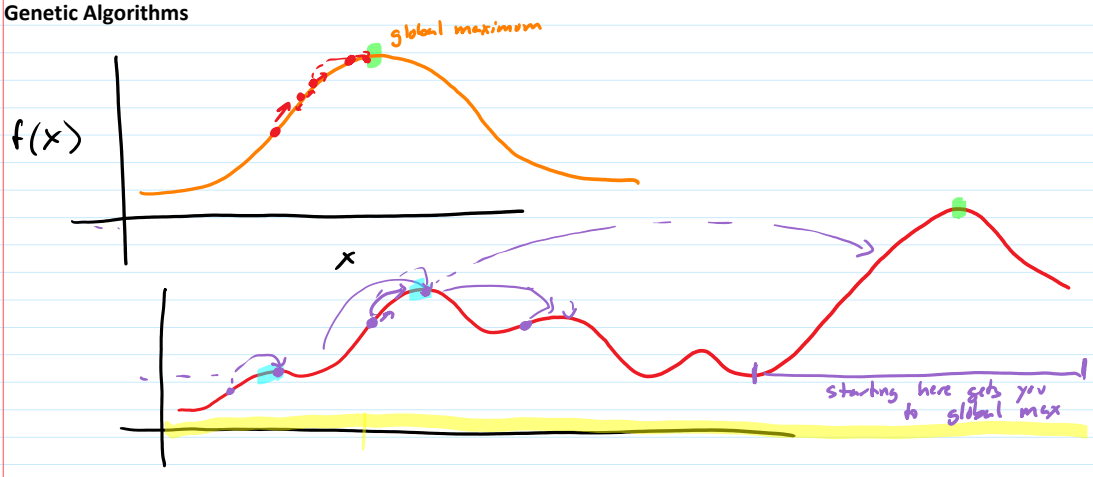
Genetic Algorithms



Evolutionary Computation
mimic natural selection — nature-inspired
Individuals have genes (numbers for numeric functions)
genes determine phenotype — inputs to function
phenotype contributes to fitness
fitness contributes to propagation

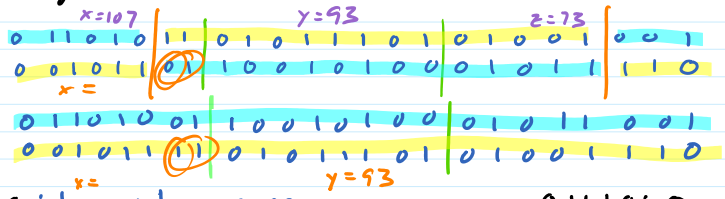
Genetic Algorithms

start with random population - collection of individuals with randomly selected genes (random 0's and 1's)
while not done (out of time, no improvement, found good enough solution)
  evaluate each individual — genes → inputs to f → evaluate f → fitness
  select for crossover — select pairs of individuals, biased towards higher fitness
  crossover — randomly select genes from each parent to create offspring genes
  select for survival — replacement or fitness-based
  mutate — randomly change genes in some individuals (typically low rate)

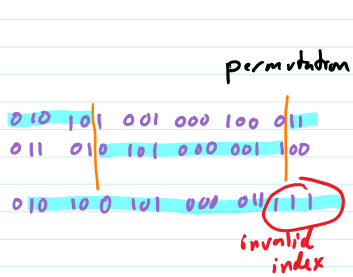
Evolution Strategy: 1 individual, no crossover, offspring are mutants / used if fitness fxn is expensive

Representation: what is genetic code?

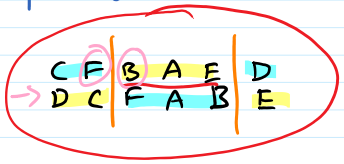
$f(x,y,z)$ $f: \mathbb{Z}\times\mathbb{Z}\times\mathbb{Z} \Rightarrow \mathbb{R}$

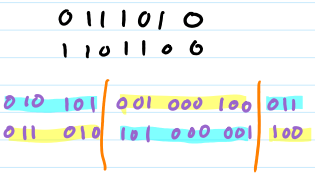


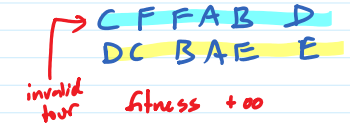
Crossover: bitstrings: two-point crossover

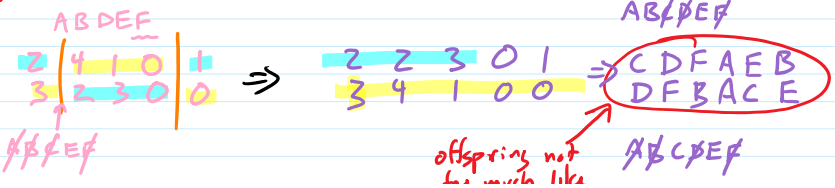


permutation









too much like parents

CFBAED
DCFABE

A	B	C	D	E	F
50	70	100	20	10	80
32	30	50	75	26	40
50	70	50	75	26	80

A	B	C	D	E	F
50	70	100	75	26	80
32	30	50	20	10	40

CFDBAE
CFABDE

C C F F C | B B F A C C E | A B
D C F C F | B C B B E A D | C F

⇒ C F B A E D
D C F B E A

C C F F C B C B B E A D A B ⇒ C F B E A D

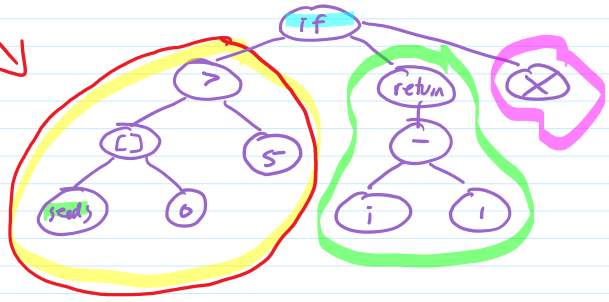
Genetic Programming:

```
if seeds[0] > 5
{
    many++
}
```

```
if seeds[i] == 0 and seeds[i - 1] == 1
{
    return i-1
}
```

if seeds[0][i-1] == 1
return i- ++

```
if seeds[0] > 5
{
return i - 1;
}
```



represent programs as syntax trees

crossover: swapping branches

Optimization

measure of strength of player

Given a function $f(x_1, \ldots, x_n)$, find $x_1, \ldots, x_n$ that yield max $f(x_1, \ldots, x_n)$

constants in heuristic

find x, y to maximize $4x + 5y - 2xy - x^2 - y^2$

NP-complete

find ordering of Seattle, Denver, Allentown, Baltimore, Amherst, Washington to minimize distance of corresponding tour

find assignment of classrooms to minimize student conflicts

find ordering of teams to minimize upsets during previous season

NP-complete

	A	B	C	H	W
A		W	W	L	W
B			W	L	W
C				W	W
H					W

X = beat ↑

W A C B H

Solitaire Yahtzee: Estimate start-of-turn position value by counting

3 for 1's open
6 for 2's
⋮
18 for 6's open
+6/open upper if on pace for bonus

20 for open 3 of a kind
10 4
15 for FH
⋮
10 for Yahtzee

Play each turn to maximize turn score + next position value using heuristic

Find parameters to maximize avg. score

0. David et al

Chess Heuristic: for each player, count

9	for queen
5	for rook
3	for bishop
3	for knight
1	for pawn
+1	for rook mobility
+1	for king protection
+3	for breakaway pawn
⋮	

optimization: find params of heuristic to maximize Elo

- 3400 comp
- 2800 top few
- 2500 grandmaster
- 1500 HS champ
- 700

Elo = numeric rating of strength

Find parameters to maximize Elo

Problems for GA: what is fitness?

performance against existing agents (hard to do, slow)
or
how well it matches moves (using database)

Step 1 mentor: how often individual's choice matches mentor's

Step 2 coevolution: play individuals against each other

often doesn't work well on random initial population

fitness = # wins in round-robin

(doesn't require external evaluator)