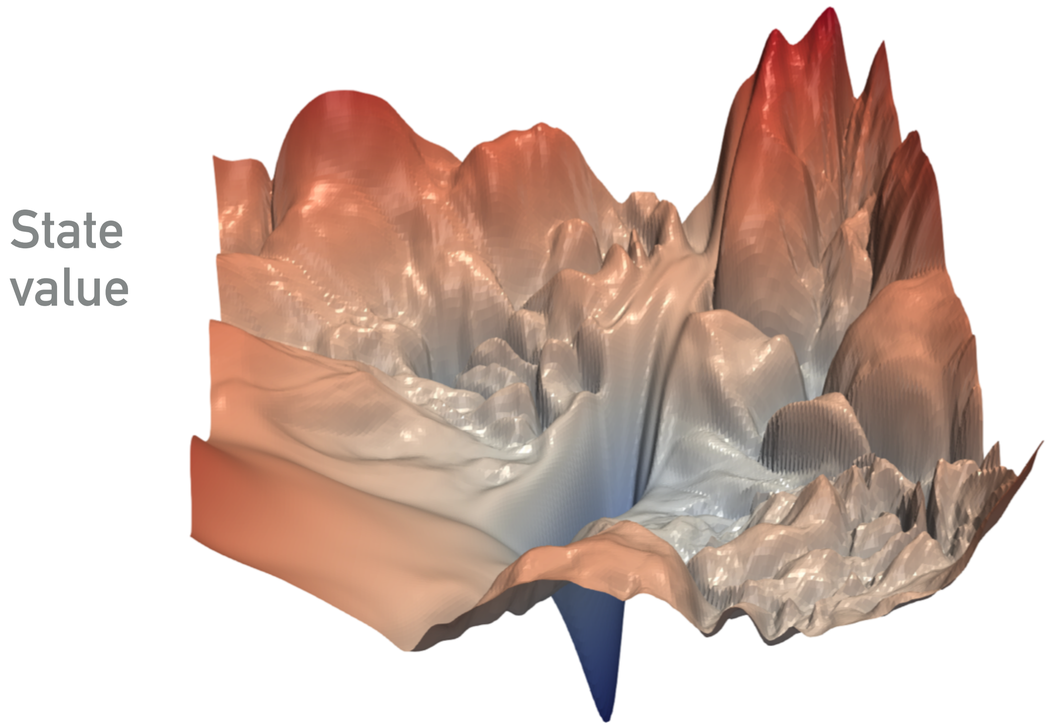
PURDUE CS47100

INTRODUCTION TO AI

RECAP: GRAPH SEARCH & LOCAL SEARCH

- Graph search
  - Never expand the same state twice!
  - A* needs an consistent heuristic for the graph search algorithm to be optimal
- Local search algorithm
  - Useful when path to goal state does not matter / solve pure optimization problem
  - Hill-climbing search: always move to a neighboring state with the optimal objective function value

STATE SPACE LANDSCAPES IN PRACTICE (FOR NEURAL NETWORKS)



VARIANTS OF LOCAL HILL-CLIMBING

- Stochastic hill-climbing
  - Select randomly from all moves that improve the value of the objective function
- Random-restart hill-climbing
  - Conducts a series of hill-climbing searches, starting from random positions
  - Very frequently used general method in AI

SIMULATED ANNEALING

- Inspired by statistical physics/metallurgy
- Key idea: mostly goes “uphill”, but occasionally travels “downhill” to escape local optimum
  - The likelihood to go downhill is controlled by a “temperature schedule”
  - More and more “conservative” (i.e., less likely to go downhill) as the search progress

SIMULATED ANNEALING SEARCH

function SIMULATED_ANNEALING (problem, schedule) return a solution state

current ← initial_state;

for t = 1 **to** ∞**:**

T ← schedule(t)

if T = 0 **then return** current

next ← a randomly selected neighbor of current

ΔE = next.value - current.value

if $\Delta E > 0$:

current ← next

else:

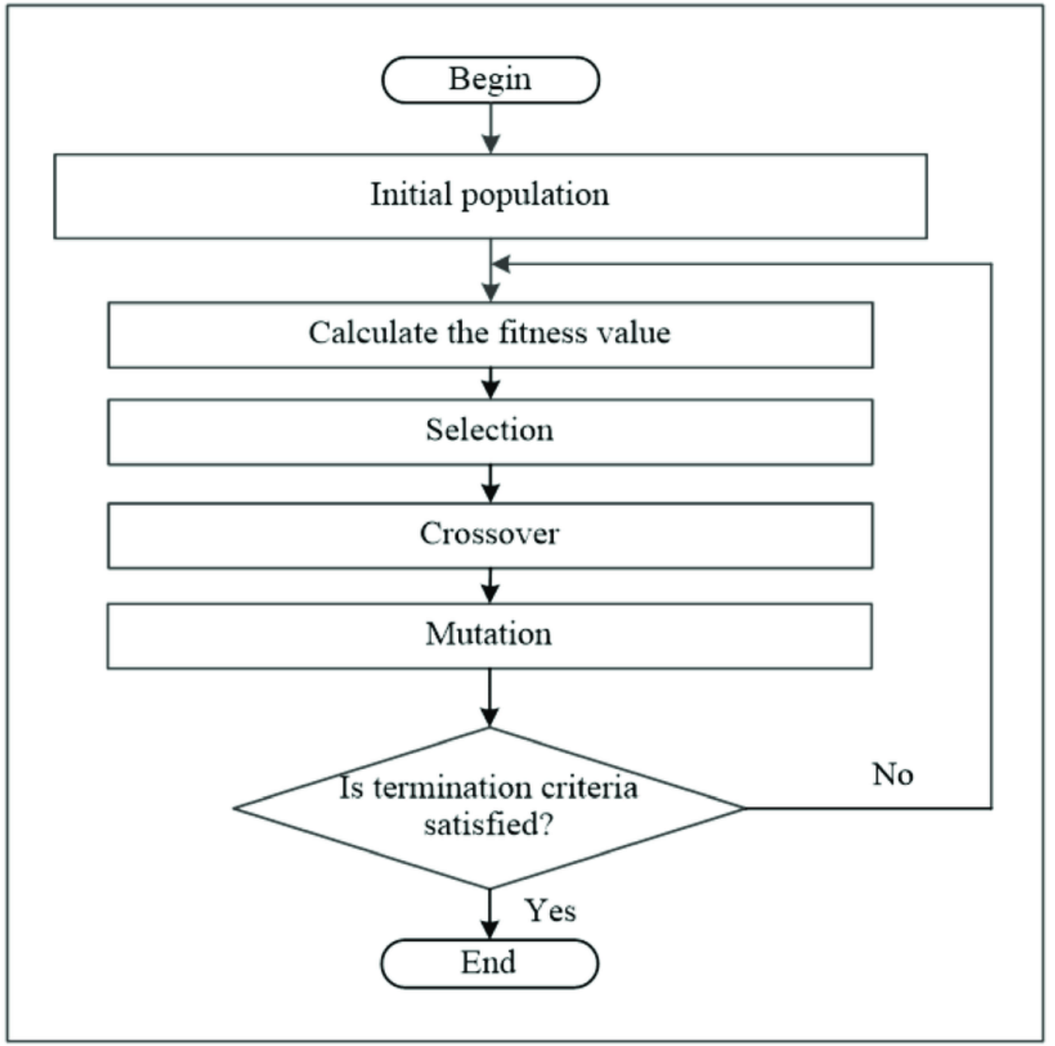
current ← next with probability $e^{\Delta E/T}$

LOCAL BEAM SEARCH

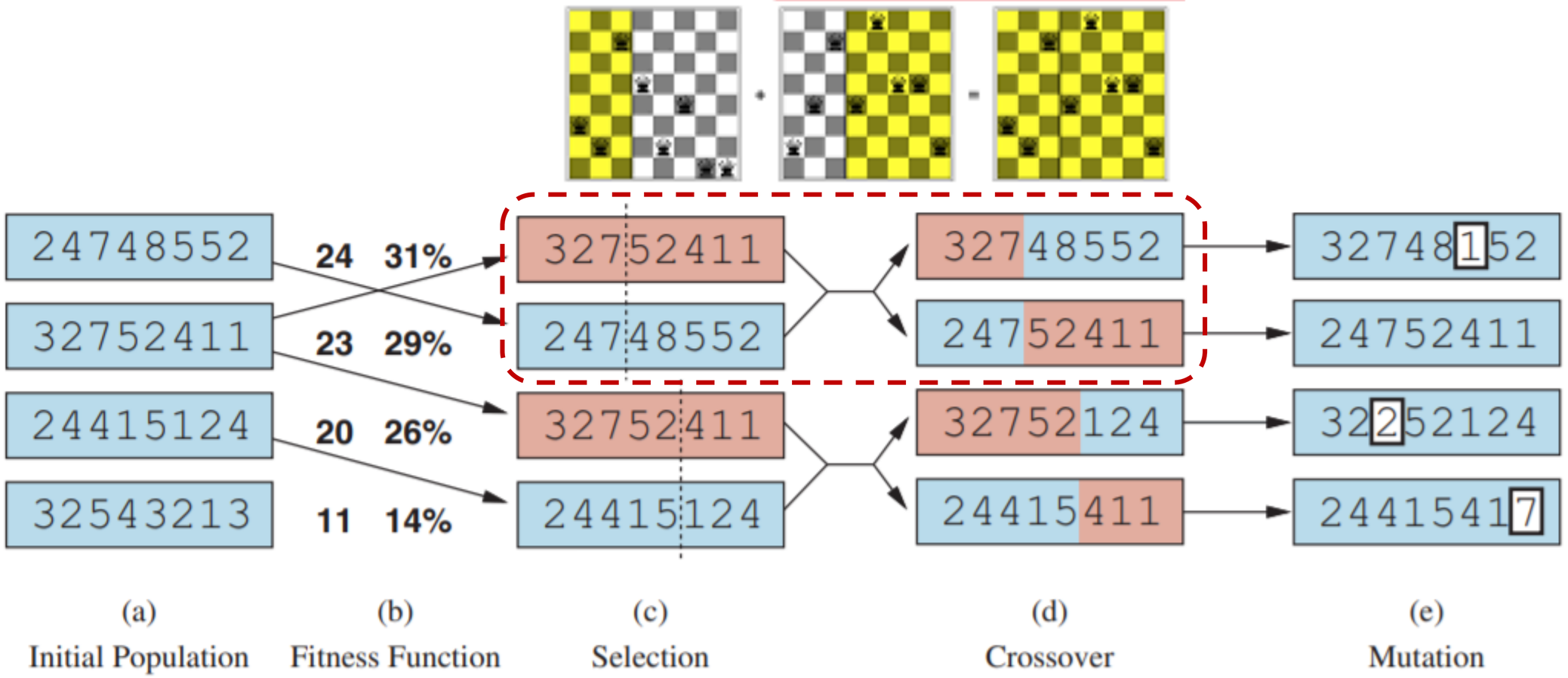
- Similar to hill-climbing, but…
  - Keeps track of *k* current states rather than just a single current state
  - Select the *k* best neighbors among all neighbors of the *k* current states
- Analogy
  - Hill-climbing: "…trying to find the top of Mt. Everest in a thick fog while suffering amnesia"
  - Local beam search: "Doing this with several friends, each of whom has a short-range radio and an altimeter"
- Stochastic beam search – Select successors at random weighted by value

GENETIC ALGORITHM

- Inspired by evolutionary biology
- Mimic the evolution of a population under natural selection: the fittest (i.e., the ones with better objective function values) survives



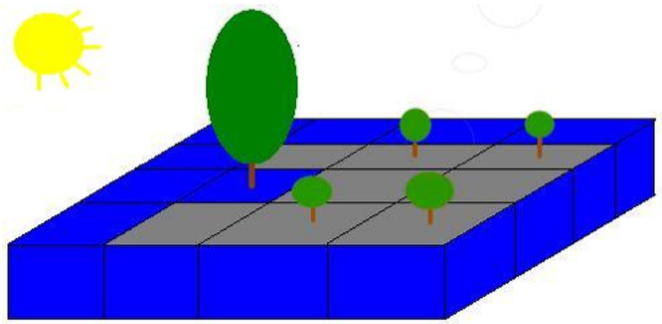
GENETIC ALGORITHM EXAMPLE: EIGHT-QUEENS PROBLEM



Fitness function: number of pairs of non-attacking queens

APPLICATION: OPTIMIZING PLANT PLACEMENT TO SAVE WATER

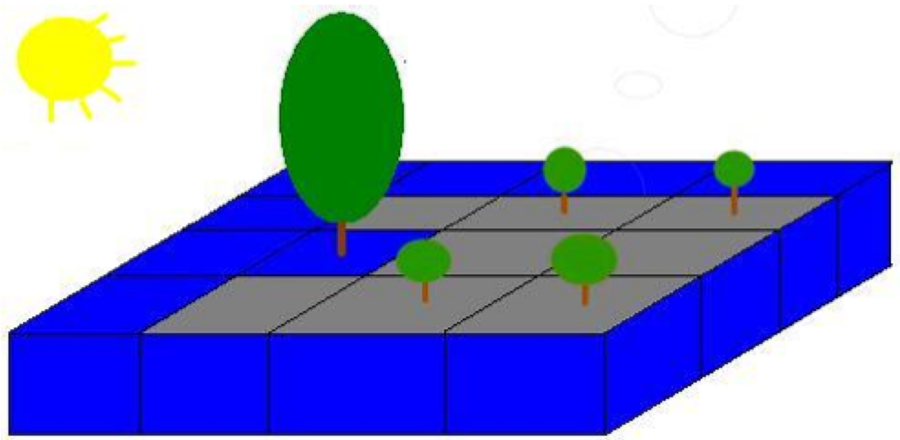
- Species have differences in size, light requirement, and water use
- Complex interactions between species, e.g., shading changes water requirements



Facilitation: when a "nurse plant" improves conditions for other plants' growth

Water conservation through facilitation on residential landscapes. AAAI 2011

APPLICATION: OPTIMIZING PLANT PLACEMENT TO SAVE WATER



$$landscape\_fitness = \frac{1}{n}\sum^{n} fitness_i$$

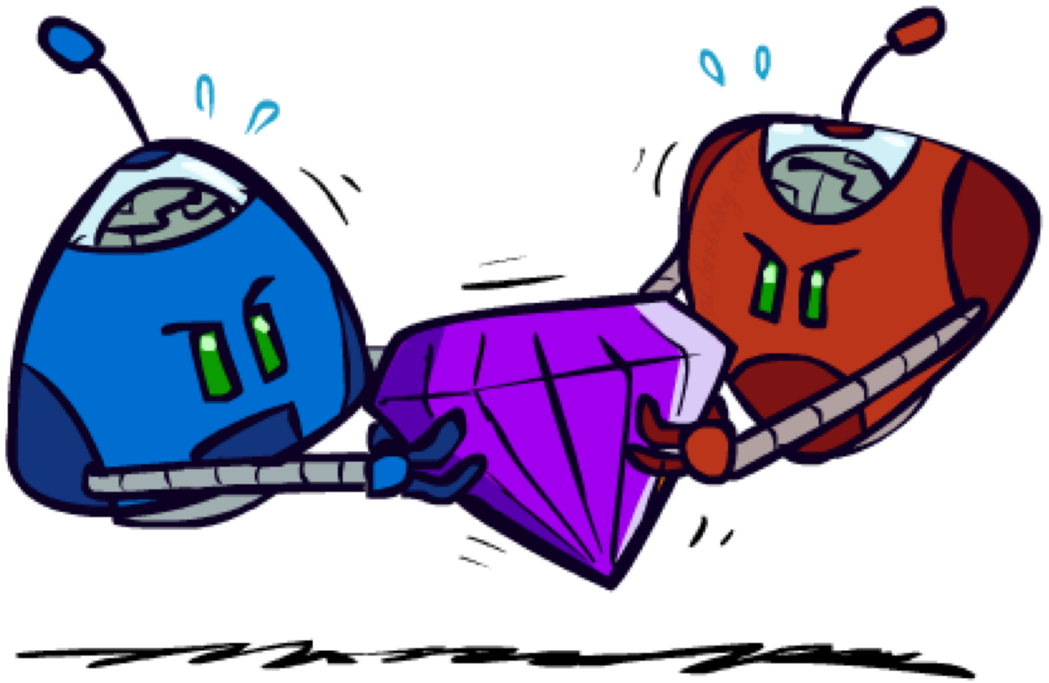
$$fitness_i = \alpha * growth_i + \beta * water\_efficiency_i$$

$$growth_i = f(lightReq_i, waterReq_i, light_{(x,y)}, water_{(x,y)})$$

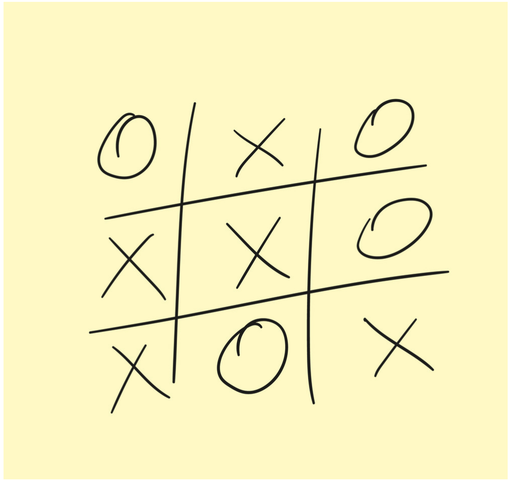
$$water\_efficiency = g(\sum_{x,y} water_{(x,y)})$$

Simulated annealing: Grow k plants in a $N \times M$ landscape while determine a watering plan to maximize the landscape fitness score

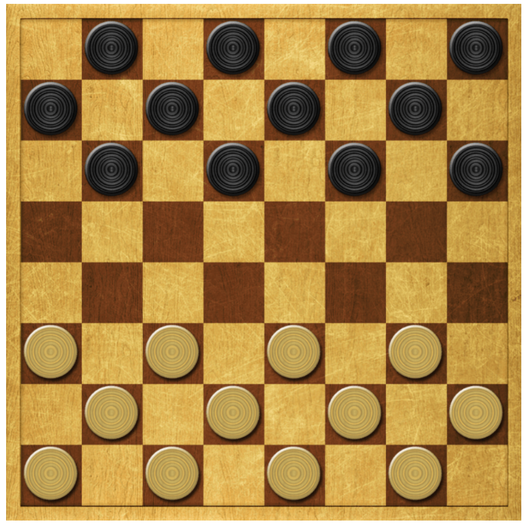
ADVERSARIAL GAMES



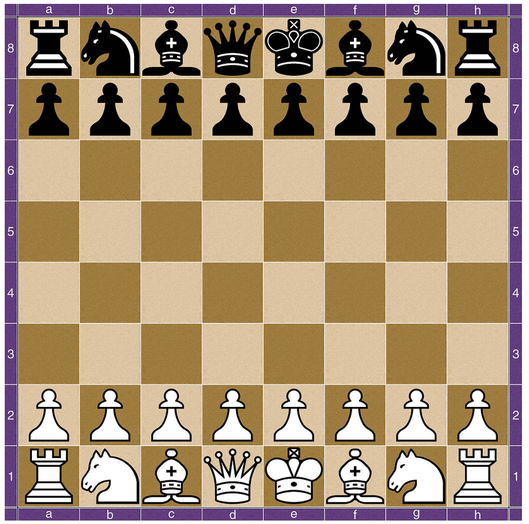
GAMES ARE EVERYWHERE



Tic-Tac-Toe



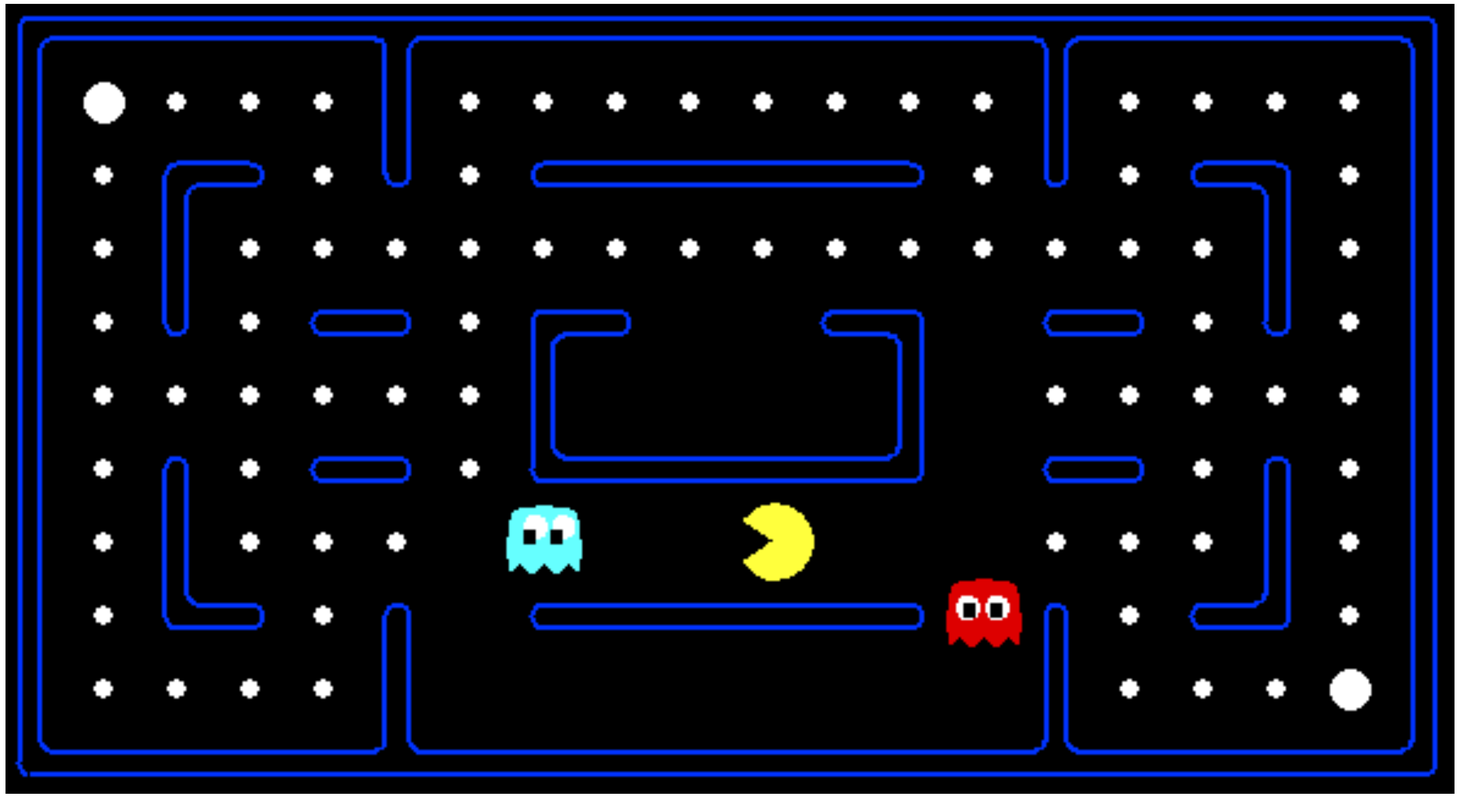
Checkers



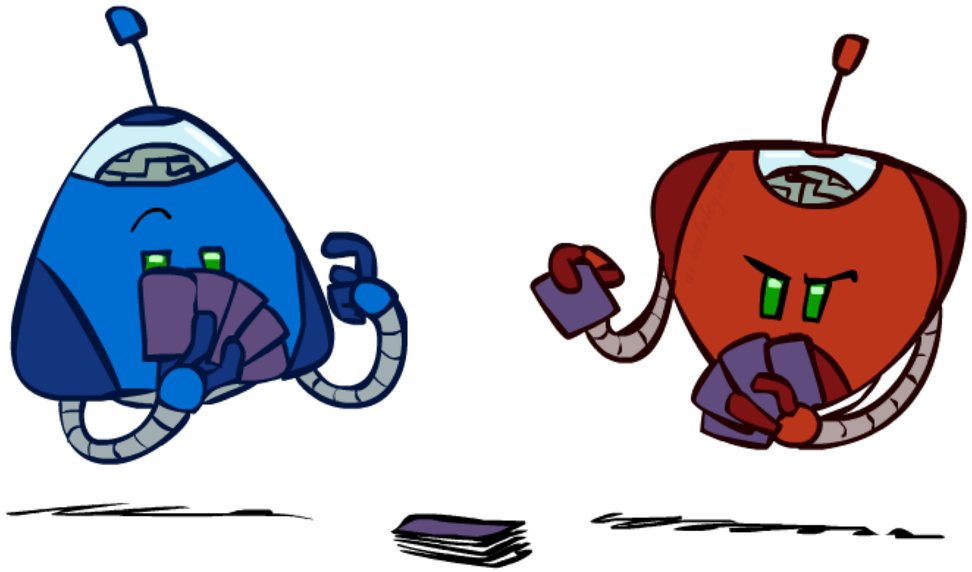
Chess

Go

BEHAVIOR FROM COMPUTATION



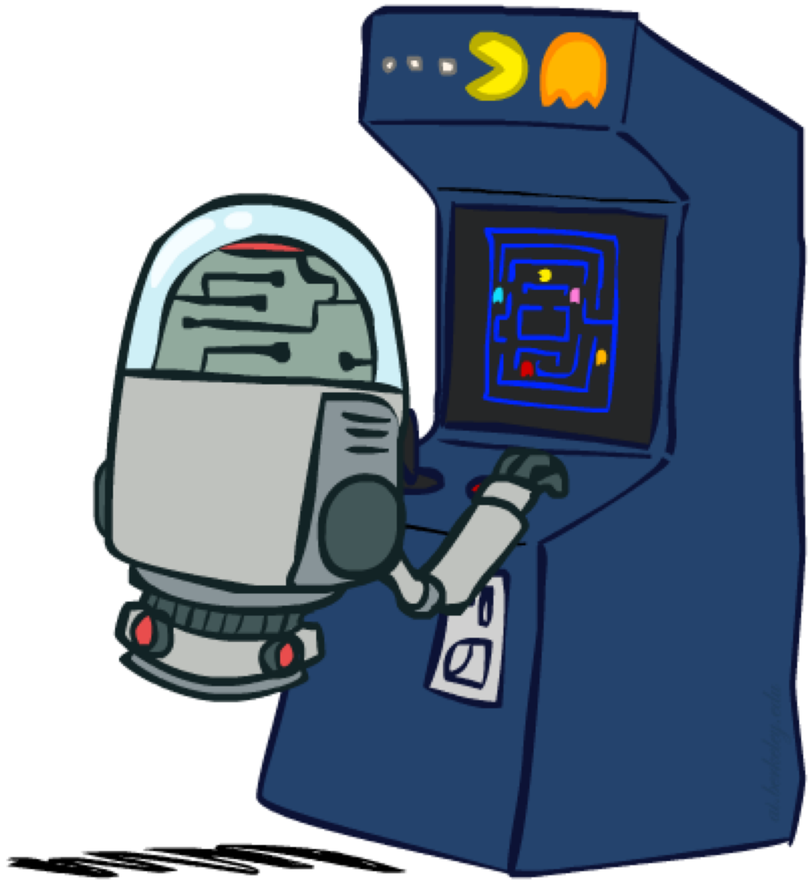
TYPES OF GAMES



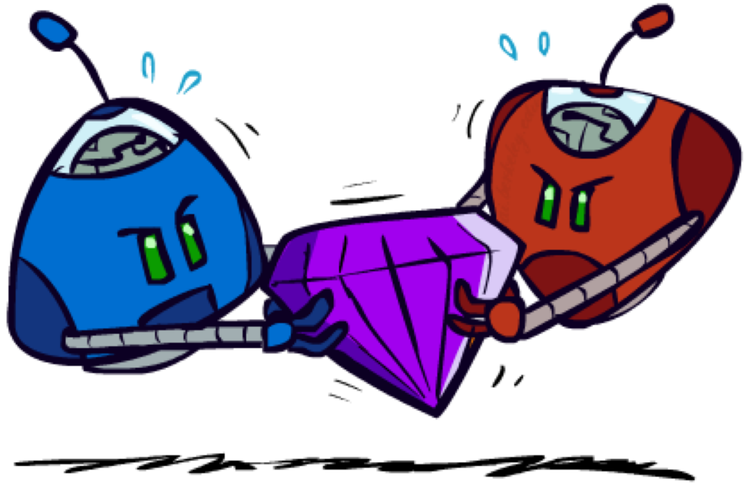
- Many different kinds of games
- Axes:
  - Deterministic vs. stochastic
  - One, two or more players
  - Zero sum vs. general sum
  - Perfect information (can you see the state) vs. partial information
- Algorithms need to calculate a strategy (**policy**) which recommends a move from each state

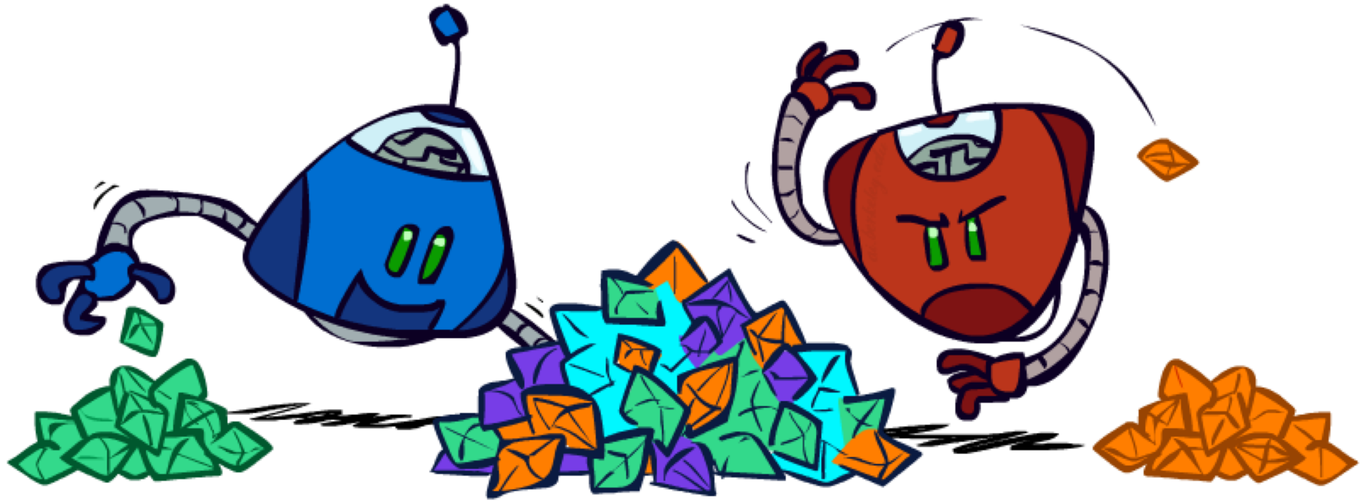
DETERMINISTIC GAMES

- Problem formulation:
  - States: S (start at s_0)
  - Players: P={1…N} (usually take turns)
  - Actions: A (may depend on player / state)
  - Transition Function: S x A → S
  - Terminal Test: S → {t,f}
  - Terminal Utilities: S x P → R
- Solution for a player is a **policy**: S → A



ZERO-SUM VS. GENERAL GAMES





- **Zero-Sum Games**
  - Agents have opposite *utilities* (values on outcomes)
  - Can then think of outcome as a single value that one maximizes and the other minimizes
  - Adversarial, pure competition
- **General Games**
  - Agents have independent *utilities* (values on outcomes)
  - Cooperation, indifference, competition, and more are all possible
  - More later on non-zero-sum games