CSCI 561 - Foundation for Artificial Intelligence

Discussion Section (Week 3)

PROF WEI-MIN SHEN WMSHEN@USC.EDU

Outline

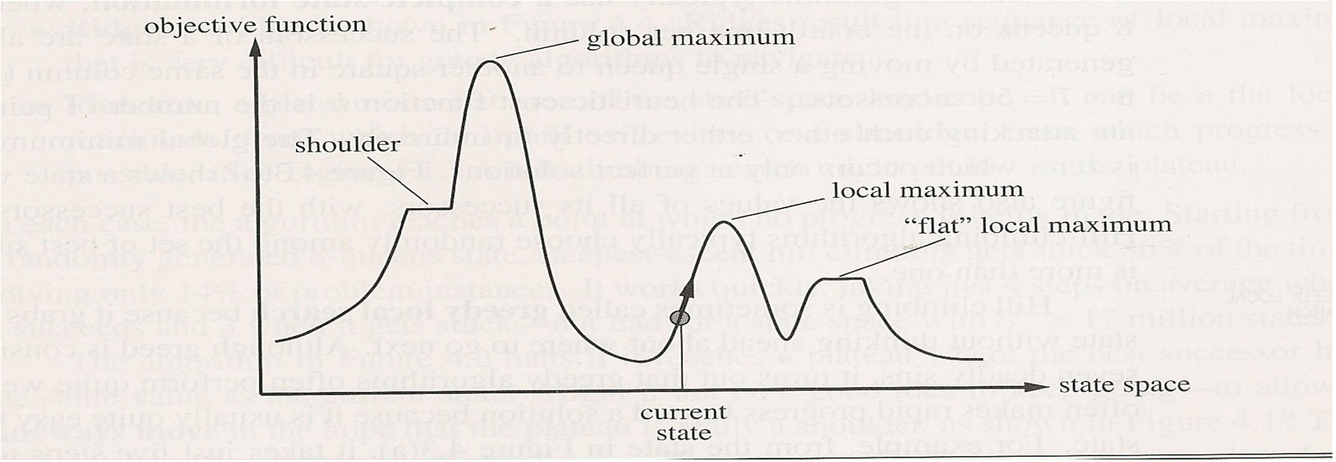
1. Function Optimization
2. Constraint Satisfaction
3. Game Playing

What Is Optimization? So "Simple"?

Given: a hidden objective function $F(x)$

Find: a x^* such that $F(x^*)$ is the global extreme:

$$F(x^*) = \max\{ F(x) \} \text{ or } F(x^*) = \min\{ F(x) \}$$



Why Is It So Important?

All engineering problems are optimizations!

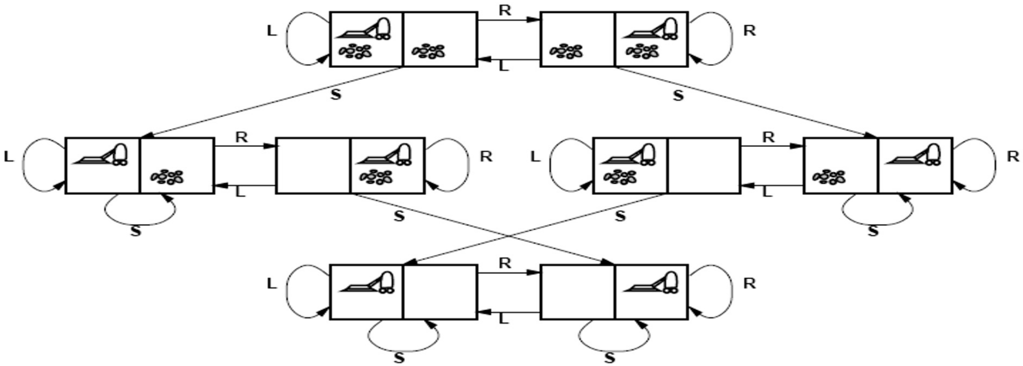
- Design: architecture, software, robots, website, …
- AI systems: industry, agriculture, military, finance, …
- For yourself: Getting good grades ☺

The key challenge is: How to represent them properly

For Search, you can represent your desires as the objective function

- Choice 1: *x* as a state, F(x) as the “rewards” of states
- Choice 2: x as a path, F(x) as the “rewards” of paths
- Degree of “goodness”: goal related, cost related, etc.

Represent Problems as Optimization



How to represent this problem as an optimization problem?

x: 00, 01, 10, 11

- State: LeftRoom_RightRoom, (dirty=0, clean=1)

F(x)=x; “the cleaner the room, the higher F(x)”

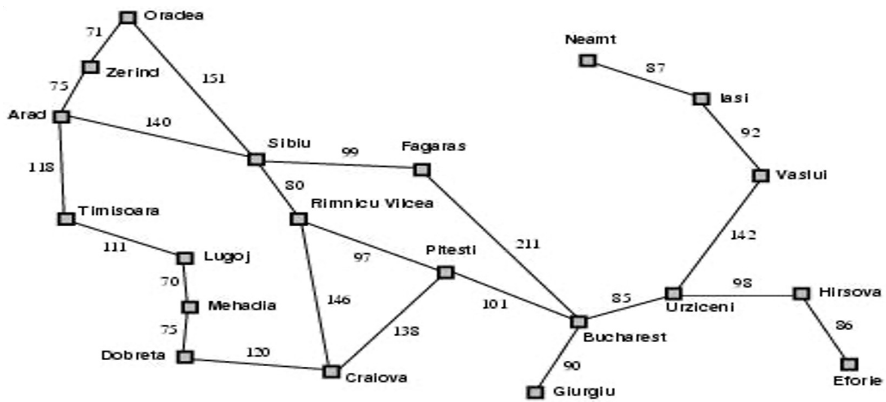
Representing as optimization

Choice 1: x as a city

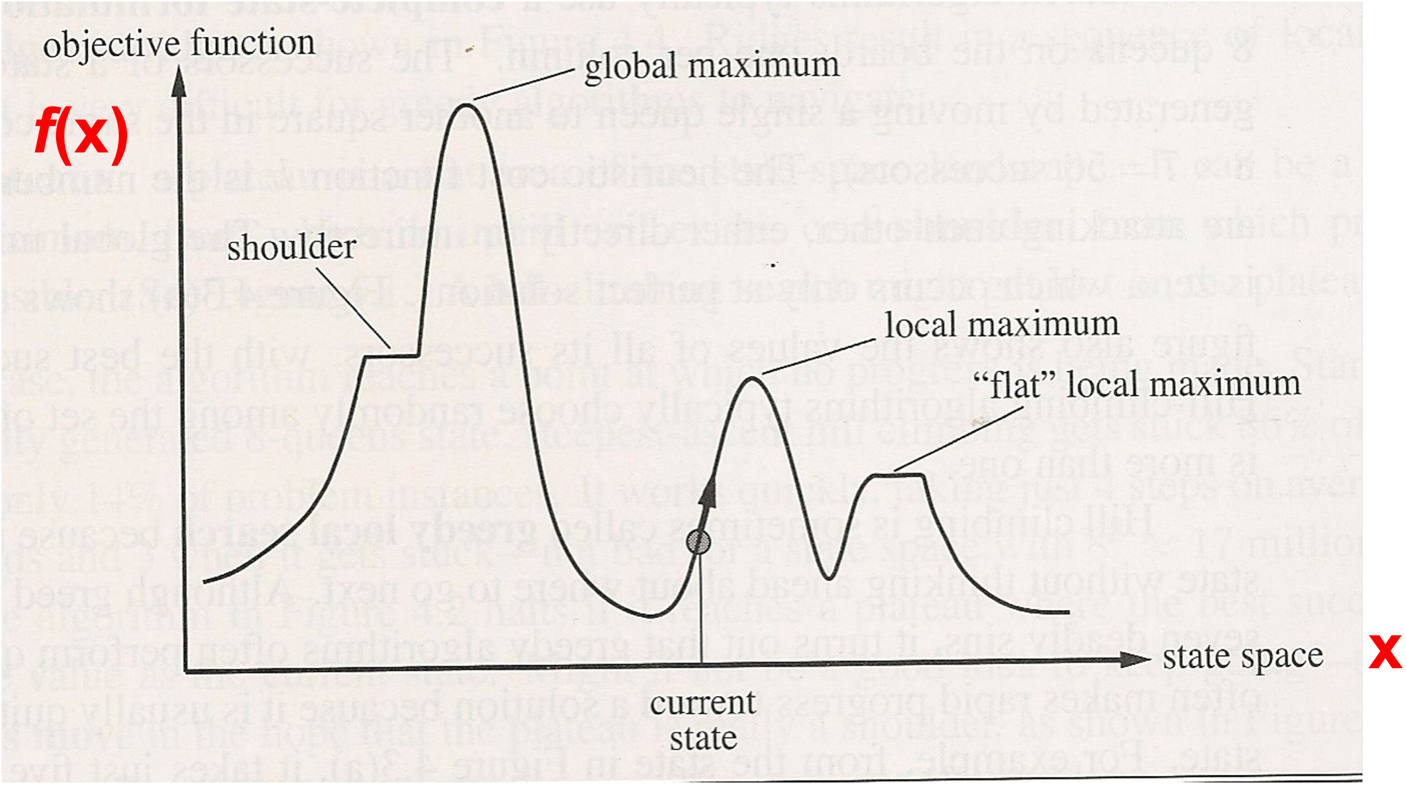
- F(x) is the distance from x to Budapest

Choice 2: x is a sequence of cities starting from Arad

- F(x) has a higher value if the path ends at Budapest
- What about “leads to Budapest”? What about the “cost”?



Why is optimization so hard?



Why is it so hard?

Which way to go next?

How much can you see? (local/partial vs global/complete sensors)

How many points can you remember (incremental)?

How small is your step? (not to skip x*)

How to get from one x to another (bound by actions)?

How well can you guess (the next x)?

What do you know about the function (continuous)?

How to check if you are done (avoid local extremes)?

Will the function F(x) change by itself (often does)?

How to design the objective function?

In AI, we call it the Representation Challenge

We still don’t have a general solution for all

We will illustrate all these using the following story

Representation Challenge for AI

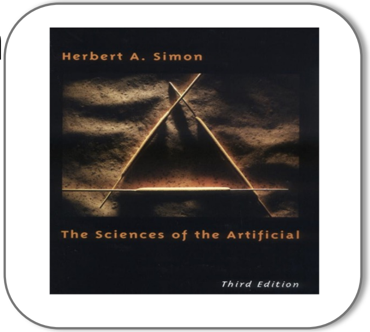
Why is "representation" so important?

A great example from Herbert A. Simon

- Game: Make a book of 15 from 9 cards
- Goal: first does wins (can you always win?)
- "The Science of the Artificial" p131

How to find a good representation

- is still an open challenge for AI



4	9	2
3	5	7
8	1	6

Some Optimization Methods

Dynamic programming

Hill climbing

- Idea: Use local gradient(x) to determine direction, always heads to the better
- Pros: simple, local, incremental, no memory
- Cons: may be trapped in local extreme

Simulated Annealing

- Ideas: long and random jumps when temperature is high
- Pros: may avoid local extreme
- Cons: expensive, not always find x*,

Genetic algorithms

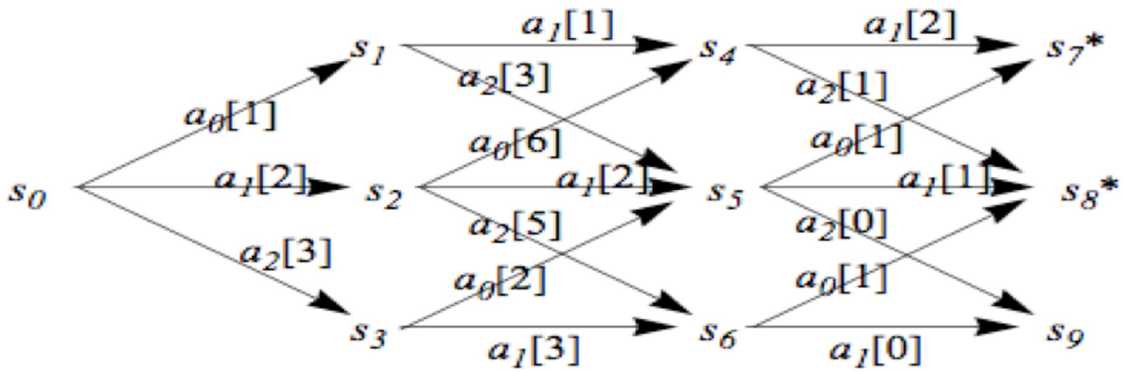
Sampling techniques (e.g., random walks)

Online (incremental) search

Non-stationary search techniques

Many more “new” methods are being invented as we speak

Dynamic Programming for Optimization (see ALFE 6.1.1)



Note: x is a state, but f(x) is about the most rewarding path from x to a goal sate

$$
\begin{aligned}
V(s_1) &= \max\{R(s_1, a_1) + V(s_4), R(s_1, a_2) + V(s_5)\} = \max\{1+2, 3+1\} = 4 \\
V(s_2) &= \max\{R(s_2, a_0) + V(s_4), R(s_2, a_1) + V(s_5), R(s_2, a_2) + V(s_6)\} \\
&= \max\{6+2, 2+1, 5+1\} = 8 \\
V(s_3) &= \max\{R(s_3, a_0) + V(s_5), R(s_3, a_1) + V(s_6)\} = \max\{2+1, 3+1\} = 4
\end{aligned}
$$

$$V(s_i) = \max_a \{R(s_i, a) + V(s_j)\} \text{ where } s_i \xrightarrow{a} s_j$$

Iterative Improvement

In many optimization problems, path is irrelevant;

the goal state itself is the solution.

Then, state space = space of "complete" configurations.

Algorithm goal:

- find optimal configuration (e.g., TSP), or,

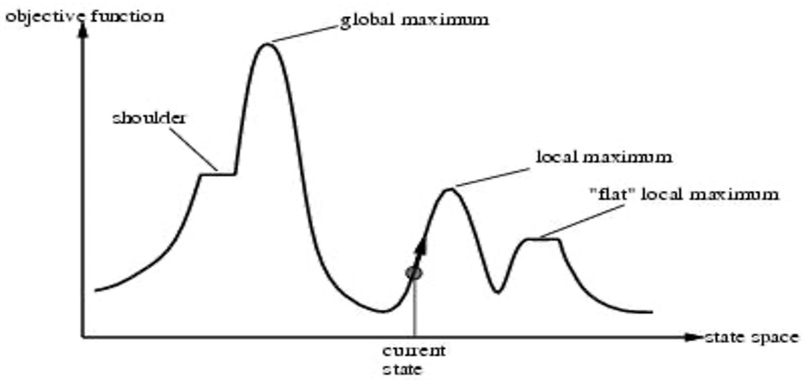
- find configuration satisfying constraints

(e.g., n-queens)

In such cases, can use iterative improvement algorithms: keep a single "current" state, and try to improve it.

2、

Hill Climbing



Move continuously in the direction of increasing value

- Go to best successor of current state, based on evaluation
  - If more than one best successor, pick randomly among them

Terminate when reach a peak

- May only find a *local maximum*

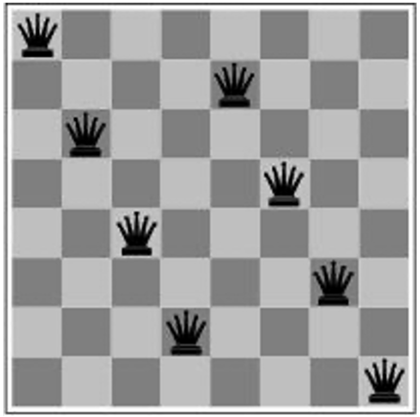
This form of hill climbing is also referred to as

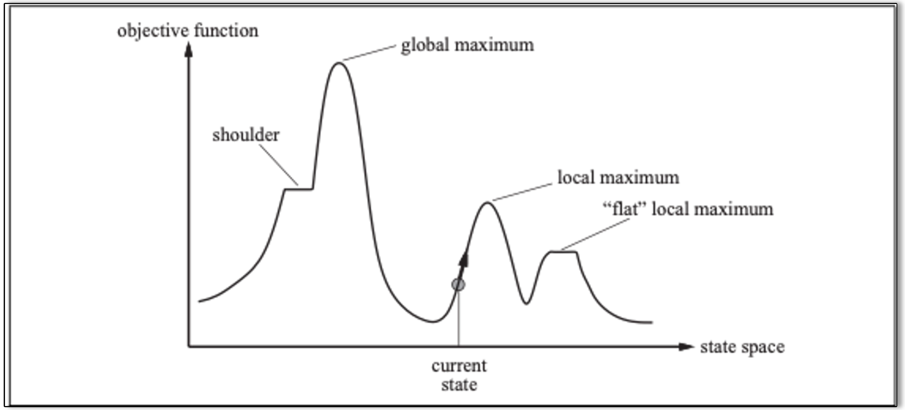
- Steepest-ascent hill climbing
- Greedy local search
- Continuous analogue is *gradient ascent*

Hill Climbing

- What are the advantages and disadvantages of classical hill-climbing?
- Can you think of real-world examples where hill-climbing would be particularly good?

- What about bad?





Key Questions

How big is your step?

You “walk”, but do you “jump”? When? How far?

When do you stop?

Is you technique deterministic and complete?

Simulated Annealing

Escape local maxima by allowing "jump" moves

- **When to jump?**
  - If you always jump too much, then you never settle down
  - Gradually decrease the likelihood to jump over time
  - Use temperature to determine the likelihood to jump
  - Reduce temperature ***T*** slowly over the time
- **How far to jump?**
  - Too close? Too far? Random?

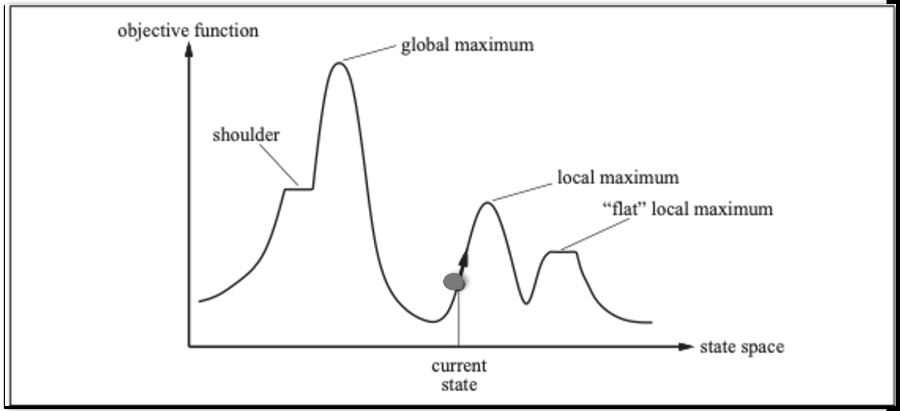
If *T* decreases slowly "enough," then you can find the optimal extreme

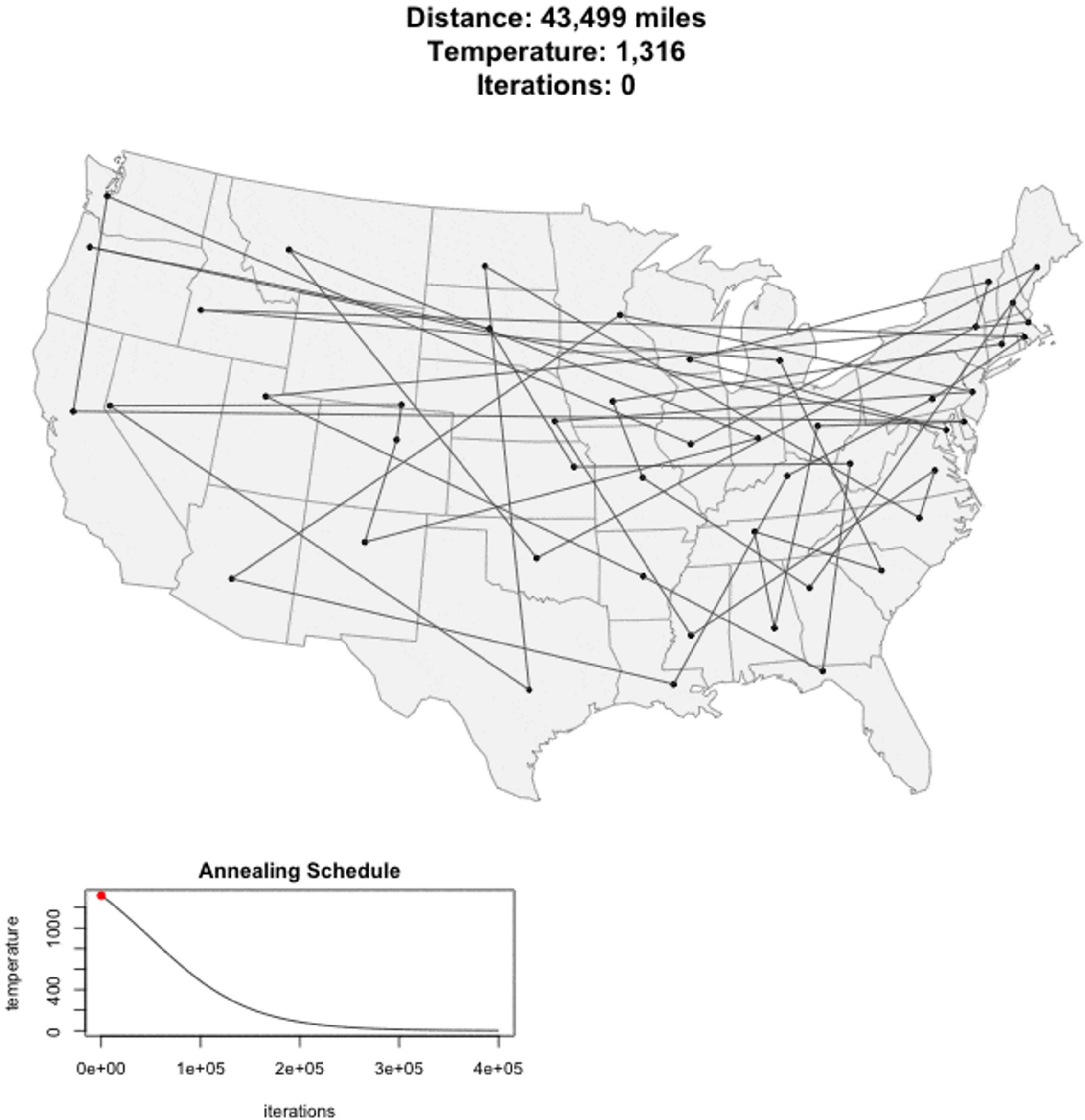
1. **How slow? Infinitely slow**
2. **Theoretically OK, but infeasible in practice**

Simulated Annealing

The AIMA book says that simulated annealing is <u>complete</u> (page 125).

- **In real-world applications will that be true?**
- **Why or why not?**

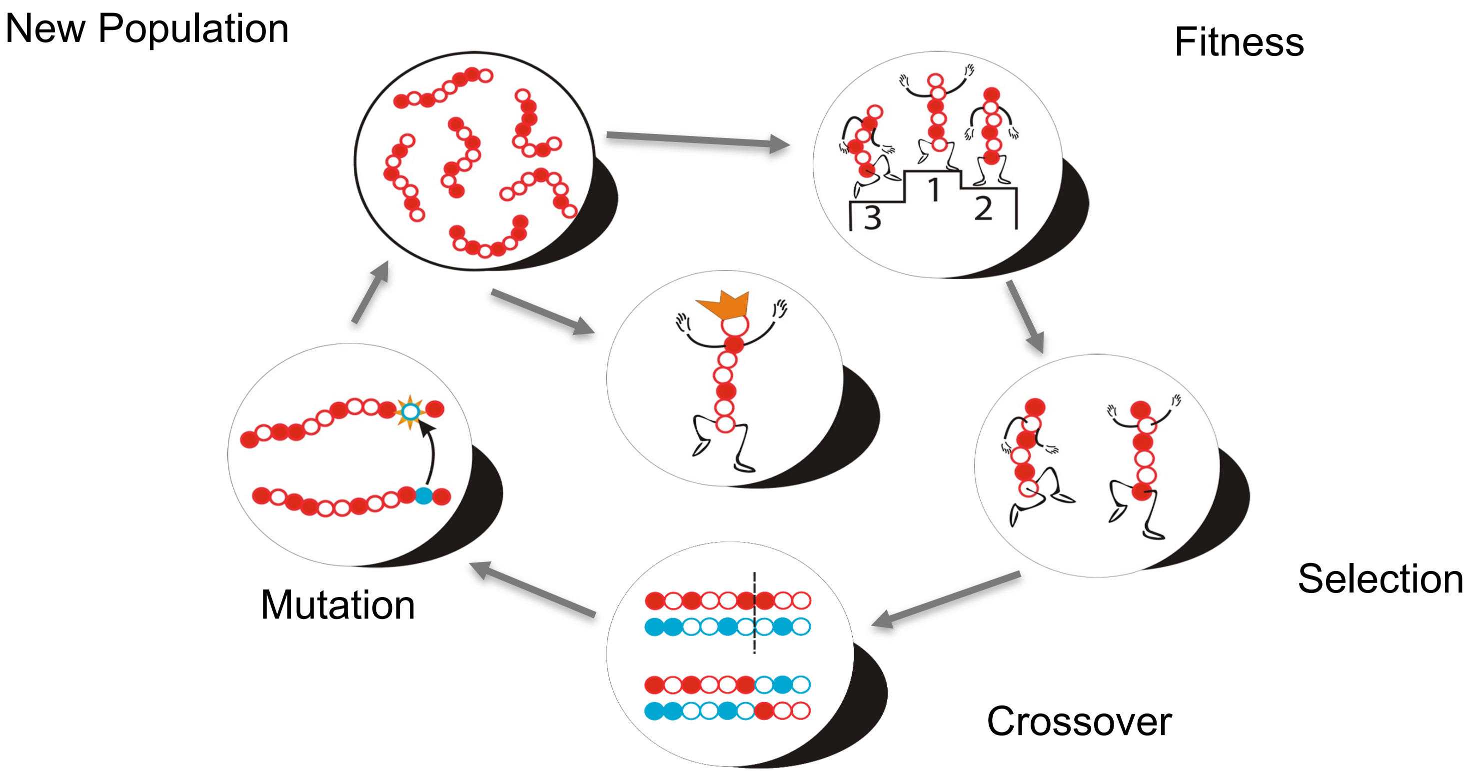


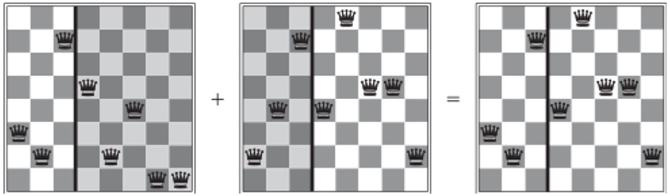


http://toddwschneider.com/posts/traveling-salesman-with-simulated-annealing-r-and-shiny/

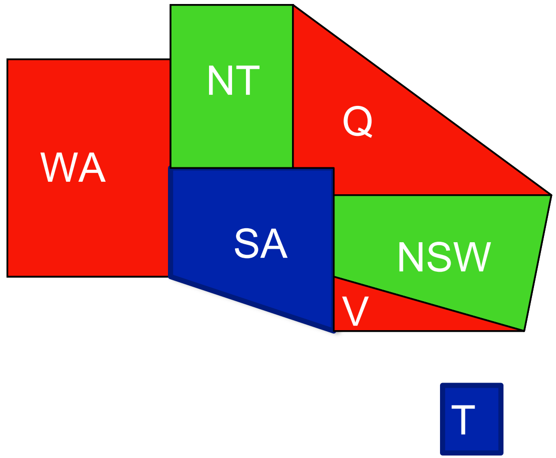
4-

The Genetic Algorithm Cycle





CSP Example: Map Coloring

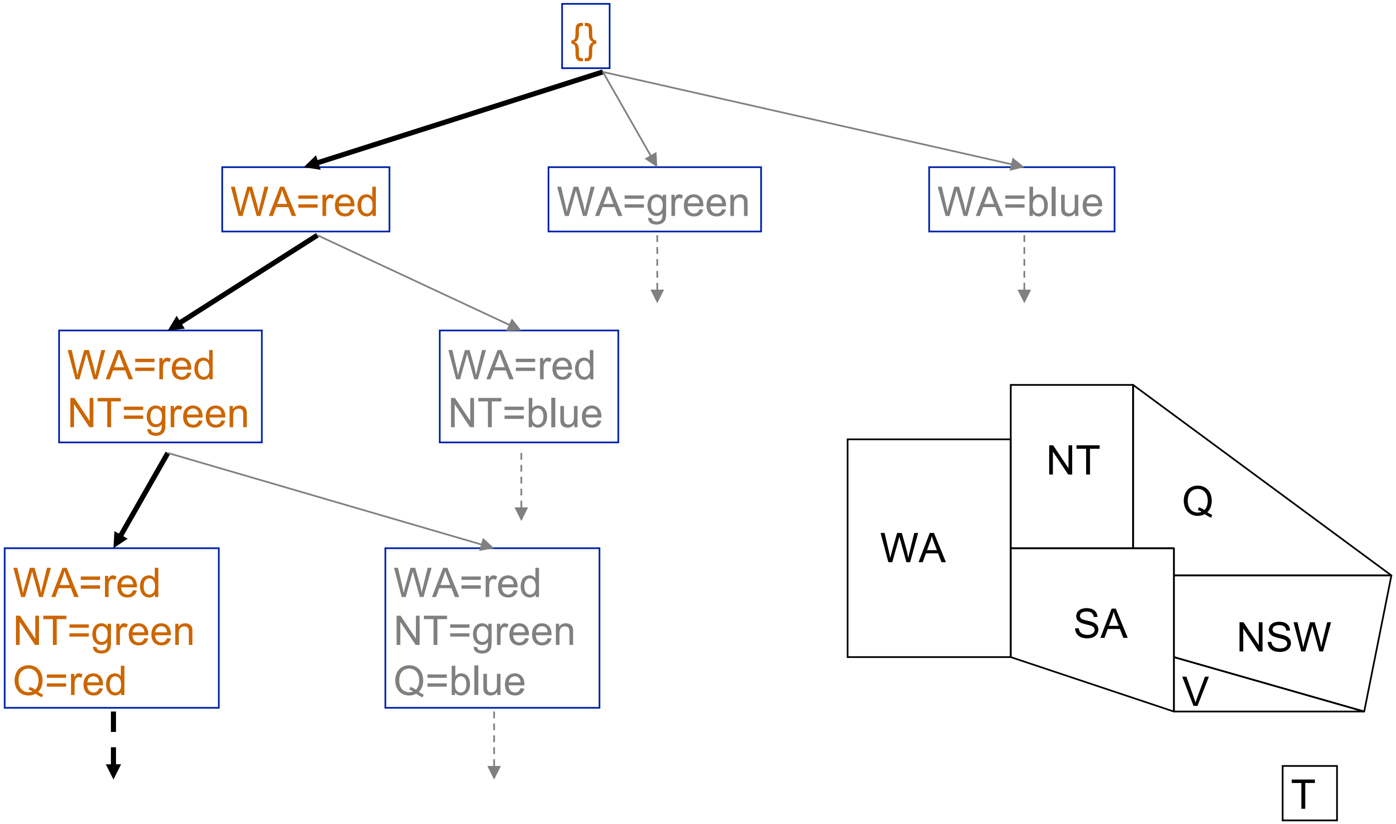


- 7 variables {WA,NT,SA,Q,NSW,V,T}
- Each variable has the same domain {red, green, blue}
- No two adjacent variables have the same value:

  WA≠NT, WA≠SA, NT≠SA, NT≠Q, SA≠Q, SA≠NSW, SA≠V,Q≠NSW, NSW≠V

2、

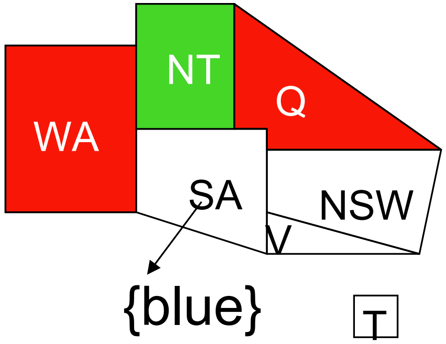
Backtracking Search: Map Coloring



3-

Constraint Propagation

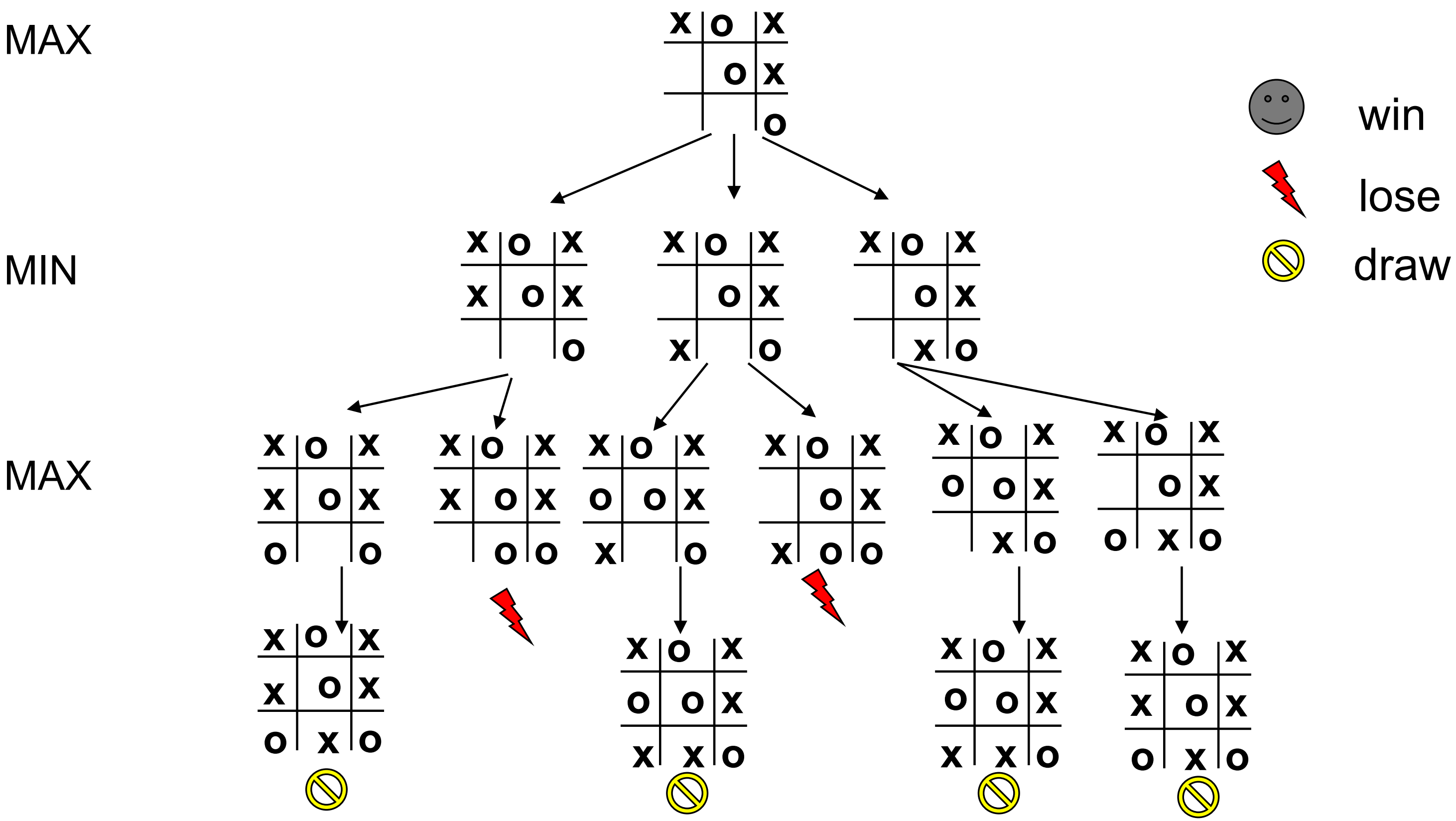
- Which variable X should be assigned a value next?
- In which order should its values be tried?
- **Variable Selection:**
  - "Most constrained variable" or "Minimum Remaining Values"
  - Degree: variable involved in most constraints on others (tiebreaker)
- **Value Selection:**
  - Least constraining value



When to use CSP Techniques?

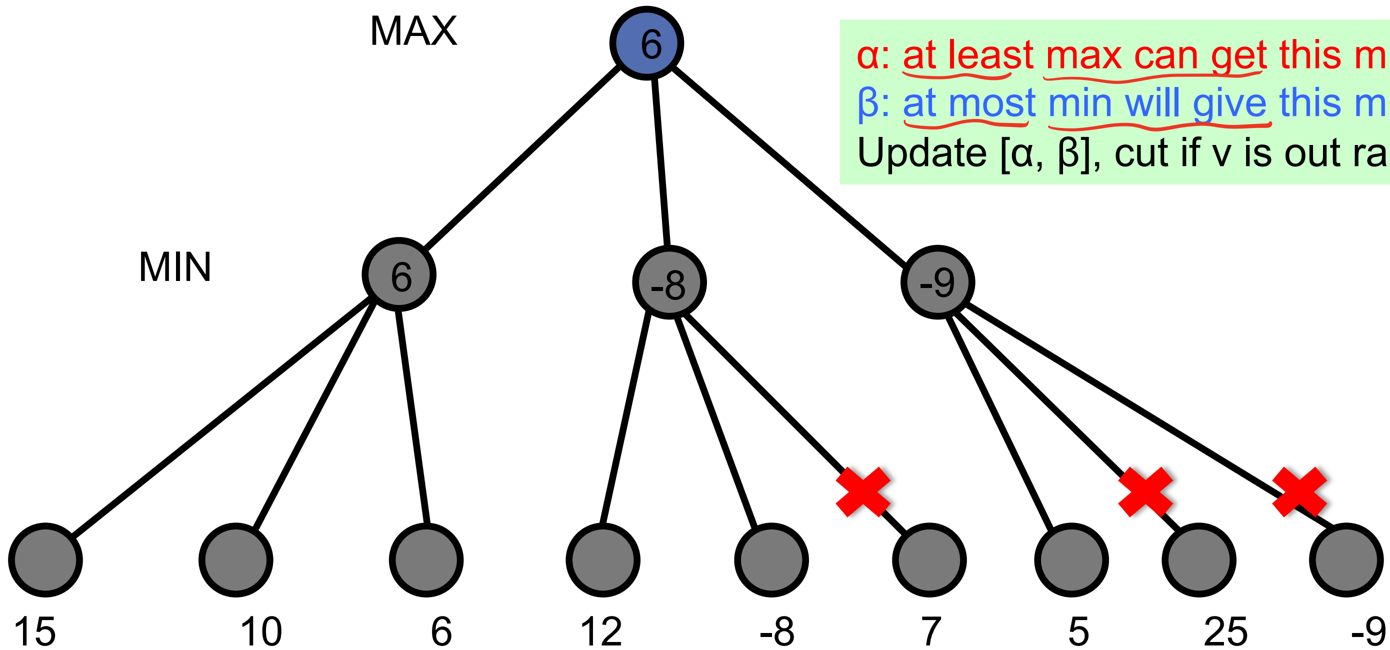
- When the problem can be expressed by a set of variables with constraints on their values
- When constraints are relatively simple (e.g., binary)
- When constraints propagate well (AC3 eliminates many values)
- Local Search: when the solutions are “densely” distributed in the space of possible assignments

Game Playing -- Adversarial Search



2, ☆

Minimax Algorithm

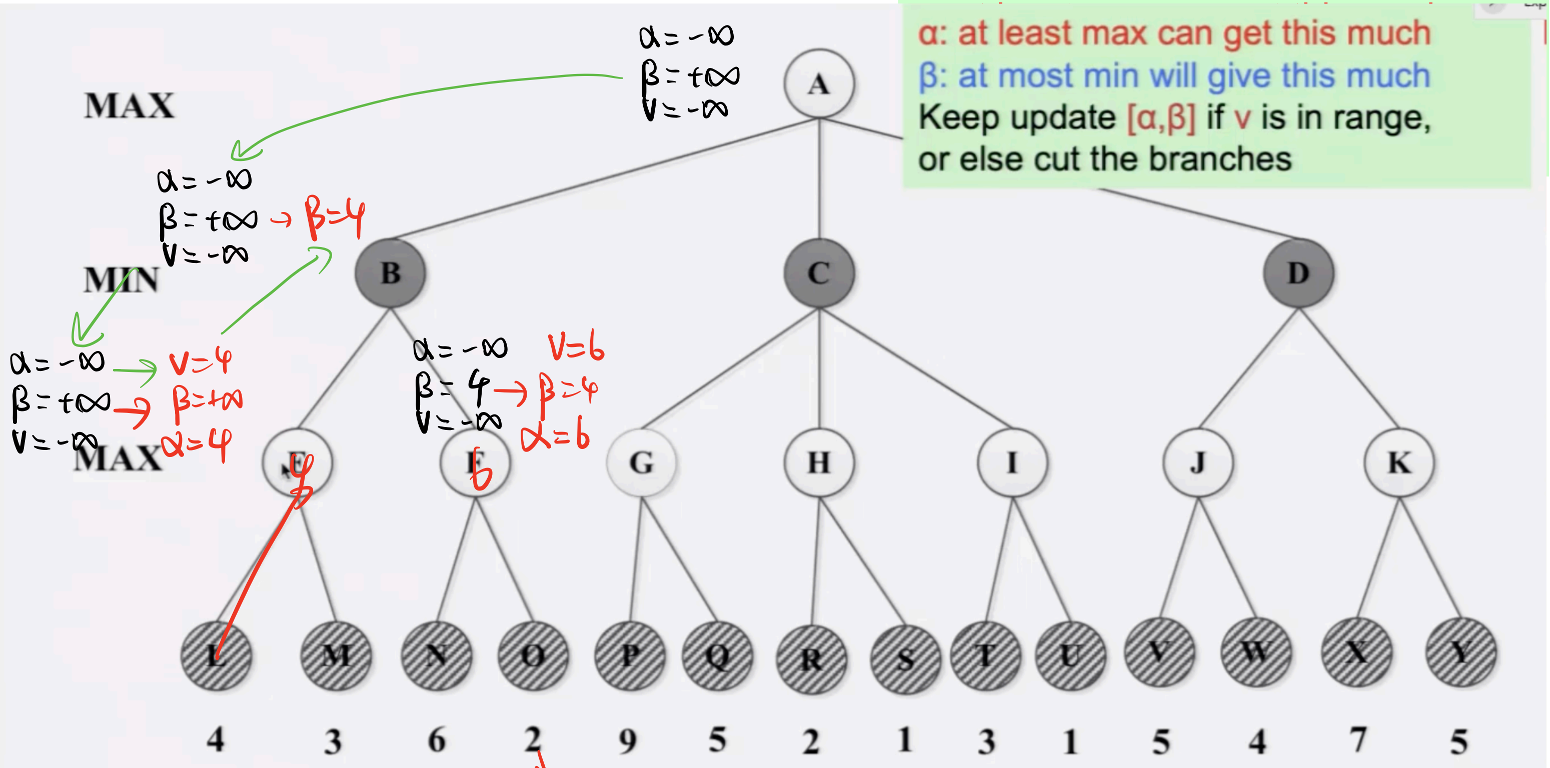


α: at least max can get this much
β: at most min will give this much
Update [α, β], cut if v is out range

α: at least max can get this much

β: at most min will give this much

Keep update [α,β] if v is in range, or else cut the branches



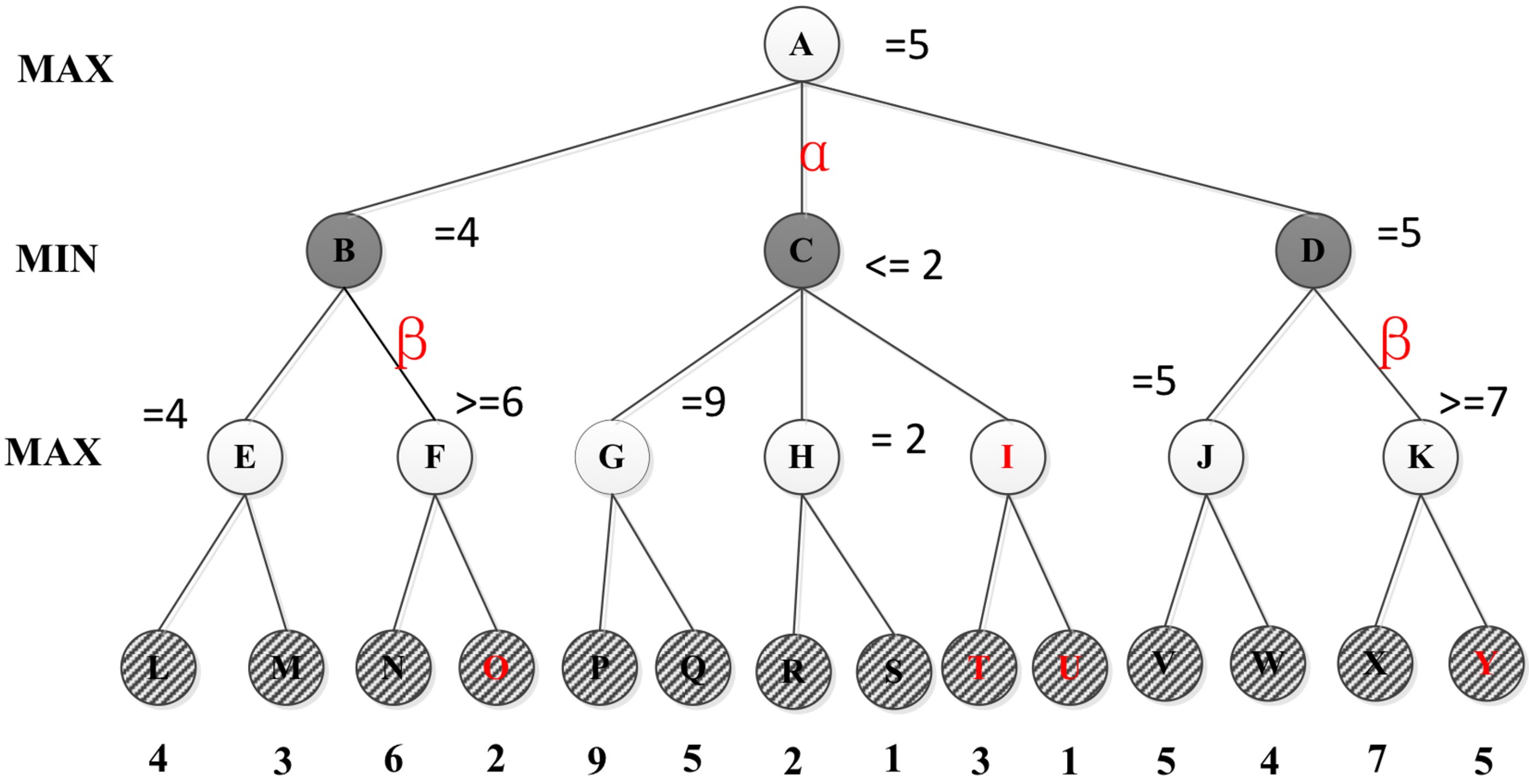
can cut,因为B中有α至少为4

Max Node

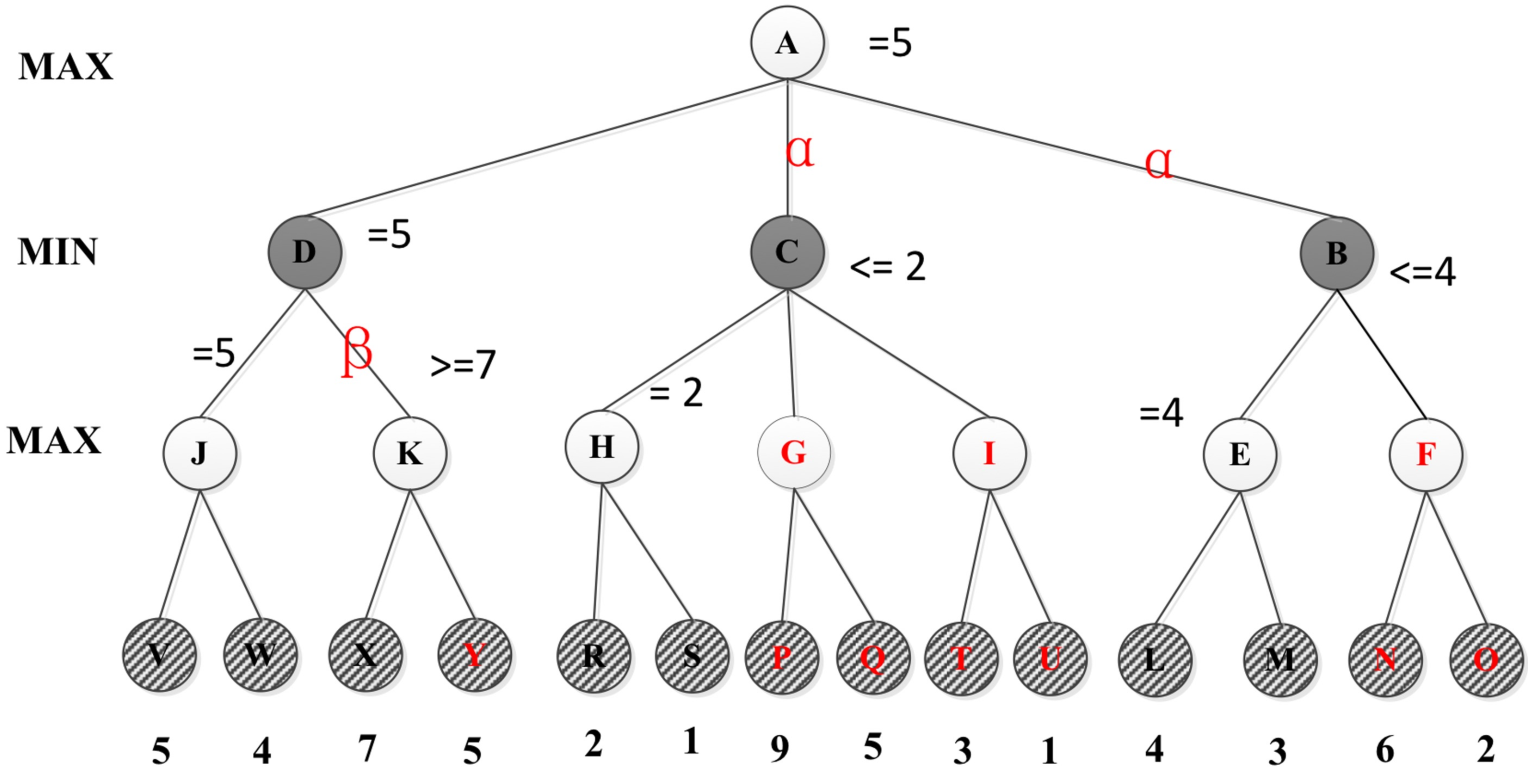
for each a **in** ACTIONS(state) **do**
  v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
  if v ≥ β **then return** v
  α ← MAX(α, v)
return v

Min Node

for each a **in** ACTIONS(state) **do**
  v ← MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
  if v ≤ α **then return** v
  β ← MIN(β, v)
return v



MAX
MIN
MAX
A
=5
α
B
=4
C
<= 2
D
=5
β
β
=4
E
F
>=6
G
=9
H
= 2
I
=5
J
K
>=7
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
4
3
6
2
9
5
2
1
3
1
5
4
7
5


MAX
MIN
MAX
A
=5
α
α
D
=5
C
<= 2
B
<=4
=5
β
>=7
= 2
=4
J
K
H
G
I
E
F
V
W
X
Y
R
S
P
Q
T
U
L
M
N
O
5
4
7
5
2
1
9
5
3
1
4
3
6
2

What you should know

- **What are the characteristics of local search? Is it complete? Optimal? Time and Space complexity?**
- **What problem domains are well-suited to each type of search technique? Ill-suited? Why?**
- **Know how to compare their performance in general and in a specific domain or problem.**
- What is the difference between uninformed and informed search? Which ones are optimal?
- What are the advantages and disadvantages of depth-first search?
- Why does a search heuristic need to be "admissible"?
- What are the characteristics of Adversarial Search?
- Why is *meta-reasoning* important in adversarial search?

Want more?

Check out these demos:

http://toddwschneider.com/posts/traveling-salesman-with-simulated-annealing-r-and-shiny/
http://codecapsule.com/2010/04/06/simulated-annealing-traveling-salesman/
http://www.biostat.jhsph.edu/~iruczins/teaching/misc/annealing/animation.html

http://math.hws.edu/eck/jsdemo/jsGeneticAlgorithm.html
http://rednuht.org/genetic_walkers/
http://rednuht.org/genetic_cars_2/

Alpha-beta search demo:

https://www.yosenspace.com/posts/computer-science-game-trees.html

A* and heuristics:

http://www.briangrinstead.com/files/astar/

Practice Exercises: **Chapter 4: # 4.1, Chapter 6: # 6.1, 6.5**