Reminder

- Quiz for Lecture 3 (9/10, 10pm)
- Paper Bidding Result
- Paper Reading Assignment 1 (9/13, 10pm)
  - Peer reviewed (Due 1 week after assignment due)
- Confirm group members for course project (9/13, 10pm)

Advanced Topics in

Machine Learning and Game Theory

Lecture 3: Incremental Strategy Generation

17599/17759

Fei Fang

feifang@cmu.edu

Outline

- Security Games
- Double Oracle

Security Games to Model Security Challenges

Physical Infrastructure

Transportation Networks

Cyber Systems

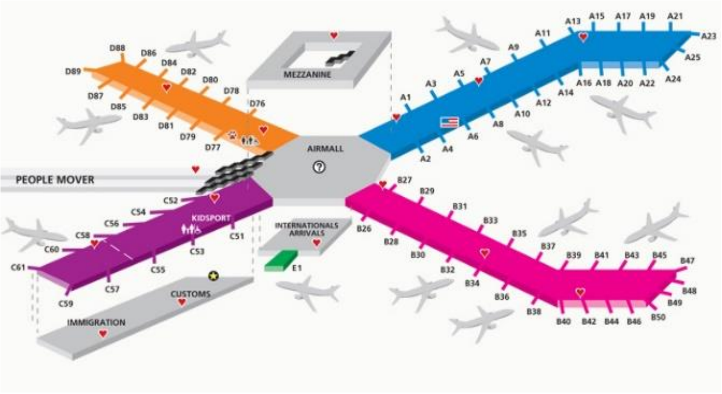
Environmental Resources

Endangered Wildlife

Fisheries

Security Games

- Limited resource allocation
- Adversary surveillance



Adversary

	Target #1	Target #2
Target #1	5, -3	-1, 1
Target #2	-5, 4	2, -1

Defender

Security Games

- Randomization make defender unpredictable
- Stackelberg game
  - Leader: Defender; Commits to mixed strategy
  - Follower: Adversary; Conduct surveillance and best responds

Adversary

Defender

		Target #1	Target #2
55.6%	Target #1	5, -3	-1, 1
44.4%	Target #2	-5, 4	2, -1

Stackelberg Security Game (SSG)

- Leader: defender; Follower: attacker
- Defender allocate K resources to protect N targets
- Each target is associated with 4 values: $R_i^d, P_i^d, R_i^a, P_i^a$
  - If attacker attacks target i and succeeds: attacker gets R_i^a and defender gets P_i^d
  - If attacker attacks target i and fails: attacker gets $P_i^a (\leq R_i^a)$ and defender gets $R_i^d (\geq P_i^d)$

Adversary

Defender

	T1	T2	T3
T1	5, -3	-1, 1	
T2	-5, 4	2, -1	
T3			

	T1	T2	T3
R_i^d			3
P_i^d			-2
R_i^a			6
P_i^a			-2

Q: how many numbers do we need to represent utility function?

Stackelberg Security Game (SSG)

- Leader: defender; Follower: attacker
- Defender allocate K resources to protect N targets
- Each target is associated with 4 values: $R_i^d, P_i^d, R_i^a, P_i^a$
  - If attacker attacks target i and succeeds: attacker gets R_i^a and defender gets P_i^d
  - If attacker attacks target i and fails: attacker gets $P_i^a (\leq R_i^a)$ and defender gets $R_i^d (\geq P_i^d)$

Adversary

Defender	T1	T2	T3
T1	5, -3	-1, 1	-2, 6
T2	-5, 4	2, -1	-2, 6
T3	-5, 4	-1, 1	3, -2

⟷

	T1	T2	T3
R_i^d	5	2	3
P_i^d	-5	-1	-2
R_i^a	4	1	6
P_i^a	-3	-1	-2

Q: how many numbers do we need to represent utility function?

Quiz 1

If attacker attacks target i and succeeds: attacker gets R_i^a and defender gets P_i^d

If attacker attacks target i and fails: attacker gets $P_i^a (\leq R_i^a)$ and defender gets $R_i^d (\geq P_i^d)$

- Let c_i be the probability the defender will protect target i in a Stackelberg security game, which ones of the following are the defender's expected utility when attacker attacks target i?
- A: $c_i P_i^a + (1 - c_i) R_i^a$
- B: $c_i R_i^d + (1 - c_i) P_i^d$
- C: $P_i^d + c_i (R_i^d - P_i^d)$
- D: $R_i^a + c_i (P_i^a - R_i^a)$
- E: None of the above

Compute SSE in SSG

$$AttEU(i) = c_i P_i^a + (1 - c_i) R_i^a$$
$$DefEU(i) = c_i R_i^d + (1 - c_i) P_i^d$$

- Strong Stackelberg Equilibrium
  - Attacker break tie in favor of defender
  - AttEU(1)=0.556*(-3)+0.444*4=0.11
  - AttEU(2)=0.556*1+0.444*(-1)=0.11
  - DefEU(1)=0.556*5+0.444*(-5)=0.56
  - DefEU(2)=0.556*(-1)+0.444*2=0.332
  - Equilibrium: DefStrat=(0.556,0.444), AttStrat=(1,0)

Adversary

Defender

		Target #1	Target #2
55.6%	Target #1	5, -3	-1, 1
44.4%	Target #2	-5, 4	2, -1

Computing SSE

$$AttEU(i) = c_i P_i^a + (1 - c_i) R_i^a$$
$$DefEU(i) = c_i R_i^d + (1 - c_i) P_i^d$$

- General-sum
  - Multiple LP (Conitzer & Sandholm, 2006)
    - One LP for each target: Assume attacks target i^*
    - Choose the solution of the LP with the highest optimal value

This approach applies to general Stackelberg games

Computing SSE

$$AttEU(i) = c_i P_i^a + (1 - c_i) R_i^a$$
$$DefEU(i) = c_i R_i^d + (1 - c_i) P_i^d$$

- General-sum
  - Multiple LP (Conitzer & Sandholm, 2006)
    - One LP for each target: Assume attacks target i^*

$$\boxed{\begin{aligned} &\max_{\mathbf{c}} DefEU(i^*) \\ \text{s.t. } &AttEU(i^*) \geq AttEU(i), \forall i = 1 \dots N \\ &\sum_i c_i \leq 1 \\ &c_i \in [0,1] \end{aligned}}$$

    - Choose the solution of the LP with the highest optimal value

This approach applies to general Stackelberg games

Computing SSE

$$AttEU(i) = c_i P_i^a + (1 - c_i) R_i^a$$
$$DefEU(i) = c_i R_i^d + (1 - c_i) P_i^d$$

- General-sum
  - MILP
    - Let $q_i \in \{0,1\}$ to indicate whether attacker attacks target i
    - Let M be a large constant, say 10^5

$$\max_{\mathbf{c},\mathbf{q},v} \sum_i DefEU(i) q_i$$
$$\text{s.t. } 0 \leq v - AttEU(i) \leq (1 - q_i) M, \forall i$$
$$\sum_i c_i \leq 1$$
$$\sum_i q_i = 1$$
$$c_i \in [0,1], q_i \in \{0,1\}$$

Computing SSE

$$AttEU(i) = c_i P_i^a + (1 - c_i) R_i^a$$
$$DefEU(i) = c_i R_i^d + (1 - c_i) P_i^d$$

- Zero-sum
  - Single LP
  - SSE=NE=Minimax=Maximin

$$\min_{\mathbf{c},v} v$$
$$\text{s.t. } v \geq AttEU(i), \forall i = 1 \ldots N$$
$$\sum_i c_i \leq 1$$
$$c_i \in [0,1]$$

ARMOR: Optimizing Security Resource Allocation [2007]

First application: Computational game theory for operational security

January 2009

- January 3rd — *Loaded 9/mm pistol*
- January 9th — *16-handguns, 1000 rounds of ammo*
- January 10th — *Two unloaded shotguns*
- January 12th — *Loaded 22/cal rifle*
- January 17th — *Loaded 9/mm pistol*
- January 22nd — *Unloaded 9/mm pistol*

ARMOR for AIRPORT SECURITY at LAX [2008]
Congressional Subcommittee Hearings

Commendations
City of Los Angeles

Erroll Southers testimony
Congressional subcommittee

Newsweek

ARMOR…throws a digital cloak of invisibility….

Protect Ferry Line

Compute optimal defender strategy

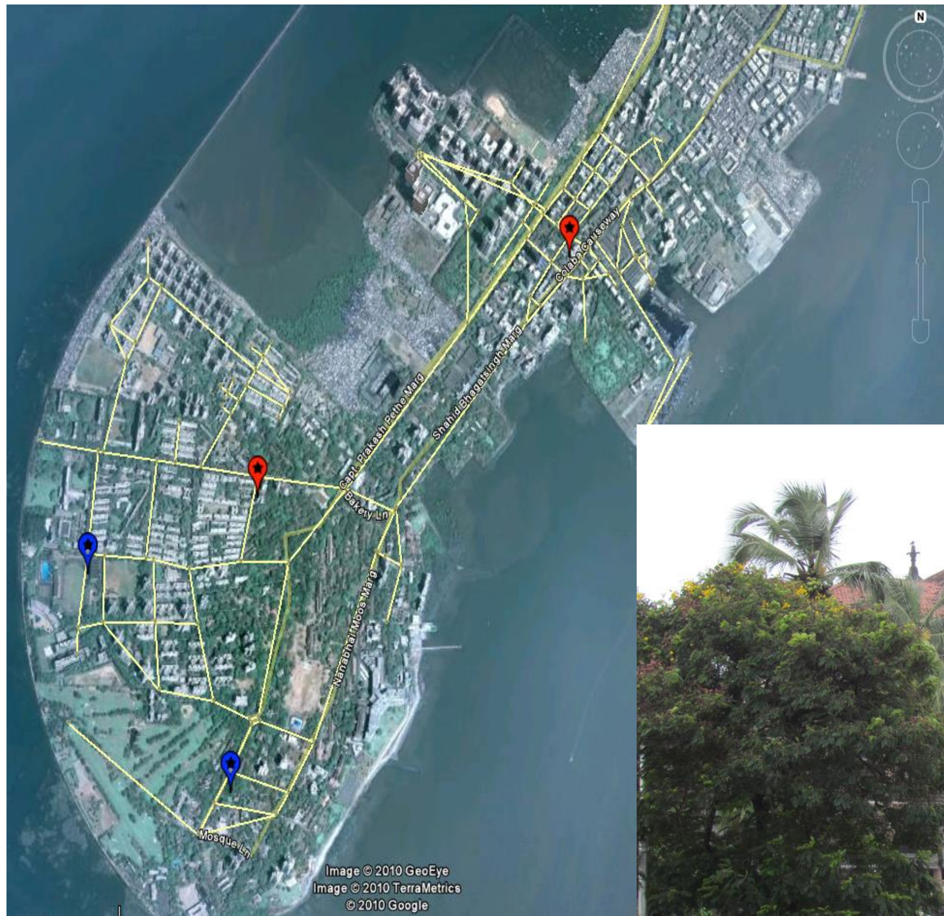
- Polynomial time solvable in games with finite actions and simple structures [Conitzer06]
- NP-Hard in general settings [Korzhyk10]
- SSE=NE for zero-sum games, SSE⊂NE for games with special properties [Yin10]

Outline

- Security Games
- Double Oracle

Challenge: Scheduling Constraints and Scalability

- Mumbai Police Checkpoints

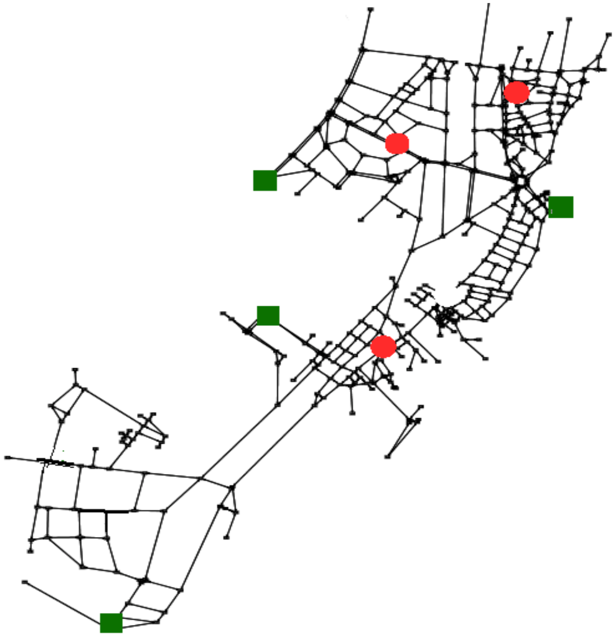


Challenge: Scheduling Constraints and Scalability

- Defender: Choose K checkpoints
- Attacker: Choose a target node (red) and a path from an entry node (green) to the target node
- Exponentially many pure strategies

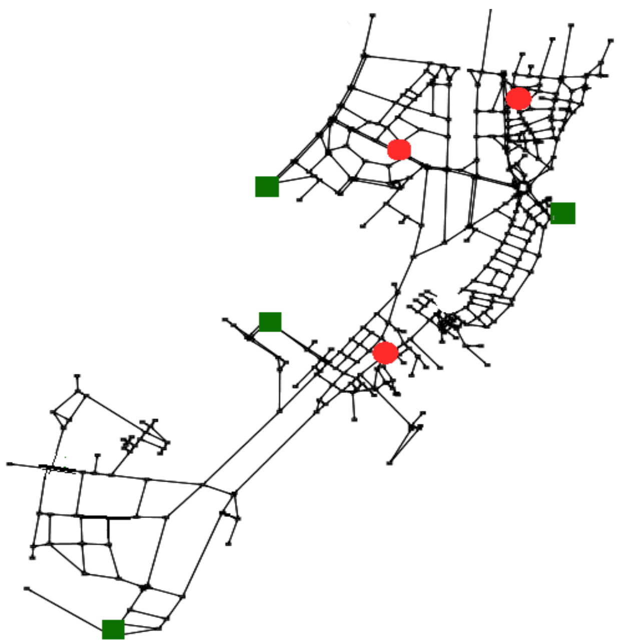
Fully connected road network
20 intersections, 190 roads
5 resources, 1 target
~ 2 billion defender allocations
6.6 *quintillion* (10^{18}) attacker paths

Real Problem:
~500 intersections
~2000 roads



Double Oracle

- Intuition: No need to consider all pure strategies
- Start with a small set of pure strategies
- Iteratively add new pure strategies to be considered
- Provably converge to equilibrium
  in zero-sum games



Payoff Matrix (When Zero-Sum)

Attacker Paths

Defender Allocations

$$
\begin{array}{cc}
 & \begin{array}{ccccc} A_1 & A_2 & A_3 & A_4 & \cdots \end{array} \\
\begin{array}{c} X_1 : \\ X_2 : \\ \vdots \end{array} &
\begin{bmatrix}
-5 & -8 & 0 & -9 & \cdots \\
0 & -8 & -15 & 0 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots
\end{bmatrix}
\end{array}
$$

Double Oracle Algorithm

$$\begin{array}{cc} & \begin{array}{cc} A_1 & A_2 \end{array} \\ X_1: & \begin{bmatrix} -5 & -8 \end{bmatrix} \end{array}$$

Minimax

$$\langle \mathbf{x}, \mathbf{a} \rangle$$

$$\begin{array}{cc} & \begin{array}{cc} A_1 & A_2 \end{array} \\ \begin{array}{c} X_1: \\ X_2: \end{array} & \begin{bmatrix} -5 & -8 \\ 0 & -8 \end{bmatrix} \end{array}$$

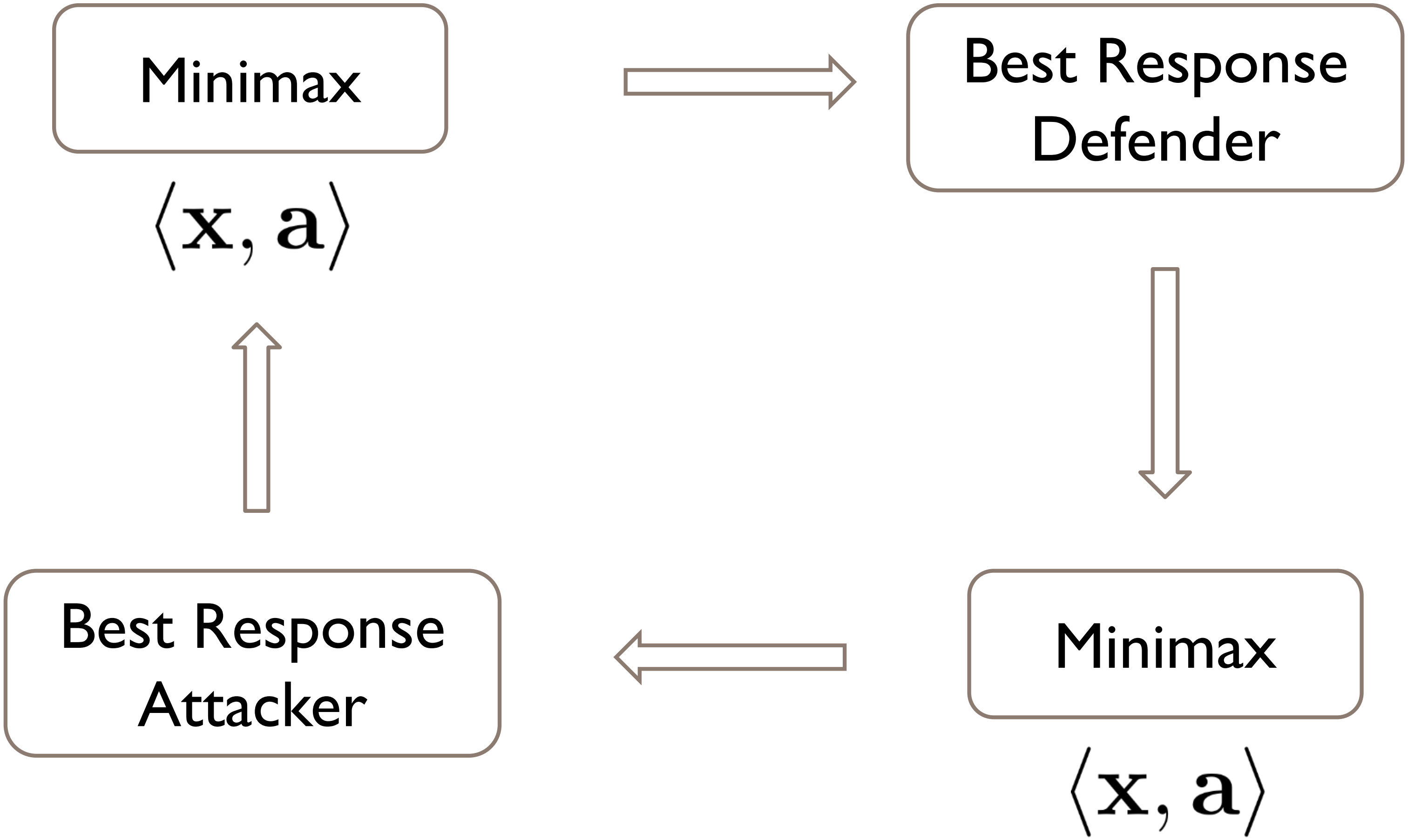
Best Response Defender

$$\begin{array}{cc} & \begin{array}{ccc} A_1 & A_2 & A_3 \end{array} \\ \begin{array}{c} X_1: \\ X_2: \end{array} & \begin{bmatrix} -5 & -8 & 0 \\ 0 & -8 & -15 \end{bmatrix} \end{array}$$

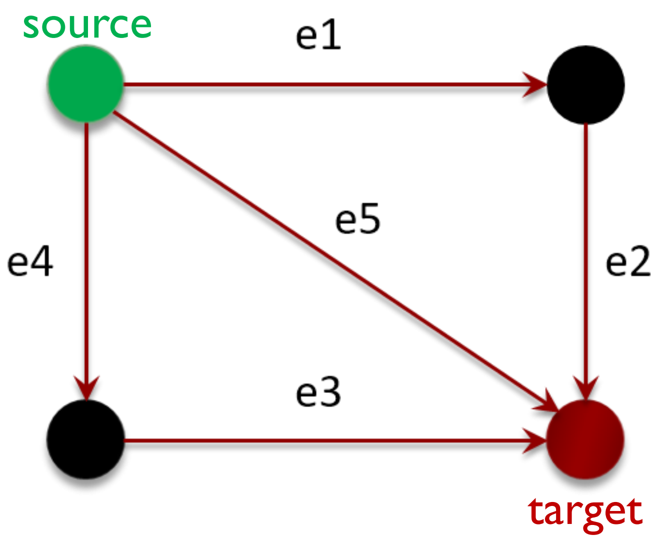
Best Response Attacker

[McMahan et. al 2003]

Variation


Minimax
$\langle \mathbf{x}, \mathbf{a} \rangle$
Best Response Defender
Minimax
$\langle \mathbf{x}, \mathbf{a} \rangle$
Best Response Attacker

Example



1 Defender Resource

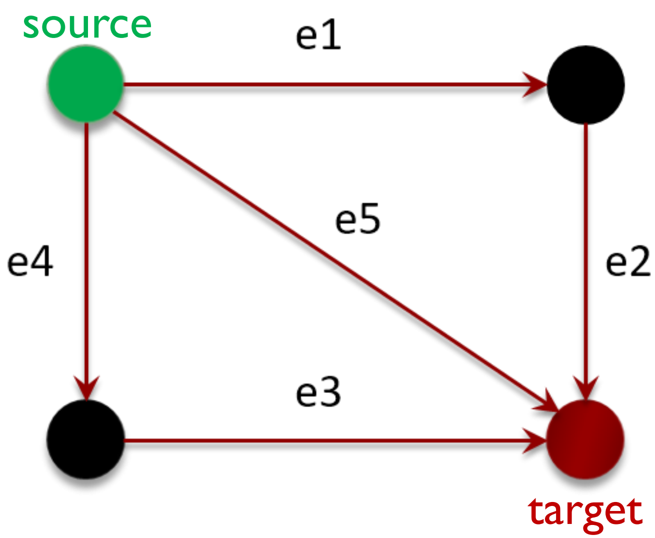
	Defender Payoff	Attacker Payoff
Attack Successful	-T	T
Attack Failure	0	0

Attacker Paths

Defender Allocations

	s->e1->e2->t	s->e5->t	s->e4->e3->t
e1			
e2			
e3			
e4			
e5			

Example



1 Defender Resource

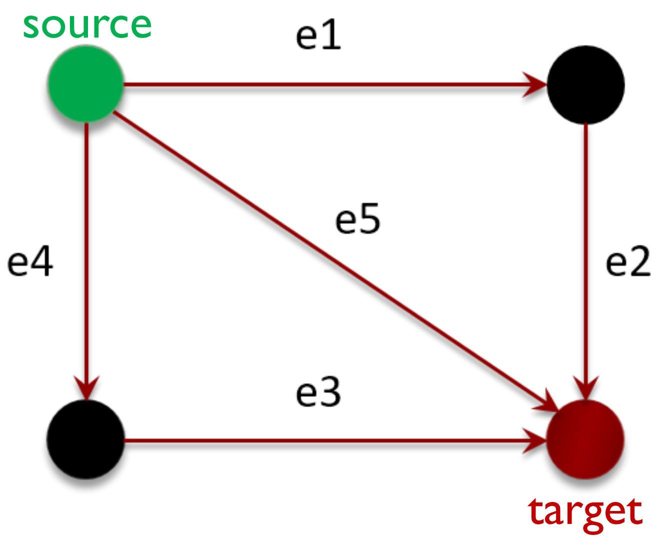
	Defender Payoff	Attacker Payoff
Attack Successful	-T	T
Attack Failure	0	0

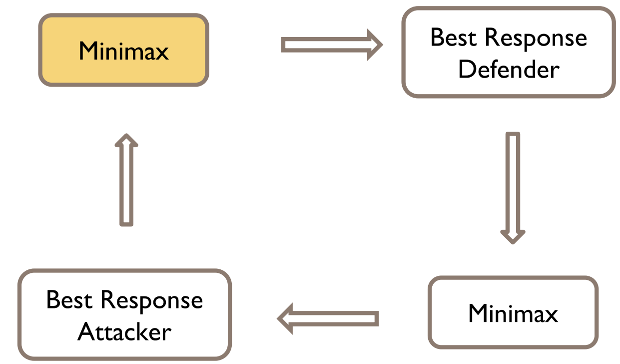
Attacker Paths

Defender Allocations

	s->e1->e2->t	s->e5->t	s->e4->e3->t
e1		-T, T	-T, T
e2		-T, T	-T, T
e3	-T, T	-T, T	
e4	-T, T	-T, T	
e5	-T, T		-T, T

Example





Attacker Paths

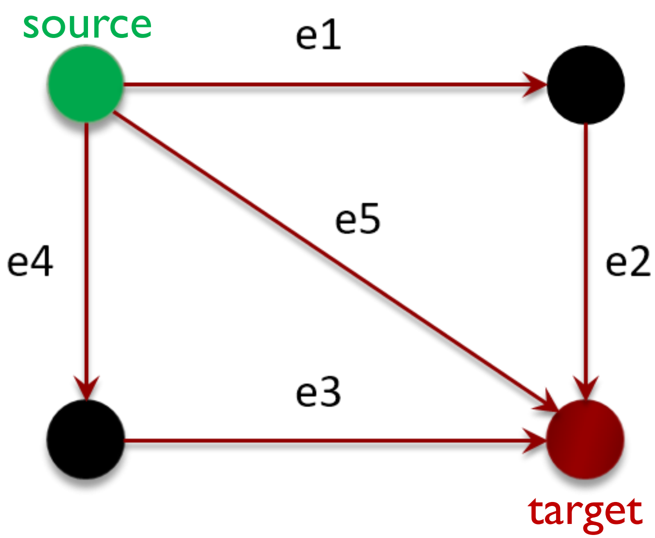
Defender Allocations

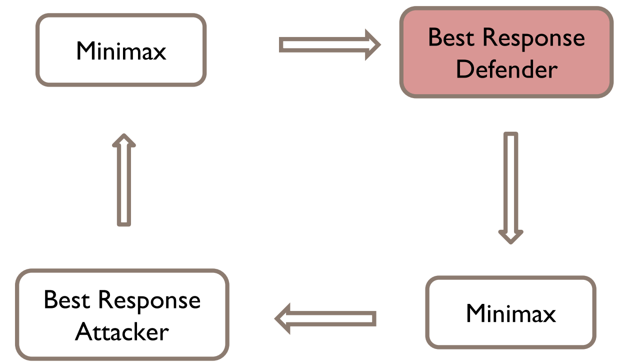
	s->e1->e2->t
e1	0,0

Minimax strategy:
Defender Strategy: [1.0]
Attacker Strategy: [1.0]

Example

Minimax strategy:
Defender Strategy: [1.0]
Attacker Strategy: [1.0]





Attacker Paths

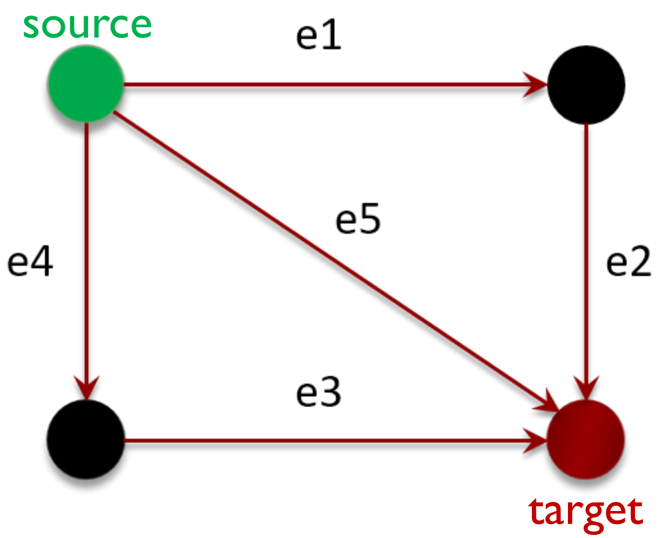
Defender Allocations

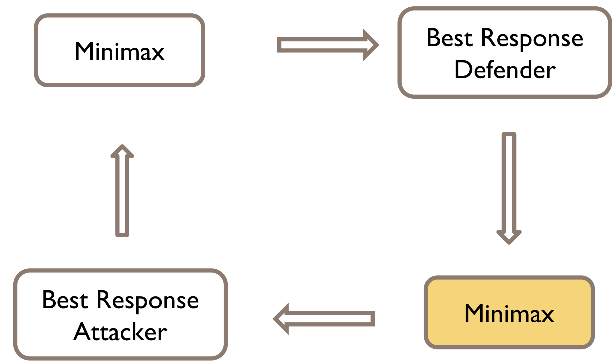
	s->e1->e2->t
e1	0,0

Defender’s best response: e1 or e2

Best response already in the table, no change

Example





Attacker Paths

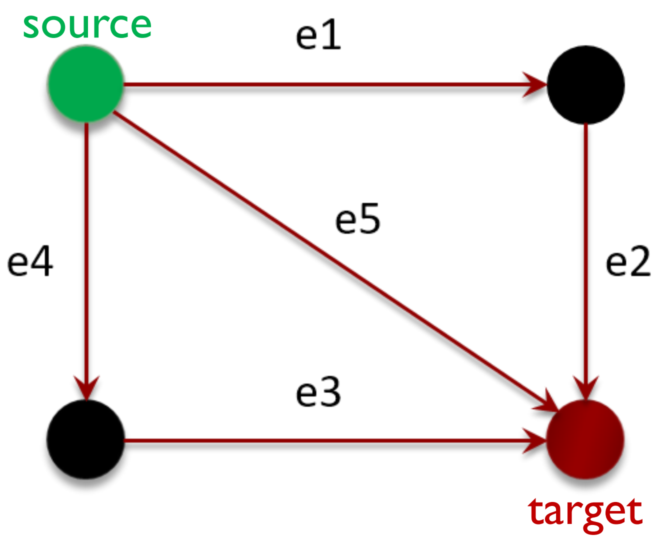
	s->e1->e2->t
e1	0,0

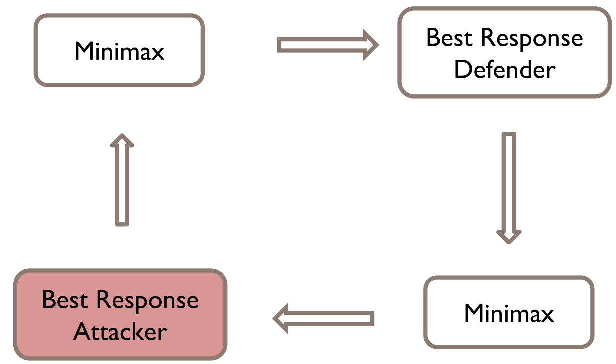
Defender Allocations

Minimax strategy: no change

Example

Minimax strategy:
Defender Strategy: [1.0]
Attacker Strategy: [1.0]





Attacker Paths

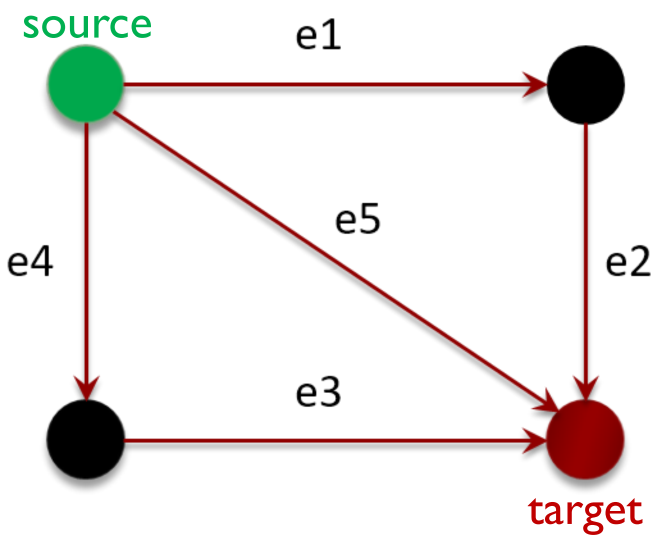
Defender Allocations

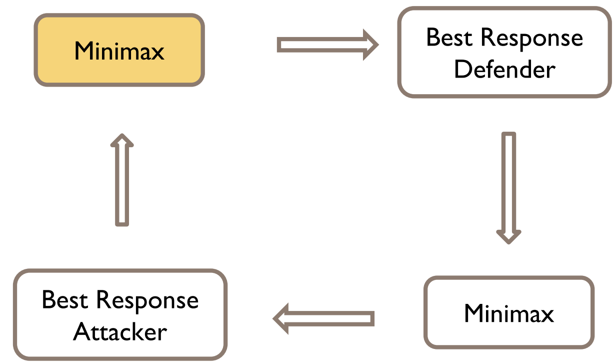
	s->e1->e2->t
e1	0,0

Attacker's best response: s->e4->e3->t or s->e5->t

Pick an arbitrary one, say s->e4->e3->t

Example





Attacker Paths

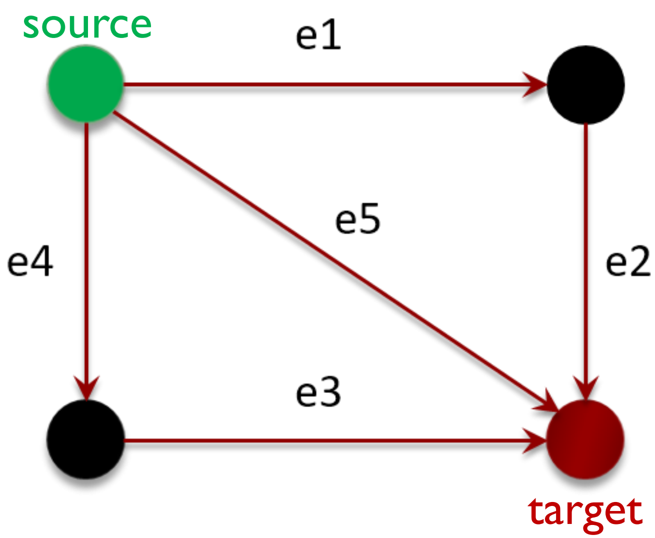
	s->e1->e2->t	s->e4->e3->t
e1		-T,T

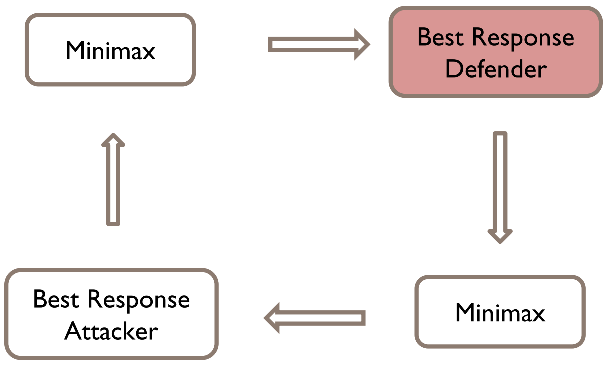
Defender Allocations

Minimax strategy:
Defender Strategy: [1.0]
Attacker Strategy: [0.0, 1.0]

Example

Minimax strategy:
Defender Strategy: [1.0]
Attacker Strategy: [0.0, 1.0]





Attacker Paths

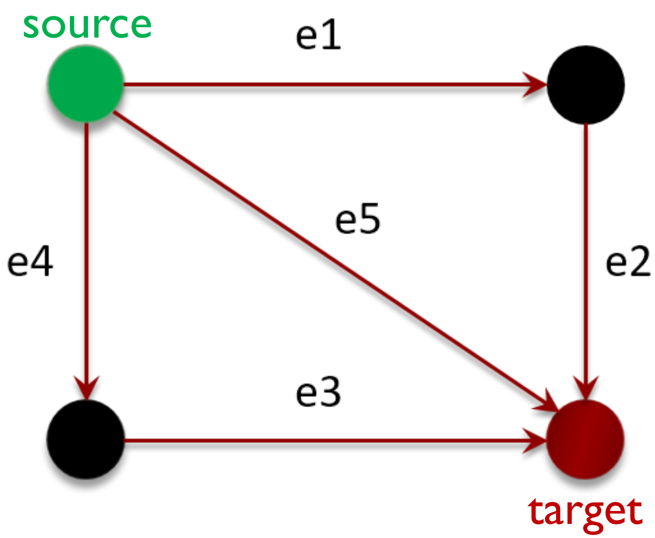
Defender Allocations

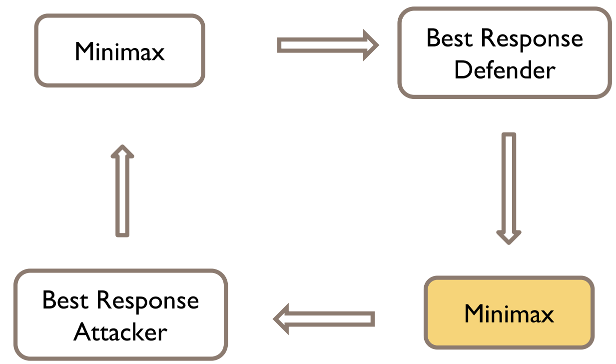
	s->e1->e2->t	s->e4->e3->t
e1		-T,T

Defender's best response: e3 or e4

Pick e3

Example





Attacker Paths

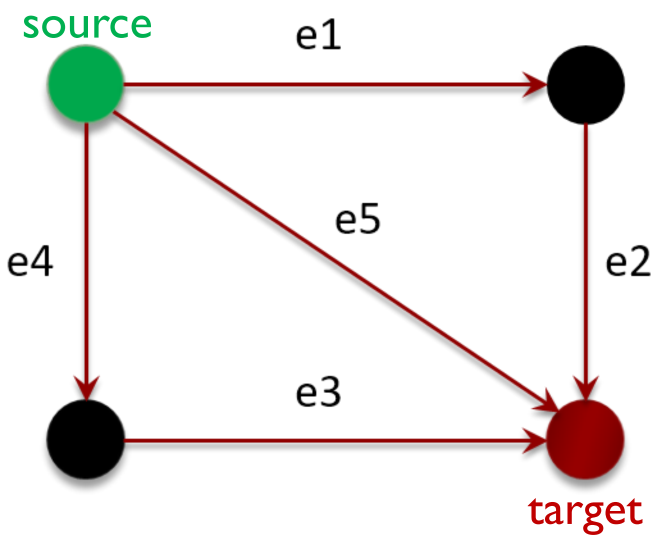
Defender Allocations

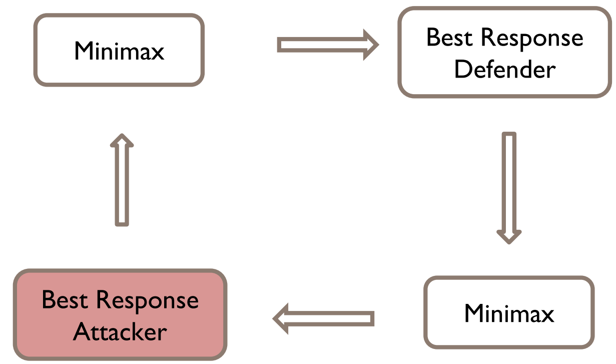
	s->e1->e2->t	s->e4->e3->t
e1		-T,T
e3	-T,T	

Minimax strategy:
Defender Strategy: [0.5, 0.5]
Attacker Strategy: [0.5, 0.5]

Example

Minimax strategy:
Defender Strategy: [0.5, 0.5]
Attacker Strategy: [0.5, 0.5]





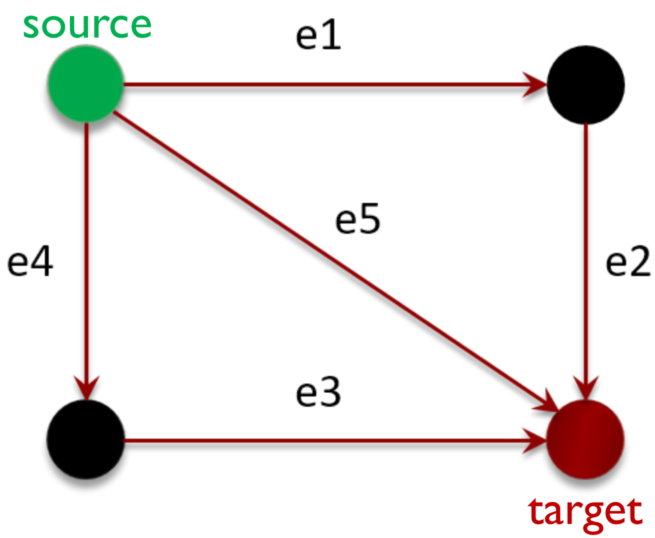
Attacker Paths

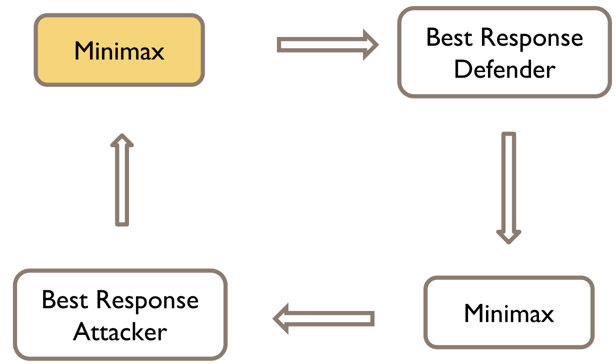
Defender Allocations

	s->e1->e2->t	s->e4->e3->t
e1	0,0	-T,T
e3	-T,T	

Attacker's best response: s->e5->t

Example





Attacker Paths

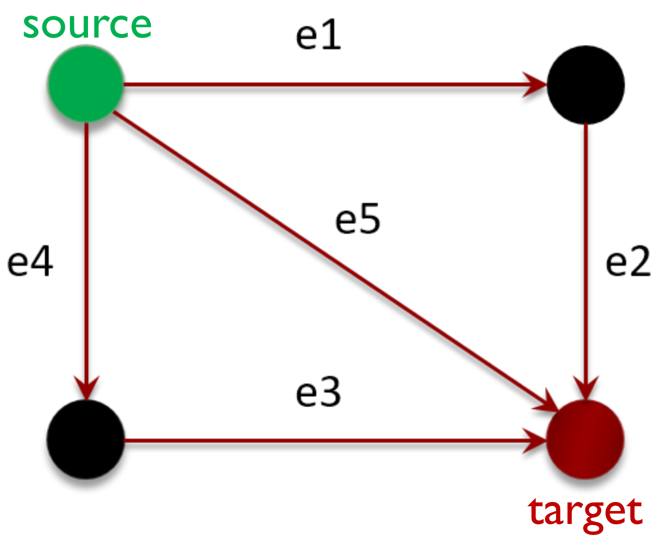
Defender Allocations

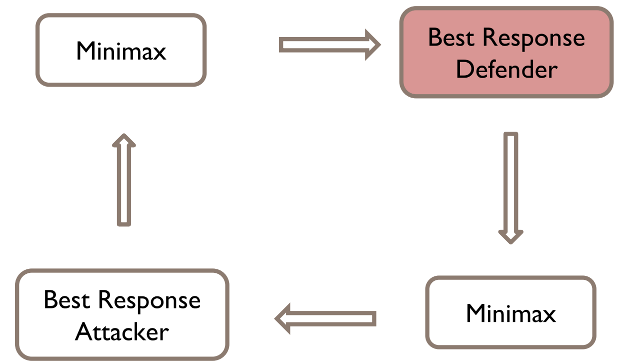
	s->e1->e2->t	s->e4->e3->t	s->e5->t
e1		-T,T	-T,T
e3	-T,T		-T,T

Minimax strategy:
Defender Strategy: arbitrary, say [1.0, 0.0]
Attacker Strategy: [0.0, 0.0, 1.0]

Example

Minimax strategy:
Defender Strategy: [1.0, 0.0]
Attacker Strategy: [0.0, 0.0, 1.0]





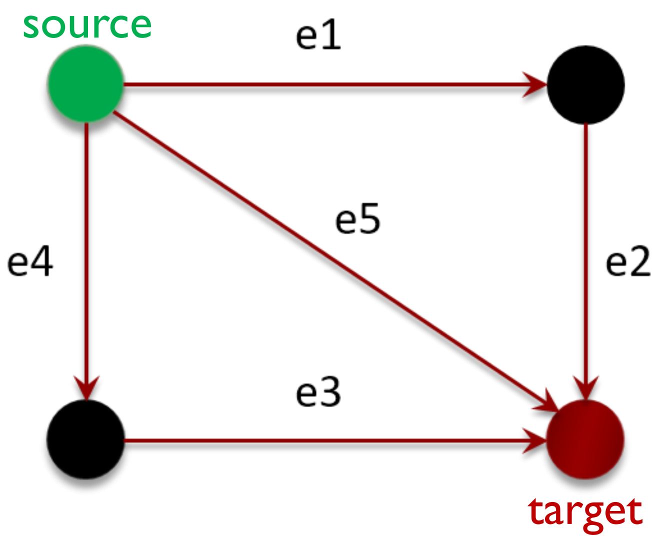
Attacker Paths

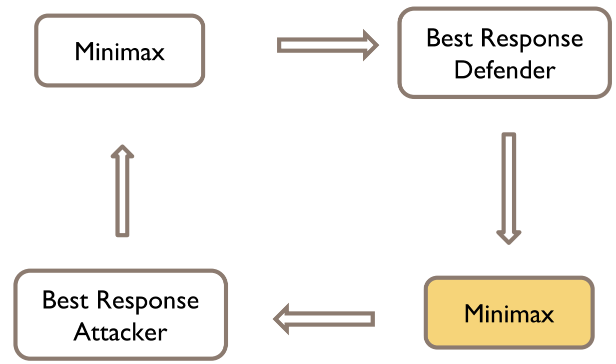
Defender Allocations

	s->e1->e2->t	s->e4->e3->t	s->e5->t
e1		-T,T	-T,T
e3	-T,T		-T,T

Defender's best response: e5

Example





Defender Allocations / **Attacker Paths**

	s->e1->e2->t	s->e4->e3->t	s->e5->t
e1		-T,T	-T,T
e3	-T,T		-T,T
e5	-T,T		-T,T

Defender Strategy: [1/3, 1/3, 1/3]
Attacker Strategy: [1/3, 1/3, 1/3]

No new best responses will be added in the next iteration. Terminate.

Quiz 2

- Assume the following table is the game matrix (zero-sum). At some point in the process of the double oracle algorithm, a smaller game is being considered, with rows 1, 2 and columns 3,4. What action should be added in the next iteration?
- A: A_1
- B: A_2
- C: X_1
- D: X_2
- E: None

Attacker Paths

$$\begin{array}{ll} & \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \text{Defender } X_1: & \begin{bmatrix} -5 & -8 & 0 & -9 \\ 0 & -8 & -15 & 0 \end{bmatrix} \\ \text{Allocations } X_2: & \end{array}$$

Quiz 2

- Assume the following table is the game matrix (zero-sum). At some point in the process of the double oracle algorithm, a smaller game is being considered, with row 1, 2 and column 3,4. What action should be added in the next iteration?
- A_1
- A_2
- X_1
- X_2
- None

The minimax strategy of this smaller game is Def: (5/8, 3/8), Att: (3/8,5/8). Expected utility for attacker of taking each of the action is 5*5/8, 8, 15*3/8, 9*5/8

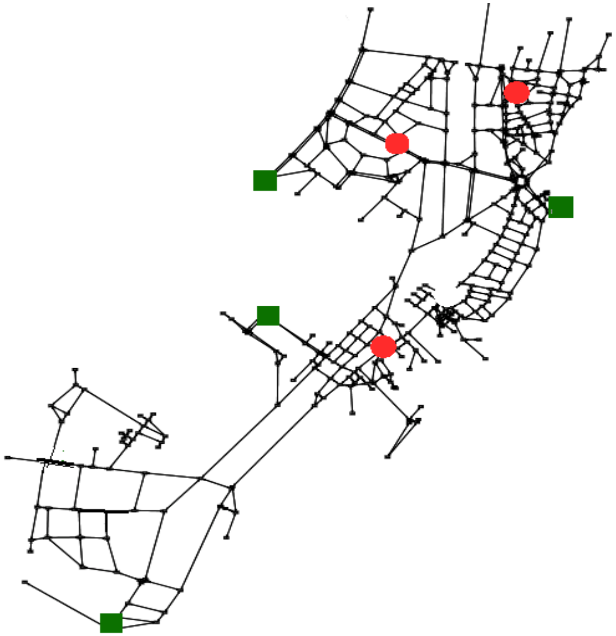
Attacker Paths

Defender Allocations

$$\begin{array}{r|cccc} & A_1 & A_2 & A_3 & A_4 \\ \hline X_1: & -5 & -8 & 0 & -9 \\ X_2: & 0 & -8 & -15 & 0 \end{array}$$

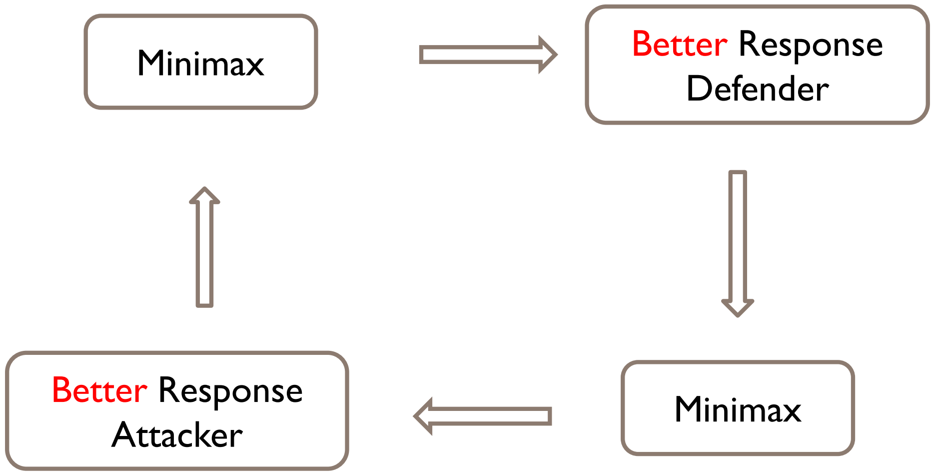
Warm Start

- Initialize with some subset of pure strategies (e.g., for defender, K edges in the min-cut)

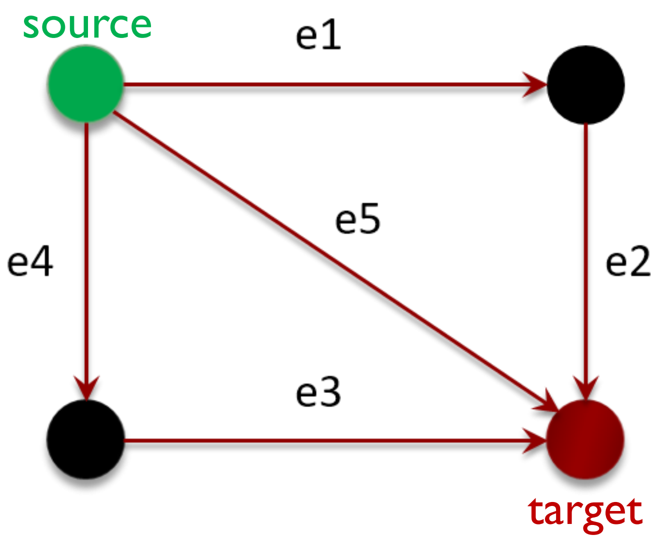


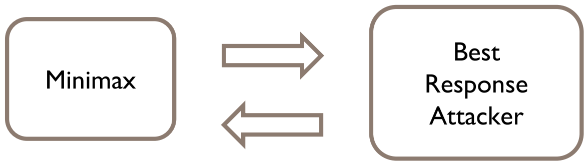
Better Responses

- No need to find the best response
- If you find a better response but not sure if it is the best response, it is OK to add it and move on
- If you cannot find a better response, it means the best response is already in the current support
- Impact on computation time varies



Column Generation: Using One Oracle Only





Attacker Paths

Defender Allocations

	s->e1->e2->t
e1	
e2	
e3	-T,T
e4	-T,T
e5	-T,T

Discussion

- How Machine Learning can potentially be used together with Double Oracle for large-scale zero-sum game solving?

Summary

- Key take-aways
  - Game theory can be used to model security challenges
  - Equilibrium strategies in security games often has a small support
  - Incrementally increase the support size to save time and memory

Additional Resources

- A Double Oracle Algorithm for Zero-Sum Security Games on Graphs;
- An Exact Double-Oracle Algorithm for Zero-Sum Extensive-Form Games with Imperfect Information;
- Double-oracle sampling method for Stackelberg Equilibrium approximation in general-sum extensive-form games

References

- Conitzer, Vincent, and Tuomas Sandholm. "Computing the optimal strategy to commit to." In *Proceedings of the 7th ACM conference on Electronic commerce*, pp. 82-90. 2006.
- McMahan, H. Brendan, Geoffrey J. Gordon, and Avrim Blum. "Planning in the presence of cost functions controlled by an adversary." In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 536-543. 2003.

Backup Slides

Column Generation for Linear Programs

- Column generation is an approach to solving large-scale linear programs with a massive number of variables

- Recall:

$$\max_x c^T x$$
$$\text{s.t. } Ax \le b$$

  - $c \in \mathbb{R}^n$
  - $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$
  - Optimal solution is at a vertex
  - Simplex algorithm: Iteratively move to a neighboring vertex

Column Generation for Linear Programs

- Consider LP in the following form (all LPs can be converted into this form)

$$\max_x c^T x$$
$$\text{s.t. } Ax \le b$$
$$x \ge 0$$

If a variable , say z is unrestricted in the original problem, then introduce two non-negative variables z_+ and z_- substitute z with $z_+ - z_-$

  - $c \in \mathbb{R}^n$
  - $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$

Column Generation for Linear Programs

- If $n \gg m$, many variables will be zero at the optimal solution

  Why? The optimal solution is at a vertex. A vertex in the feasible space (which is a subset of $\mathbb{R}^n$) is determined by n equalities. We can get at most m equalities from boundary hyperplanes of constraints in $Ax \leq b$. So we need to use at least $n - m$ boundary lines of the inequality constraints $x \geq 0$, which means those corresponding variables are 0.

- What if $n \ll m$? Then the dual problem would have a lot of zero-valued variables. We can then try to solve the dual problem using column generation, which is called constraint generation.

Column Generation for Linear Programs

- Column generation: Iteratively solve a main problem and a subproblem
- Main problem: The original LP but with a subset of variables (assuming all other variables are zero)
- Subproblem: Identify a new variable to be added to the subset of variables considered by the main problem

Main Problem

$$\max_{x_i : i \in L} c^T x$$
$$\text{s.t. } Ax \leq b$$
$$x \geq 0$$

Subproblem

Find a new variable x_i and add i to L

Column Generation for Linear Programs

- What is the goal of the subprolem?
- Add a variable that can increase the objective function the most

$$\max_x c^T x \quad \text{s.t. } Ax \le b, \quad x \ge 0$$

Dual LP

$$\min_y b^T y \quad \text{s.t. } A^T y \ge c, \quad y \ge 0$$

- Assume the optimal solution with only a set L of variables considered is x_L^*, the corresponding optimal dual solution is y_L^*
- The new variable chosen, say x_i, should have the highest "reduced cost", calculated as $c_i - A_i^T y_L^*$ where A_i is the ith column of A, i.e., coefficients w.r.t. to x_i. If the highest reduced cost is non-positive, then no variable will be added, x_L^* is the optimal solution of the original problem with all variables

Reduced Cost Explained

- Reduced cost is an important quantity in LP
- First, convert the LP into "canonical form" by adding slack variables $x_{n+1}, \dots, x_{n+m}$

$$\max_{x} c^T x \quad \text{s.t. } Ax \le b \quad x \ge 0$$

$$\begin{aligned} \max_{x_1,\dots,x_{n+m}} \quad & c_1 x_1 + \cdots + c_n x_n \\ \text{s.t. } & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n + x_{n+1} = b_1 \\ & a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n + x_{n+2} = b_2 \\ & \dots \\ & a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n + x_{n+m} = b_m \\ & x_i \ge 0, \forall i \in \{1..n+m\} \end{aligned}$$

- Assume we choose a set of "basic variables" from $\{1..n+m\}$ of size m, called J. Set all variables not in J as 0. The constraints will then be simplified to constraints w.r.t. basic variables only. Then solve this linear system with the m basic variables and m constraints. The solution corresponds to a vertex of the feasible region of the LP in the canonical form shown above. Subselect $x_1, \dots, x_n$ from the solution + the zero-valued non-basic variables lead to a vertex of the feasible region of the original LP.

Reduced Cost Explained

- Formally, denote the new coefficient matrix with slack variables as $\tilde{A} = [A \;\; I], \tilde{c} = \begin{bmatrix} c \\ \mathbf{0} \end{bmatrix}$
- Let $\tilde{A}_J$ be the submatrix of $\tilde{A}$ containing only columns corresponding to variables in J
- Then $x_J = \tilde{A}_J^{-1} b$ and $x_j = 0, \forall j \notin J$ represents a vertex of the feasible region of the following LP

$$\max_{x_1,\ldots,x_{n+m}} \quad c_1 x_1 + \cdots + c_n x_n$$

$$\text{s.t. } a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n + x_{n+1} = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n + x_{n+2} = b_2$$
$$\cdots$$
$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n + x_{n+m} = b_m$$
$$x_i \geq 0, \forall i \in \{1..n+m\}$$

$\Longleftrightarrow$

$$\max_{x \in \mathbb{R}^{n+m}} \tilde{c}^T x$$
$$\text{s.t. } \tilde{A} x = b$$
$$x \geq 0$$

Reduced Cost Explained

- Given $x = (x_1, \dots x_{n+m})$ with $x_J = \tilde{A}_J^{-1} b$ and $x_j = 0, \forall j \notin J$
- Consider adjusting x to x' by setting $x_j' = \alpha > 0$ for some $j \notin J$ while ensuring $x_i' = 0 \ \forall i \notin J, i \neq j$ and $\tilde{A}x' = b, x' \geq 0$, i.e., introducing one variable to the current basic variable set
- All $x_i, i \in J$ has to change accordingly
- Denote $x_J' = x_J + \alpha d_J$, then

$$
\begin{gathered}
\tilde{A}x' = b \Rightarrow \tilde{A}_J(x_J + \alpha d_J) + \alpha \tilde{A}_j = b \\
\Rightarrow \tilde{A}_J\left(\cancel{\tilde{A}_J^{-1} b} + \alpha d_J\right) + \alpha \tilde{A}_j \cancel{= b} \\
\Rightarrow \alpha \tilde{A}_J d_J + \alpha \tilde{A}_j = 0 \\
\Rightarrow d_J = -\tilde{A}_J^{-1} \tilde{A}_j
\end{gathered}
$$

Reduced Cost Explained

$$\max_x c^T x \quad \text{s.t. } Ax \le b \quad x \ge 0$$

$$\min_y b^T y \quad \text{s.t. } A^T y \ge c \quad y \ge 0$$

- If $j \in [1..n]$, the new objective value is

$$f(x') = \tilde{c}^T x' = \tilde{c}^T x + \alpha(\tilde{c}_j + \tilde{c}_J^T d_J)$$

- Rewritten as $f(x') = \tilde{c}^T x + \alpha \bar{c}_j$ where

$$\bar{c}_j = \tilde{c}_j + \tilde{c}_J^T d_J = \tilde{c}_j - \tilde{c}_J^T \tilde{A}_J^{-1} \tilde{A}_j$$

Therefore $f(x') > \tilde{c}^T x$ if $\bar{c}_j > 0$

For $j \in \{1..n\}, \bar{c}_j$ is called *reduced cost*

Reduced Cost Explained

$$f(x') = \tilde{c}^T x + \alpha \bar{c}_j$$

$$\bar{c}_j = \tilde{c}_j - \tilde{c}_J^T \tilde{A}_J^{-1} \tilde{A}_j$$

- If $\bar{c}_j$ is non-positive for all non-basic variables of a vertex corresponding to basic variable set J, then the vertex is the optimal solution
- If $\bar{c}_j$ is positive for some j, then moving from x to x' can lead to a higher objective value, the higher the value of $\bar{c}_j$, the higher the increase rate. The Simplex algorithm move towards the neighboring vertex with the highest $\bar{c}_j$

Reduced Cost Explained

- If $x^* \in \mathbb{R}^{n+m}$ is the optimal solution of the primal LP in canonical form, and it corresponds to a set of basis J, then consider the corresponding optimal dual solution $y^* \in \mathbb{R}^m$
  - According to complementary slackness, if x_j is in J, then the corresponding dual constraint is tight, i.e., $A_j^T y^* = c_j$ if $j \in \{1..n\}$ and $y_{j-n}^* = 0$ if $j \in \{n+1, \dots, n+m\}$
- Together with the fact $\tilde{A} = [A \;\; I], \tilde{c} = \begin{bmatrix} c \\ \mathbf{0} \end{bmatrix}$, we have $\tilde{A}_J^T y^* = \tilde{c}_J$
- We can conclude: at optimal solution, $\bar{c}_j = \tilde{c}_j - \tilde{c}_J^T \tilde{A}_J^{-1} \tilde{A}_j$ can be rewritten as $\bar{c}_j = c_j - A_j^T y^*$ for $j \in \{1..n\}$

Reduced Cost Explained

- Assume that after you solved an LP and get x^* and the corresponding y^*, you are asked to add a new variable x_j to the LP with coefficient c_j and matrix column A_j
- x^* still corresponds to a vertex in the augmented LP, but it may not be the optimal solution
- We need to check if we introduce j to the basis, whether the objective value will increase
- This can be done by directly checking the reduced cost

Subproblem and Reduced Cost

- Now consider the column generation process.
- It can be viewed as add variables one by one.
- Again, whether and how much a new variable x_j will improve the objective value depends on its reduced cost, computed as $c_i - A_i^T y_L^*$ where y_L^* is the optimal dual solution (without slack variables) before x_j is added

Double Oracle

- Double oracle is similar to applying column generation to the primal and dual problem of the minimax LP with alternation

$$\max_{x,v} v$$
$$\text{s.t. } v \le \textstyle\sum_i x_i U_{ij}, \forall j$$
$$\sum_i x_i = 1$$
$$x_i \ge 0$$

$$\begin{array}{cc} & \begin{array}{ccccc} A_1 & A_2 & A_3 & A_4 & \dots \end{array} \\ \begin{array}{c} X_1: \\ X_2: \\ \vdots \end{array} & \begin{bmatrix} -5 & -8 & 0 & -9 & \dots \\ 0 & -8 & -15 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \end{array}$$

x_i: prob of for row X_i