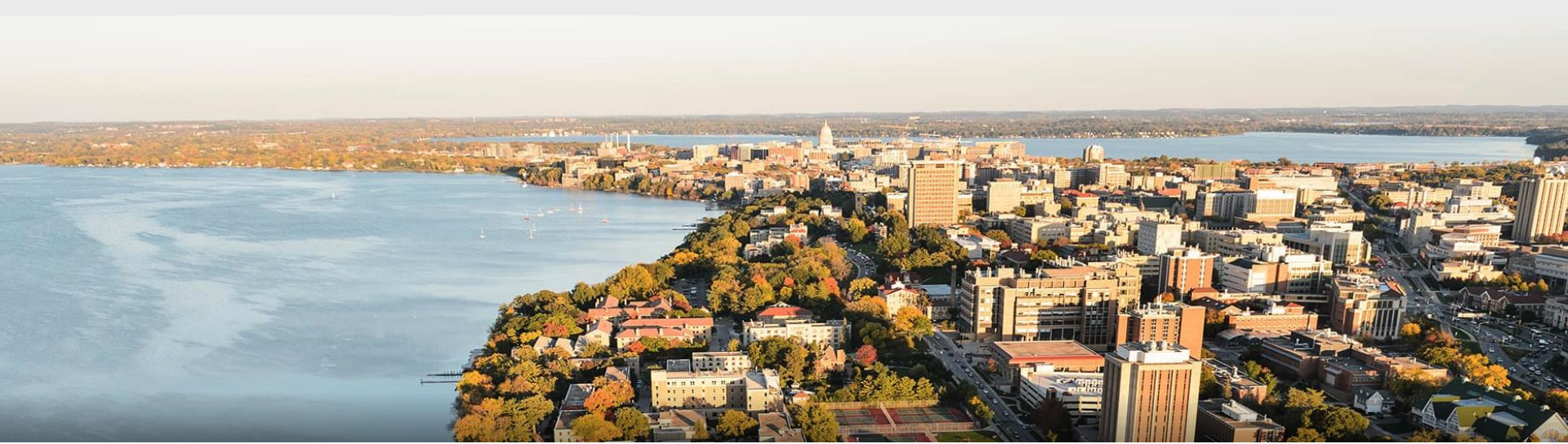


CS 540 Introduction to Artificial Intelligence

Search III: Optimization, continued

University of Wisconsin–Madison
December 3, 2025
Fall 2025

Announcements

- Homework:
  - HW10 due Dec 9 at 11:59 PM

- Final Exam
  - Saturday, Dec 13, 12:25-2:25 PM
  - CHEM S249

Advanced Search
Ethics and Trust in AI

HelioCampus Course Evaluations

- Please do them! Feedback used to design future versions of 540
- Final exam incentive
  - With >50% participation, instructors will release an “excluded topics” list for final exam
  - With >70%, more topics excluded
- Survey open until 12/10

Outline

- Review Optimization & Hill-climbing
  - More difficult problems, basics, local optima, variations
- Genetic Algorithms

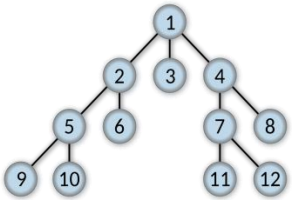
- Fill out course evaluations

REVIEW

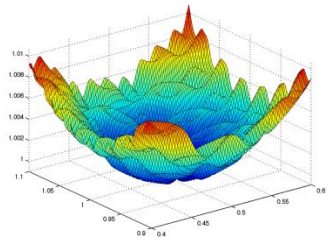
Search vs. Optimization

Before: wanted a **path** from start state to goal state

- Uninformed search, informed search



Wiki


TuringFin

New setting: optimization

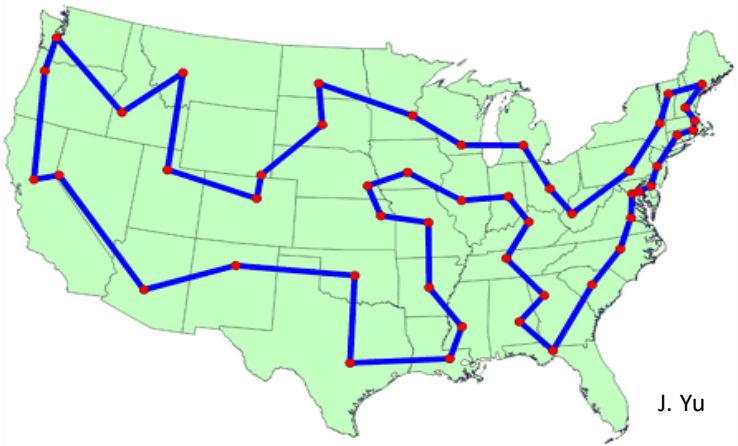
- States *s* have values ***f***(*s*)
- Want: Find *s* with optimal value ***f***(*s*) (i.e, **optimize** over states)
- Challenging settings: **too many states** for previous search approaches, but maybe not a differentiable function for gradient descent.

Examples: TSP

Famous graph theory problem.

- Get a graph G = (V,E). **Goal**: a path that visits each node exactly once and returns to the initial node (a **tour**).
  - State: a particular tour (i.e., ordered list of nodes)
  - $f(s)$: total weight of the tour

  (e.g., total miles traveled)



Examples: Satisfiability

Boolean satisfiability (e.g., 3-SAT)

- Recall our logic lecture. Conjunctive normal form

$$(A \vee \neg B \vee C) \wedge (\neg A \vee C \vee D) \wedge (B \vee D \vee \neg E) \wedge (\neg C \vee \neg D \vee \neg E) \wedge (\neg A \vee \neg C \vee E)$$

  - Goal: find if satisfactory assignment exists.
  - State: assignment to variables
  - ***f***(*s*): # satisfied clauses

$$R(x,a,d) \wedge R(y,b,d) \wedge R(a,b,e) \wedge R(c,d,f) \wedge R(z,c,0)$$

$$\begin{array}{l}
R(0,a,d) \wedge R(0,b,d) \wedge R(a,b,e) \wedge R(c,d,f) \wedge R(0,c,0) \\
R(0,a,d) \wedge R(0,b,d) \wedge R(a,b,e) \wedge R(c,d,f) \wedge R(1,c,0) \\
R(0,a,d) \wedge R(1,b,d) \wedge R(a,b,e) \wedge R(c,d,f) \wedge R(0,c,0) \\
R(0,a,d) \wedge R(1,b,d) \wedge R(a,b,e) \wedge R(c,d,f) \wedge R(1,c,0) \\
R(1,a,d) \wedge R(0,b,d) \wedge R(a,b,e) \wedge R(c,d,f) \wedge R(0,c,0) \\
R(1,a,d) \wedge R(0,b,d) \wedge R(a,b,e) \wedge R(c,d,f) \wedge R(1,c,0) \\
R(1,a,d) \wedge R(1,b,d) \wedge R(a,b,e) \wedge R(c,d,f) \wedge R(0,c,0) \\
R(1,a,d) \wedge R(1,b,d) \wedge R(a,b,e) \wedge R(c,d,f) \wedge R(1,c,0)
\end{array}$$

$$R(\neg x,a,b) \wedge R(b,y,c) \wedge R(c,d,\neg z)$$

$$\begin{array}{l}
R(\ 1,a,b) \wedge R(b,0,c) \wedge R(c,d,\ 1) \\
R(\ 1,a,b) \wedge R(b,0,c) \wedge R(c,d,\ 0) \\
R(\ 1,a,b) \wedge R(b,1,c) \wedge R(c,d,\ 1) \\
R(\ 1,a,b) \wedge R(b,1,c) \wedge R(c,d,\ 0) \\
R(\ 0,a,b) \wedge R(b,0,c) \wedge R(c,d,\ 1) \\
R(\ 0,a,b) \wedge R(b,0,c) \wedge R(c,d,\ 0) \\
R(\ 0,a,b) \wedge R(b,1,c) \wedge R(c,d,\ 1) \\
R(\ 0,a,b) \wedge R(b,1,c) \wedge R(c,d,\ 0)
\end{array}$$

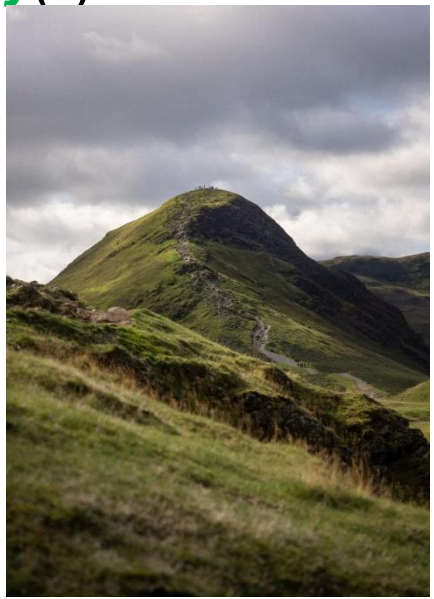
Wiki

Hill Climbing

One approach to such optimization problems

- Basic idea: start at one state, move to a neighbor with a better $f(s)$ value, repeat until no neighbors have better $f(s)$ value.

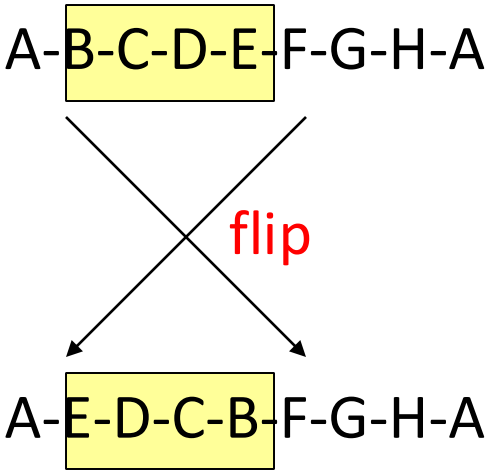
- **Q**: how do we define **neighbor**?
  - Not as obvious as our successors in search
  - Problem-specific
  - As we'll see, needs a careful choice

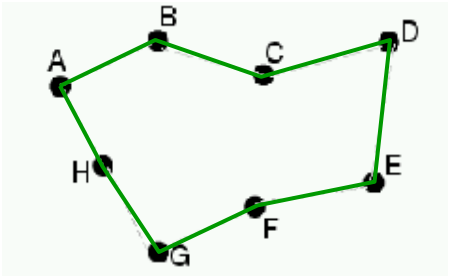


Defining Neighbors: TSP

Define neighbors by small changes

- Example: 2-change: A-E and B-F





Defining Neighbors: SAT

For Boolean satisfiability,

- Define neighbors by flipping one assignment of one variable

Starting state: (A=T, B=F, C=T, D=T, F=T)

(A=**F**, B=F, C=T, D=T, E=T)
(A=T, B=**T**, C=T, D=T, E=T)
(A=T, B=F, C=**F**, D=T, E=T)
(A=T, B=F, C=T, D=**F**, E=T)
(A=T, B=F, C=T, D=T, E=**F**)

$A \vee \neg B \vee C$
$\neg A \vee C \vee D$
$B \vee D \vee \neg E$
$\neg C \vee \neg D \vee \neg E$
$\neg A \vee \neg C \vee E$

Evolution & Choosing Neighbors

- Neighboring state ⇔ genetic mutation
- What happens if mutation rate is very high?
  - ie, neighbors very different from current state
- What happens if mutation rate is very low?
  - ie, neighbors very similar to current state

Hill Climbing Algorithm

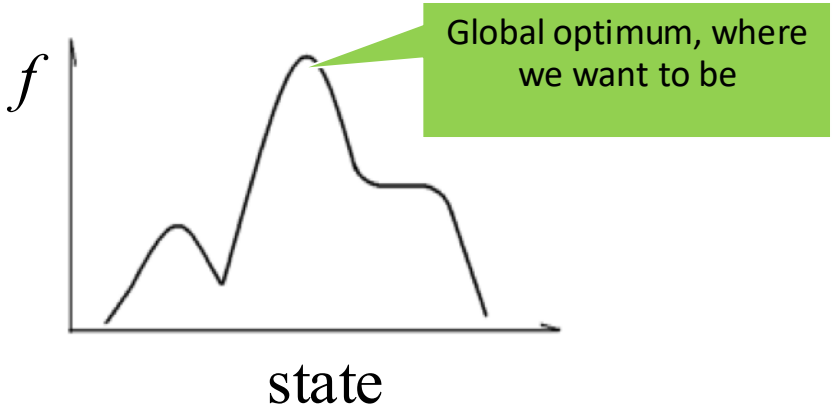
Pseudocode:

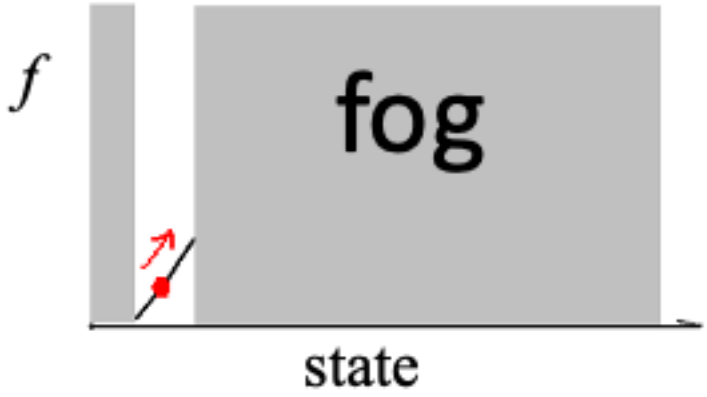
1. Pick initial state *s*
2. Pick *t* in **neighbors**(*s*) with the best ***f***(*t*)
3. if ***f***(*t*) is not better than ***f***(*s*) THEN stop, return *s*
4. *s* ← *t*. goto 2.

What could happen? **Local optima!**

Hill Climbing: Local Optima

Q: Why is it called hill climbing?





L: What's actually going on.

R: What we get to see.

Escaping Local Optima

Simple idea 1: random restarts

- Stuck: pick a random new starting point, re-run.
- Do *k* times, return best of the *k* runs.

Simple idea 2: reduce greed

- "Stochastic" hill climbing: randomly select between neighbors.
- Probability of selecting a neighbor should be proportional to the value of that neighbor.

Simulated Annealing

- Algorithm slowly increases greed over time

- Pseudocode

Pick initial state s; T=1
For k = 0 through K:
    $T \leftarrow T*0.99$ *(cool down)*
    Pick a random neighbor $t \leftarrow$ neighbor(s)
    If $f(t)$ better than $f(s)$, then $s \leftarrow t$
    Else with prob. $P(f(s), f(t), \boldsymbol{T})$ still do $s \leftarrow t$
Output: the best state ever seen

The interesting bit

Simulated Annealing: Picking Probability

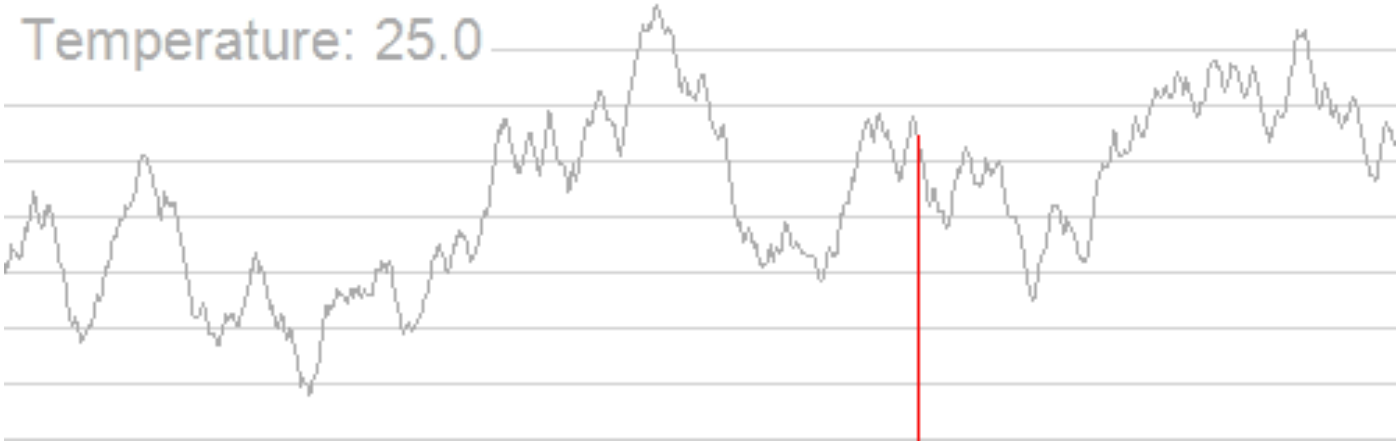
How do we pick probability P? Note 3 parameters.

- Decrease with time
- Decrease with gap $|f(s) - f(t)|$: $\exp\left(-\frac{|f(s)-f(t)|}{Temp}\right)$

- Temperature cools over time.
  - So: high temperature, accept any *t*
  - But, low temperature, behaves like hill-climbing
  - Still, $|f(s) - f(t)|$ plays a role: if big, replacement probability low.

Simulated Annealing: Visualization

What does it look like in practice?



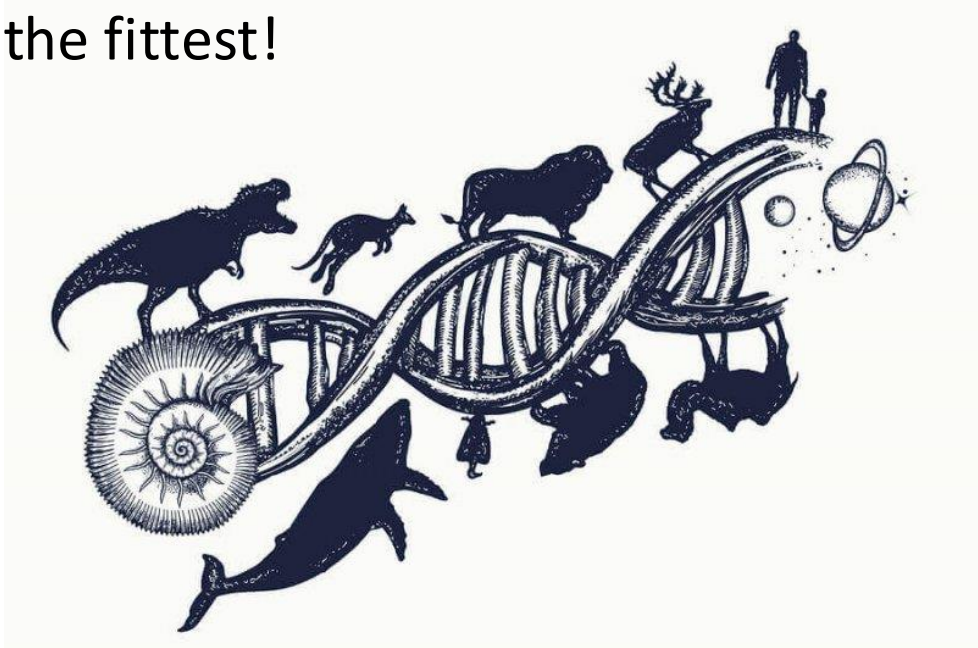
Wiki

GENETIC ALGORITHMS

Genetic Algorithms

Optimization approach based on nature

- Survival of the fittest!

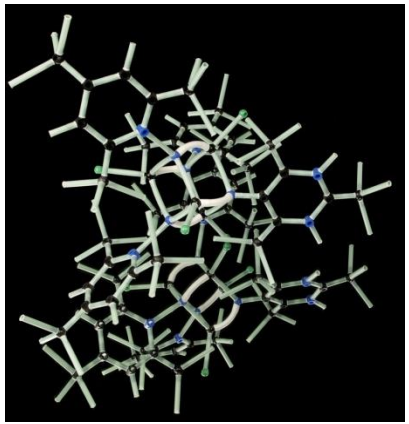


Evolution Review

Encode genetic information in DNA (four bases)

- A/C/T/G: nucleobases acting as symbols

- Two types of changes
  - Crossover: exchange between parents' codes
  - Mutation: rarer random process
    - Happens at individual level



Natural Selection

Competition for resources

- Organisms with better fitness ➔ better probability of reproducing
- Repeated process: fit become larger proportion of population

Goal: use these principles for optimization

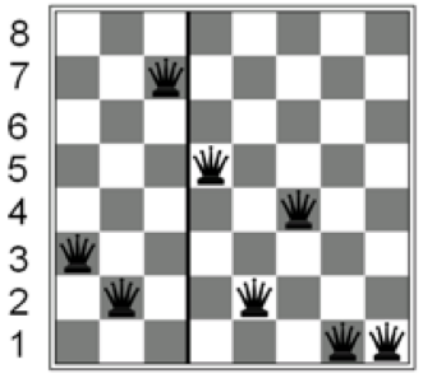
- New terminology: state is '**individual**'
- Value $f(s)$ is now the '**fitness**'

Genetic Algorithms Setup I

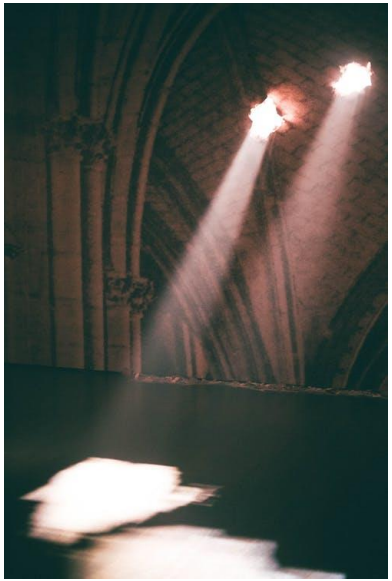
Keep around a fixed number of states/individuals

- Call this the **population**

For our n Queens game example, an individual:



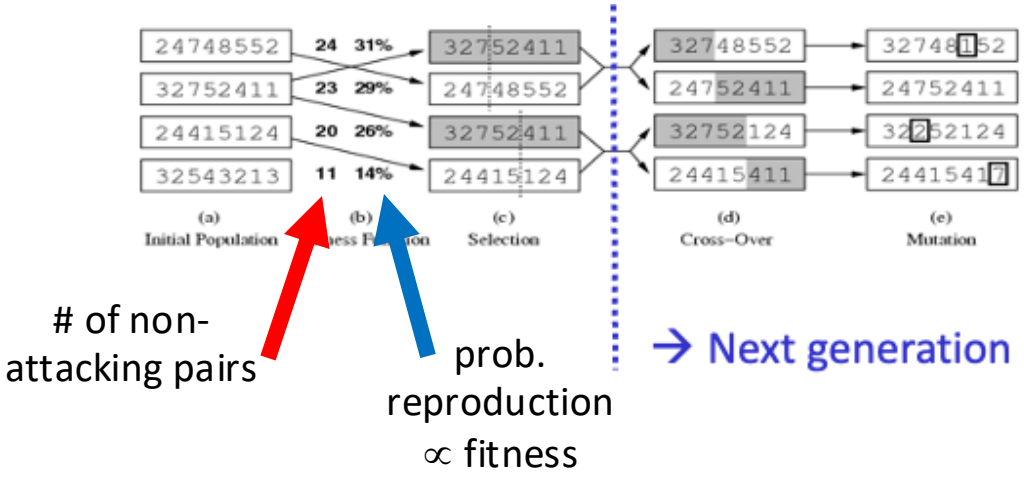
(3 2 7 5 2 4 1 1)



Genetic Algorithms Setup II

Goal of genetic algorithms: optimize using principles inspired by mechanism for evolution

- Analogous to **natural selection, cross-over**, and **mutation**



Genetic Algorithms Pseudocode

Just one variant:

1. Let $s_1, \ldots, s_N$ be the current population
2. Let $p_i = \boldsymbol{f}(s_i) / \Sigma_j \boldsymbol{f}(s_j)$ be the reproduction probability
3. for $k = 1$; $k<N$; k+=2
   - parent1 = randomly pick according to p
   - parent2 = randomly pick another
   - randomly select a crossover point, swap strings of parents 1, 2 to generate children $t[k]$, $t[k+1]$
4. for $k = 1$; $k<=N$; k++
   - Randomly mutate each position in $t[k]$ with a small probability (mutation rate)
5. The new generation replaces the old: $\{ s \} \leftarrow \{ t \}$. Repeat

Reproduction: Proportional Selection

Reproduction probability: $p_i = f(s_i) / \Sigma_j f(s_j)$

- **Example**: $\Sigma_j f(s_j) = 5+20+11+8+6=50$
- $p_1=5/50=10\%$

Individual	Fitness	Prob.
A	5	10%
B	20	40%
C	11	22%
D	8	16%
E	6	12%

Example: Scheduling Courses

Let's run through an example:

- **5 courses: A,B,C,D,E**
- *3 time slots:* Mon/Wed, Tue/Thu, Fri/Sat
- Students wish to enroll in three courses
- Goal: maximize student enrollment

Courses	Students
A B C	2
A B D	7
A D E	3
B C D	4
B D E	10
C D E	5

Example: Scheduling Courses

Let's run through an example:

- State: course assignment to time slot

M	M	F	T	M
A	B	C	D	E

= MMFTM

- Here:
  - Courses A, B, E scheduled Mon/Wed
  - Course D scheduled Tue/Thu
  - Course C scheduled Fri/Sat

Courses	Students
A B C	2
A B D	7
A D E	3
B C D	4
B D E	10
C D E	5

Example: Scheduling Courses

Value of a state? Say `MMFTM`

Courses	Students	Can enroll?
A B C	2	No
A B D	7	No
A D E	3	No
B C D	4	Yes
B D E	10	No
C D E	5	Yes

- Here 4+5=9 students can enroll in desired courses

Example: Scheduling Courses

First step:

- Randomly initialize and evaluate states

`MMFTM` = 9

`TTFMM` = 4

`FMTTF` = 19

`MTTTF` = 3

`MMFTM` = 26%

`TTFMM` = 11%

`FMTTF` = 54%

`MTTTF` = 9%

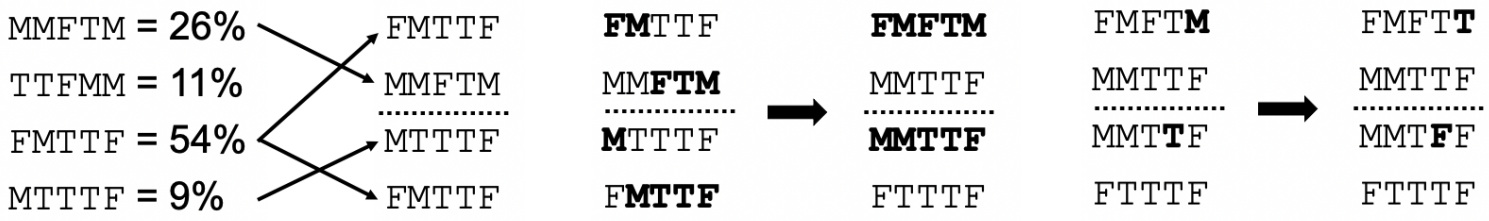
- Calculate reproduction probabilities

Courses	Students
A B C	2
A B D	7
A D E	3
B C D	4
B D E	10
C D E	5

Example: Scheduling Courses

Next steps:

- Select parents using reproduction probabilities
- Perform crossover
- Randomly mutate new children



Example: Scheduling Courses

Continue:

- Now, get our function values for updated population
- Calculate reproduction probabilities

FMFTT = 11	FMFTT = 39%
MMTTF = 13	MMTTF = 46%
MMTFF = 4	MMTFF = 14%
FTTTF = 0	FTTTF = 0%

Courses	Students
A B C	2
A B D	7
A D E	3
B C D	4
B D E	10
C D E	5

Variations & Concerns

Many **possibilities**:

- Parents survive to next generation
- Use ranking instead of exact value of $f(s)$ for reproduction probabilities (reduce influence of extreme f values)

Some **challenges**

- Formulating a good state encoding
- Lack of diversity: converge too soon
- Must pick a lot of parameters

Summary

- Challenging optimization problems
  - First, try hill climbing. Simplest solution
- Simulated annealing
  - More sophisticated approach; helps with local optima
- Genetic algorithms
  - Biology-inspired optimization routine

HelioCampus Course Evaluations

- Please do them! Feedback used to design future versions of 540
- Final exam incentive
  - With >50% participation, instructors will release an “excluded topics” list for final exam
  - With >70%, more topics excluded
- Survey open until 12/10