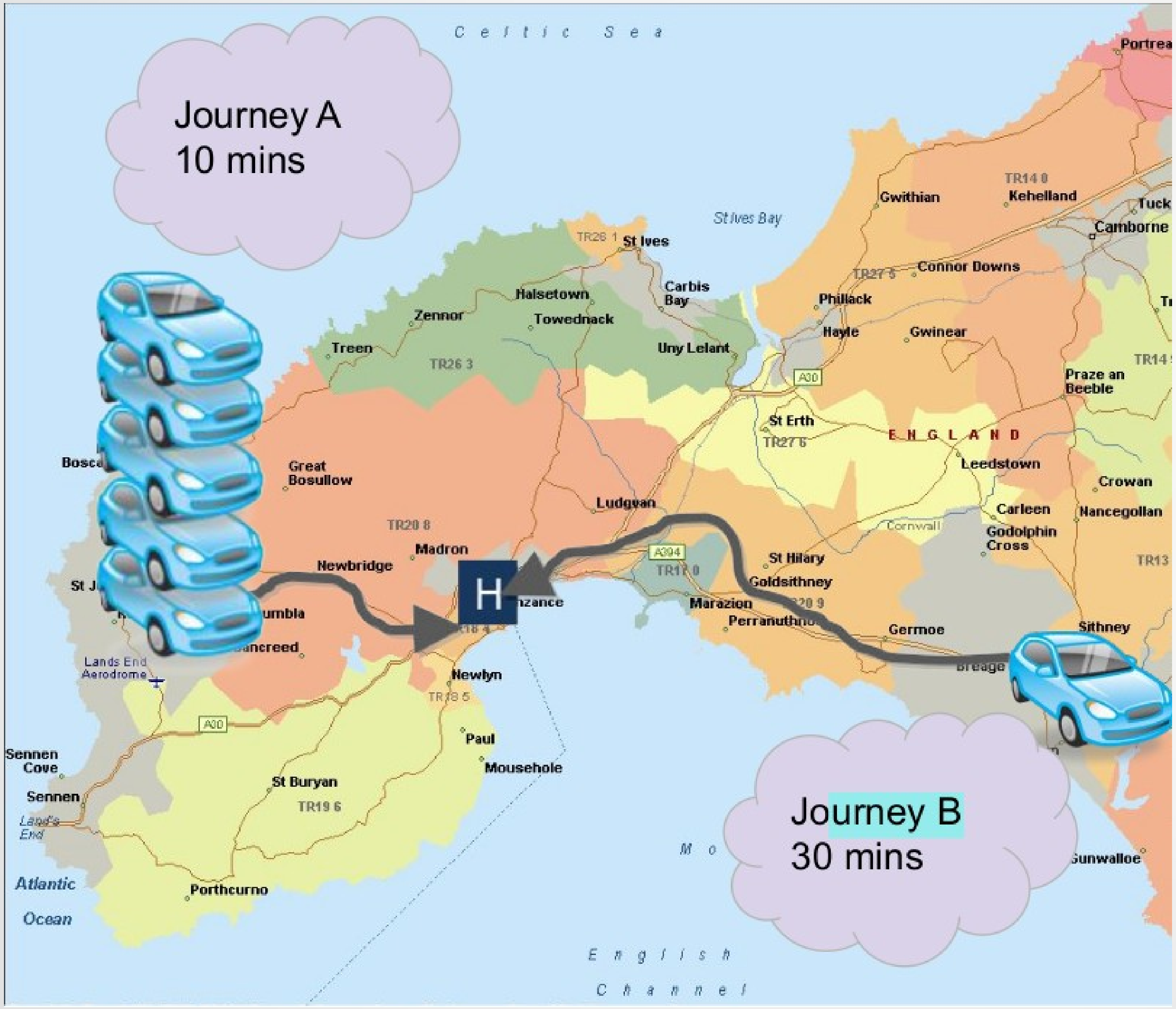
Location-allocation: use of travel times

- People often have to travel to health facilities to use services
- Generally felt that the closer the service to its community the better
  - May have life critical consequences in emergency health care.
- Travel times usually taken from GIS software
  - E.g. Routino, Open Source Routing Machine
- These are typically car travel times
  - may mean limitations to the analysis?
  - Ambulance service may be able to provide historical ‘blue light’ travel times.
- Public transport times more difficult to access
  - car ownership in study population?

Formulations of a location-allocation problem

- **p-median problem**
  - Given discrete demands locate a number (*p* or less) of health facilities so that the total weighted travel distance (***See next slide***) or time between facilities and demand centres is minimised
- **Location set covering problem**
  - Find the minimum number of facilities (and their locations) such that each and every demand centre is covered by at least one facility within a given maximal service distance (time) *E.g., 'everyone within 30 minutes of a service' – this may require 50 pins → 50 mins may only take 30 pins*
- **Maximal covering location problem**
  - Locate the facilities in such a way that as few people as possible lie outside the desired service distance *i.e., trade off between number of facilities and number of service users that are close to them. Relaxes the constraint of 'location set coverage problem')*

Weighted Average Explained



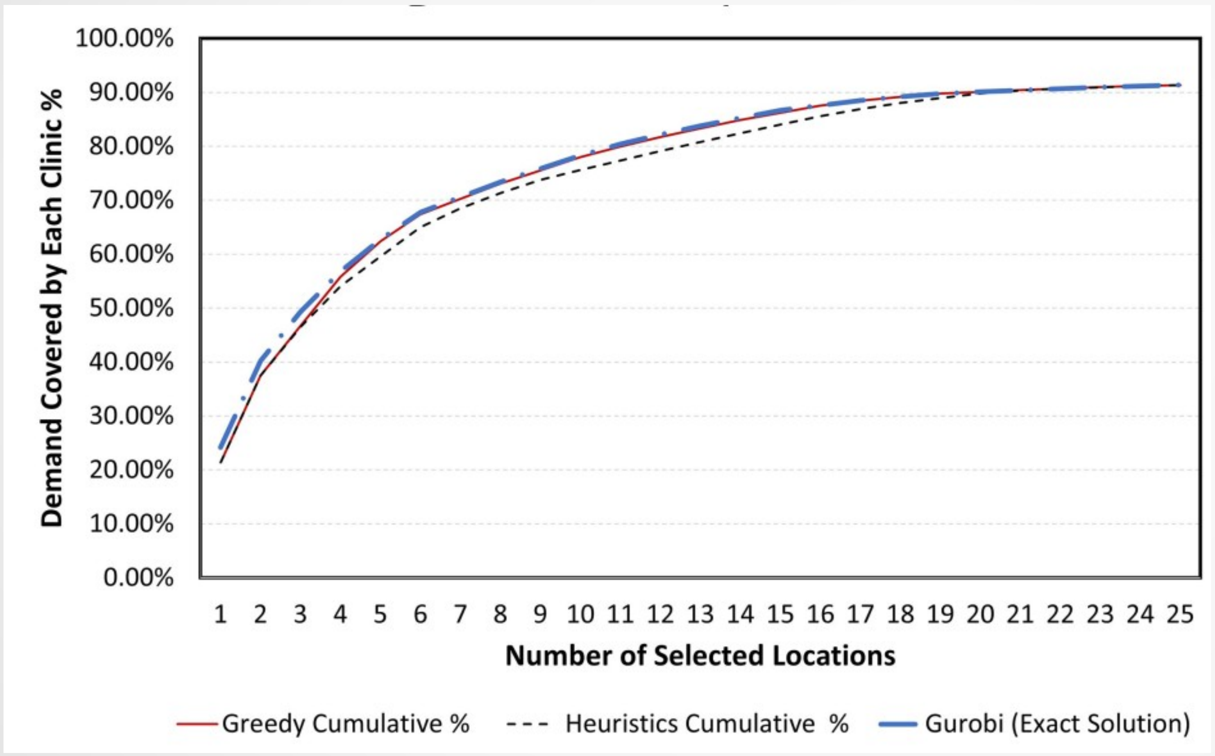
Weighted Average (WA)

A: 10 mins x 5 = 50
B: 30 mins x 1 = 30
80

WA = 80 / 6 trips = 13.3 mins

Average
10 + 30 / 2 = 20 minutes

Maximal covering location problem

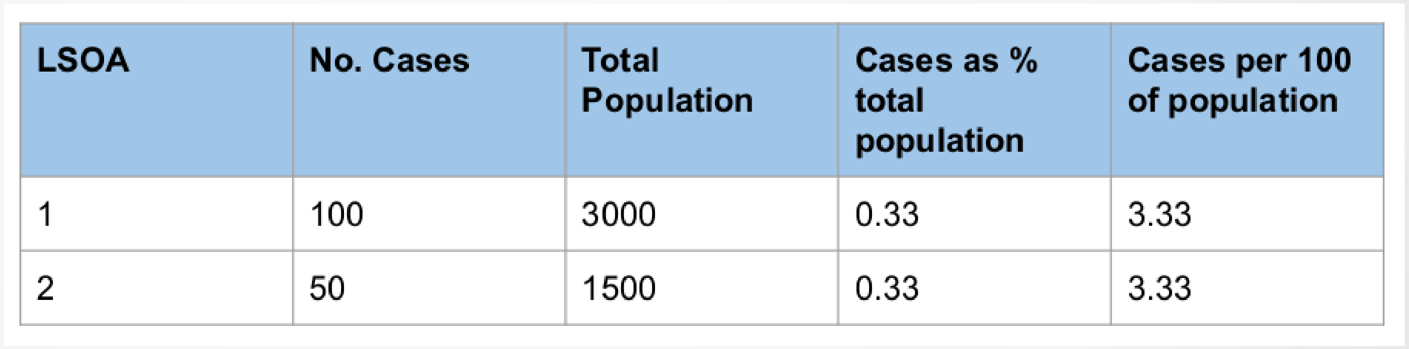


3 Different methods compared..
As number of locations included increases, so does % population covered – however eventually it flattens out around 18 facilities.

Considering 16 Vs. 18 – it's not just 16+2… The 'solution' may be made up of totally different locations

Raw vs. standardised geospatial demand

Just because No. of cases in LSOA1 is double that of LSOA2 – the rate is the same. So take care when presenting data!



LSOA	No. Cases	Total Population	Cases as % total population	Cases per 100 of population
1	100	3000	0.33	3.33
2	50	1500	0.33	3.33

Regional borders and edge cases

- If you are analysing an area it is possible demand will drop off as you approach the region’s border i.e., pop^n from edge of Cornwall, people may be travelling to Devon
- You need to assess how influential these low demand border areas are on your analysis and recommendations
  - Not always easy!
  - Sensitivity analysis that excludes low demand areas can help.
- What happens if you place a new facility near to a border?
  - Is it possible that it may attract new demand not seen before? Known to happen in health care!
  - Can you source extra data?
  - Can you estimate this demand and would it make a difference to the analysis and recommendations?

When Size Matter Even More!

Not always possible to compute all possible combinations!

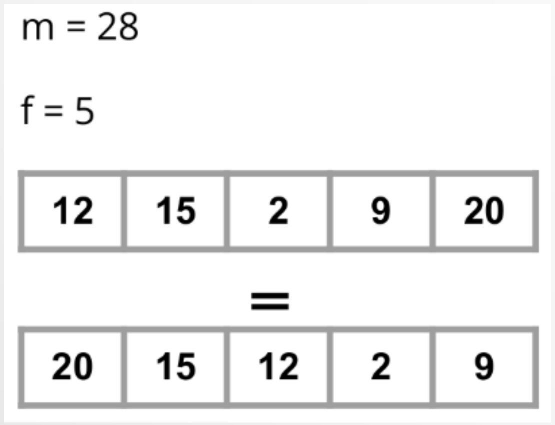
Evolutionary algorithms (EAs)

- Seeded with a **population** of possible solutions
- **Breeds and evolves** the population to find ‘good’ or ‘optimal’ solutions
- *In the context of facility location, each solution represents a possible combination of locations*
- The approach can be also be viewed as:
  - reinforcement learning or stochastic optimisation
- Common EAs are
  - Evolutionary strategies: **(μ, λ)** and **$(\mu + \lambda)$**
  - Genetic Algorithms (GAs)
- Single or multi-objective
- **Can be computationally expensive!**

Integer Representation - Chromosome

Integer *aka variable length array*

- A chromosome is of length *f*: number of facilities in a feasible solution
- Each element is encoded 1, 2, …, *m*-1, *m*



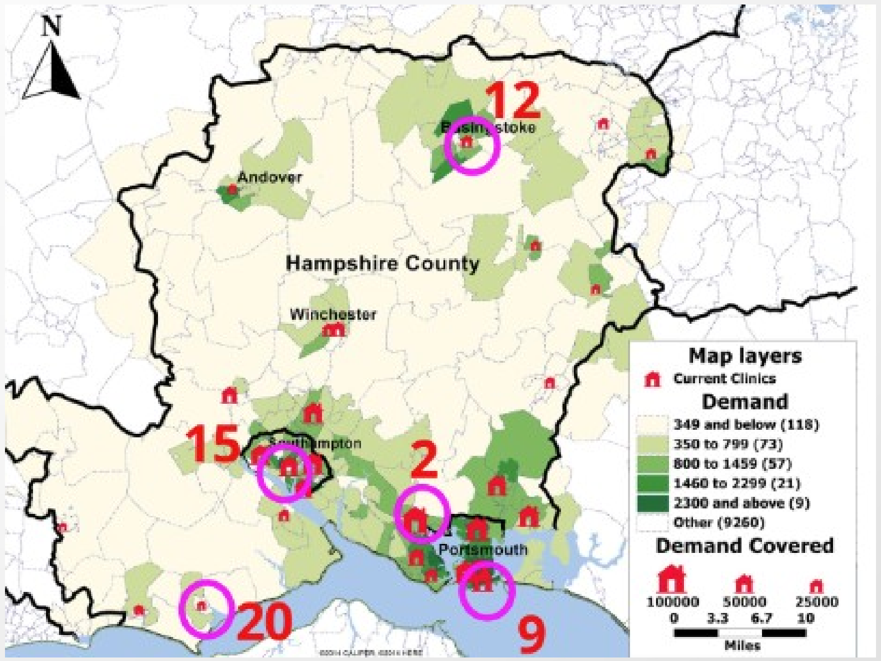
m = 28

f = 5

12	15	2	9	20

=

20	15	12	2	9



‘fitness’ Function

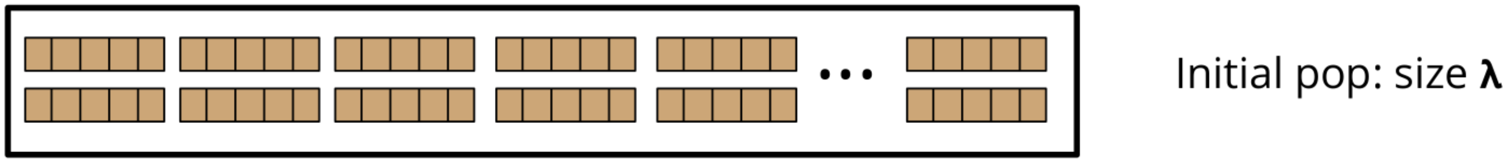
- In an EA, the quality of a solution is called its fitness
  - This is the target of the optimising algorithm *c.f.* cost

- In a single objective facility location problem fitness may be:
  - Weighted average travel time (p-median problem), or
  - Proportion of demand within a target travel time (maximal covering location problem).

- Likely requirements:
  - Candidate solution (your representation of a solution)
  - Travel cost matrix (e.g. travel time in minutes)
  - Demand by geographic location (e.g. stroke admissions by LSOA)

Evolutionary Algorithms

(λ) "lambda" represents initial population size (i.e. how many solutions)
(μ) "mu" represents the number of best solutions to keep, remainder are deleted



General EA Pseudo-Code

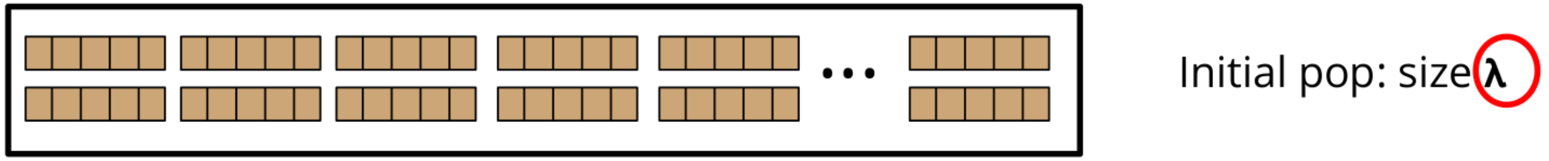
```
pop ← initial_population()
best = []

repeat:
    evaluate_fitness(pop)
    for individual in pop:
        if best = [] or fitness(individual) > fitness(best):
            best ← copy(individual)
     pop ← Join(pop, breed(pop))
until time_limit or max_iter
return best
```

(μ, λ) and $(\mu + \lambda)$

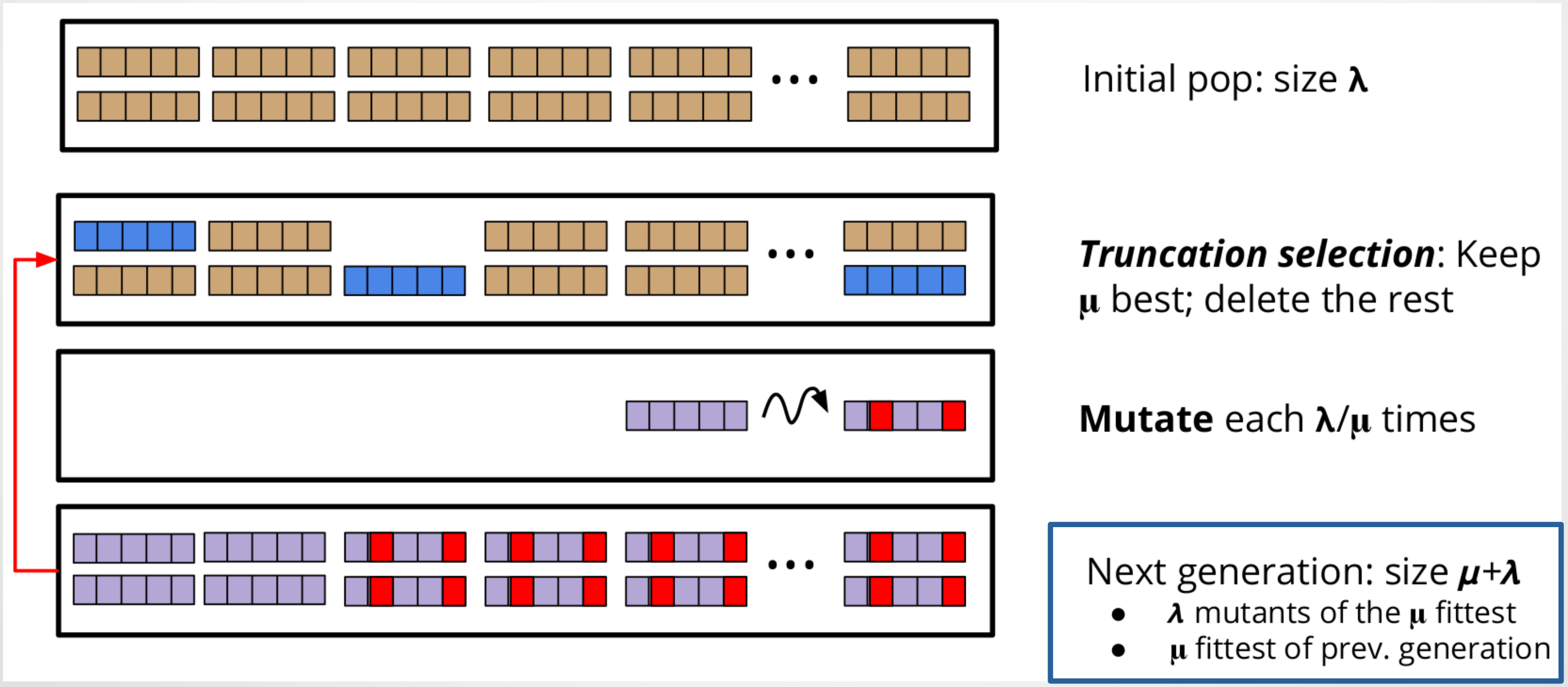
(μ, λ)

(λ) "lambda" represents initial population size (i.e. how many solutions)
(μ) "mu" represents the number of best solutions to keep, remainder are deleted



$(\mu + \lambda)$

(λ) "lambda" represents initial population size (i.e. how many solutions)
(μ) "mu" represents the number of best solutions to keep, remainder are deleted


Initial pop: size **λ**
Truncation selection: Keep **μ** best; delete the rest
Mutate each **λ/μ** times
Next generation: size ***μ*+*λ***
● ***λ*** mutants of the **μ** fittest
● **μ** fittest of prev. generation

Comparing (μ, λ) and $(\mu + \lambda)$

The main difference between the two algorithms is **ELITISM**

- The fittest parents from the previous generation persist

Benefits of ELITISM

- High performing solutions have opportunity to breed with other new high performing solutions

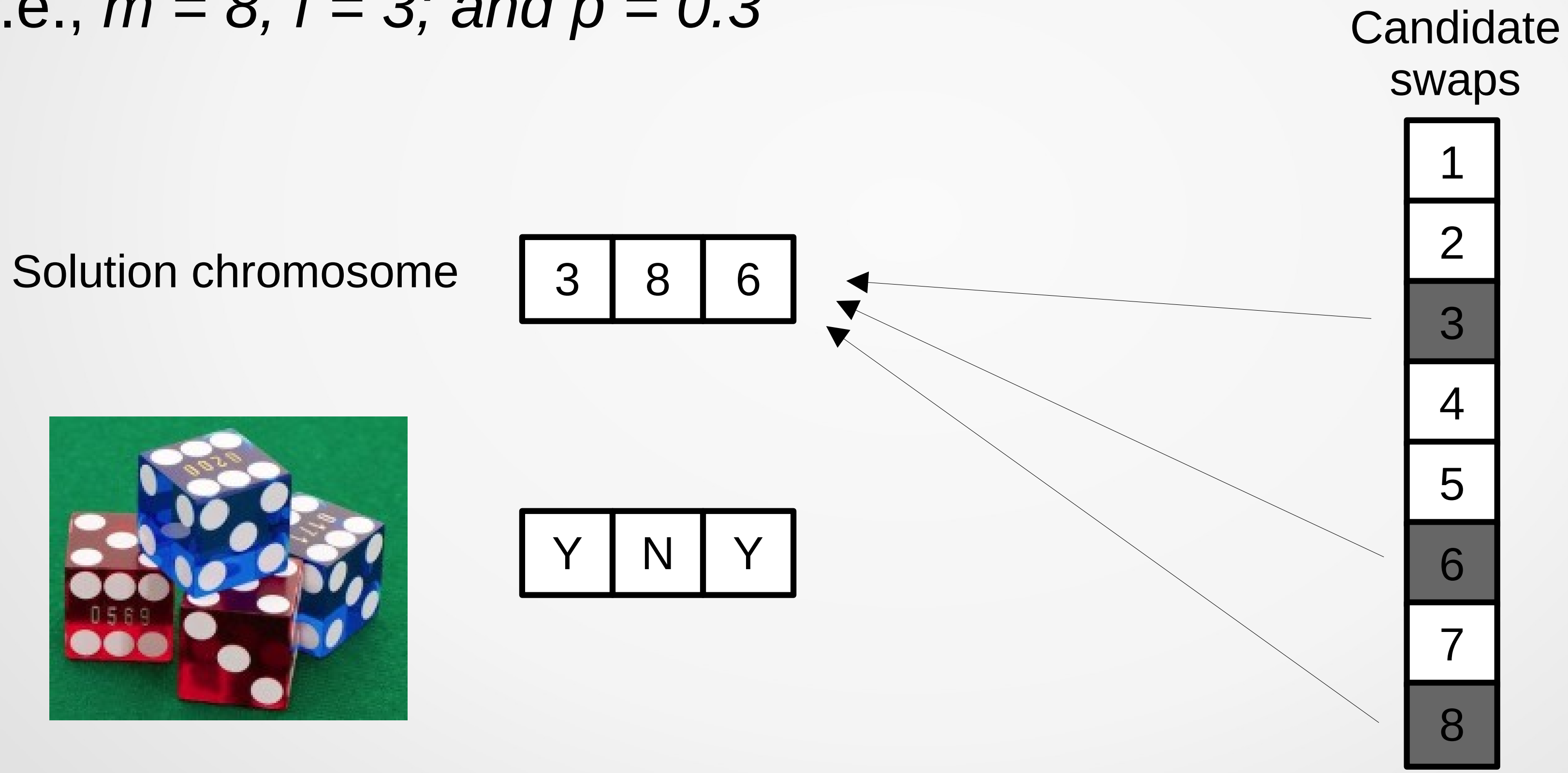
Risk

- Potentially reduces EX**PLOR**ATION in favour of more EX**PLOIT**ATION.
- Failure to learn and getting stuck in a local optimum.

Mutation Operator

Mutate each element within chromosome with probability *p*

i.e., $m = 8$, $f = 3$; and $p = 0.3$



Sample "Y" (Yes!) with probability *p*

Mutation Operator

Mutate each element within chromosome with probability *p*

i.e., *m = 8, f = 3; and p = 0.3*

Candidate swaps

1
~~2~~
3
4
5
6
~~7~~
8

Solution chromosome

3	8	6

Y	N	Y

Swap if "higher" than 0.3

2

7

Sample "Y" (Yes!) with p

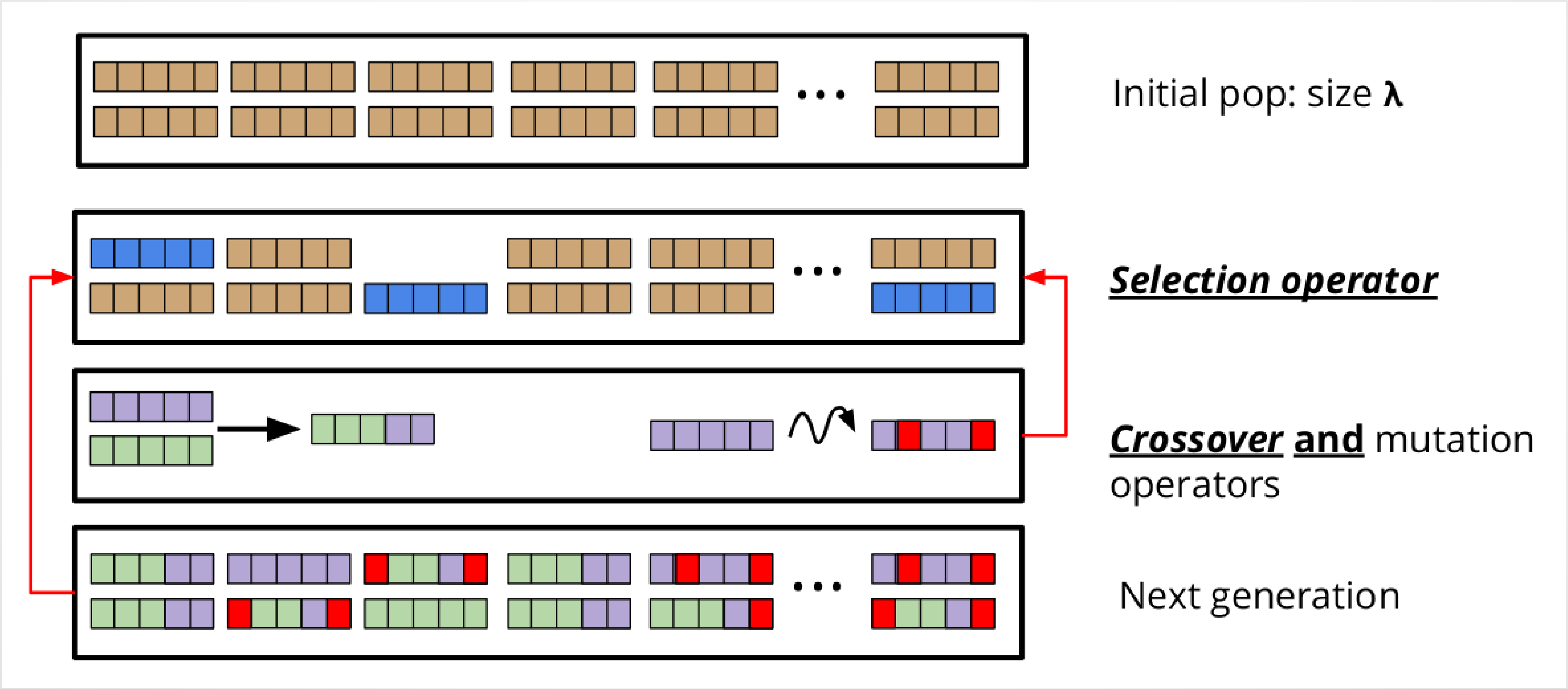
Mutant

2	8	7

Sample without replacement

Genetic Algorithms

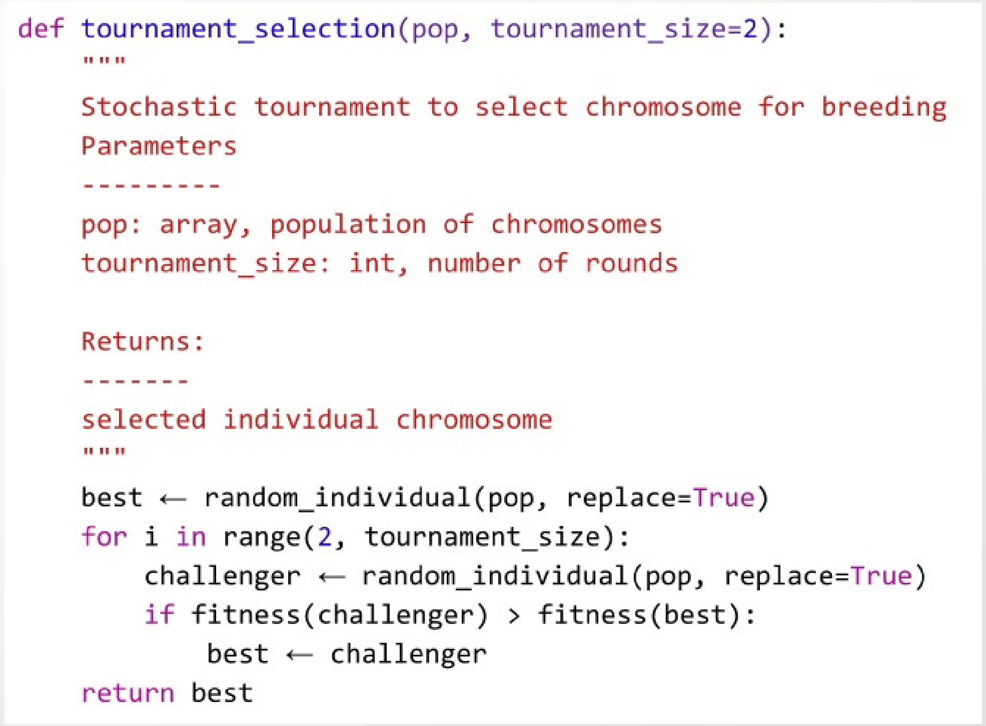
Genetic Algorithms



Selection operator

Classic approaches

- Roulette Selection
  - Not used much these days
  - Use random sampling, where probability for selection is proportional to its fitness. More here.
- **Tournament Selection**
  - Pick a ‘Champion’ at random then pick some ‘Challengers’
  - ‘Challengers’ try and beat that ’Champion’.

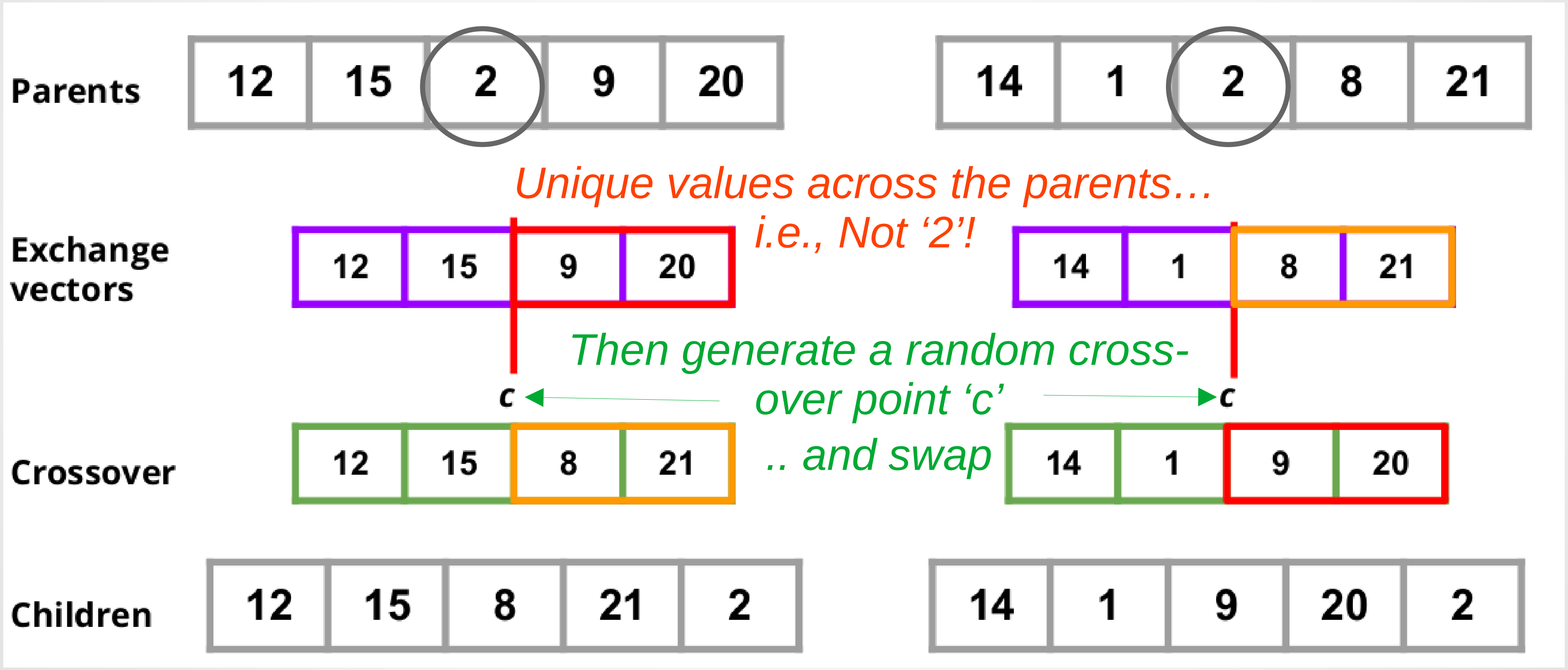


```
def tournament_selection(pop, tournament_size=2):
    """
    Stochastic tournament to select chromosome for breeding
    Parameters
    ----------
    pop: array, population of chromosomes
    tournament_size: int, number of rounds

    Returns:
    -------
    selected individual chromosome
    """
    best ← random_individual(pop, replace=True)
    for i in range(2, tournament_size):
        challenger ← random_individual(pop, replace=True)
        if fitness(challenger) > fitness(best):
            best ← challenger
    return best
```

Crossover: Single Point

(As opposed to multi-point...)



Advice on practical use of EAs

Remember the approach is powerful and also computationally expensive

- Only use for medium to large problems
- Preliminary work should involve setting a benchmark (similar to forecasting!)
  - A **greedy method** (i.e., only considering the next step rather than overall problem) and or **random search** are easy to implement and quick to run
    - Also easy for a health service stakeholders to understand (compare to GAs)
  - Only use if it evidence that EAs offer improvement
- GAs vs. Evolutionary strategies
  - GAs need **more** tuning and have **longer** runtimes (as more to “do”)
- Often a bottleneck in execution is the fitness function
  - Large populations, lots of generations and complex objectives → careful implementation (Implementation in Python Vs. Numpy Vs. C; using a dictionary to cache values of chromosomes to save unnecessarily running fitness function)
- The state-of-the-art literature will have *ideas*, but **unlikely** to include code

Summary

- Evolutionary strategies versus full Genetic Algorithm
- Key steps:
  - Decide how to represent the problem
  - Code a function to evaluate the fitness of a solution
  - Choose and code selection, crossover and mutation operators
  - Bring it all together in a ES or GA framework
  - Tune parameters

What We'll Cover….

During this session you'll be shown a live demo of the following steps. You'll then work in your groups to tackle a different problem, using similar steps for yourselves.

- Library Imports
- Data Imports
- Representing a Solution
- Constructing a Random Solution
- Evaluating a Solution
- Small Problem: Enumerating all Possible Combinations
- Bruteforce Solution
- Graphical Representation of Bruteforce Solution
- Medium to Large: Using random restarts (Demo Only)

What You’ll Need...

Please open the following Jupyter Notebooks:

- *Code Along...*
- *Group Exercise…*

Demonstration only:

- *Evolutionary and Genetic Algorithms*