Evolutionary Algorithms Q&A

Oana Cocarascu & Helen Yannakoudakis

Department of Informatics
King's College London

Genetic Algorithms (GAs)

- Hypothesise a population of candidate solutions to a particular problem.
- Evaluate each member of the population to decide how good that candidate is at solving the problem.
- Select those members of the population that are the best.
- Reproduce to obtain new members of the population.
- Repeat, hypothesising with this new set of candidate solutions.

GAs: The Basic Algorithm

GA(*Fitness*, *Fitness_threshold*, p, r, m)

Fitness: A function that assigns an evaluation score, given a hypothesis.
Fitness_threshold: A threshold specifying the termination criterion.
p: The number of hypotheses to be included in the population.
r: The fraction of the population to be replaced by Crossover at each step.
m: The mutation rate.

- *Initialize:* $P \leftarrow p$ random hypotheses
- *Evaluate:* for each h in P, compute $\mathit{Fitness}(h)$
- While $[\max_h \mathit{Fitness}(h)] < \mathit{Fitness\_threshold}$
  1. *Select*
  2. *Crossover*
  3. *Mutate*
  4. *Update*
  5. *Evaluate*
- Return the hypothesis from P that has the highest fitness.

GAs: The Basic Algorithm

GA(*Fitness*, *Fitness_threshold*, p, r, m)

- *Initialize:* $P \leftarrow p$ random hypotheses
- *Evaluate:* for each h in P, compute $Fitness(h)$
- While $[\max_h Fitness(h)] < Fitness\_threshold$
  1. *Select:* Probabilistically select $(1 - r)p$ members of P to add to P_S.

  $$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$$

  2. *Crossover:* Probabilistically select $\frac{r \cdot p}{2}$ pairs of hypotheses from P. For each pair, $\langle h_1, h_2 \rangle$, produce two offspring by applying the Crossover operator. Add all offspring to P_s.
  3. *Mutate:* Choose m percent of the members of P_s, with uniform probability. For each, invert one randomly selected bit in its representation.
  4. *Update:* $P \leftarrow P_s$.
  5. *Evaluate:* for each h in P, compute $Fitness(h)$.
- Return the hypothesis from P that has the highest fitness.

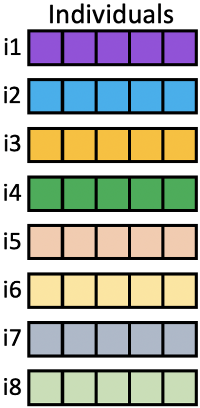
GAs

- In classic algorithm, where population size = N
  - Crossover rate = r, where $0 \leqslant r \leqslant 1$
  - Mutation rate = m, where $0 \leqslant m \leqslant 1$
- New population is generated with two steps:

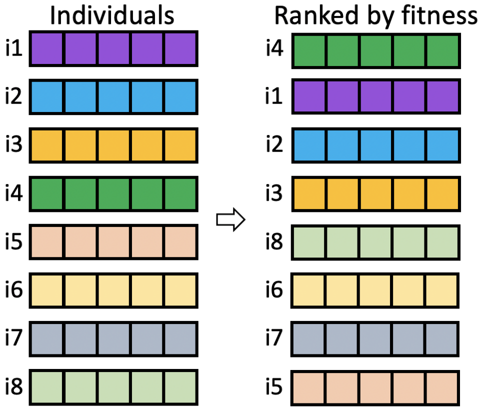
$$\text{first} \rightarrow \boxed{\begin{array}{c}(r \cdot N)\\ \text{crossed over}\end{array}} + \boxed{\begin{array}{c}((1-r)\cdot N)\\ \text{selected as is}\end{array}}$$

$$\text{second} \rightarrow \boxed{\begin{array}{c}(m \cdot N)\\ \text{mutated}\end{array}} + \boxed{\begin{array}{c}((1-m)\cdot N)\\ \text{selected as is}\end{array}}$$

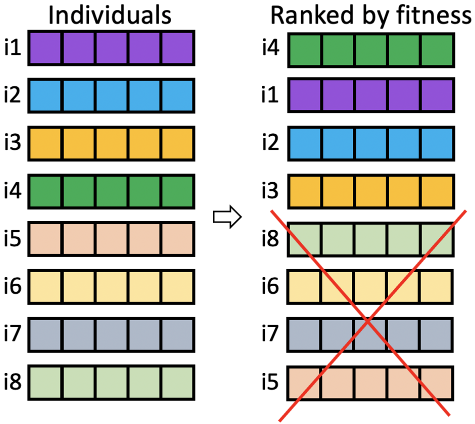
GAs: simplified view



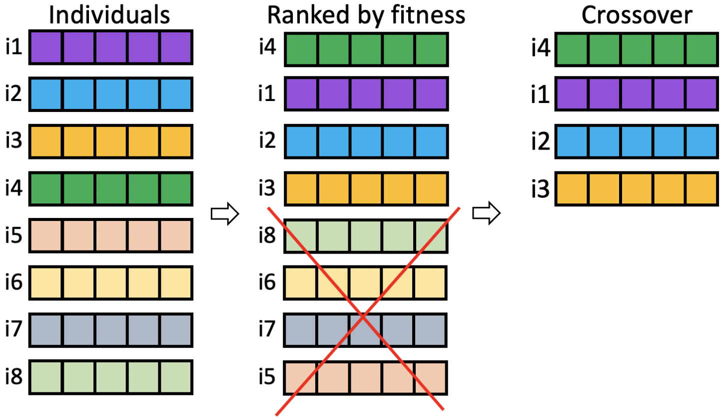
GAs: simplified view



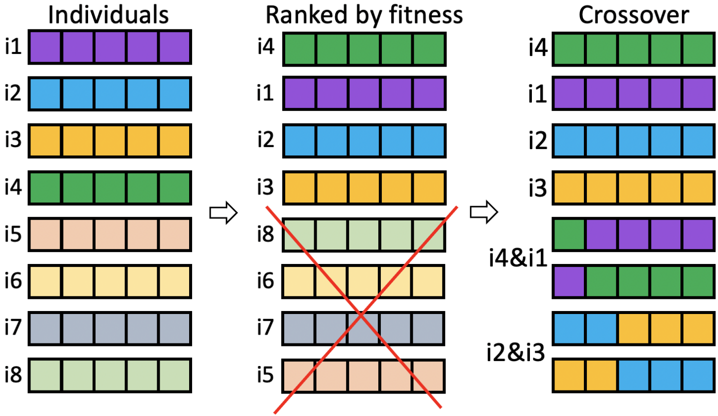
GAs: simplified view



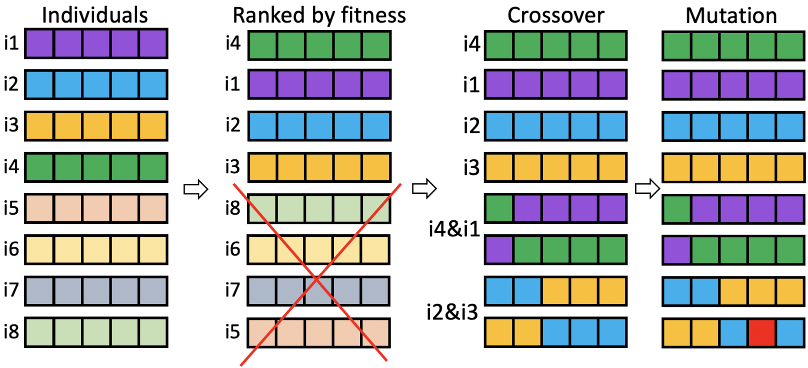
GAs: simplified view



GAs: simplified view



GAs: simplified view



Domain dependent vs domain independent

- Representation is domain ...
- Fitness is domain ...
- Selection methods are domain ...
- Fixed threshold value is domain ...

Domain dependent vs domain independent

- Representation is domain dependent.
- Fitness is domain dependent.
- Selection methods are domain independent.
- Fixed threshold value is domain dependent.

GAs: Fitness proportionate selection

- Fitness proportionate selection (roulette wheel selection):

$$Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{N} Fitness(h_j)}$$

GAs: Fitness proportionate selection

- Fitness proportionate selection (roulette wheel selection):

$$Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{N} Fitness(h_j)}$$

- Example:

$$Fitness(h_1) = 3$$
$$Fitness(h_2) = 2$$
$$Fitness(h_3) = 1$$
$$Fitness(h_4) = 4$$

KING'S College LONDON

GAs: Fitness proportionate selection

- Fitness proportionate selection (roulette wheel selection):

$$Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{N} Fitness(h_j)}$$

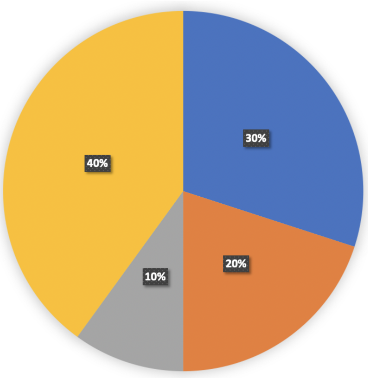
- Example:

$$\begin{aligned}
Fitness(h_1) &= 3 \Rightarrow Pr(h_1) = 0.3 \\
Fitness(h_2) &= 2 \Rightarrow Pr(h_2) = 0.2 \\
Fitness(h_3) &= 1 \Rightarrow Pr(h_3) = 0.1 \\
Fitness(h_4) &= 4 \Rightarrow Pr(h_4) = 0.4
\end{aligned}$$

KING'S College LONDON

GAs: Fitness proportionate selection

- $Pr(h_1) = 0.3, Pr(h_2) = 0.2, Pr(h_3) = 0.1, Pr(h_4) = 0.4$
- Individuals with a high fitness will have a larger proportion of the circle.

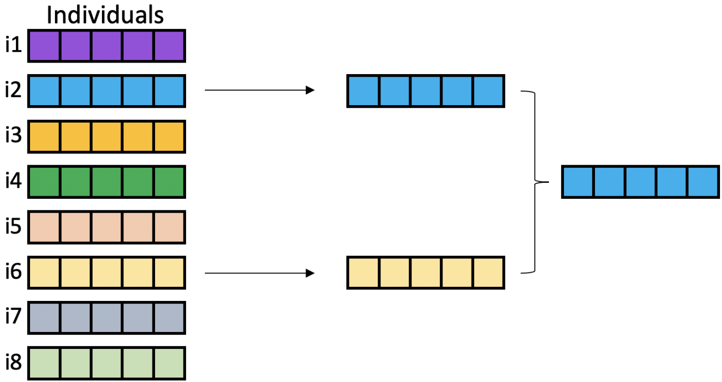


GAs: Tournament selection

- Randomly select two candidates from the population.
- Keep the candidate with the higher fitness.

GAs: Tournament selection

- Randomly select two candidates from the population.
- Keep the candidate with the higher fitness.



GAs: Rank selection

- Sort candidates and rank them according to their fitness.
- The probability that a hypothesis will be selected is proportional to its rank in the sorted list.

GAs: Rank selection

- Sort candidates and rank them according to their fitness.
- The probability that a hypothesis will be selected is proportional to its rank in the sorted list.
- Example:

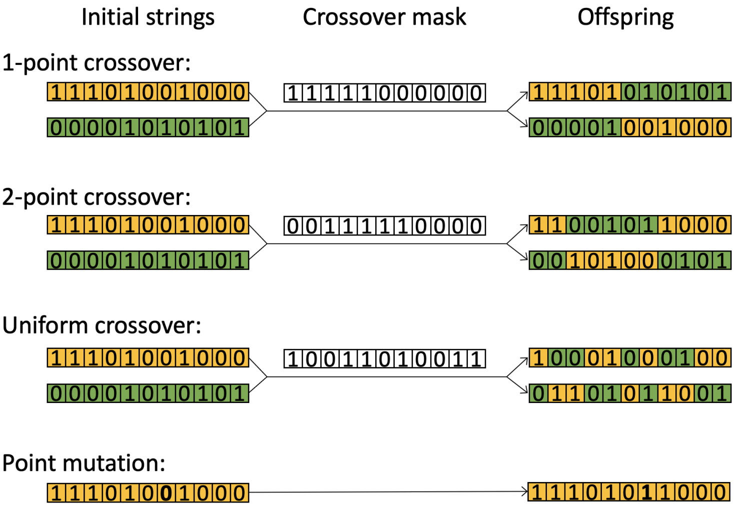
$$
\begin{aligned}
Fitness(h_1) &= 7.5 \\
Fitness(h_2) &= 6.2 \\
Fitness(h_3) &= 9.4 \\
Fitness(h_4) &= 5.1
\end{aligned}
$$

GAs: Rank selection

- Sort candidates and rank them according to their fitness.
- The probability that a hypothesis will be selected is proportional to its rank in the sorted list.
- Example:

$$
\begin{aligned}
Fitness(h_1) &= 7.5 \Rightarrow Rank(h_1) = 3 \\
Fitness(h_2) &= 6.2 \Rightarrow Rank(h_2) = 2 \\
Fitness(h_3) &= 9.4 \Rightarrow Rank(h_3) = 4 \\
Fitness(h_4) &= 5.1 \Rightarrow Rank(h_4) = 1
\end{aligned}
$$

GAs: Crossover and mutation



Function with one parameter

Maximise $f(x) = x^2, 0 \leqslant x \leqslant 31$

Function with one parameter

$$\text{Maximise } f(x) = x^2, 0 \leqslant x \leqslant 31$$

- How can we represent this?

Function with one parameter

$$\text{Maximise } f(x) = x^2, 0 \leqslant x \leqslant 31$$

- How can we represent this?
  Representation: 5 bits ($11111 \to 31$)

Function with one parameter

$$\text{Maximise } f(x) = x^2, 0 \leqslant x \leqslant 31$$

- How can we represent this?

  Representation: 5 bits (11111 $\rightarrow$ 31)
- What is the fitness function?

Function with one parameter

$$\text{Maximise } f(x) = x^2, 0 \leqslant x \leqslant 31$$

- How can we represent this?

  Representation: 5 bits ($11111 \rightarrow 31$)
- What is the fitness function?

  Fitness function: $f(x)$

Function with one parameter

$$\text{Maximise } f(x) = x^2, 0 \leqslant x \leqslant 31$$

- Representation: 5 bits ($11111 \rightarrow 31$).
- Fitness function: $f(x)$
- Example of initial population where the population size is 4:

x
13
24
8
19

Function with one parameter

$$\text{Maximise } f(x) = x^2, 0 \leqslant x \leqslant 31$$

- Representation: 5 bits ($11111 \rightarrow 31$)
- Fitness function: $f(x)$
- Example of initial population where the population size is 4:

x	Individual h
13	01101
24	11000
8	01000
19	10011

KING'S College LONDON

Function with one parameter

$$\text{Maximise } f(x) = x^2, 0 \leqslant x \leqslant 31$$

- Representation: 5 bits ($11111 \rightarrow 31$)
- Fitness function: $f(x)$
- Example of initial population where the population size is 4:

x	Individual h	$f(x)$
13	01101	169
24	11000	576
8	01000	64
19	10011	361
	sum	1170

KING'S College LONDON

Function with one parameter

$$\text{Maximise } f(x) = x^2, 0 \leqslant x \leqslant 31$$

- Representation: 5 bits ($11111 \rightarrow 31$)
- Fitness function: $f(x)$
- Example of initial population where the population size is 4:

x	Individual h	$f(x)$	$Pr(h)$
13	01101	169	0.14
24	11000	576	0.49
8	01000	64	0.06
19	10011	361	0.31
	sum	1170	

Function with one parameter

- Selection

  Roulette wheel which gives: #1, #2, #2, #4

#	Initial	x	$f(x)$	
1	01101	13	169	01101
2	11000	24	576	11000
3	01000	8	64	11000
4	10011	19	361	10011
	sum	1170		

Function with one parameter

- Selection

  Roulette wheel which gives: $\#1$, $\#2$, $\#2$, $\#4$
- Reproduction operators

  Crossover

  Mutation

#	Initial	x	$f(x)$
1	01101	13	169
2	11000	24	576
3	01000	8	64
4	10011	19	361
	sum	1170	

0110|1
1100|0
11|000
10|011

Function with one parameter

- Selection

  Roulette wheel which gives: #1, #2, #2, #4
- Reproduction operators

  Crossover

  Mutation

#	Initial	x	$f(x)$
1	01101	13	169
2	11000	24	576
3	01000	8	64
4	10011	19	361
	sum	1170	

	Crossover
0110|1	01100
1100|0	11001
11|000	11011
10|011	10000

Function with one parameter

- Selection

  Roulette wheel which gives: #1, #2, #2, #4

- Reproduction operators

  Crossover

  Mutation

#	Initial	x	$f(x)$
1	01101	13	169
2	11000	24	576
3	01000	8	64
4	10011	19	361
	sum	1170	

	Crossover	Mutation
0110|1	01100	01100
1100|0	11001	11001
11|000	11011	11011
10|011	10000	10010

Function with one parameter

- Selection

  Roulette wheel which gives: $\#1$, $\#2$, $\#2$, $\#4$
- Reproduction operators

  Crossover

  Mutation

x	$f(x)$
13	169
24	576
8	64
19	361
sum	1170

	Crossover	Mutation	x	$f(x)$
0110|1	01100	01100	12	144
1100|0	11001	11001	25	625
11|000	11011	11011	27	729
10|011	10000	10010	18	324
			sum	1822

Function with one parameter

- Selection

  Roulette wheel which gives: $\#1$, $\#2$, $\#2$, $\#4$
- Reproduction operators

  Crossover

  Mutation

x	$f(x)$
13	169
24	576
8	64
19	361
sum	1170

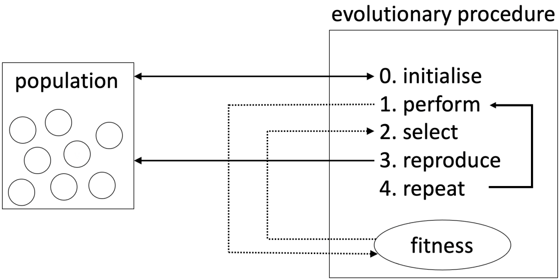
	Crossover	Mutation	x	$f(x)$
0110|1	01100	01100	12	144
1100|0	11001	11001	25	625
11|000	11011	11011	27	729
10|011	10000	10010	18	324
			sum	1822

- The total fitness is now 1822 which is higher than the fitness of the initial population.
- The population has improved.

KING'S College LONDON

Co-evolution

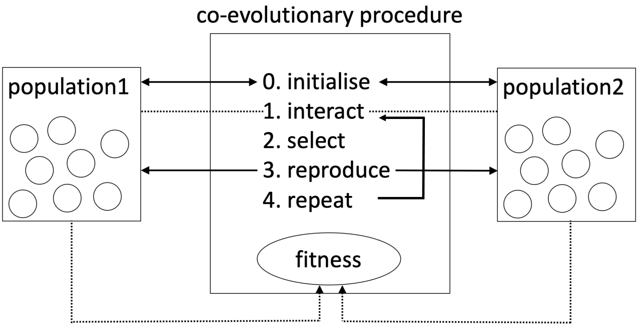
- With traditional evolutionary algorithms, we have one evolving population.
- The success of candidate solutions in the population is measured against a fixed fitness function and a fitness threshold. You evaluate each individual in the population and you want to obtain a better population.



Co-evolution

- In several situations when you want to obtain solutions, what is best depends on what the others are doing.
- When there is no easy way to define a fixed fitness function or one simply does not exist, we can use co-evolution (still need a way to compare two solutions and determine which one is better between them).
- With co-evolution, there are two evolving populations. The fitness of a candidate solution is measured by comparing members of opposing populations.

Co-evolutionary Learning



- Game playing is a classic example, where two candidate solutions play games against each other and the candidate that wins more games is declared the fittest.

Co-evolution

Co-evolution

- Remember that in traditional evolutionary algorithms, the success of candidate solutions in the population is measured against a fixed fitness function and a fitness threshold.
- Difficult (for a new player) to learn to play well if you're not playing against a reasonably competent player.
- The score you obtain on your own is not necessary a good measure of fitness but playing against an opponent (and beating them) is a good measure of fitness.

Co-evolution

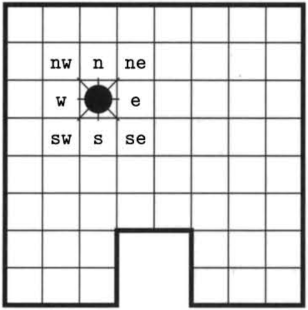
- In co-evolution, there are two populations.
- The first/second population tries to adapt to the environment consisting of the second/first population.
- We test the performance of each individual in the first/second population against each individual from the second/first population.
- The relative fitness of an individual in co-evolution is not an absolute measure of fitness against an optimal opponent, but a relative measure when the individual is tested against the current opposing population.
- Some individuals in each population are better at dealing with the current opposing population.

Wall-following robot

- Build the program up from four actions (North, East, South, West) and four primitive functions:

```
AND(x, y) = 0 if x = 0; else y
OR(x, y) = 1 if x = 1; else y
NOT(x) = 0 if x = 1; else 1
IF(x, y, z) = y if x = 1; else z
```

- As input we have the robot's current sensory data (value 0 if the corresponding cell is free, otherwise value 1).



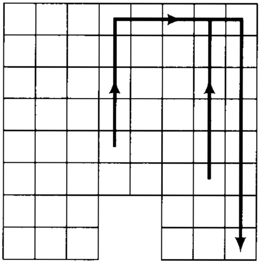
Wall-following robot

- Build the program up from four actions (North, East, South, West) and four primitive functions:

```
AND(x, y) = 0 if x = 0; else y
OR(x, y) = 1 if x = 1; else y
NOT(x) = 0 if x = 1; else 1
IF(x, y, z) = y if x = 1; else z
```

- Example programs:
  - `(AND (sw) (ne))` evaluates the first argument (`sw`), terminates if result is `0`, otherwise it evaluates the second argument and terminates.
  - If the program `(OR (e) (w))` evaluates west, then it moves west and terminates.

Wall-following robot



```
(NOT (AND (IF (ne)
               (IF (se)(south)(east))
               (north))
          (NOT (NOT (e)))))
```

`IF(x, y, z) = y if x = 1; else z`

- `ne=0` then move north; . . .
- `ne=1` then evaluate `(IF (se) (south) (east))`
- `se=0` then move east; . . .
- `se=1` then move south; . . .