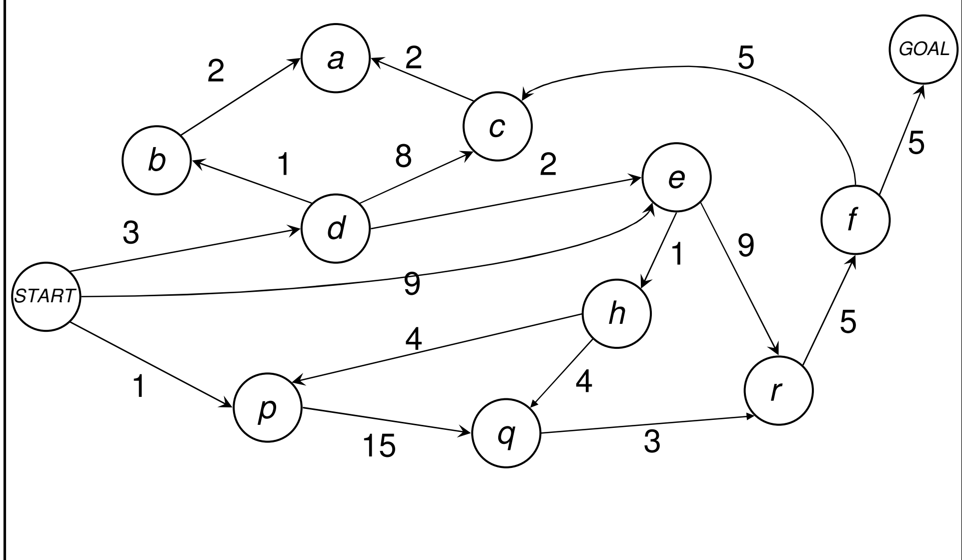
# Review

# Uninformed Search

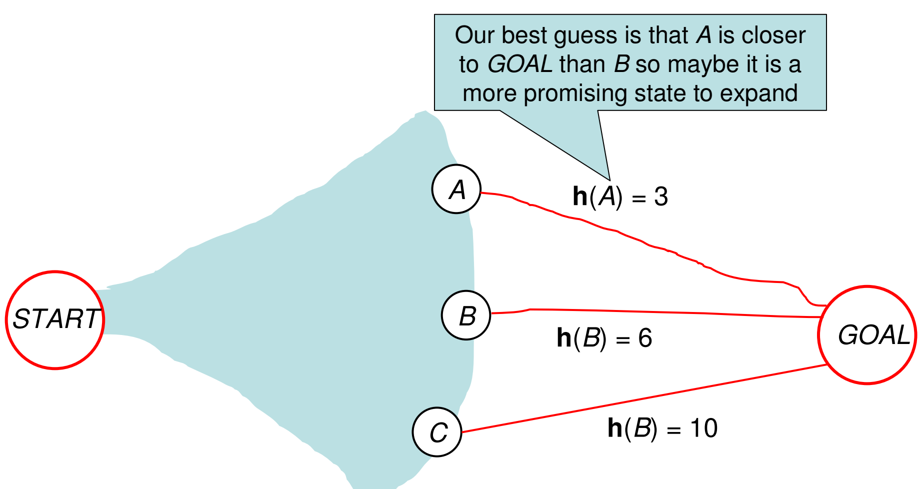


# Complexity

- $N$ = Total number of states
- $B$ = Average number of successors (branching factor)
- $L$ = Length for start to goal with smallest number of steps
- Q = Average size of the priority queue
- $Lmax$ = Length of longest path from *START* to any state

| | Algorithm | Complete | Optimal | Time | Space |
|---|---|---|---|---|---|
| BFS | Breadth First Search | Y | Y, If all trans. have same cost | $O(Min(N,B^L))$ | $O(Min(N,B^L))$ |
| BIBFS | Bi- Direction. BFS | Y | Y, If all trans. have same cost | $O(Min(N,2B^{L/2}))$ | $O(Min(N,2B^{L/2}))$ |
| UCS | Uniform Cost Search | Y, If cost > 0 | Y, If cost > 0 | $O(log(Q)*Min(N,B^L))$ | $O(Min(N,B^L))$ |
| PCDFS | Path Check DFS | Y | N | $O(B^{Lmax})$ | $O(BL_{max})$ |
| MEMDFS | Memorizing DFS | Y | N | $O(Min(N,B^{Lmax}))$ | $O(Min(N,B^{Lmax}))$ |
| IDS | Iterative Deepening | Y | Y, If all trans. have same cost | $O(B^L)$ | $O(BL)$ |

# Informed Search

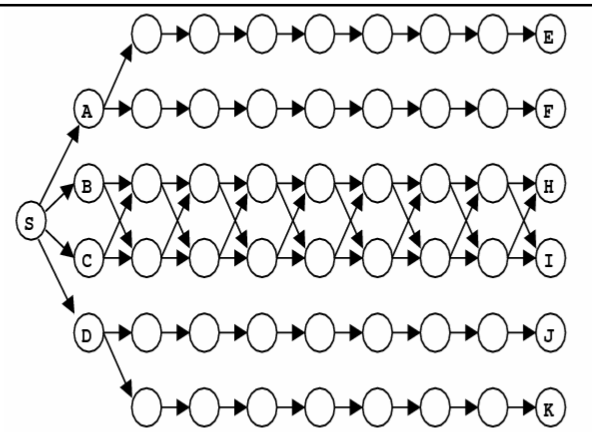


# Informed Search

- Best-First Search: Expand node with minimum *h*(*s*)
- A*: Expand node with minimum $f(s) = g(s) + h(s)$
- Guaranteed to be optimal if h admissible $h(s) <= h^*(s)$
- No nodes revisited if monotonic:

$$\mathbf{h}(s) < \mathbf{h}(s') + \mathbf{cost}(s,s')$$

- IDA* = Equivalent of Iterative Deepening for A* → guarantees low memory usage




**(a)** Find a goal location for which DFS does much less work than BFS and ID

**(b)** Find a goal location for which BFS and ID do much less work than DFS

**(c)** Find a goal location for which ID does much less work than BFS and DFS

**(d)** Find a goal location for which BFS does much less work than DFS and ID

$$f(s) = (2 - w)g(s) + wh(s)$$

## Most Basic Algorithm: Hill-Climbing (Greedy Local Search)

- X ← Initial configuration
- Iterate:
  1. $E$ ← *Eval*($X$)
  2. $\mathcal{N}$ ← *Neighbors*($X$)
  3. For each $X_i$ in $\mathcal{N}$
     $E_i$ ← *Eval*($X_i$)
  4. If all $E_i$'s are lower than $E$
     Return $X$
     Else
     $i^* = \text{argmax}_i\ (E_i)$ $X \leftarrow X_{i^*}$ $E \leftarrow E_{i^*}$

## Stochastic Search: Randomized Hill-Climbing

- $X \leftarrow$ Initial configuration
- Iterate:

Until when?

1. $E \leftarrow Eval(X)$
2. $X' \leftarrow$ one configuration randomly selected in *Neighbors* ($X$)
3. $E' \leftarrow Eval(X')$
4. If $E' > E$

   $X \leftarrow X'$

   $E \leftarrow E'$

Critical change: We no longer select the best move in the entire neighborhood

## Simulated Annealing

1. Do $K$ times:

   1.1 $E \leftarrow Eval(X)$

   1.2 $X' \leftarrow$ one configuration randomly selected in *Neighbors* ($X$)

   1.3 $E' \leftarrow Eval(X')$

   1.4 If $E' >= E$

   $X \leftarrow X'; E \leftarrow E';$

   Else accept the move with probability $p = e^{-(E - E')/T}$ :

   $X \leftarrow X'; E \leftarrow E';$

2. $T \leftarrow \alpha T$

# Basic GA Outline

- Create initial population $X = \{X_1,..,X_P\}$
- Iterate:
  1. Select $K$ random pairs of parents $(X,X')$
  2. For each pair of parents $(X,X')$:
     1.1 Generate offsprings $(Y_1, Y_2)$ using crossover operation
     1.2 For each offspring $Y_i$:
        Replace randomly selected element of the population by $Y_i$
        With probability $\mu$:
           Apply a random mutation to $Y_i$
- Return the best individual in the population

1. Let X :=initial object
2. Let E :=Eval(X)
3. Let X' :=randomly chosen configuration chosen from the moveset of X
4. Let E' :=Eval(X')
5. Let $z$:= a number drawn randomly uniformly between 0 and 1
6. If $E' > E$ or $\exp(-(E - E')/T) > z$ then
   - X :=X'
   - E :=E'
7. $T$:=$r \times T$
8. If a convergence test is satisfied then halt. Else go to Step 3.

(a) Normally $r$, the temperature decay rate, is chosen in the range $0 < r < 1$. How would the behavior of simulated annealing change if $r > 1$?

*The change will always be accepted and we'll do a random walk.*

(b) Alternatively, how would it change if $r = 0$?

(c) If we simplified the conditional test in Step 6 to

If $\exp(-(E - E')/T) > z$ then

how would the behavior of simulated annealing change?

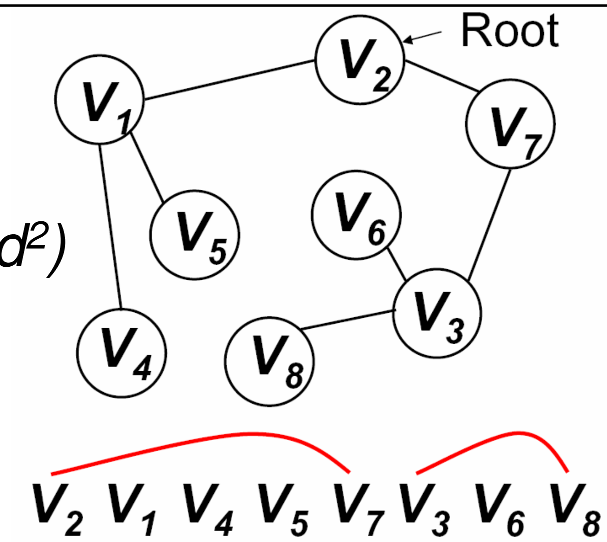
# CSP

- Definitions
- Standard search
- Improvements
  - Backtracking
  - Forward checking
  - Constraint propagation
- Heuristics:
  - Variable ordering
  - Value ordering
- Examples
- Tree-structured CSP
- Local search for CSP problems

# Constraint Propagation

- $A$ = queue of active arcs ($\boldsymbol{V}_i, \boldsymbol{V}_j$)
- Repeat while $A$ not empty:
  - ($\boldsymbol{V}_i, \boldsymbol{V}_j$) ← next element of $A$
  - For each $x$ in $D(\boldsymbol{V}_i)$:
    - Remove $x$ from $D(\boldsymbol{V}_i)$ if there is no $y$ in $D(\boldsymbol{V}_j)$ for which ($x$,$y$) satisfies the constraint between $\boldsymbol{V}_i$ and $\boldsymbol{V}_j$.
  - If $D(\boldsymbol{V}_i)$ has changed:
    - Add all the pairs ($\boldsymbol{V}_k, \boldsymbol{V}_i$), where $\boldsymbol{V}_k$ is a neighbor of $\boldsymbol{V}_i$ ($k$ not equal to $j$) to $A$

$O(N\,d^2)$




Order the variables such that the parent of a node appears always before that node in the list

- *Most Constraining Variable*
  - Selecting a variable which contributes to the *largest* number of constraints will have the largest effect on the other variables → Hopefully will prune a larger part of the search
  - This amounts to finding the variable that is connected to the largest number of variables in the constraint graph.
- *Minimum Remaining Values (MRV)*
  - Selecting the variable that has the least number of candidate values is most likely to cause a failure early ("fail-first" heuristic)
- *Least Constraining Value*
  - Choose the value which causes the smallest reduction in the number of available values for the neighboring variables

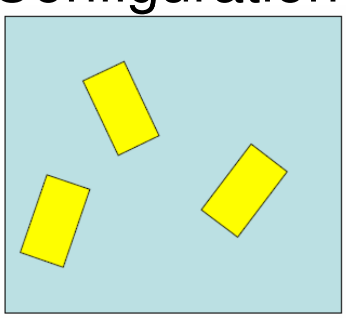
Consider the perennial problem of scheduling classrooms in Wean Hall. We have 4 instructors ($I1, I2, I3, I4$) and 3 rooms ($R1, R2, R3$). We need to assign rooms to instructors. We assume that the instructors need the rooms at the following times:

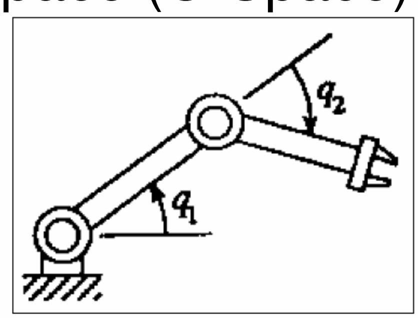
I1: 9am to 11am
I2: 10am to 2pm
I3: 1pm to 5pm
I4: 1pm to 3pm

We assume that a room can be use by only one instructor at a time and that room $R3$ is too small for instructor $I1$ and that rooms $R2$ and $R3$ are too small for instructor $I3$.

| Variable Instantiated | I1 | I2 | I3 | I4 |
|---|---|---|---|---|
| *Initial Domains* | R1,R2 | R1,R2 R3 | R1 | R1,R2 R3 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# Configuration Space (C-Space)



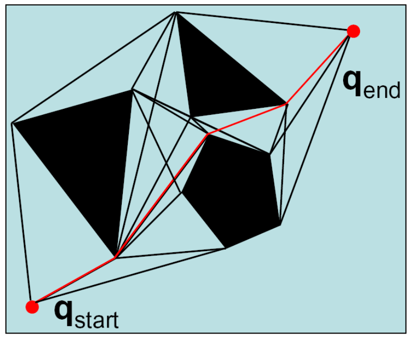



$\boldsymbol{q} = (x,y,\theta)$
$\mathcal{C} = \Re^2$ x set of 2-D rotations

$\boldsymbol{q} = (q_1,q_2)$
$\mathcal{C} =$ 2-D rotations x 2-D rotations

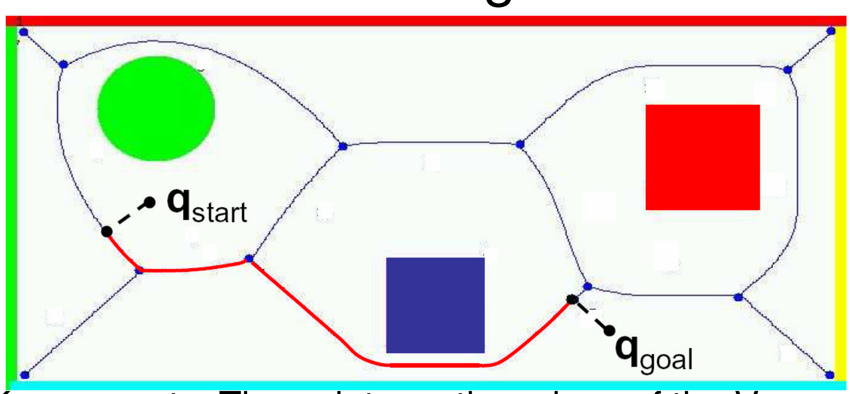
- Configuration space $\mathcal{C}$ = set of values of $\boldsymbol{q}$ corresponding to legal configurations of the robot
- Defines the set of possible parameters (the search space) and the set of allowed paths
- Grow the forbidden parts of C-Space → Can assume that robot is a point in C-Space

# Visibility Graphs



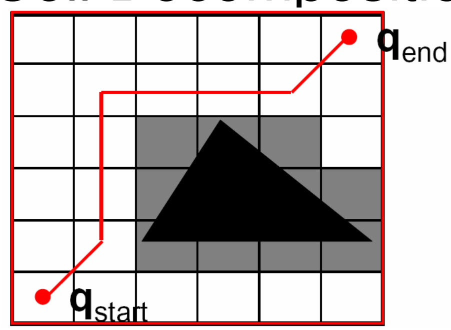

- $N$ = total number of vertices of the obstacle polygons
- Naïve: $O(N^3)$
- Sweep: $O(N^2 \log N)$
- Optimal: $O(N^2)$

# Voronoi Diagrams



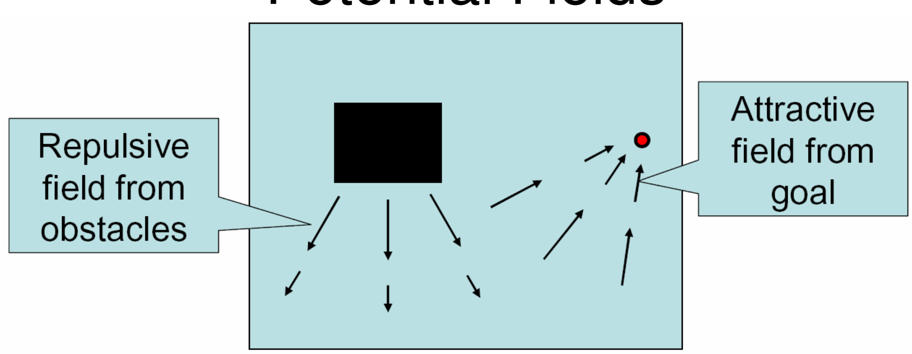

- Key property: The points on the edges of the Voronoi diagram are the *furthest* from the obstacles
- Idea: Construct a path between $\mathbf{q}_{start}$ and $\mathbf{q}_{goal}$ by following edges on the Voronoi diagram
- (Use the Voronoi diagram as a roadmap graph instead of the visibility graph)

# Cell Decomposition



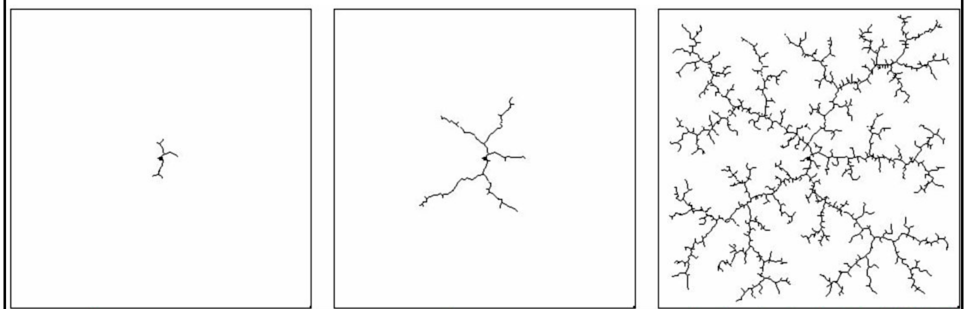

- Define cells in C-Space
- Mark any cell of the grid that intersects $\mathcal{C}_{obs}$ as blocked
- Find path through remaining cells by using (for example) A* (e.g., use Euclidean distance as heuristic)
- Approximate → Easy to compute but not *complete*
- Exact → Hard to compute but complete

# Potential Fields

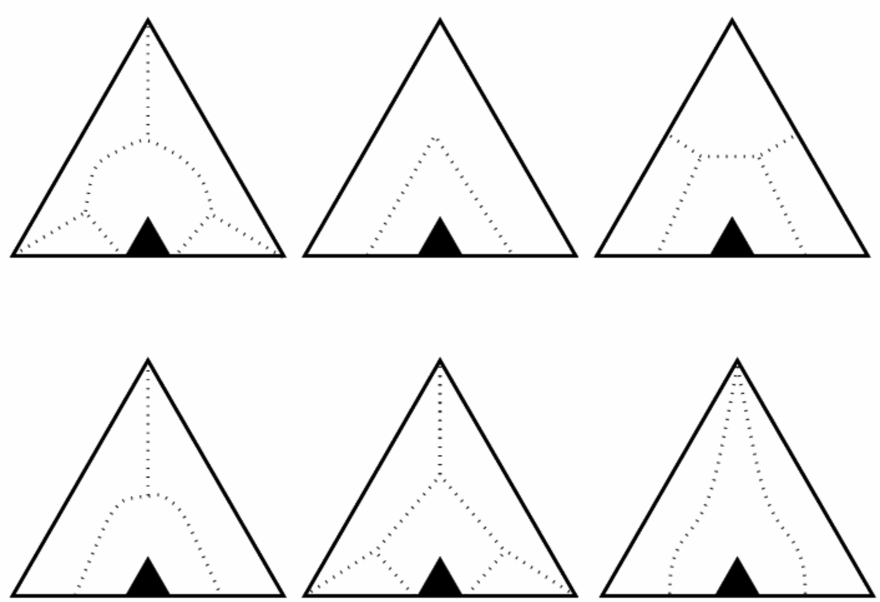



- Stay away from obstacles: Imagine that the obstacles are made of a material that generate a *repulsive* field
- Move closer to the goal: Imagine that the goal location is a particle that generates an *attractive* field
- Key issue: Local minima
  - Deterministic exploration of local minima
  - Stochastic exploration
- In very high dimension → Randomize traversal of neighbors

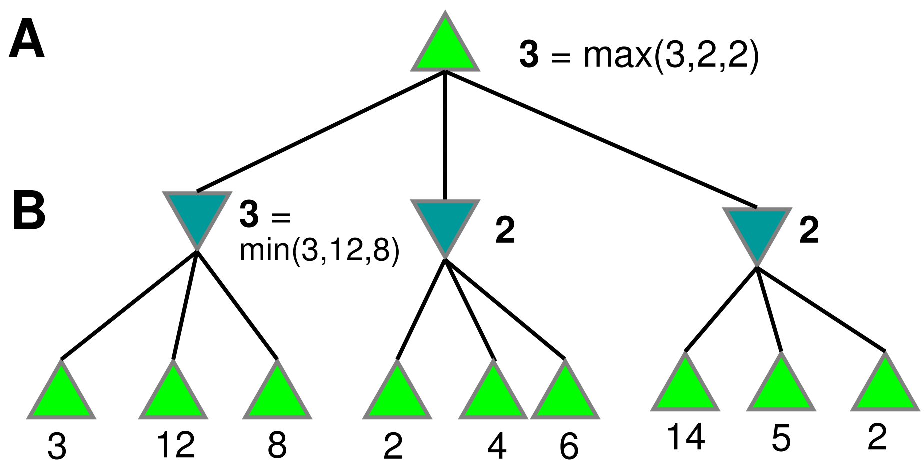
# Sampling Techniques



- Tends to explore the space rapidly in all directions
- Does not require extensive pre-processing
- Single query/multiple query problems
- Needs only collision detection test → No need to represent/pre-compute the entire C-space
- For a large class of problems:
  - Prob(finding a path) → 1 exponentially with the number of samples
- *But*, cannot detect that a path does not exist



Games




# Minimax

Minimax ($s$)

If $s$ is terminal

Return $U(s)$

If next move is $A$

Return $\max_{s' \in Succs(s)} \text{Minimax}(s')$

Else

Return $\min_{s' \in Succs(s)} \text{Minimax}(s')$

# Minimax Properties

- *Complete:* If finite game
- *Optimal:* If opponent plays optimally
- *Complexity:* Essentially DFS, so:
  - Time: O($B^m$)
  - Space: O($Bm$)
  - $B$ = number of possible moves from any state (branching factor)
  - $m$ = depth of search (length of game)
- Pruning ($\alpha\beta$):
  - Guaranteed to find same solution
  - O($B^{m/2}$) with proper ordering of the nodes → At "A" node, the successor are in order from high to low score

# Non-Deterministic Minimax

Minimax ($s$)

If $s$ is terminal

Return $U(s)$

If next move is $A$: Return $\boxed{\max_{s' \in Succs(s)} \text{Minimax}(s')}$

If next move is $B$ Return $\boxed{\min_{s' \in Succs(s)} \text{Minimax}(s')}$

If chance node Return $\boxed{\sum_{s' \in Succs(s)} p(s')\text{Minimax}(s')}$

Max value =
game value = +2

Note that we find the same value and same strategies in both cases. Is that always the case?

| | | I | II | III | IV |
|---|---|---|---|---|---|
| -1 | I | -1 | -1 | +2 | +2 |
| +2 | II | +4 | +4 | +2 | $\boxed{+2}$ |
| +1 | III | +5 | +1 | +5 | +1 |
| +1 | IV | +5 | +1 | +5 | +1 |

Min value across each row

$$\underset{Rows\ i}{Max}\ \underset{Columns\ j}{Min}\ \mathbf{M}(i,j)$$

Max value across each column

| | I | II | III | IV |
|---|---|---|---|---|
| I | -1 | -1 | +2 | +2 |
| II | +4 | +4 | +2 | $\boxed{+2}$ |
| III | +5 | +1 | +5 | +1 |
| IV | +5 | +1 | +5 | +1 |
| | +5 | +4 | +5 | +2 |

Min value =
game value = +2

$$\underset{Columns\ j}{Min}\ \underset{Rows\ i}{Max}\ \mathbf{M}(i,j)$$

# Minimax vs. Maximin

- Fundamental Theorem I (von Neumann):
  - For a two-player, zero-sum game with perfect information:
    - *There always exists an optimal pure strategy for each player*
    - *Minimax = Maximin*

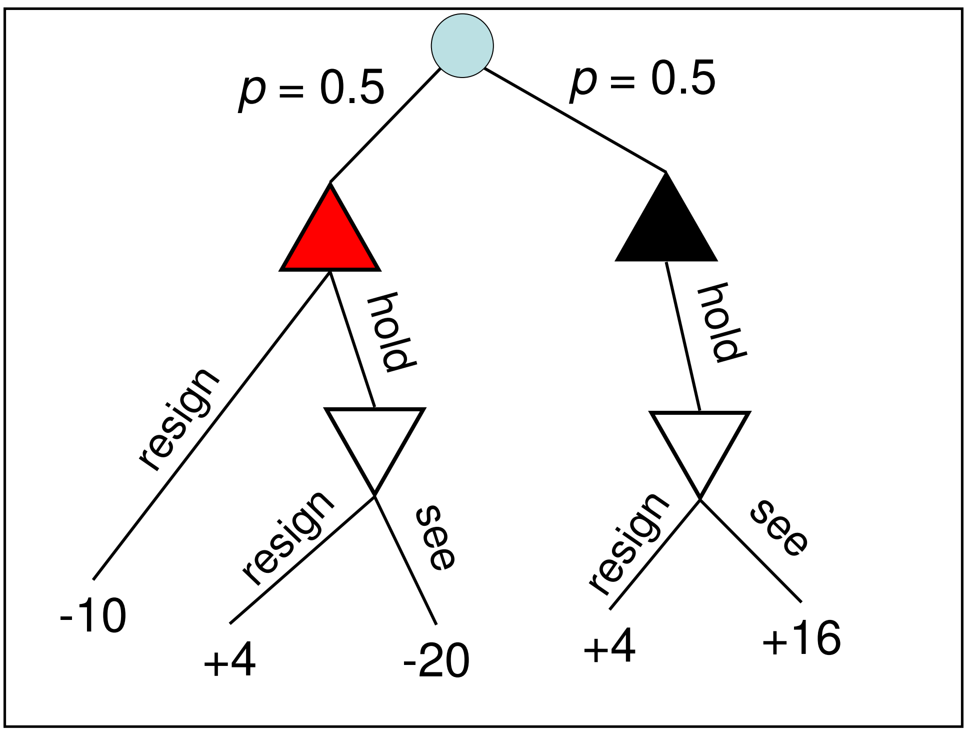
- *Note: This is a game-theoretic formalization of the minimax search algorithm that we studied earlier.*

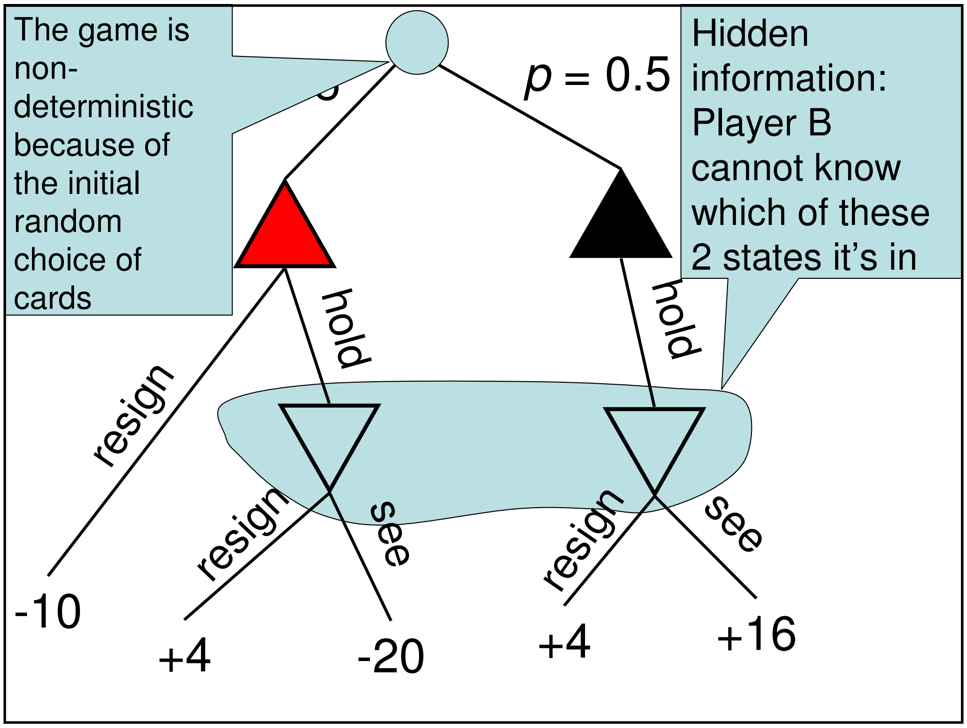
## Minimax with Mixed Strategies

- Theorem II (von Neumann):
  – For a two-player, zero-sum game with hidden information:
    - *There always exists an optimal mixed strategy*
    - *In addition, just like for games with perfect information, it does not matter in which order we look at the players, minimax is the same as maximin*

$$\max_{\boldsymbol{p}} \min(\boldsymbol{p}\times \boldsymbol{m}_{11} + (1-\boldsymbol{p})\times \boldsymbol{m}_{21}, \boldsymbol{p}\times \boldsymbol{m}_{12} + (1-\boldsymbol{p})\times \boldsymbol{m}_{22}) =$$
$$\min_{\boldsymbol{q}} \max(\boldsymbol{q}\times \boldsymbol{m}_{11} + (1-\boldsymbol{q})\times \boldsymbol{m}_{12}, \boldsymbol{q}\times \boldsymbol{m}_{21} + (1-\boldsymbol{q})\times \boldsymbol{m}_{22})$$

- *Note: This is a direct generalization of the minimax result to mixed strategies.*






**Player B**

**Player A**

| | Resign | See |
|---|---|---|
| Resign | | |
| Hold | | |

- Generate the matrix form of the game (be careful: It's not a deterministic game)
- Find the optimal mixed strategy
- Find the expected payoff for Player A

## Strategies

| | | | | |
|---|---|---|---|---|
| I | ~~3,0~~ | ~~4,1~~ | ~~5,9~~ | ~~5,6~~ |
| II | 5,3 | 5,8 | 9,7 | 9,0 |
| III | 2,3 | 8,4 | 6,2 | 6,3 |
| IV | ~~3,8~~ | ~~3,1~~ | ~~2,3~~ | ~~4,5~~ |

## Pure Strategy Nash Equilibrium

$$u_i\left(s_1^*,\cdots,s_{i-1}^*,s_i\ ,s_{i+1}^*,\cdots,s_n^*\right)\le u_i\left(s_1^*,\cdots,s_{i-1}^*,s_i^*,s_{i+1}^*,\cdots,s_n^*\right)$$

$$s_i^* = \underset{s_i}{\arg\max}\, u_i\left(s_1^*,\cdots,s_{i-1}^*,s_i\ ,s_{i+1}^*,\cdots,s_n^*\right)$$

- Does not always exist
- Is not always unique
- For continuous games:

$$\frac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{s}_i}\left(\boldsymbol{s}_1^*,\cdots,\boldsymbol{s}_n^*\right)=0$$

## Mixed Strategy Nash Equilibrium

$$\bar{u}_A(p, q^*) \le \bar{u}_A(p^*, q^*)$$

$$p^* = \arg\max_{p} \bar{u}_A(p, q^*)$$

- Player A plays strategy $s_i$ with probability $p_i$
- For non-zero games → Mixed Nash equilibrium *always* exists
- Is not always unique

$t_B$=*meet*

| | Hockey | Movie |
|---|---|---|
| **Hockey** | **+2,+1** | **0,0** |
| **Movie** | **0,0** | **+1,+2** |

$t_B$=*avoid*

| | Hockey | Movie |
|---|---|---|
| **Hockey** | **+2,0** | **0,+2** |
| **Movie** | **0,+1** | **+1,0** |

$P(t_A=meet \mid t_B=meet) = 1$ $P(t_B=meet \mid t_A=meet) = 1/2$
$P(t_A=meet \mid t_B=avoid) = 1$ $P(t_B=meet \mid t_A=avoid) = 1/2$
$P(t_A=avoid \mid t_B=meet) = 0$ $P(t_B=avoid \mid t_A=meet) = 1/2$
$P(t_A=avoid \mid t_B=avoid) = 0$ $P(t_B=avoid \mid t_A=avoid) = 1/2$

Payoff if Player A knows that Player B is of type $t_B$

Probability that Player B is indeed of type $t_B$

$$\bar{u}_A = \sum_{\substack{\text{all possible types} \\ t_B \text{ of Player B}}} u_A\big(s_A(t_A), s_B(t_B)\big)\mathrm{P}(t_B \mid t_A)$$

Since Player A does not know Player B's type, it has to sum over all possible types to get the expected value

| Player 1 | | D | E | F |
|---|---|---|---|---|
| | A | 0, 1 | 3, 5 | 2, 1 |
| | B | 6, 3 | 1, 3 | 5, 2 |
| | C | 4, 2 | 3, 4 | 7, 7 |

Player 2

| Player 1 | C | D |
|---|---|---|
| A | 2 | 0 |
| B | 0 | 1 |

Now consider a very different game. Two companies, A and B, both make elbow warmers. The more they spend on advertising, the more sales they get, but there are diminishing returns. A's advertising somewhat helps B, and B's advertising somewhat helps A. the exact fomulas are

$$
\begin{aligned}
P_a &= \overbrace{\log(2a+b)}^{revenue} - \overbrace{a}^{expense} \\
P_b &= \log(a+2b) - b
\end{aligned}
$$

where

$$
\begin{aligned}
\text{a} &= \text{\# dollars that A spends on advertising} \\
\text{b} &= \text{\# dollars that B spends on advertising} \\
P_a &= \text{profit to A} \\
P_b &= \text{profit to B}
\end{aligned}
$$