Lecture 13: Review, Summary, and Potential Directions

Admin

Due end next week
- Quiz
- Homework
- Project

Let me know if you need an extension

No office hours next week
- Must be conducted virtually in the evening
- I should be around on Friday
- Or simply email me

Teaching feedback (please submit!)

Today's lecture

This class only touched the tip of the surface

- Modeling vs theory/algorithms

Hopefully, people outside of Game Theory have learnt something

- Originally meant to be a pure research seminar

We covered a broad range of topics (we'll review them in a moment)

An overview of what research topics I'm interested in

Today's lecture

This class only touched the tip of the surface

- Modeling vs theory/algorithms

Hopefully, people outside of Game Theory have learnt something

- Originally meant to be a pure research seminar

We covered a broad range of topics (we'll review them in a moment)

An overview of what research topics I'm interested in

I'm **actively** looking for students (more later)

First: Any Questions For Quiz/Homework?

(Do on whiteboard)

Classic Material

Nash equilibrium in matrix games

- Definition, Best responses, existence via Brouwer/Kakutani
  - Refinements: Strong Nash equilibrium, Stable Nash equilibrium, Coalition Proof NE, Evolutionary Stable Strategies
  - Weakening: correlated equilibria, coarse CE, rationalizable strategy (players are BR-ing to some *beliefs* as to what opponents are doing)
- Limitations
  - PPAD-hard, inapproximable, hard to find →weak predictive power?
  - Need complete specification of game, payoffs, actions
  - Bounded rationality: Quantal response equilibria
- Algorithms
  - General-sum games
    - Vertex/Support enumeration, Nash as LCP, Lemke Howson (2 players) & path following, Integer programming (2 players) [Qn: what are some advantages/disadvantages?] Multiplayer general-sum game: homotopy methods: Govindan-Wilson, social welfare optimum equilibria, index of equilibria
  - Zero-sum games (next few lectures)
- Public libraries for game solving
  - gambit, NashPy
  - Most aren't super performant, but gets the job done

2-player zero-sum games

Minimax Theorem (important!)

- Conditions, Uses, Proof (many ways, e.g., Farkas Lemma, existence of no-regret methods)
- Extensions: Sions minimax theorem, Glicksberg theorem for continuous games (mathematically, not all have a value!), useful for market equilibria
- Equivalence to finding saddle-point

Importance of Exploitability (holds for general-sum games)

- Even if you are not into game solving, at least **evaluate** things properly
  - [Point out recent work in evaluation]
- Should evaluate based on worst case performance, BR of opponent
  - What makes a strong player? Beating noobs all the time or beating everyone by a small margin?
  - Non-transitivity in performance, A beats B, B beats C, C beats A
  - Example: Go players beat AlphaGo by exploiting oversights in AI (executed manually)
- In practice, difficult to evaluate exploitability for large games, but still an objective
- Alternatives to ELO scores that are based on equilibria (many others exist)
  - AlphaRank, Nash

Strategy spaces beyond simplexes: graphs, other constraints

Algorithms for solving 2p0s games

Linear programming + dualization

- Change min-max problem to min-min problem
  - Works on domains beyond the simplex too!

Decoupled methods

- Approximate equilibria, saddle point residual
- Fictitious play
- No-regret learning (outside of games)
  - Putting 2 no-regret agents against each other →converges in average iterate to Nash
    - Average iterate convergence vs Last iterate convergence. Best iterate convergence
    - Usually much faster (and space efficient) than LPs
  - Regret minimizers: RM, RM+, Hedge (+others, even projected gradient descent works)
    - Bandits (not full feedback, e.g., Exp3)
    - Relationship to optimization, in particular online mirror descent, e.g., Hedge = OMD with entropy DGF
    - Relationship to Blackwell approachability
    - Can be used to prove minimax theorem

Extensive Form Games

Definition, Game Trees

- Perfect information games, backward induction, subgame perfect nash equilibria, pure strategy equilibria, credible vs non-credible threats
- Imperfect information games
  - Information sets
    - Generalizes matrix games, Lying/bluffing/playing randomly
    - More “nice” structures: perfect recall, no-absent mindedness, timeability
  - Conversion to normal form strategies
  - Behavioral strategies
  - Sequence form (perfect recall), treeplexes

Algorithms

- General-sum: 2 players, Lemke’s algorithm, LCPs, integer programming
- Zero-sum games
  - Online learning, CFR, Scaled Extensions
- Meta-game algorithms: double oracle, PSRO

Other equilibria

Mediated equilibra

- Correlated equilibria, Coarse correlated equilibria
- Strong connections to no-regret learning, external regret vs. internal regret
- Poly-time in matrix games (LP or self-play), only very recently poly-time for EFGs (was open for a long time!)
- EFCE/EFCCE, Phi-regret, hindsight rationality

Stackelberg leadership

- Bi-level optimization: Usually super general and hard, but in certain cases can be solved efficiently using multiple LP method, poly-time algorithms, in EFG is NP-hard (Conitzer & Sandholm) apart from 1-2 special cases (perfect info without chance)

Some applications of both

- Security, Battleship, Social welfare optimal strategies

Some other “rare” equilibria, e.g., Nash Stackelberg equilibria

Markov Games

Generalization of MDPs

- POMDPs would be POSGs (very hard to scale in practice)

Difference with EFGs

- Trajectories with cycles exist

Centralized method

- Minimax-Q learning (very, very old)

Decentralized methods

- V-learning (CE, CCE, NE for 2p0s)

Deep learning-based methods

- MADDPG + many, many others
- Check out Stefano Albrecht’s textbook for a reasonable coalition

Other topics

Learning Dynamics and Games

- Hedge, ”energy” conservation, Euler discretization, difference between continuous dynamics/EGT and discrete variants
- Last iterate convergence, RVU bounds

Subgame solving

- Value function at a node level does **not** exist in IIEFG
- Need value ”functional” sometimes, value depends on what opponent strategy chooses. Key difference between multiplayer and single player, e.g., POMDP has belief state space and belief-values
- Extend method in chess (roughly) to IIEFG
  - What makes a good value in practice? E.g., darkchess → use perfect information value?
- Safe subgame solving, Max-margin solving (other methods: ReBel, reach subgame solving, DeepStack), compatibility with deep learning based methods
- Extension to general-sum games, set of achievable payoffs

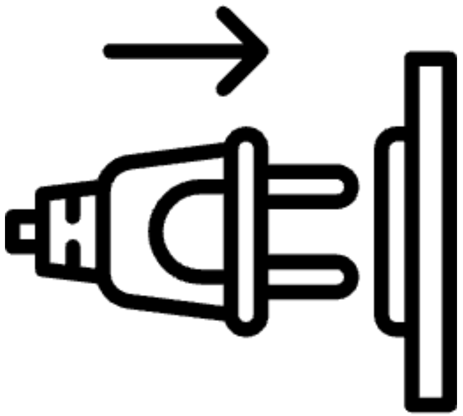
My interests and thoughts

What I’m interested in

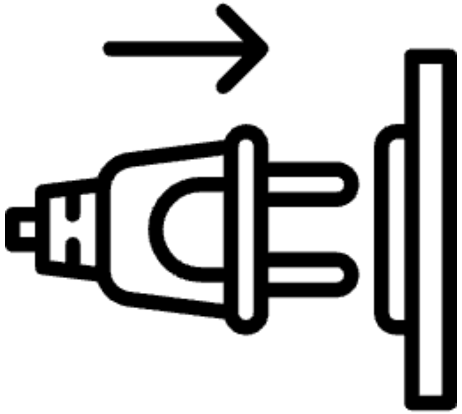
… because everyone has their own agenda

"So far the field has struggled to take techniques like this *out of the laboratory and game environments and into the real world…"*

-Gary Marcus on Alphastar, 2019



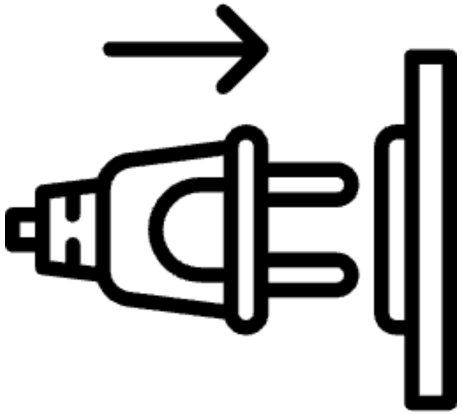
Game Theory in the real-world



Game Theory in the real-world

Traditional

More modern

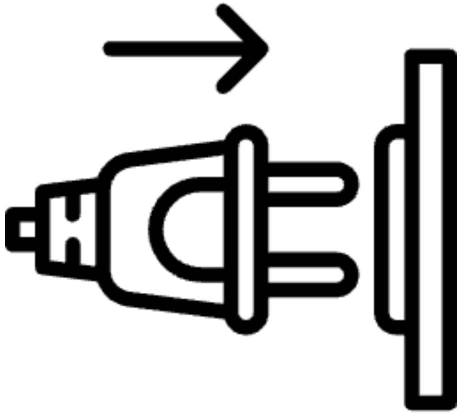


Game Theory in the real-world

Traditional

More modern

1. Algorithms &
Structure

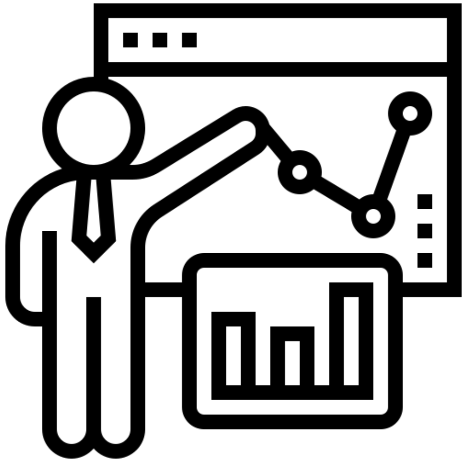


Game Theory in the real-world

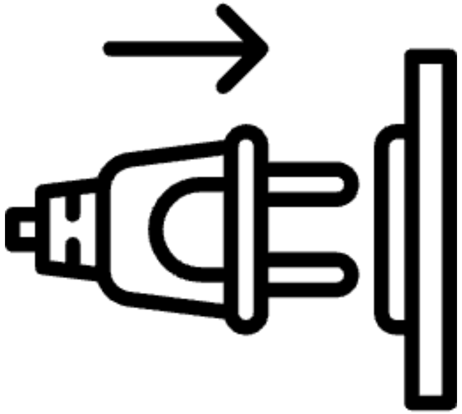
Traditional

More modern

1. Algorithms & Structure



2. Interpretability & sensemaking

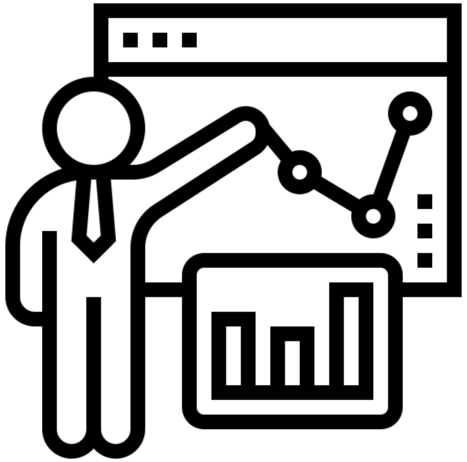


Game Theory in the real-world

Traditional

More modern

1. Algorithms & Structure



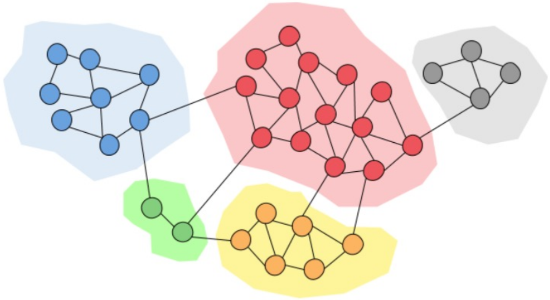
2. Interpretability & sensemaking

3. Challenging assumptions

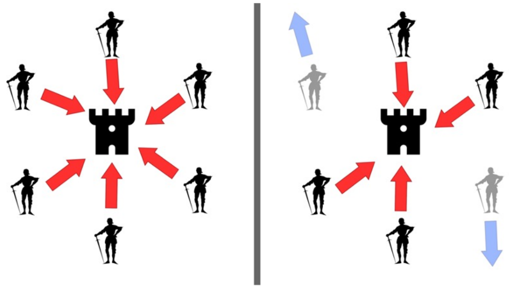
Direction 1: Algorithms & Structure

Real world problems are tough (large) but have lots of structure!

Ridesharing



Social media analysis



Distributed systems

Logistics

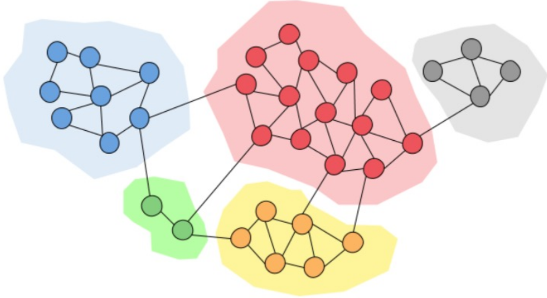
Cybersecurity

Politics

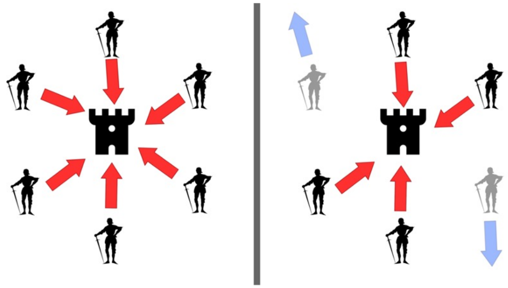
Direction 1: Algorithms & Structure

Real world problems are tough (large) but have lots of structure!

Ridesharing



Social media analysis



Distributed systems

Logistics

Cybersecurity

Politics

- Multiplayer, team games
- General-sum
- Imperfect information
- Jointly continuous & discrete
- Huge, combinatorial structure

How do we **model** real-world scenarios?

- Applications
  - Logistics and Patrolling
  - Cyber and network security
- Equilibrium concepts
  - Multi-defender security games
  - Extensive-form correlated equilibria
  - Sparse equilibrium

How do we **solve** these games we have defined?

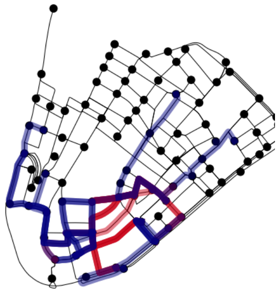
- Machine learning
  - Function approximation for general-sum game solving
- Optimization
  - Differentiable game solvers (in EFGs)
  - Variants of double oracle/PSRO/iterative strategy generation
  - Variants of self-play + regret minimization
  - Subgame solving

How do we **model** real-world scenarios?

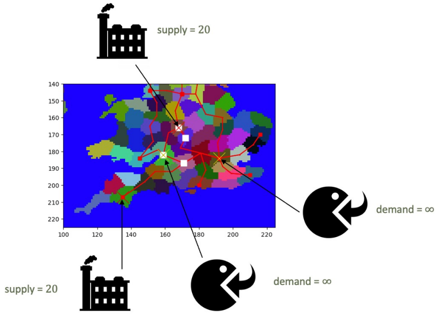
- Applications
  - Logistics and Patrolling
  - Cyber and network security
- Equilibrium concepts
  - Multi-defender security games
  - Extensive-form correlated equilibria
  - Sparse equilibrium

How do we **solve** these games we have defined?

- Machine learning
  - Function approximation for general-sum game solving
- Optimization
  - Differentiable game solvers (in EFGs)
  - Variants of double oracle/PSRO/iterative strategy generation
  - Variants of self-play + regret minimization
  - Subgame solving



https://arxiv.org/abs/2405.03070



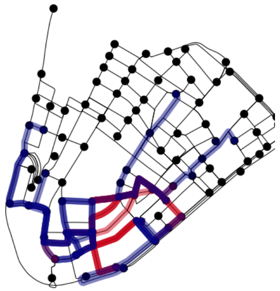
https://arxiv.org/abs/2408.13057

How do we **model** real-world scenarios?

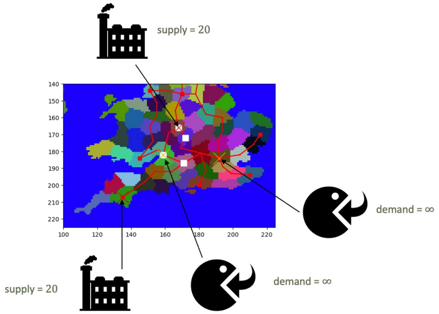
- Applications
  - Logistics and Patrolling
  - Cyber and network security
- Equilibrium concepts
  - Multi-defender security games
  - Extensive-form correlated equilibria
  - Sparse equilibrium

How do we **solve** these games we have defined?

- Machine learning
  - Function approximation for general-sum game solving
- Optimization
  - Differentiable game solvers (in EFGs)
  - Variants of double oracle/PSRO/iterative strategy generation
  - Variants of self-play + regret minimization
  - Subgame solving



https://arxiv.org/abs/2405.03070



https://arxiv.org/abs/2408.13057

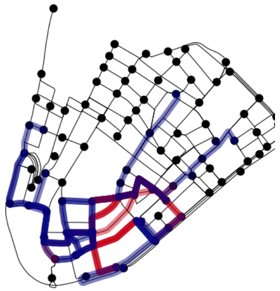
https://arxiv.org/abs/2311.16392

How do we **model** real-world scenarios?

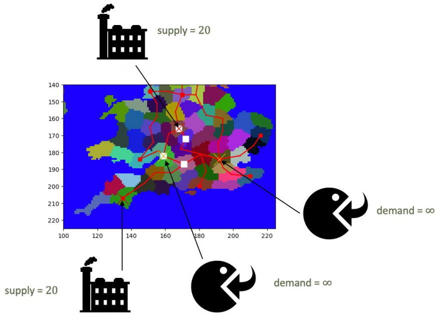
- Applications
  - Logistics and Patrolling
  - Cyber and network security
- Equilibrium concepts
  - Multi-defender security games
  - Extensive-form correlated equilibria
  - Sparse equilibrium

How do we **solve** these games we have defined?

- Machine learning
  - Function approximation for general-sum game solving
- Optimization
  - Differentiable game solvers (in EFGs)
  - Variants of double oracle/PSRO/iterative strategy generation
  - Variants of self-play + regret minimization
  - Subgame solving



https://arxiv.org/abs/2405.03070



https://arxiv.org/abs/2408.13057

https://arxiv.org/abs/2311.16392

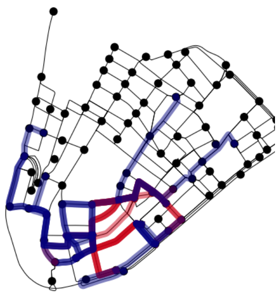
https://arxiv.org/abs/1905.12564
https://arxiv.org/abs/1910.12450

How do we **model** real-world scenarios?

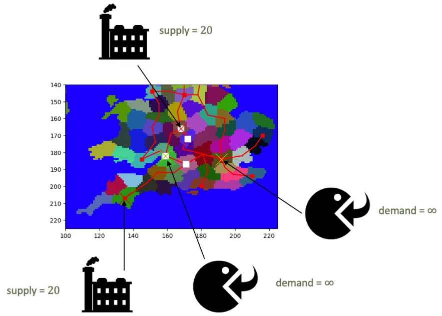
- Applications
  - Logistics and Patrolling
  - Cyber and network security
- Equilibrium concepts
  - Multi-defender security games
  - Extensive-form correlated equilibria
  - Sparse equilibrium

How do we **solve** these games we have defined?

- Machine learning
  - Function approximation for general-sum game solving
- Optimization
  - Differentiable game solvers (in EFGs)
  - Variants of double oracle/PSRO/iterative strategy generation
  - Variants of self-play + regret minimization
  - Subgame solving



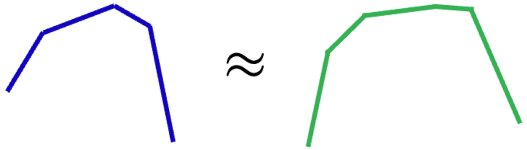
https://arxiv.org/abs/2405.03070



https://arxiv.org/abs/2408.13057

https://arxiv.org/abs/2311.16392

https://arxiv.org/abs/1905.12564
https://arxiv.org/abs/1910.12450



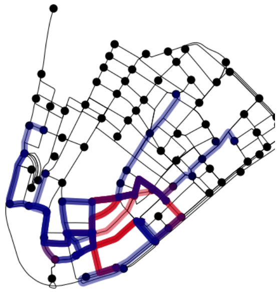
https://arxiv.org/abs/2212.14431

How do we **model** real-world scenarios?

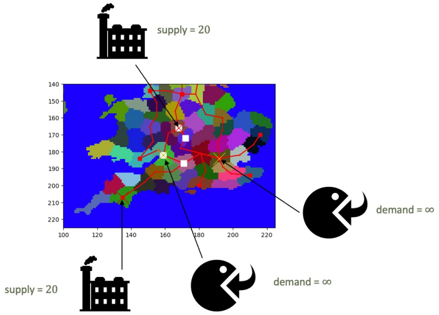
- Applications
  - Logistics and Patrolling
  - Cyber and network security
- Equilibrium concepts
  - Multi-defender security games
  - Extensive-form correlated equilibria
  - Sparse equilibrium

How do we **solve** these games we have defined?

- Machine learning
  - Function approximation for general-sum game solving
- Optimization
  - Differentiable game solvers (in EFGs)
  - Variants of double oracle/PSRO/iterative strategy generation
  - Variants of self-play + regret minimization
  - Subgame solving



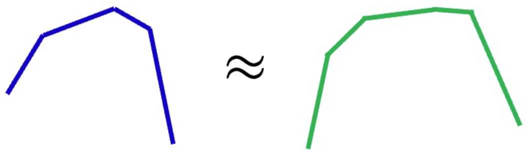
https://arxiv.org/abs/2405.03070



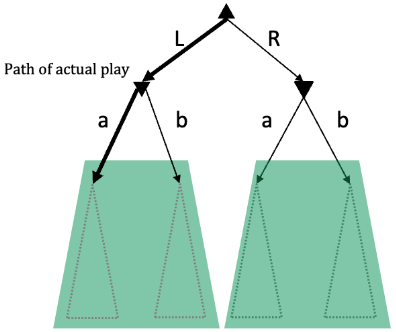
https://arxiv.org/abs/2408.13057

https://arxiv.org/abs/2311.16392

https://arxiv.org/abs/1905.12564
https://arxiv.org/abs/1910.12450



https://arxiv.org/abs/2212.14431

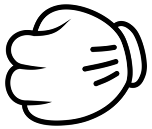
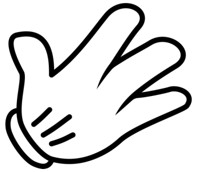
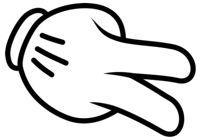
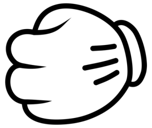
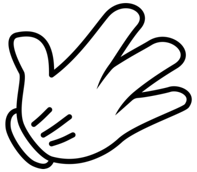
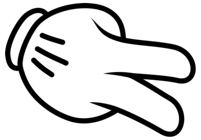


https://arxiv.org/abs/2102.01775

https://arxiv.org/abs/2212.14317

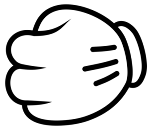
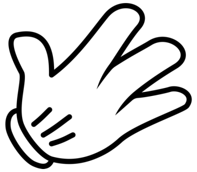
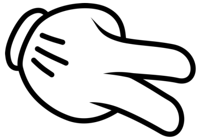
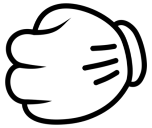
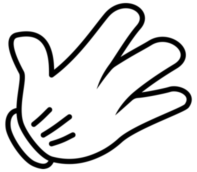
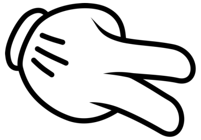
Inverse Game Theory

Learning player utilities from their past actions played in games

			
	0	$-b_1(x)$	$b_2(x)$
	$b_1(x)$	0	$-b_3(x)$
	$-b_2(x)$	$b_3(x)$	0

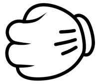
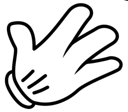
Inverse Game Theory

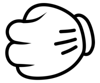
Learning player utilities from their past actions played in games

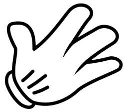
			
	0	$-b_1(x)$	$b_2(x)$
	$b_1(x)$	0	$-b_3(x)$
	$-b_2(x)$	$b_3(x)$	0

Learn

i.i.d samples from equilibrium strategies

$a^{(1)} =($ , $)$

$a^{(2)} =($ , $)$

$a^{(3)} =($ , $)$

...

Side Information

$$x^{(1)} = [0.1, 0.5]$$

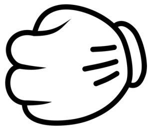
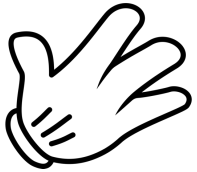
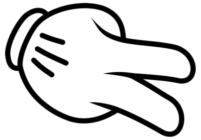
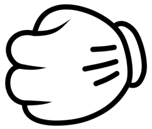
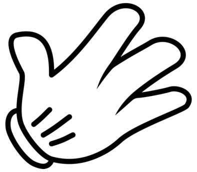
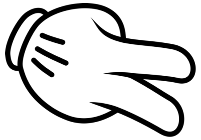
$$x^{(2)} = [0.3, 0.7]$$

$$x^{(3)} = [0.9\ 0.2]$$

...

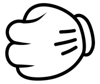
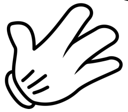
Inverse Game Theory

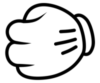
Learning player utilities from their past actions played in games

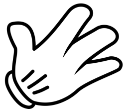
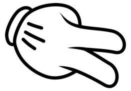
			
	0	$-b_1(x)$	$b_2(x)$
	$b_1(x)$	0	$-b_3(x)$
	$-b_2(x)$	$b_3(x)$	0

Learn

i.i.d samples from equilibrium strategies

$a^{(1)}=($  ,  $)$

$a^{(2)}=($  ,  $)$

$a^{(3)}=($  ,  $)$

…

Side Information

$$x^{(1)}=[0.1, 0.5]$$

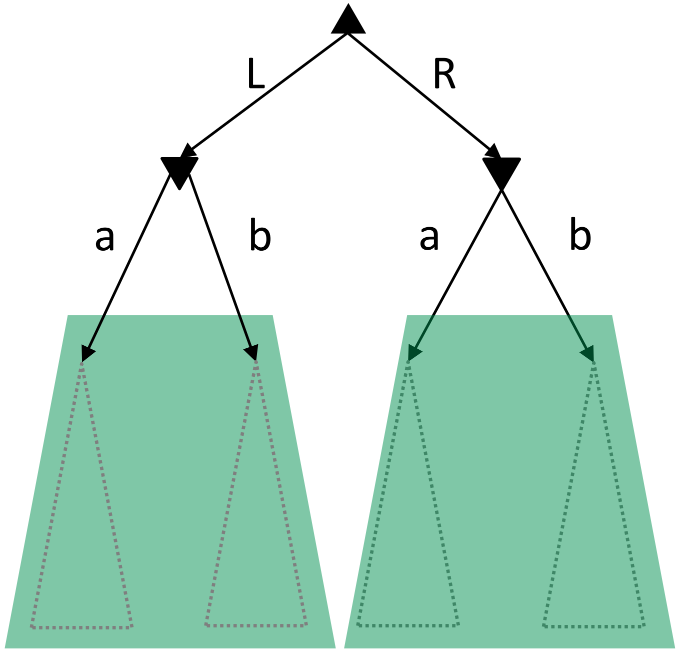
$$x^{(2)}=[0.3, 0.7]$$

$$x^{(3)}=[0.9\ 0.2]$$

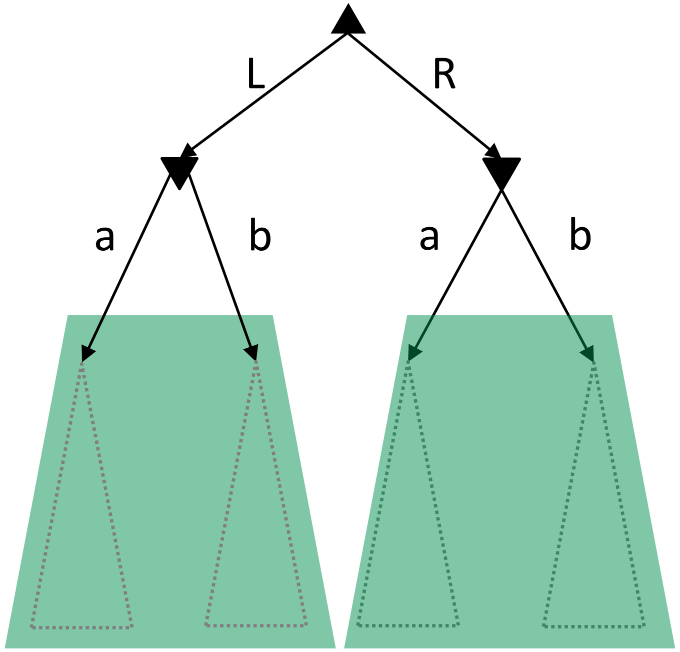
…

- Utilities are *functions* of side information x
  - E.g., utilities depend on age, demographic of players

Subgame Resolving



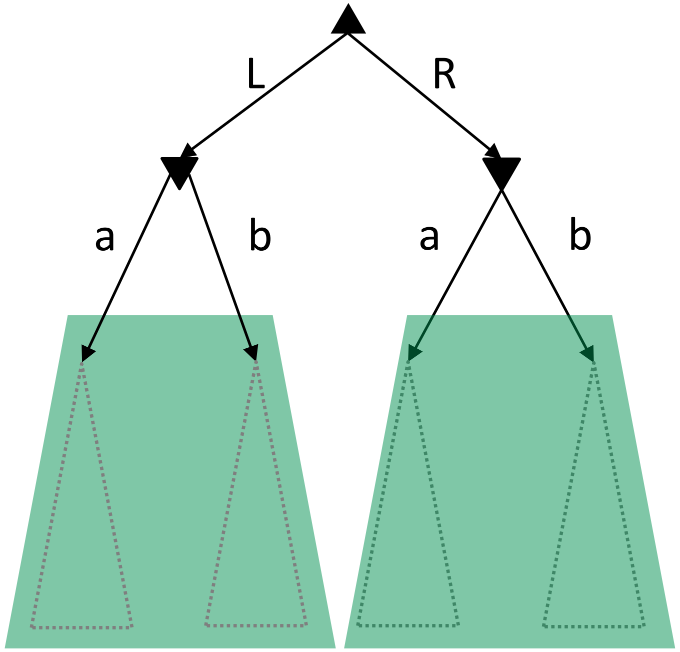
Subgame Resolving



Consider Chess

- Resolving is only initiated from states faced in **actual play**
- Avoid search from states which we've never reached

Subgame Resolving



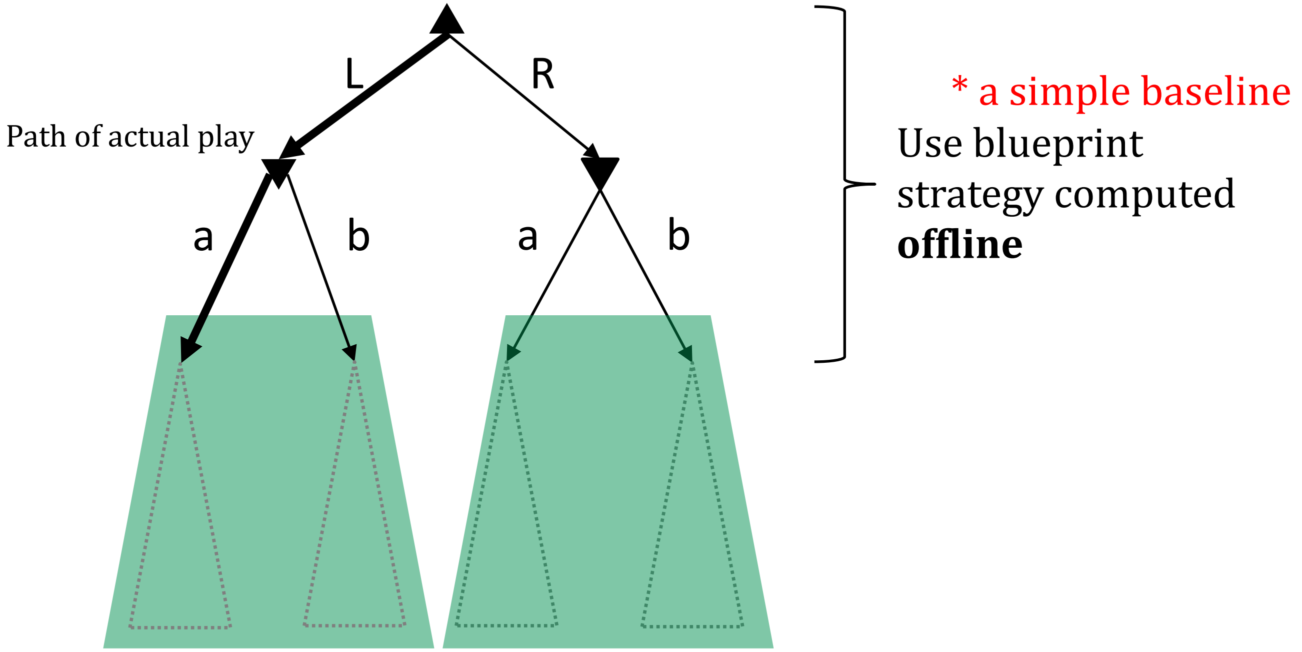
Consider Chess

- Resolving is only initiated from states faced in **actual play**
- Avoid search from states which we've never reached

Extensions to zero-sum imperfect information games

- Crucial component of top poker bots (Deepstack, Libratus)
- Us: Can be extended to various general-sum games safely!

Subgame Resolving



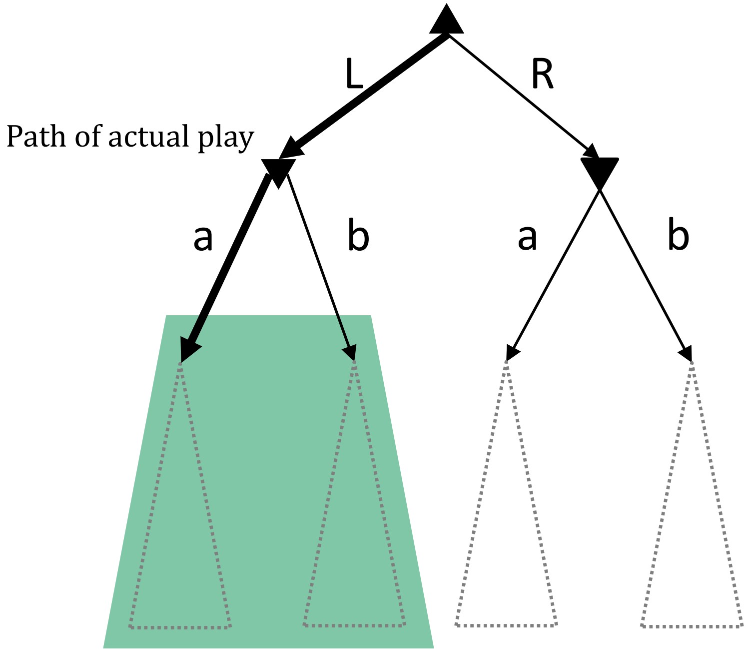
Consider Chess

- Resolving is only initiated from states faced in **actual play**
- Avoid search from states which we've never reached

Extensions to zero-sum imperfect information games

- Crucial component of top poker bots (Deepstack, Libratus)
- Us: Can be extended to various general-sum games safely!

Subgame Resolving



* a simple baseline

Use blueprint strategy computed **offline**

Compute refined strategy **online** *only for subgame entered*

Consider Chess

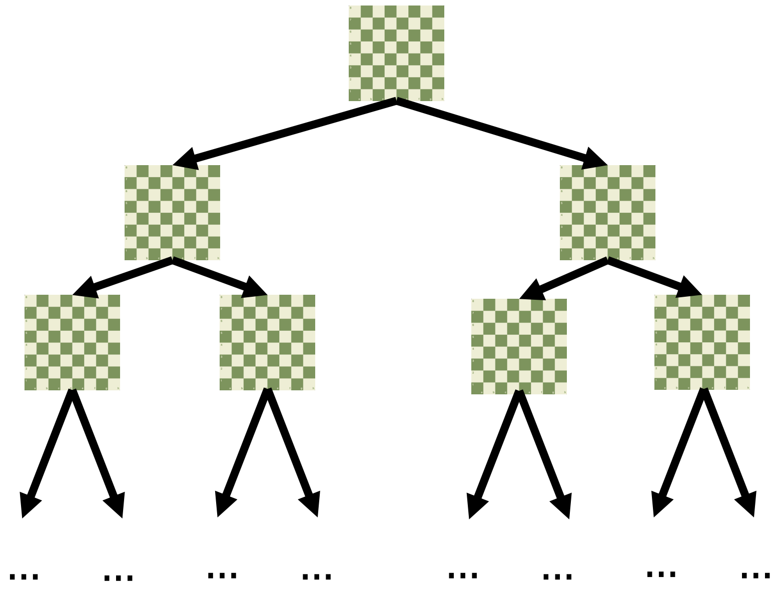
- Resolving is only initiated from states faced in **actual play**
- Avoid search from states which we've never reached

Extensions to zero-sum imperfect information games

- Crucial component of top poker bots (Deepstack, Libratus)
- Us: Can be extended to various general-sum games safely!

Resolving will do at least as well as the blueprint!

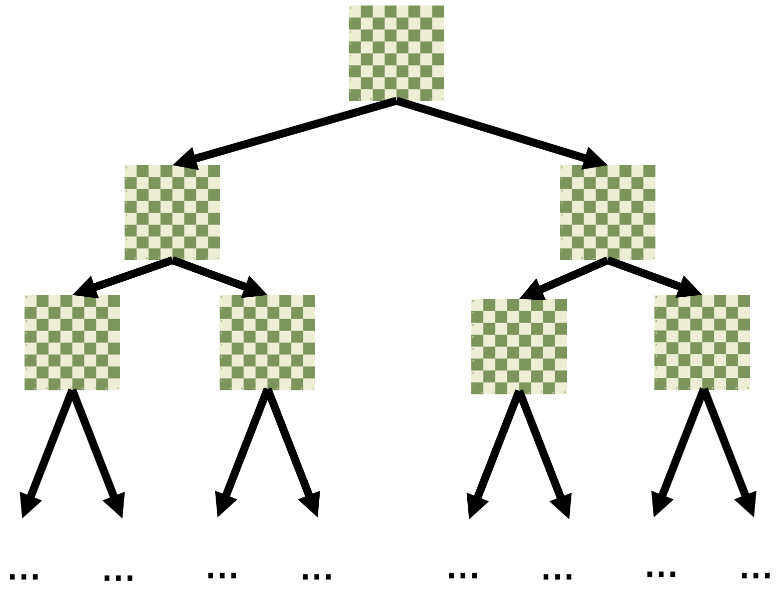
Game Solving with Function Approximation



Game Solving with Function Approximation

Consider zero-sum games

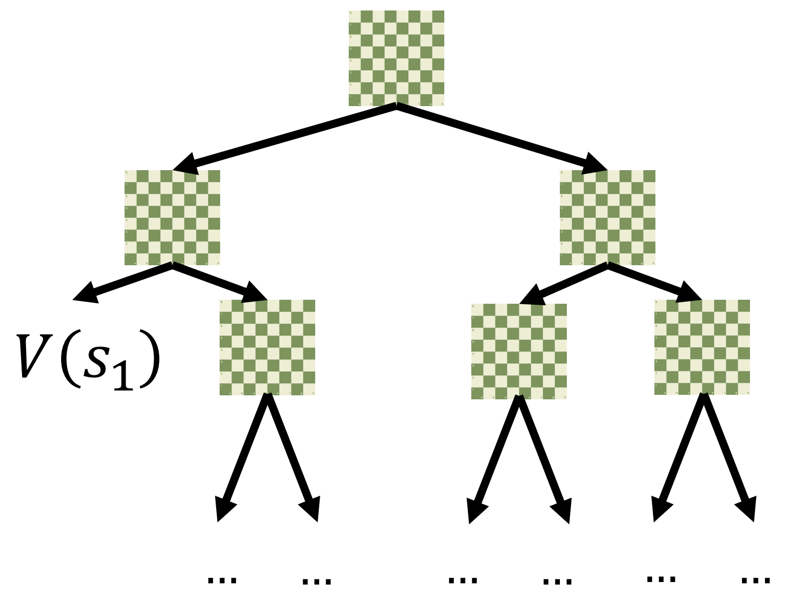
- Game tree too large to traverse explicitly
- **Value Function** $V(s)$ approximates how "good or bad" each state is



Game Solving with Function Approximation

Consider zero-sum games

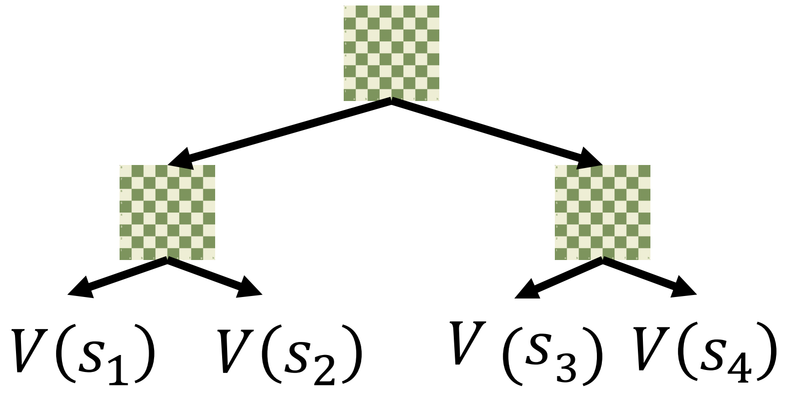
- Game tree too large to traverse explicitly
- **Value Function** $V(s)$ approximates how "good or bad" each state is



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



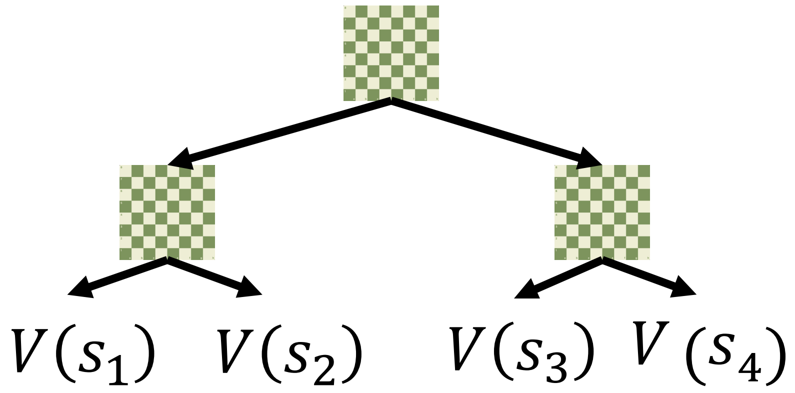
Game Solving with Function Approximation

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

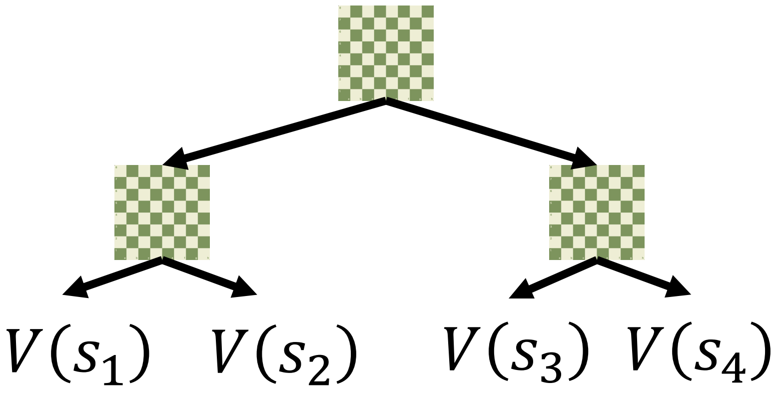
- Game tree too large to traverse explicitly
- **Value Function** $V(s)$ approximates how "good or bad" each state is

Previously (e.g., Deep Blue)

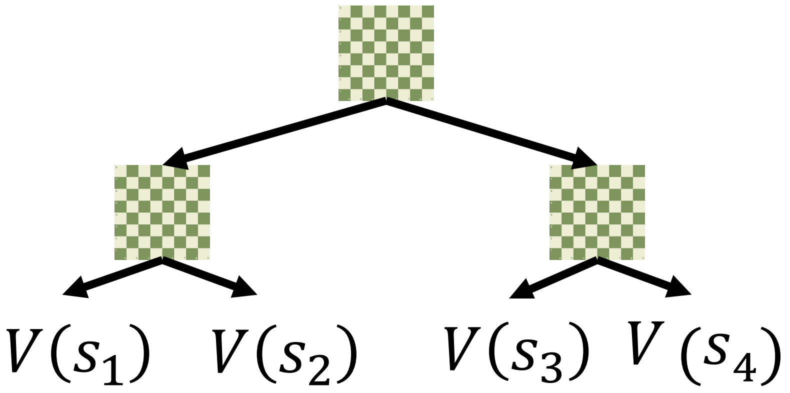
- Handcrafted evaluation function $V(s)$ based on heuristics from experts

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

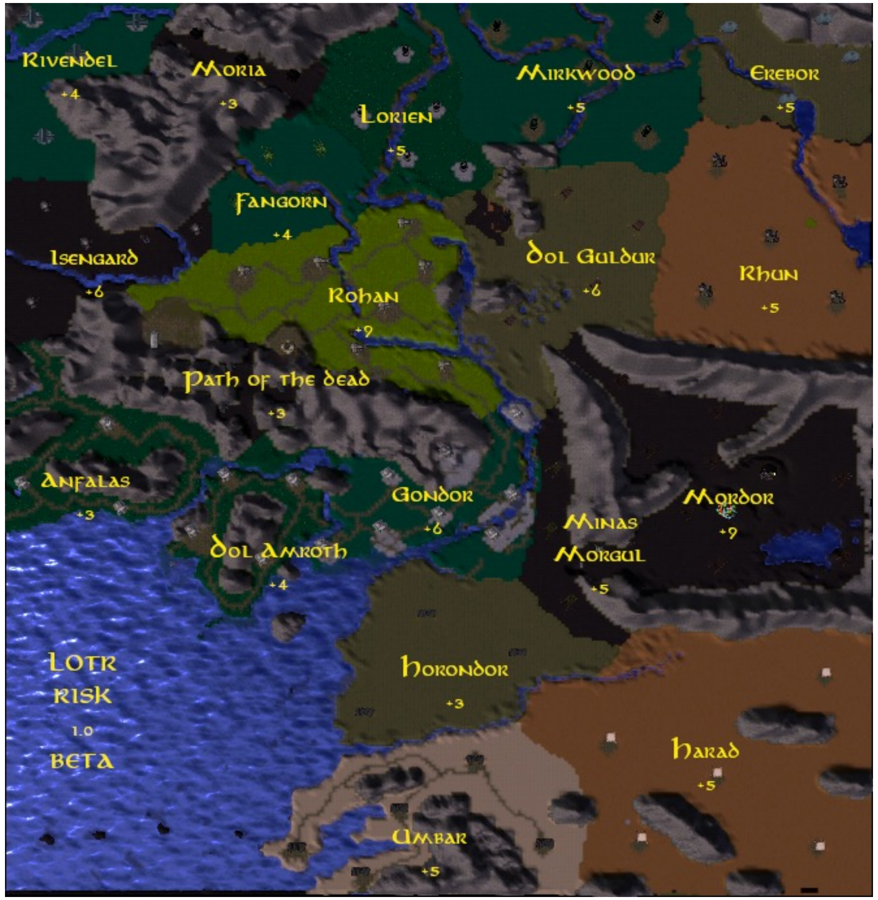
Today (e.g., AlphaGo/Zero, DeepStack)

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

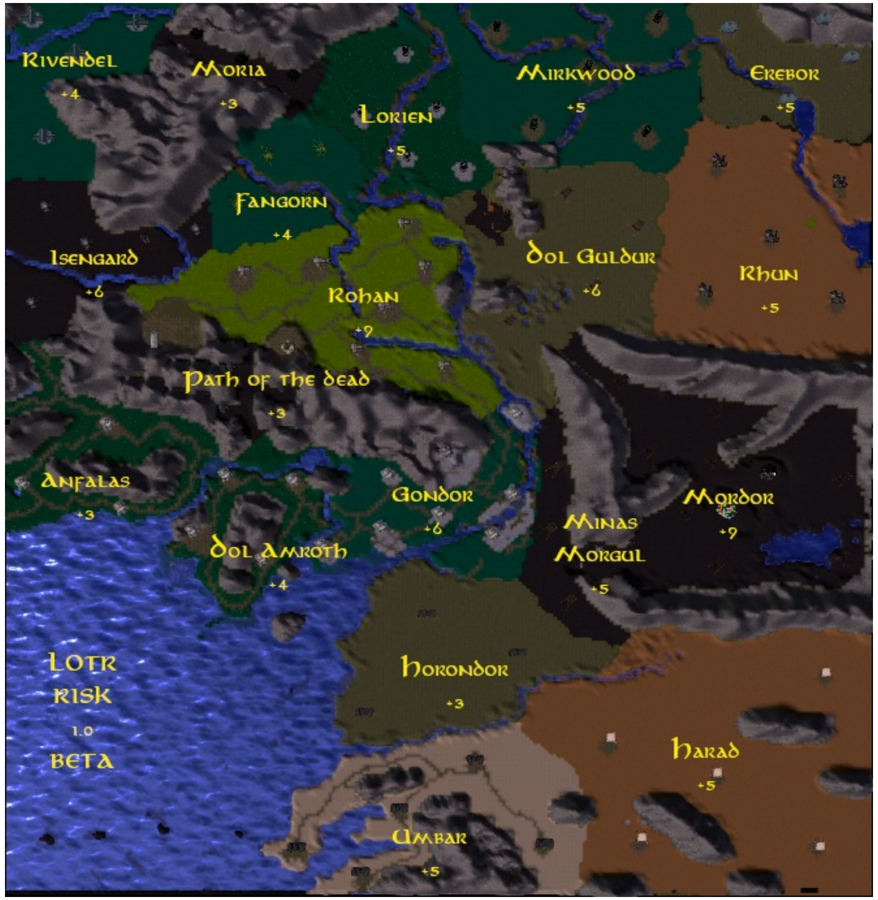
Also applies to general sum (Stackelberg) games in a principled manner!

Need to predicting **achievable sets** of payoffs, "value" of vertex is not just a scalar or fixed size vector

Multidefender Security Games



Multidefender Security Games

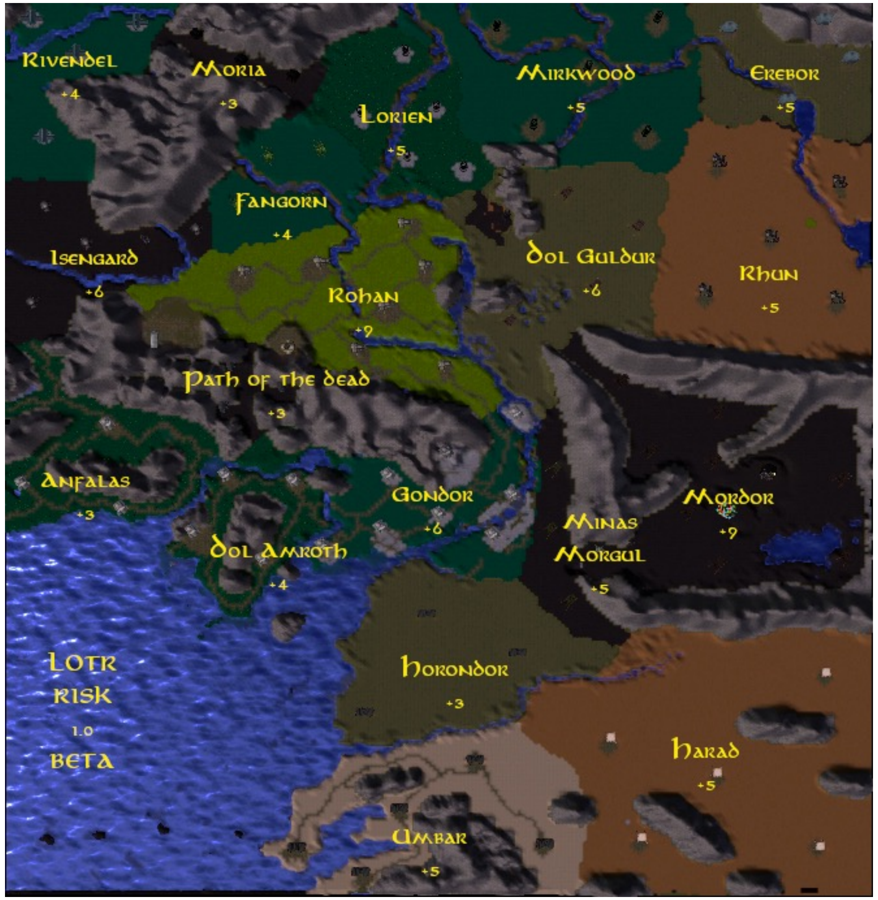


attacker

v.s.

defender**s**

Multidefender Security Games



v.s.

attacker

defender**s**

Defenders *each* choose how to distribute their armies. Orcs choose a target to attack.

Multidefender Security Games

v.s.

attacker

defender**s**

Defenders *each* choose how to distribute their armies. Orcs choose a target to attack.

Multidefender Security Games

v.s.

attacker

defender**s**

Defenders *each* choose how to distribute their armies. Orcs choose a target to attack.

Heterogenous defenders

Multidefender Security Games

v.s.



attacker

defenders

Defenders *each* choose how to distribute their armies. Orcs choose a target to attack.

Heterogenous defenders	Defensive Schedules

Multidefender Security Games

v.s.

attacker

defender**s**

Defenders *each* choose how to distribute their armies. Orcs choose a target to attack.

Heterogenous defenders

Defensive Schedules

Is there a "stable" allocation of defensive resources?

Not always but yes, under some assumptions.

Multidefender Security Games

attacker

v.s.

defenders

Defenders *each* choose how to distribute their armies. Orcs choose a target to attack.

Heterogenous defenders

Defensive Schedules

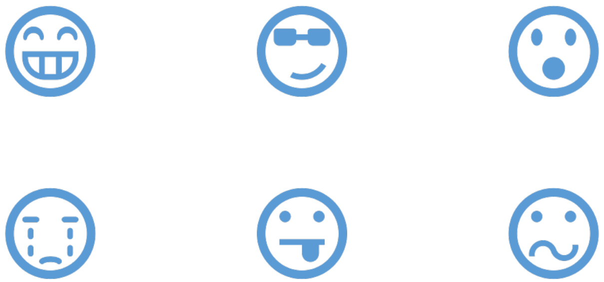
Is there a “stable” allocation of defensive resources?

Not always but yes, under some assumptions.

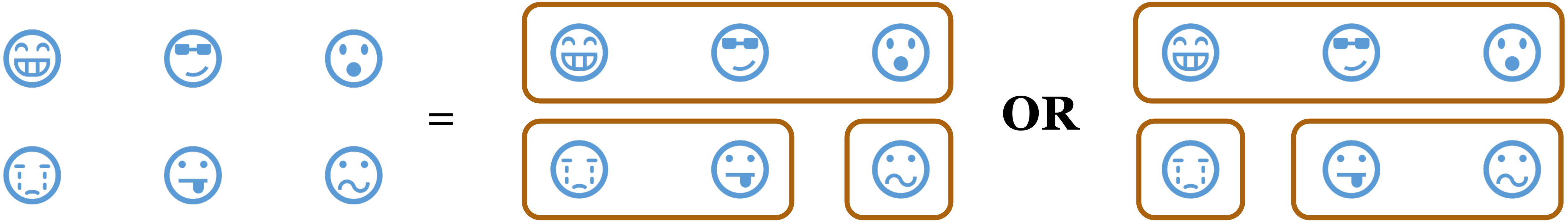
~~Democracy is two wolves and a sheep voting on whats for dinner.~~

Democracy is two or more sheet deciding who to throw to the wolves.

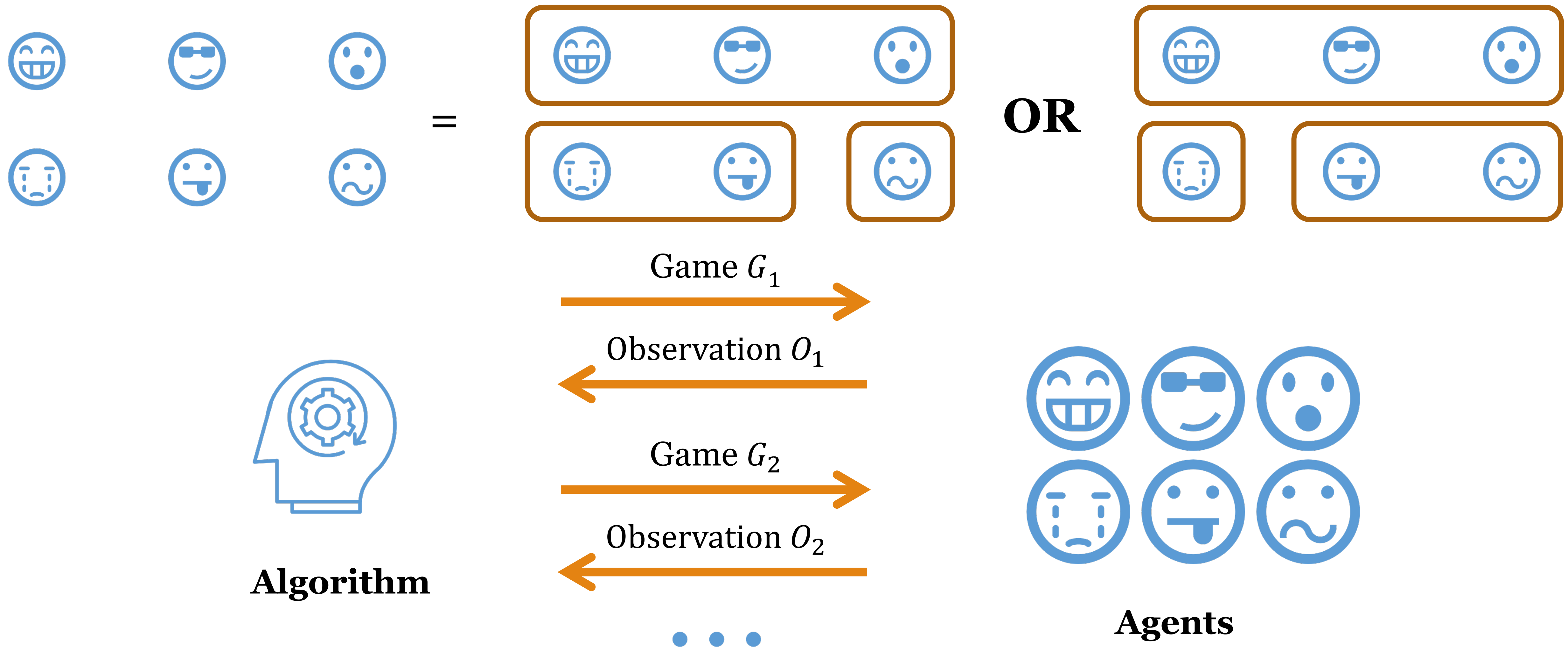
Coalition Structure Learning



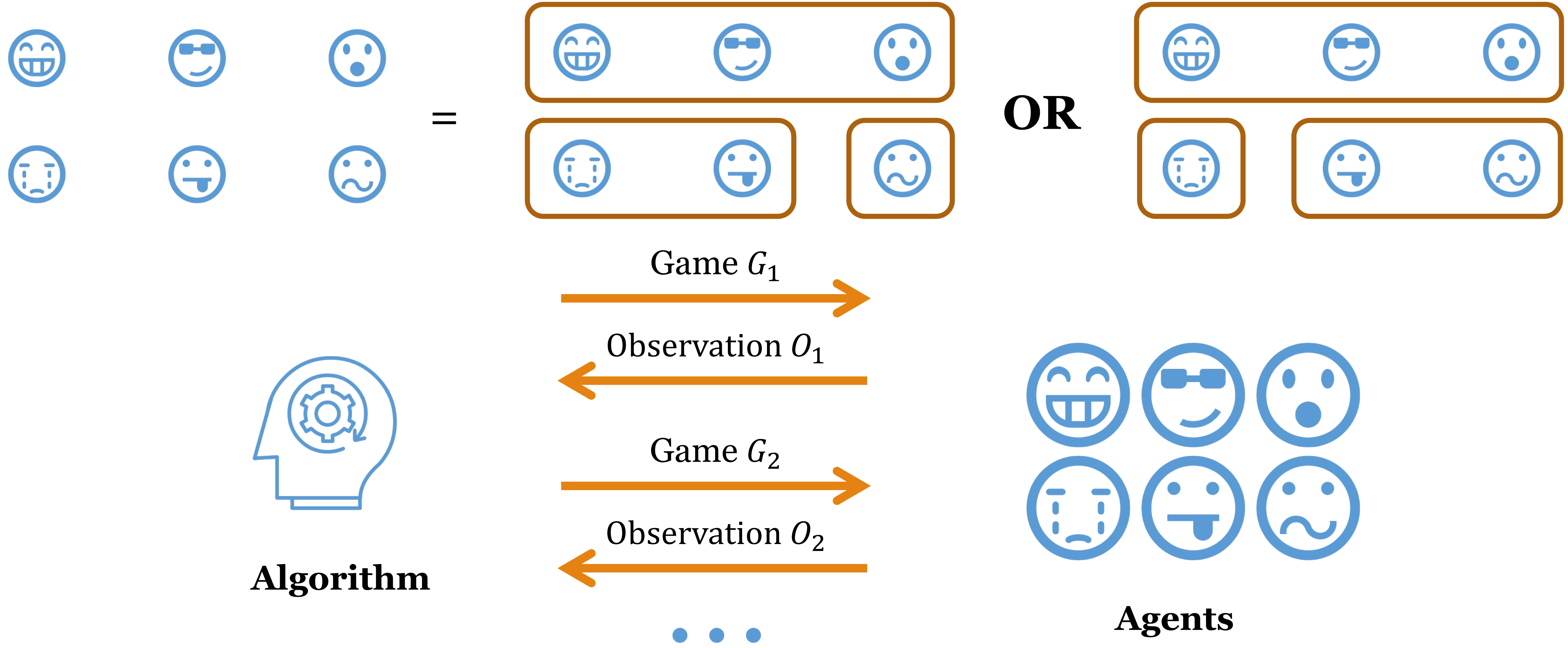
Coalition Structure Learning



Coalition Structure Learning



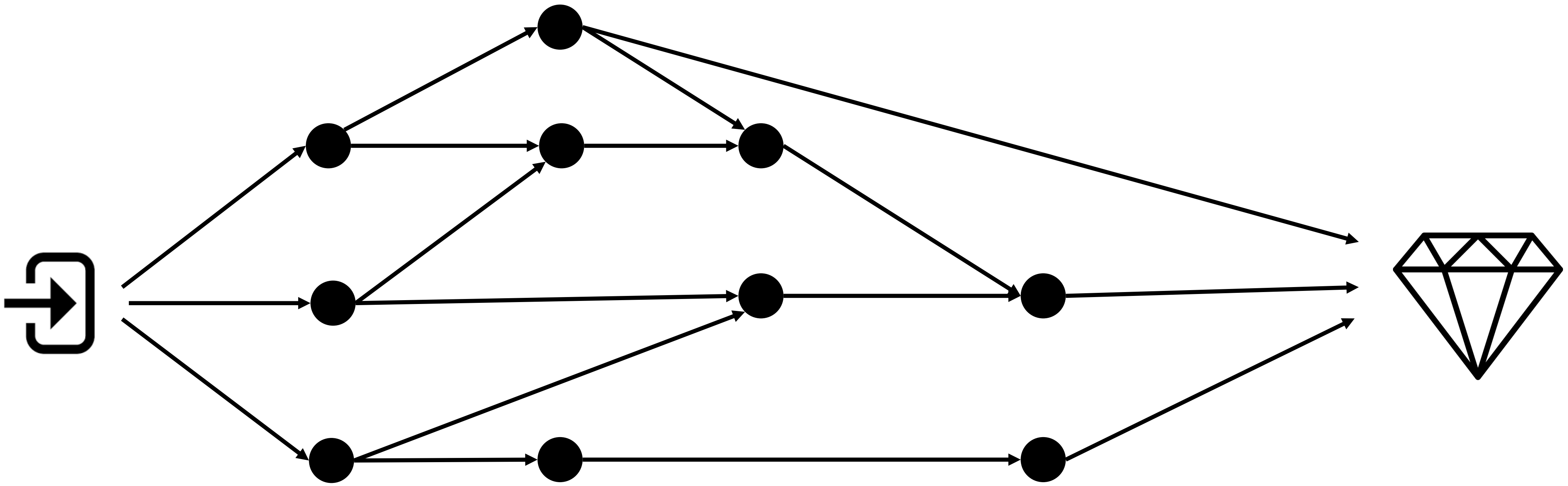
Coalition Structure Learning



Discovers true optimal in smallest number of tests!

Example: Network Protection Game

*Abstracted and super simplified

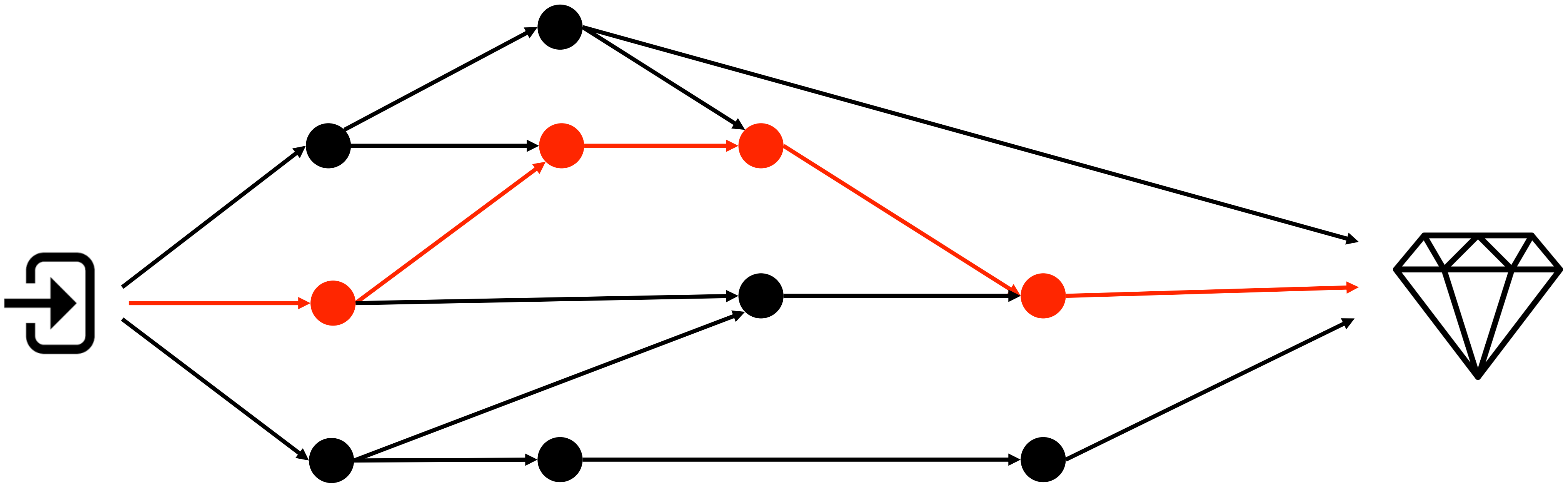


Nodes: Machines, servers, routers

Edges: Potential vulnerabilities

Example: Network Protection Game

*Abstracted and super simplified

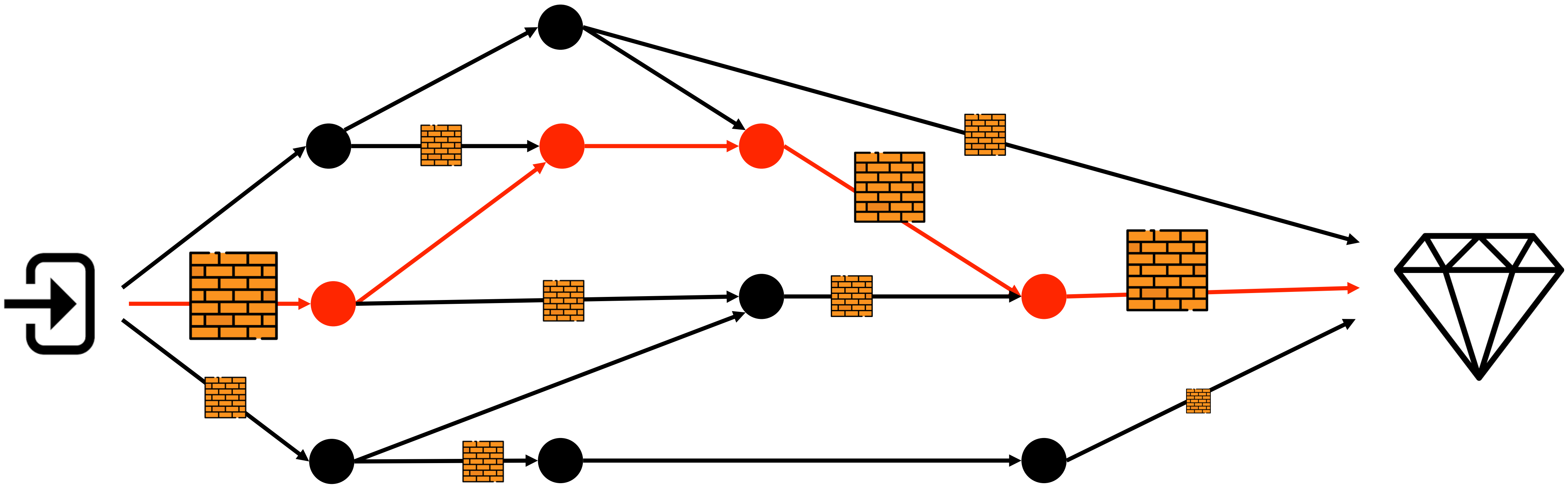


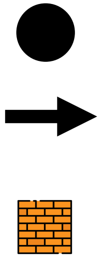
Nodes: Machines, servers, routers

Edges: Potential vulnerabilities

Example: Network Protection Game

*Abstracted and super simplified





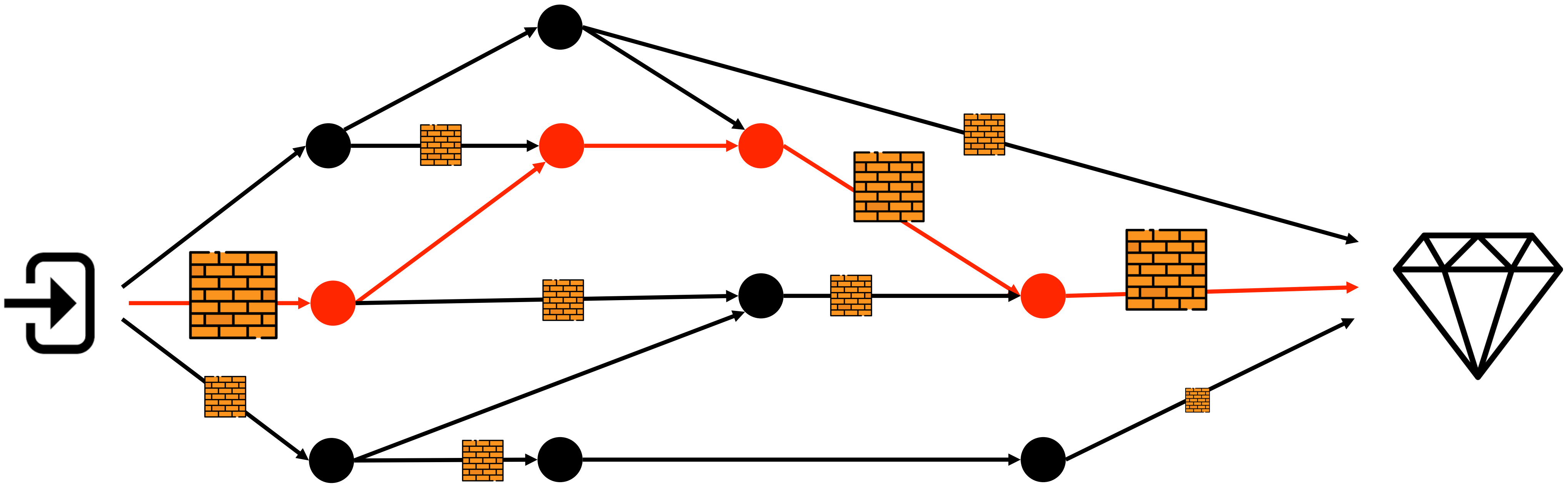
Nodes: Machines, servers, routers

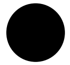
Edges: Potential vulnerabilities

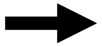
Walls: Defenses controls to be beefed up, subject to budget/performance constraint

Example: Network Protection Game

*Abstracted and super simplified



 Nodes: Machines, servers, routers

 Edges: Potential vulnerabilities

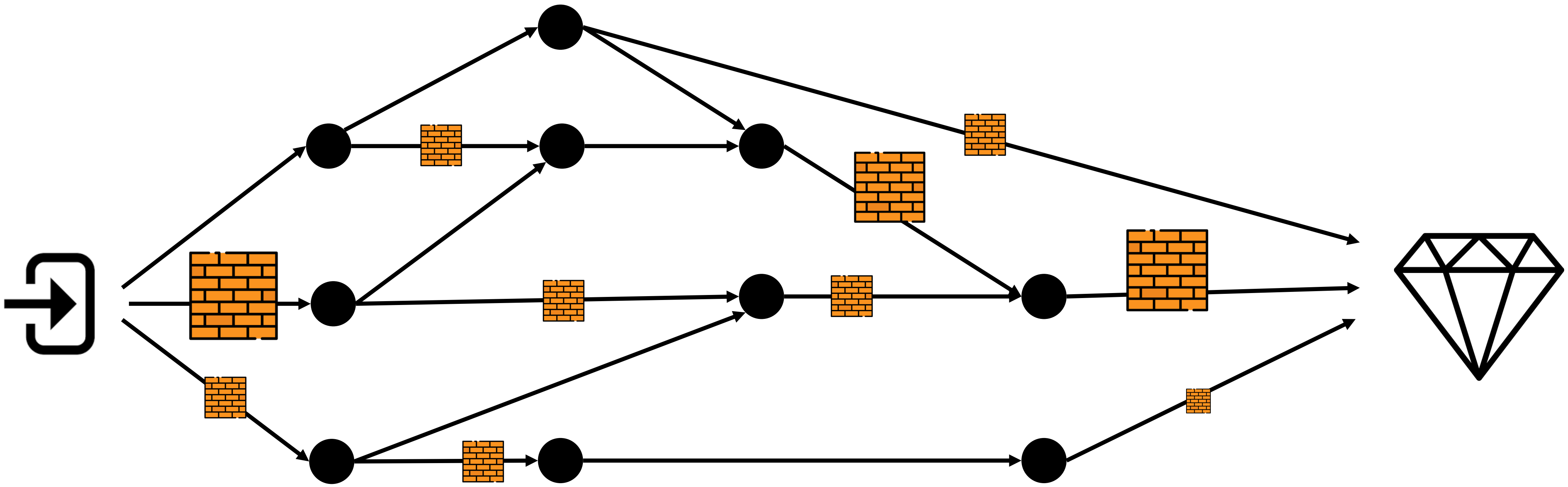
 Walls: Defenses controls to be beefed up, subject to budget/performance constraint

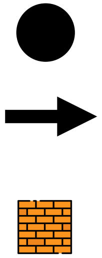
Reduces to a longest shortest path problem

(super easy for simple constraints, easily scale to around tens of millions of nodes)

Pivoting: Intruder is not stupid!

https://lingchunkai.github.io/pdf/AICS-2025.pdf





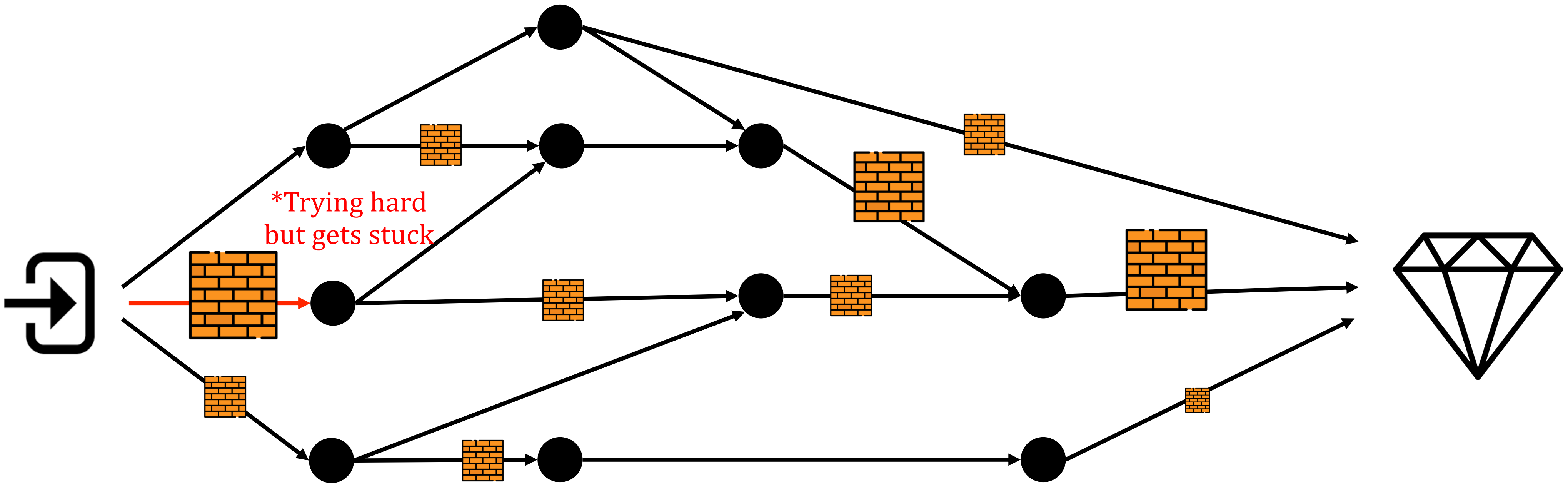
Nodes: Machines, servers, routers

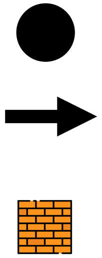
Edges: Potential vulnerabilities

Walls: Defenses controls to be beefed up, subject to budget/performance constraint

Pivoting: Intruder is not stupid!

https://lingchunkai.github.io/pdf/AICS-2025.pdf





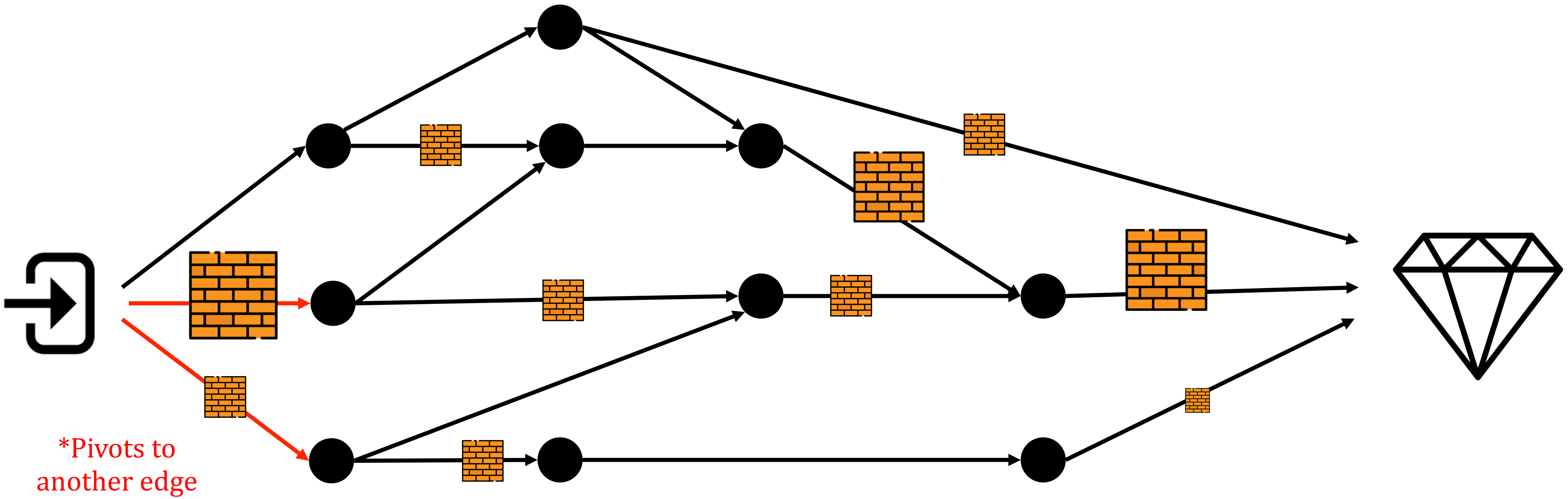
Nodes: Machines, servers, routers

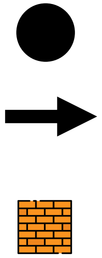
Edges: Potential vulnerabilities

Walls: Defenses controls to be beefed up, subject to budget/performance constraint

Pivoting: Intruder is not stupid!

https://lingchunkai.github.io/pdf/AICS-2025.pdf





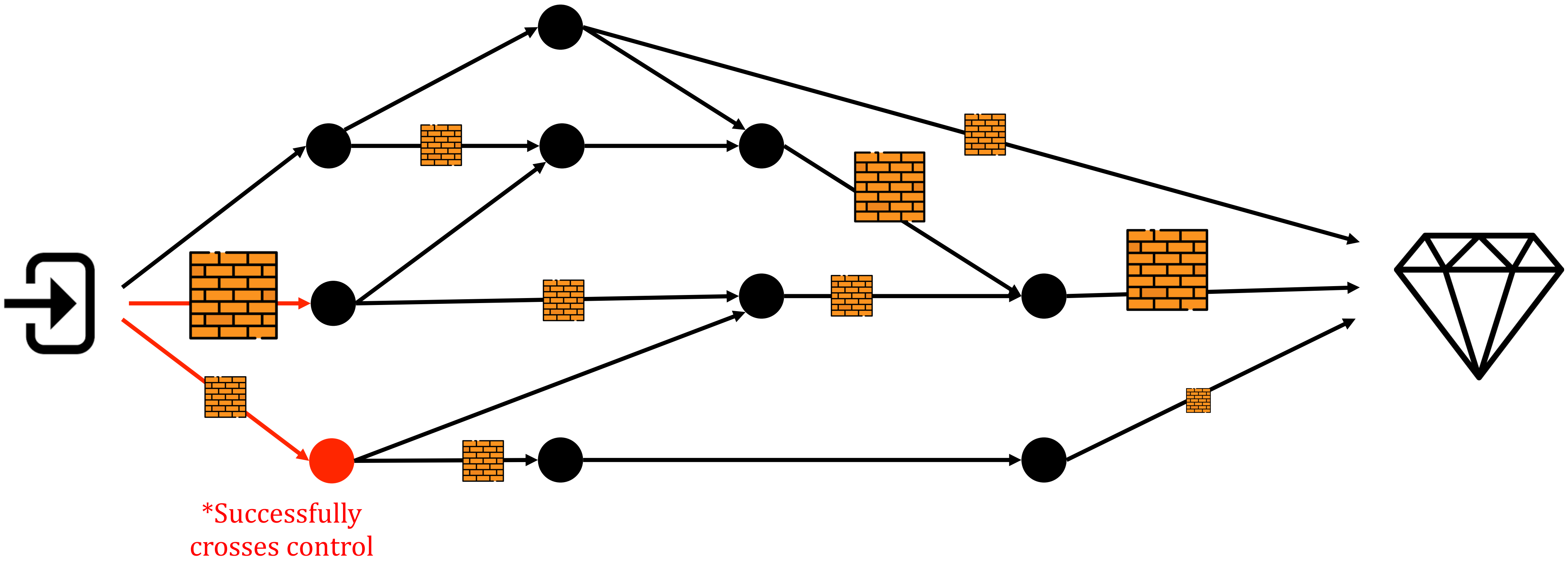
Nodes: Machines, servers, routers

Edges: Potential vulnerabilities

Walls: Defenses controls to be beefed up, subject to budget/performance constraint

Pivoting: Intruder is not stupid!

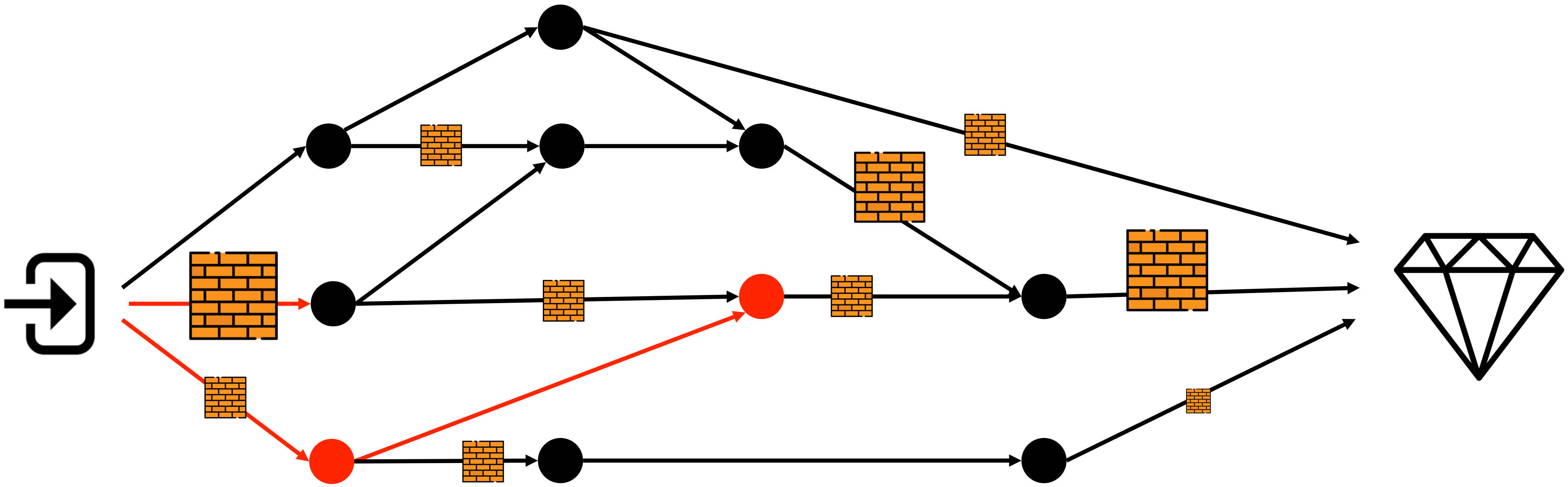
https://lingchunkai.github.io/pdf/AICS-2025.pdf

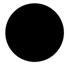


- Nodes: Machines, servers, routers
- Edges: Potential vulnerabilities
- Walls: Defenses controls to be beefed up, subject to budget/performance constraint

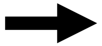
Pivoting: Intruder is not stupid!

https://lingchunkai.github.io/pdf/AICS-2025.pdf





Nodes: Machines, servers, routers

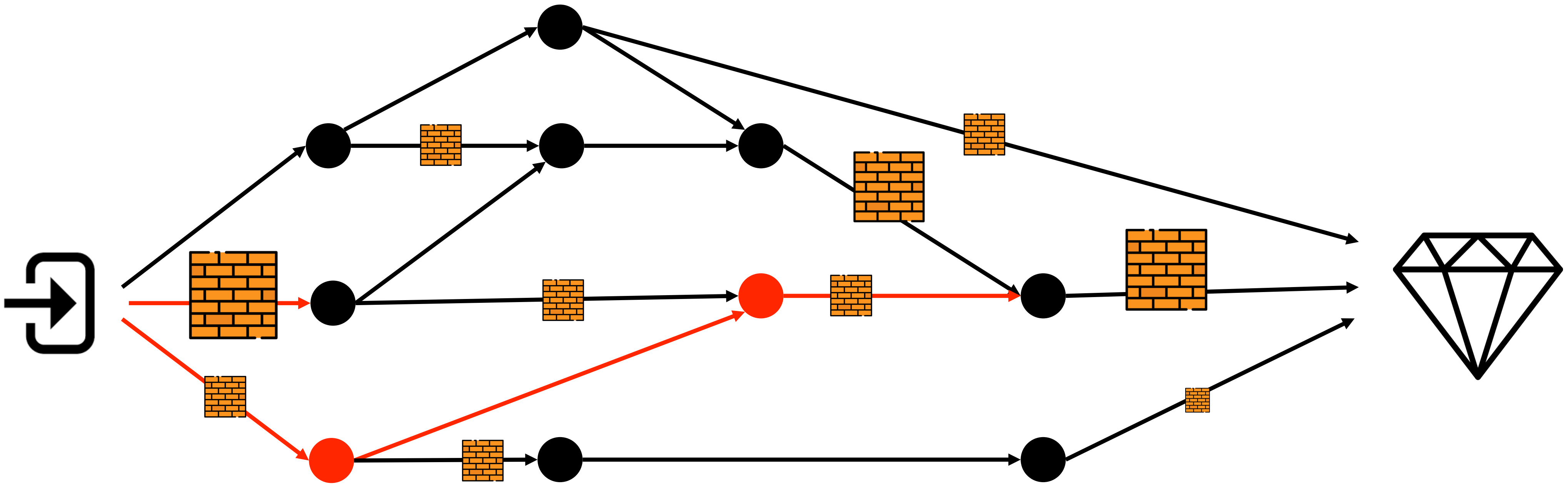


Edges: Potential vulnerabilities

Walls: Defenses controls to be beefed up, subject to budget/performance constraint

Pivoting: Intruder is not stupid!

https://lingchunkai.github.io/pdf/AICS-2025.pdf



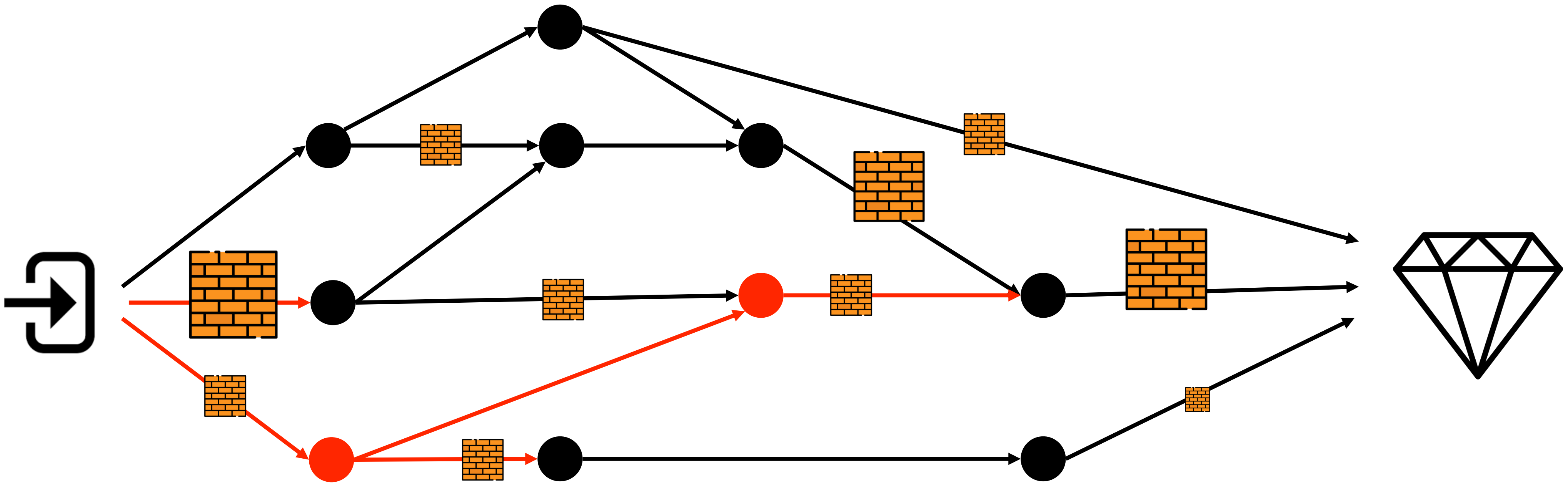
Nodes: Machines, servers, routers

Edges: Potential vulnerabilities

Walls: Defenses controls to be beefed up, subject to budget/performance constraint

Pivoting: Intruder is not stupid!

https://lingchunkai.github.io/pdf/AICS-2025.pdf



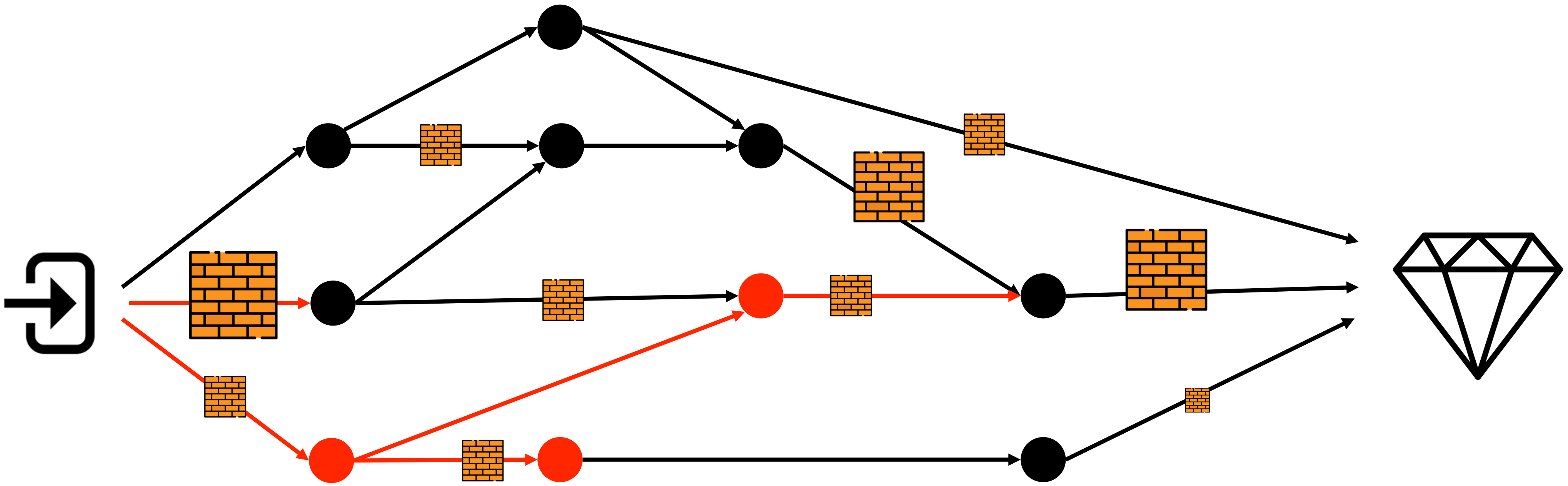
Nodes: Machines, servers, routers

Edges: Potential vulnerabilities

Walls: Defenses controls to be beefed up, subject to budget/performance constraint

Pivoting: Intruder is not stupid!

https://lingchunkai.github.io/pdf/AICS-2025.pdf



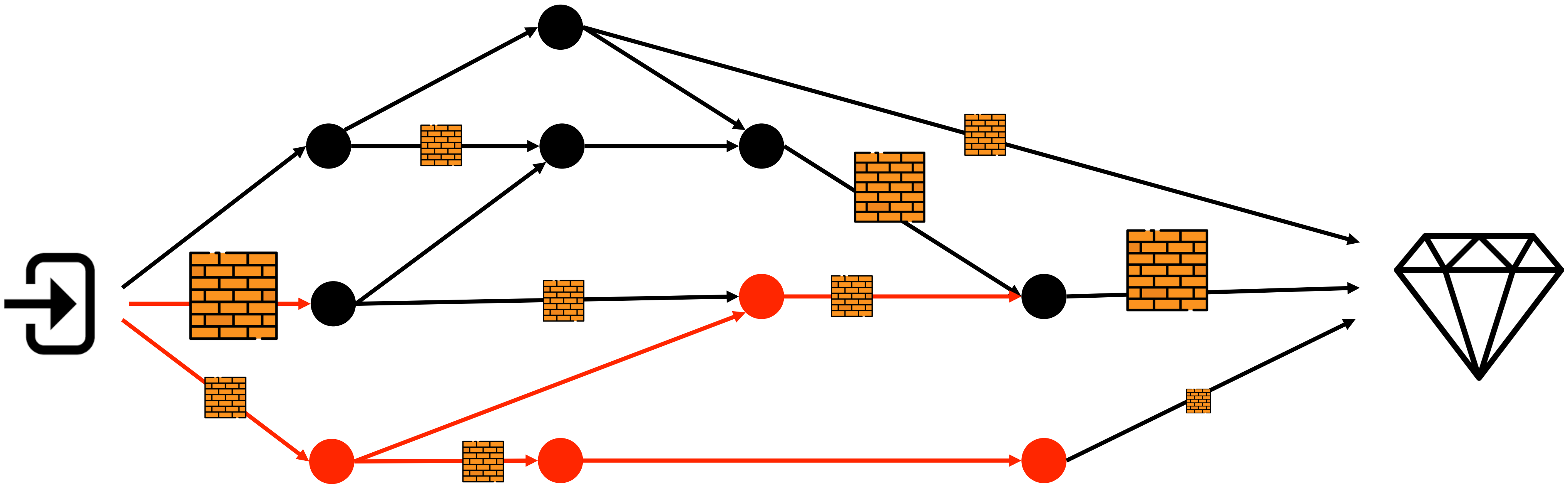
Nodes: Machines, servers, routers

Edges: Potential vulnerabilities

Walls: Defenses controls to be beefed up, subject to budget/performance constraint

Pivoting: Intruder is not stupid!

https://lingchunkai.github.io/pdf/AICS-2025.pdf



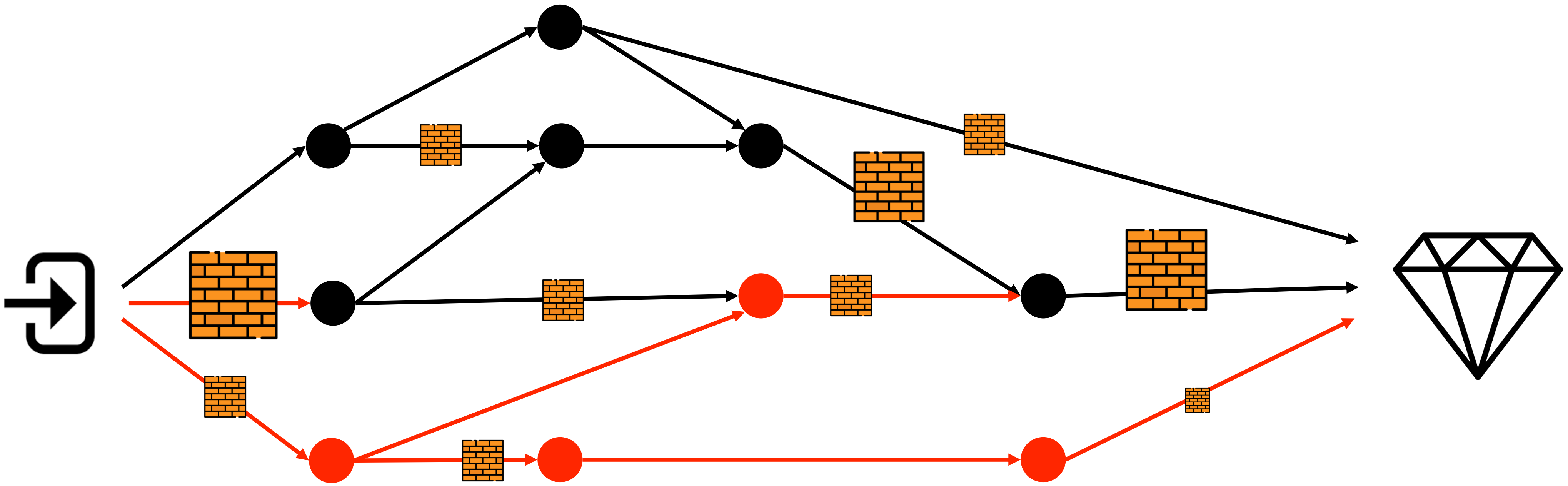
Nodes: Machines, servers, routers

Edges: Potential vulnerabilities

Walls: Defenses controls to be beefed up, subject to budget/performance constraint

Pivoting: Intruder is not stupid!

https://lingchunkai.github.io/pdf/AICS-2025.pdf



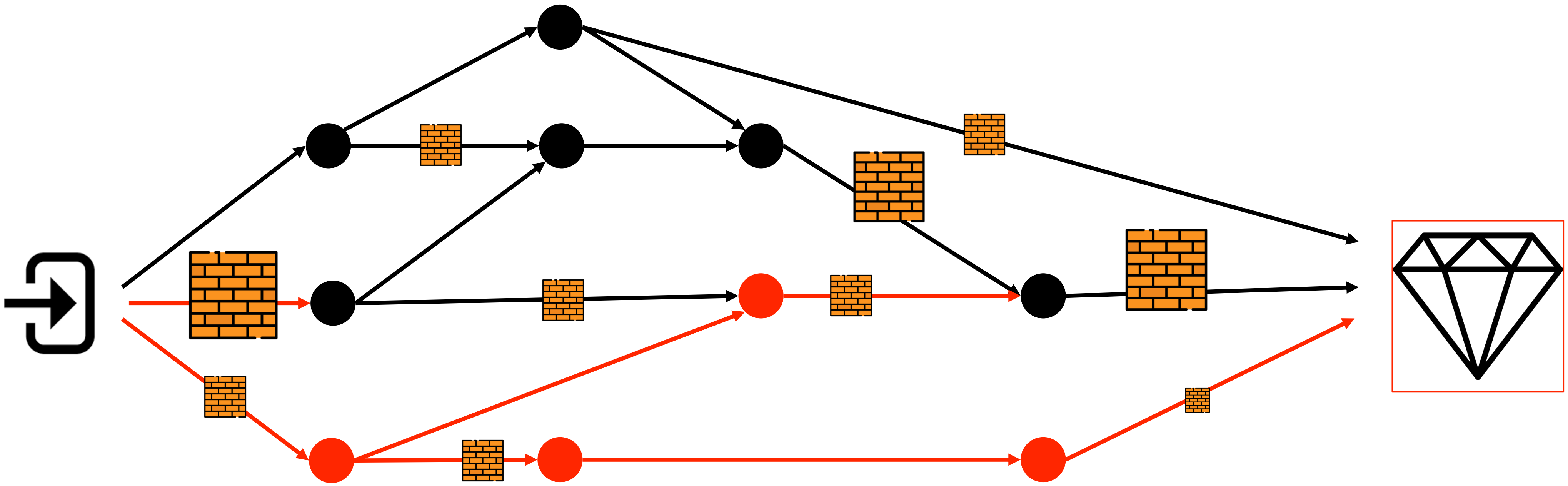
Nodes: Machines, servers, routers

Edges: Potential vulnerabilities

Walls: Defenses controls to be beefed up, subject to budget/performance constraint

Pivoting: Intruder is not stupid!

https://lingchunkai.github.io/pdf/AICS-2025.pdf



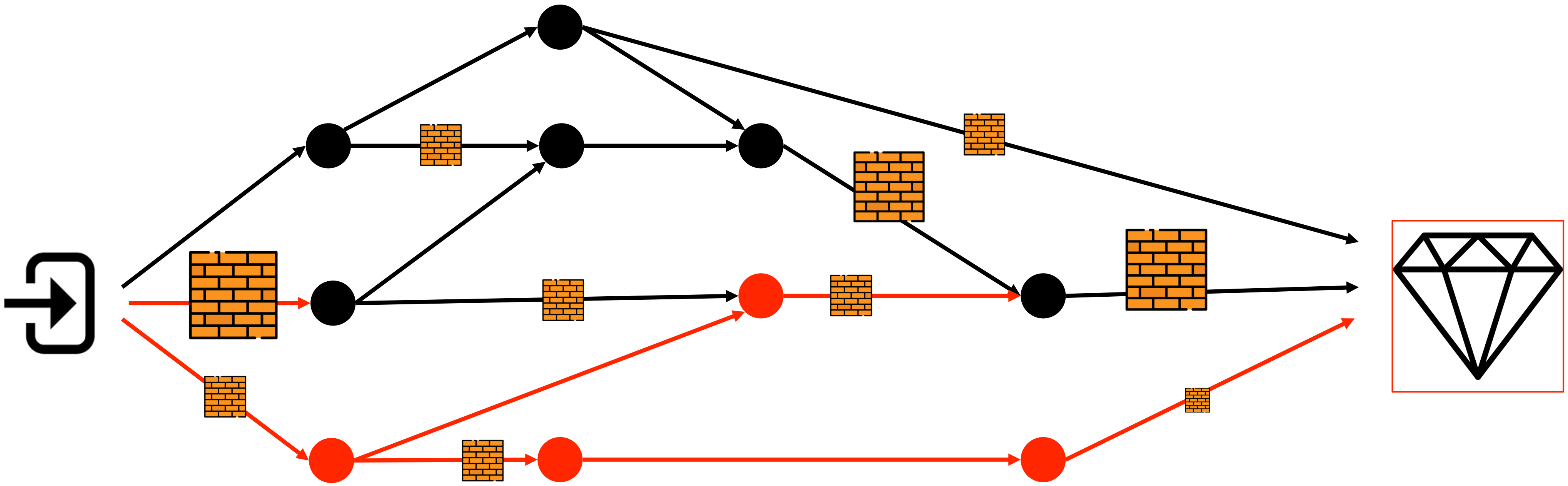
Nodes: Machines, servers, routers

Edges: Potential vulnerabilities

Walls: Defenses controls to be beefed up, subject to budget/performance constraint

Pivoting: Intruder is not stupid!

https://lingchunkai.github.io/pdf/AICS-2025.pdf



- Nodes: Machines, servers, routers
- Edges: Potential vulnerabilities
- Walls: Defenses controls to be beefed up, subject to budget/performance constraint

Given intruder's action **changes with time as they explore the network,** how should defenses be placed?

*Interesting connection with the Gittins index, ongoing work

Ongoing research

Dealing with stochastic actions sets (looking for students!)
- Not all actions available, can we still solve this **efficiently**?

Security games with discovery (looking for students)
- Attacker doesn't actually know the full network, need to account for that or else defender is at too big a disadvantage

Structured reward functions that are defined formally by commonly used semantics (e.g., PCTL, LTL)
- Can we still get fast algorithms?

Structured eqm are very interesting, but assumptions not necessarily true
- E.g., Gittins policy can be quite brittle, make several assumptions, but necessary for tractibility
- How can we handle slight deviations from these assumptions?
- Do we swing all the way to deep learning? Or is there a better balance between the 2?

Direction 2: Sensemaking in Gams

Direction 2: Sensemaking in Gams

What do people truly care about in applications? *In my arguably limited experience

Direction 2: Sensemaking in Gams

What do people truly care about in applications? *In my arguably limited experience

- Reasonable models of the problem
  - Modeling the entire world is virtually impossible!
  - Want to capture the key *strategic* aspects of gameplay
  - Possibly an iterative process
    - Don’t expect exact models given to you

Direction 2: Sensemaking in Gams

What do people truly care about in applications? *In my arguably limited experience

- Reasonable models of the problem
  - Modeling the entire world is virtually impossible!
  - Want to capture the key *strategic* aspects of gameplay
  - Possibly an iterative process
    - Don't expect exact models given to you
- Interesting strategies
  - Can game theoretic approaches **teach us about new ways of behaving**?
  - Simple strategies are also a plus!
  - Some quantitative guarantees
    - Saying XYZ "converged" isn't enough

Direction 2: Sensemaking in Gams

What do people truly care about in applications? *In my arguably limited experience

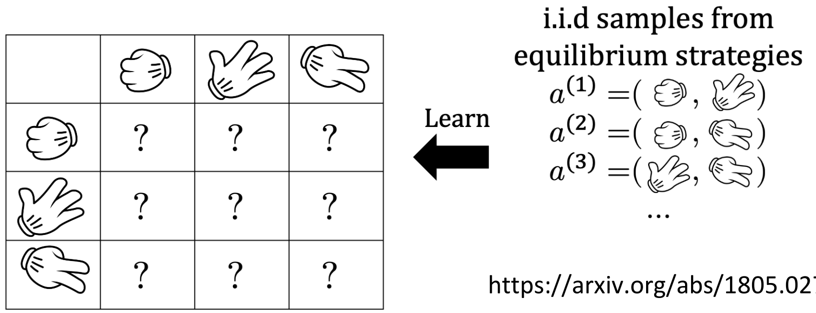
- Reasonable models of the problem
  - Modeling the entire world is virtually impossible!
  - Want to capture the key *strategic* aspects of gameplay
  - Possibly an iterative process
    - Don't expect exact models given to you
- Interesting strategies
  - Can game theoretic approaches **teach us about new ways of behaving**?
  - Simple strategies are also a plus!
  - Some quantitative guarantees
    - Saying XYZ "converged" isn't enough

*Yes, I know the audience here…

- Exact solutions: very rarely (outside of recreational games)
  - Does the 6^{th} significant digit matter if the model itself is inaccurate?

Inverse Game Theory/Computational Rationalization

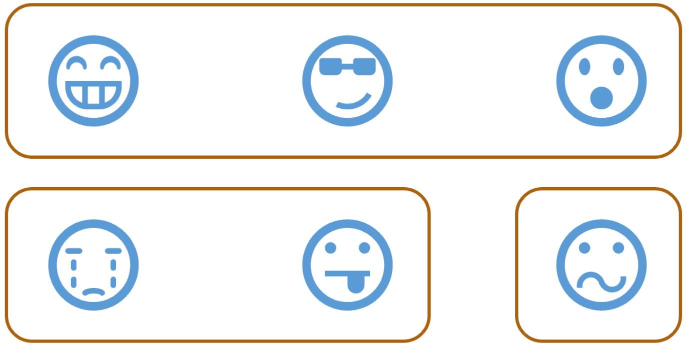
- Explain **why** players are behaving the way they do



https://arxiv.org/abs/1805.02777

https://arxiv.org/abs/1903.04101

IGT via differentiable optimization

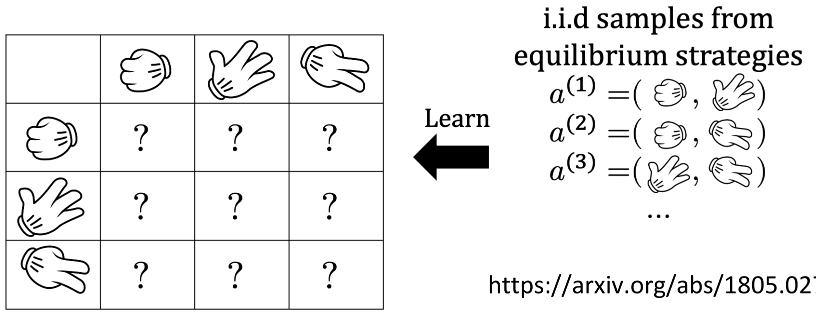


https://arxiv.org/abs/2312.09058

Coalition Structure Learning

Inverse Game Theory/Computational Rationalization

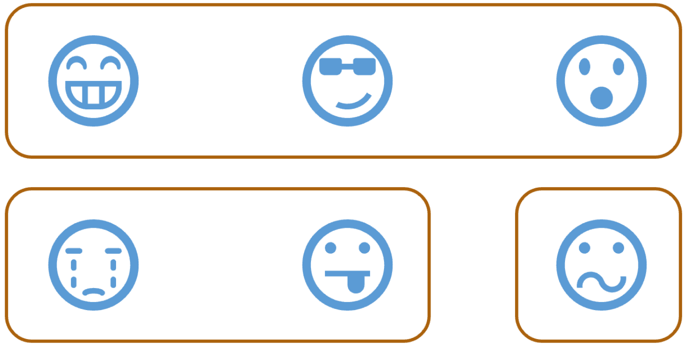
- Explain **why** players are behaving the way they do



https://arxiv.org/abs/1805.02777

https://arxiv.org/abs/1903.04101

IGT via differentiable optimization



https://arxiv.org/abs/2312.09058

Coalition Structure Learning

Learning strategies that **are** interpretable/operationalizable

- Restriction to strategies that are compact e.g., product distributions, sparse
- Regularization to “human-like” strategies

Explaining equilibrium selection

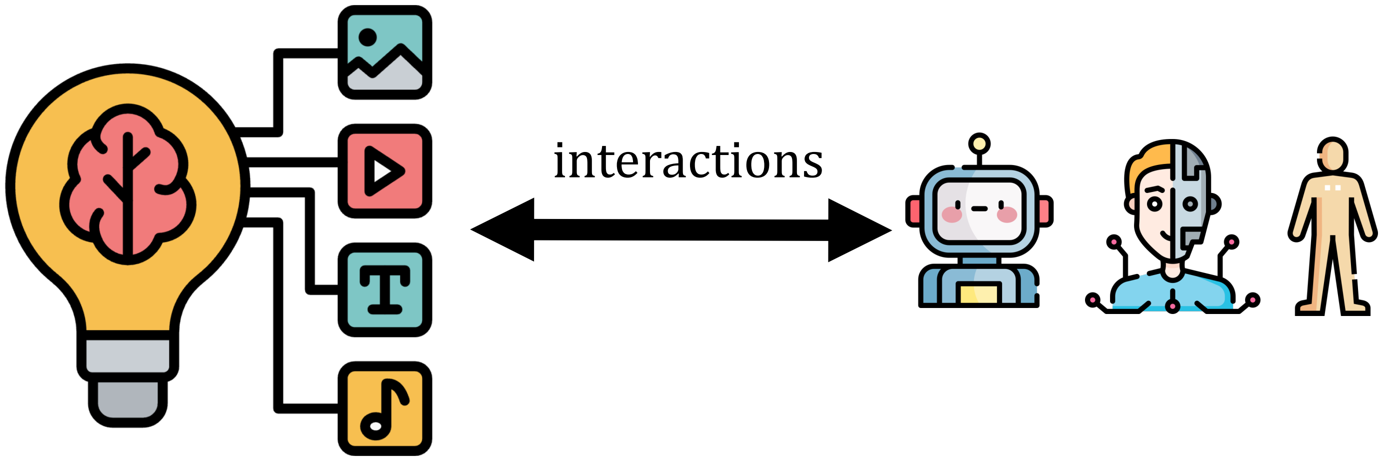
*ongoing work

Direction 3: Challenging Assumptions

Direction 3: Challenging Assumptions

Tackle new *problems* by allowing interactions in human modalities

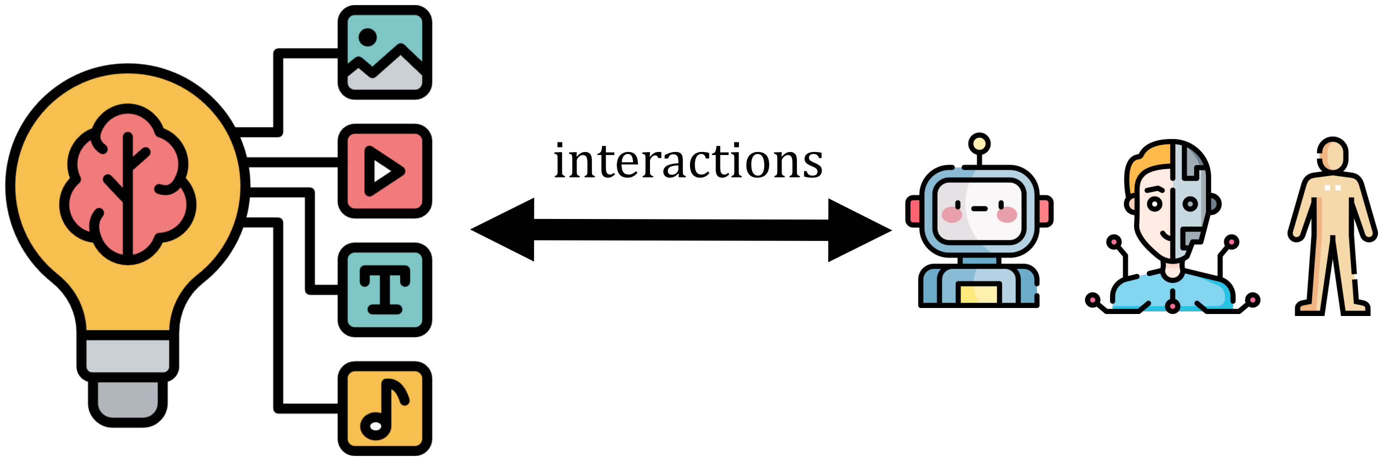
- Example: language models used in bargaining, negotiations
- How to incorporate strategy into such high-modality, fluid action spaces?
    - Strategically eliciting information from other language agents *ongoing work



Direction 3: Challenging Assumptions

Tackle new *problems* by allowing interactions in human modalities

- Example: language models used in bargaining, negotiations
- How to incorporate strategy into such high-modality, fluid action spaces?
  - Strategically eliciting information from other language agents *ongoing work



What about games without common prior?

Are equilibrium even the right concept?

Ongoing Projects

Learning **robust** strategies in a interactive setting

- Game of twenty questions

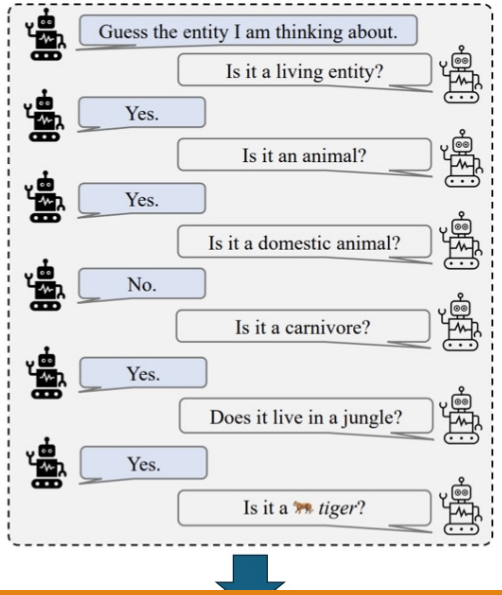
Explaining **strategies** to humans in a **meaningful** way

- Useful for persuasion, or just learning in general
- Much more tricky, need to care about **counterfactuals**

In general, for many of these problems…

- Behavior off the equilibrium path matters **a lot**
- Big difference between single agent and multiagent

Game of 20 Questions as an Extensive Form Game (EFG



Any Questions?

chunkail@nus.edu.sg

lingchunkai.github.io

Contact me if you'd like to collaborate!