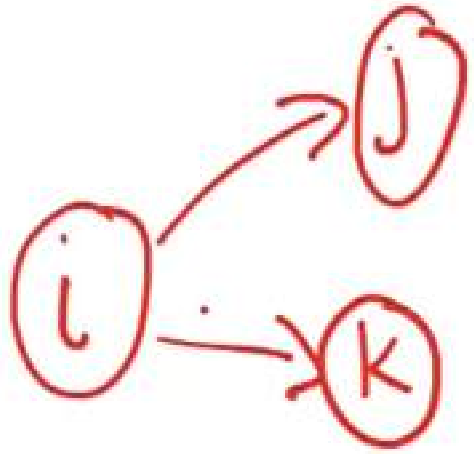
Adjacency Representation: Crossover Operators

- Alternating Edges Crossover
  - Construct a child as follows
  - From a given city A choose the next city B from P_1
  - From the city B choose the next city from P_2 ?
  - … and so on



- Heuristic Crossover
  - For each city choose the from that parent (P_1 or P_2) which is closer

Adjacency representation facilitates these choices

TSP: Order Crossover

C_2	O	G	L	H	M	A	D	K	I	C	B	J	F	N	E

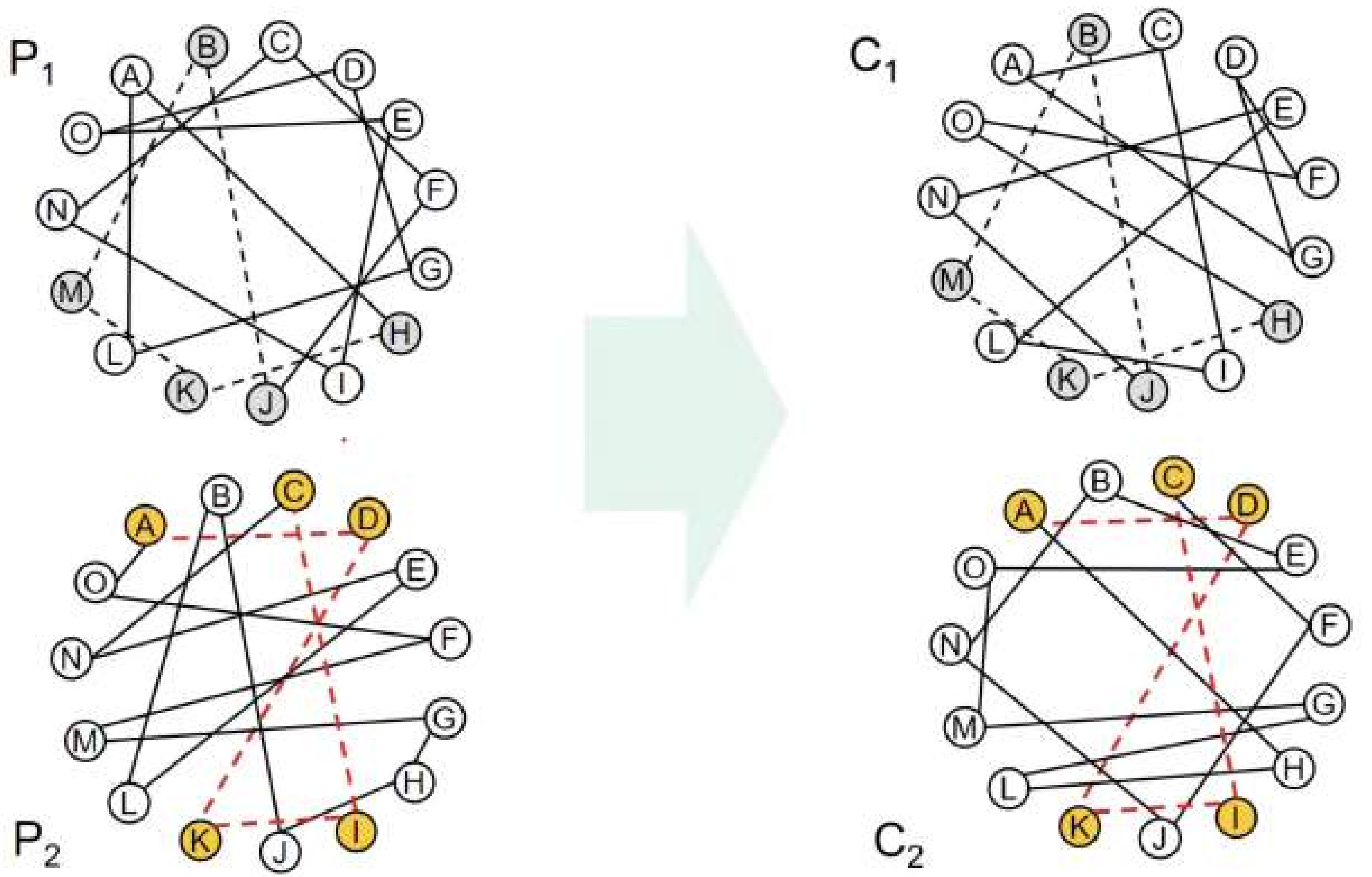
The second child C_2 is constructed in a similar manner, first copying the subtour from P_2

P_1	O	D	G	L	A	H	K	M	B	J	F	C	N	I	E
P_2	H	G	M	F	O	A	D	K	I	C	N	E	L	B	J
C_1	G	F	O	A	D	H	K	M	B	J	I	C	N	E	L

Copy a subtour from P_1 into C_1 and the remaining from P_2 in the order they occur in P_2.

TSP: PMX

Dashed edges show copied subtour



TSP: Partially Mapped Crossover (PMX)

C_1

A	G	D	F	O	H	K	M	B	J	N	E	L	I	C

The second child C_2 is constructed in a similar manner, first copying the subtour from P_2

P_1

O	D	G	L	A	H	K	M	B	J	F	C	N	I	E

P_2

H	G	M	F	O	A	D	K	I	C	N	E	L	B	J

Likewise for cities I and C

Remember that city K is already in C_1…

Copy the remaining cities directly from P_2

C_2

O	M	G	L	H	A	D	K	I	C	F	J	N	B	E

TSP: Partially Mapped Crossover (PMX)

NPTEL

C_1	A					H	K	M	B	J					
P_1	O	D	G	L	A	H	K	M	B	J	F	C	N	I	E
P_2	H	G	M	F	O	A	D	K	I	C	N	E	L	B	J

Copy subtour from P_1 into C_1

Where should city A be in C_1?

Follow the partial map…

TSP: Partially Mapped Crossover (PMX)

C_1						H	K	M	B	J					
P_1	O	D	G	L	A	H	K	M	B	J	F	C	N	I	E
P_2	H	G	M	F	O	A	D	K	I	C	N	E	L	B	J

Copy subtour from P_1 into C_1

Choose a subtour and establish a mapping

Would like to copy remaining cities from P_2
but
the locations for cities A, D, I, C are occupied
by cities H, K, B,J respectively

TSP: Partially Mapped Crossover (PMX)

P_1	O	D	G	L	A	H	K	M	B	J	F	C	N	I	E
P_2	H	G	M	F	O	A	D	K	I	C	N	E	L	B	J

Choose a subtour and establish a mapping

TSP: Cycle Crossover

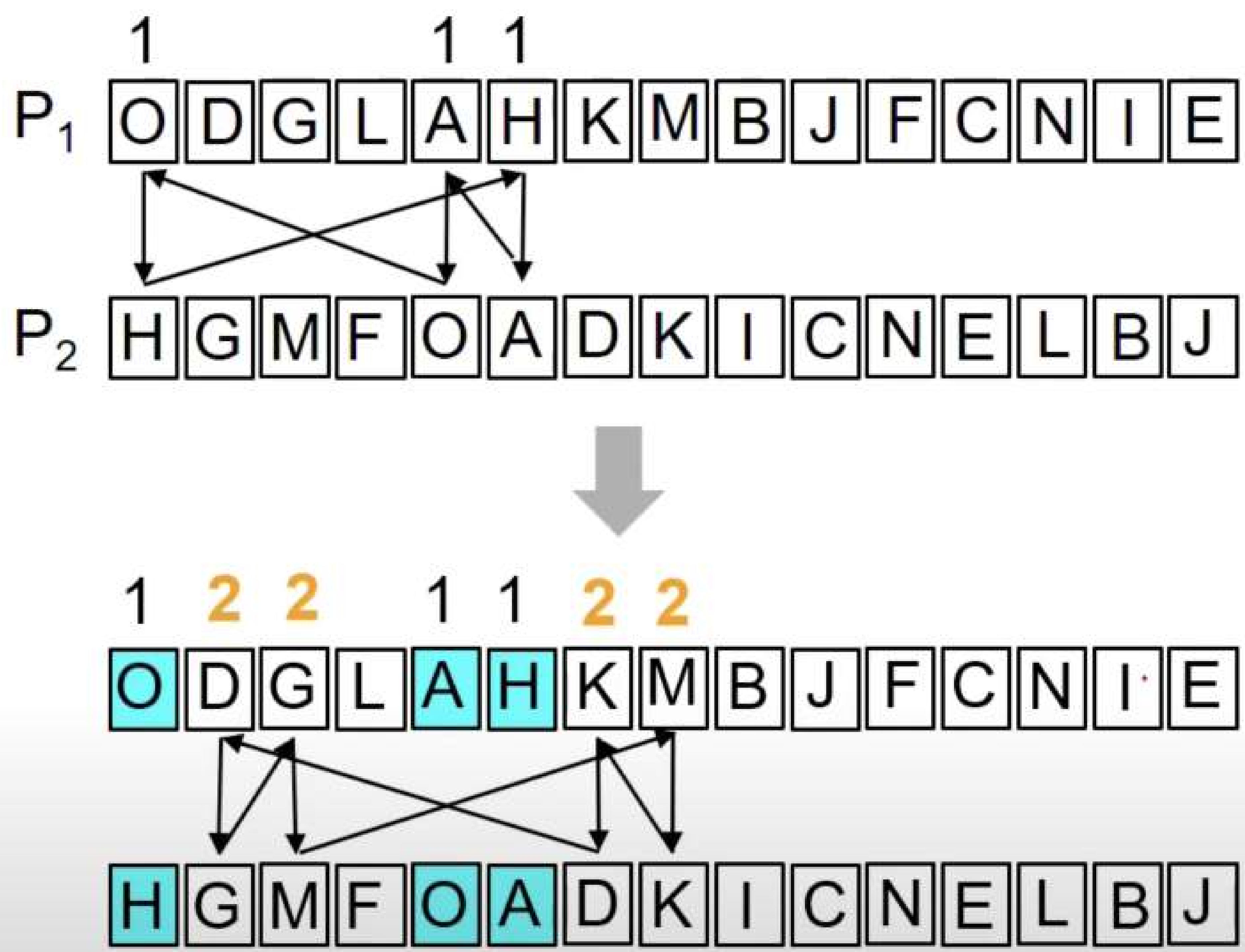
C_1	O	G	M	L	A	H	D	K	I	J	F	C	N	B	E
	O			L	A	H				J	F	C	N		E
P_1	O	D	G	L	A	H	K	M	B	J	F	C	N	I	E
	1	2	2	3	1	1	2	2	4	5	3	5	3	4	5
P_2	H	G	M	F	O	A	D	K	I	C	N	E	L	B	J

C_1 gets even numbered cycles from P_2

C_1 gets odd numbered cycles from P_1

6:36 / 29:54

TSP: Cycle Crossover



Step 1: Identify cycles

TSP: Cycle Crossover

P_1	O	D	G	L	A	H	K	M	B	J	F	C	N	I	E
	1	2	2	3	1	1	2	2	4	5	3	5	3	4	5
P_2	H	G	M	F	O	A	D	K	I	C	N	E	L	B	J

TSP: Single point crossover does not work

O	D	G	L	A	H	K	M	B	J	F	C	N	I	E
H	G	M	F	O	A	D	K	I	C	N	E	L	B	J

↓

O	D	G	L	A	H	K	M	B	J	N	E	L	B	J
H	G	M	F	O	A	D	K	I	C	F	C	N	I	E

Both offspring are not valid tours

GAs for TSP

NPTEL

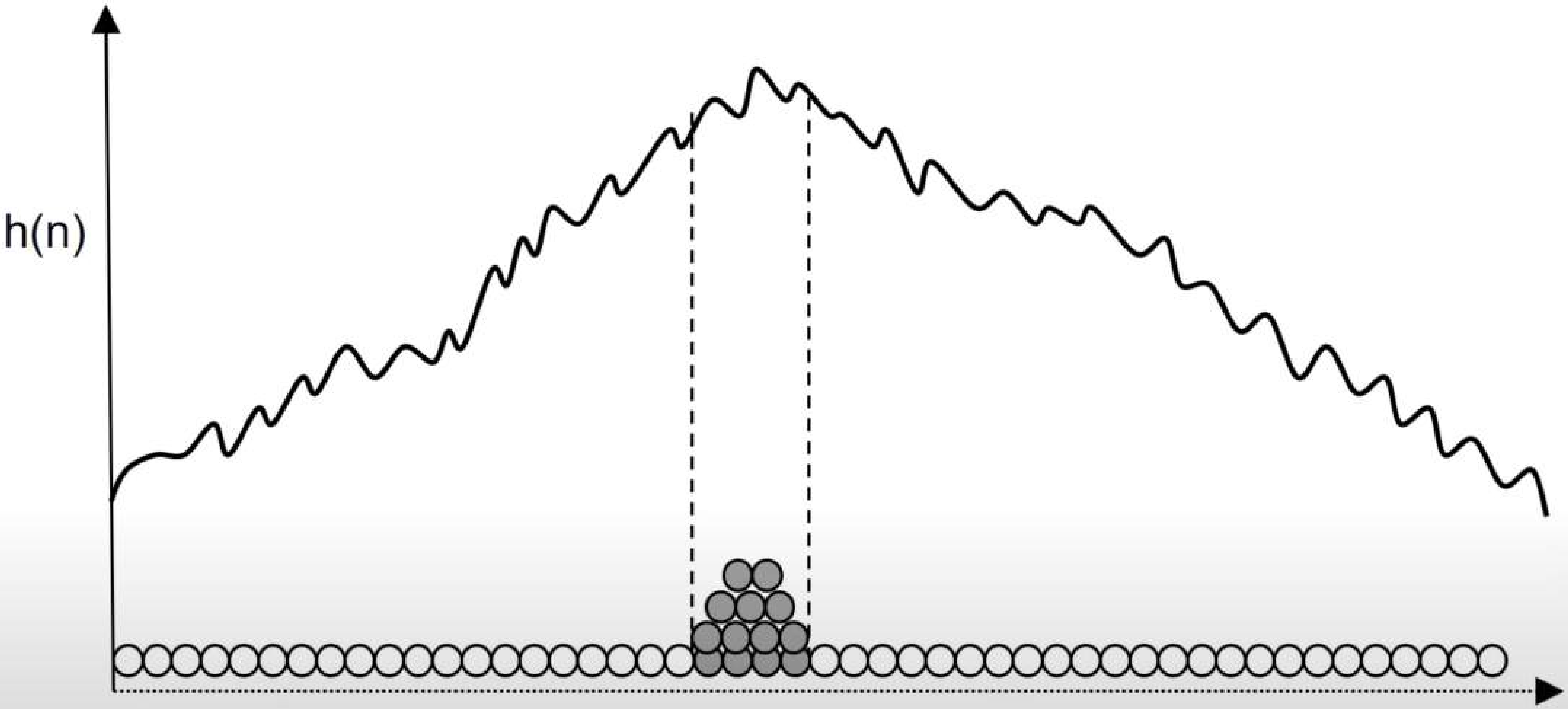
Genetic Algorithms can be used for the TSP problem as follows –

Create a population of candidate TSP solutions. Let the fitness function be the cost of the tour. It is a *minimization problem*.

In the *Path Representation* the tour is represented by a permutation of the cities, with the assumption that one returns from the last city in the permutation to the first.

Selection:	Clone each tour in proportion to fitness. The cheapest tours are the fittest.
Crossover:	Randomly pair the resulting population and perform crossover.
Mutation:	Randomly permute a tour once in a while

The population may become less diversified



able to adapt and that is a totally
different aspect of

A Tiny Example: Cycle 2

NPTEL

Crossed-over	Binary	f(x)	Prob.	Expected	Actual
01100	12	144	0.08	0.33	0
11001	25	625	0.36	1.42	2
11011	27	729	0.42	1.66	2
10000	16	256	0.16	0.58	0

Total 1754
Avg. 493

and third candidates and let's assume
that this is what really

A Tiny Example: Single Point Crossover

Selected	Crossed-over	Binary	f(x)
01101	01100	12	144
11000	11001	25	625
11000	11011	27	729
10011	10000	16	256

crossover

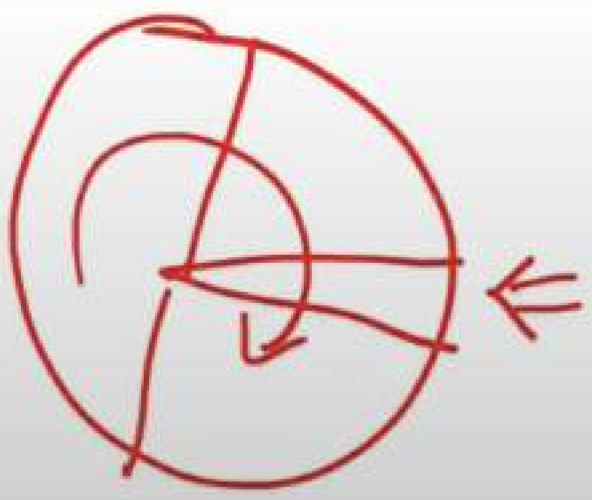
Fitter population

Total 1754
Avg. 493

we are interested in we get these values
144 625 729 and

A Tiny Example: Selection

Initial	Binary	f(x)	Prob.	Expected	Actual	Selected
01101	13	169	0.14	0.58	1	01101
11000	24	576	0.49	1.97	2	11000
01000	8	64	0.06	0.22	0	11000
10011	19	361	0.31	1.23	1	10011



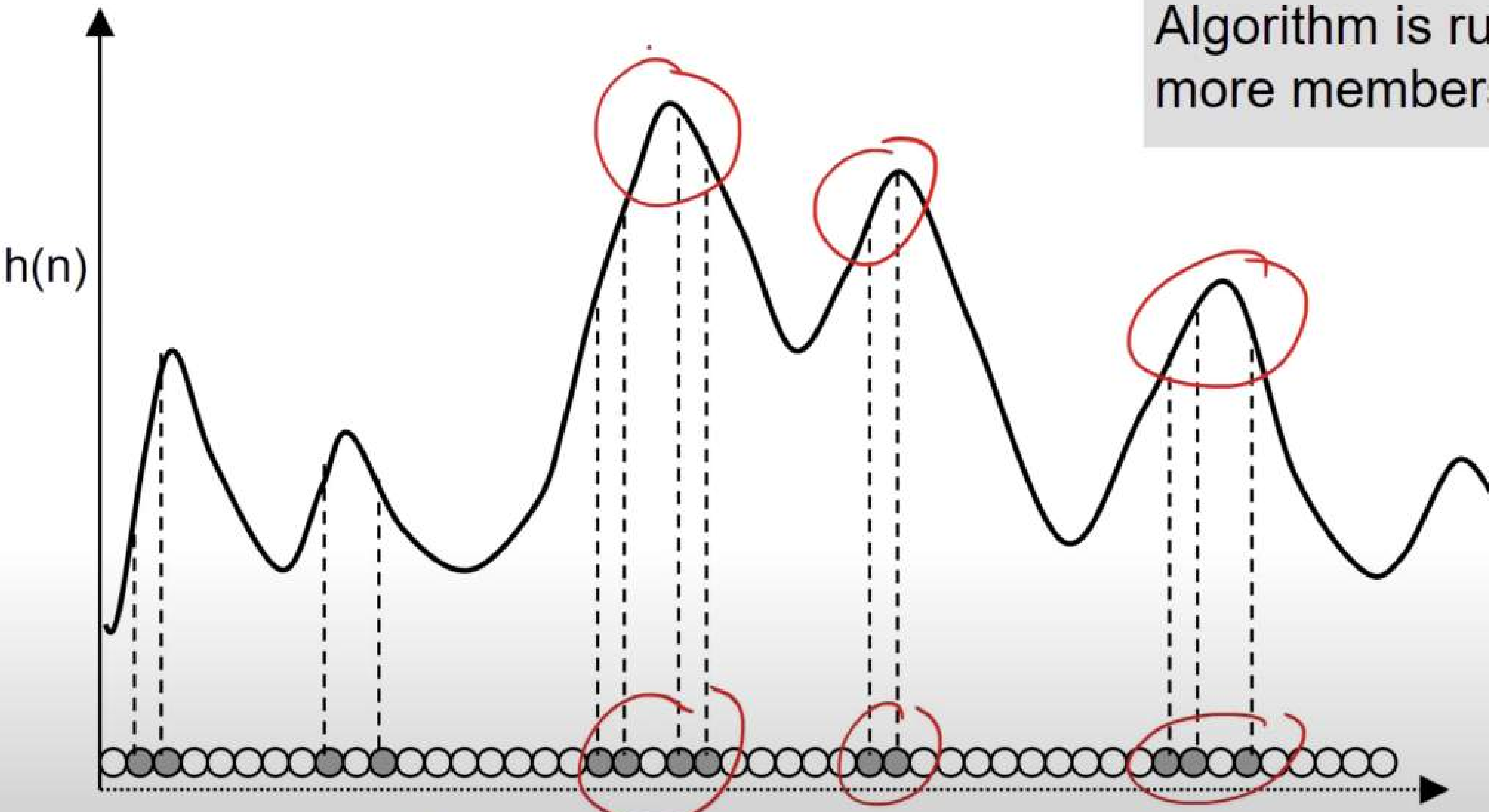
Total	1170
Avg.	293

fourth one has one so this is the select
we have

26:12 / 34:42

GAs: The evolved population

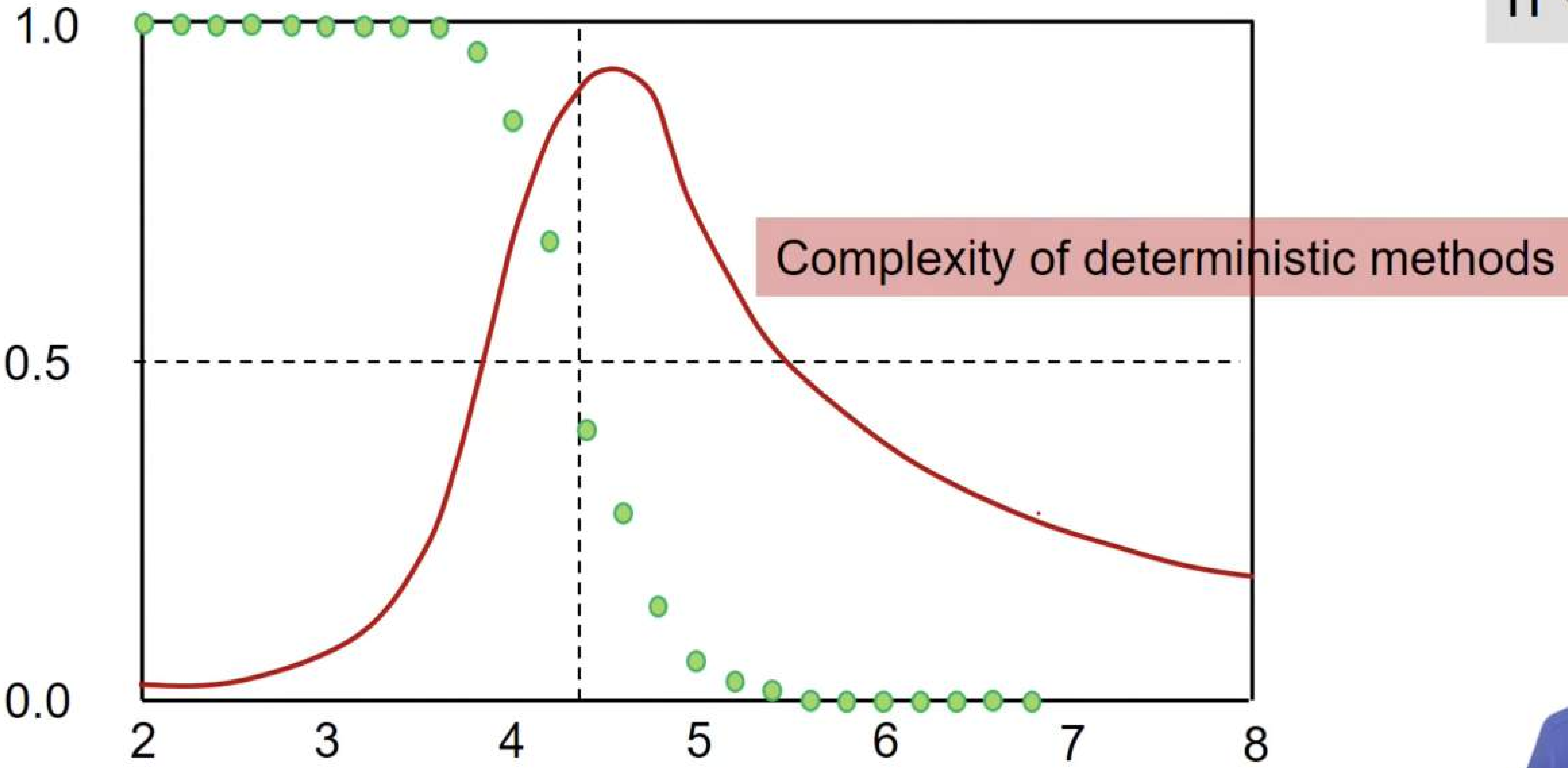
The Initial population may be randomly distributed, but as Genetic Algorithm is run the population has more members around the peaks.



and so on and in general we find that populations become as we

Probability of 3SAT being satisfiable

m clauses
n variables



$m/n = 4.3$

complexity of Set uh and in particular
we are talking about three

Genetic Algorithms and SAT

A SAT problem with 6 variables

$$F = (a \lor \neg b \lor e) \land (\neg a \lor b \lor c) \land (\neg c \lor d \lor f) \land (d \lor \neg e \lor f) \land (\neg a \lor b \lor d) \land (a \lor \neg d \lor f)$$

h = number of satisfied clauses

Single Point Crossover

P_1	010|110	0 + 1 + 1 + 1 + 1 + 0 = 4
P_2	111|010	1 + 1 + 0 + 0 + 1 + 1 = 4
C_1	111110	1 + 1 + 1 + 1 + 1 + 1 = 6
C_2	010010	0 + 1 + 1 + 0 + 1 + 1 = 4

and this means that we have actually found a solution to the S problem

17:03 / 34:42

The Algorithm

```
GENETIC-ALGORITHM()
P ← create N candidate solutions        ▷ initial population
repeat
    compute fitness value for each member of P
    S ← with probability proportional to fitness value,
        randomly select N members from P
    offspring ← partition S into two halves, and randomly mate
                and crossover members to generate N offsprings
    with a low probability mutate some offsprings
    replace k weakest members of P with k strongest offsprings
until some termination criteria
return the best member of P
```

ChatGPT
Explore GPTs
Today
SAT Problem Explanation
Kazirhut.com Inquiry
Yesterday
Call Option Price Proof
Economic Concepts Overview
Cross-Price Elasticity Explainec
Previous 7 Days
Arbitrage Proof Forward Contra
Forward Contract Positions
Wastewater Treatment Process
DFA NFA Equivalence
Sewage Treatment Summary
Minimal Cover FD
Update Anomalies in DB
Error Recovery in Parsers
Upgrade plan
More access to the best models

Key Differences

- **Local vs. Global Search**:
  - *Normal Hill Climbing*: Focuses on a single trajectory; once it reaches a local peak, it stops.
  - *Iterated Hill Climbing*: Incorporates multiple runs with perturbations, giving it a chance to explore beyond the local peak.
- **Escaping Local Optima**:
  - *Normal Hill Climbing*: Lacks a built-in mechanism to escape local optima.
  - *Iterated Hill Climbing*: Actively perturbs the solution to escape and continue the search, enhancing its ability to find the global optimum.
- **Robustness**:
  - *Normal Hill Climbing*: Can be effective in simple landscapes but often fails in complex, multimodal problems.
  - *Iterated Hill Climbing*: Generally more robust and effective in complex search spaces due to its iterative restart mechanism.

Iterated hill climbing essentially builds on the normal hill climbing algorithm by adding a strategy to overcome its inherent limitations, making it a more powerful tool for finding optimal solutions in challenging optimization problems.