Admin

Quiz 2 posted, deadline Friday of reading week

- Individual submissions

Please start early and attend office hours next week if needed

- Email, canvas (preferred) is also fine

I will **not** be around on 17 Nov (1st day of reading week)

- No office hours, email replies may be delayed

Admin

Quiz 2 posted, deadline Friday of reading week

- Individual submissions

Please start early and attend office hours next week if needed

- Email, canvas (preferred) is also fine

I will **not** be around on 17 Nov (1st day of reading week)

- No office hours, email replies may be delayed

Course feedback

- You should have gotten email about this by now
- Please try to complete, this affects my teaching reviews ☹

Lecture 12: Markov Games & other topics

How many of you are familiar with

How many of you are familiar with

Markov Decision Processes?

How many of you are familiar with

Markov Decision Processes?

Partially Observable Markov Decision Processes?

How many of you are familiar with

Markov Decision Processes?

Partially Observable Markov Decision Processes?

My opinion

- Much less useful in the multiagent setting than what is purported
- Perhaps I'm biased?

Review: MDPs

Review: MDPs

$M = (S, A, T, R, \gamma)$

- States, Actions, Transitions, Reward, discount
- Very common, traditional model
- $T(s, a, s')$ is the transition probability from moving from state $s \rightarrow s'$ given action a is taken
- $R(s)$ is a reward, can be $R(s, a)$ or $R(s, a, s')$

Review: MDPs

$M = (S, A, T, R, \gamma)$

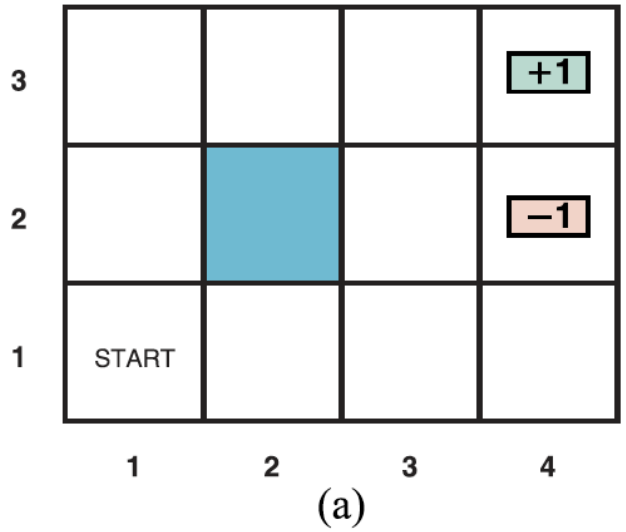
- States, Actions, Transitions, Reward, discount
- Very common, traditional model
- $T(s, a, s')$ is the transition probability from moving from state $s \rightarrow s'$ given action a is taken
- $R(s)$ is a reward, can be $R(s, a)$ or $R(s, a, s')$

Policy $\pi(s) = a$

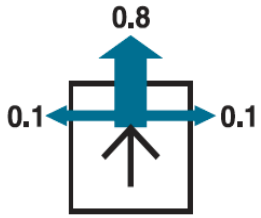
- Optimal policy π^*, maps states to action, optimizes for expected total discounted reward $r_0 + \gamma r_1 + \gamma r_2$ …, bounded reward for nontrivial discounting

Important: transitions, rewards etc are all **Markovian**

Example MDP: Navigation in Grid World



(a)

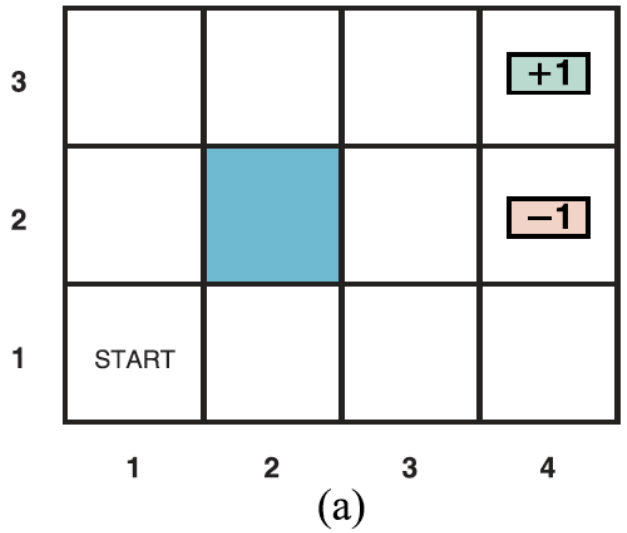


(b)

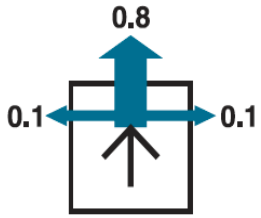
Source: RN Figure 17.1

Example MDP: Navigation in Grid World

Fully observable environment



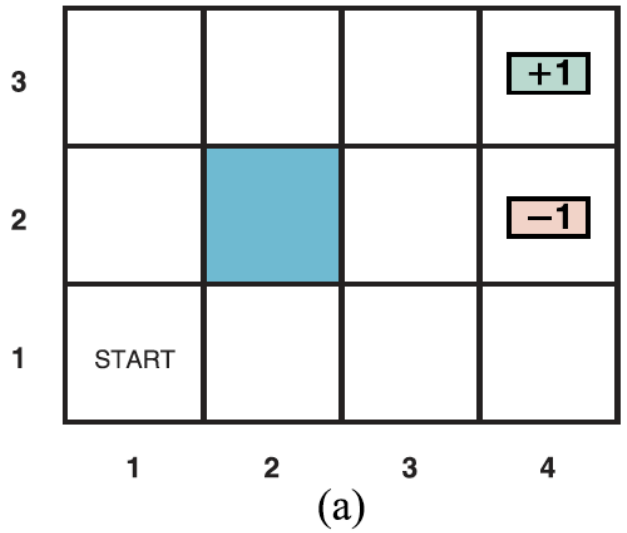
(a)

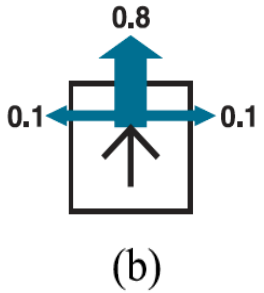


(b)

Source: RN Figure 17.1

Example MDP: Navigation in Grid World





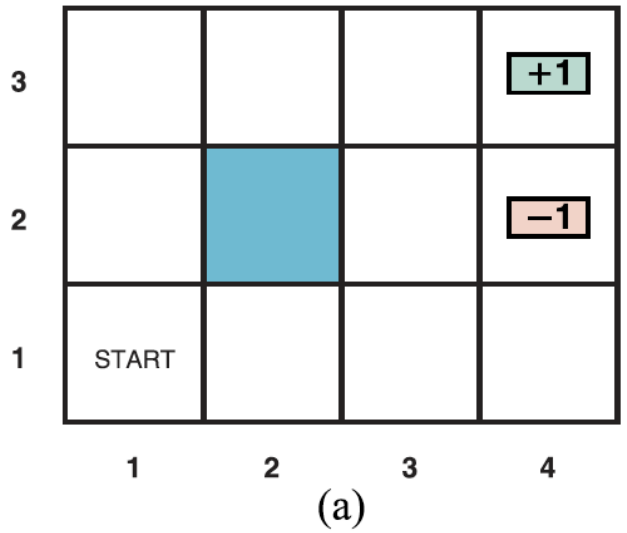
Source: RN Figure 17.1

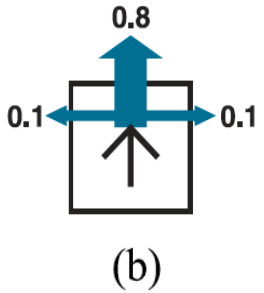
Fully observable environment

States: 4x3 grid (where your robot can be)

- Initial state: (1,1) , terminal states: marked with +1/-1

Example MDP: Navigation in Grid World





Source: RN Figure 17.1

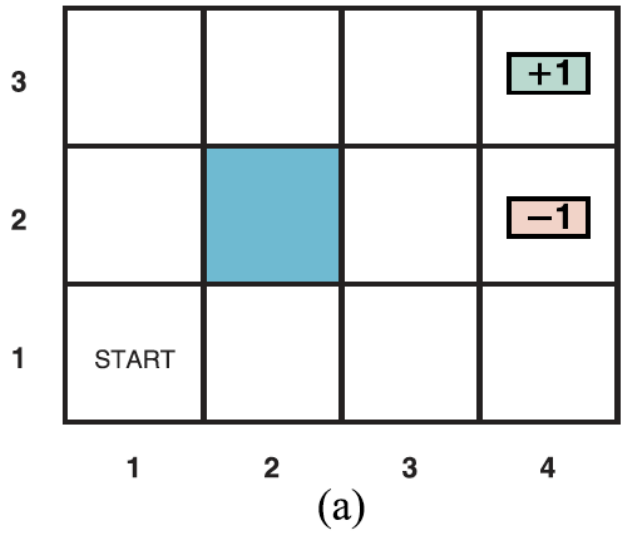
Fully observable environment

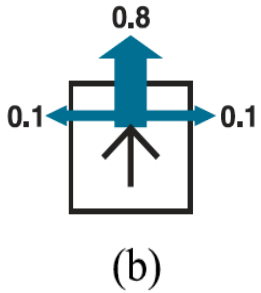
States: 4x3 grid (where your robot can be)

- Initial state: (1,1) , terminal states: marked with +1/-1

Actions: Up, Down, Left, Right

Example MDP: Navigation in Grid World





Source: RN Figure 17.1

Fully observable environment

States: 4x3 grid (where your robot can be)

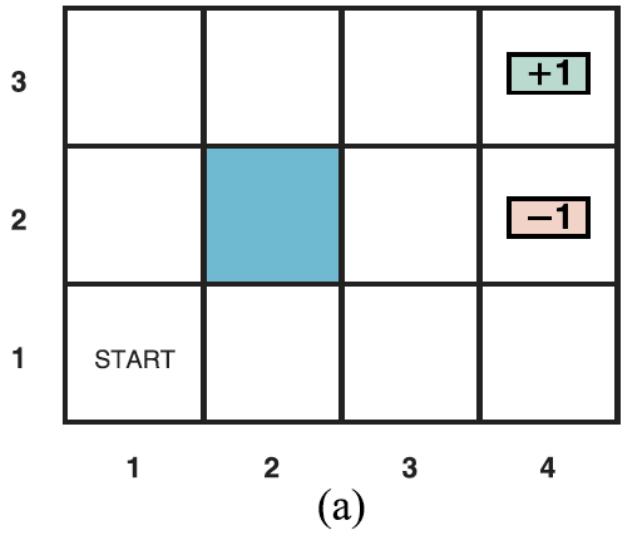
- Initial state: (1,1) , terminal states: marked with +1/-1

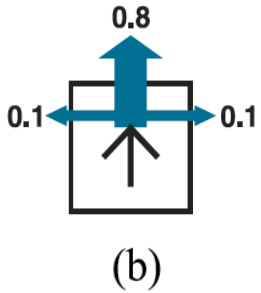
Actions: Up, Down, Left, Right

Transitions:

- Uncertain: 80% as intended, 10% perpendicular
  - Stay put if colliding with wall/obstacle
- Only depends on the current state, not history of past states (markov property)

Example MDP: Navigation in Grid World





Source: RN Figure 17.1

Fully observable environment

States: 4x3 grid (where your robot can be)

- Initial state: (1,1) , terminal states: marked with +1/-1

Actions: Up, Down, Left, Right

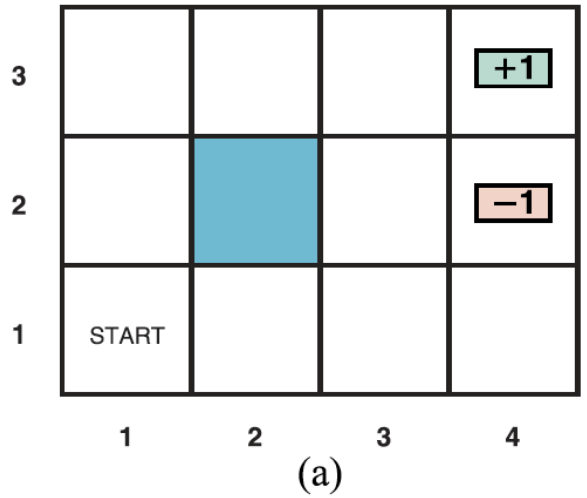
Transitions:

- Uncertain: 80% as intended, 10% perpendicular
  - Stay put if colliding with wall/obstacle
- Only depends on the current state, not history of past states (markov property)

Rewards:

- +1/-1 at terminal states
- + potentially some cost/reward at all other states (not shown)

Some intuition for solutions



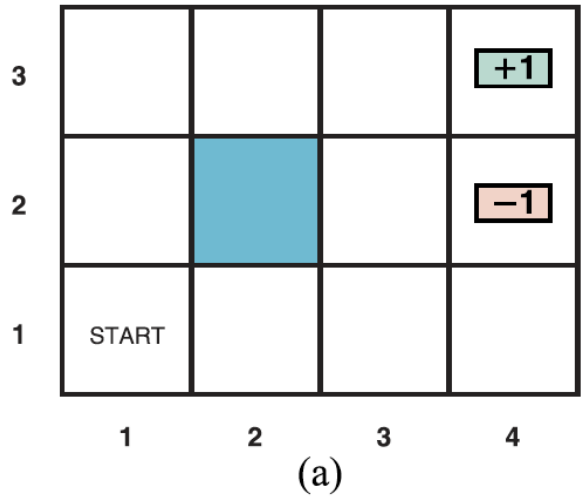
(a)

$r_{non-terminal} = -0.001$

Very **small** penalty if the game is not over

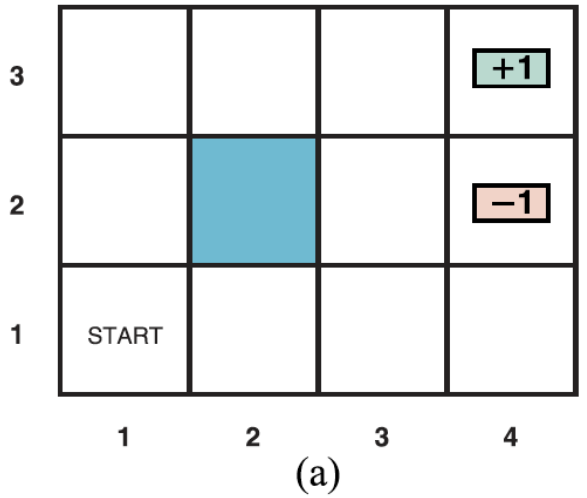
Source: RN Figure 17.1

Some intuition for solutions



$r_{non-terminal} = -0.001$

Very **small** penalty if the game is not over

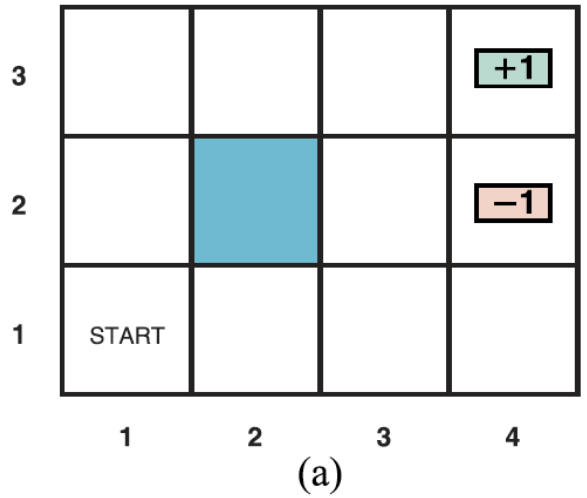


$r_{non-terminal} = -1$

Very **large** penalty if game is not over

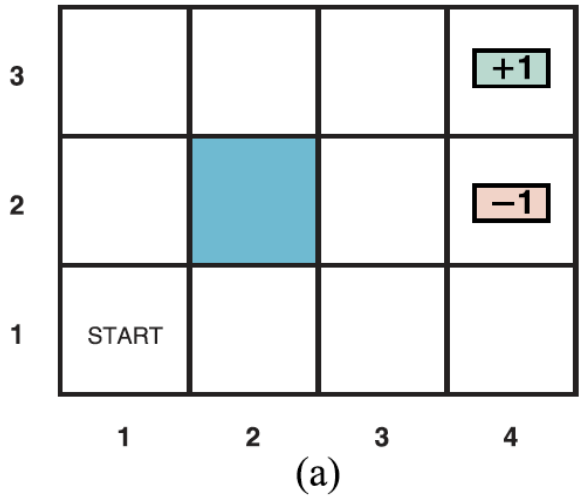
Source: RN Figure 17.1

Some intuition for solutions



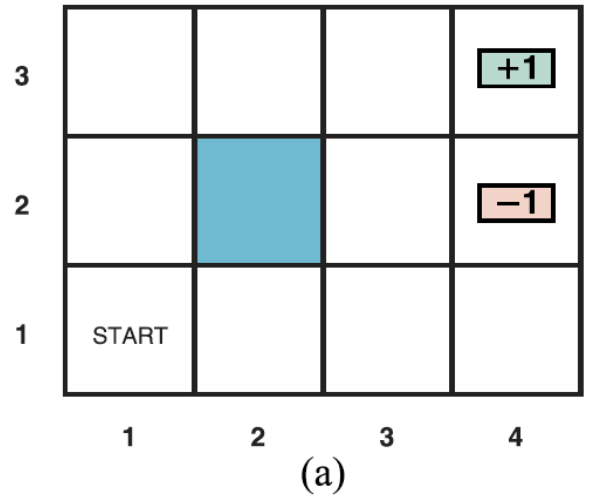
$r_{non-terminal} = -0.001$

Very **small** penalty if the game is not over



$r_{non-terminal} = -1$

Very **large** penalty if game is not over

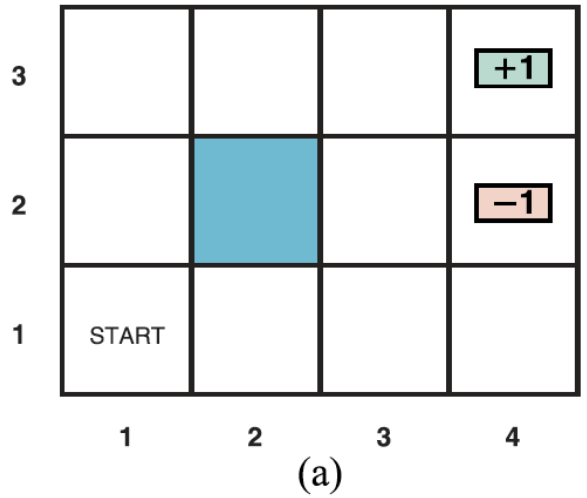


$r_{non-terminal} = +0.1$

Get **rewarded** for dragging things out

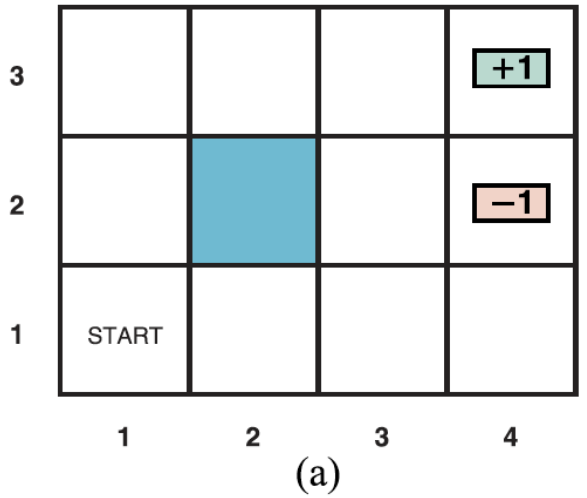
Source: RN Figure 17.1

Some intuition for solutions



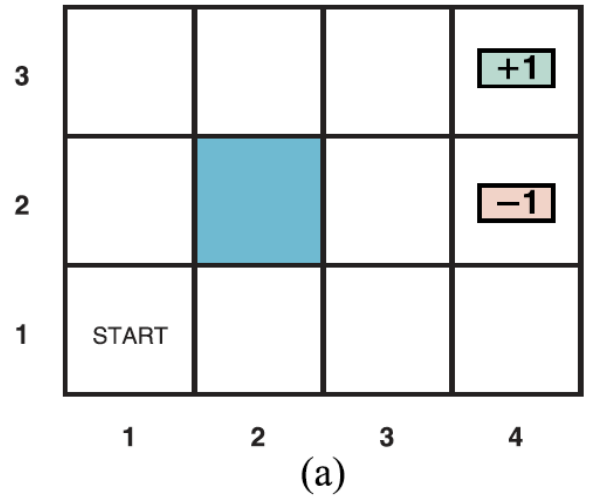
$r_{non-terminal} = -0.001$

Very **small** penalty if the game is not over



$r_{non-terminal} = -1$

Very **large** penalty if game is not over



$r_{non-terminal} = +0.1$

Get **rewarded** for dragging things out

How do we do this systemically?

Source: RN Figure 17.1

Value Iteration (Infinite Horizon)

Value Iteration (Infinite Horizon)

$$V^*(s) = \max_{a \in A} [R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s')]$$

Self referencing **fixed point** equations!

Value Iteration (Infinite Horizon)

$$V^*(s) = \max_{a \in A} [R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s')]$$

Self referencing **fixed point** equations!

Algorithm: Fixed point iteration

- Pretend we have an extremely long horizon
- Perform dynamic programming

Initialize $V_0^*(s) \leftarrow 0$

Iterate $V_{i+1}^*(s) \leftarrow \max_{a \in A} [R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_i^*(s')]$

- aka 'Value update', 'Bellman backups/updates'

Value Iteration (Infinite Horizon)

$$V^*(s) = \max_{a \in A} [R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s')]$$

Self referencing **fixed point** equations!

Algorithm: Fixed point iteration

- Pretend we have an extremely long horizon
- Perform dynamic programming

Initialize $V_0^*(s) \leftarrow 0$

Iterate $V_{i+1}^*(s) \leftarrow \max_{a \in A} [R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_i^*(s')]$

- aka 'Value update', 'Bellman backups/updates'

Q-function: $Q^\pi(s,a)$,

- Take action a at state s, follow π thereafter. What is total expected discounted reward?
- Q function of optimal policy implies optimal policy (one step lookahead)

Nice properties in the 1-Player case

Nice properties in the 1-Player case

π^* can be only dependent on **current state**

- No need to to worry about histories, time index etc
- Markovian dynamics →only need to consider Markovian strategies
- Note: there are other more complex objectives where this does not necessarily hold true (see the “reward hypothesis”)

Nice properties in the 1-Player case

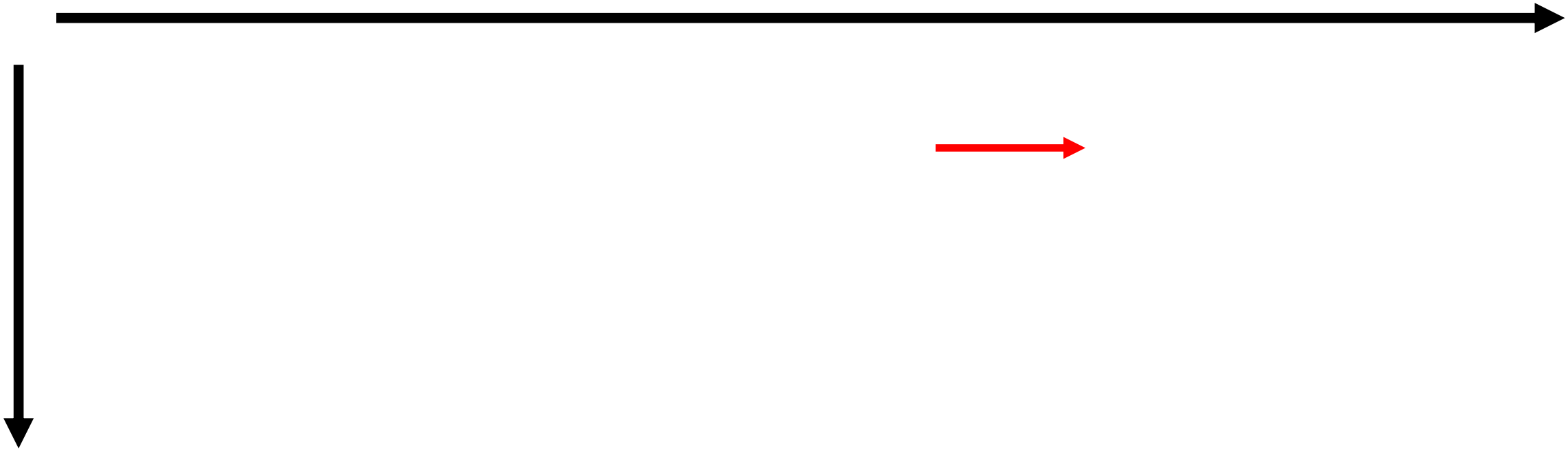
π^* can be only dependent on **current state**

- No need to to worry about histories, time index etc
- Markovian dynamics →only need to consider Markovian strategies
- Note: there are other more complex objectives where this does not necessarily hold true (see the "reward hypothesis")

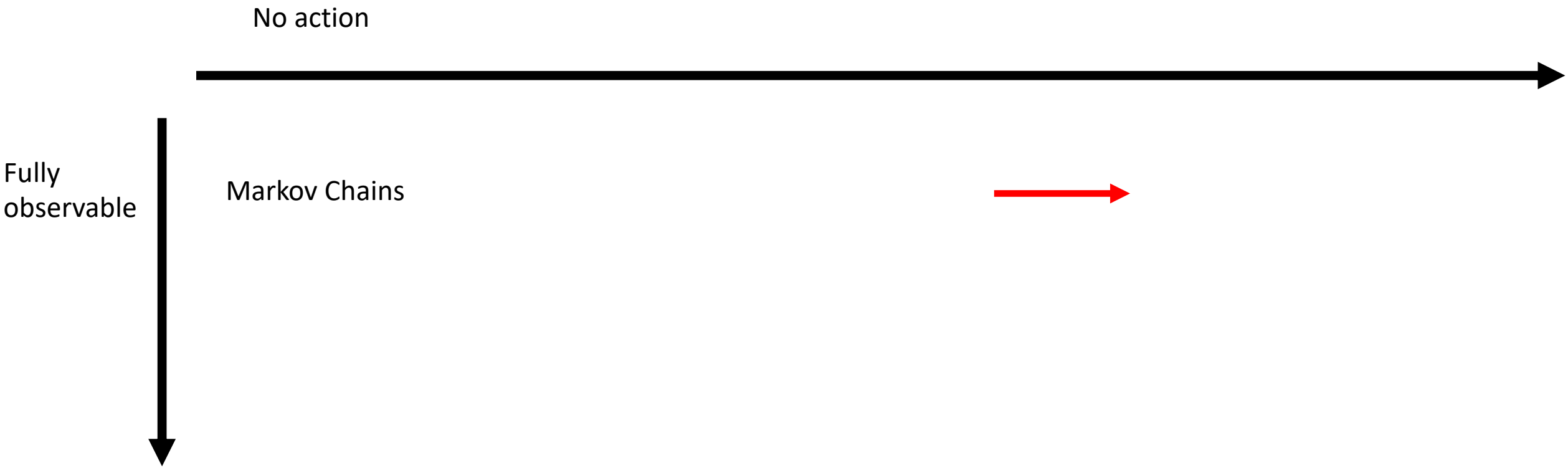
π^* can always be **deterministic**

- Just "search" deterministic strategies
- Optimal action is just to take the argmax of Q-function
- $\arg\max_{a \in A} [Q^*(s,a)]$
- $\arg\max_{a \in A} [R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_i^*(s')]$
- Again, not necessarily true for all objectives

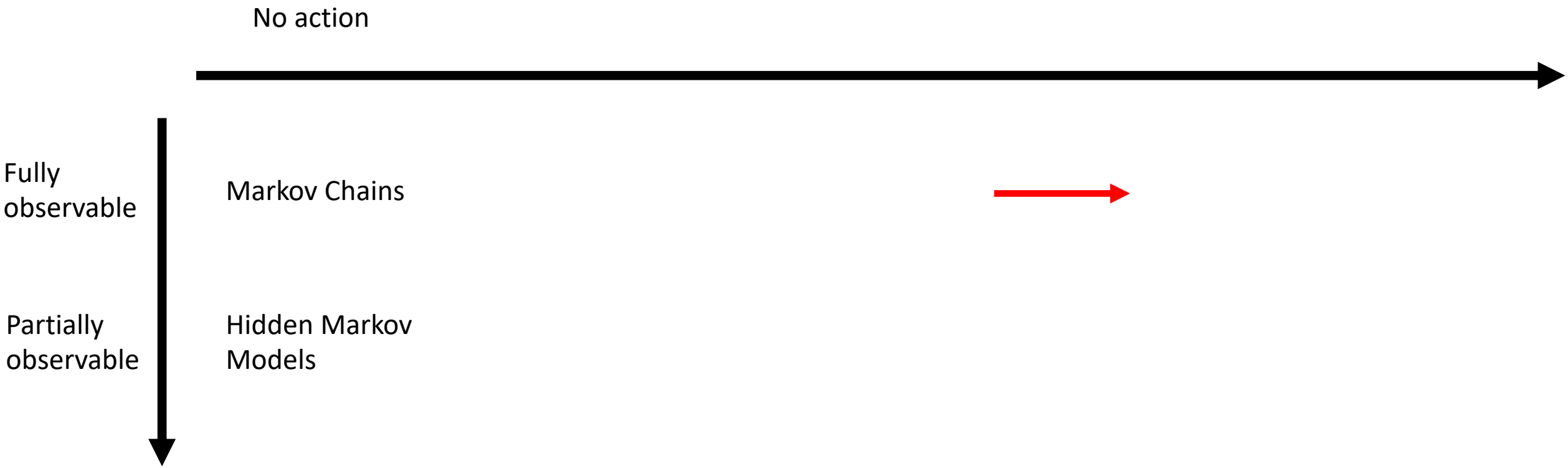
Relationship to other Markov Models



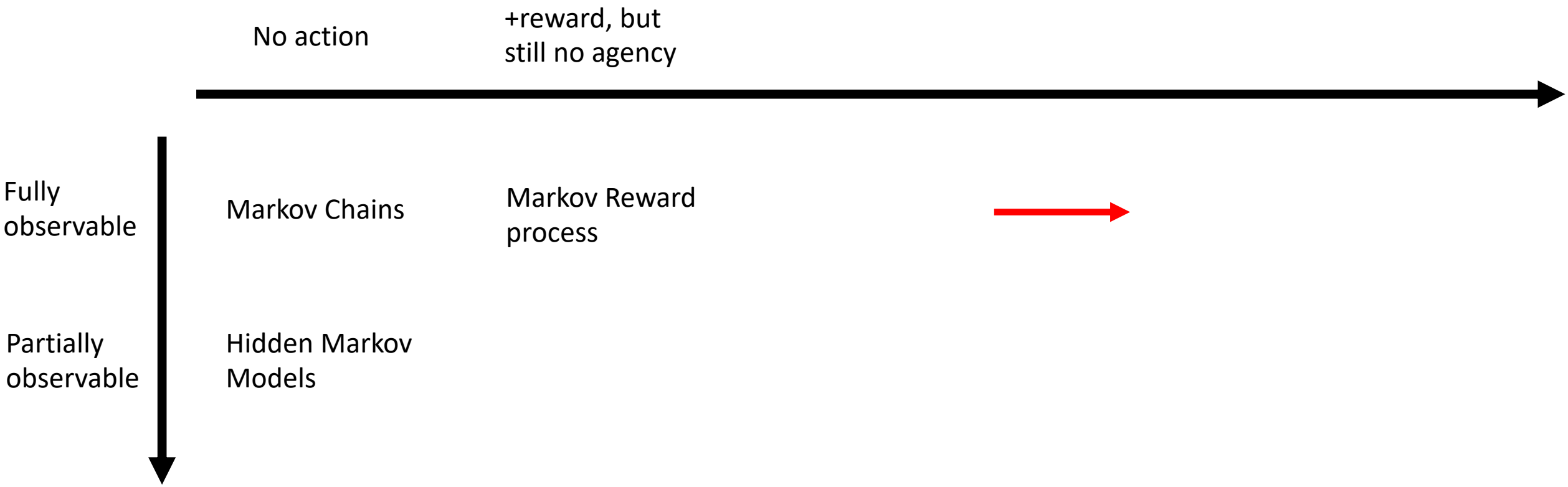
Relationship to other Markov Models



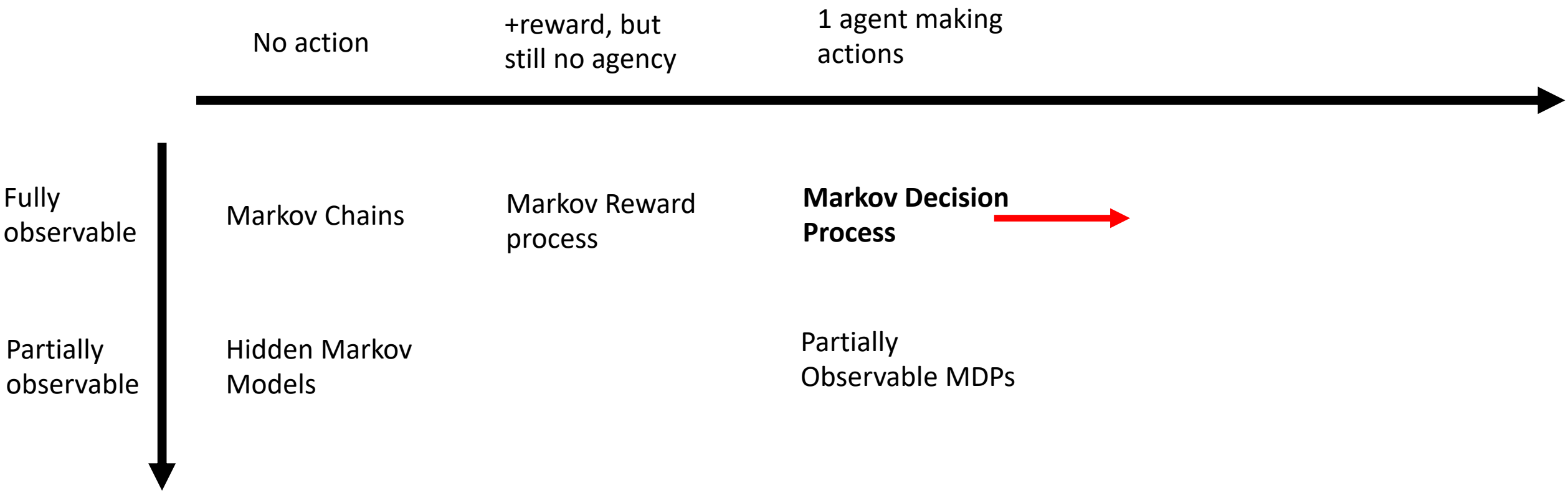
Relationship to other Markov Models



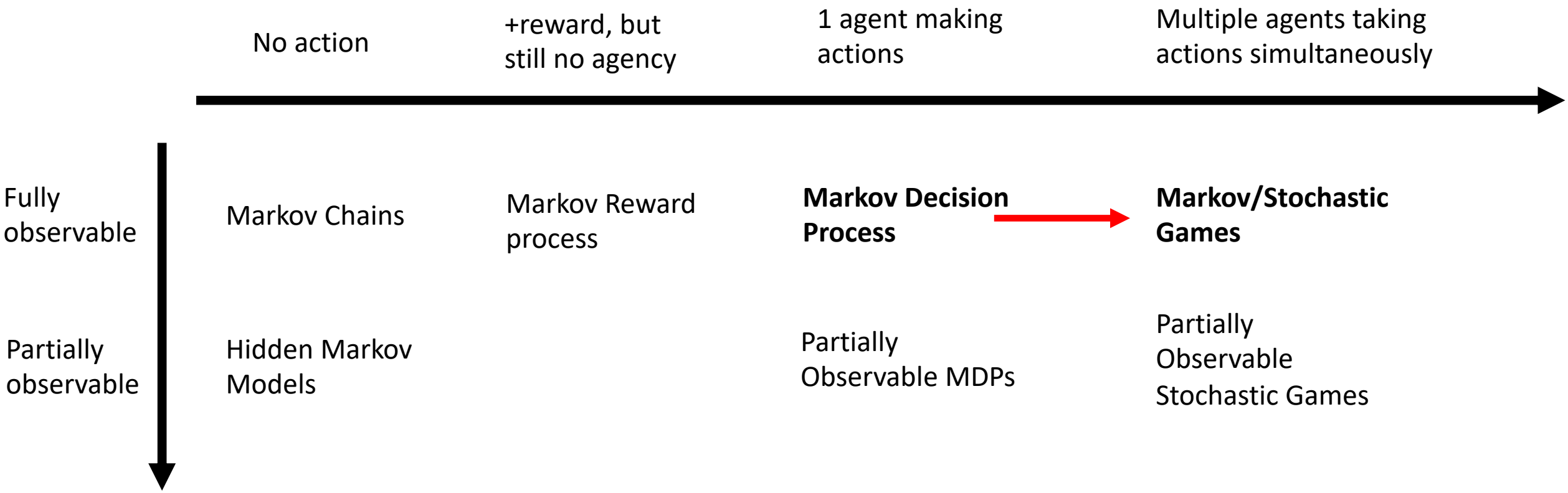
Relationship to other Markov Models



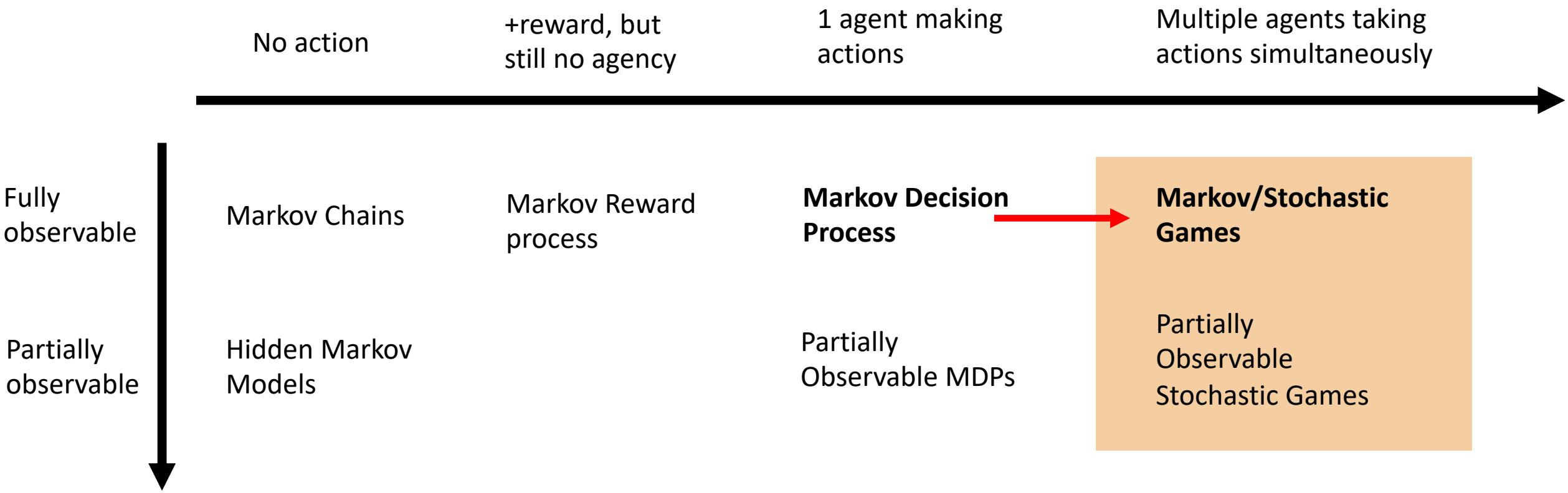
Relationship to other Markov Models



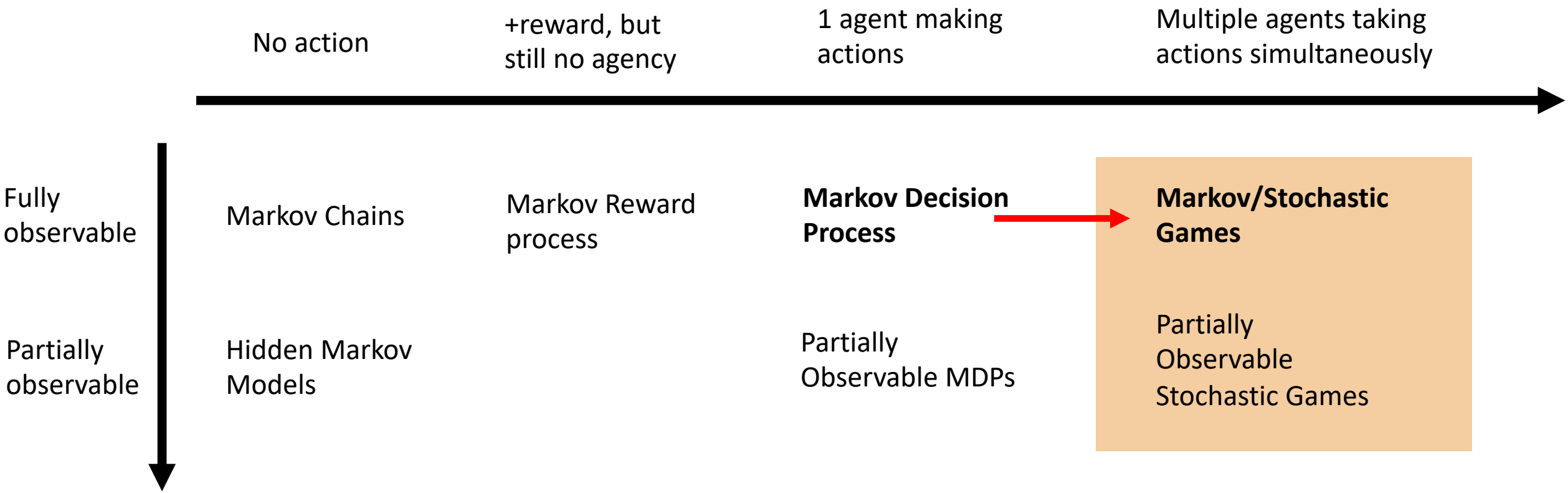
Relationship to other Markov Models



Relationship to other Markov Models



Relationship to other Markov Models



Many other dimensions possible, e.g., constraints, relaxing Markovianity etc.

What changes when we add more players?

What changes when we add more players?

$S, A_1, A_2, R_1(s, a_1, a_2 s'), R_2(s, a_1, a_2 s'), T(s, a_1, a_2, s'), \gamma$

- **Fully observable** (more on some nuances about this later), but at least current state is observed by each player
- Other special versions have one state-per player s_1, s_2 (factored variants), we won't consider them here

What changes when we add more players?

$S, A_1, A_2, R_1(s, a_1, a_2 s'), R_2(s, a_1, a_2 s'), T(s, a_1, a_2, s'), \gamma$

- **Fully observable** (more on some nuances about this later), but at least current state is observed by each player
- Other special versions have one state-per player s_1, s_2 (factored variants), we won't consider them here

Cooperative

- Aren't the players just one giant player?

Competitive/Zero-sum [Shapeley, 1953]

- Stationary equilibria always exists

General-sum [Fink, 1964]

- Stationary equilibria also always exists

What changes when we add more players?

$S, A_1, A_2, R_1(s, a_1, a_2 s'), R_2(s, a_1, a_2 s'), T(s, a_1, a_2, s'), \gamma$

- **Fully observable** (more on some nuances about this later), but at least current state is observed by each player
- Other special versions have one state-per player s_1, s_2 (factored variants), we won't consider them here

Cooperative

- Aren't the players just one giant player?

Competitive/Zero-sum [Shapeley, 1953]

- Stationary equilibria always exists

General-sum [Fink, 1964]

- Stationary equilibria also always exists

What about our algorithms?

- Stick to tabular ones for now…

The 2p0s case

Players move simultaneously but have common knowledge as to which state they are in

- Each player is best-responding to other player's policy. Note: BR can always be deterministic (why?)

The 2p0s case

Players move simultaneously but have common knowledge as to which state they are in

- Each player is best-responding to other player's policy. Note: BR can always be deterministic (why?)

Can show that **optimal strategy is Markovian/Stationary**

- **Warning!** This is tricky. Some *define* a strategy as being stationary to begin with.
- Others do not and show strategic equivalence between stationary and non-stationary strategies
- In 2p0s Markov games, it is the latter.

The 2p0s case

Players move simultaneously but have common knowledge as to which state they are in

- Each player is best-responding to other player's policy. Note: BR can always be deterministic (why?)

Can show that **optimal strategy is Markovian/Stationary**

- **Warning!** This is tricky. Some *define* a strategy as being stationary to begin with.
- Others do not and show strategic equivalence between stationary and non-stationary strategies
- In 2p0s Markov games, it is the latter.

Life is generally good!

- Variant of value iteration works
- Even in the tabular RL-setting things work out (minimax Q-learning)

Bellman updates

Just change **max** to **minimax**

Iterate $V_{i+1}^*(s) \leftarrow \max_{a \in A} [R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_i^*(s')]$

Iterate $V_{i+1}^*(s) \leftarrow \min_{a_1 \in A_1} \max_{a_2 \in A_2} [R(s,a_1,a_2) + \gamma \sum_{s'} P(s'|s,a_1,a_2) V_i^*(s')]$

$\min_{x \in \Delta_{A_1}} \max_{y \in \Delta_{A_2}} \sum_{a_1,a_2} x(a_1) y(a_2) [R(s,a_1,a_2) + \gamma \sum_{s'} P(s'|s,a_1,a_2) V_i^*(s')]$

Can be extended to Q-functions also

- Part of a larger framework known as “Nash-Q-learning”

Bellman updates

Just change **max** to **minimax**

Iterate $V^*_{i+1}(s) \leftarrow \max_{a \in A} [R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*_i(s')]$

Iterate $V^*_{i+1}(s) \leftarrow \min_{a_1 \in A_1} \max_{a_2 \in A_2} [R(s,a_1,a_2) + \gamma \sum_{s'} P(s'|s,a_1,a_2) V^*_i(s')]$

No! What's wrong?

$$\min_{x \in \Delta_{A_1}} \max_{y \in \Delta_{A_2}} \sum_{a_1,a_2} x(a_1) y(a_2) [R(s,a_1,a_2) + \gamma \sum_{s'} P(s'|s,a_1,a_2) V^*_i(s')]$$

Can be extended to Q-functions also

- Part of a larger framework known as "Nash-Q-learning"

	b_1	b_2
a_1	$[R(s, a_1, b_1) + \gamma \sum_{s'} P(s' \mid s, a_1, a_2) V_i^*(s')]$	$[R(s, a_1, b\_2) + \gamma \sum_{s'} P(s' \mid s, a_1, a_2) V_i^*(s')]$
a_2	$[R(s, a_2, b_1) + \gamma \sum_{s'} P(s' \mid s, a_1, a_2) V_i^*(s')]$	$[R(s, a_2, b_2) + \gamma \sum_{s'} P(s' \mid s, a_1, a_2) V_i^*(s')]$

Bellman updates just update the state values by solving the one-step lookahead problem

The general-sum case

Much more tricky!

- Possible that no deterministic, stationary strategy exists
- Bellman updates like these **may not converge**
  - Zero-sum, potential, cooperative games are a special case

Many misconceptions here!

The general-sum case

Much more tricky!

- Possible that no deterministic, stationary strategy exists
- Bellman updates like these **may not converge**
  - Zero-sum, potential, cooperative games are a special case

Many misconceptions here!

Can we just replace min-max by [whatever equilibrium concept we want?

- E.g., correlated Q-learning (Greenwald et. al., 2003)
- Yes, but…what equilibrium concept are you talking about?
  - For example, multiple definitions of corelated equilibria in CE
  - Is there **monitoring** or not? (players are able to see history of play, not just current state) Both cases "make sense"

NoSDE: A famous pathology by Zinkevich

Cyclic Equilibria in Markov Games, Neurips 2005

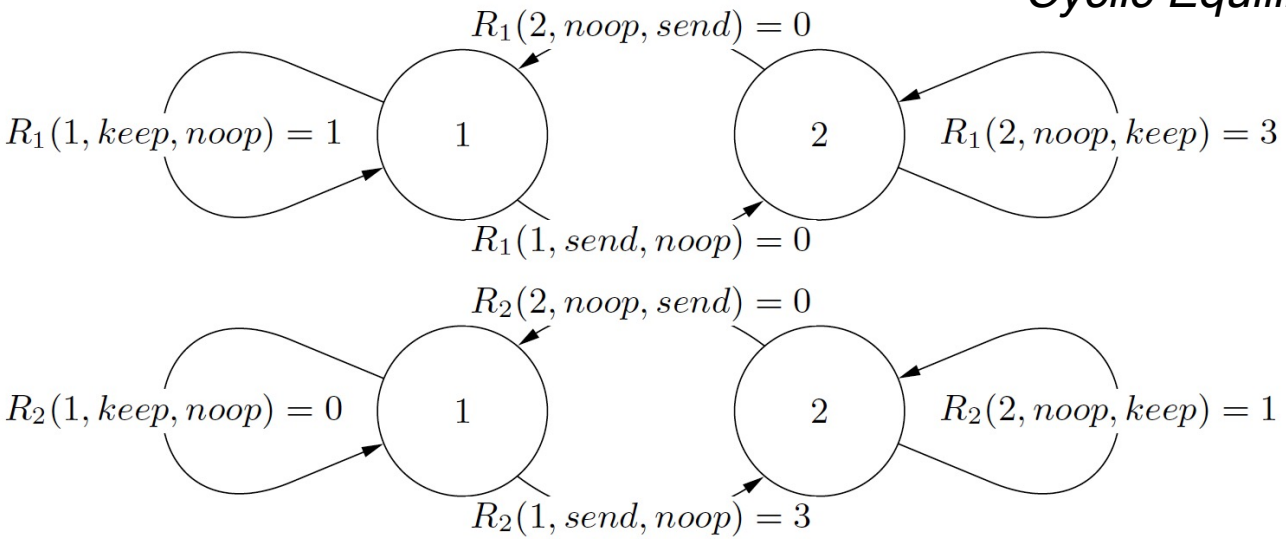


Figure 1: An example of a NoSDE game. Here, $S = \{1, 2\}$, $A_{1,1} = A_{2,2} = \{keep, send\}$, $A_{1,2} = A_{2,1} = \{noop\}$, $T(1, keep, noop) = 1$, $T(1, send, noop) = 2$, $T(2, noop, keep) = 2$, $T(2, noop, send) = 1$, and $\gamma = 3/4$. In the unique stationary equilibrium, Player 1 sends with probability $2/3$ and Player 2 sends with probability $5/12$.

Can this happen in zero-sum Markov games?

Has just 2 states, 2 actions per state, no simultaneous moves, deterministic state transition (as simple as you get)

- No stationary deterministic strategy exists! **Must randomize even here**
- **Why?**

Another result by Zinkevich

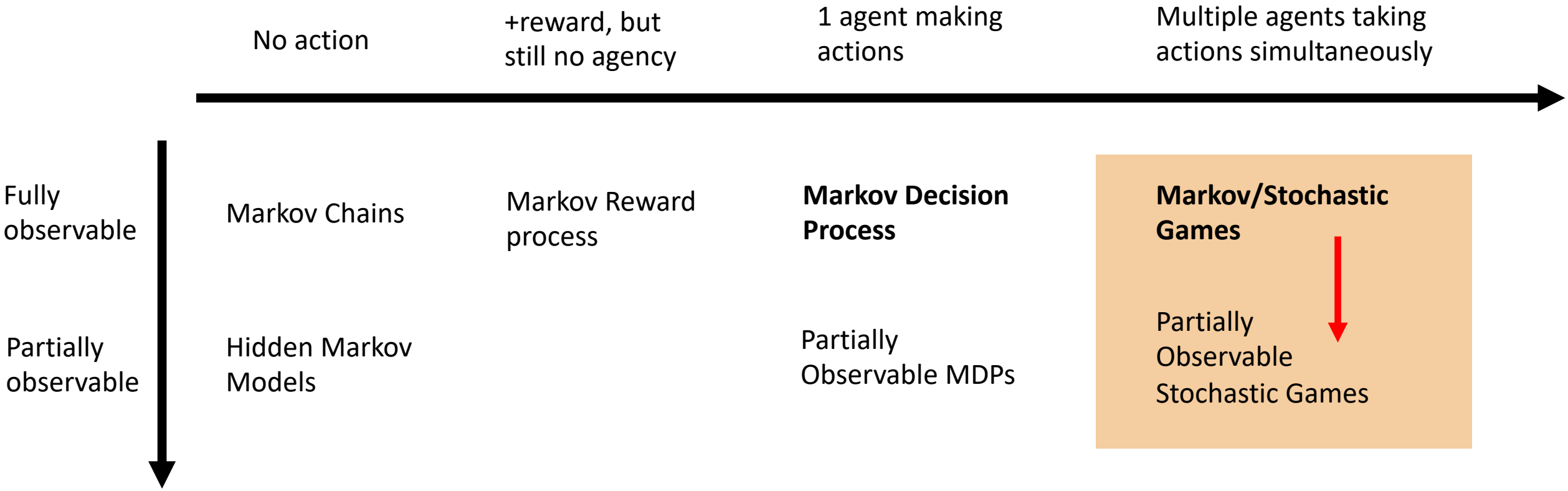
Theorem 1 *For any NoSDE game* $\Gamma = [S, N, \mathbf{A}, T, R]$ *with a unique equilibrium policy* π*, there exists another NoSDE game* $\Gamma' = [S, N, \mathbf{A}, T, R']$ *with its own unique equilibrium policy* π' *such that* $Q^{\pi,\Gamma} = Q^{\pi',\Gamma'}$ *but* $\pi \neq \pi'$ *and* $V^{\pi,\Gamma} \neq V^{\pi',\Gamma'}$.

Learning Q-values alone does **not** correctly identify an optimal strategy

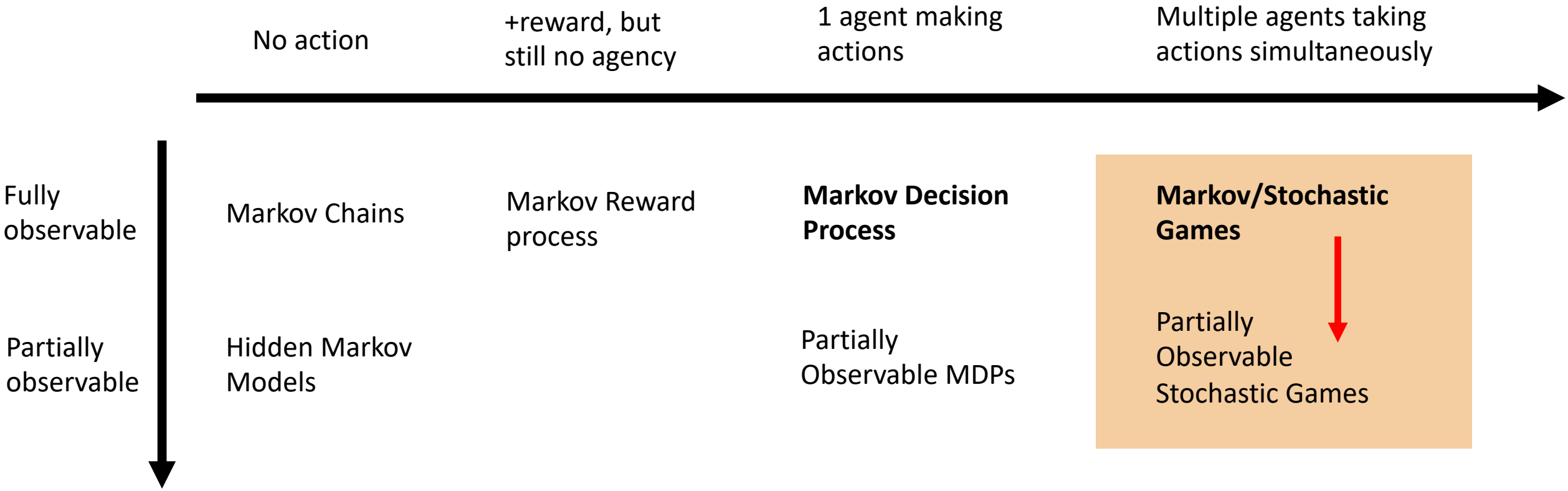
Contrast with 1 Player case, 2p0s

No matter how well you learn Q-values, you will never get an equilibrium by one-step lookahead!

What happens when there is more imperfect information?



What happens when there is more imperfect information?



Immediately intractable (POMDPs are already hard, or even undecidable)

This is why we chose to focus on EFGs!

Part 2: Working on "your own" equilibrium concepts

Disclaimer

Everything in this half is “my work” which is not “mainstream”

Don’t quote what I’m doing here and expect random people to understand, let alone agree

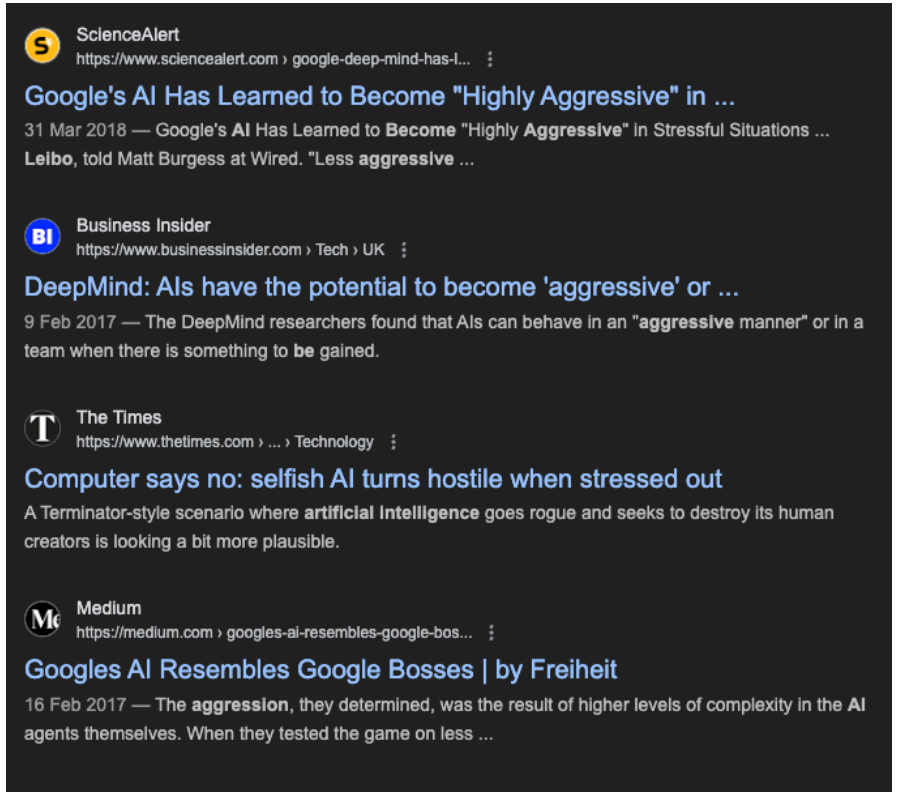
- Most people will know what Markov games are
- Most people probably won’t know what Stackelberg Extensive Form Correlated Equilibria is…

This is more of a personal sharing session

Still, **general principles apply**, which is what I’m trying to get at

A bit of background…

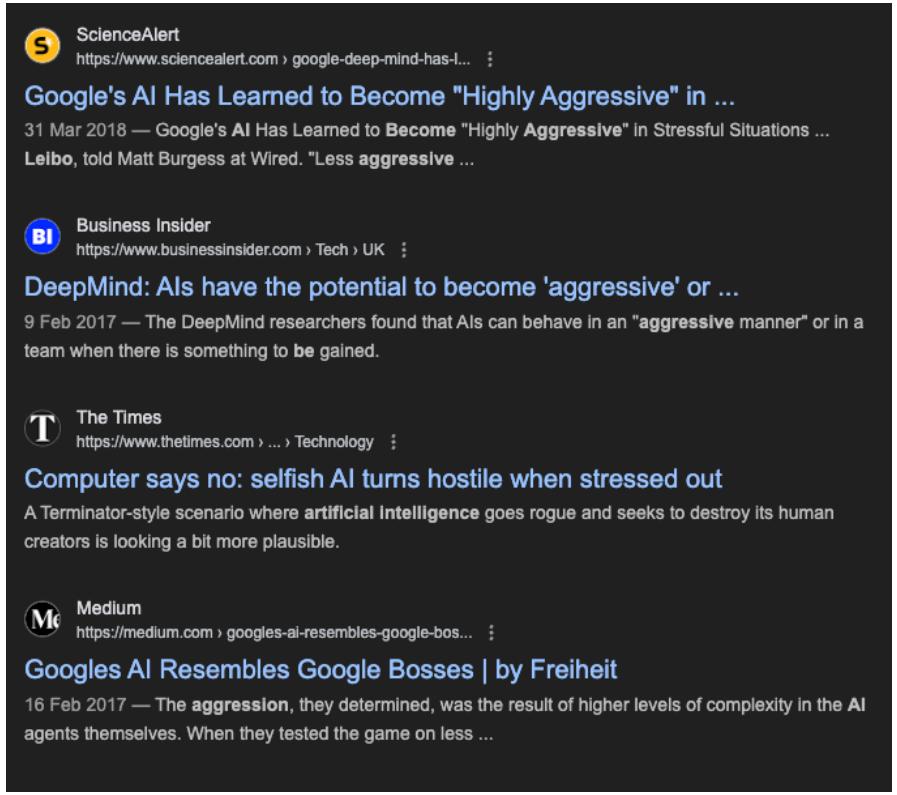
A bit of background…



Multi-agent Reinforcement Learning in Sequential Social Dilemmas (Leibo et. al., 2017)

A bit of background…

Why is it that my peers in MARL can just “run some experiments”, report “cute results” and get hundreds of citations? What am I doing?! I want some of that!

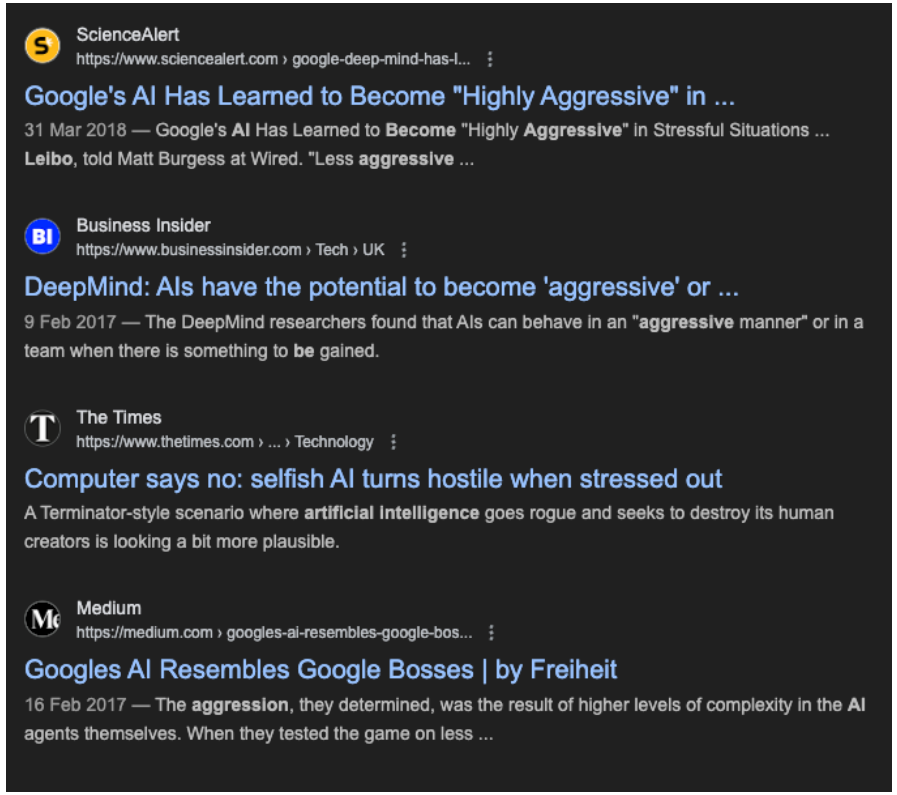


Multi-agent Reinforcement Learning in Sequential Social Dilemmas (Leibo et. al., 2017)

A bit of background…

Why is it that my peers in MARL can just “run some experiments”, report “cute results” and get hundreds of citations? What am I doing?! I want some of that!

Let me ignore theory and just throw the “RL kitchen sink!”



Multi-agent Reinforcement Learning in Sequential Social Dilemmas (Leibo et. al., 2017)

The grand plan for an industry project

Imagine basketball, but with potentially selfish players

- Example: some players want to selfishly score, potentially at the expense of the team

The grand plan for an industry project

Imagine basketball, but with potentially selfish players

- Example: some players want to selfishly score, potentially at the expense of the team

You are the coach of one team (versus the other team's coach)

- What strategy can you recommend to the team, while respecting the fact that they are selfish?
- Your own team members can deviate if their "needs" are not met
- Can **exploit** selfish player on the other team

The grand plan for an industry project

Imagine basketball, but with potentially selfish players

- Example: some players want to selfishly score, potentially at the expense of the team

You are the coach of one team (versus the other team's coach)

- What strategy can you recommend to the team, while respecting the fact that they are selfish?
- Your own team members can deviate if their "needs" are not met
- Can **exploit** selfish player on the other team

What I did

- Don't bother with problem formulation, equilibrium concepts, throw in a bunch of PPO/MADDPG agents and let things ruuuuun!

The grand plan for an industry project

Imagine basketball, but with potentially selfish players

- Example: some players want to selfishly score, potentially at the expense of the team

You are the coach of one team (versus the other team's coach)

- What strategy can you recommend to the team, while respecting the fact that they are selfish?
- Your own team members can deviate if their "needs" are not met
- Can **exploit** selfish player on the other team

What I did

- Don't bother with problem formulation, equilibrium concepts, throw in a bunch of PPO/MADDPG agents and let things ruuuuun!

Result: complete failure

- No or unstable convergence.
- Why? Equilibrium may not even exist! Chasing an **impossible** target

Tip: always start small

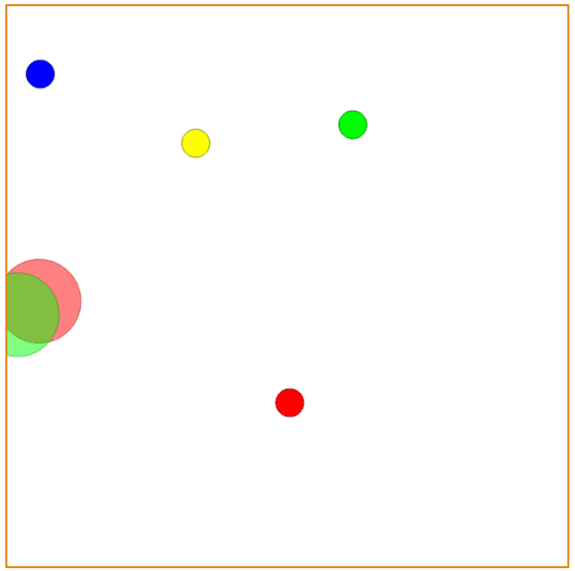
Understand the problem first

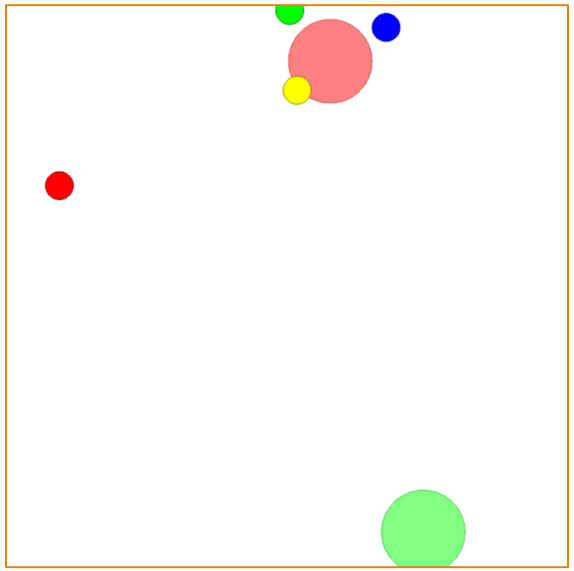
- Why is this even multiagent?
- What are you looking for? Equilibrium? Or just something that looks good?

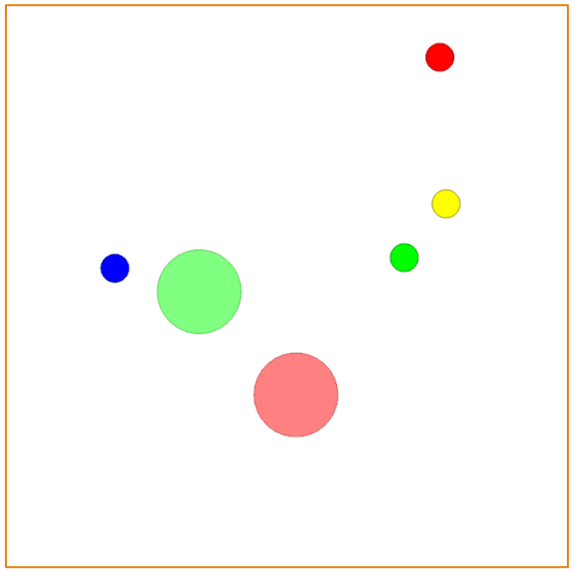
Start with simple problems!

Simulated Chicken Game

Player 1 \ Player 2	Dare	Chicken
Dare	0,0	7,2
Chicken	2,7	6,6



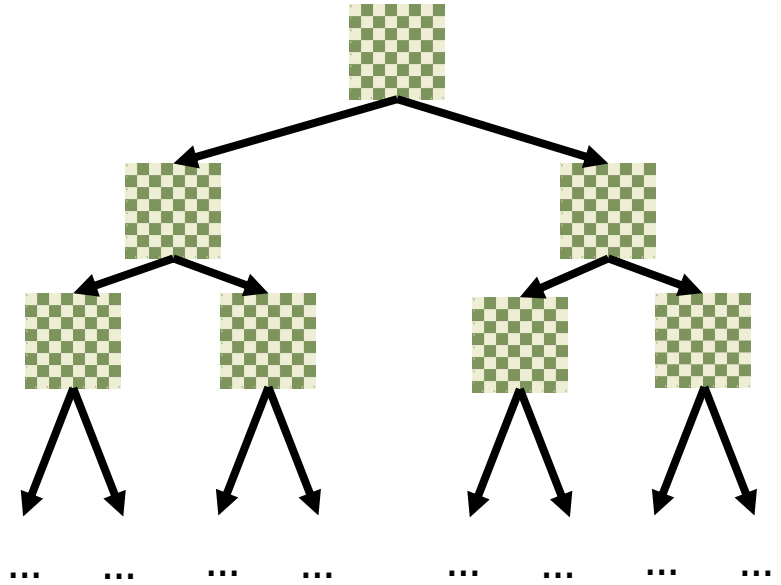




Function Approximation for General-sum Games

Function Approximation for Solving Stackelberg Equilibrium in Large Perfect Information Games (AAAI '23) [oral]

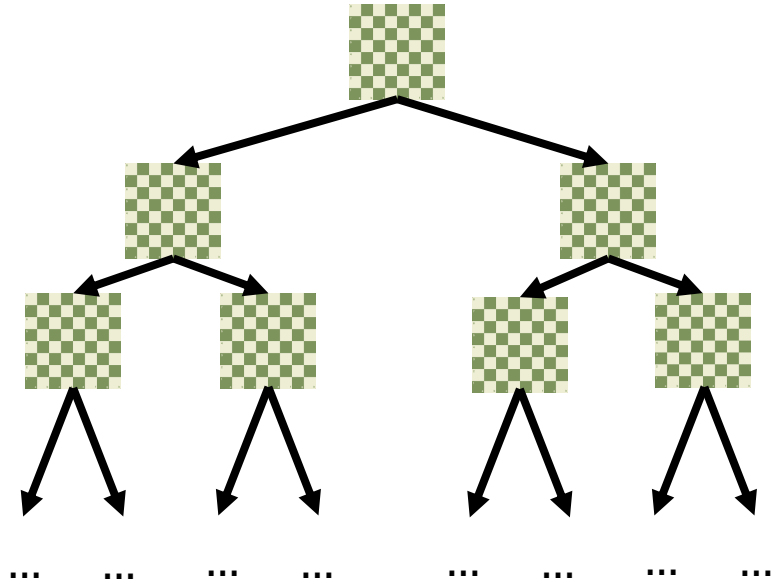
Game Solving with Function Approximation



Game Solving with Function Approximation

Consider zero-sum games

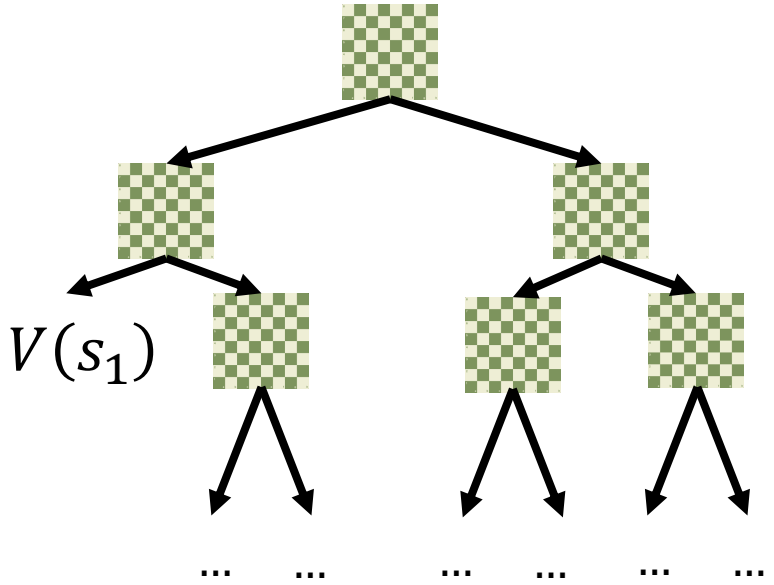
- Game tree too large to traverse explicitly
- **Value Function** $V(s)$ approximates how "good or bad" each state is



Game Solving with Function Approximation

Consider zero-sum games

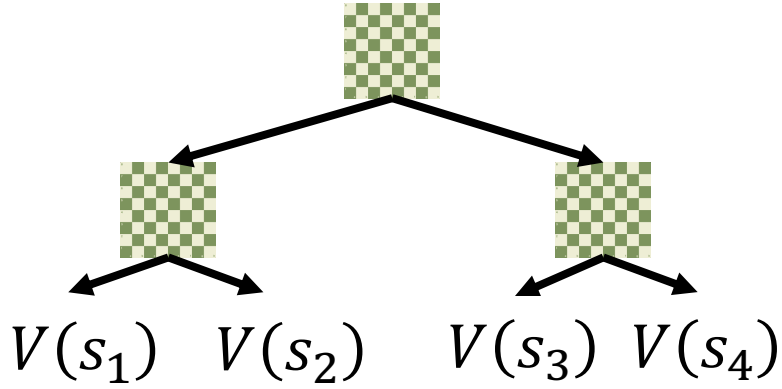
- Game tree too large to traverse explicitly
- **Value Function** $V(s)$ approximates how "good or bad" each state is



Game Solving with Function Approximation

Consider zero-sum games

- Game tree too large to traverse explicitly
- **Value Function** $V(s)$ approximates how "good or bad" each state is



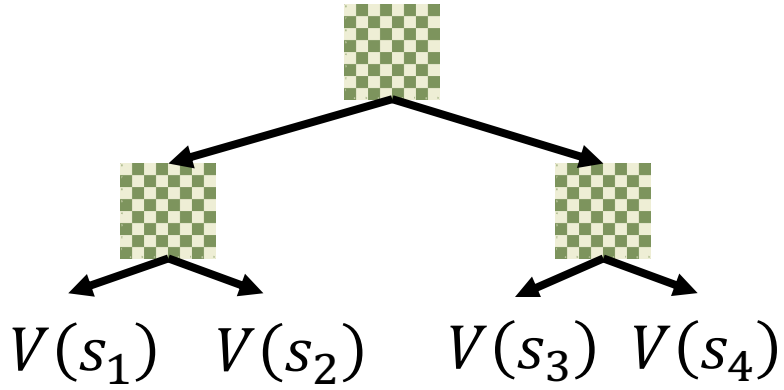
Game Solving with Function Approximation

Consider zero-sum games

- Game tree too large to traverse explicitly
- **Value Function** $V(s)$ approximates how "good or bad" each state is

Previously (e.g., Deep Blue)

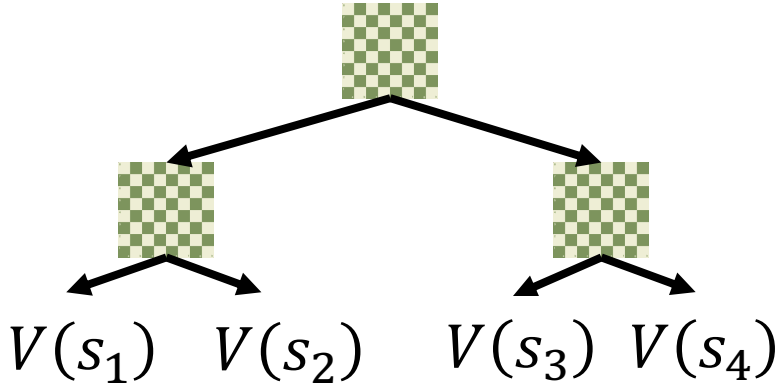
- Handcrafted evaluation function $V(s)$ based on heuristics from experts



Game Solving with Function Approximation

Consider zero-sum games

- Game tree too large to traverse explicitly
- **Value Function** $V(s)$ approximates how "good or bad" each state is



Previously (e.g., Deep Blue)

- Handcrafted evaluation function $V(s)$ based on heuristics from experts

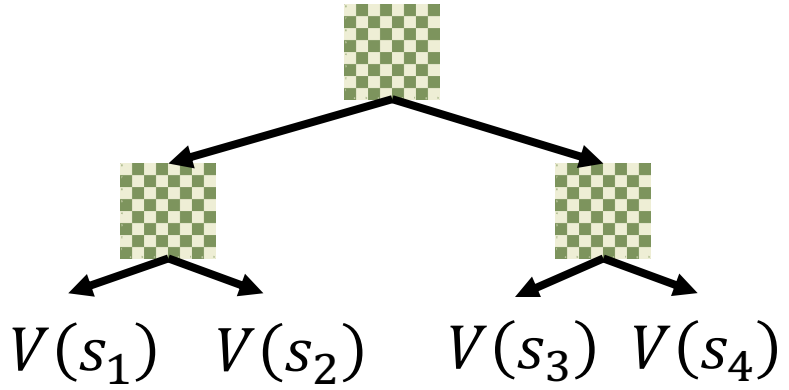
Today (e.g., AlphaGo/Zero, DeepStack)

- **Learn** how good each state s is, generalize to states not seen before
- $V_\phi(s)$ is a network parameterized by ϕ approximating the value of s

Game Solving with Function Approximation

Consider zero-sum games

- Game tree too large to traverse explicitly
- **Value Function** $V(s)$ approximates how "good or bad" each state is



Previously (e.g., Deep Blue)

- Handcrafted evaluation function $V(s)$ based on heuristics from experts

Today (e.g., AlphaGo/Zero, DeepStack)

- **Learn** how good each state s is, generalize to states not seen before
- $V_\phi(s)$ is a network parameterized by ϕ approximating the value of s

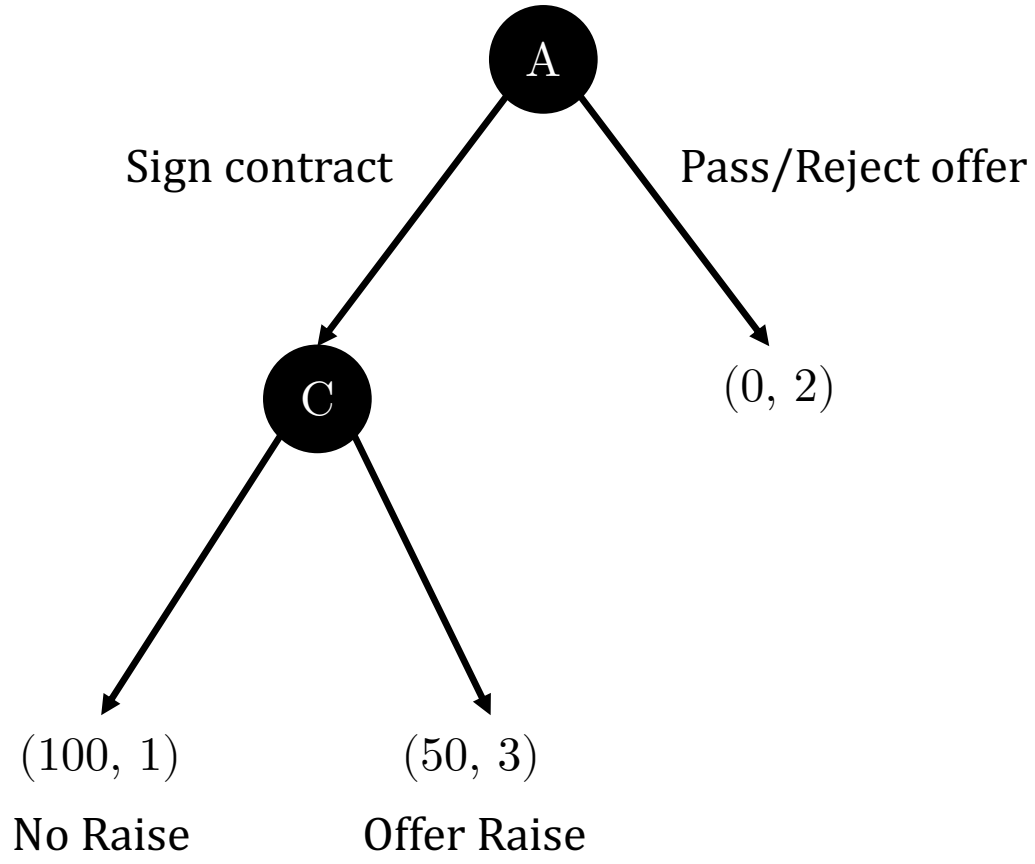
How to apply FA to general sum (Stackelberg) games in a principled manner?

Recall: The Hiring Game

Payoffs are shown as (Company, Applicant), or (Leader, Follower)

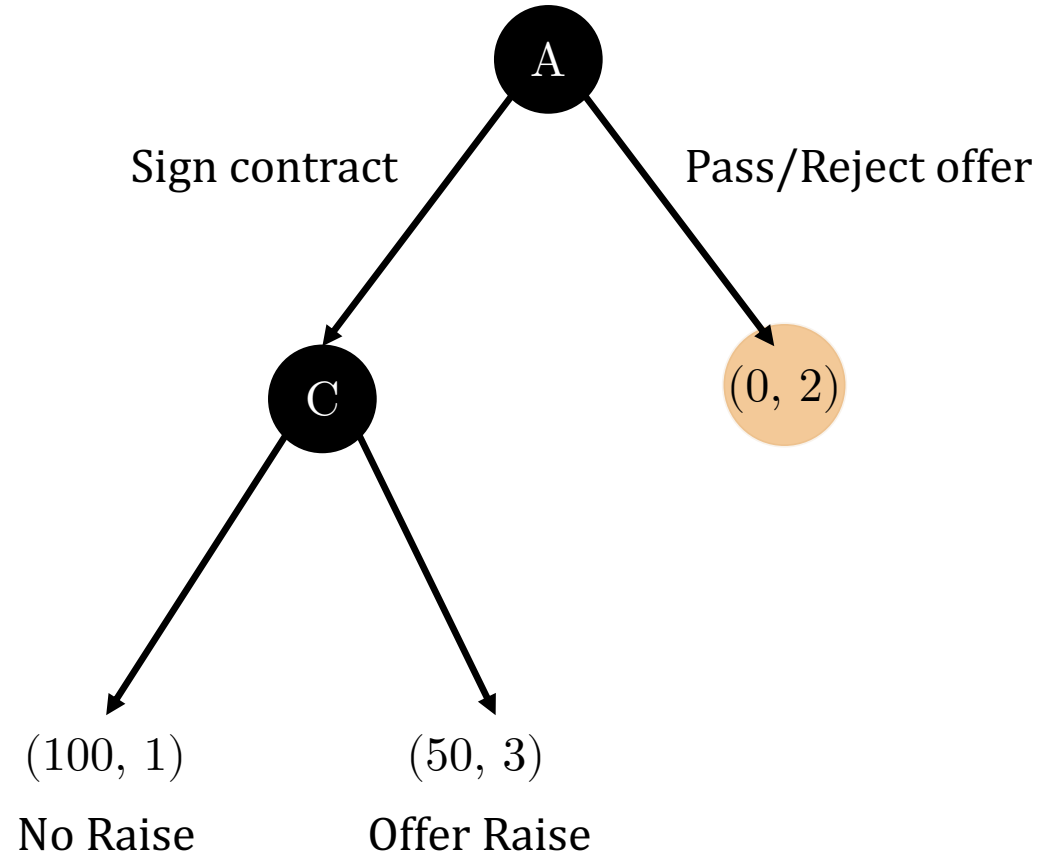
*AKA the Leader

Played between **Company** (C) and **Applicant** (A) *AKA the follower



Recall: The Hiring Game

Payoffs are shown as (Company, Applicant), or (Leader, Follower)



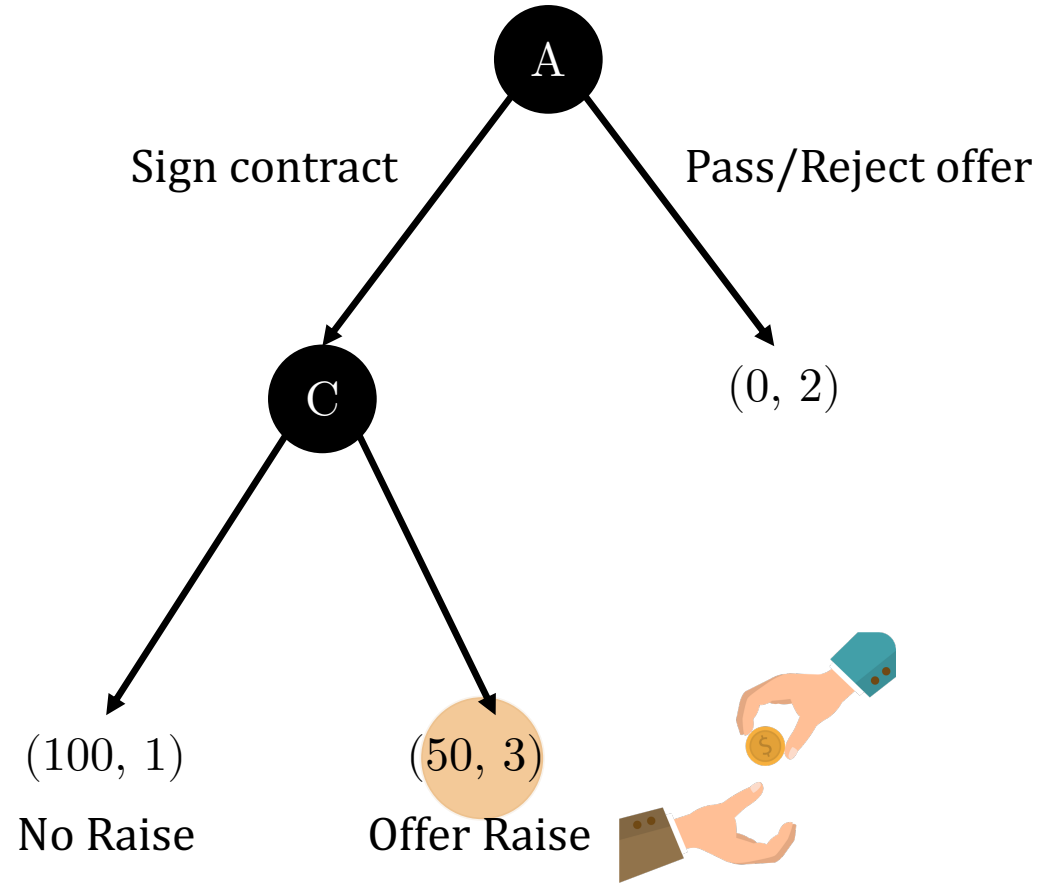
*AKA the Leader

Played between **Company** (C) and **Applicant** (A) *AKA the follower

Applicant has an option of signing a 6-year contract with the company

Recall: The Hiring Game

Payoffs are shown as (Company, Applicant), or (Leader, Follower)



*AKA the Leader

Played between **Company** (C) and **Applicant** (A) *AKA the follower

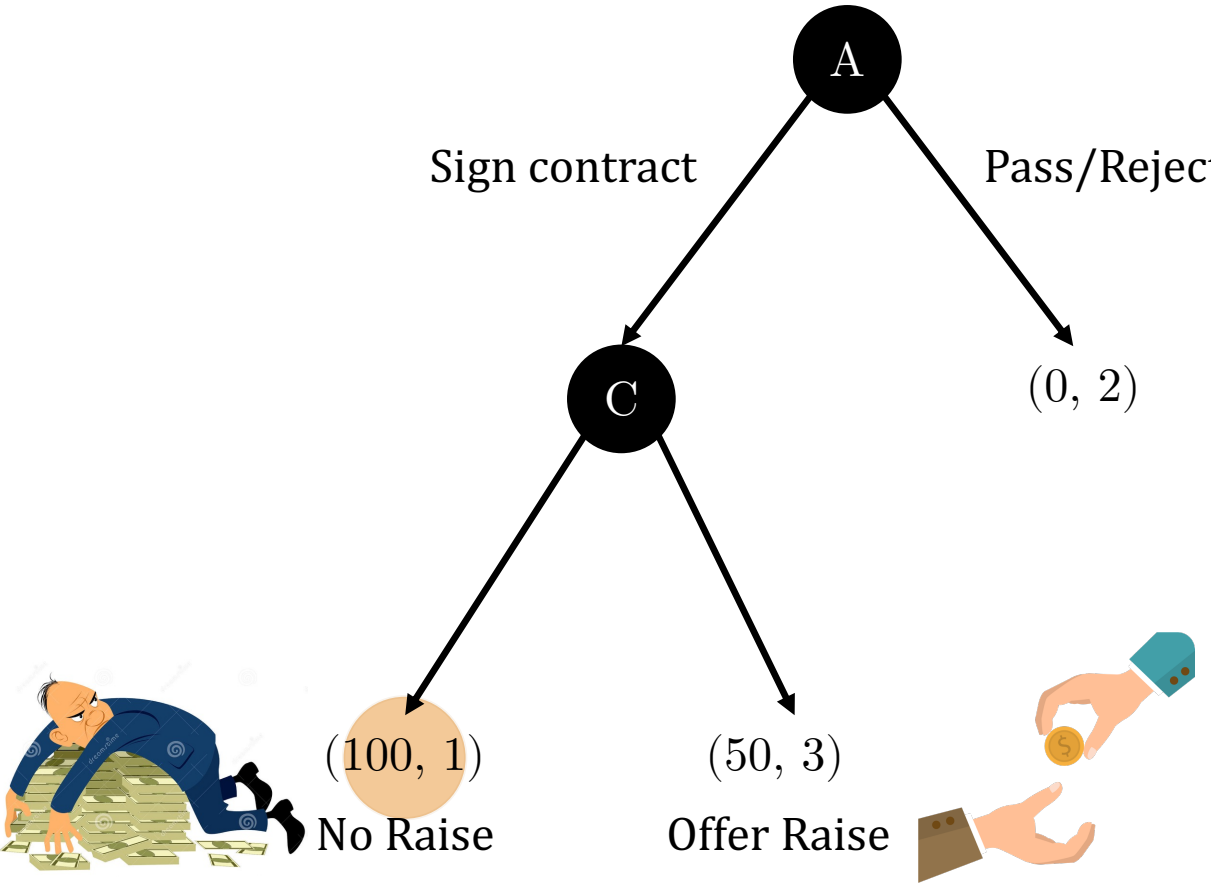
Applicant has an option of signing a 6-year contract with the company

After 3 years, company can decide to give a raise or otherwise

Image sources:
https://www.pngitem.com/
https://www.dreamstime.com/

Recall: The Hiring Game

Payoffs are shown as (Company, Applicant), or (Leader, Follower)



*AKA the Leader

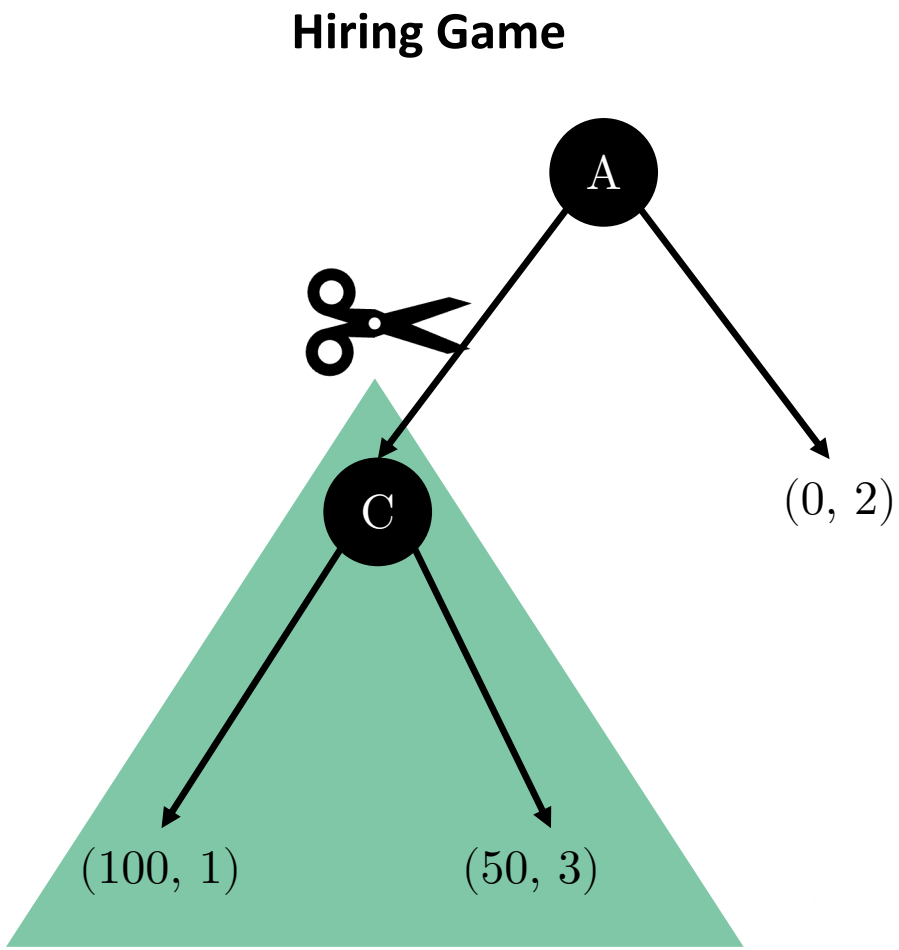
Played between **Company** (C) and **Applicant** (A) *AKA the follower

Applicant has an option of signing a 6-year contract with the company

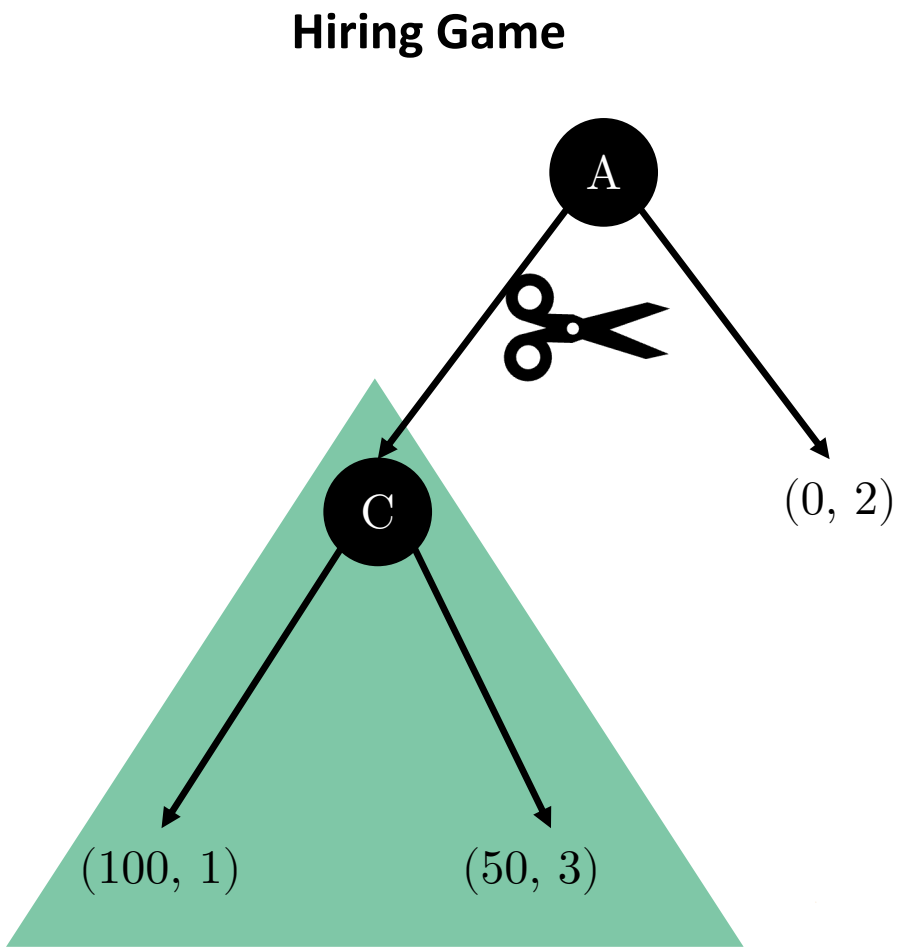
After 3 years, company can decide to give a raise or otherwise

Image sources:
https://www.pngitem.com/
https://www.dreamstime.com/

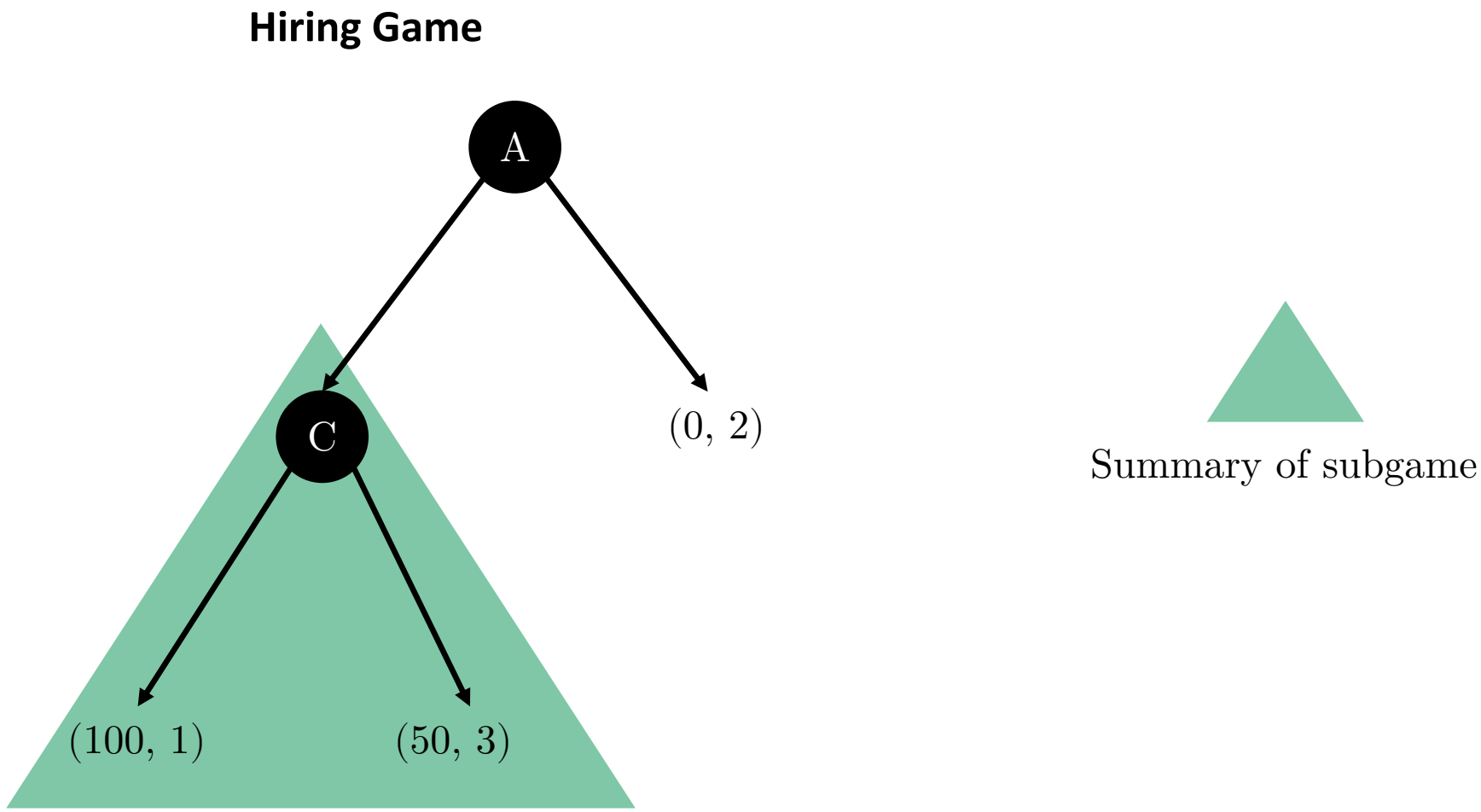
Assumption: Markovianity in FA



Assumption: Markovianity in FA

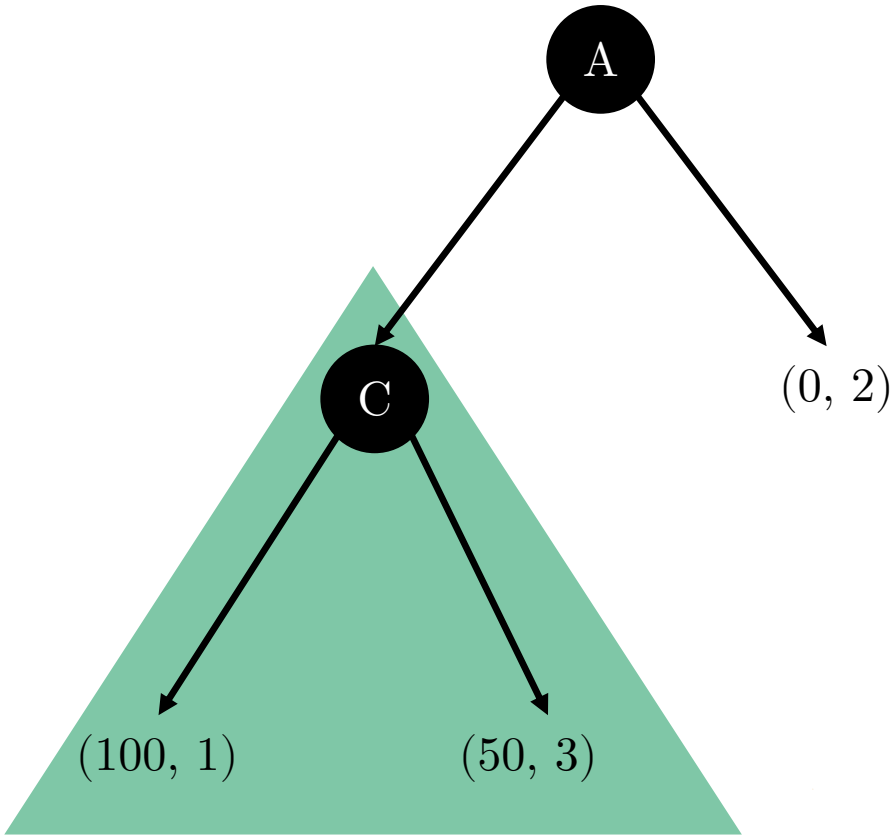


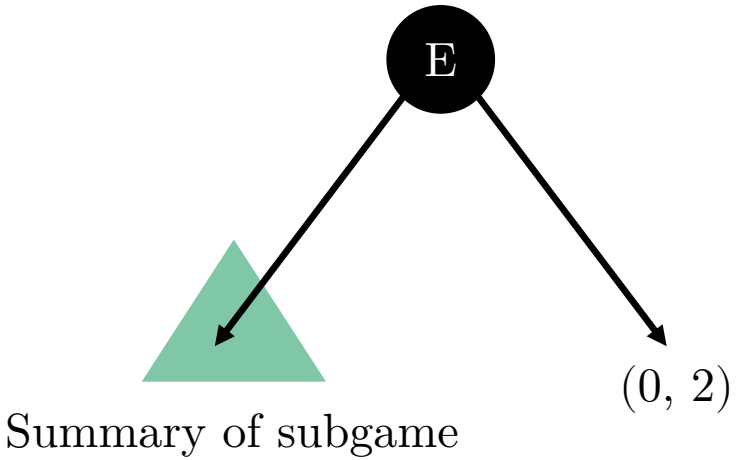
Assumption: Markovianity in FA



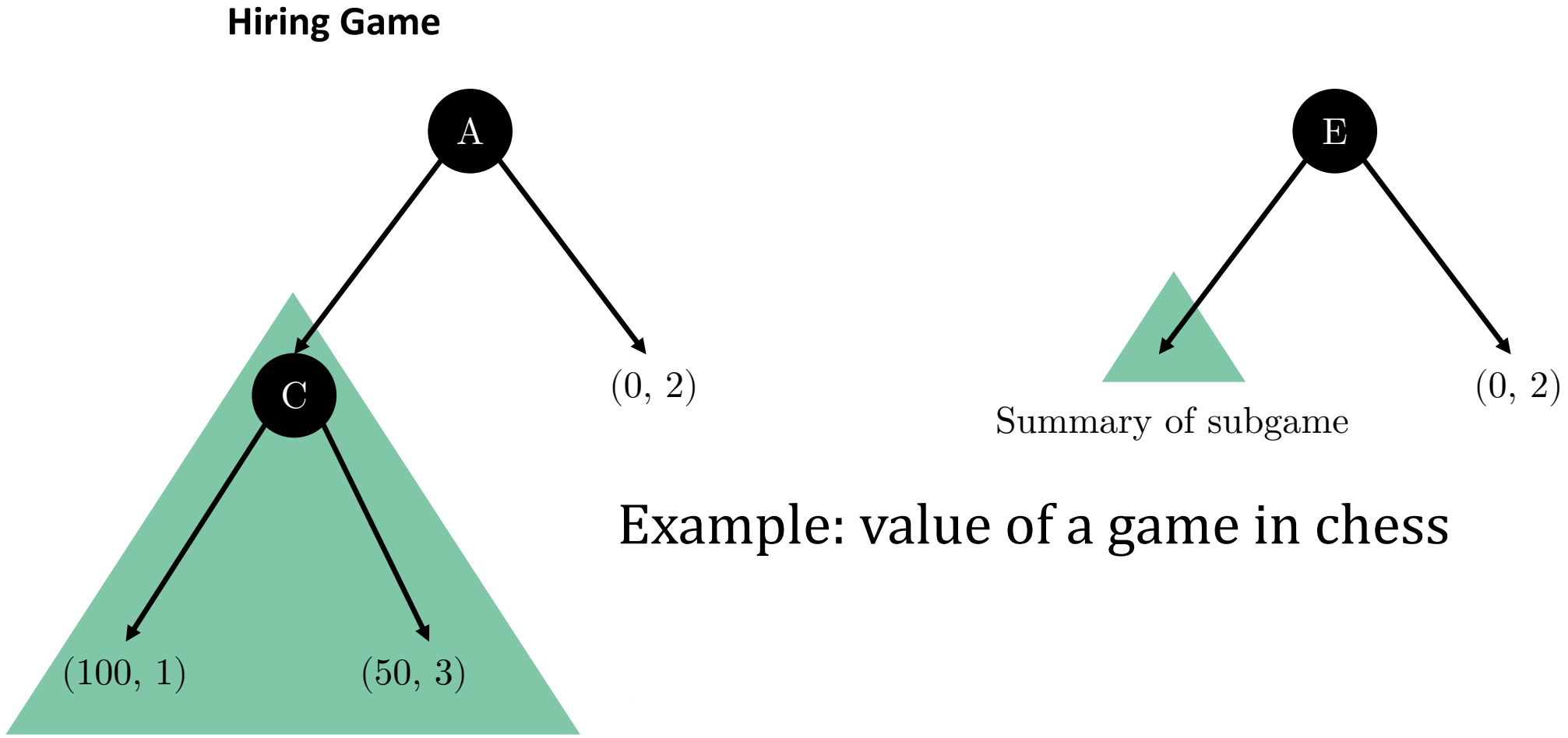
Assumption: Markovianity in FA

Hiring Game



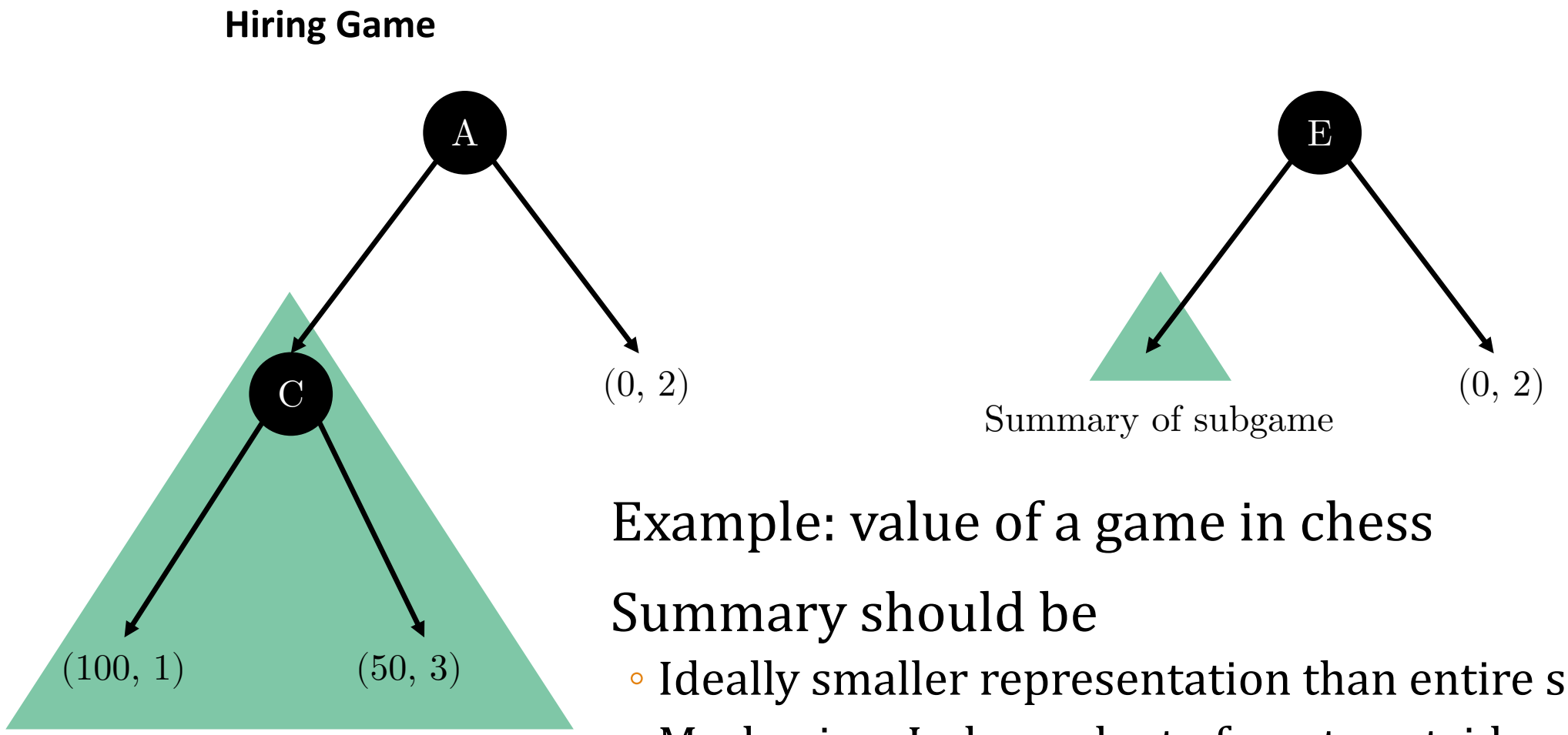


Assumption: Markovianity in FA



Example: value of a game in chess

Assumption: Markovianity in FA

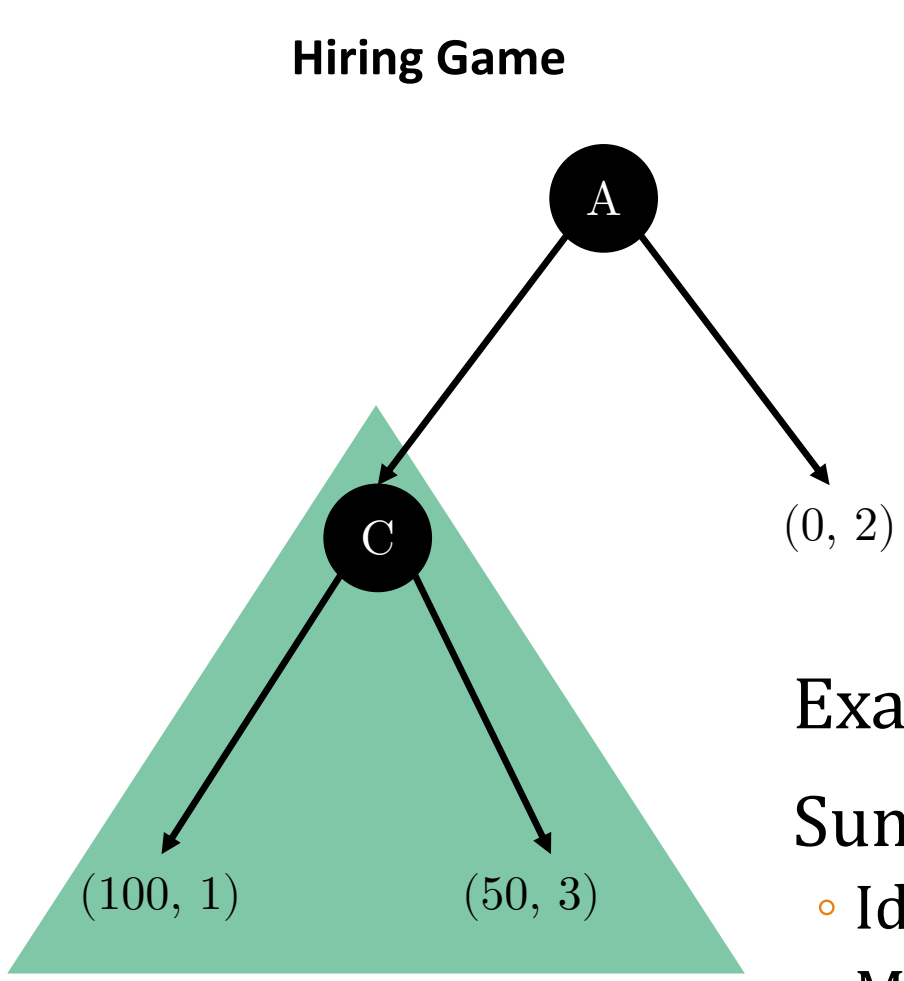


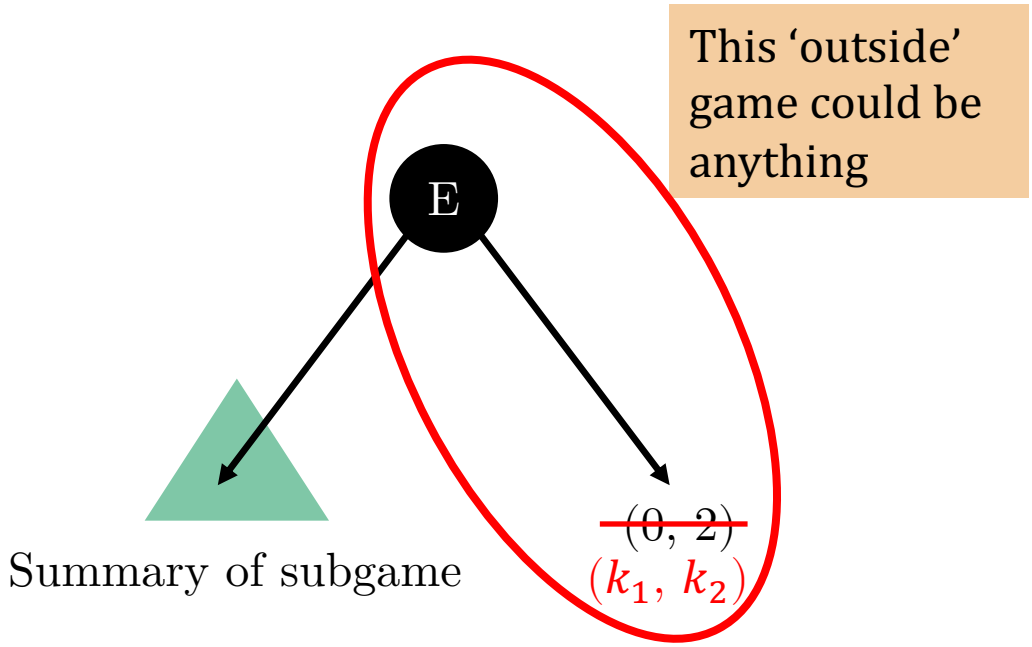
Example: value of a game in chess

Summary should be

- Ideally smaller representation than entire subgame
- Markovian: Independent of parts outside subgame

Assumption: Markovianity in FA





Example: value of a game in chess

Summary should be

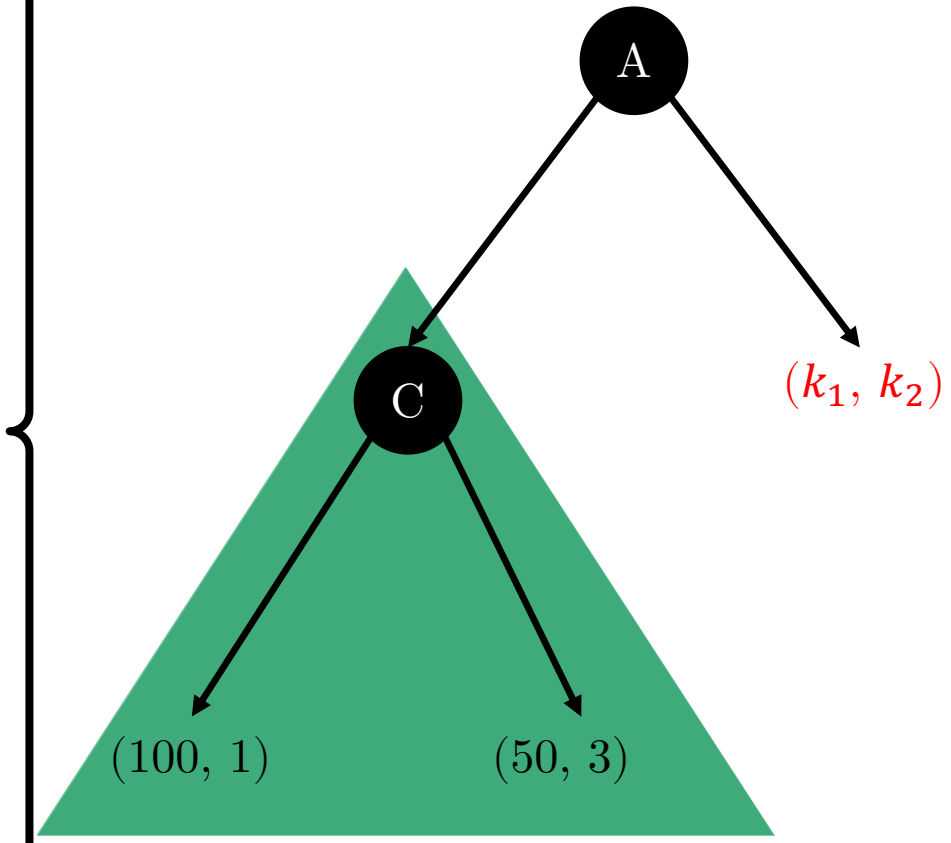
- Ideally smaller representation than entire subgame
- Markovian: Independent of parts outside subgame

Enforceable Payoff Frontiers (EPF)

Summary **cannot be** a scalar/fixed sized tuple

Enforceable Payoff Frontiers (EPF)

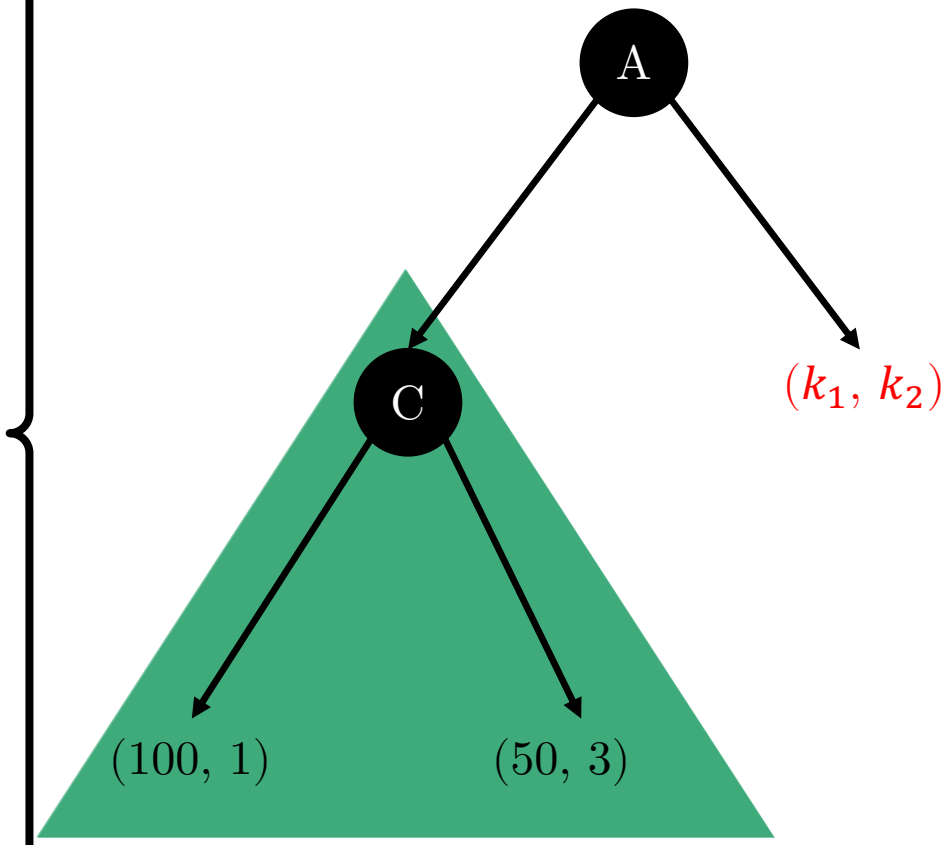
Summary **cannot be** a scalar/fixed sized tuple



Summary needs to solve game for all (k_1, k_2)

Enforceable Payoff Frontiers (EPF)

Summary **cannot be** a scalar/fixed sized tuple



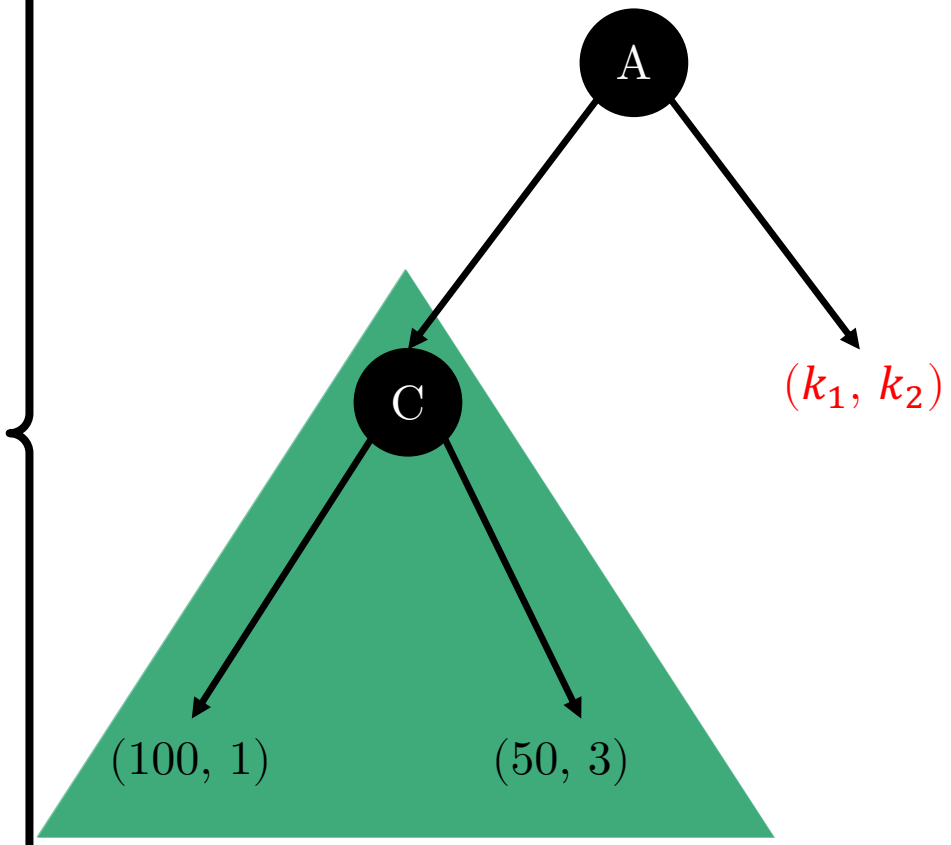
Summary needs to solve game for all (k_1, k_2)

How much can Leader get in subgame while giving Follower k_2?

- If $> k_1$, go left, if $< k_1$, let the follower quit
- Need to query this for all possible k_2!

Enforceable Payoff Frontiers (EPF)

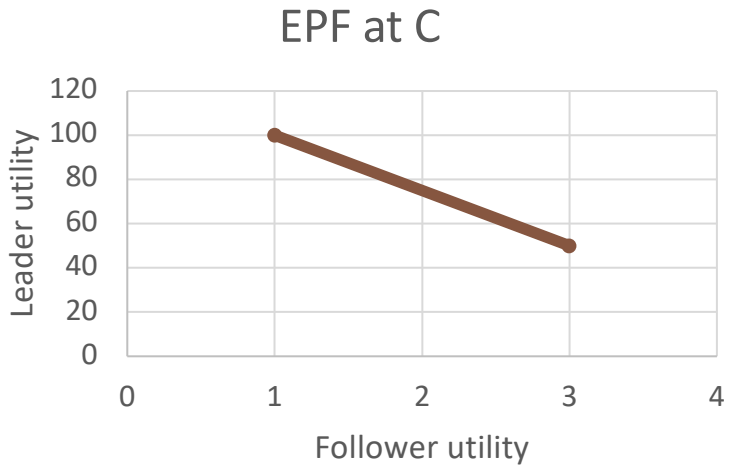
Summary **cannot be** a scalar/fixed sized tuple



Summary needs to solve game for all (k_1, k_2)

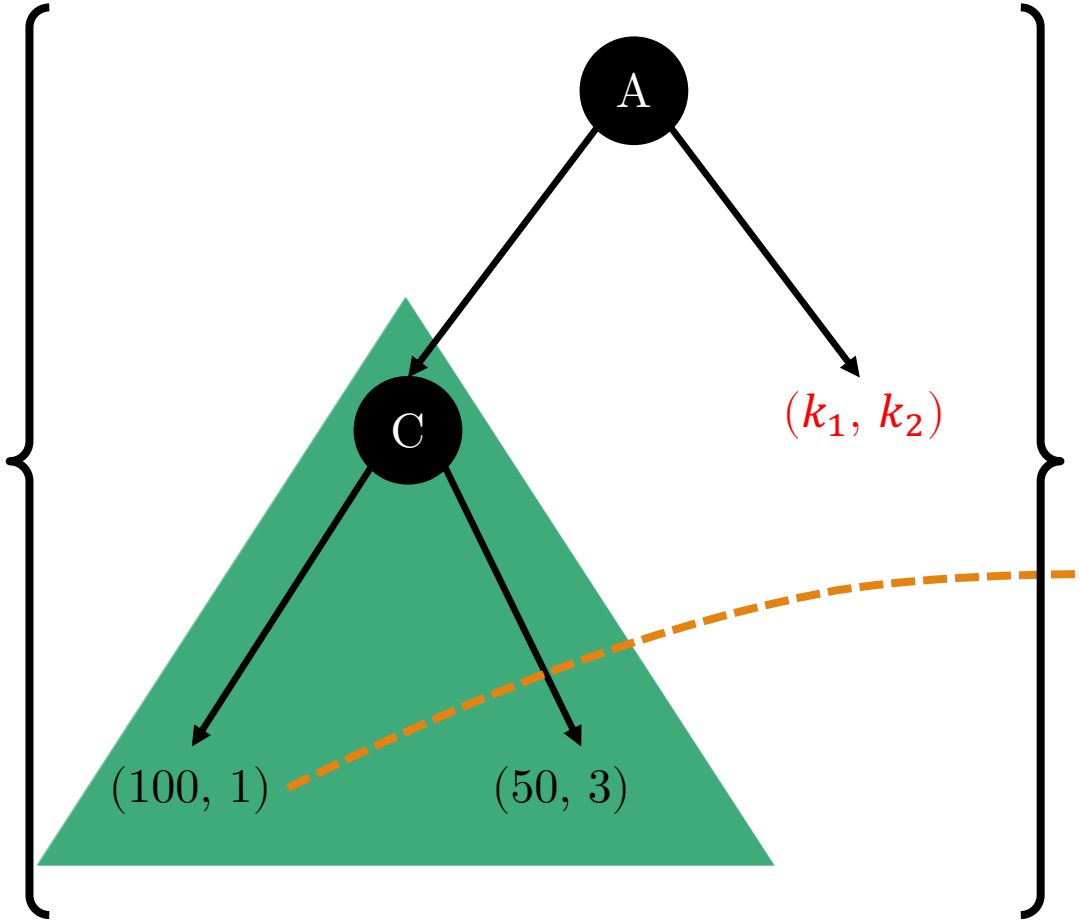
How much can Leader get in subgame while giving Follower k_2?

- If $> k_1$, go left, if $< k_1$, let the follower quit
- Need to query this for all possible k_2!



Enforceable Payoff Frontiers (EPF)

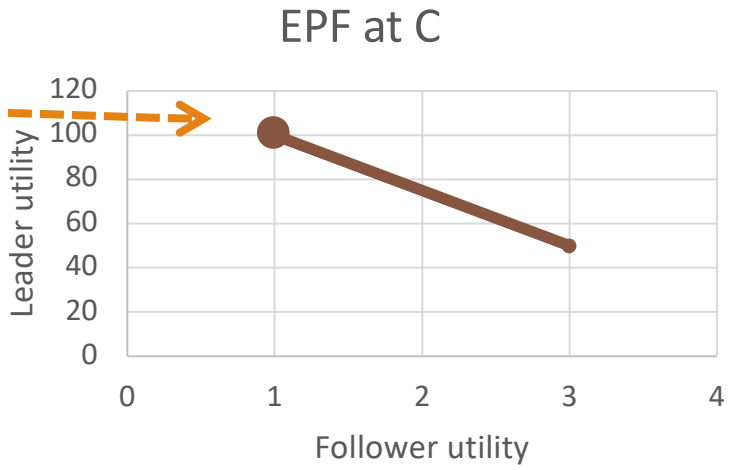
Summary **cannot be** a scalar/fixed sized tuple



Summary needs to solve game for all (k_1, k_2)

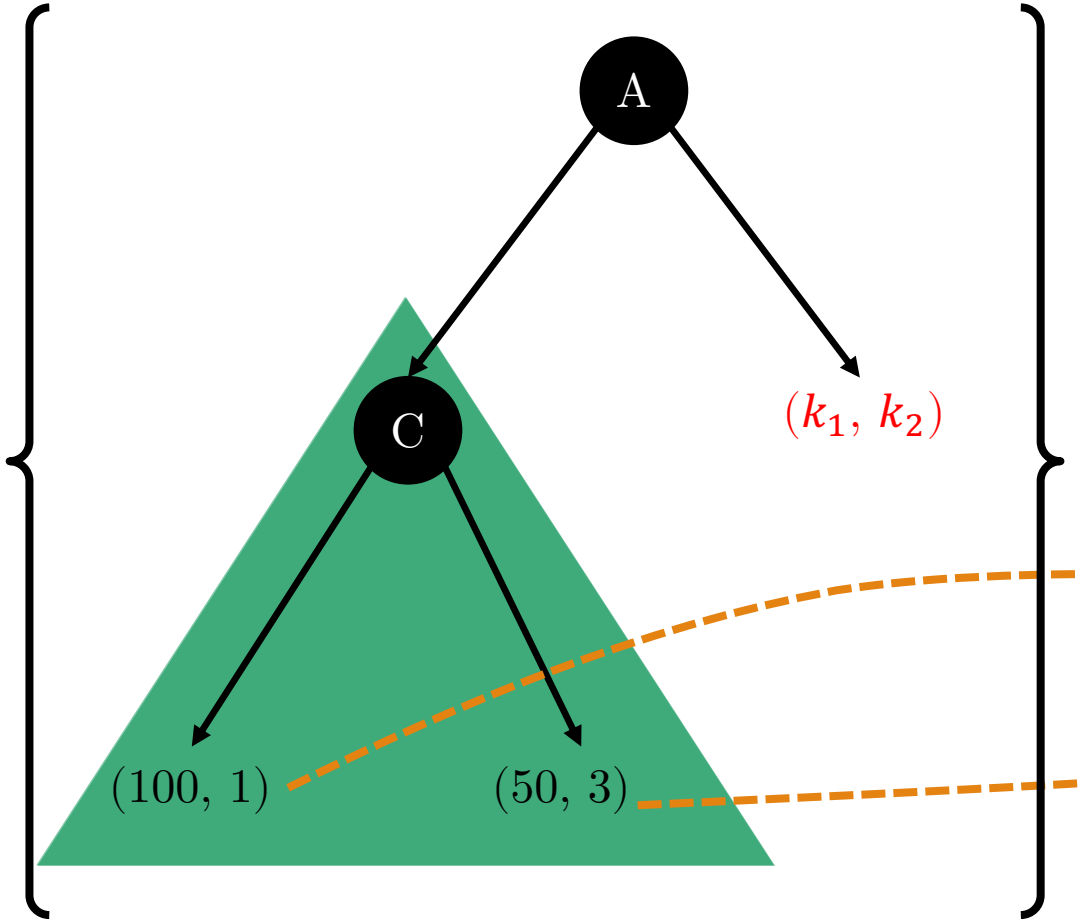
How much can Leader get in subgame while giving Follower k_2?

- If $> k_1$, go left, if $< k_1$, let the follower quit
- Need to query this for all possible k_2!



Enforceable Payoff Frontiers (EPF)

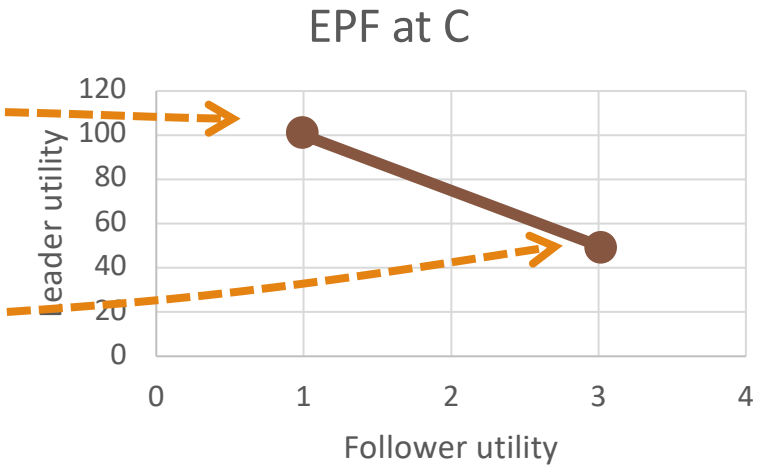
Summary **cannot be** a scalar/fixed sized tuple



Summary needs to solve game for all (k_1, k_2)

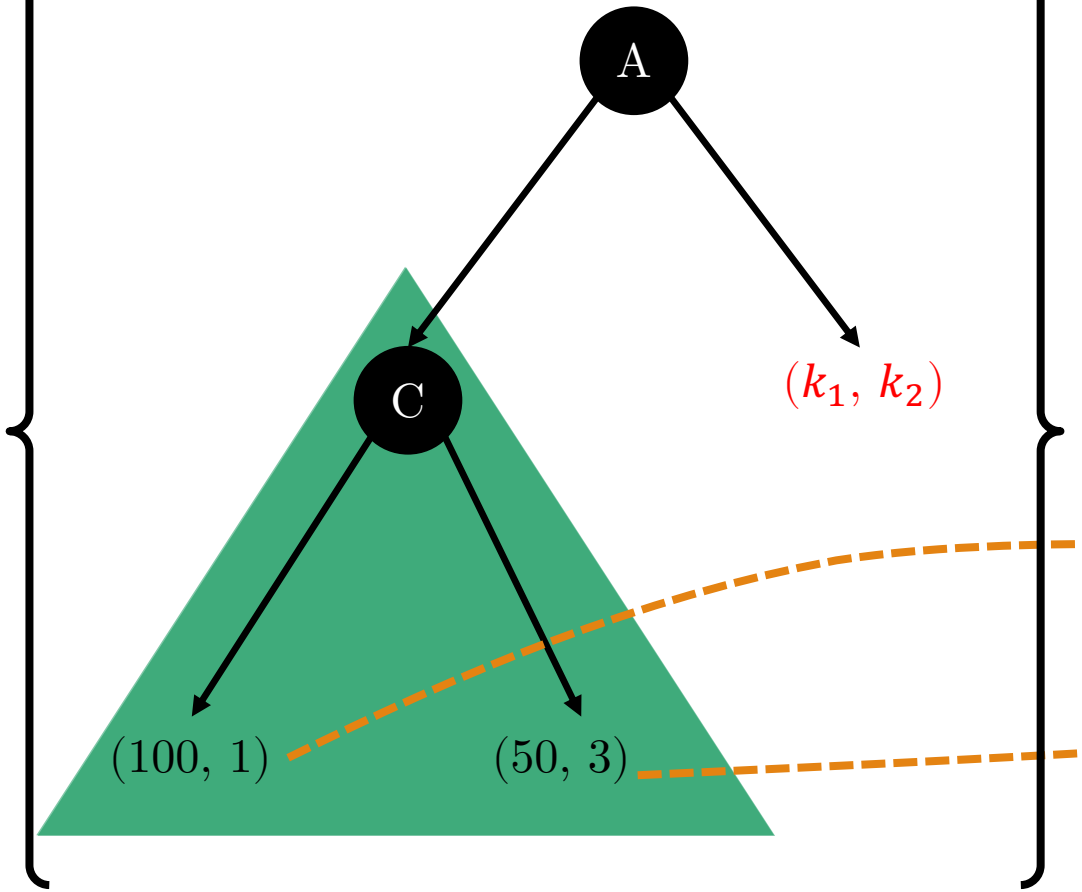
How much can Leader get in subgame while giving Follower k_2?

- If $> k_1$, go left, if $< k_1$, let the follower quit
- Need to query this for all possible k_2!



Enforceable Payoff Frontiers (EPF)

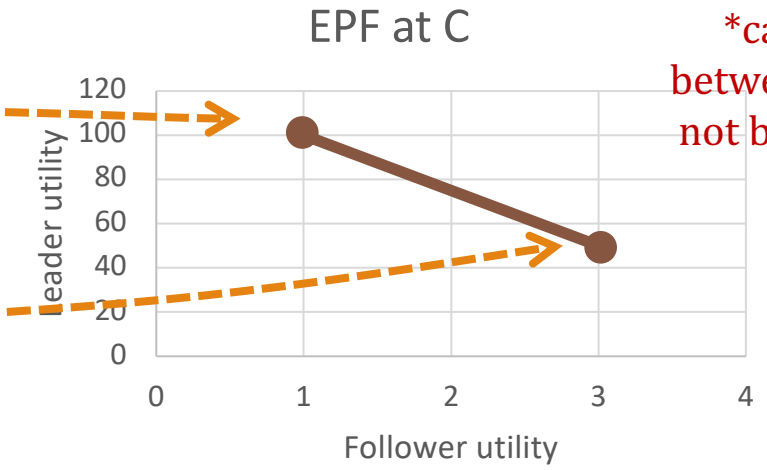
Summary **cannot be** a scalar/fixed sized tuple



Summary needs to solve game for all (k_1, k_2)

How much can Leader get in subgame while giving Follower k_2?

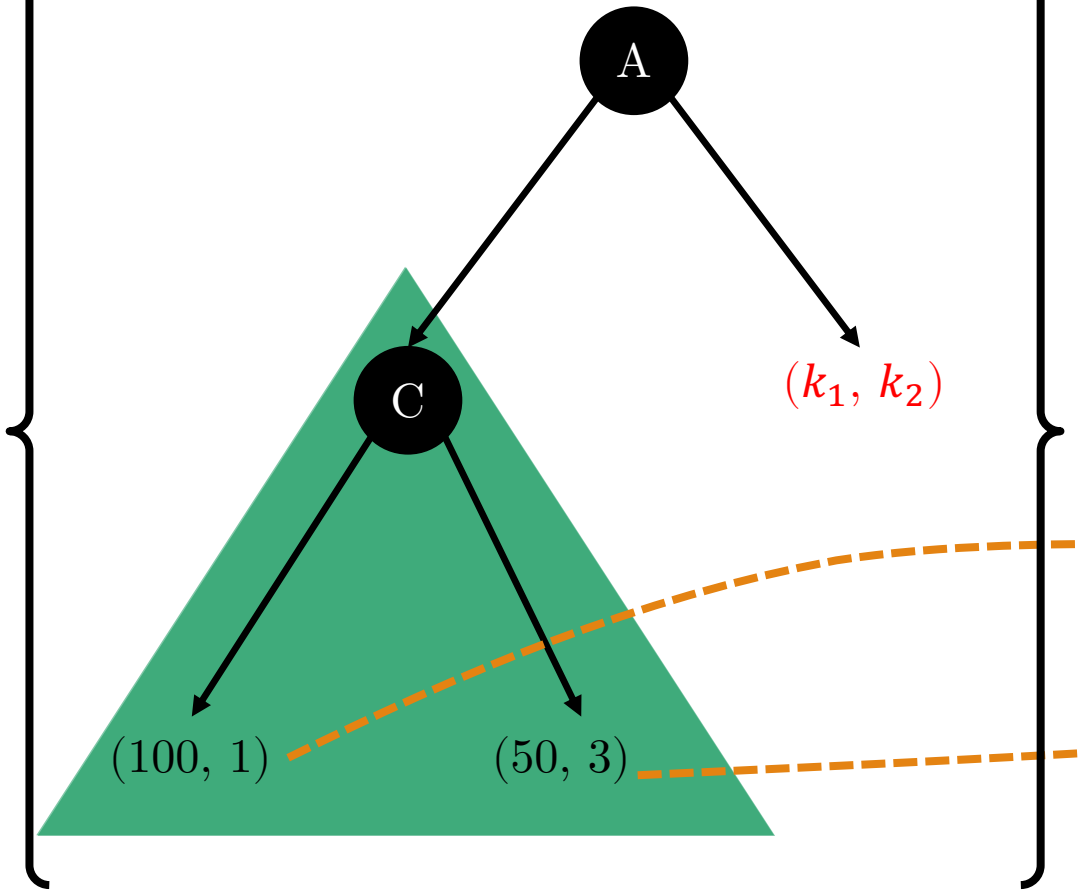
- If $> k_1$, go left, if $< k_1$, let the follower quit
- Need to query this for all possible k_2!



*captures tradeoff between 2 players, need not be linear in general

Enforceable Payoff Frontiers (EPF)

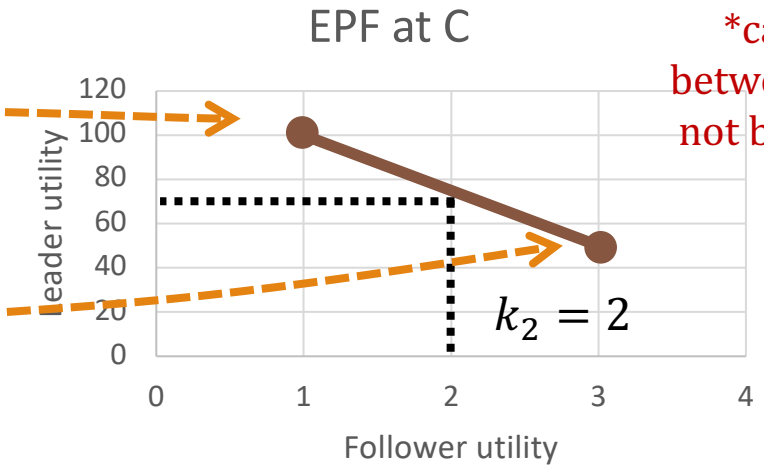
Summary **cannot be** a scalar/fixed sized tuple



Summary needs to solve game for all (k_1, k_2)

How much can Leader get in subgame while giving Follower k_2?

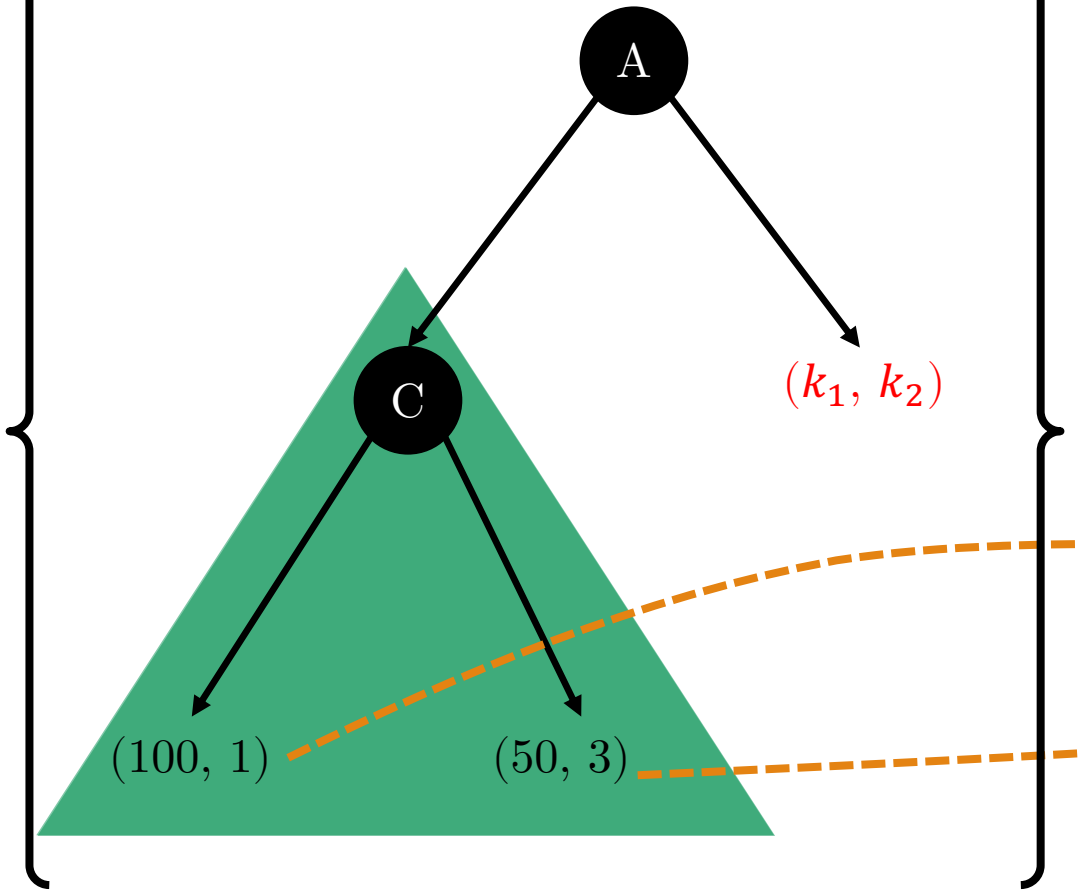
- If $> k_1$, go left, if $< k_1$, let the follower quit
- Need to query this for all possible k_2!



*captures tradeoff between 2 players, need not be linear in general

Enforceable Payoff Frontiers (EPF)

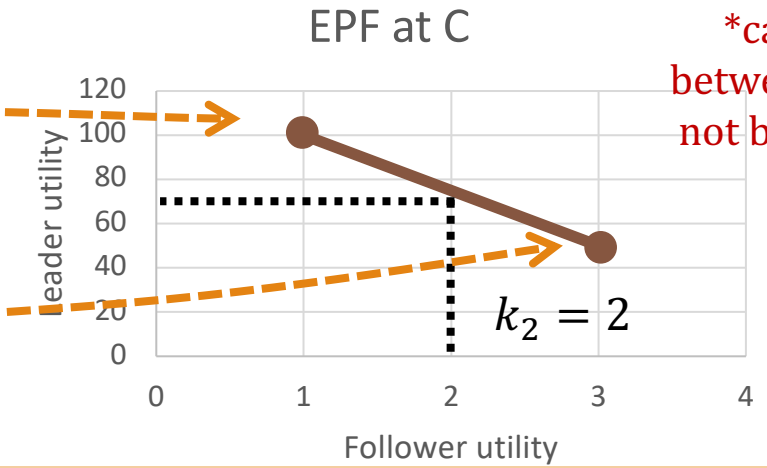
Summary **cannot be** a scalar/fixed sized tuple



Summary needs to solve game for all (k_1, k_2)

How much can Leader get in subgame while giving Follower k_2?

- If $> k_1$, go left, if $< k_1$, let the follower quit
- Need to query this for all possible k_2!



*captures tradeoff between 2 players, need not be linear in general

EPF is the summary of a state that we need to learn!

Bellman Equations

*Prior work (Letchford and Conitzer 2010, Bosansky et. al. 2017)

$$U_s(\mu) = \begin{cases} \left[\bigwedge_{s' \in \mathcal{C}(s)} U_{s'}\right](\mu) & \text{if } s \text{ is a Leader node} \\ \left[\bigwedge_{s' \in \mathcal{C}(s)} U_{s'} \triangleright \tau(s')\right](\mu) & \text{if } s \text{ is a Follower node} \end{cases}$$

Bellman Equations

*Prior work (Letchford and Conitzer 2010, Bosansky et. al. 2017)

$$U_s(\mu) = \begin{cases} \left[\bigwedge_{s' \in C(s)} U_{s'}\right](\mu) & \text{if } s \text{ is a Leader node} \\ \left[\bigwedge_{s' \in C(s)} U_{s'} \triangleright \tau(s')\right](\mu) & \text{if } s \text{ is a Follower node} \end{cases}$$

EPF at s

Target follower payoff

Children of s

Upper Concave Envelope

Left-truncation

Value of grim trigger of siblings

Bellman Equations

*Prior work (Letchford and Conitzer 2010, Bosansky et. al. 2017)

$$U_s(\mu) = \begin{cases} \left[\bigwedge_{s' \in \mathcal{C}(s)} U_{s'}\right](\mu) & \text{if } s \text{ is a Leader node} \\ \left[\bigwedge_{s' \in \mathcal{C}(s)} U_{s'} \triangleright \tau(s')\right](\mu) & \text{if } s \text{ is a Follower node} \end{cases}$$

EPF at s

Target follower payoff

Children of s

Upper Concave Envelope

Left-truncation

Value of grim trigger of siblings

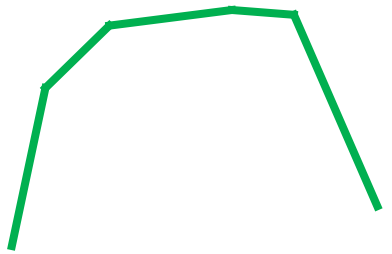
$$V(s) = \begin{cases} \max_{s' \in \mathcal{C}(s)} V(s') & \text{if } s \text{ belongs to max player} \\ \min_{s' \in \mathcal{C}(s)} V(s') & \text{if } s \text{ belongs to min player} \end{cases}$$

Analogous to 'traditional' Bellman Equations for minimax games, but with **functionals** acting on EPFs

Learning EPFs via Fitted Value Iteration

Learning EPFs via Fitted Value Iteration

Representation: Neural network predicts EPFs for each state

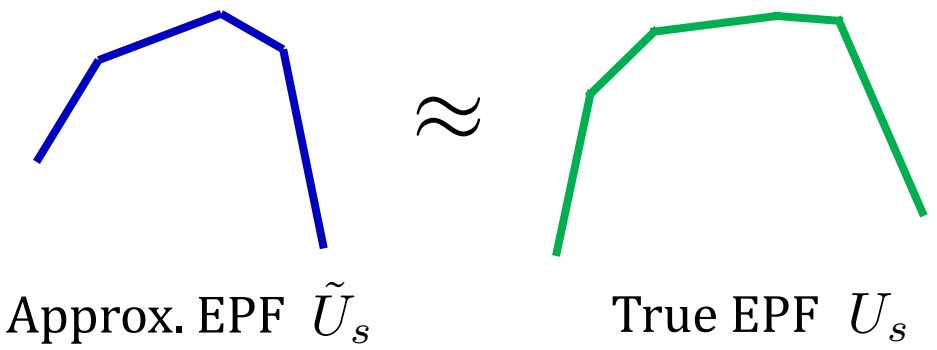


True EPF U_s

Learning EPFs via Fitted Value Iteration

Representation: Neural network predicts EPFs for each state

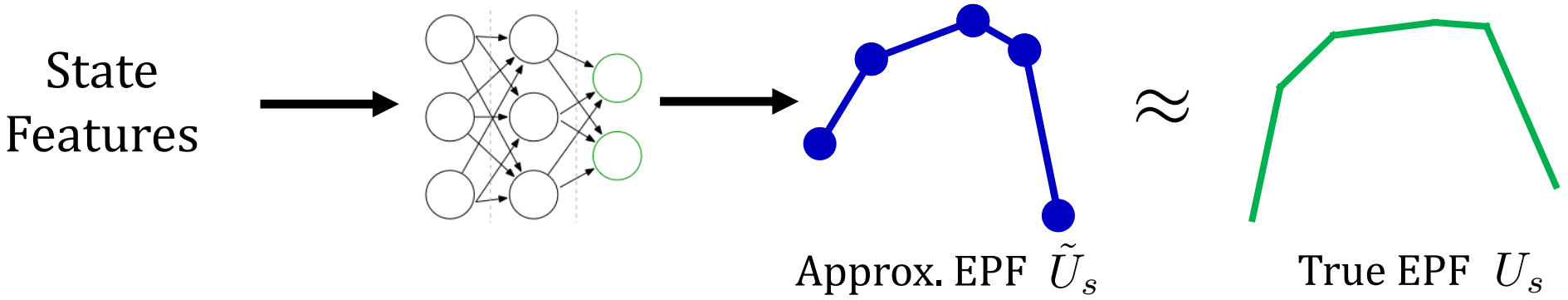
- EPF at state s, U_s is approximated by a piecewise linear function with m knots



Learning EPFs via Fitted Value Iteration

Representation: Neural network predicts EPFs for each state

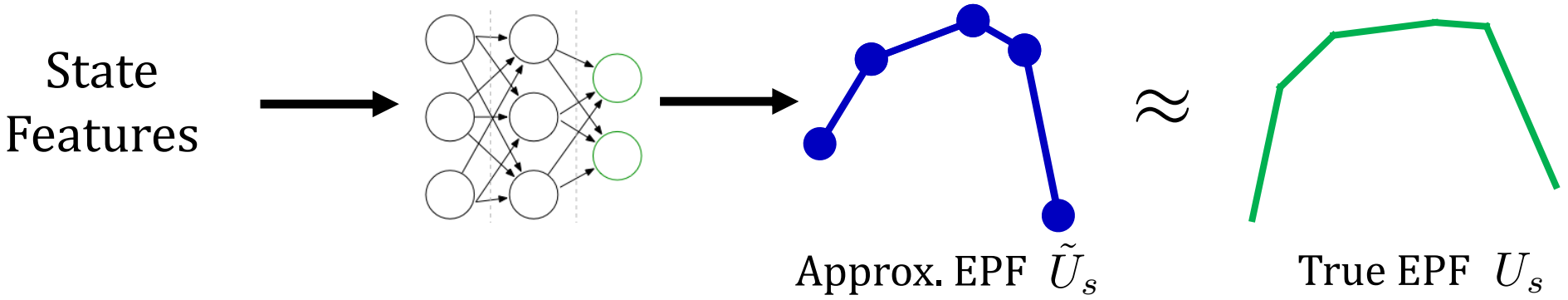
- EPF at state s, U_s is approximated by a piecewise linear function with m knots
- Network maps state features to coordinates of knots of predicted EPF



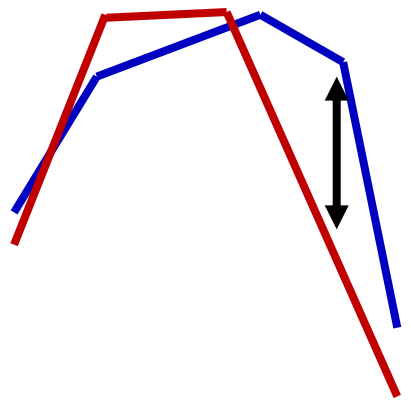
Learning EPFs via Fitted Value Iteration

Representation: Neural network predicts EPFs for each state

- EPF at state s, U_s is approximated by a piecewise linear function with m knots
- Network maps state features to coordinates of knots of predicted EPF



Training: For a set of sampled states $\{s\}$, minimize the *Bellman Loss*



$$L_\infty(\tilde{U}_s, \tilde{U}'_s) = \max_x |\tilde{U}_s(x) - \tilde{U}'_s(x)|$$

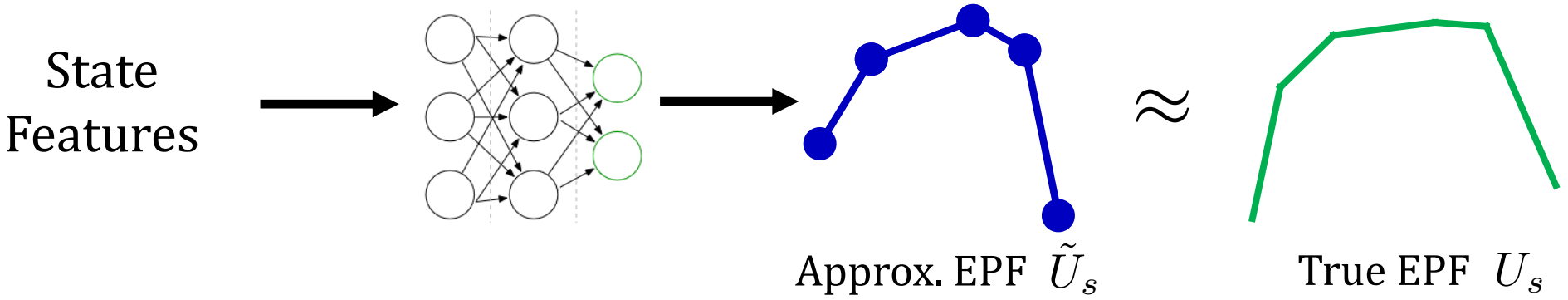
EPF estimated from network

EPF estimated using one-step lookahead from Bellman Equations

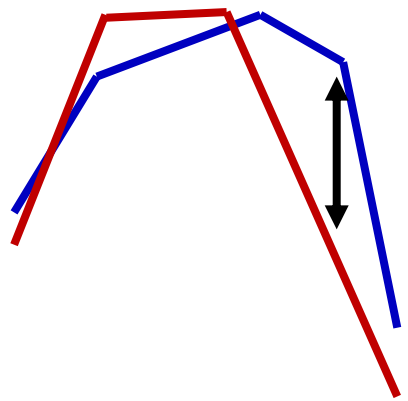
Learning EPFs via Fitted Value Iteration

Representation: Neural network predicts EPFs for each state

- EPF at state s, U_s is approximated by a piecewise linear function with m knots
- Network maps state features to coordinates of knots of predicted EPF



Training: For a set of sampled states $\{s\}$, minimize the *Bellman Loss*



$$L_\infty(\tilde{U}_s, \tilde{U}'_s) = \max_x |\tilde{U}_s(x) - \tilde{U}'_s(x)|$$

EPF estimated from network

EPF estimated using one-step lookahead from Bellman Equations

Equal to 0 iff Bellman Equations are obeyed

Theoretical Guarantees

Theorem 3 (FA Error). *If* $L_\infty(\tilde{U}_s, \tilde{U}_s^{target}) \leq \epsilon$ *for all* $s \in \mathcal{S}$, *then* $|R_1(\tilde{\pi}) - R_1(\pi^*)| = \mathcal{O}(D\epsilon)$ *where* D *is the depth of* G *and* π^* *is the optimal strategy.*

When using policy from learned EPFs, if Bellman loss is low for **all states**, then we are close to optimal.

Theoretical Guarantees

Theorem 3 (FA Error). *If* $L_\infty(\tilde{U}_s, \tilde{U}_s^{target}) \leq \epsilon$ *for all* $s \in \mathcal{S}$, *then* $|R_1(\tilde{\pi}) - R_1(\pi^*)| = \mathcal{O}(D\epsilon)$ *where* D *is the depth of* G *and* π^* *is the optimal strategy.*

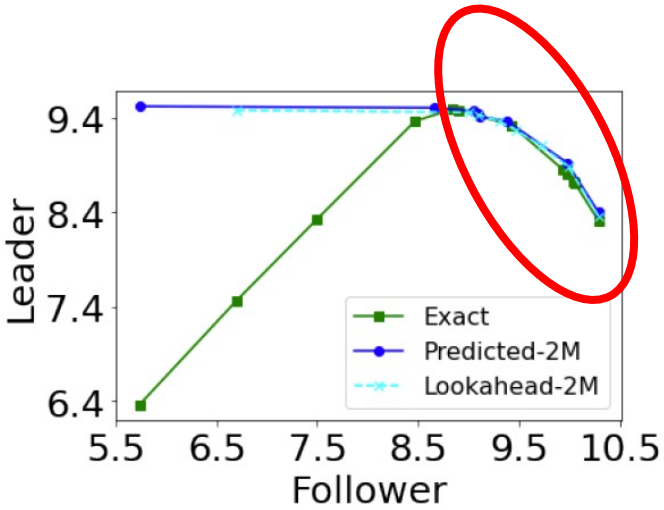
When using policy from learned EPFs, if Bellman loss is low for **all states**, then we are close to optimal.

These guarantees are not coincidental, they are achieved by a careful choice of loss function and network design

Experimental Results

Resource Collection game

- 7x7 grid, 5 steps each (depth of 10), branching factor 5
  - Small enough to solve analytically for testing
- Near optimum solution in all cases
- In 9/10 cases, outperform subgame perfect equilibrium
  - In the bad case, subgame perfect equilibrium is exactly optimal

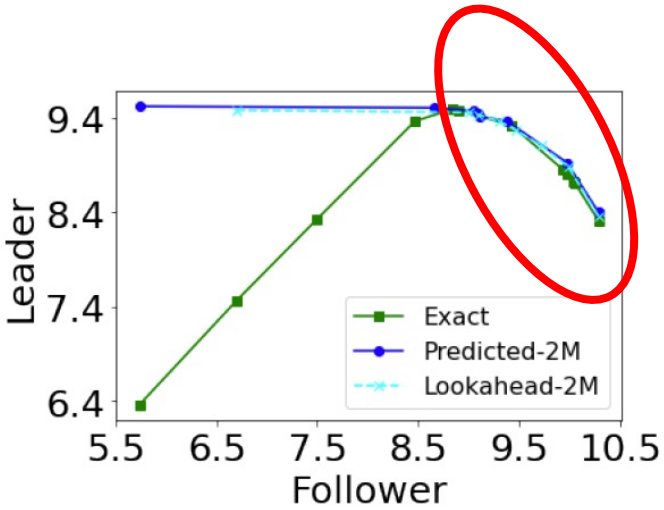


EPFs at the root node

Experimental Results

Resource Collection game

- 7x7 grid, 5 steps each (depth of 10), branching factor 5
  - Small enough to solve analytically for testing
- Near optimum solution in all cases
- In 9/10 cases, outperform subgame perfect equilibrium
  - In the bad case, subgame perfect equilibrium is exactly optimal



EPFs at the root node

Tantrum game *See paper for details!

- Repeated game with a ~1e15 states — too large to traverse
  - Has a special structure so we can compute exact equilibrium for testing
- We obtain optimal strategy in almost all cases

The takeaway here

In *some* general-sum games, sometimes there may not be any simple notions of *value*

These can take rather complex forms, e.g., *sets* of achievable payoffs, and their Bellman equations can be rather complex

Not so amenable to standard ML techniques (?)

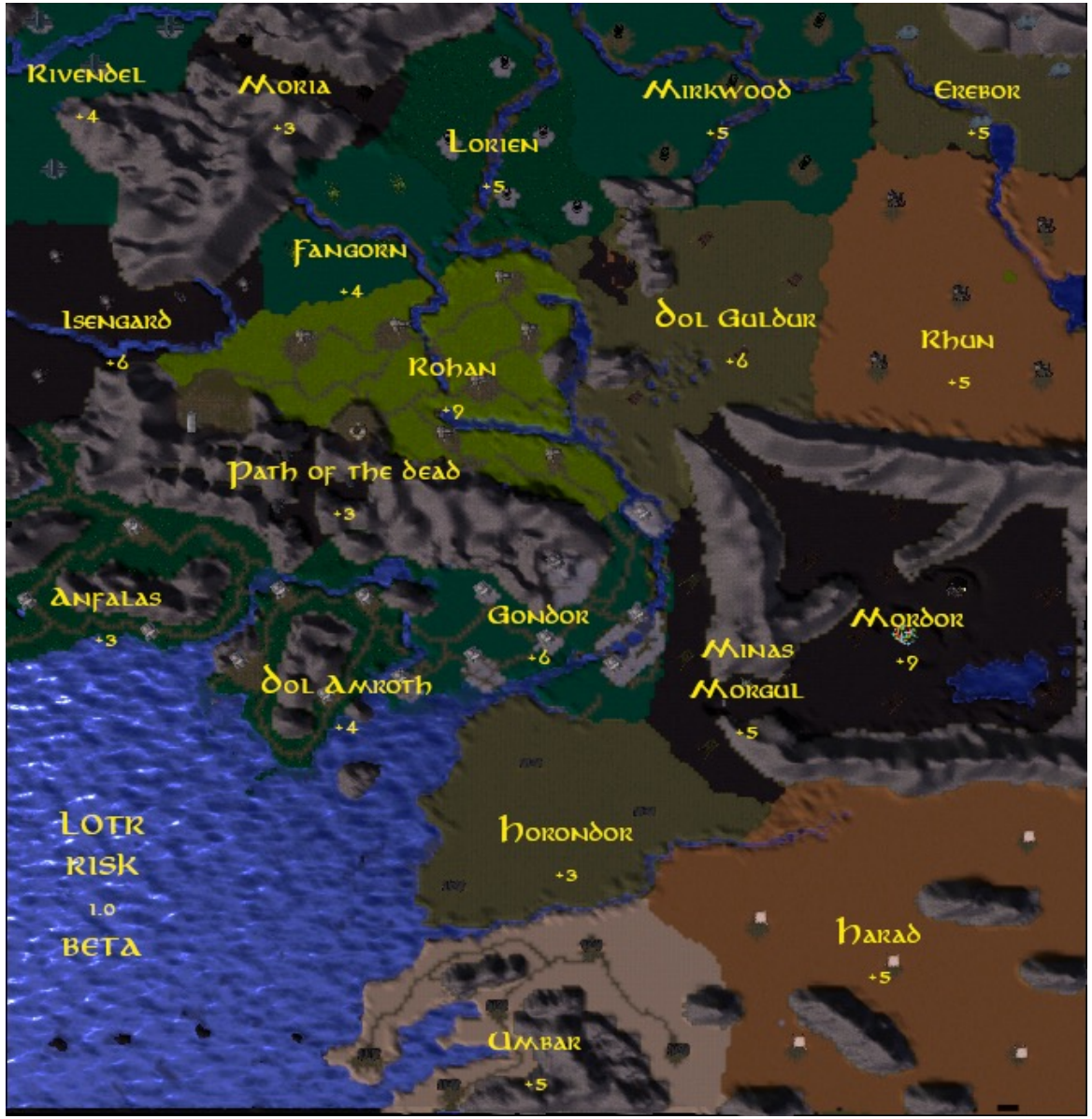
Multi-defender Security Games with Schedules

Zimeng Song[1], **Chun Kai Ling[2]**, Fei Fang[3]

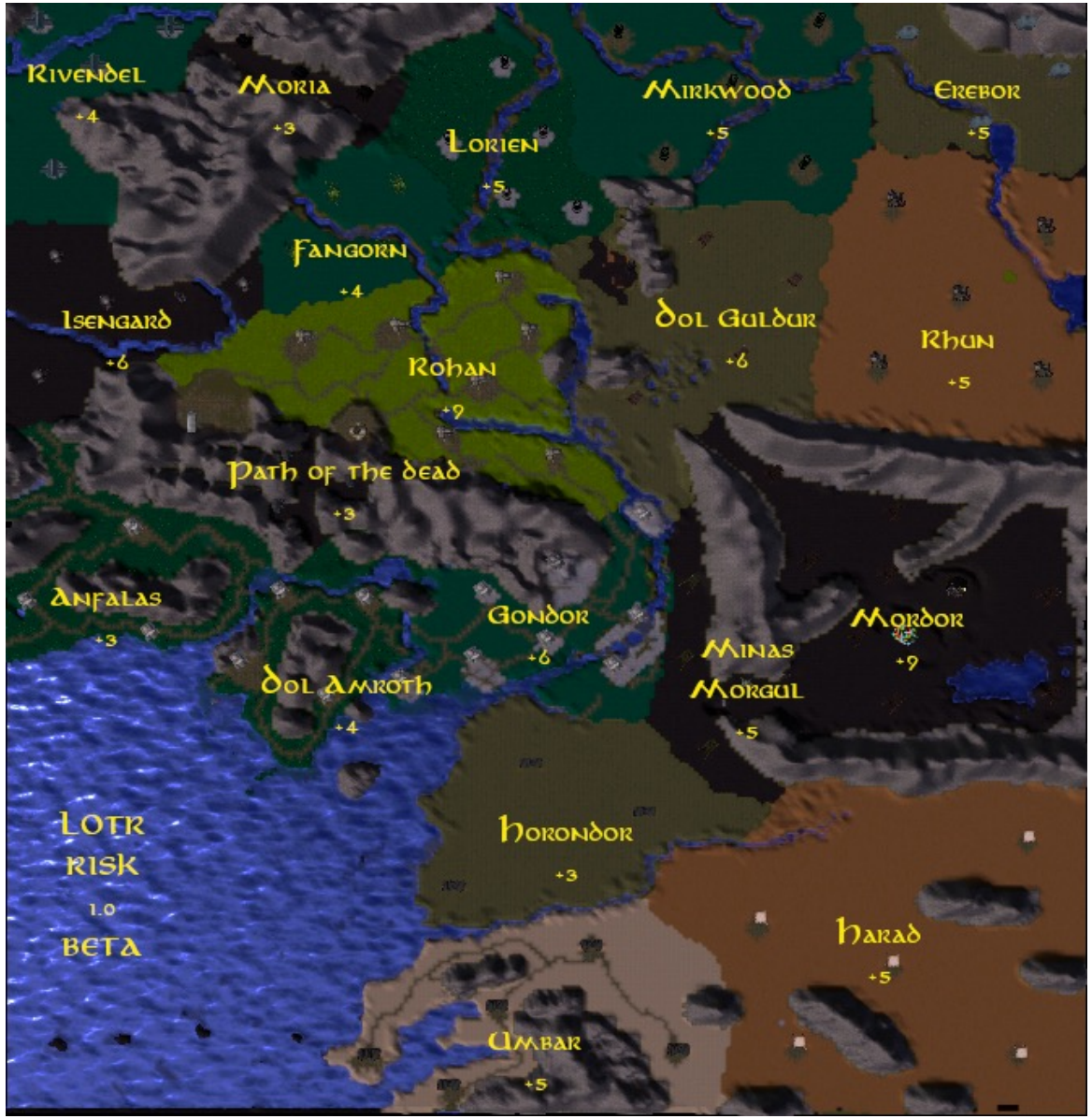
[1]Independent Researcher, [2]Columbia University, [3]Carnegie Mellon University

GameSec 2023 Best Paper Award

Battle for Middle Earth



Battle for Middle Earth

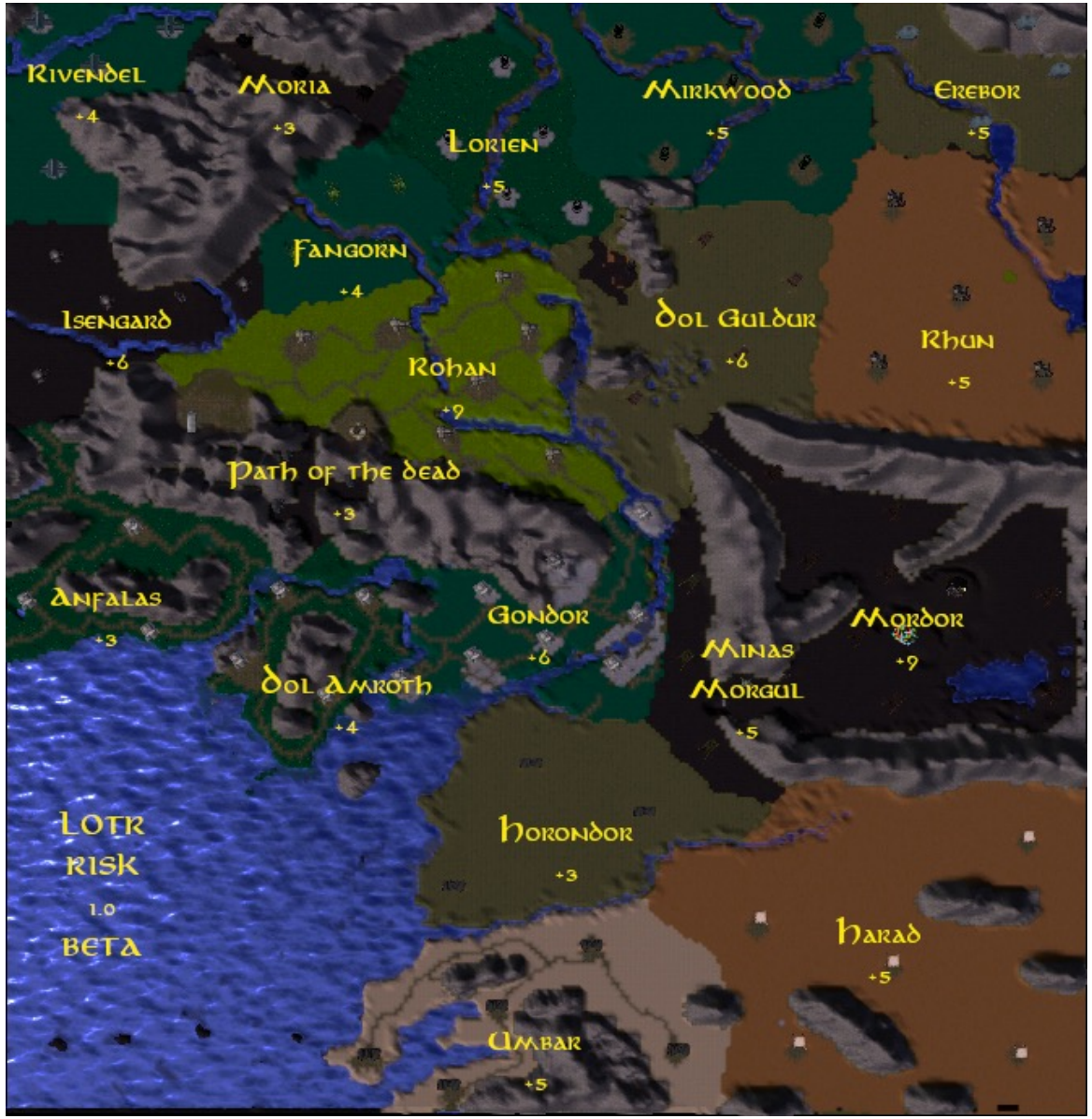


v.s.

attacker

defenders

Battle for Middle Earth



v.s.

attacker defenders

Defenders *each* choose how to distribute their armies. Orcs then choose a target (region) to attack.

Battle for Middle Earth

v.s.

attacker defenders

Defenders *each* choose how to distribute their armies. Orcs then choose a target (region) to attack.

Battle for Middle Earth

v.s.

attacker

defenders

Defenders *each* choose how to distribute their armies. Orcs then choose a target (region) to attack.

Heterogenous defenders

Battle for Middle Earth

v.s.

attacker

defenders

Defenders *each* choose how to distribute their armies. Orcs then choose a target (region) to attack.

Heterogenous defenders

Defensive Schedules

Battle for Middle Earth

v.s.

attacker

defenders

Defenders *each* choose how to distribute their armies. Orcs then choose a target (region) to attack.

Heterogenous defenders

Defensive Schedules

Is there a "stable" allocation of defensive resources?

Other Applications

Military Alliances

Coordinating Security across agencies

Cooperation between firms

Security Games

- Stackelberg leadership model [Stackelberg, 1934]
- Solving matrix games, multiple LP method [Conitzer]
- Water filling algorithm for security games [Kiekintveld et. al., 2009]

Security Games

- Stackelberg leadership model [Stackelberg, 1934]
- Solving matrix games, multiple LP method [Conitzer]
- Water filling algorithm for security games [Kiekintveld et. al., 2009]

Schedules: defend multiple targets at once

- Subset of Schedules is a Schedule [Korzhyk et. al., 2010]
- Scalable methods + Applications [Many papers]

Security Games

- Stackelberg leadership model [Stackelberg, 1934]
- Solving matrix games, multiple LP method [Conitzer]
- Water filling algorithm for security games [Kiekintveld et. al., 2009]

Schedules: defend multiple targets at once

- Subset of Schedules is a Schedule [Korzhyk et. al., 2010]
- Scalable methods + Applications [Many papers]

Multi-Defender

- Nuances when handling tiebreaking [Lou and Vorobeychik, 2015]
- Multi-defender water filling algorithm [Gan et. al., 2018, 2022]
  - Equilibrium **exists** and can be computed in **polynomial time**

Security Games

- Stackelberg leadership model [Stackelberg, 1934]
- Solving matrix games, multiple LP method [Conitzer]
- Water filling algorithm for security games [Kiekintveld et. al., 2009]

Schedules: defend multiple targets at once

- Subset of Schedules is a Schedule [Korzhyk et. al., 2010]
- Scalable methods + Applications [Many papers]

Multi-Defender

- Nuances when handling tiebreaking [Lou and Vorobeychik, 2015]
- Multi-defender water filling algorithm [Gan et. al., 2018, 2022]
  - Equilibrium **exists** and can be computed in **polynomial time**

Qn: What happens when there are multiple defenders **and** schedules?

Existence? Under what conditions? Computation? Complexity?

Outline of this section

Outline of this section

Nash Stackelberg Equilibrium

* Disclaimer: ours is a simpler setting than most security games Gan et. al.

Outline of this section

Nash Stackelberg Equilibrium

* Disclaimer: ours is a simpler setting than most security games Gan et. al.

Non-existence of equilibrium

- Equilibrium may not exist even with simple reward structures.
- Caused by combination of schedules **and** multiple defenders

Outline of this section

Nash Stackelberg Equilibrium

* Disclaimer: ours is a simpler setting than most security games Gan et. al.

Non-existence of equilibrium

- Equilibrium may not exist even with simple reward structures.
- Caused by combination of schedules **and** multiple defenders

2 defenders: equilibrium exists + poly time computation

- Two-defenders + Subset-of-Schedule-is-a-Schedule (SSAS)

Outline of this section

Nash Stackelberg Equilibrium

* Disclaimer: ours is a simpler setting than most security games Gan et. al.

Non-existence of equilibrium

- Equilibrium may not exist even with simple reward structures.
- Caused by combination of schedules **and** multiple defenders

2 defenders: equilibrium exists + poly time computation

- Two-defenders + Subset-of-Schedule-is-a-Schedule (SSAS)

Other Special Cases

- Multiple-defenders + SSAS + Monotone Schedules (MS)
- Dealing with exponentially many schedules

N defenders, T targets, single attacker

N defenders, T targets, single attacker

Defenders place **coverage** over targets *Think in terms of defensive levels, not probabilities

- High coverage ➔ better defended target

N defenders, T targets, single attacker

Defenders place **coverage** over targets *Think in terms of defensive levels, not probabilities

- High coverage ➔ better defended target

Each defender i has a finite set of m^i schedules $\mathrm{S}^i = \{s_1^i, \dots s_{m^i}^i\}$

- $s_z^i \in R_+^T$ is the coverage contribution of schedule z over each target
- Defender chooses a **convex combination** $x^i = (x_1^i, \dots, x_{m^i}^i)$ over schedules.
- Coverage is the weighted sum over every schedule $\sum_z x^i(z) \cdot s_z^i(j)$
- Set of achievable coverage for each player is given by

$$V^i = \left\{ v \in R_+^T | \exists x^i \; s.t. \forall j \in [T] \; v(j) = \sum_z x^i(z) \cdot s_z^i(j) \right\}$$

N defenders, T targets, single attacker

Defenders place **coverage** over targets *Think in terms of defensive levels, not probabilities

- High coverage ➔ better defended target

Each defender i has a finite set of m^i schedules $\mathrm{S}^i = \{s_1^i, \dots s_{m^i}^i\}$

- $s_z^i \in R_+^T$ is the coverage contribution of schedule z over each target
- Defender chooses a **convex combination** $x^i = (x_1^i, \dots, x_{m^i}^i)$ over schedules.
- Coverage is the weighted sum over every schedule $\sum_z x^i(z) \cdot s_z^i(j)$
- Set of achievable coverage for each player is given by

$$V^i = \left\{ v \in R_+^T | \exists x^i \; s.t. \forall j \in [T] \; v(j) = \sum_z x^i(z) \cdot s_z^i(j) \right\}$$

Total coverage is the **sum** of coverage from each defender

* Slightly different from usual setting

Feasible Coverage

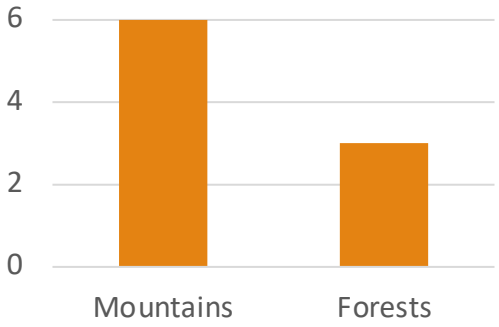
Mountains (t_1)

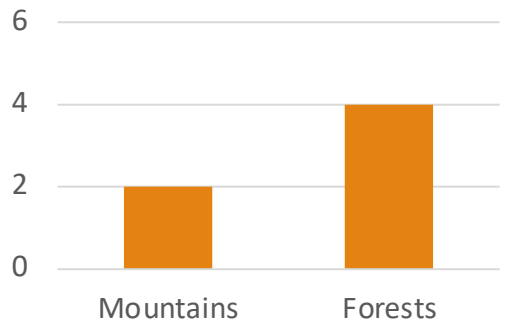
Forests (t_2)

Feasible Coverage

Mountains (t_1)

Forests (t_2)



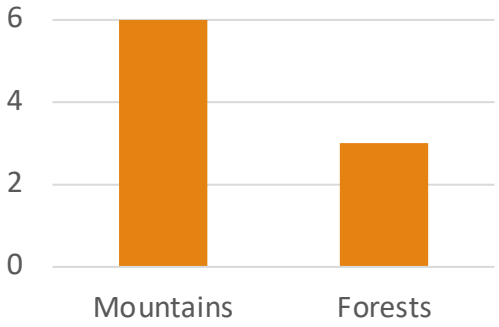


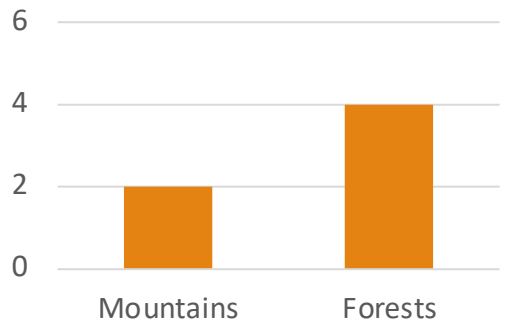
Guard mountains: $s_1^1 = (6, 3)$ Guard forests: $s_2^1 = (2, 4)$

Feasible Coverage

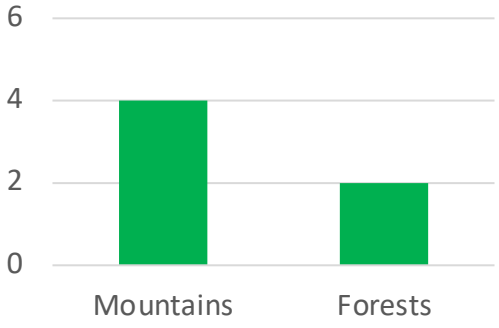
Mountains (t_1)

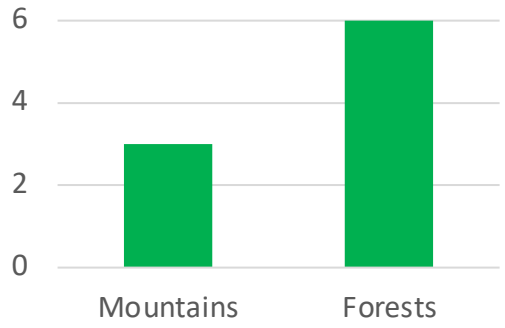
Forests (t_2)





Guard mountains: $s_1^1 = (6, 3)$ Guard forests: $s_2^1 = (2, 4)$

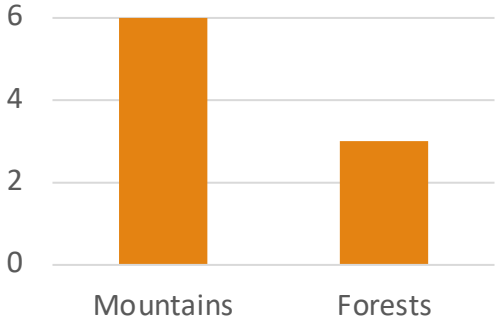


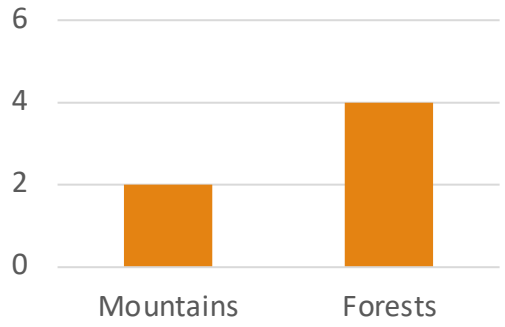


Feasible Coverage

Mountains (t_1)

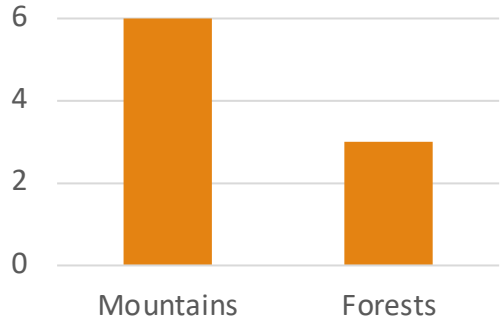
Forests (t_2)





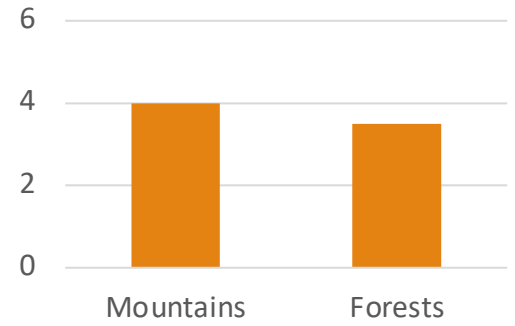
Guard mountains: $s_1^1 = (6, 3)$

Guard forests: $s_2^1 = (2, 4)$



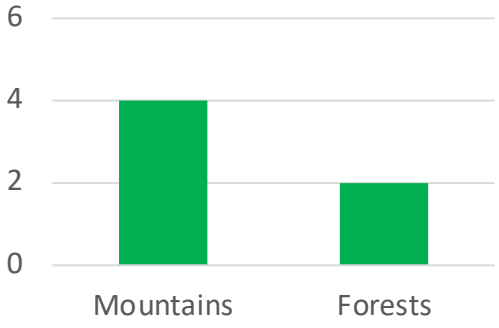
Mountains only: $x^1 = (1, 0)$

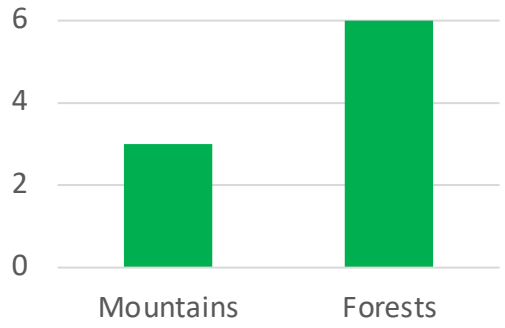
$v^1 = (6, 3)$



Even split: $x^1 = (\frac{1}{2}, \frac{1}{2})$

$v^1 = (4, 3.5)$

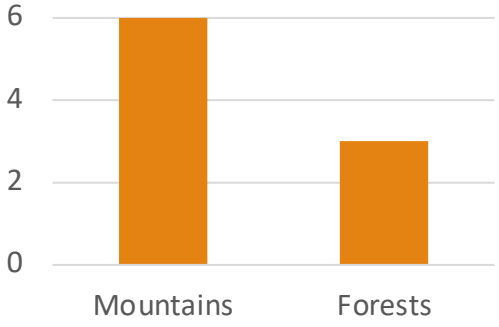




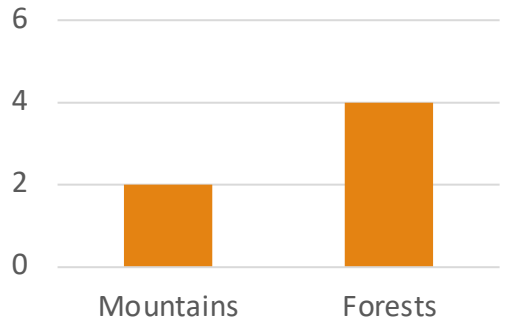
Feasible Coverage

Mountains (t_1)

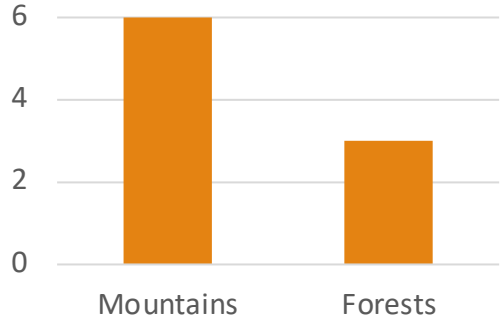
Forests (t_2)



Guard mountains: $s_1^1 = (6, 3)$

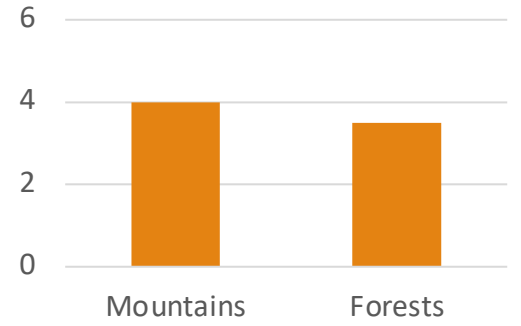


Guard forests: $s_2^1 = (2, 4)$



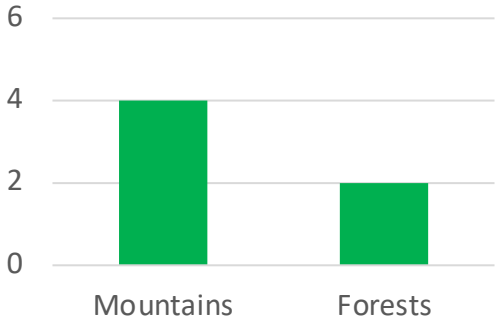
Mountains only: $x^1 = (1, 0)$

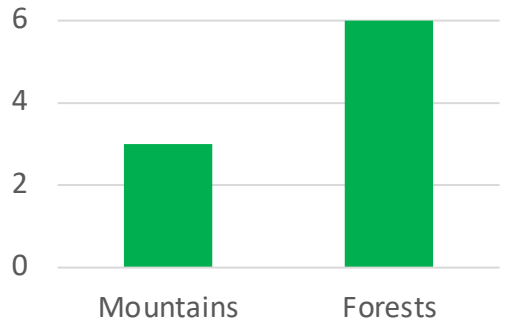
$v^1 = (6, 3)$

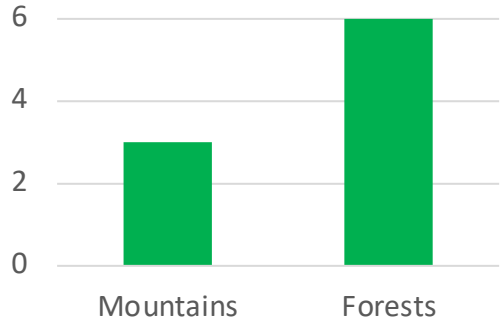


Even split: $x^1 = (\frac{1}{2}, \frac{1}{2})$

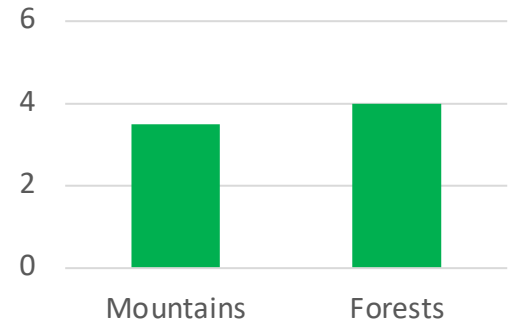
$v^1 = (4, 3.5)$







Forests only: $v^2 = (3, 6)$

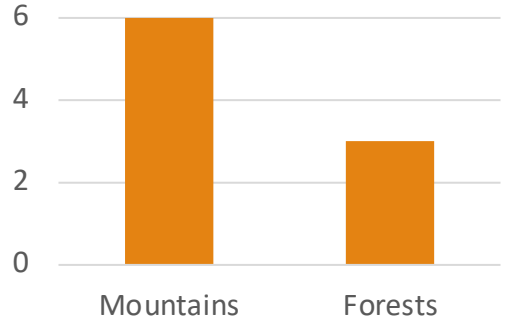


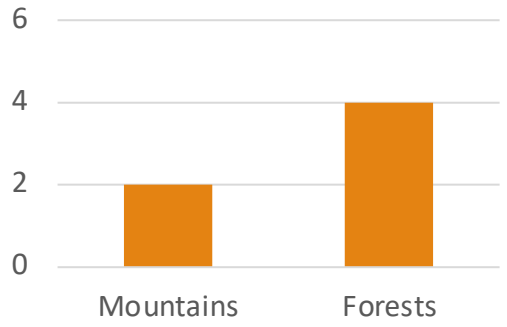
Even split: $v^2 = (3.5, 4)$

Feasible Coverage

Mountains (t_1)

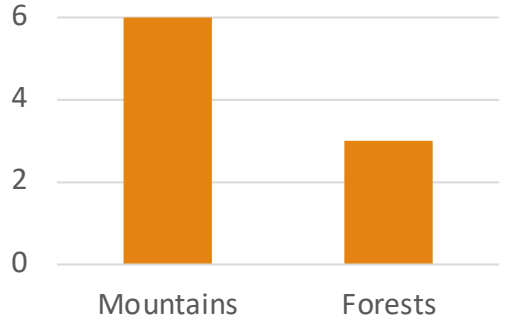
Forests (t_2)

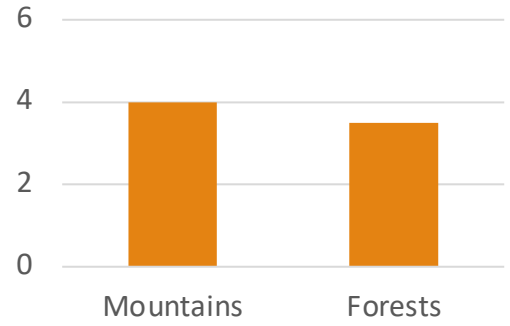




Guard mountains: $s_1^1 = (6, 3)$

Guard forests: $s_2^1 = (2, 4)$



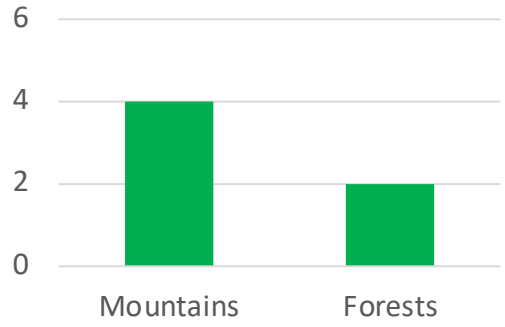


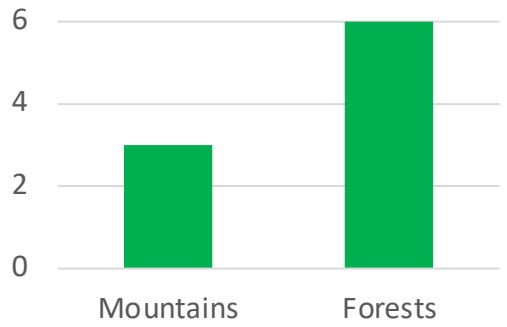
Mountains only: $x^1 = (1, 0)$

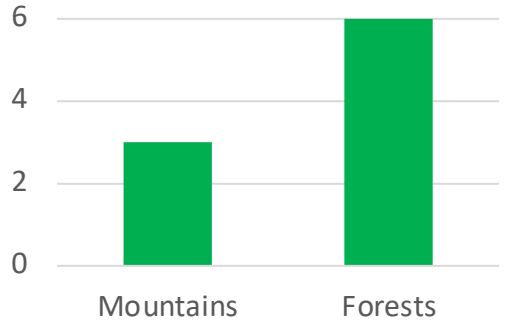
$v^1 = (6, 3)$

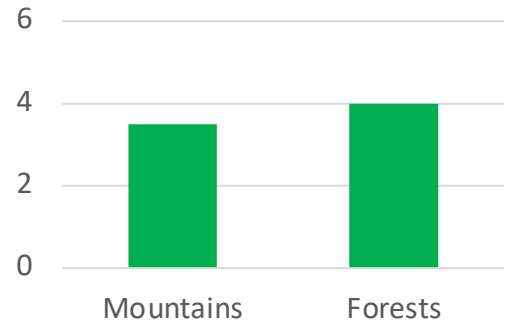
Even split: $x^1 = (\frac{1}{2}, \frac{1}{2})$

$v^1 = (4, 3.5)$



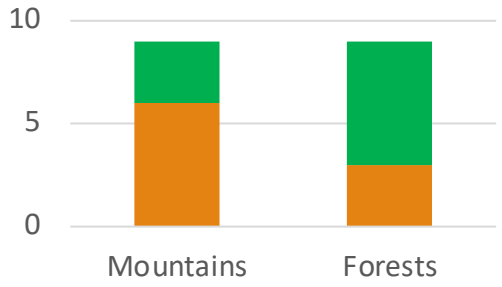


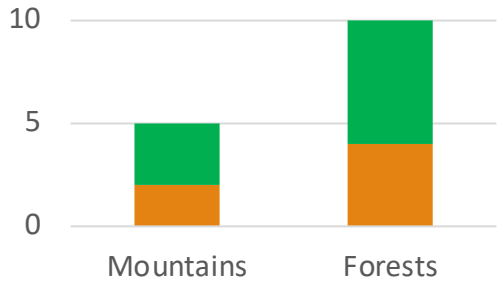


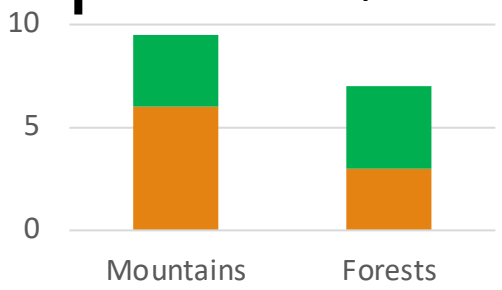


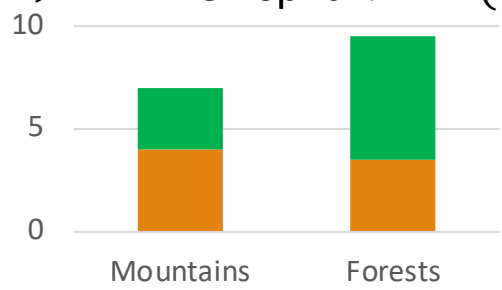
Forests only: $v^2 = (3, 6)$

Even split: $v^2 = (3.5, 4)$









Nash Stackelberg Equilibrium (NSE)

Nash Stackelberg Equilibrium (NSE)

Each defender i has a **preference ordering** over targets $t \in [T]$

- $t \succ_i t'$ iff defender i prefers t to be attacked over t' * caution: t' is more "valuable" than t
- Simple case of coverage-independent payoffs (Gan et. al., 2022)

Nash Stackelberg Equilibrium (NSE)

Each defender i has a **preference ordering** over targets $t \in [T]$

- $t \succ_i t'$ iff defender i prefers t to be attacked over t' * caution: t' is more "valuable" than t
- Simple case of coverage-independent payoffs (Gan et. al., 2022)

Attacker chooses some $t^* \in [T]$ **with lowest coverage** to attack

- Ties can be broken arbitrarily

Nash Stackelberg Equilibrium (NSE)

Each defender i has a **preference ordering** over targets $t \in [T]$

- $t \succ_i t'$ iff defender i prefers t to be attacked over t' * caution: t' is more "valuable" than t
- Simple case of coverage-independent payoffs (Gan et. al., 2022)

Attacker chooses some $t^* \in [T]$ **with lowest coverage** to attack

- Ties can be broken arbitrarily

If a defender **deviates** to new coverage, attacker **adjusts** t^*

- Ties are broken against deviating defender * technical assumption to avoid trivial deviations

Nash Stackelberg Equilibrium (NSE)

Each defender i has a **preference ordering** over targets $t \in [T]$

- $t \succ_i t'$ iff defender i prefers t to be attacked over t' * caution: t' is more "valuable" than t
- Simple case of coverage-independent payoffs (Gan et. al., 2022)

Attacker chooses some $t^* \in [T]$ **with lowest coverage** to attack

- Ties can be broken arbitrarily

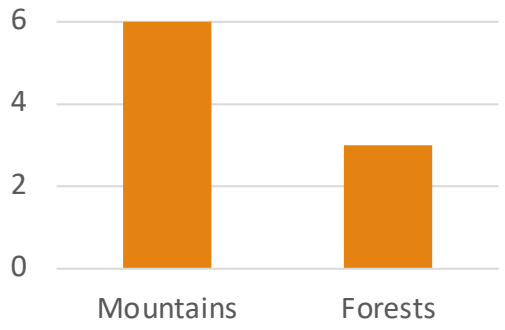
If a defender **deviates** to new coverage, attacker **adjusts** t^*

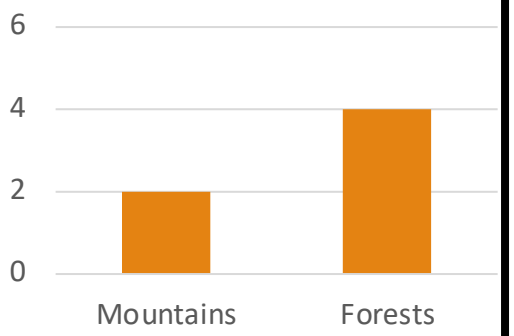
- Ties are broken against deviating defender * technical assumption to avoid trivial deviations

$(v^1, \dots, v^N, t^*)$ is a NSE iff

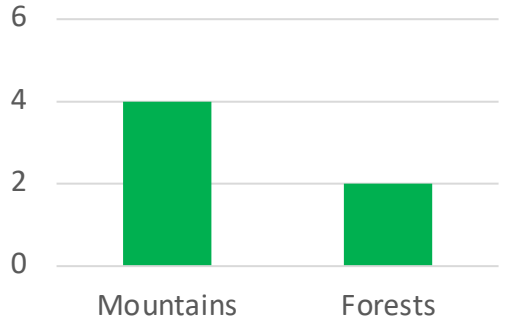
- $v^i \in V^i$ [defender coverages are all achievable]
- $t^* \in argmin(v^1 + \cdots + v^N)$ [attacker is best responding]
- No defender i can **unilaterally** deviate $v^i \to \tilde{v}^i$ such that attacker is incentivized to attack another $\tilde{t}^* \succ_i t^*$ [incentive compatibility]

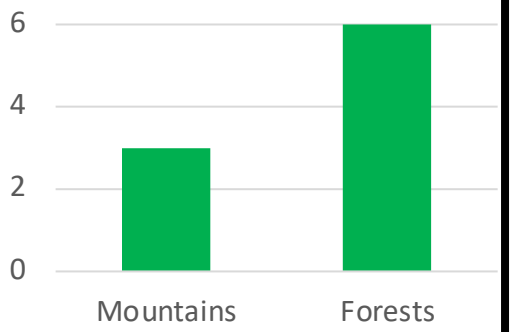
Example of NSE





Forests $\succ_1$ Mountains

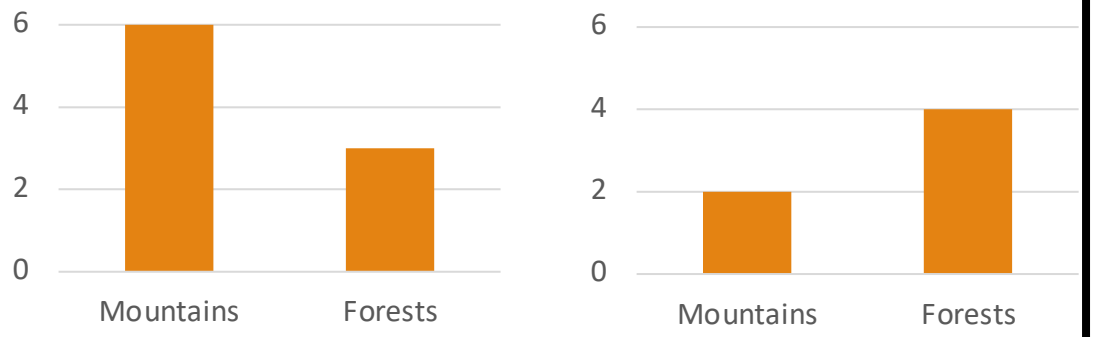




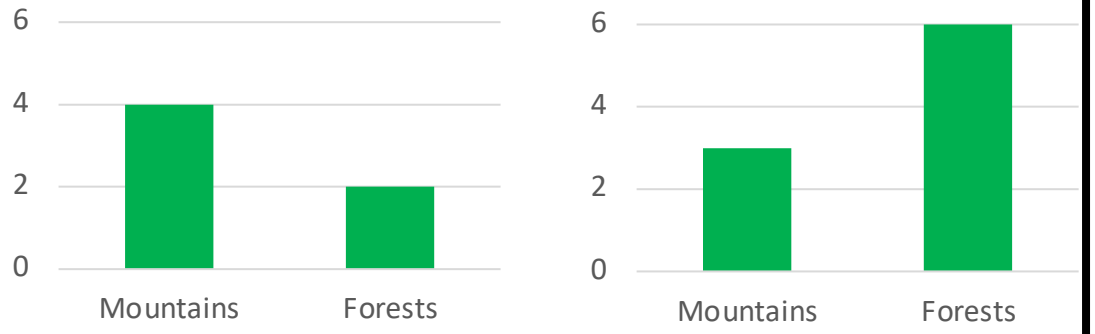
Mountains $\succ_2$ Forests

Example of NSE

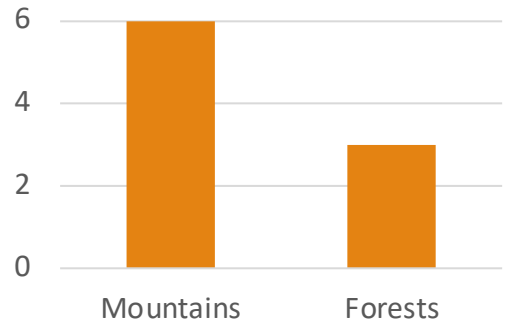
Claim: $((6,3),(3,6),Mountain)$ is NSE



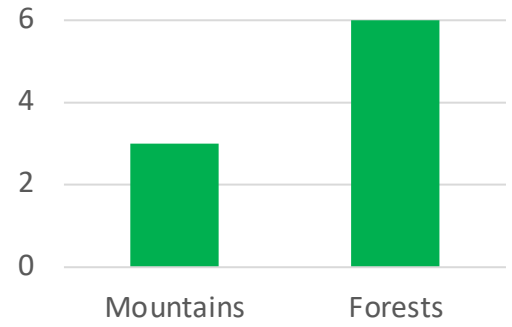
Forests $\succ_1$ Mountains



Mountains $\succ_2$ Forests



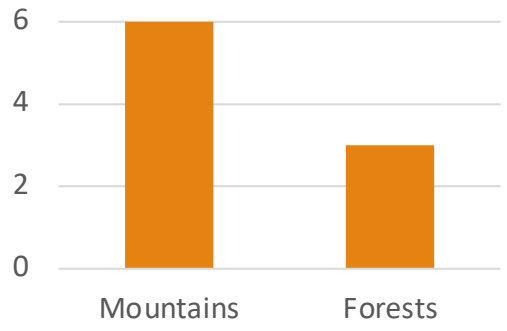
$x^1 = (1,0) \rightarrow v^1 = (6,3)$

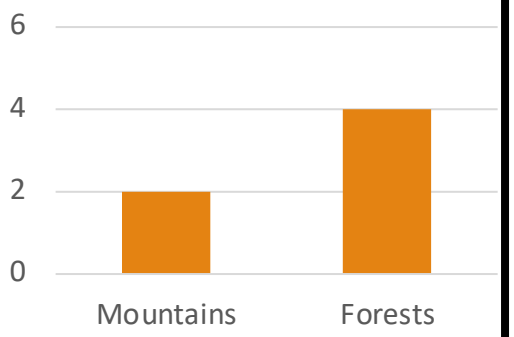


$x^2 = (0,1) \rightarrow v^2 = (3,6)$

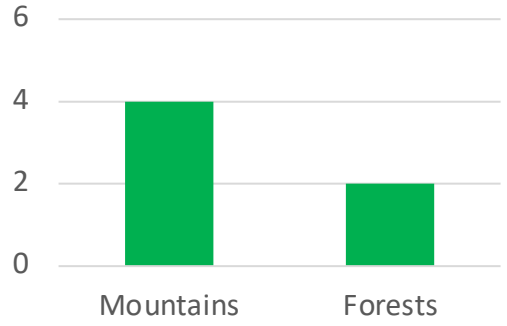
Example of NSE

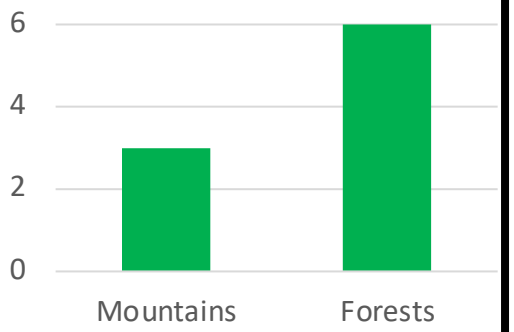
Claim: $((6,3),(3,6),Mountain)$ is NSE



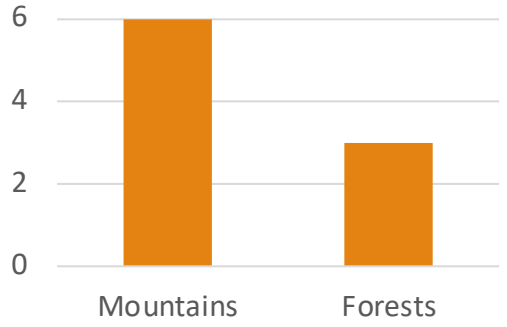


Forests $\succ_1$ Mountains



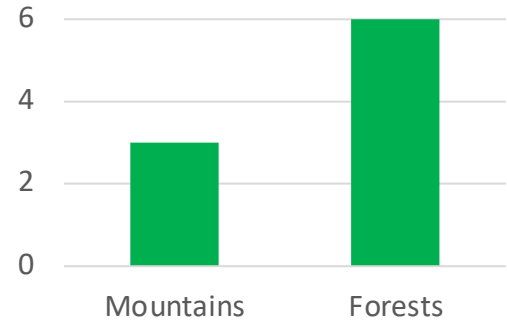


Mountains $\succ_2$ Forests



$x^1 = (1,0) \rightarrow v^1 = (6,3)$

Feasible!

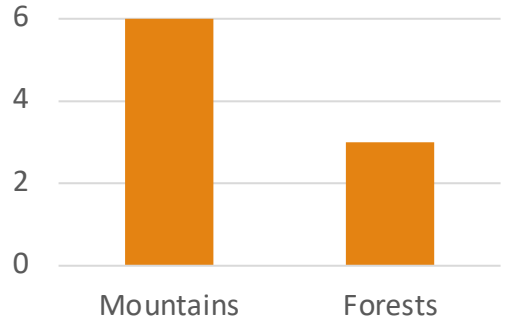


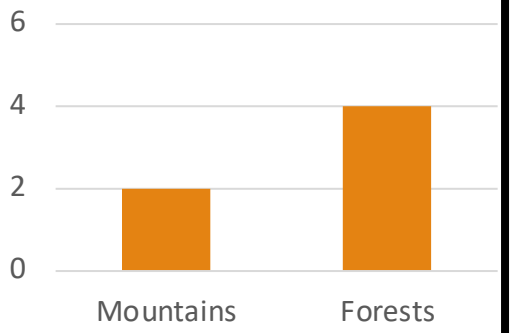
$x^2 = (0,1) \rightarrow v^2 = (3,6)$

Feasible!

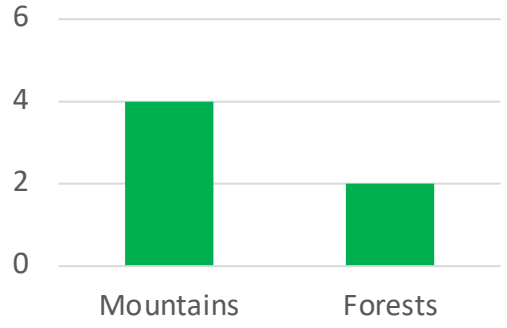
Example of NSE

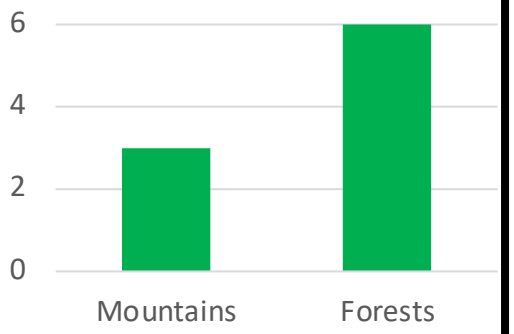
Claim: $((6,3),(3,6),Mountain)$ is NSE



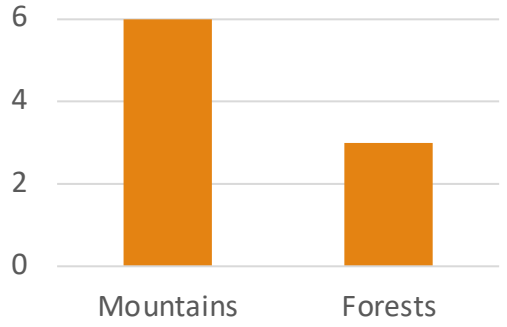


Forests $\succ_1$ Mountains



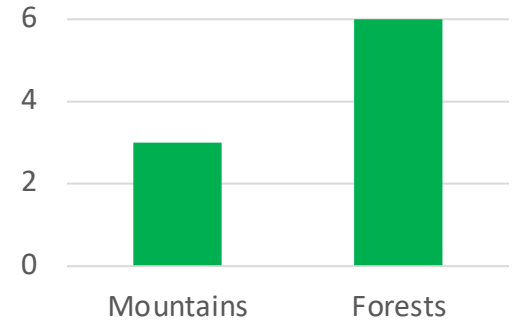


Mountains $\succ_2$ Forests



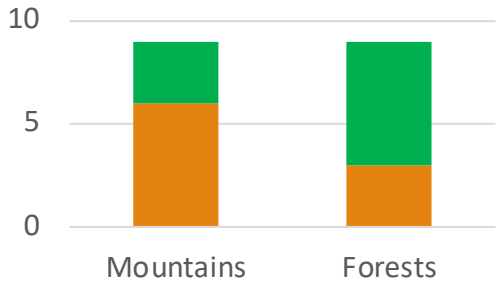
$x^1 = (1,0) \rightarrow v^1 = (6,3)$

Feasible!



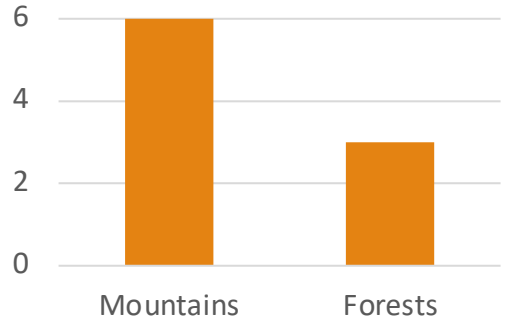
$x^2 = (0,1) \rightarrow v^2 = (3,6)$

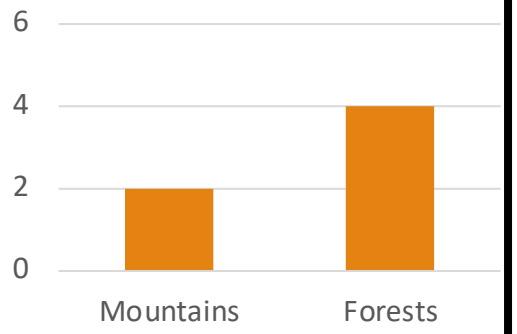
Feasible!



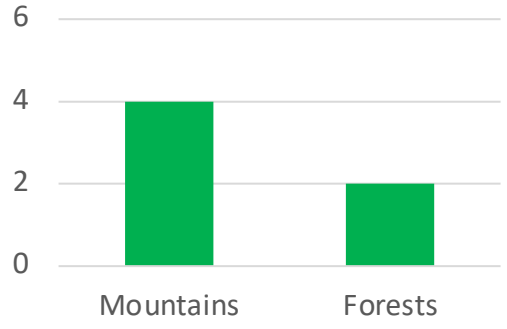
Attacker is best-responding!

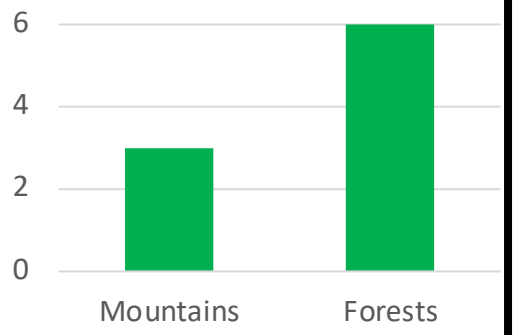
Example of NSE





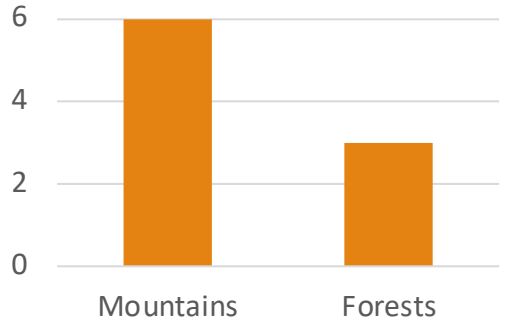
Forests $\succ_1$ Mountains





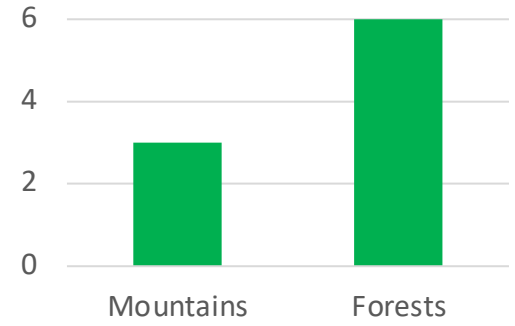
Mountains $\succ_2$ Forests

Claim: $((6,3),(3,6),Mountain)$ is NSE



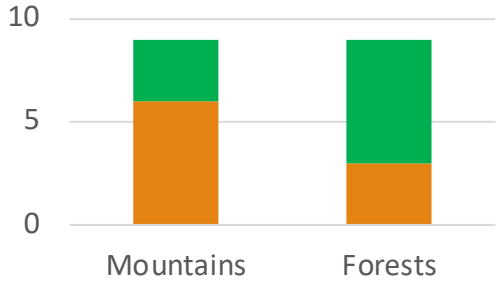
$x^1 = (1,0) \rightarrow v^1 = (6,3)$

Feasible!



$x^2 = (0,1) \rightarrow v^2 = (3,6)$

Feasible!

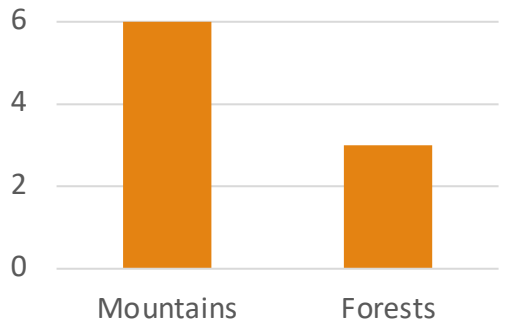


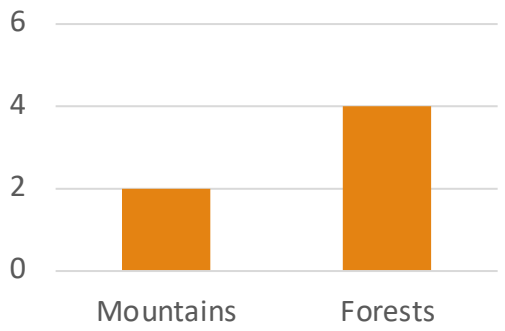
Attacker is best-responding!

Dwarves cannot **strictly** close the gap in coverage to induce attack on forests

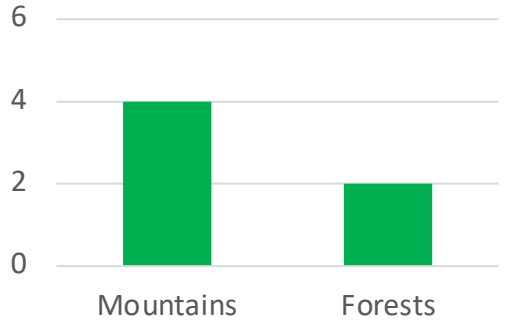
Incentive Compatible!

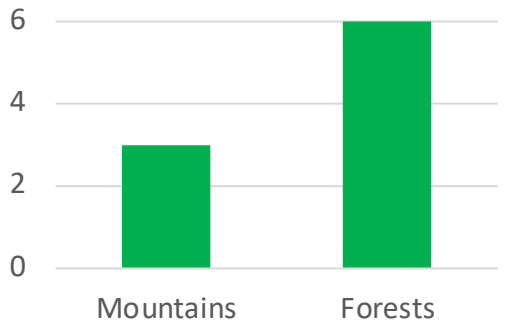
Example of non-NSE





Forests $\succ_1$ Mountains

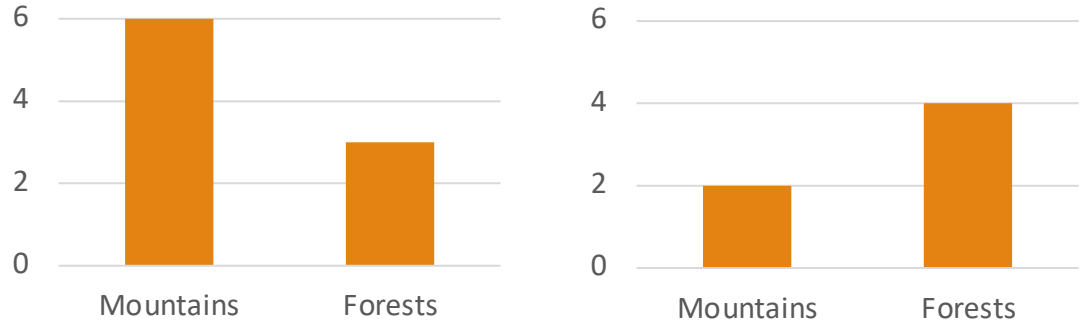




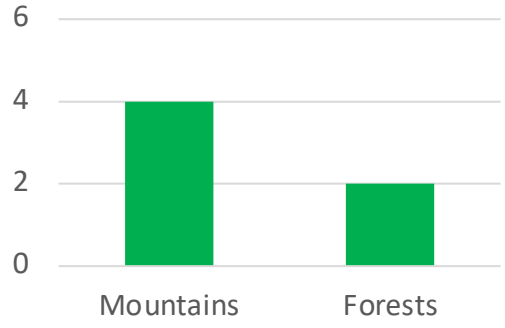
Mountains $\succ_2$ Forests

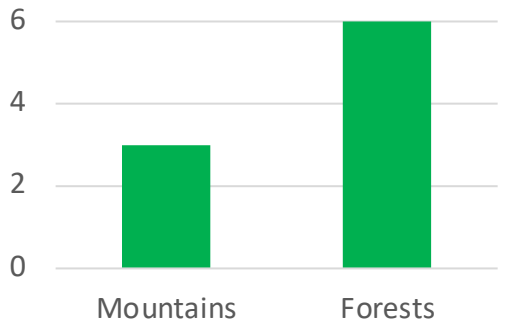
Example of non-NSE

Claim: $((4, 3.5), (3.5, 4), Mountain)$ is **not** NSE

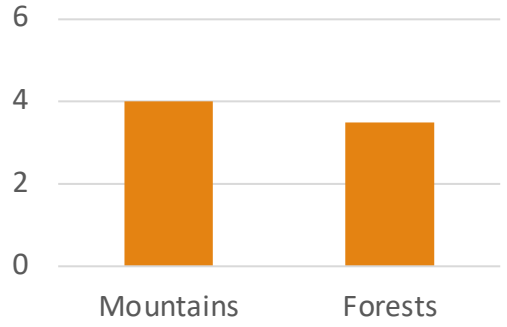


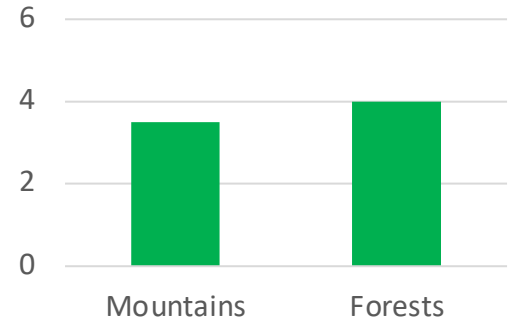
Forests $\succ_1$ Mountains





Mountains $\succ_2$ Forests



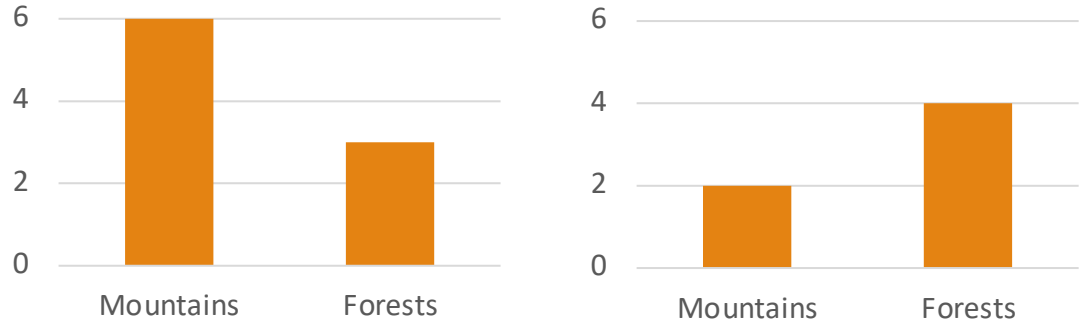


$x^1 = (\frac{1}{2}, \frac{1}{2}) \rightarrow v^1 = (4, 3.5)$

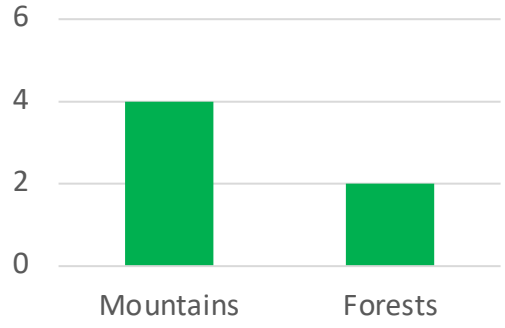
$x^2 = (\frac{1}{2}, \frac{1}{2}) \rightarrow v^2 = (3.5, 4)$

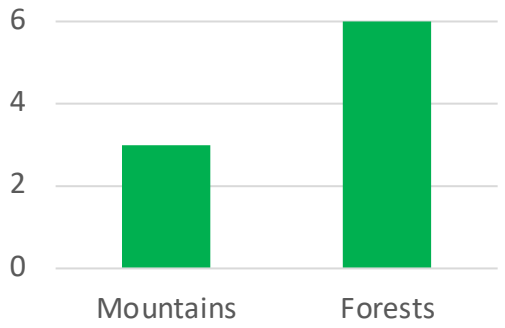
Example of non-NSE

Claim: $((4, 3.5), (3.5, 4), Mountain)$ is **not** NSE

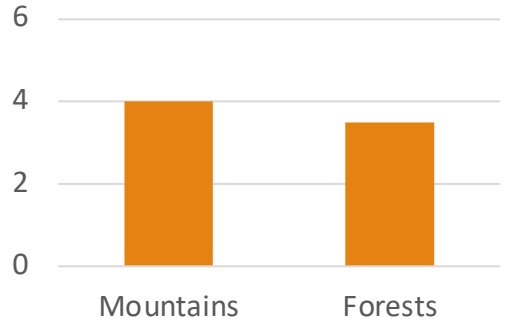


Forests $\succ_1$ Mountains



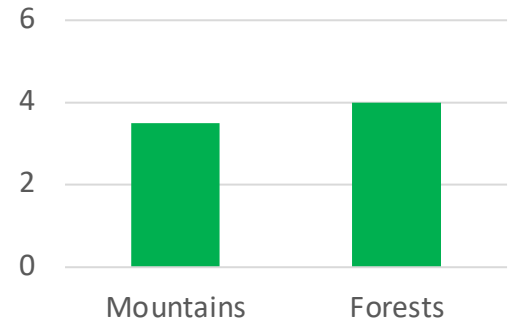


Mountains $\succ_2$ Forests



$x^1 = (\frac{1}{2}, \frac{1}{2}) \rightarrow v^1 = (4, 3.5)$

Feasible!

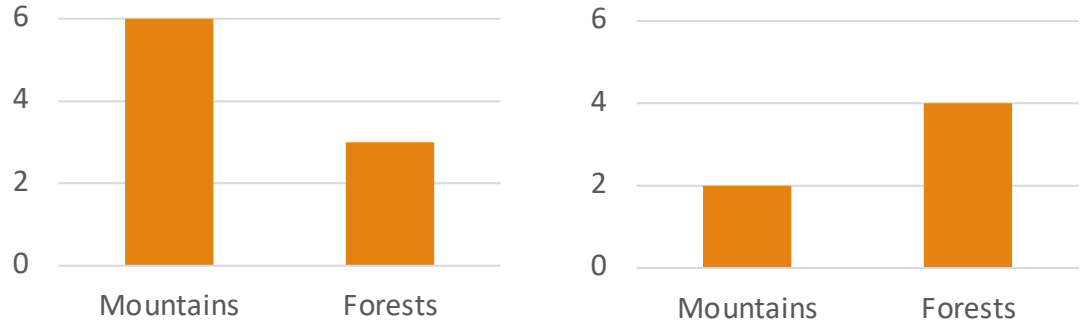


$x^2 = (\frac{1}{2}, \frac{1}{2}) \rightarrow v^2 = (3.5, 4)$

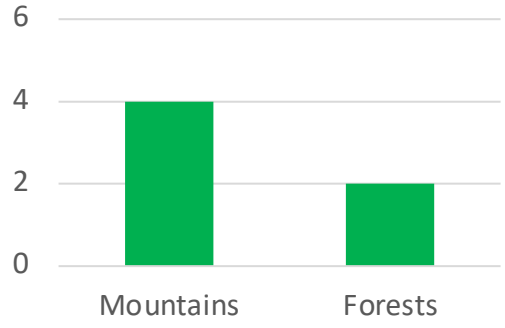
Feasible!

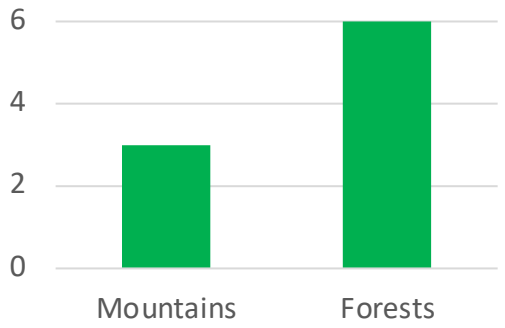
Example of non-NSE

Claim: $((4, 3.5), (3.5, 4), Mountain)$ is **not** NSE

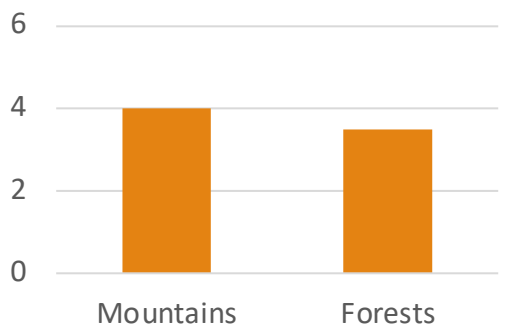


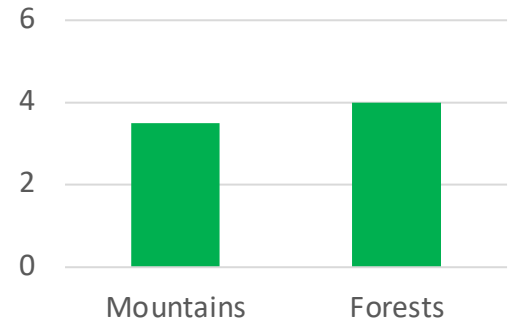
Forests $\succ_1$ Mountains





Mountains $\succ_2$ Forests



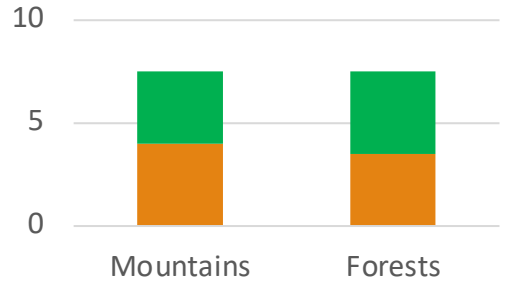


$x^1 = (\frac{1}{2}, \frac{1}{2}) \rightarrow v^1 = (4, 3.5)$

Feasible!

$x^2 = (\frac{1}{2}, \frac{1}{2}) \rightarrow v^2 = (3.5, 4)$

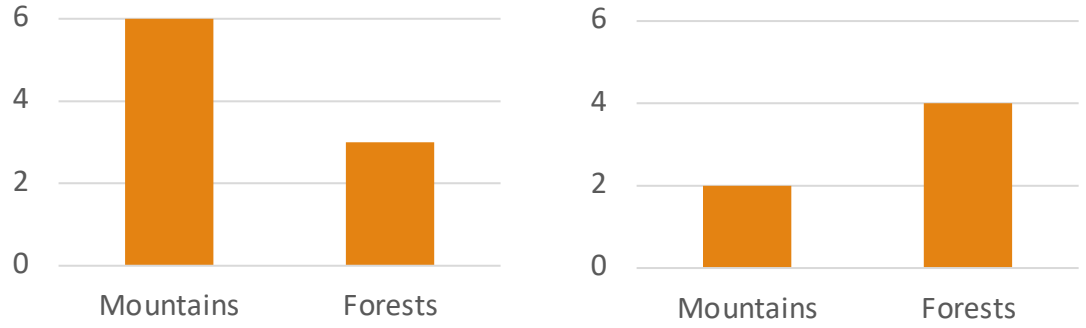
Feasible!



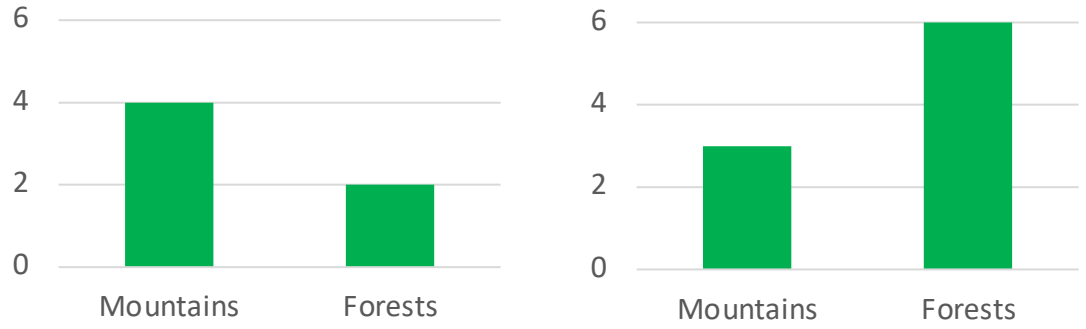
Attacker is best-responding!

Example of non-NSE

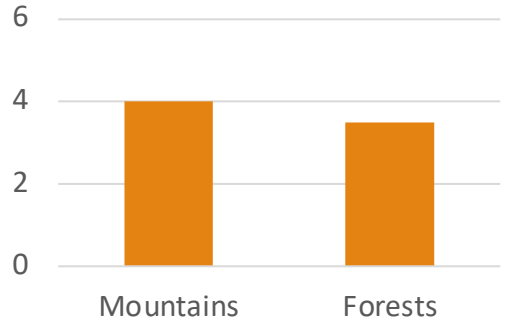
Claim: $((4, 3.5), (3.5, 4), Mountain)$ is **not** NSE



Forests $\succ_1$ Mountains

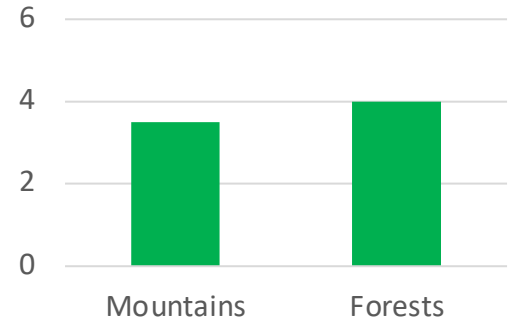


Mountains $\succ_2$ Forests



$x^1 = (\frac{1}{2}, \frac{1}{2}) \rightarrow v^1 = (4, 3.5)$

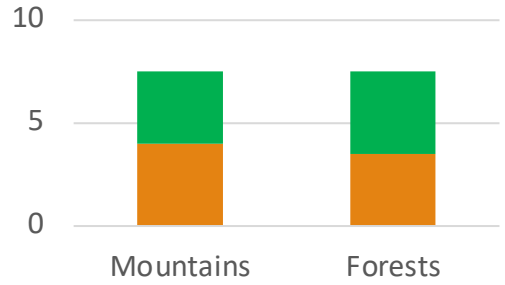
Feasible!



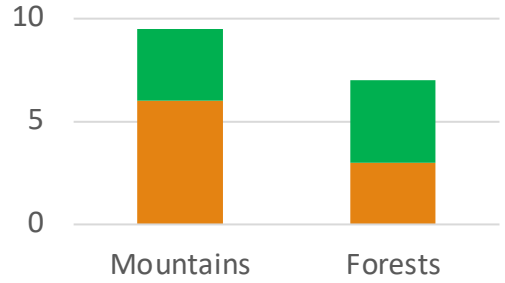
$x^2 = (\frac{1}{2}, \frac{1}{2}) \rightarrow v^2 = (3.5, 4)$

Feasible!

Dwarves can deviate to $\tilde{v}^1 = (6, 3) \rightarrow$ Forest is new target!



Attacker is best-responding!



Not Incentive Compatible!

When #defenders, #schedules, #targets are large, finding NSE can be tricky…

But verifying if a given $(v^1, \ldots, v^N, t^*)$ is a NSE can be done using linear programming

Contribution 1: NSE may not exist

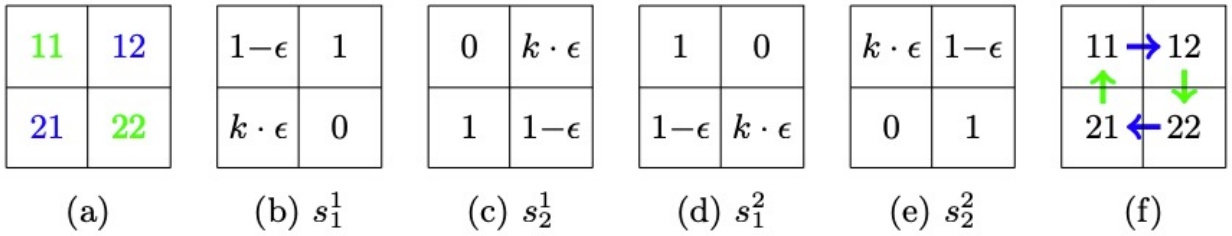


Fig. 2: Illustration of Example 1. (a) Target labels. Defender 1 prefers diagonal (green) targets attacked; defender 2 prefers off diagonal (blue) targets. (b-e) Defender schedules s_1^1, s_2^1, s_1^2 and s_2^2. (f) Cyclic behavior. Green/blue arrows indicate changes in targets induced by defender 1/2 deviating.

- Each cell is a target, assume $\epsilon = k = 0$
  - P1 prefers green targets attacked, P2 prefers blue targets attacked
  - P1 splits defenses across rows, P2 splits them across columns
- No possible NSE
  - If top left target attacked, P2 defends column 1, induces attack on top right
  - *Technically*, uniform strategy is an equilibrium
  - If $k = 100, \epsilon = 1e - 3$, then no equilibrium exists, not even approximate
- Problem: defenders are forced to protect targets that they'll rather not defend!

What makes finding NSE difficult?

What makes finding NSE difficult?

Water-filling methods hard to reason about

- Forced to defend some targets even if we would prefer not to
- E.g., dwarves forced to defend forests even if they prefer forests to be attacked
- Difficult to reason about ``target'' levels of coverage (unlike Gan et. al., 2022)

What makes finding NSE difficult?

Water-filling methods hard to reason about

- Forced to defend some targets even if we would prefer not to
- E.g., dwarves forced to defend forests even if they prefer forests to be attacked
- Difficult to reason about ``target'' levels of coverage (unlike Gan et. al., 2022)

Also difficult to reason about *deviations*

- Can deviate to *mixed strategies* → Hard to transform into LP (or even IP)

What makes finding NSE difficult?

Water-filling methods hard to reason about

- Forced to defend some targets even if we would prefer not to
- E.g., dwarves forced to defend forests even if they prefer forests to be attacked
- Difficult to reason about ``target'' levels of coverage (unlike Gan et. al., 2022)

Also difficult to reason about *deviations*

- Can deviate to *mixed strategies* → Hard to transform into LP (or even IP)

Feasible set of coverage can be somewhat rigid

$$V^i = \left\{ v \in R_+^T | \exists x^i \; s.t. \forall j \in [T] \; v(j) = \sum_z x^i(z) \cdot s_z^i(j) \right\}$$

What makes finding NSE difficult?

Water-filling methods hard to reason about

- Forced to defend some targets even if we would prefer not to
- E.g., dwarves forced to defend forests even if they prefer forests to be attacked
- Difficult to reason about ``target'' levels of coverage (unlike Gan et. al., 2022)

Also difficult to reason about *deviations*

- Can deviate to *mixed strategies* → Hard to transform into LP (or even IP)

Feasible set of coverage can be somewhat rigid

$$V^i = \left\{ v \in R_+^T | \exists x^i \; s.t. \; \forall j \in [T] \; v(j) = \sum_z x^i(z) \cdot s_z^i(j) \right\}$$

Directly applying fixed-point theorems quite difficult

- Graph of best-response set function is not necessarily closed

Subset of Schedule is a Schedule (SSAS)

Problem with previous example: defenders are providing coverage to targets which they'd rather **not** defend!

Our fix: let defenders provide strictly less coverage than allowed

- Deliberately "let their guard down" when patrolling a particular target that they would prefer attacked
- First proposed for computational reasons

Subset of Schedule is a Schedule (SSAS)

Problem with previous example: defenders are providing coverage to targets which they'd rather **not** defend!

Our fix: let defenders provide strictly less coverage than allowed

- Deliberately "let their guard down" when patrolling a particular target that they would prefer attacked
- First proposed for computational reasons

Original Feasible Coverage Set (clearance)

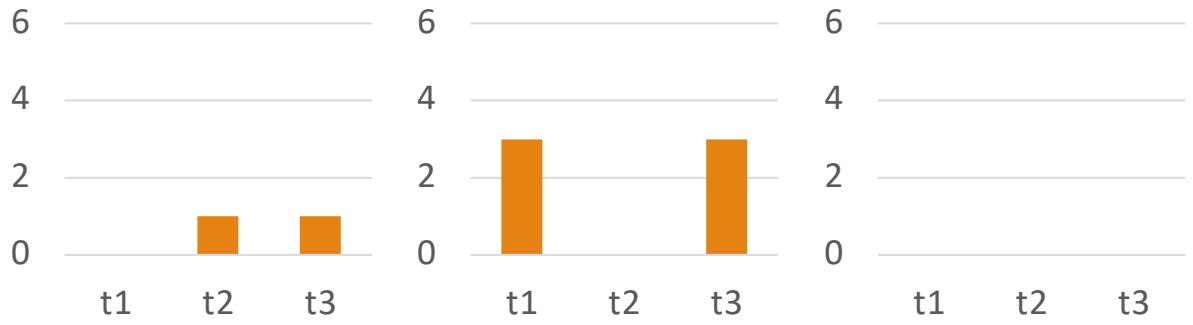
$$V^i = \left\{ v \in R_+^T | \exists x^i \; s.t. \forall j \in [T] \; v(j) = \sum_z x^i(z) \cdot s_z^i(j) \right\}$$

New Feasible Coverage Set (SSAS)

$$V^i = \left\{ v \in R_+^T | \exists x^i \; s.t. \forall j \in [T] \; v(j) \leq \sum_z x^i(z) \cdot s_z^i(j) \right\}$$

Example of our algorithm

$t_2 \succ_1 t_1 \succ_1 t_3$

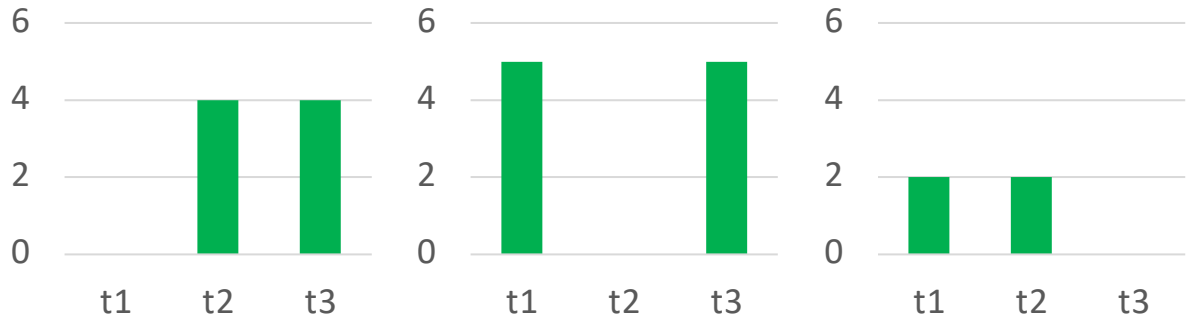


$s_1^1 = (0,1,1)$

$s_2^1 = (3,0,3)$

$s_3^1 = (0,0,0)$

$t_3 \succ_2 t_1 \succ_2 t_2$



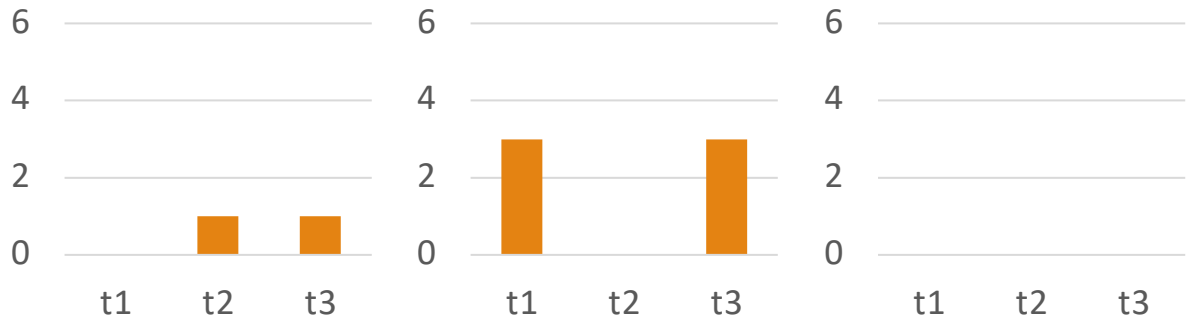
$s_1^2 = (0,4,4)$

$s_2^2 = (5,0,5)$

$s_3^2 = (2,2,0)$

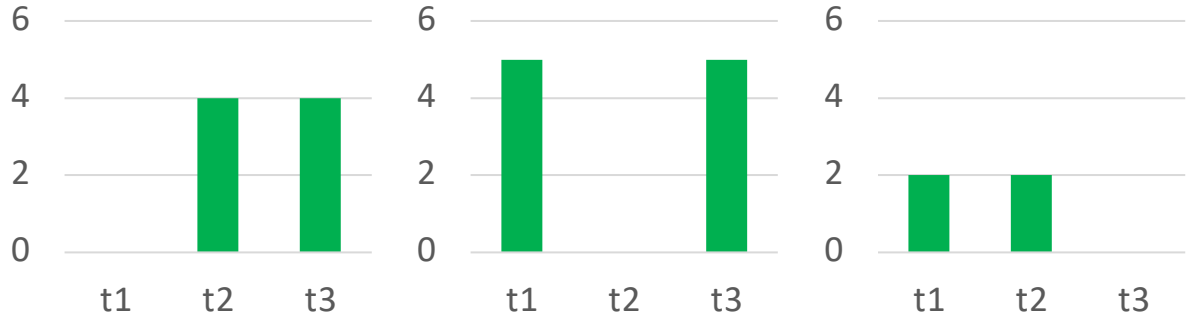
Example of our algorithm

$t_2 \succ_1 t_1 \succ_1 t_3$



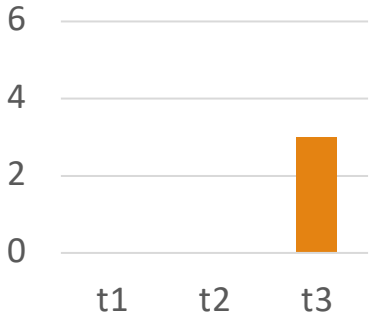
$s_1^1 = (0,1,1)$ $s_2^1 = (3,0,3)$ $s_3^1 = (0,0,0)$

$t_3 \succ_2 t_1 \succ_2 t_2$



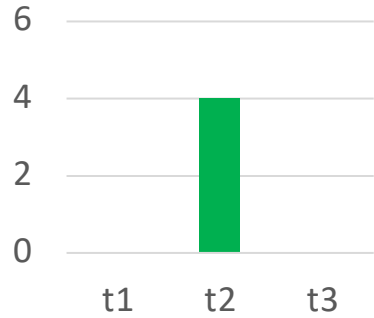
$s_1^2 = (0,4,4)$ $s_2^2 = (5,0,5)$ $s_3^2 = (2,2,0)$

Claim: $((0,0,3),(0,4,0),t_1)$ is NSE



$x^1 = (0,1,0)$ and $v^1 \leq (3,0,3)$

Feasible!

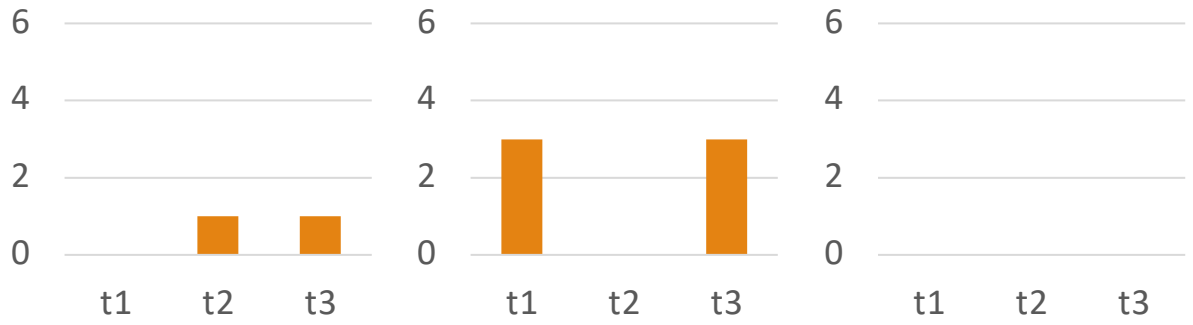


$x^2 = (1,0,0)$ and $v^2 \leq (0,4,4)$

Feasible!

Example of our algorithm

$t_2 \succ_1 t_1 \succ_1 t_3$

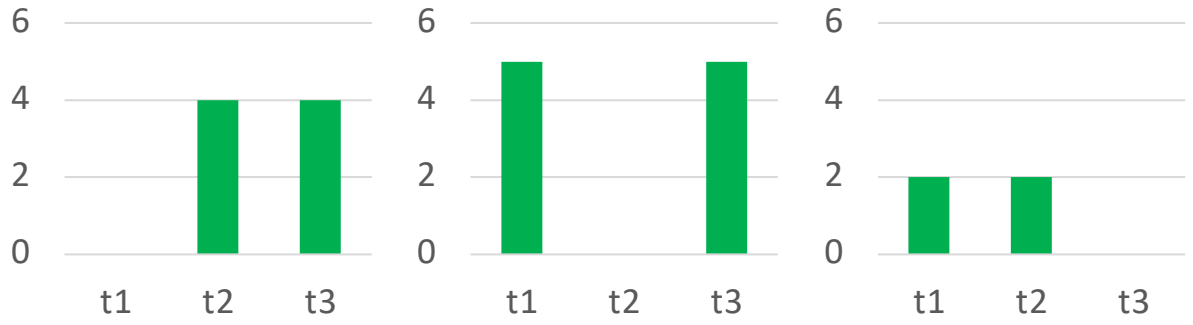


$s_1^1 = (0,1,1)$

$s_2^1 = (3,0,3)$

$s_3^1 = (0,0,0)$

$t_3 \succ_2 t_1 \succ_2 t_2$

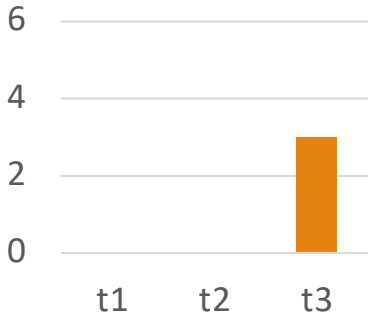


$s_1^2 = (0,4,4)$

$s_2^2 = (5,0,5)$

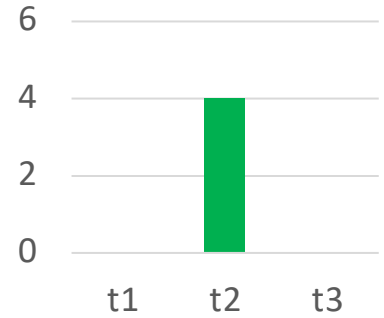
$s_3^2 = (2,2,0)$

Claim: $((0,0,3),(0,4,0),t_1)$ is NSE



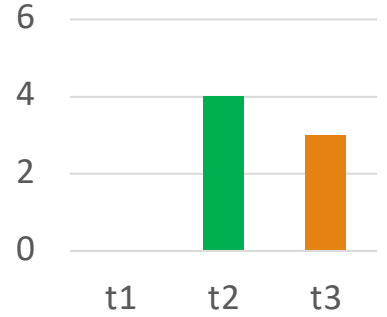
$x^1 = (0,1,0)$ and $v^1 \leq (3,0,3)$

Feasible!



$x^2 = (1,0,0)$ and $v^2 \leq (0,4,4)$

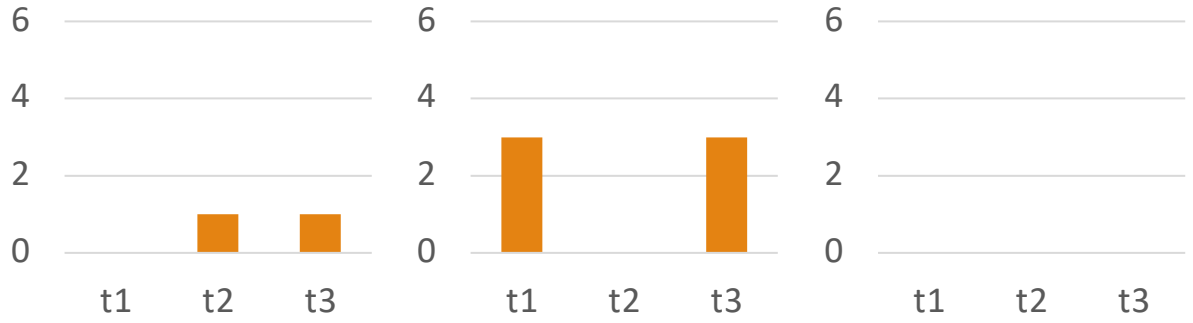
Feasible!



Attacker is best responding by attacking t_1

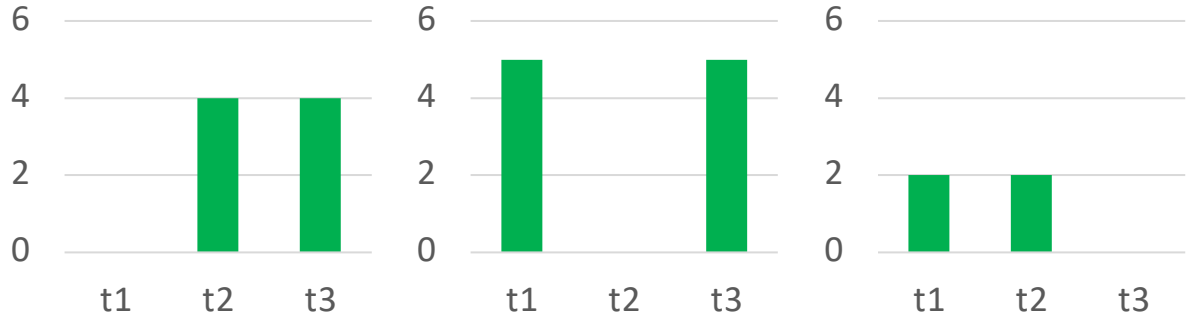
Example of our algorithm

$t_2 \succ_1 t_1 \succ_1 t_3$



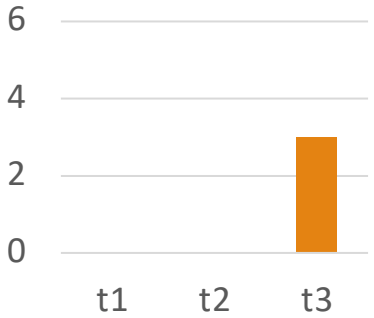
$s_1^1 = (0,1,1)$ $s_2^1 = (3,0,3)$ $s_3^1 = (0,0,0)$

$t_3 \succ_2 t_1 \succ_2 t_2$



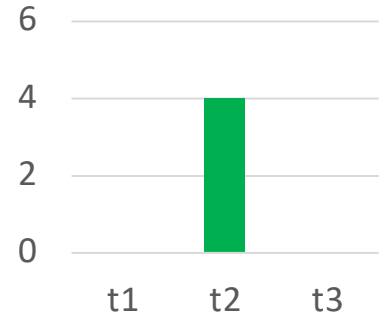
$s_1^2 = (0,4,4)$ $s_2^2 = (5,0,5)$ $s_3^2 = (2,2,0)$

Claim: $((0,0,3),(0,4,0),t_1)$ is NSE



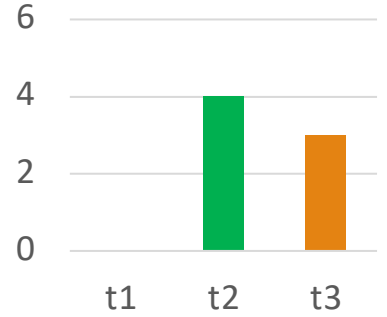
$x^1 = (0,1,0)$ and $v^1 \leq (3,0,3)$

Feasible!



$x^2 = (1,0,0)$ and $v^2 \leq (0,4,4)$

Feasible!



Attacker is best responding by attacking t_1

Suppose defender 2 tries to induce attack on t_3 instead→ it must increase coverage in **both** t_1 and t_2 to be > 3

Impossible, **Maximin coverage** of $\{t_1, t_2\}$ in V^2 is 20/9 < 3, obtained by playing s_1^2 with 5/9 and s_2^2 with 4/9.

Similar argument for defender 1: incentive compatible!

Contribution 2: Existence and Algorithm

Contribution 2: Existence and Algorithm

When #defenders = 2 & assuming SSAS, NSE **exists** and can be computed in **polynomial time**

Contribution 2: Existence and Algorithm

When #defenders = 2 & assuming SSAS, NSE **exists** and can be computed in **polynomial time**

Assume t is the target at equilibrium and defender 1 is trying to induce a new attack target $t' \succ_1 t$

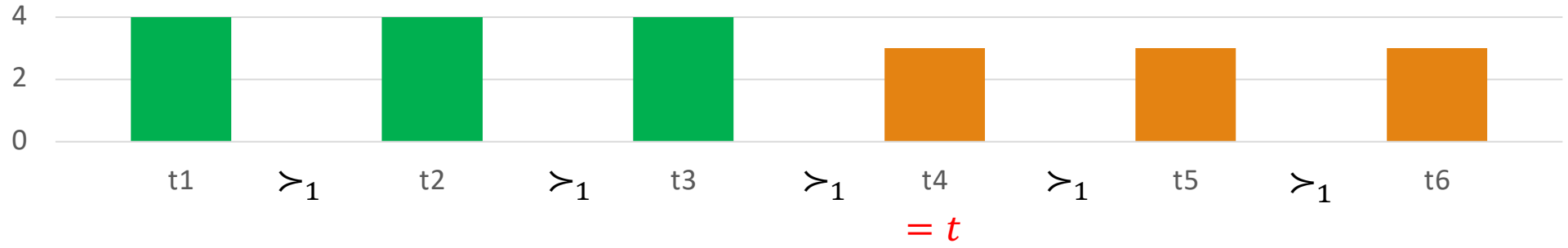
- Best way that defender 2 allocates coverage to prevent changes in target?
- Best way that defender 1 allocates coverage to induce a new attack target?

Contribution 2: Existence and Algorithm

When #defenders = 2 & assuming SSAS, NSE **exists** and can be computed in **polynomial time**

Assume t is the target at equilibrium and defender 1 is trying to induce a new attack target $t' \succ_1 t$

- Best way that defender 2 allocates coverage to prevent changes in target?
- Best way that defender 1 allocates coverage to induce a new attack target?

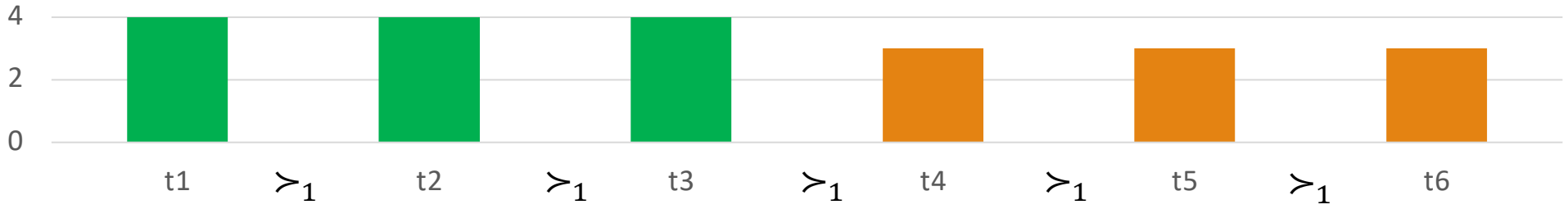


Contribution 2: Existence and Algorithm

When #defenders = 2 & assuming SSAS, NSE **exists** and can be computed in **polynomial time**

Assume t is the target at equilibrium and defender 1 is trying to induce a new attack target $t' \succ_1 t$

- Best way that defender 2 allocates coverage to prevent changes in target?
- Best way that defender 1 allocates coverage to induce a new attack target?



if MaximinCoverage of Player 2 $\geq$ MaximinCoverage of Player 1

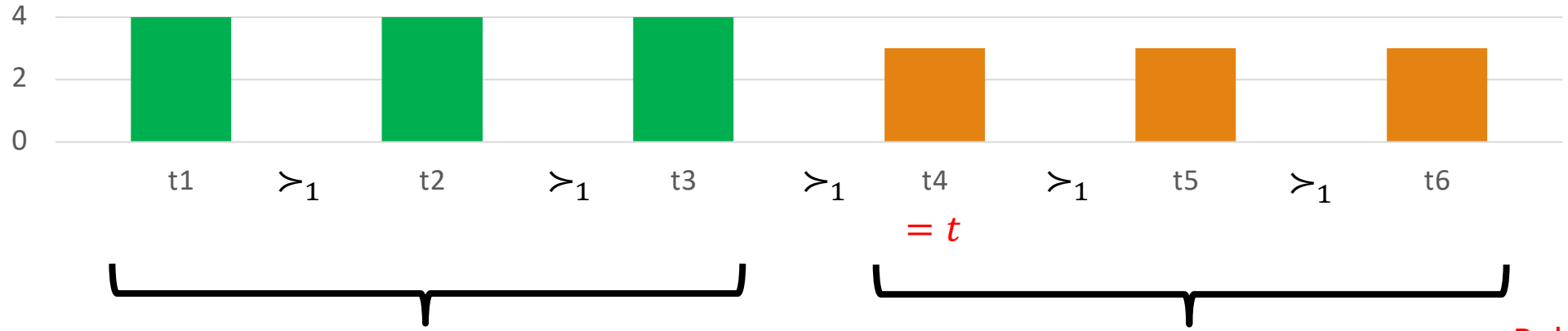
Robust to defender 1's deviations

Contribution 2: Existence and Algorithm

When #defenders = 2 & assuming SSAS, NSE **exists** and can be computed in **polynomial time**

Assume t is the target at equilibrium and defender 1 is trying to induce a new attack target $t' \succ_1 t$

- Best way that defender 2 allocates coverage to prevent changes in target?
- Best way that defender 1 allocates coverage to induce a new attack target?



if MaximinCoverage of Player 2 $\geq$ MaximinCoverage of Player 1

Robust to defender 1's deviations

if MaximinCoverage of Player 2 $<$ MaximinCoverage of Player 1

Defender 1 can deviate

Our Proposed Algorithm

Algorithm 1 NSE for 2 defenders

Input: n, T, V^1, V^2
Output: an NSE $(\mathbf{v}, t)$
for $t = 1$ **to** T **do**
  $h^1 \leftarrow \textsc{MaximinCov}(\mathcal{T}_t^{\succ 2}, V^1)$
  $g^1 \leftarrow \textsc{MaximinCov}(\mathcal{T}_t^{\preceq 2}, V^2)$
  $h^2 \leftarrow \textsc{MaximinCov}(\mathcal{T}_t^{\succ 1}, V^2)$
  $g^2 \leftarrow \textsc{MaximinCov}(\mathcal{T}_t^{\preceq 1}, V^1)$
  if $h^1 \geq g^1$ and $h^2 \geq g^2$ **then**
    for $j = 1$ **to** T **do**
      $v^1(j) \leftarrow h^1$ if $j \in \mathcal{T}_t^{\succ 2}$ else 0
      $v^2(j) \leftarrow h^2$ if $j \in \mathcal{T}_t^{\succ 1}$ else 0
    return (v^1, v^2, t)

Iterate over all possible equilibrium targets

Compute robustness towards deviations from defender 2

Compute robustness towards deviations from defender 1

If robust for **both** defenders, we are done!

Our Proposed Algorithm

Algorithm 1 NSE for 2 defenders

Input: n, T, V^1, V^2
Output: an NSE $(\mathbf{v}, t)$
for $t = 1$ **to** T **do**
  $h^1 \leftarrow \textsc{MaximinCov}(\mathcal{T}_t^{\succ 2}, V^1)$
  $g^1 \leftarrow \textsc{MaximinCov}(\mathcal{T}_t^{\preceq 2}, V^2)$
  $h^2 \leftarrow \textsc{MaximinCov}(\mathcal{T}_t^{\succ 1}, V^2)$
  $g^2 \leftarrow \textsc{MaximinCov}(\mathcal{T}_t^{\preceq 1}, V^1)$
  if $h^1 \geq g^1$ and $h^2 \geq g^2$ **then**
    for $j = 1$ **to** T **do**
      $v^1(j) \leftarrow h^1$ if $j \in \mathcal{T}_t^{\succ 2}$ else 0
      $v^2(j) \leftarrow h^2$ if $j \in \mathcal{T}_t^{\succ 1}$ else 0
    return (v^1, v^2, t)

Iterate over all possible equilibrium targets

Compute robustness towards deviations from defender 2

Compute robustness towards deviations from defender 1

If robust for **both** defenders, we are done!

Theorem: algorithm always returns *some* NSE (existence)

Our Proposed Algorithm

Algorithm 1 NSE for 2 defenders

Input: n, T, V^1, V^2
Output: an NSE $(\mathbf{v}, t)$
for $t = 1$ **to** T **do**
  $h^1 \leftarrow \textsc{MaximinCov}(\mathcal{T}_t^{\succ 2}, V^1)$
  $g^1 \leftarrow \textsc{MaximinCov}(\mathcal{T}_t^{\preceq 2}, V^2)$
  $h^2 \leftarrow \textsc{MaximinCov}(\mathcal{T}_t^{\succ 1}, V^2)$
  $g^2 \leftarrow \textsc{MaximinCov}(\mathcal{T}_t^{\preceq 1}, V^1)$
  if $h^1 \geq g^1$ and $h^2 \geq g^2$ **then**
    for $j = 1$ **to** T **do**
      $v^1(j) \leftarrow h^1$ if $j \in \mathcal{T}_t^{\succ 2}$ else 0
      $v^2(j) \leftarrow h^2$ if $j \in \mathcal{T}_t^{\succ 1}$ else 0
    return (v^1, v^2, t)

Poly Time!

Iterate over all possible equilibrium targets

Compute robustness towards deviations from defender 2

Compute robustness towards deviations from defender 1

If robust for **both** defenders, we are done!

Theorem: algorithm always returns *some* NSE (existence)

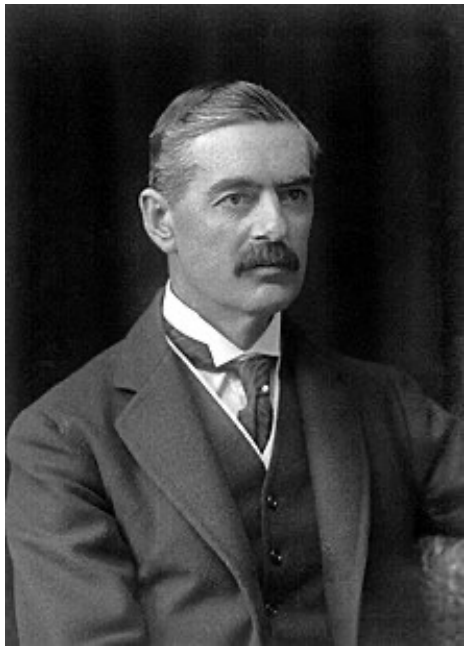
Takeaways & Modeling limitations

Are the defenders really in a team…?

Ben Franklin

* Let's ignore the second half of the quote…

The walking dead (Season 2)



Neville Chamberlain and appeasement