GENETIC ALGORITHM AND PROGRAMMING, AND THEIR APPLICATION

OVERVIEW

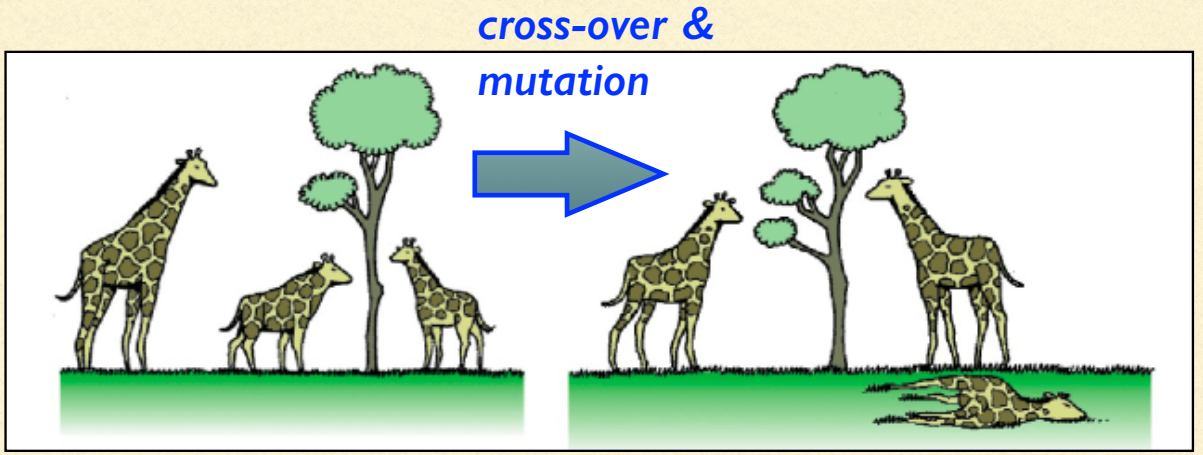
- What is Genetic Algorithm: one-max, TSP
- What is Genetic Programming: symbolic regression, even parity bit generator

WHAT IS GENETIC ALGORITHM?

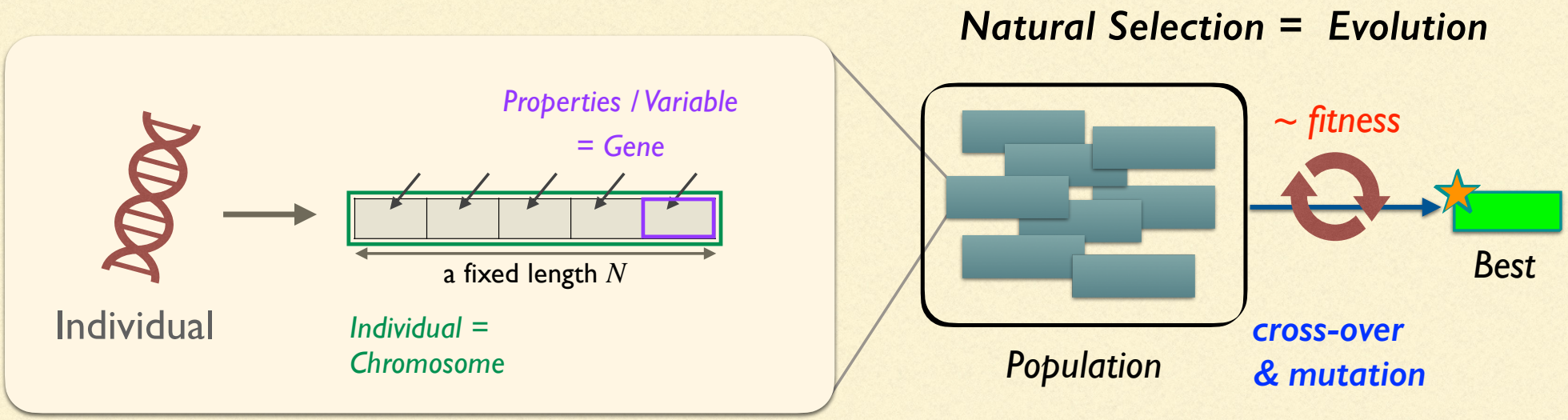
Genetic Algorithm (GA)

- A meta-heuristic algorithm that mimics the evolution process in nature: *survival of the* ***fittest***

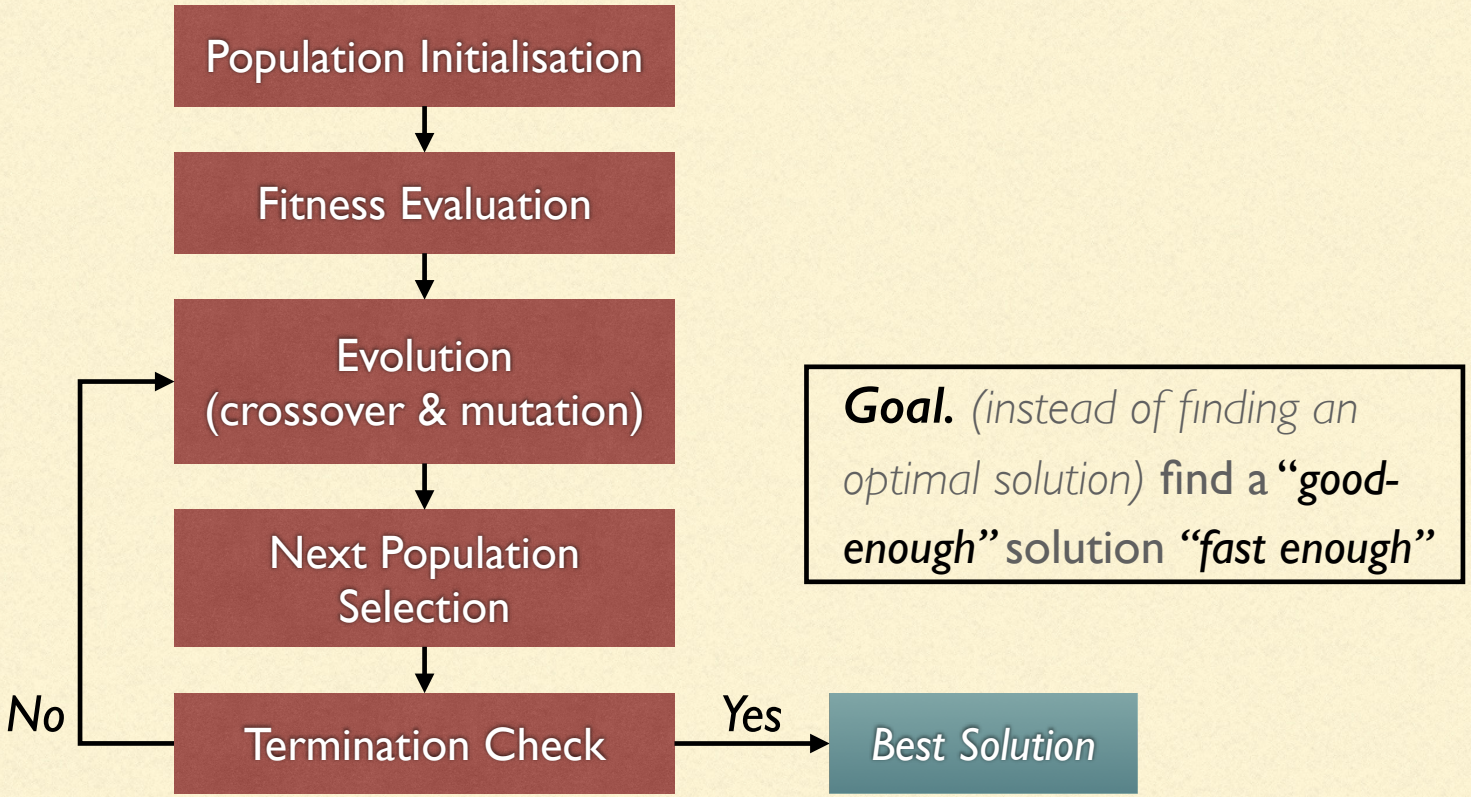
Natural Selection



WHAT IS GENETIC ALGORITHM?



WHAT IS GENETIC ALGORITHM?: OVERFLOW

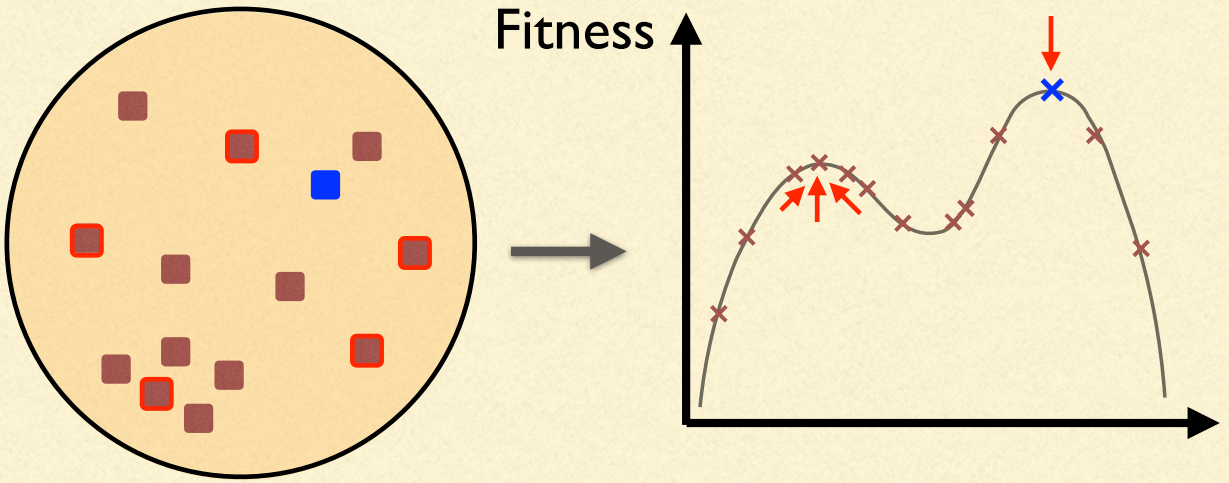


Goal. *(instead of finding an optimal solution)* **find a** ***"good-enough"*** **solution** ***"fast enough"***

GA: POPULATION INITIALISATION

Population Initialisation

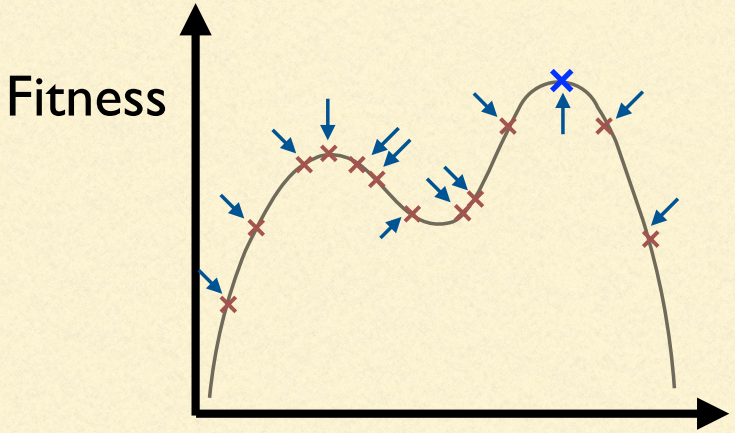
- Random Initialisation
- Heuristic Initialisation



Solution
(Computation) Space

GA: FITNESS FUNCTION

Fitness Evaluation



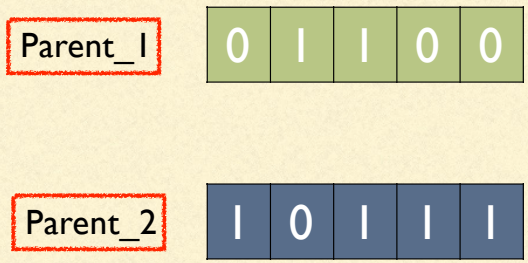
Guide the evolution

- Should quantitatively measure how fit an individual solution is
- Fast to compute

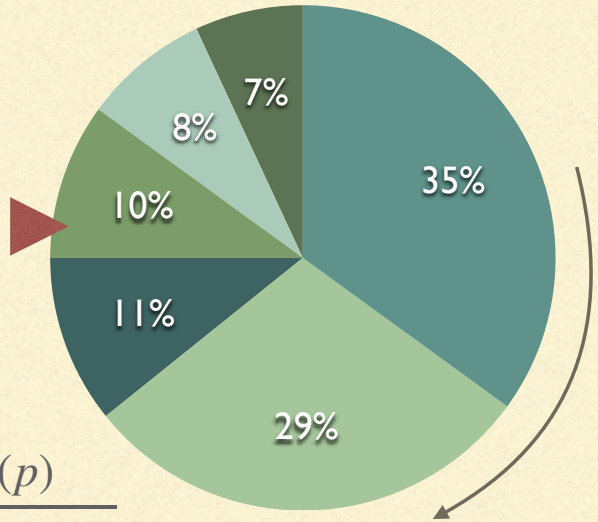
GA: CROSSOVER AND MUTATION?

Evolution
(crossover & mutation)

- Crossover *
- Mutation



Natural selection (fittest) &
Avoid early convergence



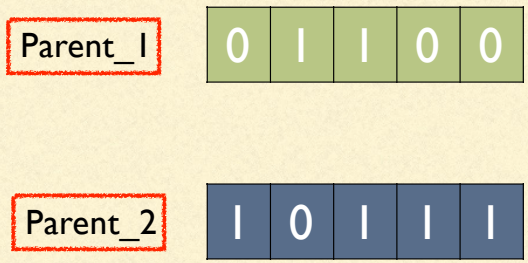
$$Prob(p) = \frac{\text{fitness}(p)}{\sum_{p \in P} \text{fitness}(p)}$$

Roulette Wheel Selection

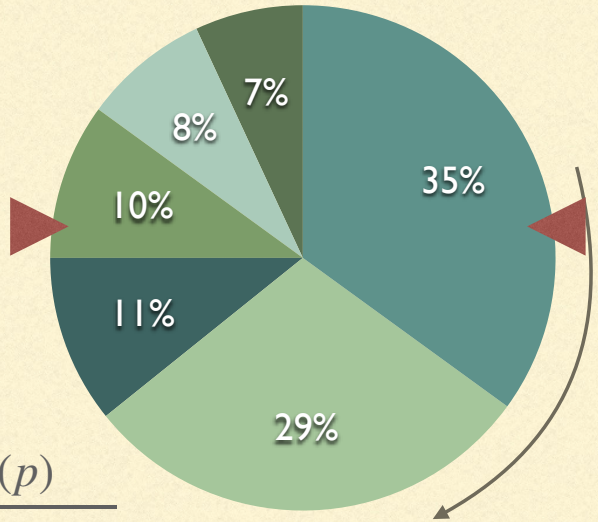
GA: CROSSOVER AND MUTATION?

Evolution
(crossover & mutation)

- Crossover *
- Mutation



Natural selection (fittest) &
Avoid early convergence



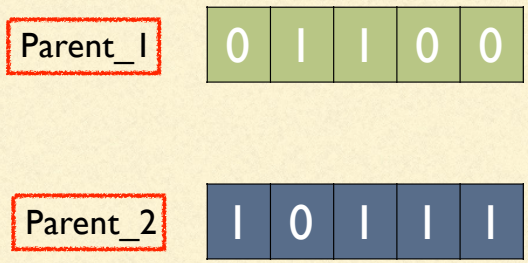
$$Prob(p) = \frac{\text{fitness}(p)}{\sum_{p \in P} \text{fitness}(p)}$$

Stochastic Universal Sampling

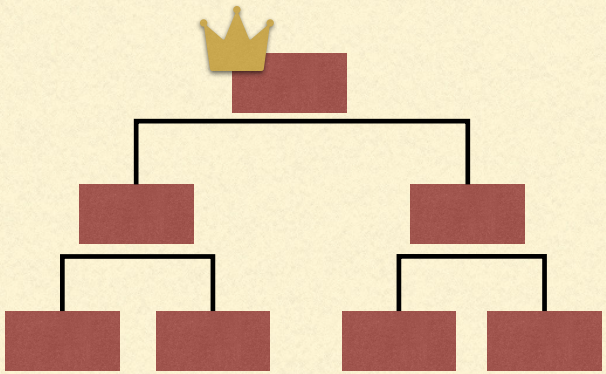
GA: CROSSOVER AND MUTATION?

Evolution
(crossover & mutation)

- Crossover *
- Mutation



Natural selection (fittest) &
Avoid early convergence



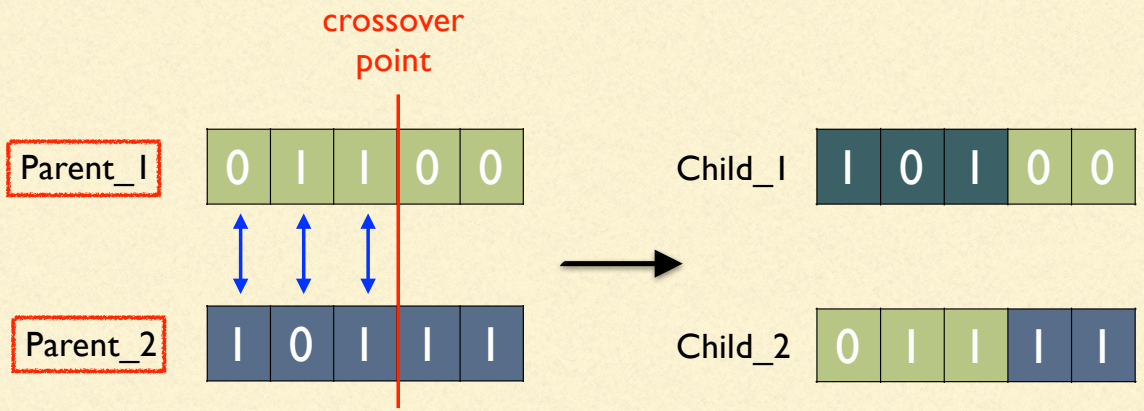
Tournament Selection

K (e.g., = 4) individuals to participate in the tournament

GA: CROSSOVER AND MUTATION?

Evolution
(crossover & mutation)

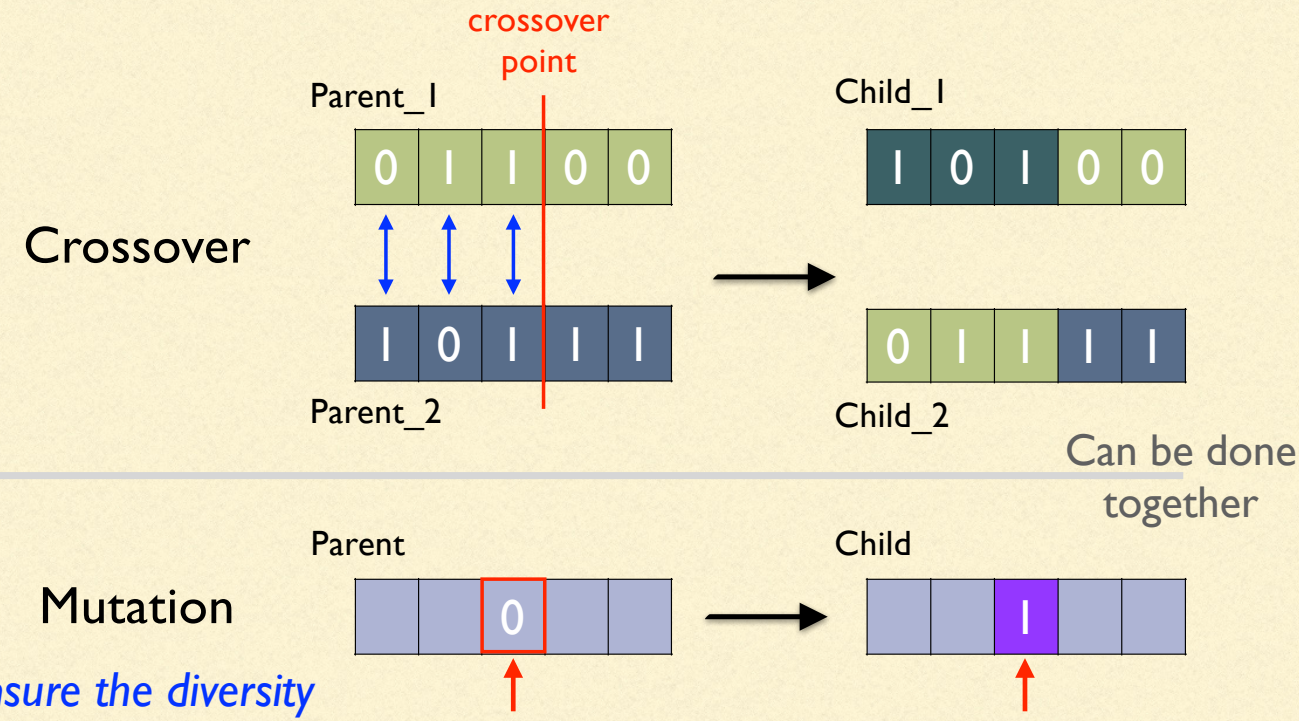
- Crossover *
- Mutation



GA: CROSSOVER AND MUTATION?

Evolution (crossover & mutation)

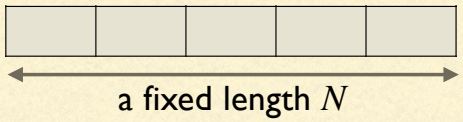
- Crossover
- Mutation *



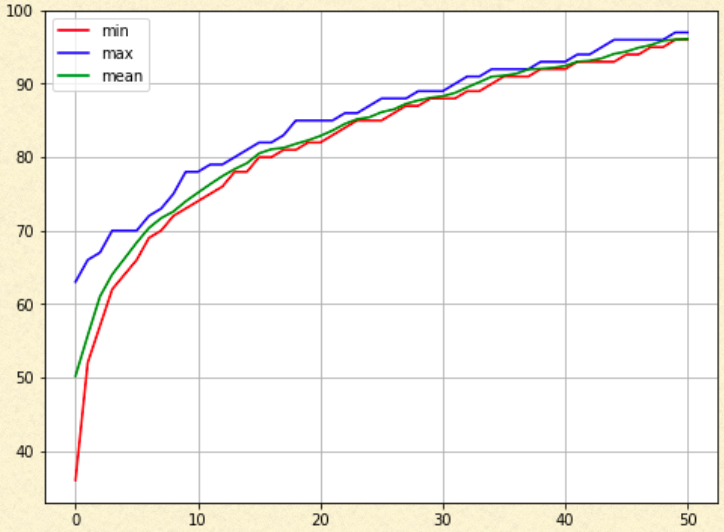
GA: ONE-MAX PROBLEM EXAMPLE

One Max problem:

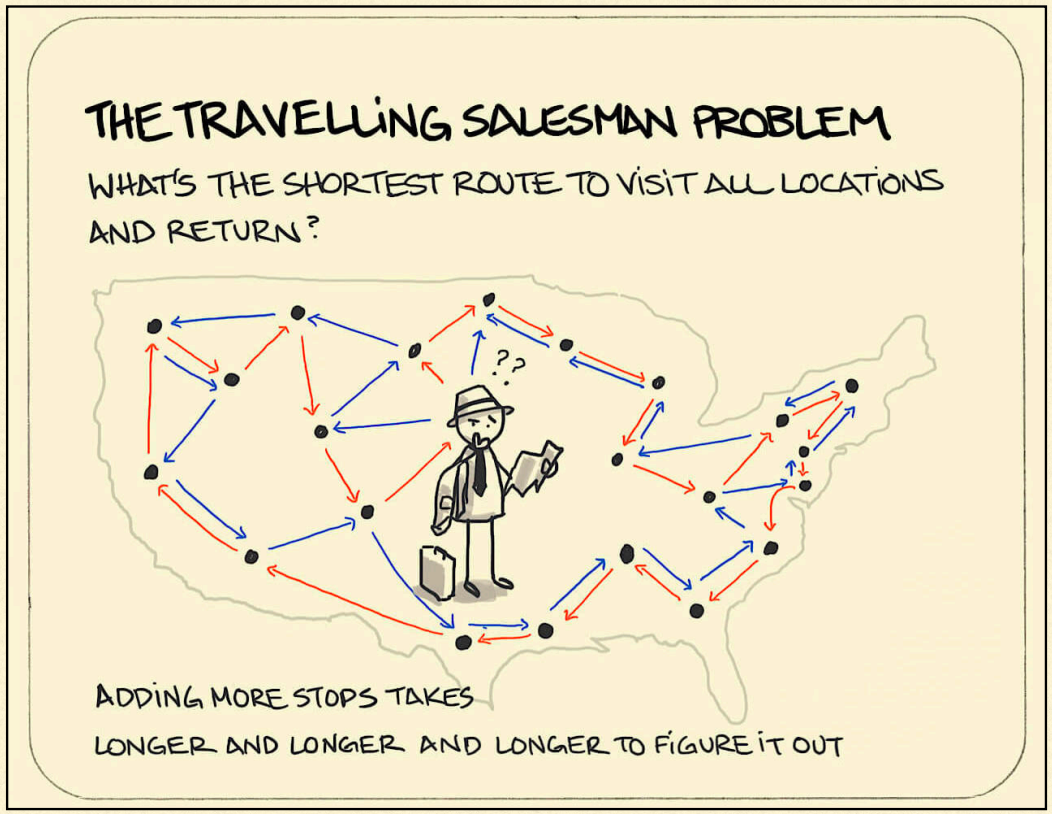
maximise the number of ones in a bitstring => Fitness



- Individual: a binary list
- Population: a population of size 300
- Crossover: ind1[:i] <-> ind2[:i]
- Mutation: flip i'th bit



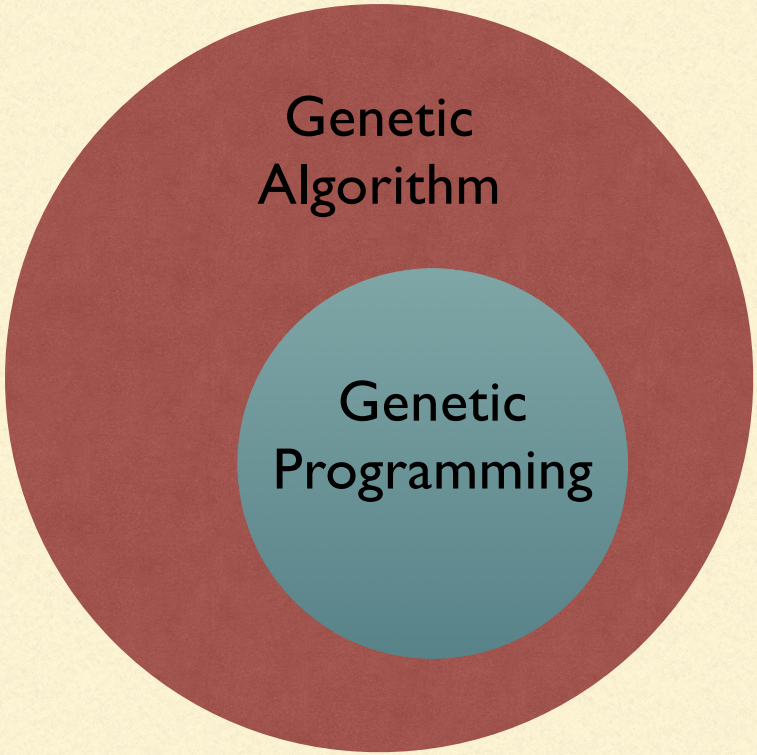
GA: TRAVELLING SALESMAN PROBLEM (EXERCISE)



Travelling Salesman Problem (NP-hard)

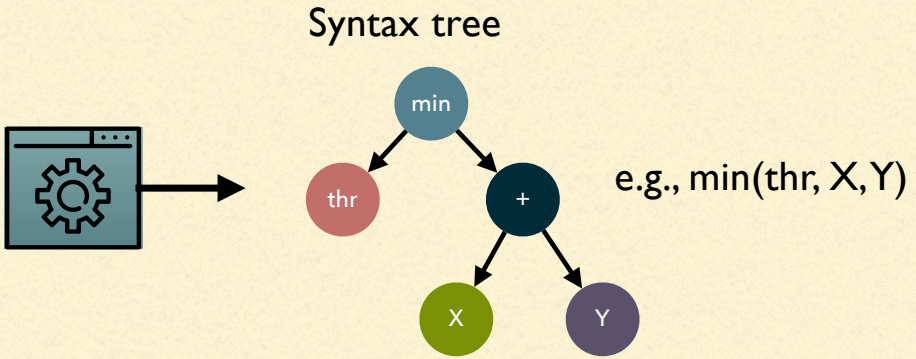
- Individual: a single visiting order (List)
- Fitness:
  - Minimise the total distance when visiting the cities sequentially as indicated in the individual
- Crossover:
  - a single-point crossover (tools.cxOrdered)
- Mutation:
  - shuffling (tools.mutShuffleIndexes)

GENETIC PROGRAMMING IS A SPECIFIC BRANCH OF GENETIC ALGORITHM

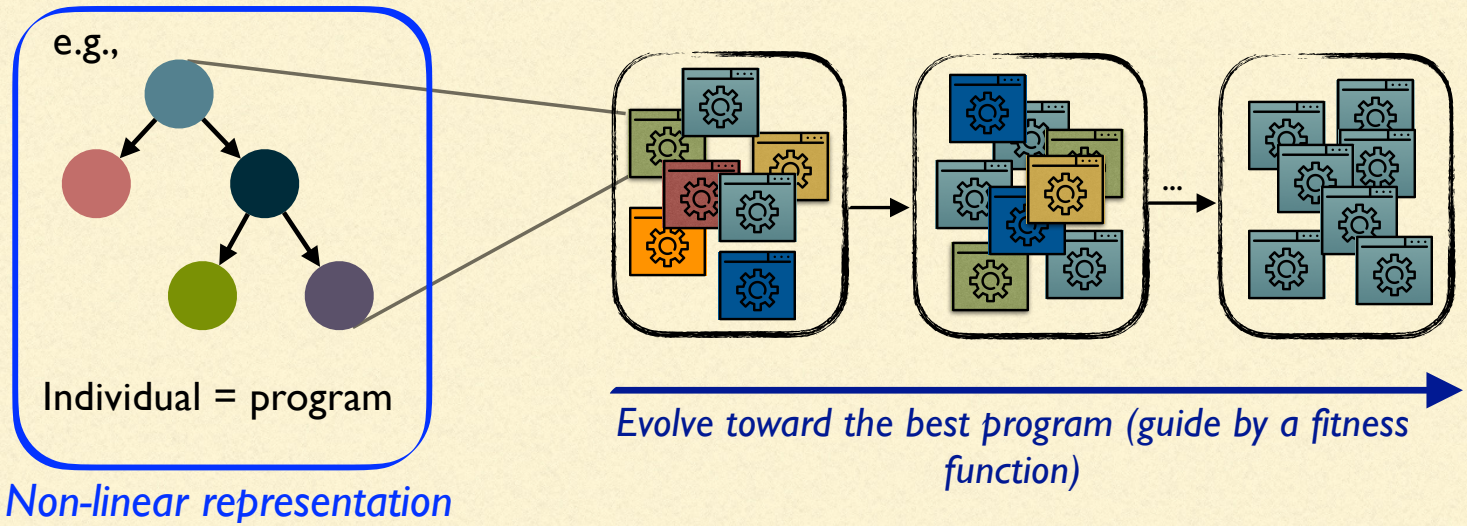


Genetic Algorithm: automatically evolve a good-enough solution for a given problem

Genetic Programming: automatically evolve a good-enough "*program*" for a given problem



GENETIC PROGRAMMING IS A SPECIFIC BRANCH OF GENETIC ALGORITHM



Non-linear representation

Theoretically, can generate any program with proper primitive operators, terminal and enough resources

→

Application in various domains: e.g., fault localisation, APR, etc.

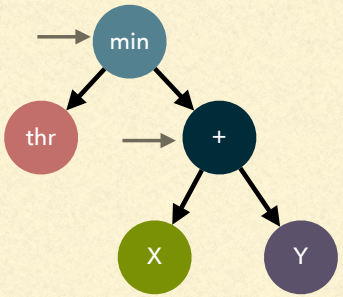
GP OVERFLOW: THE SAME AS GA

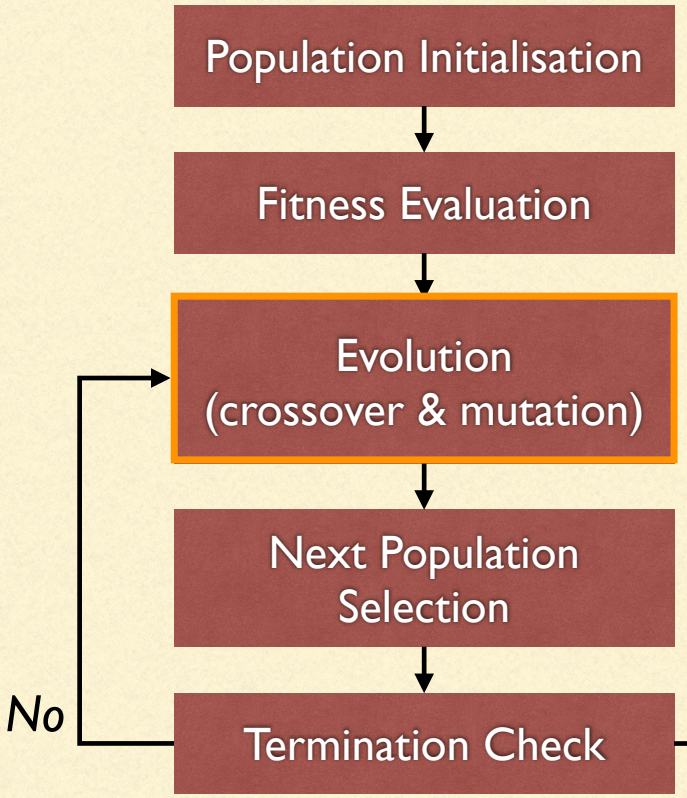
Define Primitives

Primitives:

- the smallest unit of processing available in a programming language

~= define internal nodes of a syntax tree





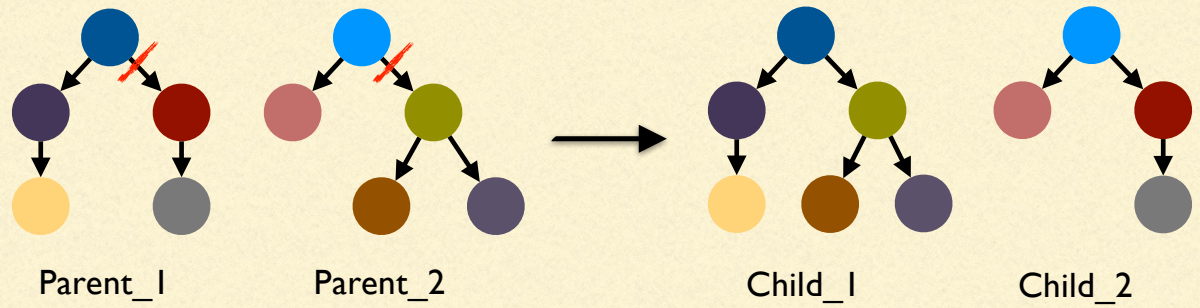
Goal. *(instead of finding an optimal solution)* **find a** ***"good-enough" solution "fast enough"*** **program**

GP: CROSSOVER AND MUTATION?

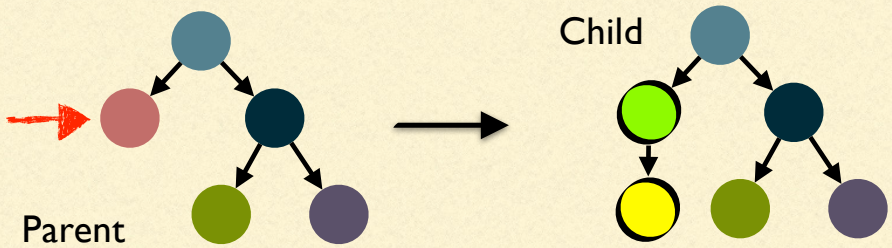
Evolution
(crossover & mutation)

- Crossover
- Mutation

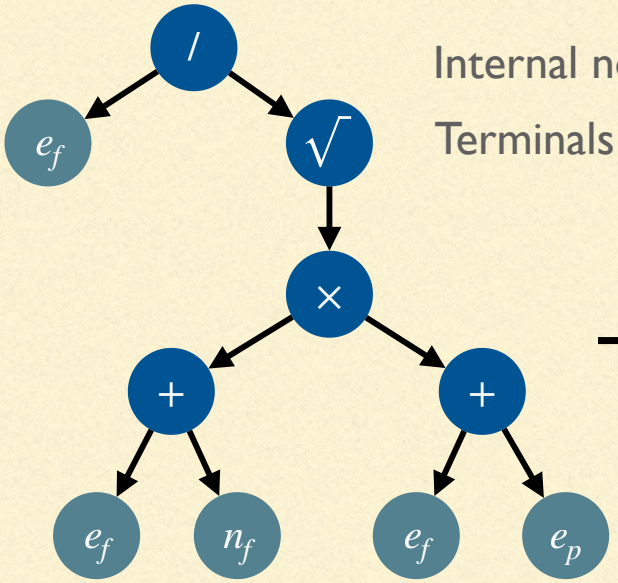
Crossover



Mutation



GENETIC PROGRAMMING: INDIVUDAL EVALUATION



Internal nodes = primitives / operators

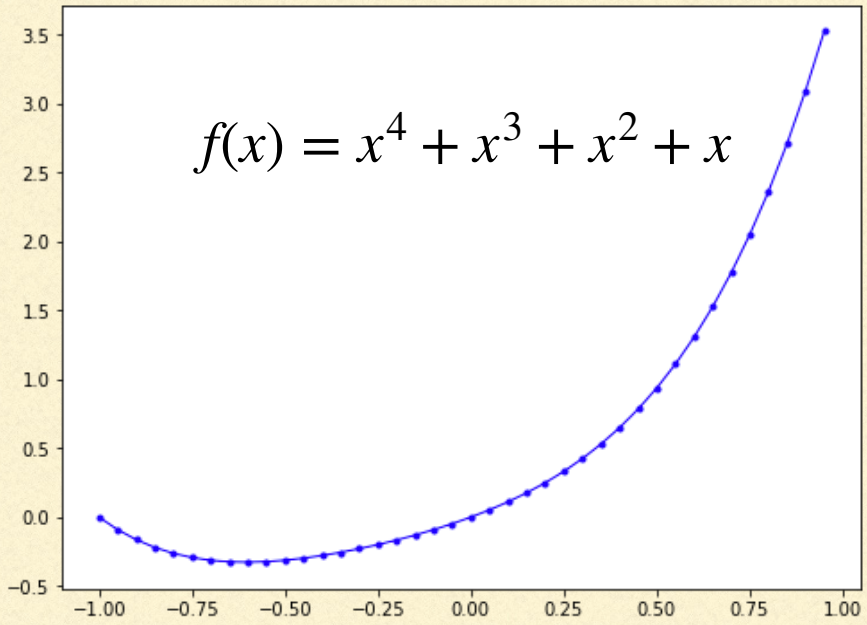
Terminals = operands

Compile

Output

→ assess/reflect the likelihood of being the optimal solution

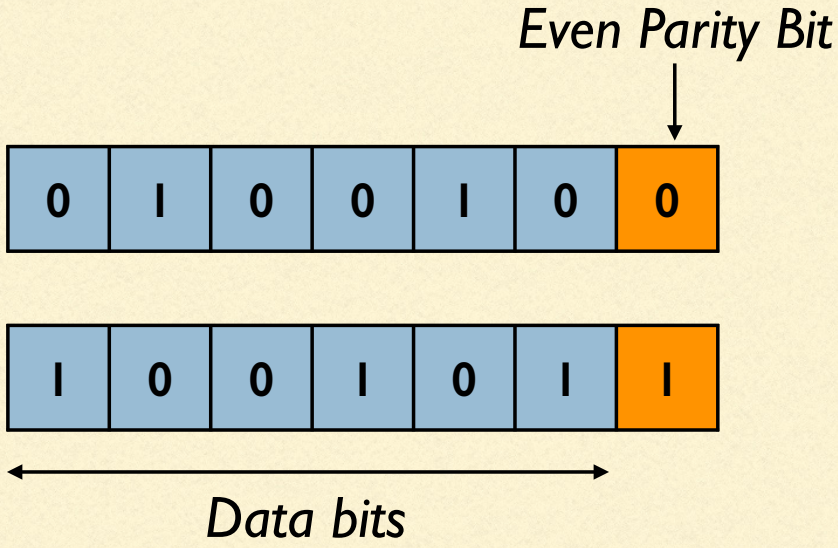
GP: SYMBOLIC REGRESSION



Symbolic Regression

- Individual: a candidate expression f'
- Fitness:
  - Minimise the mean squared error between $f'(x)$ and $f(x)$
- Crossover:
  - a single-point crossover (gp.cxOnepoint)
- Mutation:
  - a sinple-point replacement (gp.mutUniform)
- Primitives: add, multiply, subtract, negative, etc.

GP:EVEN PARITY GENERATOR (EXERCISE)

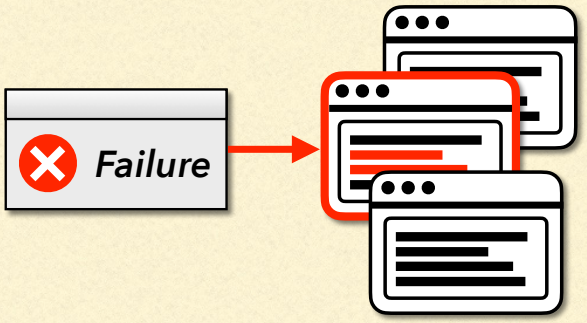


Even Parity Bit Generator

- Individual: a candidate expression G'
- Fitness:
  - Maximise the number of correctly computed even parity bits
  - … Or Minimise the number of wrongly computed even parity bits
- Crossover:
  - a single-point crossover (gp.cxOnepoint)
- Mutation:
  - shuffling (gp.mutUniform)

APPLICATION OF GP: SPECTRUM BASED FAULT LOCALISATION

Localise the fault that caused the observed failure



Spectrum-Based Fault Localisation (SBFL)

"The code is more likely to be faulty if it is executed by less passing test cases and more failing test cases"

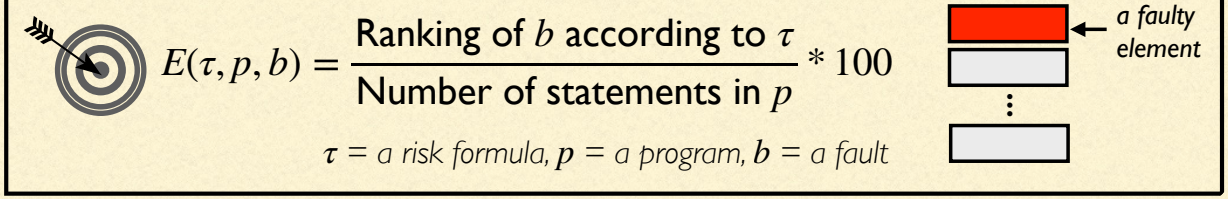
Risk evaluation formula → $\text{Ochiai} = \frac{e_f}{\sqrt{(e_f+f) \times (e_f + e_p)}}$

	t_1 (PASS)	t_2 (FAIL)	t_3 (PASS)	Spectrum e_p	Spectrum e_f	Spectrum n_p	Spectrum n_f	Ochiai	Rank
p_1	✓			1	0	1	1	0.00	4
p_2		✓		0	1	2	0	1.00	1
p_3		✓	✓	1	1	1	0	0.71	2
p_4	✓	✓	✓	2	1	0	0	0.58	3

[1] S. Yoo, X. Xie, F.-C. Kuo, T. Y. Chen, and M. Harman, "Human competitiveness of genetic programming in sbfl: Theoretical and empirical analysis," ACM Transactions on Software Engineering and Methodology, vol. 26, no. 1, pp. 4:1–4:30, July 2017.

APPLICATION OF GENETIC PROGRAMMING IN FAULT LOCALISATION

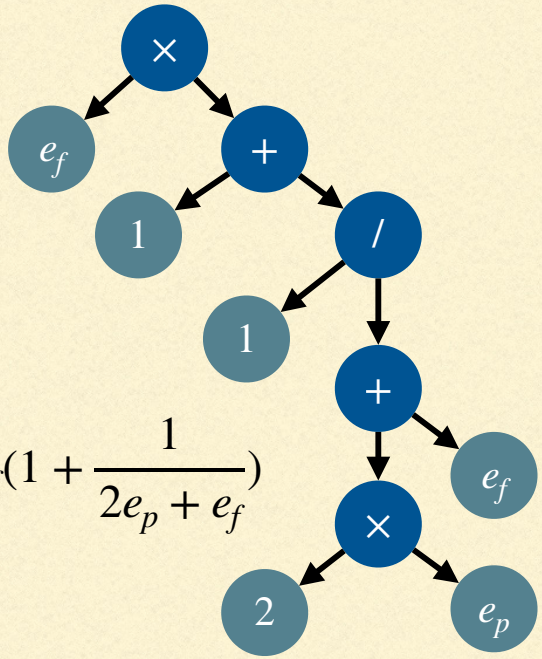
Shin Yoo. "Evolving Human Competitive Spectra-Based Fault Localisation Techniques", SSBSE'12



Minimise the average expense (E) for a set of bugs

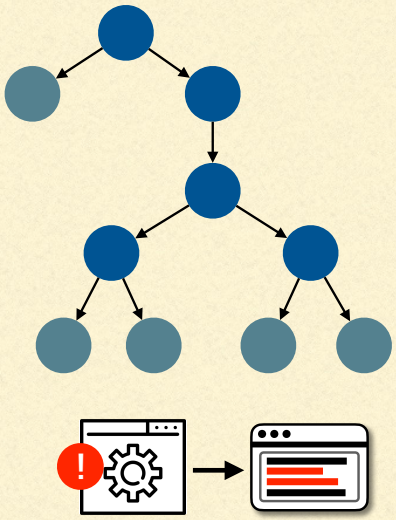
GP Operaters (Primitives):
addition, subtraction, multiplication, division, square root

Terminal Symbols: spectrum (e_p, e_f, n_p, n_f)

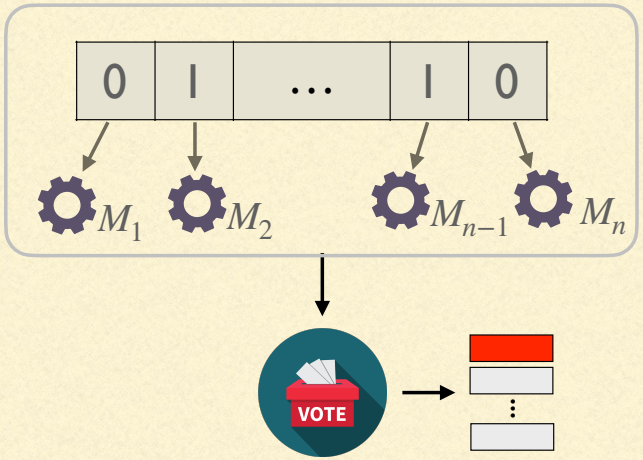


$$GP13 = e_f(1 + \frac{1}{2e_p + e_f})$$

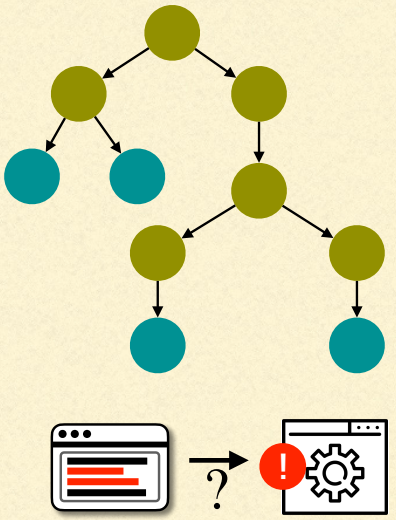
GENETIC PROGRAMMING AND GENETIC ALGORITHM IN MY RESEARCH



- GP as a learning algorithm to generate an effective FL model in FLUCCS



- GA to construct an effective candidate set of FL models for voting in EMF



- GP as a learning algorithm in Defect Prediction

SUMMARY

- As long as a problem can be re-defined as a search problem, Genetic Programming (GP) can be used to solve diverse software engineering problems.
- DEAP is an evolutionary computation framework with broad applicability that allows users to define their own types, initialisation methods, and algorithms.