Game Theoretic Foundations of Multiagent Systems: Algorithms and Applications

A case study: Playing Games for Security

Nicola Basilico
Department of Computer Science
University of Milan

A short introduction to Security Games

Nicola Basilico
Department of Computer Science
University of Milan

Introduction

- **Intelligent security** for physical infrastructures

  - *Our objective*: provide protection to physical environments with many targets against threats.

  - *Our means*: security resources.

  - *Our constraints*: resources are limited, targets are many

Introduction

- What's the challenge for a computer scientist?

- Design an intelligent system where autonomous agents are capable of providing **protection** against possible threats**:**
  - Detection: localize a threat;
  - Response: neutralize it.

- A strategy prescribes and describes what agents should do or would do:
  - *How to assign limited resources to defend targets?*
  - *What's the worst case damage that can be done in the environment when adopting some given strategy?*

- **Computing and characterizing effective strategies is a scientific/technological challenge**

Literature Overview

- Involved scientific communities include:

- Search Theory
  - **Contact investigation**: Stone and Stanshine, J. App. Math, 1971
  - **Search with false contacts**: Dobbie, Operations Research, 1973
- Operations Research
  - **Index policies for patrol**: Lin et al., Operations Research, 2013
- Game Theory
  - **Search Games:** Gal and Alpern, Int. Series in OR & Management Science, 2003
  - **Security Games**: Basilico and Gatti, Artificial Intelligence, 2012

Foundations

- Robotics
  - **Algorithmic queueing theory**: Bullo et al., IEEE Proceedings, 2011
  - **Variable resolution patrolling**: Basilico and Carpin, ICRA, 2012
  - **Live-fly validation of sensor model**: Carpin et al., Journal of Field Robotics, 2013

Applications

Literature Overview

- Research can be roughly divided into two paradigms, depending on the kind of threat one assumes to face:

- **Strategic:** the threat is the output of a rational decision maker usually called adversary. The adversary can observe, learn and plan before deciding how to attack. *(Example: terrorists)*

- **Non-Strategic:** the threat is the output of a stochastic process described under probabilistic laws. *(Example: wildfires)*

Game Theory

John von Neumann

John Nash

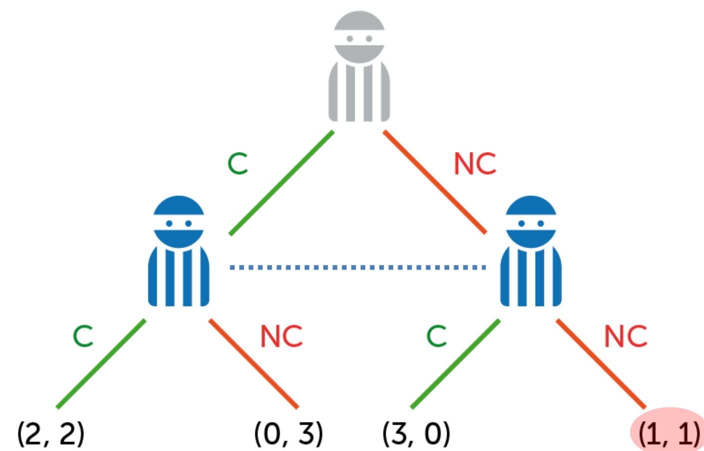
- Game Theory provides elegant mathematical frameworks to describe interactive decision making in multi-agent systems
- Applications: economics, business, political science, biology, psychology, law, urban planning …
- It gives tools to define what intelligent and rational decision makers would do (solution concepts)
- The most popular solution concept: Nash Equilibrium (NE)

The Prisoner’s Dilemma

Strategic (normal) form

	C	NC
C	(2, 2)	(0, 3)
NC	(3, 0)	(1, 1)

Extensive form



- A **strategy profile** tells the probability with which each player plays some action
- Nash Equilibrium strategy profile: no player unilaterally deviates from its strategy
- How to use this formalism for security scenarios?

Security Games

Museum (value = 2)

Bank (value = 5)

Security Games

Museum (value = 2)

Bank (value = 5)

Defender:
its objective is to protect some areas

Attacker:
its objective is to compromise some area without being detected by the defender;

Security Games

Museum (value = 2)

Bank (value = 5)

Defender:
its objective is to protect some areas

Attacker:
its objective is to compromise some area without being detected by the defender;

Attacker

Defender	bank	museum
bank	7 -1	0 2
museum	0 5	7 -1

Security Games

		Attacker: bank	Attacker: museum
Defender	bank	7 -1	0 2
	museum	0 5	7 -1

Nash Equilibrium:

D = {0.67; 0.33}, A = {0.5; 0.5}

What if the attacker can wait, observe, and **then** strike?

Security Games

		Attacker: bank	Attacker: museum
Defender	bank	7 -1	0 2
	museum	0 5	7 -1

Nash Equilibrium:

D = {0.67; 0.33}, A = {0.5; 0.5}

What if the attacker can wait, observe, and **then** strike?

The attacker can gain a correct belief about the strategy of the Defender. What does this entail?

Security Games

		Attacker: bank	Attacker: museum
Defender	bank	7 -1	0 2
Defender	museum	0 5	7 -1

Nash Equilibrium:

D = {0.67; 0.33}, A = {0.5; 0.5})

What if the attacker can wait, observe, and **then** strike?

Leader-Follower scenario

- The defender declares: “I’ll go to the bank”: commitment to **D = {1; 0}** (observability)

Security Games

Defender \ Attacker	bank	museum
bank	7 -1	0 2
museum	0 5	7 -1

Nash Equilibrium:

D = {0.67; 0.33}, A = {0.5; 0.5})

What if the attacker can wait, observe, and **then** strike?

Leader-Follower scenario

- The defender declares: “I’ll go to the bank”: commitment to **D = {1; 0}** (observability)

- The game has a trivial solution in pure strategies: **D = {1; 0}**, **A = {0; 1}** with payoffs (**0**,**2**)

Security Games

Defender \ Attacker	bank	museum
bank	7 -1	0 2
museum	0 5	7 -1

Nash Equilibrium:

D = {0.67; 0.33}, A = {0.5; 0.5})

What if the attacker can wait, observe, and **then** strike?

Leader-Follower scenario

- The defender declares: "I'll go to the bank": commitment to **D = {1; 0}** (observability)
- The game has a trivial solution in pure strategies: **D = {1; 0}**, **A = {0; 1}** with payoffs (**0**,**2**)
- What's the best strategy to commit to?
  - It's never worse than a NE [Von Stengel and Zamir, 2004]
  - At the equilibrium the attacker always plays in pure strategies [Conitzer and Sandholm, 2006]

Example

	C	D
A	(5,1)	(1,0)
B	(6,2)	(-1,5)

- Let's suppose that, before the game begins, **L** makes the following announcement:

$$\sigma_L = \begin{cases} A & \frac{3}{4} \\ B & \frac{1}{4} \end{cases}$$

Example

	C	D
A	(5,1)	(1,0)
B	(6,2)	(-1,5)

- Let's suppose that, before the game begins, **L** makes the following announcement:

$$\sigma_L = \begin{cases} A & \frac{3}{4} \\ B & \frac{1}{4} \end{cases}$$

If I play C, then $\mathrm{E}[u_F(\sigma_L, C)] = \frac{3}{4} \cdot 1 + \frac{1}{4} \cdot 2 = \frac{5}{4}$

If I play D, then $\mathrm{E}[u_F(\sigma_L, D)] = \frac{3}{4} \cdot 0 + \frac{1}{4} \cdot 5 = \frac{1}{4}$

Example

	C	D
A	(5,1)	(1,0)
B	(6,2)	(-1,5)

- Let's suppose that, before the game begins, **L** makes the following announcement:

$$\sigma_L = \begin{cases} A & \frac{3}{4} \\ B & \frac{1}{4} \end{cases}$$

If I play C, then $E[u_F(\sigma_L, C)] = \frac{3}{4} \cdot 1 + \frac{1}{4} \cdot 2 = \frac{5}{4}$

If I play D, then $E[u_F(\sigma_L, D)] = \frac{3}{4} \cdot 0 + \frac{1}{4} \cdot 5 = \frac{1}{4}$

I will play C

Example

	C	D
A	(5,1)	(1,0)
B	(6,2)	(-1,5)

If I announce $\sigma_L = \begin{cases} A & \frac{3}{4} \\ B & \frac{1}{4} \end{cases}$ the follower will play C
and I will get $\mathrm{E}[u_L(\sigma_L, C)] = \frac{3}{4} \cdot 5 + \frac{1}{4} \cdot 6 = \frac{21}{4}$

Example

	C	D
A	(5,1)	(1,0)
B	(6,2)	(-1,5)

If I announce $\sigma_L = \begin{cases} A & \frac{3}{4} \\ B & \frac{1}{4} \end{cases}$ the follower will play C and I will get $\mathrm{E}[u_L(\sigma_L, C)] = \frac{3}{4} \cdot 5 + \frac{1}{4} \cdot 6 = \frac{21}{4}$

The best σ_L^* to announce is ?

Leader follower equilibrium (LFE)

Properties of LFE

The follower does **not** randomize: it chooses the action that maximizes its expected utility. *If indifferent between one or more actions, it will break ties in favor of the leader (compliant follower)* [Conitzer and Sandholm, 2006]

⇓

fundamental property for computing the solution concept

LFE is never worse than any NE [Von Stengel and Zamir, 2004]

Computing a NE (recall)

- Zero-sum games: linear program [von Neumann, 1920]
- General-sum games: no linear programming formulation is possible
- With two agents:
  - Linear complementarity problem [Lemke and Howson, 1964]
  - Support enumeration (multi LP) [Porter, Nudelman, and Shoham, 2004]
  - MIP Nash [Sandholm, Giplin, and Conitzer, 2006]
- With more than two agents?
  - Non-linear complementarity programming
  - Other methods
- Complexity:
  - The problem is in NP
  - It is not NP-Complete unless P=NP, but complete w.r.t. PPAD (which is contained in NP and contains P) [Papadimitrou, 1991] [Chen, Deng, 2005] [Daskalakis, 2006]
  - Commonly believed that no efficient algorithm exists

Computing a LFE

- Zero sum games: linear programming

- General sum games:
  - Multiple linear programs (a polynomial number in the worst case) [Conitzer and Sandholm, 2006]
  - Alternative MILP formulations [Paruchuri, 2008]

Computing a LFE (general sum)

Idea:

1. For each action **b** of the Follower:
   - Find the best commitment C(**b**) to announce, given that **b** will be the action played by **F**

2. Select the best C(**b**)

Computing a LFE (general sum)

Idea:

1. For each action **b** of the Follower:
   - Find the best commitment C(**b**) to announce, given that **b** will be the action played by **F**

2. Select the best C(**b**)

Step 1

$$\max \sum_{a \in A_L} \sigma_L(a) u_L(a,b) \quad \text{s.t.}$$

$$\sum_{a \in A_L} \sigma_L(a) u_F(a,b) \geq \sum_{a \in A_L} \sigma_L(a) u_F(a,b') \quad \forall b' \in A_F \qquad \Rightarrow \qquad \sigma^*_{L,b}$$

$$\sum_{a \in A_L} \sigma_L(a) = 1$$

$$\sigma_L(a) \geq 0 \quad \forall a \in A_L$$

Computing a LFE (general sum)

Step 2:

$$\sigma_L^* = \sigma_{L,b^*}$$

$$b^* = \arg\max_{b \in A_F} \sum_{a \in A_L} \sigma_{L,b}(a) u_L(a, b)$$

- We need to solve a LP n times, where n is the number of actions for the Follower
- For zero-sum games: maxmin strategy
- (For multiple followers and uncertain types of followers the problem becomes harder.)

Does it really work?

LAX checkpoints and canine units (2007)

Boston coast guard (2011)

Federal Air Marshals (2009)

LAX security (2008): terminals

- **Targets**
  - 8 terminals
- **Defender resources**
  - Canine units
- **1-hour unit time**
- **Different types of attackers**

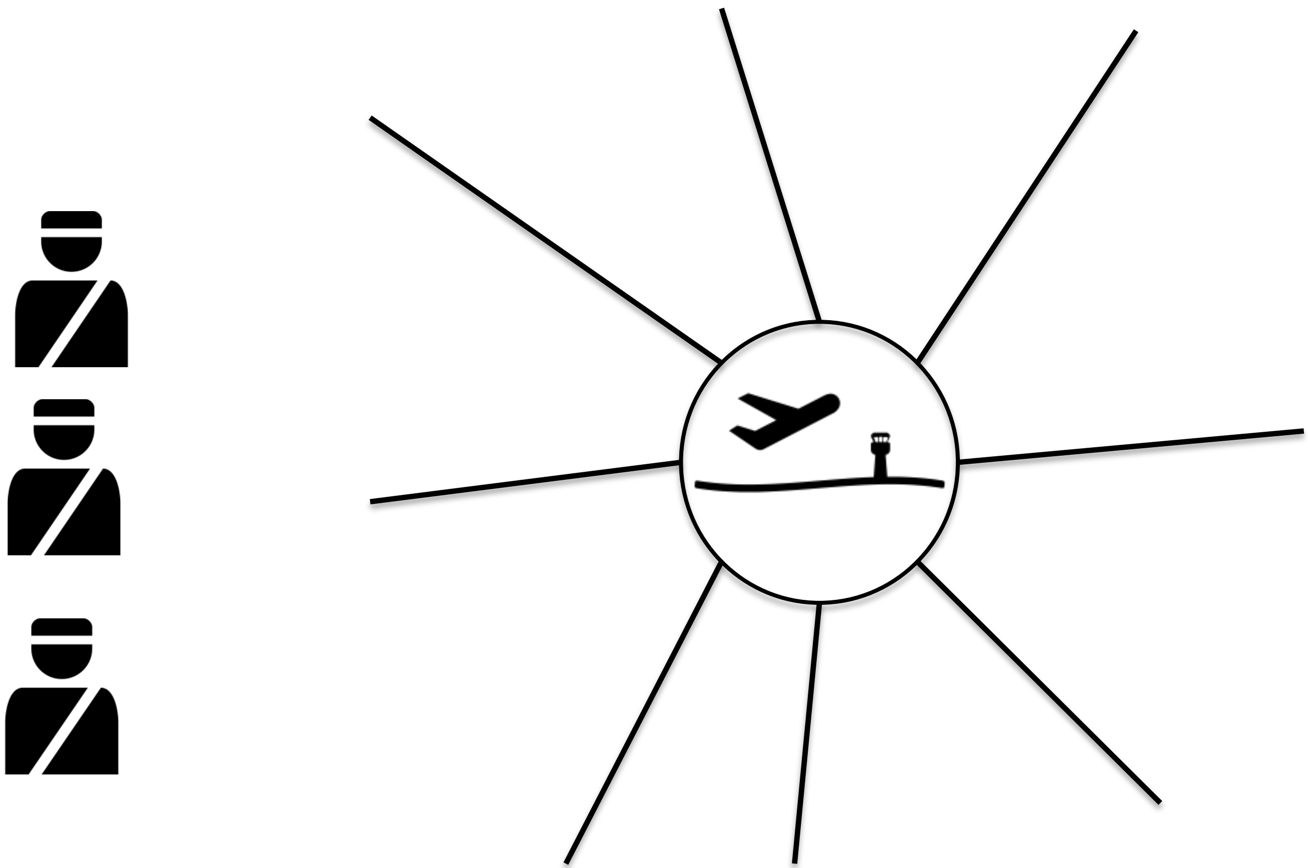
LAX security (2008): terminals

	Terminal 1			Terminal 2			Terminal 3			Terminal 4			…			Terminal 8		
05:00-06:00	0.1	0.3	0.2	0.2	0.3	0.4	0.3	0.1	0.1	0.1	0.1	0.1	…	…	…	0.2	0.1	0.1
06:00-07:00	0.1	0.3	0.2	0.2	0.3	0.4	0.3	0.1	0.1	0.1	0.1	0.1	…	…	…	0.2	0.1	0.1
07:00-08:00	0.1	0.1	0.2	0.1	0.1	0.2	0.3	0.1	0.2	0.2	0.2	0.2	…	…	…	0.2	0.2	0.1
08:00-09:00	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
10:00-11:00	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
11:00-12:00	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
12:00-13:00	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
	U1	U2	U3	U1	U2	U3	U1	U2	U3	U1	U2	U3	U1	U2	U3	U1	U2	U3

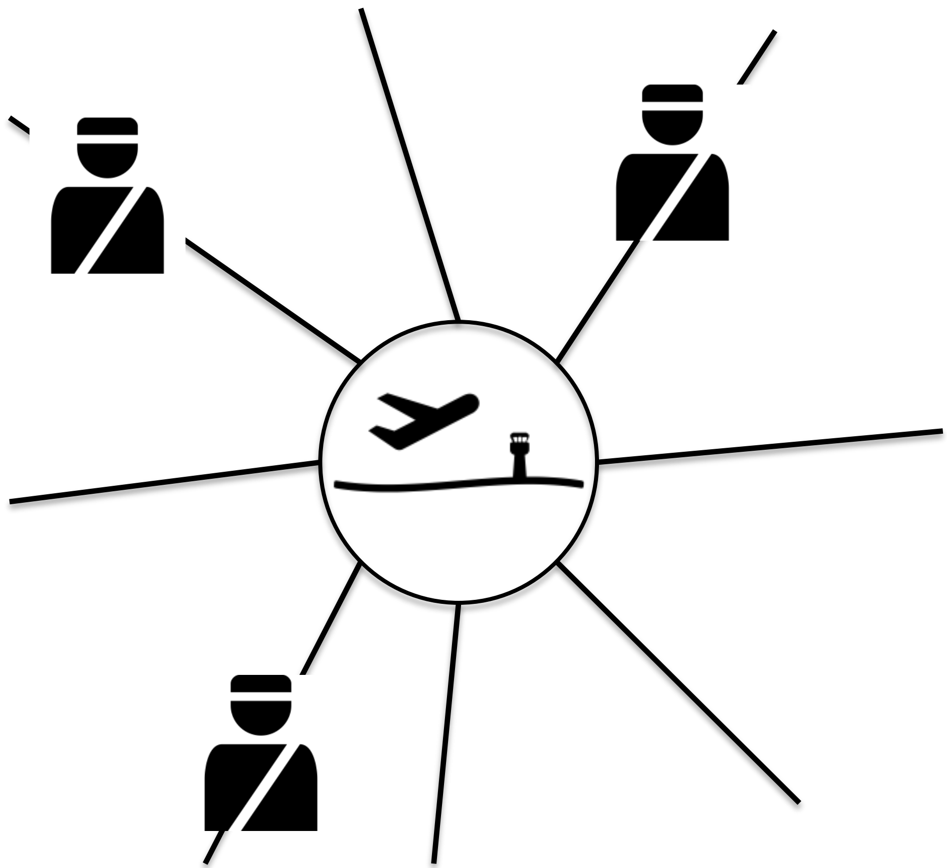
LAX security (2008): checkpoints

- **Targets**
  - Roads leading to the airport
- **Defender resources**
  - Checkpoints
- **1-hour unit time**
- **Different types of attackers**

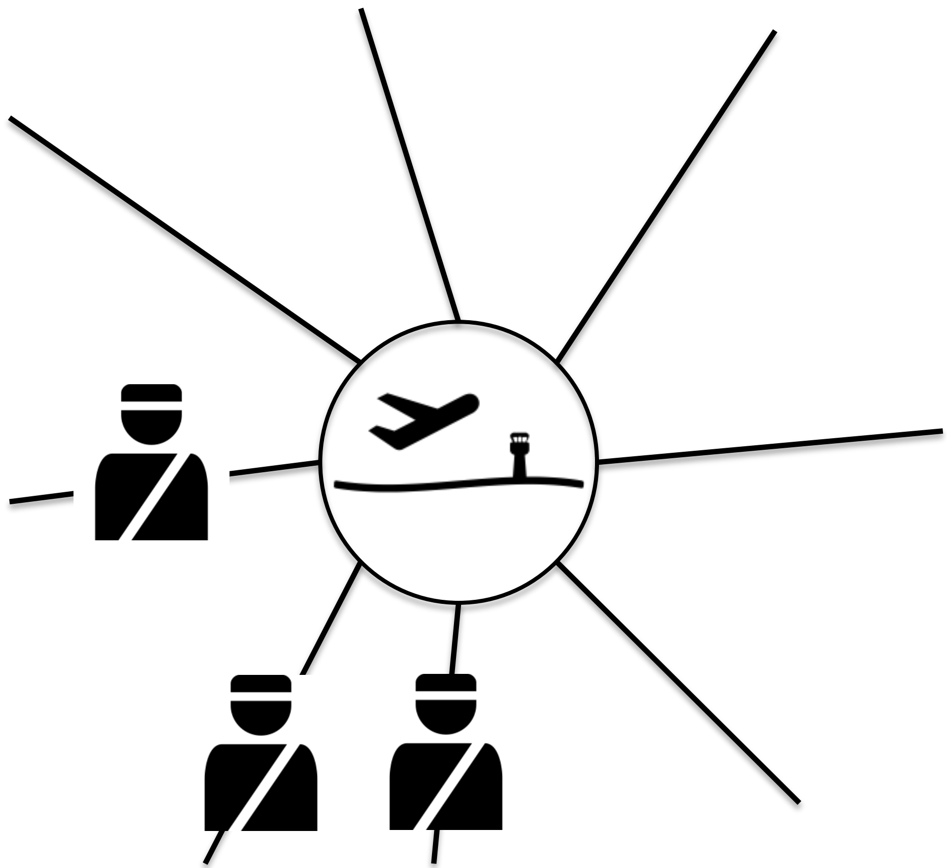
LAX security (2008): checkpoints



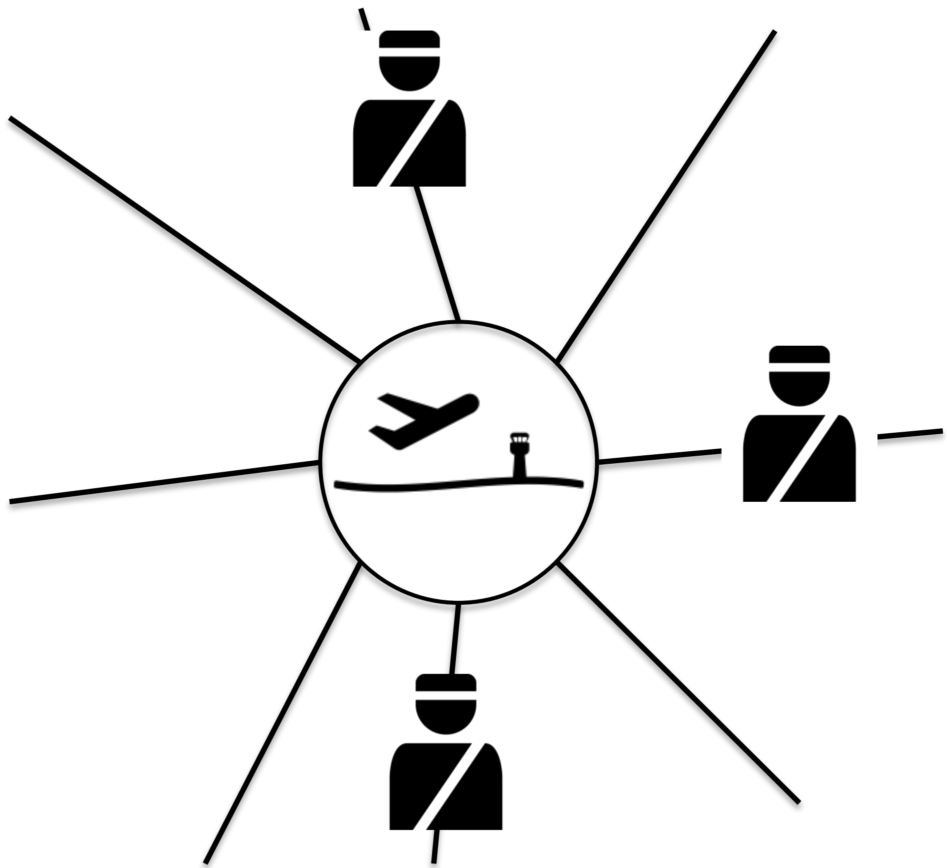
LAX security (2008): checkpoints



LAX security (2008): checkpoints



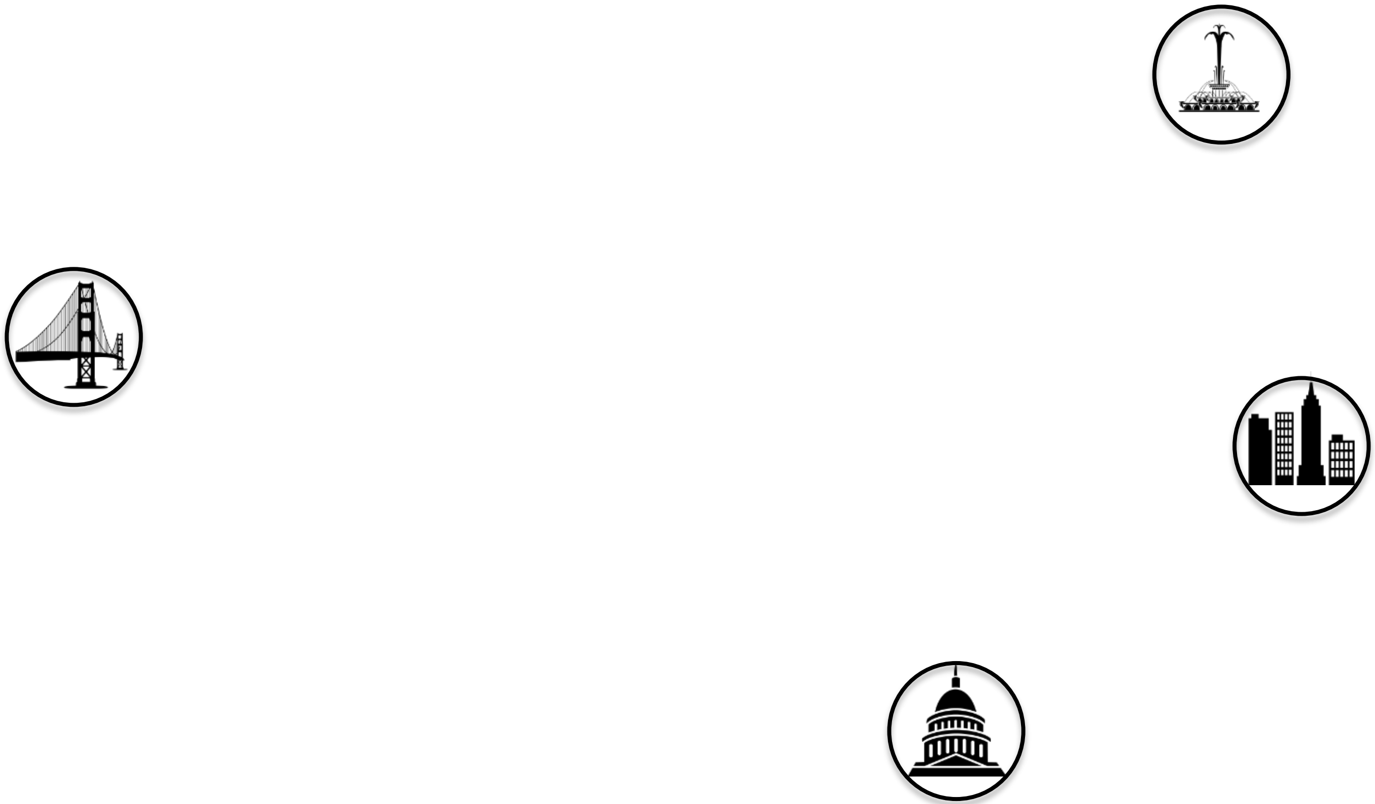
LAX security (2008): checkpoints



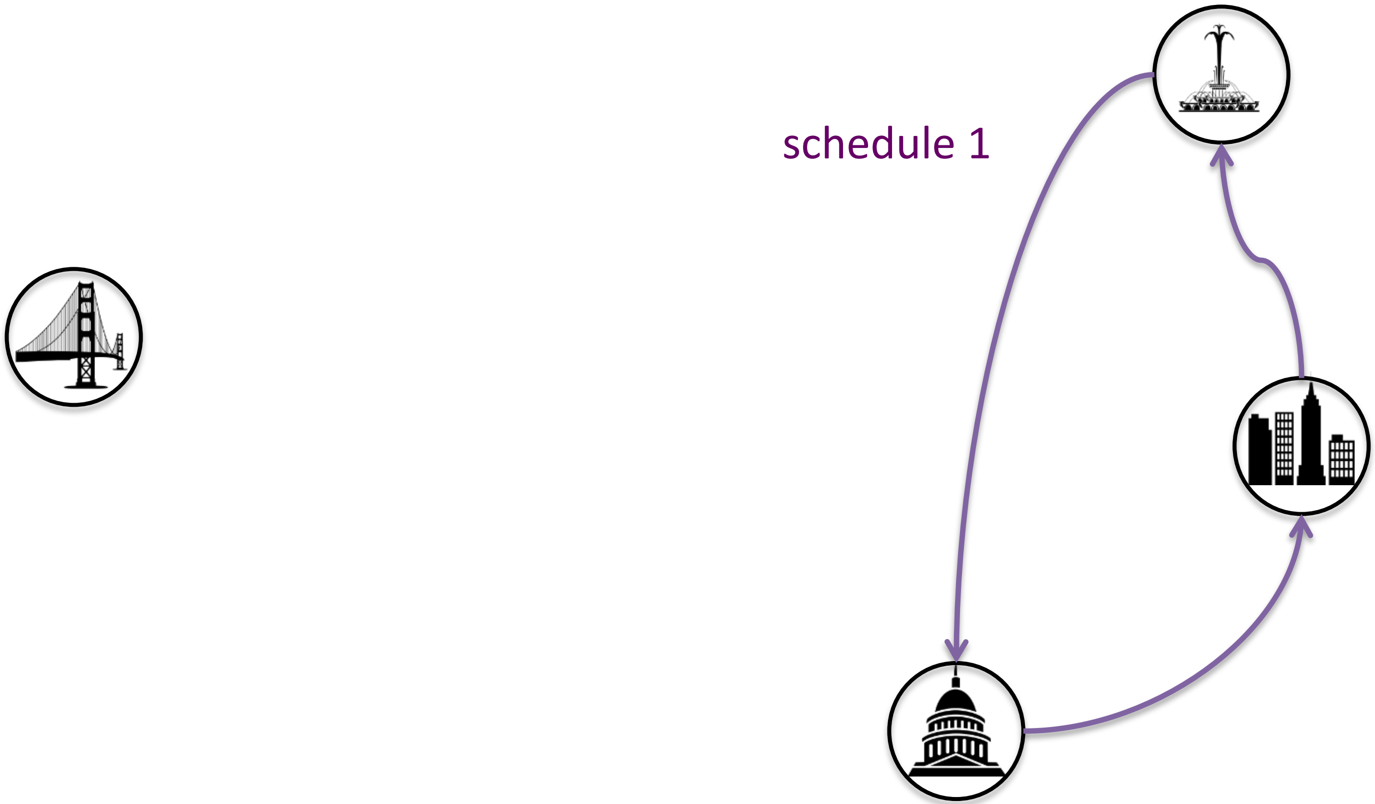
Federal air marshals (2009)

- **Targets**
  - Domestic flight (about 29,000 per day)
- **Defender resources**
  - Federal air marshals (3,000 per day)
- **1-day unit time**
- **Resources constraints**
  - Each marshal starting from a city must conclude the schedule at the same city
  - Each flight has a minimum number of resources to secure it
- **Unique type of attacker**

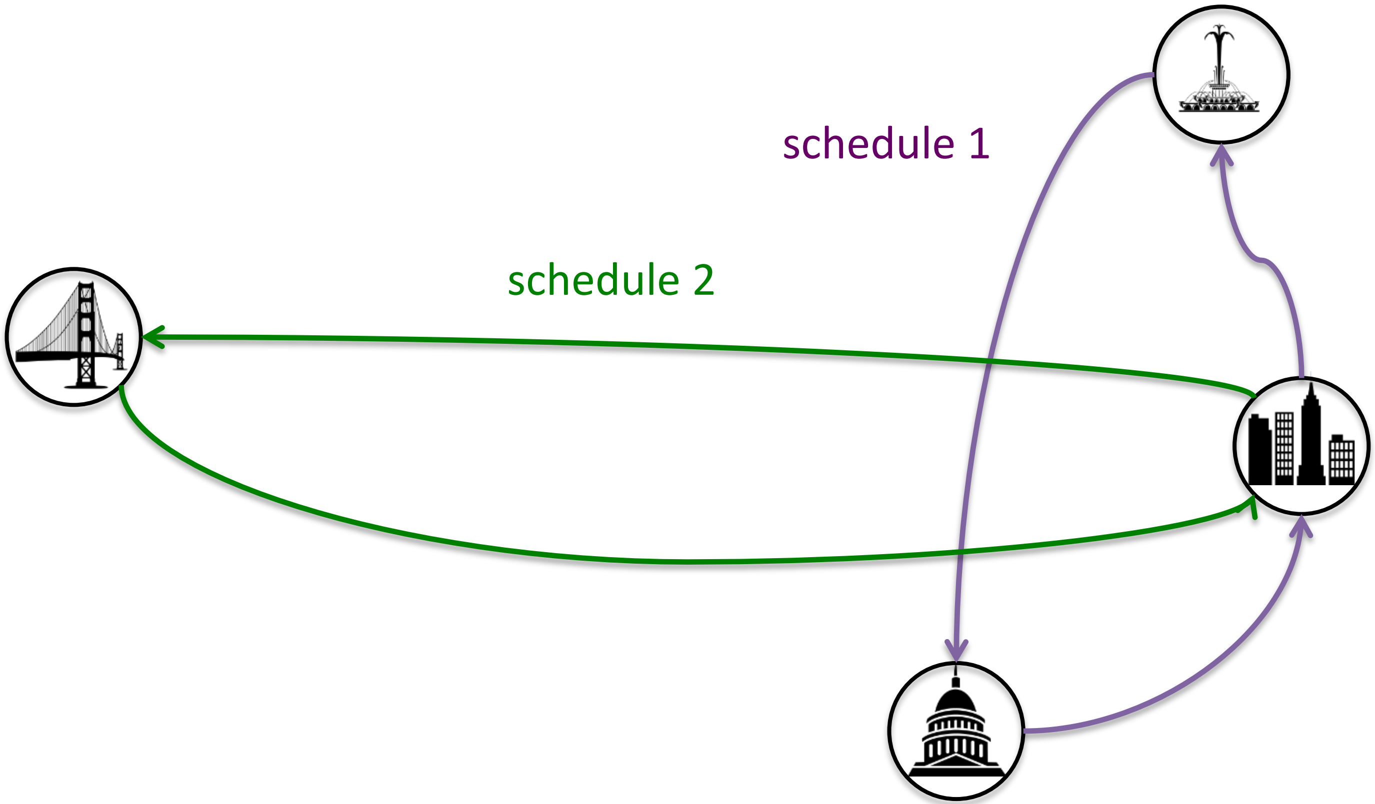
Federal air marshals (2009)



Federal air marshals (2009)



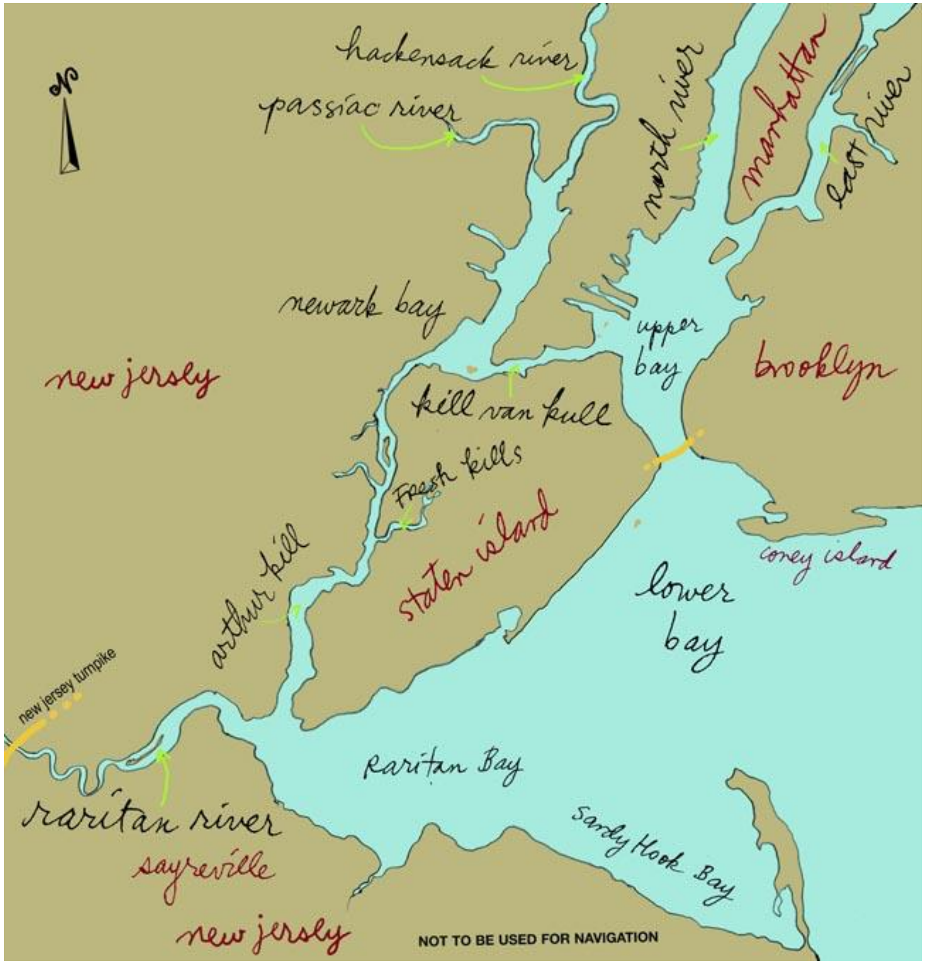
Federal air marshals (2009)



Federal air marshals (2009)

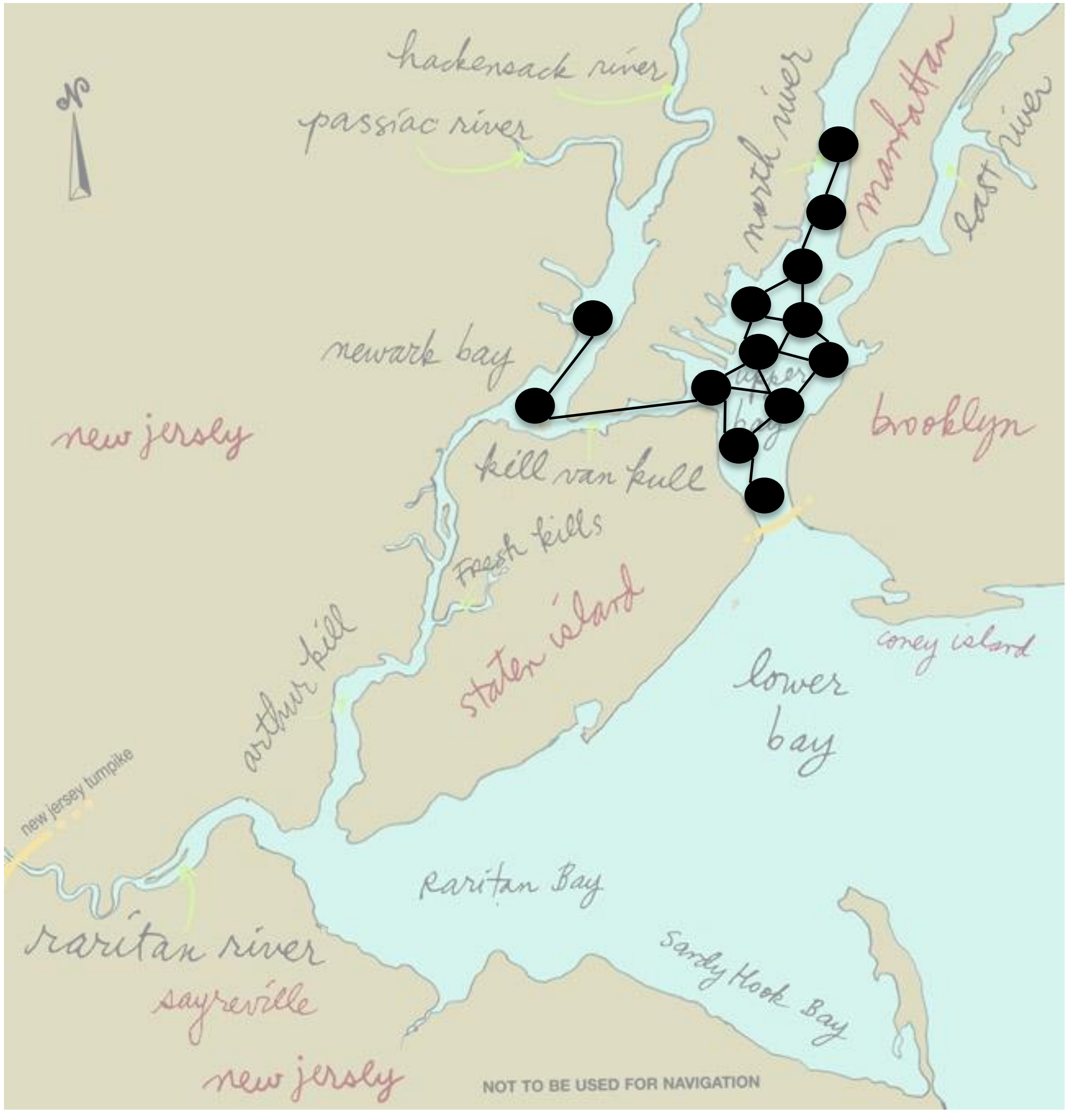
schedule 1	0.1	0.3	0.2	0.2	0.3	0.4	0.3	0.1	0.1	0.1	0.1	0.1	…	…	…	…	…	…
schedule 2	0.1	0.3	0.2	0.2	0.3	0.4	0.3	0.1	0.1	0.1	0.1	0.1	…	…	…	…	…	…
schedule 3	0.1	0.1	0.2	0.1	0.1	0.2	0.3	0.1	0.2	0.2	0.2	0.2	…	…	…	…	…	…
schedule 4	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
schedule 5	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
schedule 6	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
schedule 7	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
	A01	A02	A03	A04	A05	A06	A07	A08	A09	A10	A11	A12	A13	A14	A15	A16	A17	…

New York City area bay (2010)



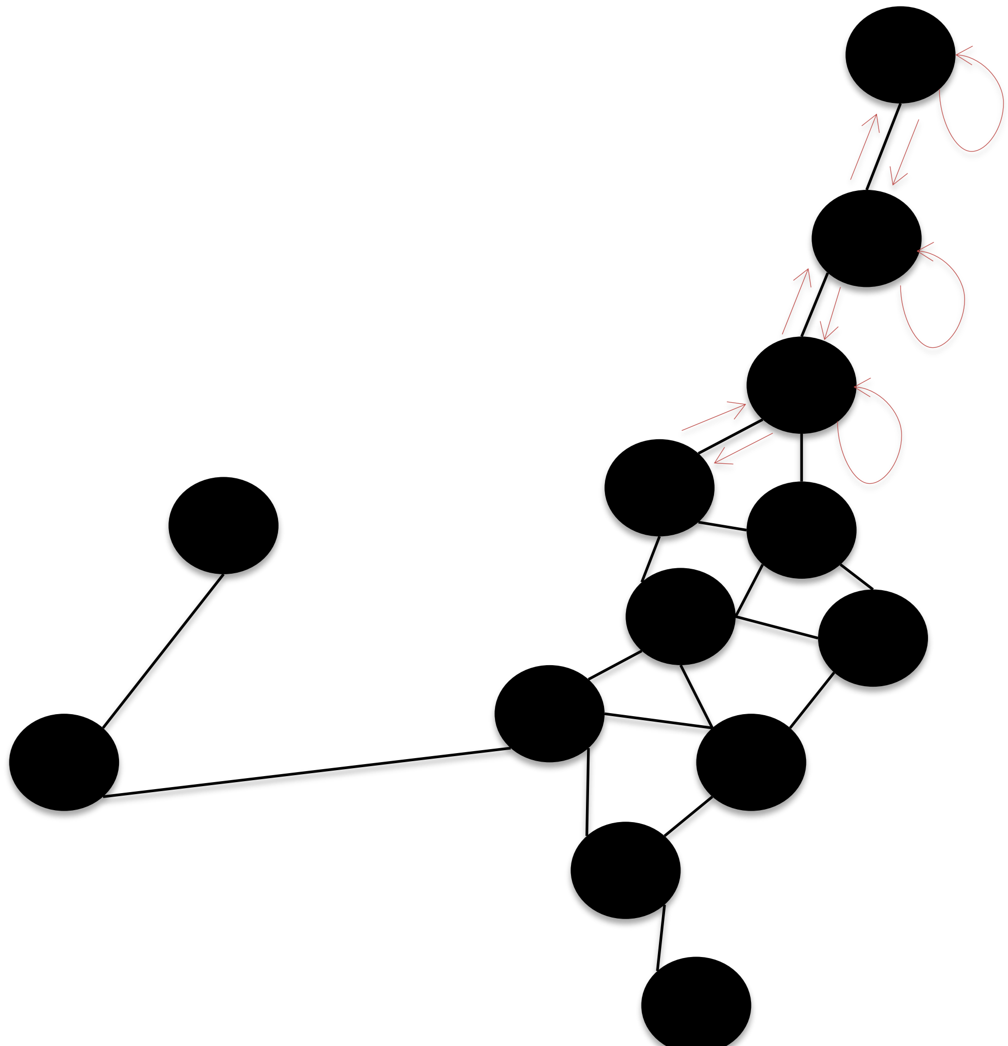
Scheduling
guard coast units

New York City area bay (2010)



Scheduling
guard coast units

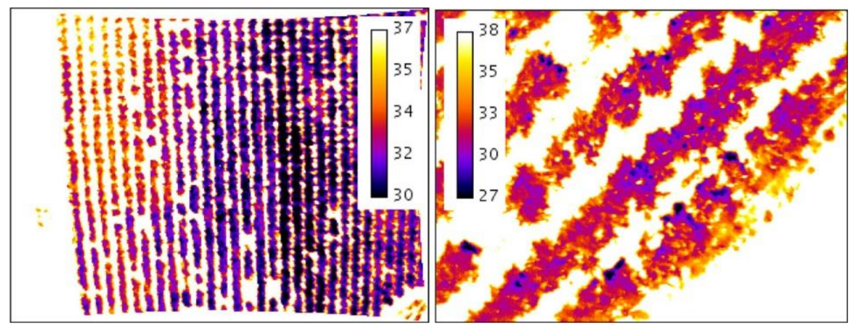
New York City area bay (2010)



Our Scenario

- We assume to have an environment extensively covered with sensors (continuous spatially distributed sensing)
- Examples:

Forests



Agriculture fields

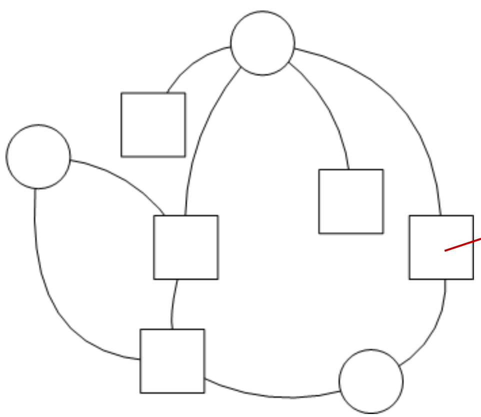
- These scenarios can require surveillance on **two levels:**
  - **Broad area level:** sensors tells that something is going on in some area (spatial uncertain readings);
  - **Local investigation level:** agents should be dispatched over the “hot” area to find out what is going on.

Adversarial Patrolling with Spatially Uncertain Alarms

The Basic Model

- Idea: a game theoretical setting where the Defender is supported by an alarm system installed in the environment

- Environment: undirected graph



Target t:
- *v(t)* value
- *d(t)* penetration time: time units needed to complete an attack during which capture can happen

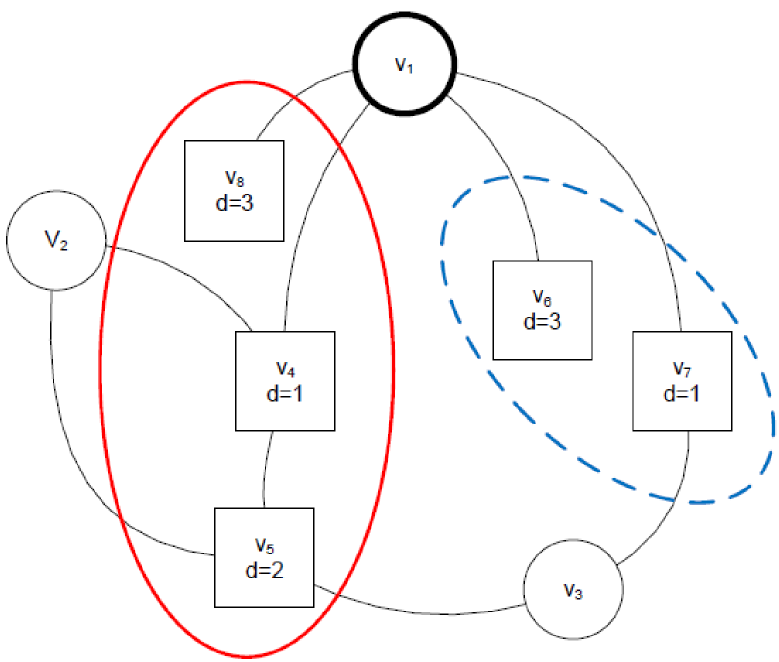
- At any stage of the game:

The Defender decides where to go next

The Attacker decides whether to attack a target or to wait

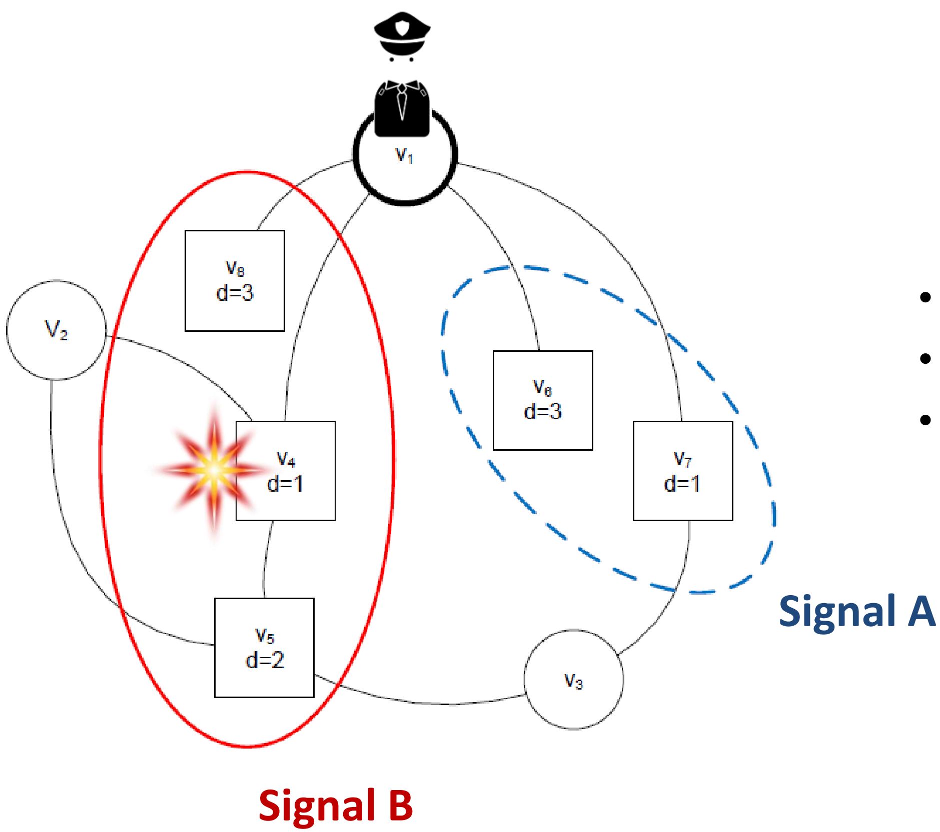
The Alarm System

- Each attack at a target t probabilistically generates a signal that is sent to the Defender
- If the Defender receives a signal it must do something (Signal Response Game)
- Otherwise it must normally patrol the environment (Patrolling Game)



Example (deterministic):
If an attack is present on tagets {8,4,5} generate B
If an attack is present on tagets {6,7} generate A

The Alarm System

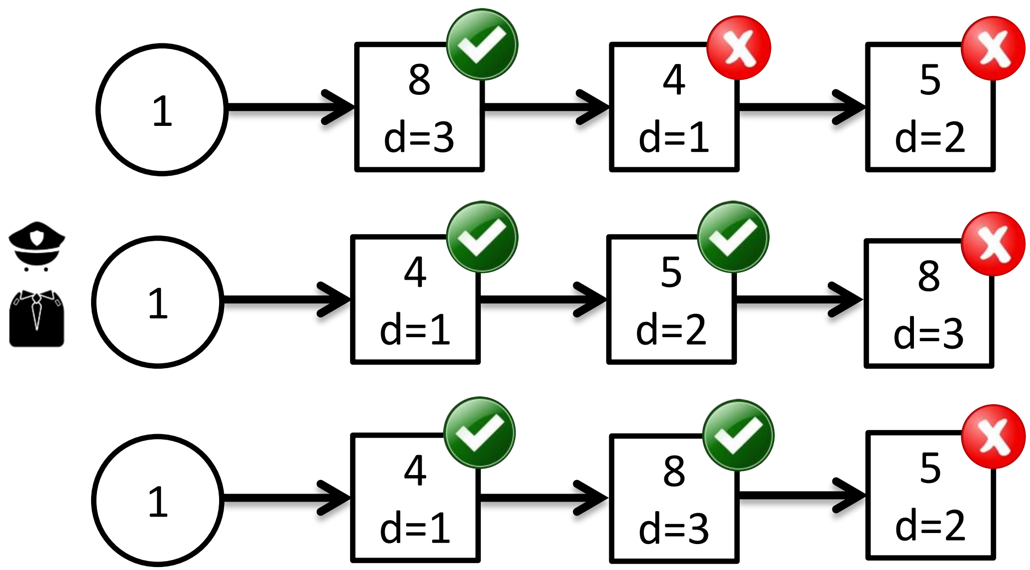


- The Defender is in 1
- The Attacker attacks 4
- The Alarm system generates with prob. 1 **signal B**

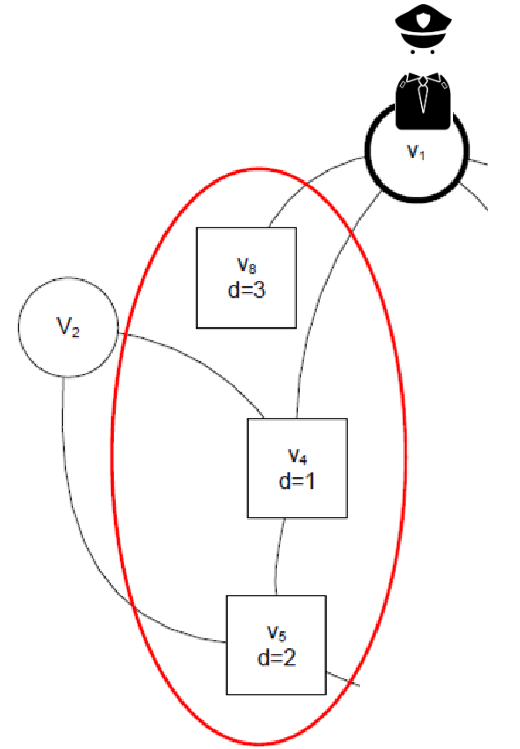
The Alarm System

- Upon receiving the signal, the Defender knows that the Attacker is in 8, 4, or 5

- In principle, it should check each target no later than d(t)



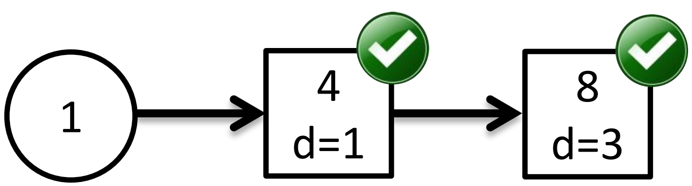
Covering routes



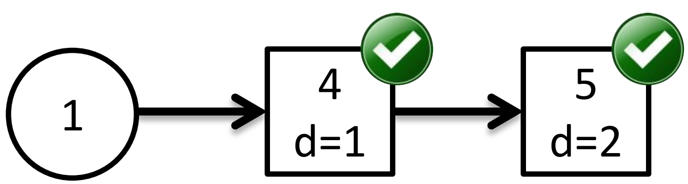
The Alarm System

- Covering routes: a permutation of targets which specifies the order of first visits (covering shortest paths) such that each target is first-visited before its deadline

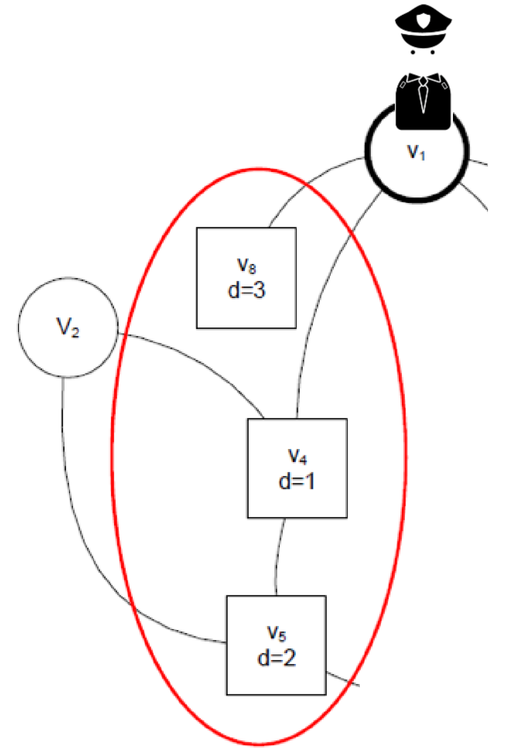
- Example



Covering route: **<4,8>**

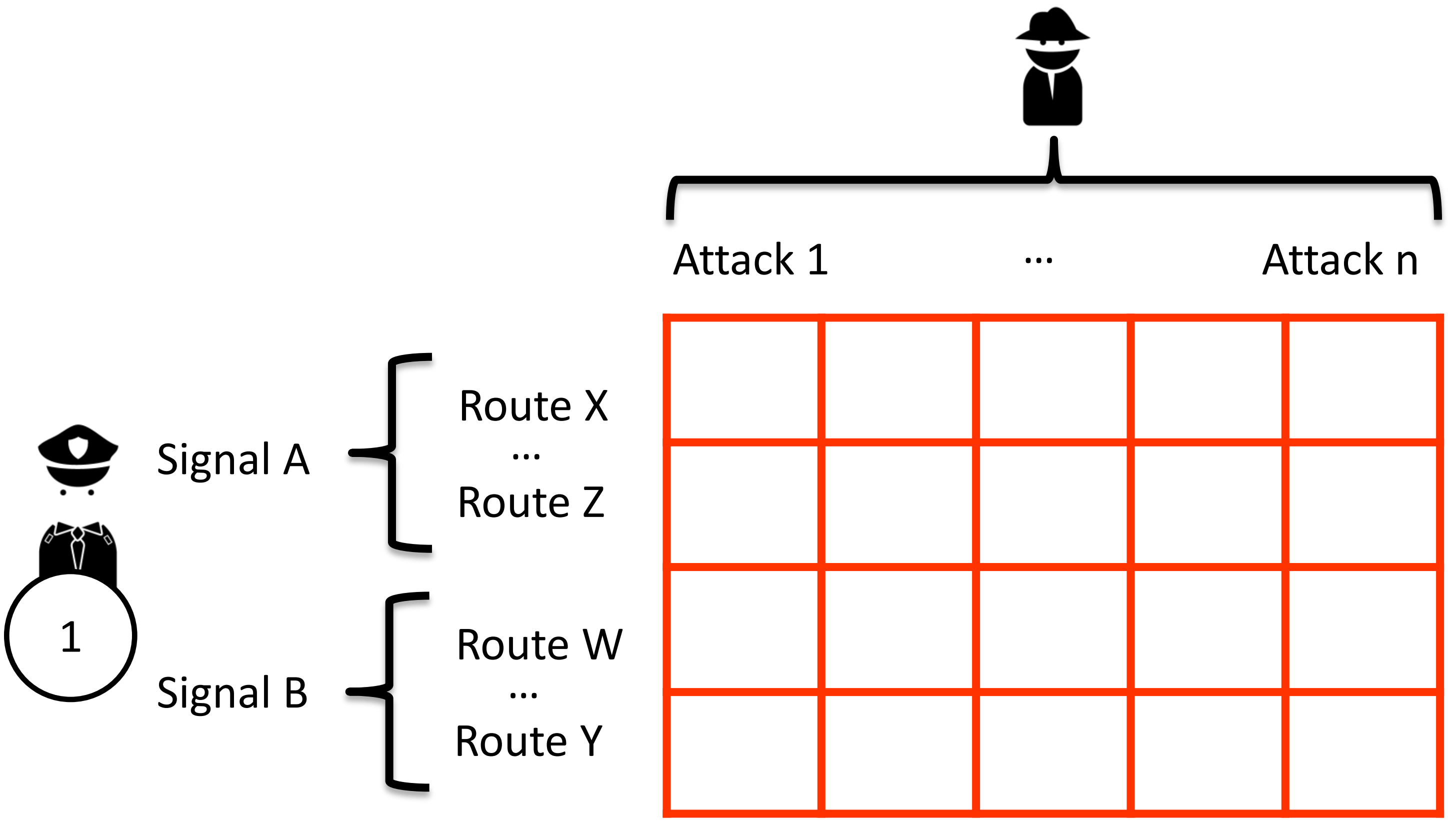


Covering route: **<4,5>**



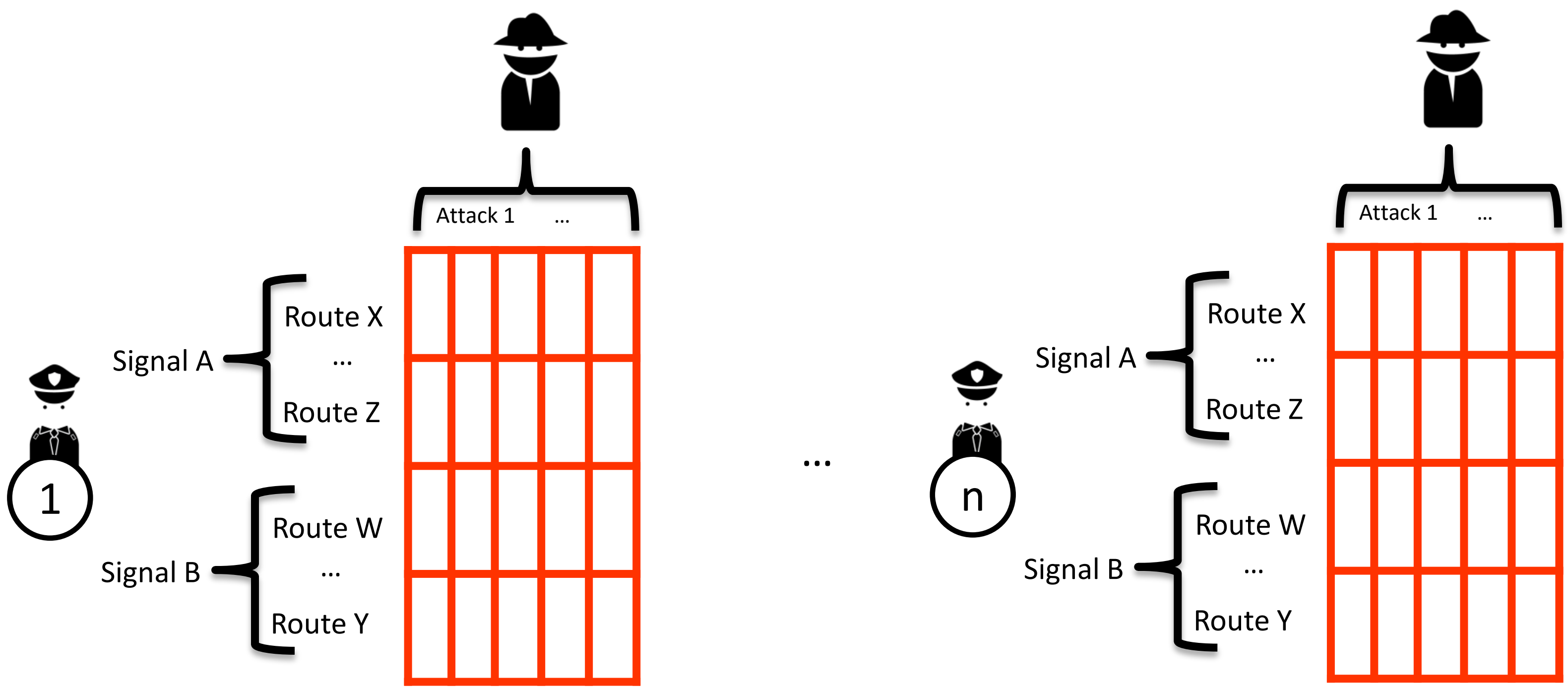
The Signal Response Game

- We can formulate the game in strategic (normal form), for vertex 1



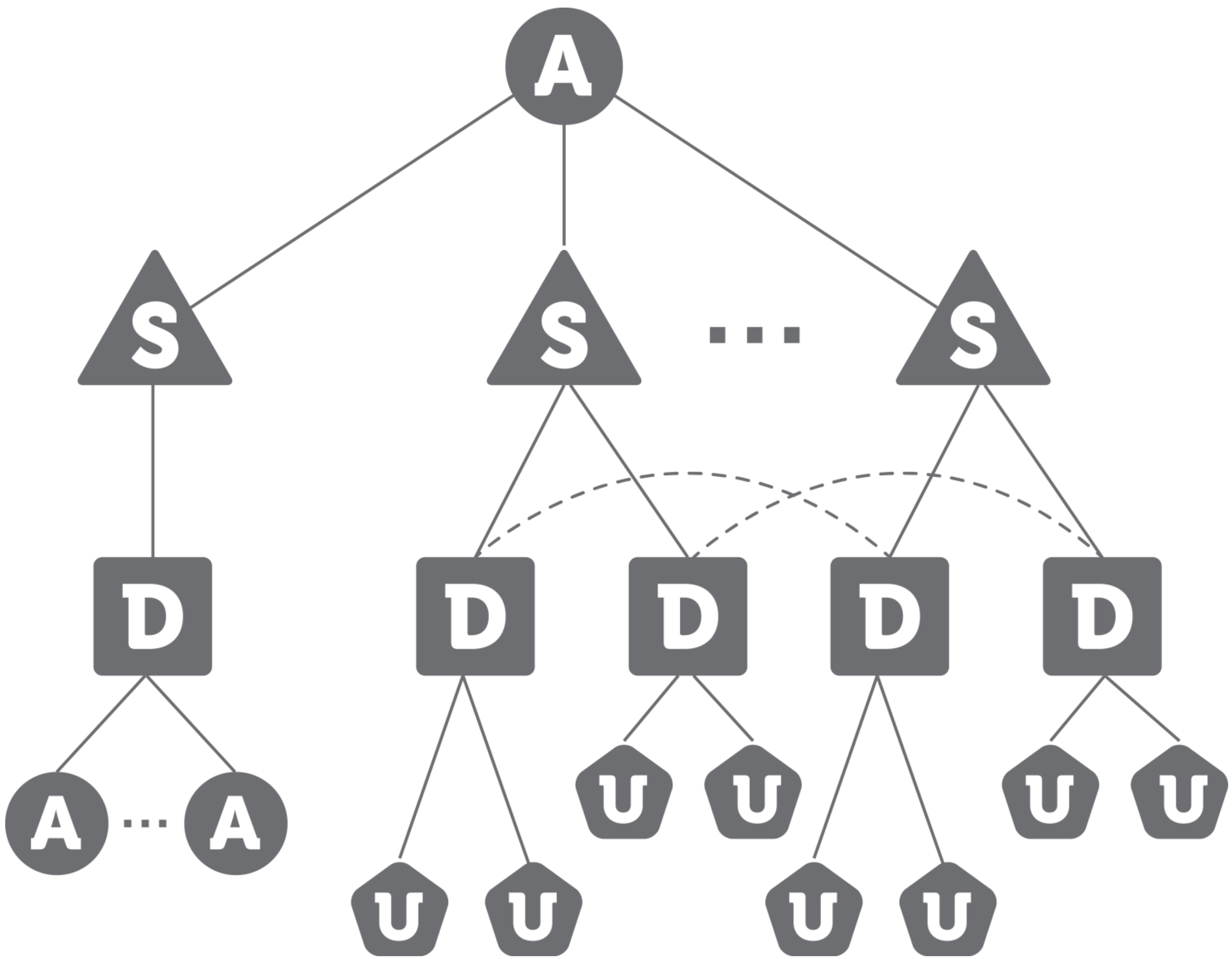
The Signal Response Game

- We can formulate the game in strategic (normal form), for all vertices

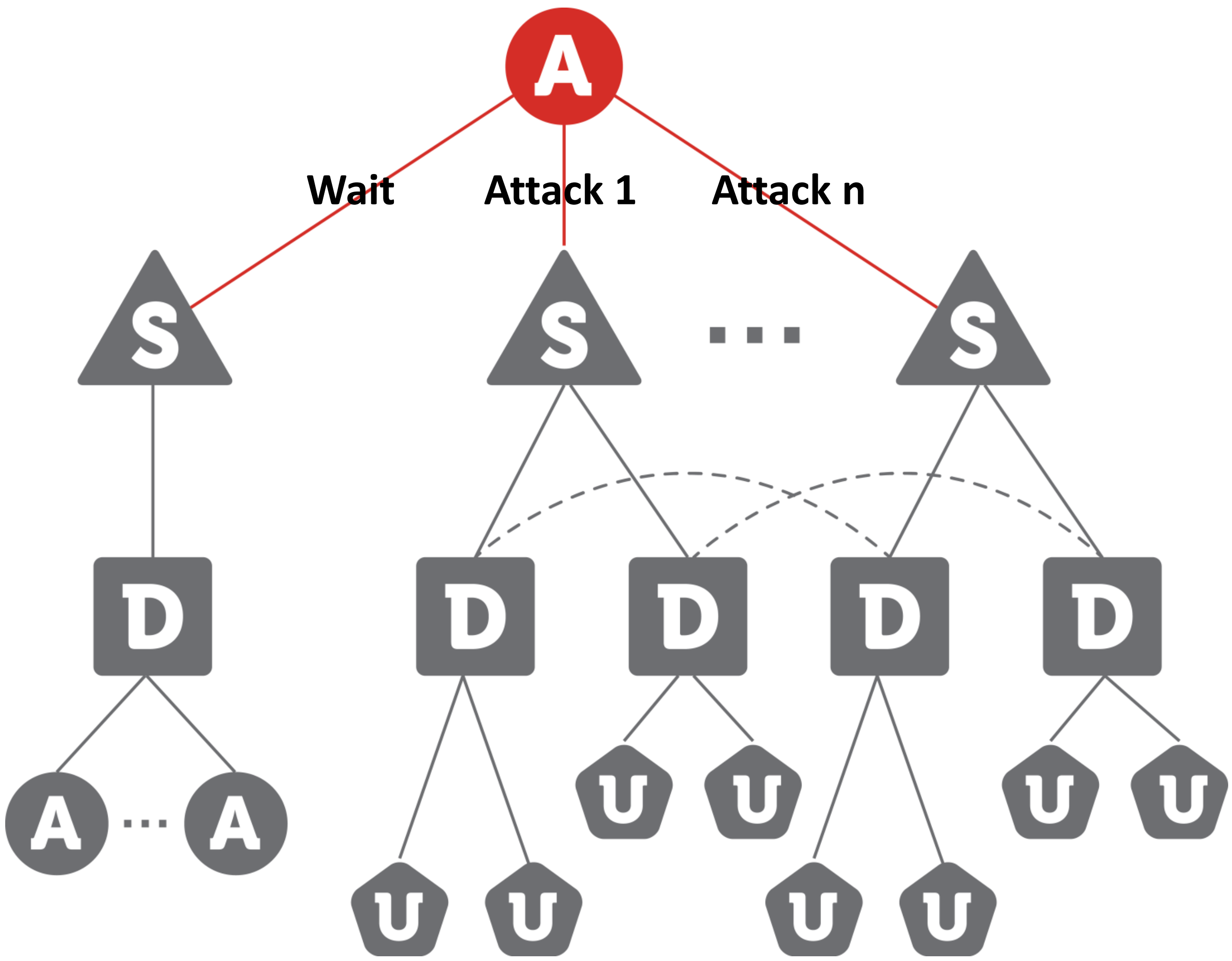


- Extensive form?

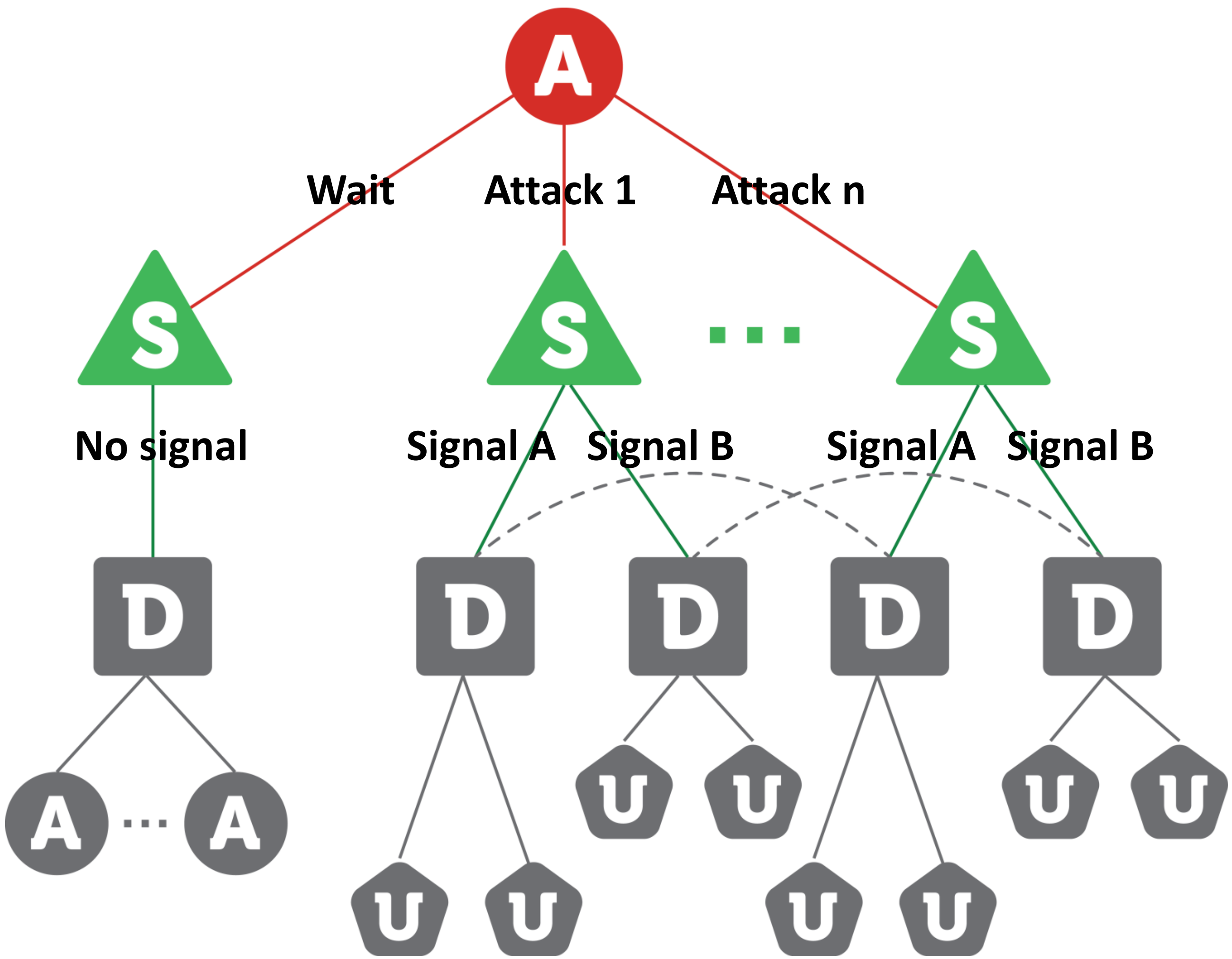
The Game Tree



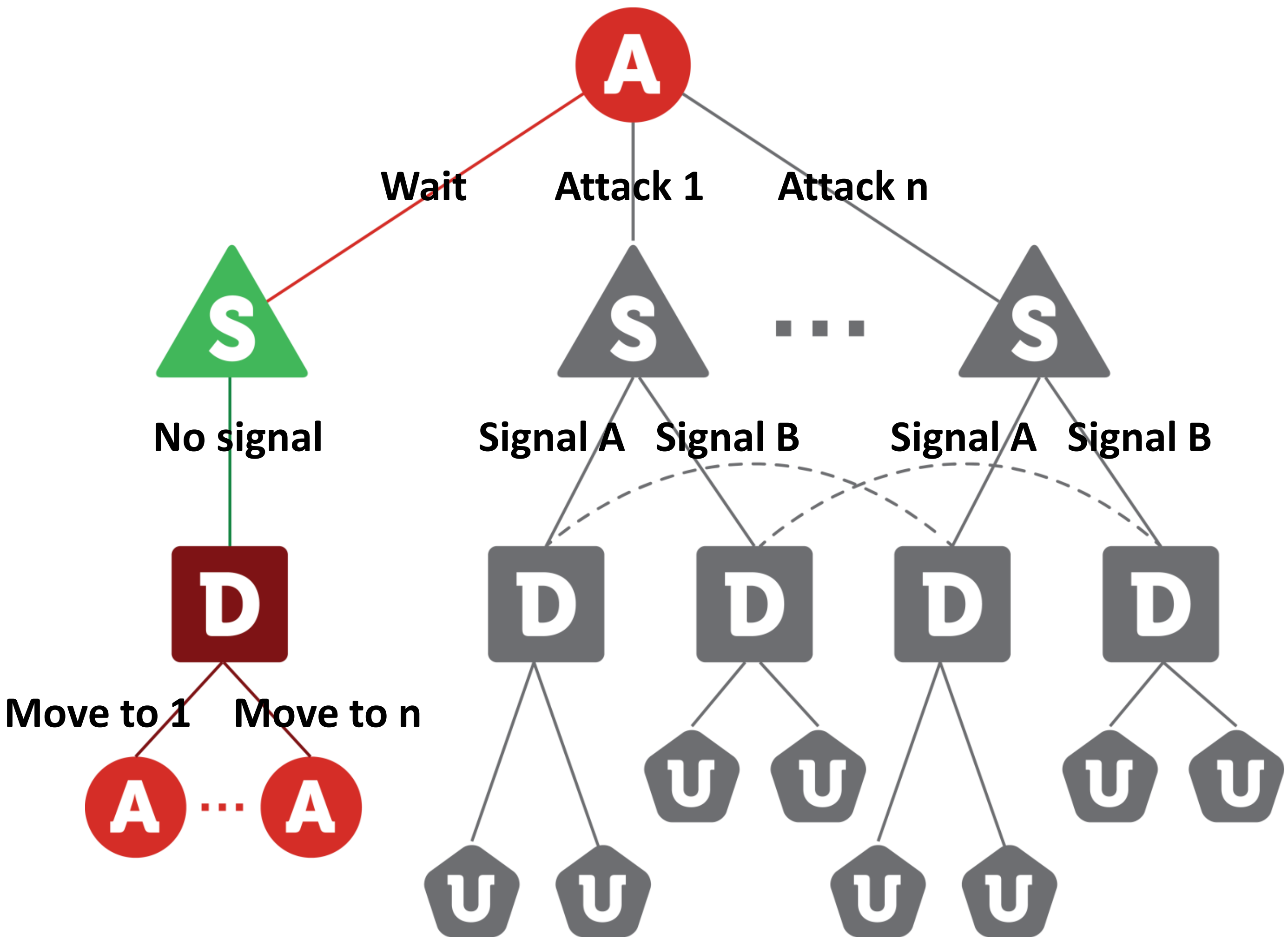
The Game Tree (Attacker)



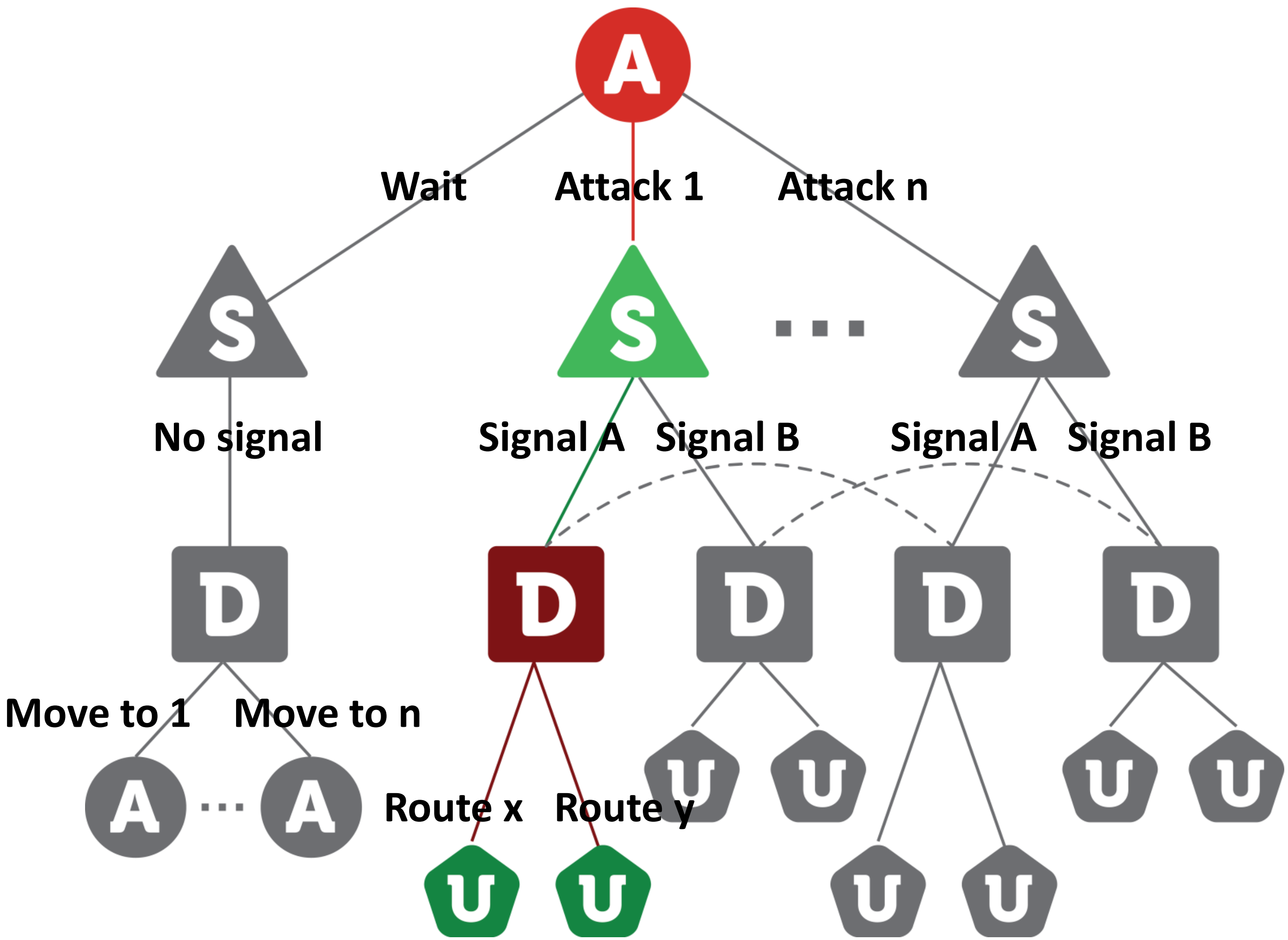
The Game Tree (Alarm System)



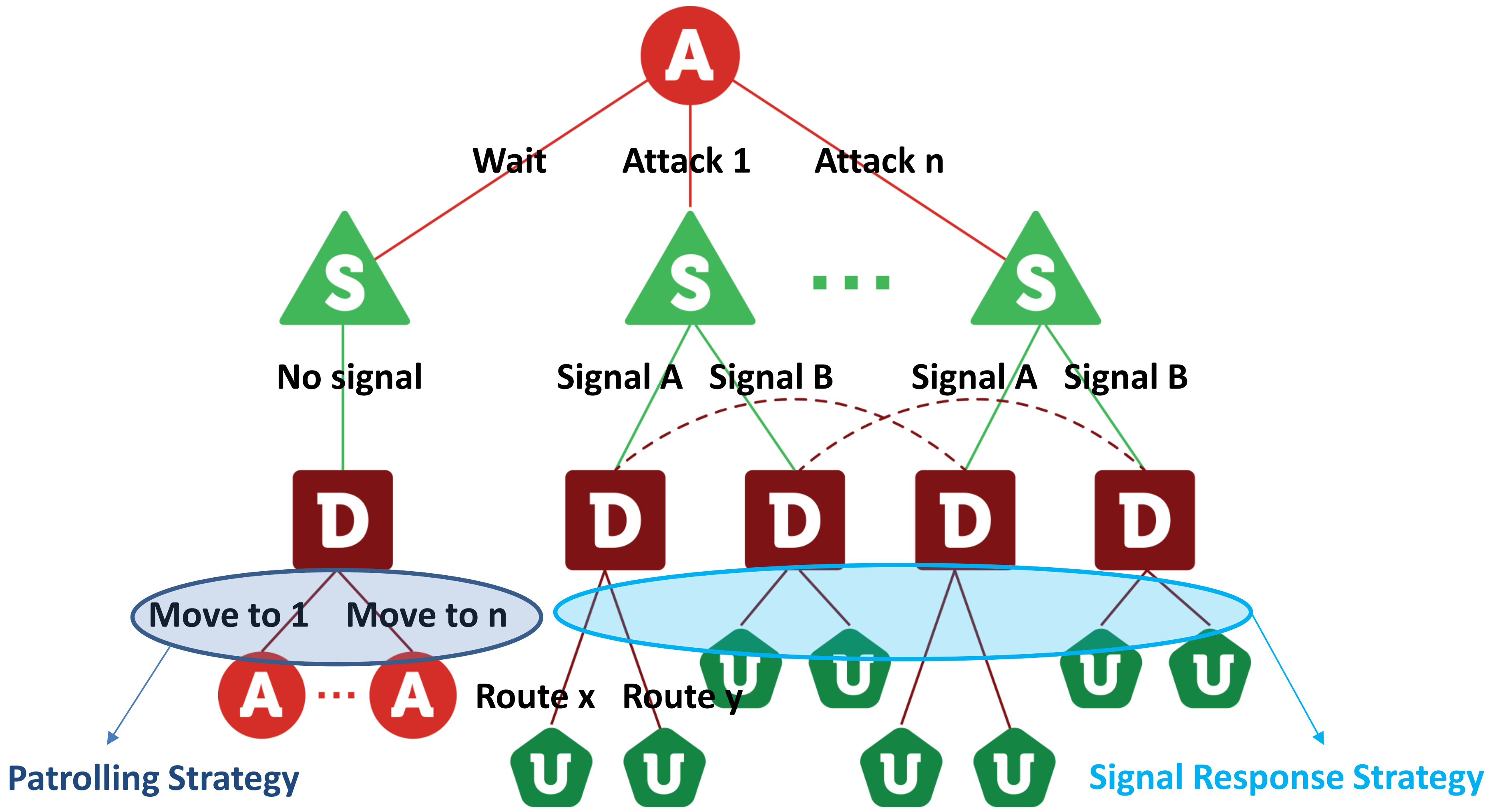
The Game Tree (Patrolling Game)



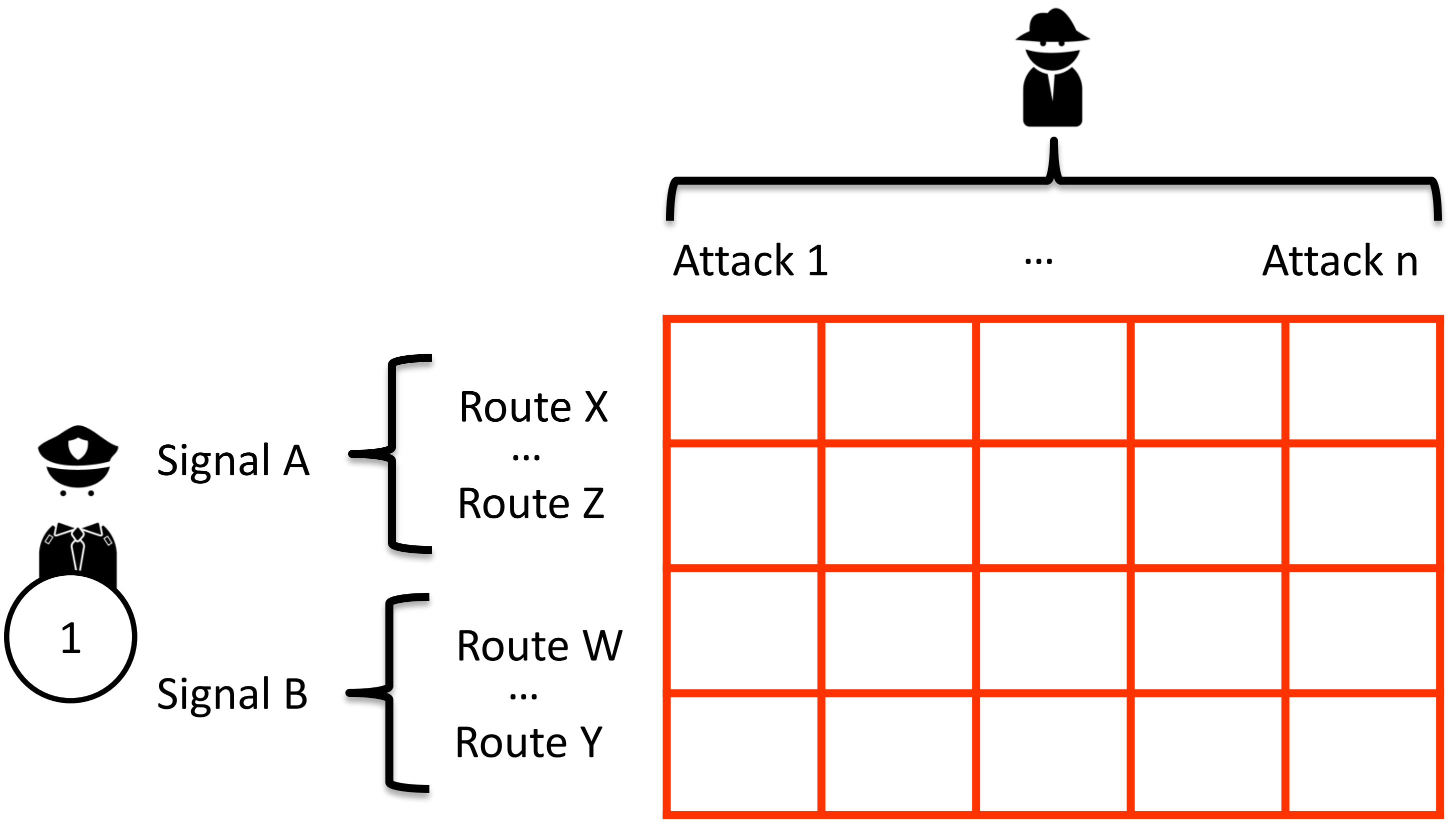
The Game Tree (Signal Response)



The Game Tree (Equilibrium Strategies)



Solving the Game



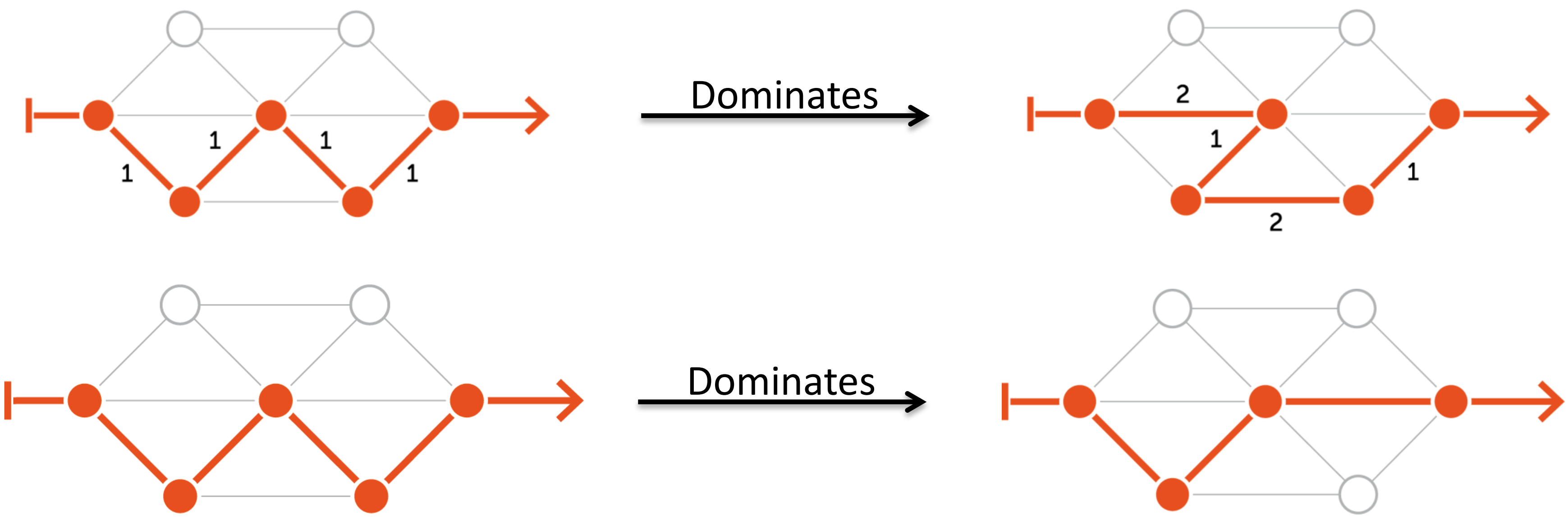
- Zero sum game: we can efficiently compute Nash Equilibrium
- How many covering routes do we need to compute?

Building the Game

- The number of covering routes is, in the worst case, prohibitive: $O(n^n)$ (all the permutations for all the subsets of targets)

Building the Game

- The number of covering routes is, in the worst case, prohibitive: $O(n^n)$ (all the permutations for all the subsets of targets)
- Should we compute all of them? No, some covering routes will never be played



- Even if we remove dominated covering routes, their number is still very large

Building the Game

- Idea: can we consider **covering sets** instead?

From $\langle t_1, t_2, t_3 \rangle$ to $\{t_1, t_2, t_3\}$

- Covering sets are in the worst case: $O(2^n)$ (still exponential but much better than before)

- Problem: we still need routes operatively!

- Solution: we find covering sets and then we try to reconstruct routes

Building the Game

INSTANCE: a covering set that admits at least a covering route

QUESTION: find one covering route

This problem is not only NP-Hard, but also *locally* NP-Hard: a solution for a *very similar* instance is of no use. ☹☹

Building the Game

- Idea: simultaneously build covering sets and the shortest associated covering route
- Dynamic programming inspired algorithm: we can compute all the covering routes in $O(2^n)$

Algorithm 1 ComputeCovSets (Basic)

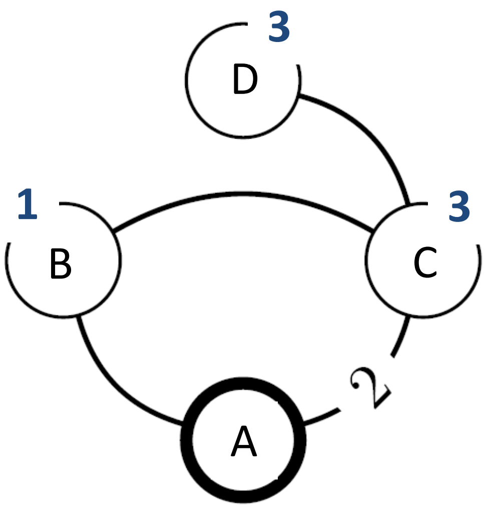
$\forall t \in T, k \in \{2, \ldots, |T|\}, C_t^1 = \{t\}, C_t^k = \emptyset$
$\forall t \in T, c(\{t\}) = \omega^*_{v,t}, c(\emptyset) = \infty$
for all $k \in \{2 \ldots |T|\}$ **do**
  for all $t \in T$ **do**
    for all $Q_t^{k-1} \in C_t^{k-1}$ **do**
      $Q^+ = \{f \in T \setminus Q_t^{k-1} \mid c(Q_t^{k-1}) + \omega^*_{t,f} \le d(f)\}$
      for all $f \in Q^+$ **do**
        $Q_f^k = Q_t^{k-1} \cup \{f\}$
        $U = Search(Q_f^k, C_f^k)$
        if $c(U) > c(Q_t^{k-1}) + \omega^*_{t,f}$ **then**
          $C_f^k = C_f^k \cup \{Q_f^k\}$
          $c(Q_f^k) = c(Q_t^{k-1}) + \omega^*_{t,f}$
        end if
      end for
    end for
  end for
end for

Is this the best we can do?

If we find a better algorithm we could build an algorithm for Hamiltonian Path which would outperform the best algorithm known in literature (for general graphs).

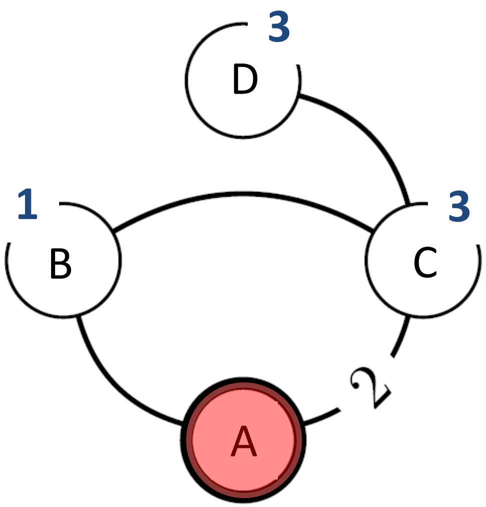
Algorithm

- Example



Algorithm

- Example

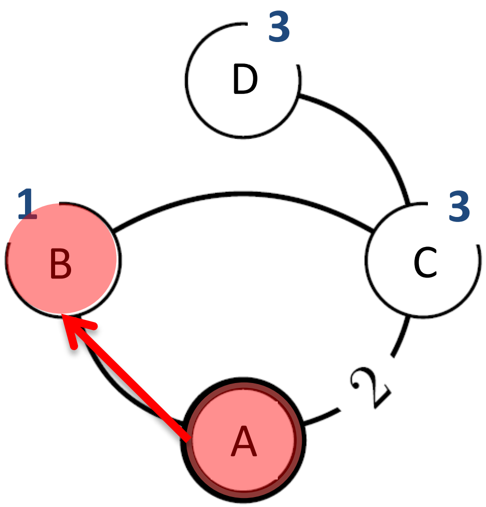


k=1

<{A}->A, 0>

Algorithm

- Example



k=1

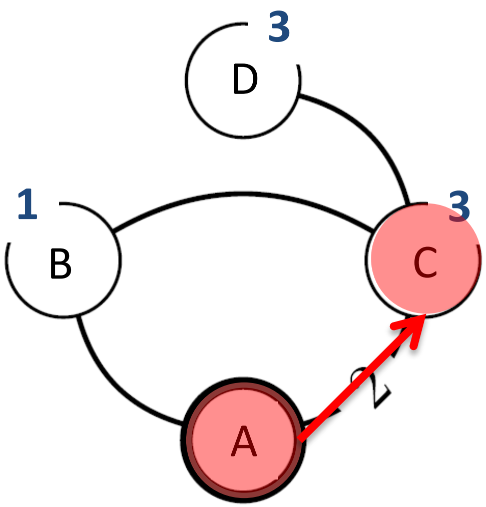
<{A}->A, 0>

k=2

<{A,B}->B, 1>

Algorithm

- Example



k=1

<{A}->A, 0>

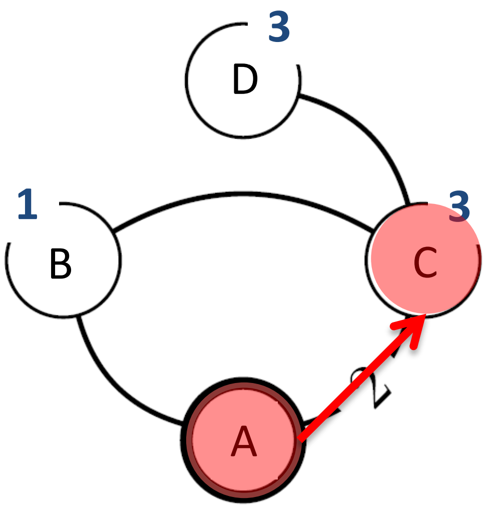
k=2

<{A,B}->B, 1>

<{A,C}->C, 2>

Algorithm

- Example



k=1

<{A}->A, 0>

dominated

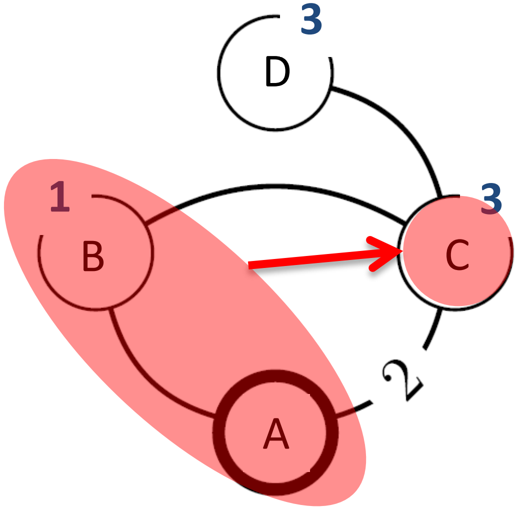
k=2

<{A,B}->B, 1>

<{A,C}->C, 2>

Algorithm

- Example



k=1

<{A}->A, 0>

dominated

k=2

<{A,B}->B, 1>

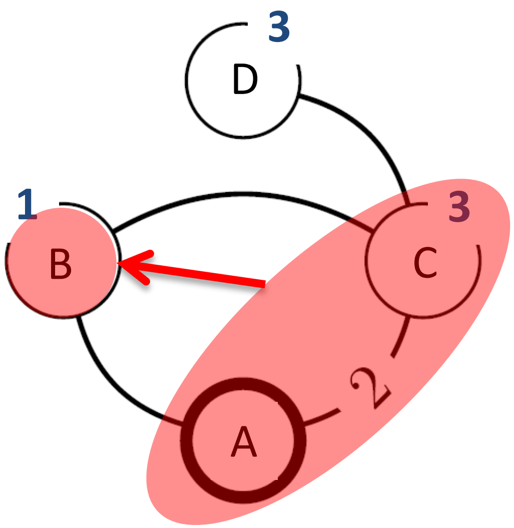
<{A,C}->C, 2>

k=3

<{A,B,C}->C, 2>

Algorithm

- Example



k=1

<{A}->A, 0>

dominated

k=2

<{A,B}->B, 1>

<{A,C}->C, 2>

k=3

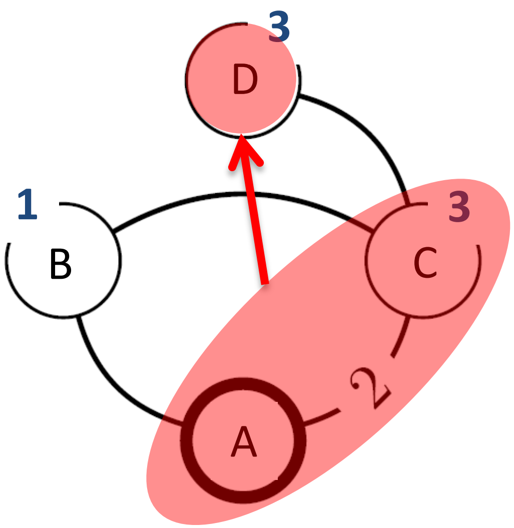
unfeasible

<{A,B,C}->B, 3>

<{A,B,C}->C, 2>

Algorithm

- Example



k=1

<{A}->A, 0>

dominated

k=2

<{A,B}->B, 1>

<{A,C}->C, 2>

k=3

unfeasible

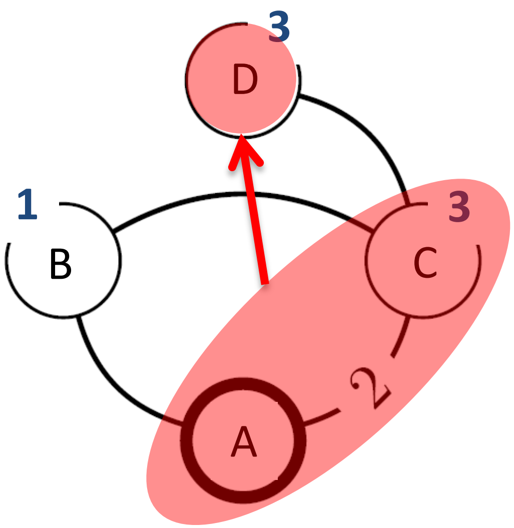
<{A,B,C}->B, 3>

<{A,B,C}->C, 2>

<{A,C,D}->D, 3>

Algorithm

- Example



k=1

<{A}->A, 0>

dominated

k=2

<{A,B}->B, 1>

<{A,C}->C, 2>

dominated

k=3

unfeasible

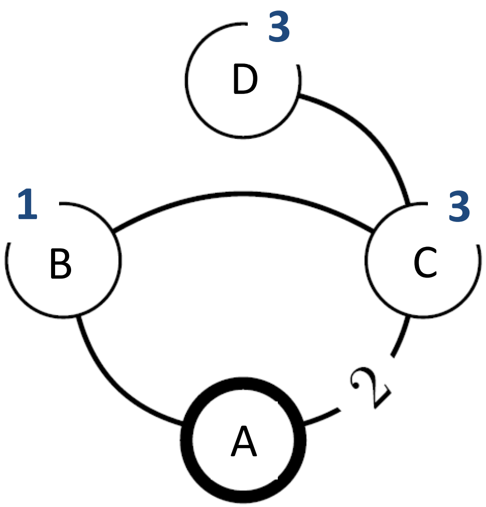
<{A,B,C}->B, 3>

<{A,B,C}->C, 2>

<{A,C,D}->D, 3>

Algorithm

- Example



k=1

<{A}->A, 0>

dominated

k=2

<{A,B}->B, 1>

<{A,C}->C, 2>

dominated

k=3

unfeasible

<{A,B,C}->B, 3>

<{A,B,C}->C, 2>

<{A,C,D}->D, 3>

k=4? All unfeasible

Building the Game (some numbers)

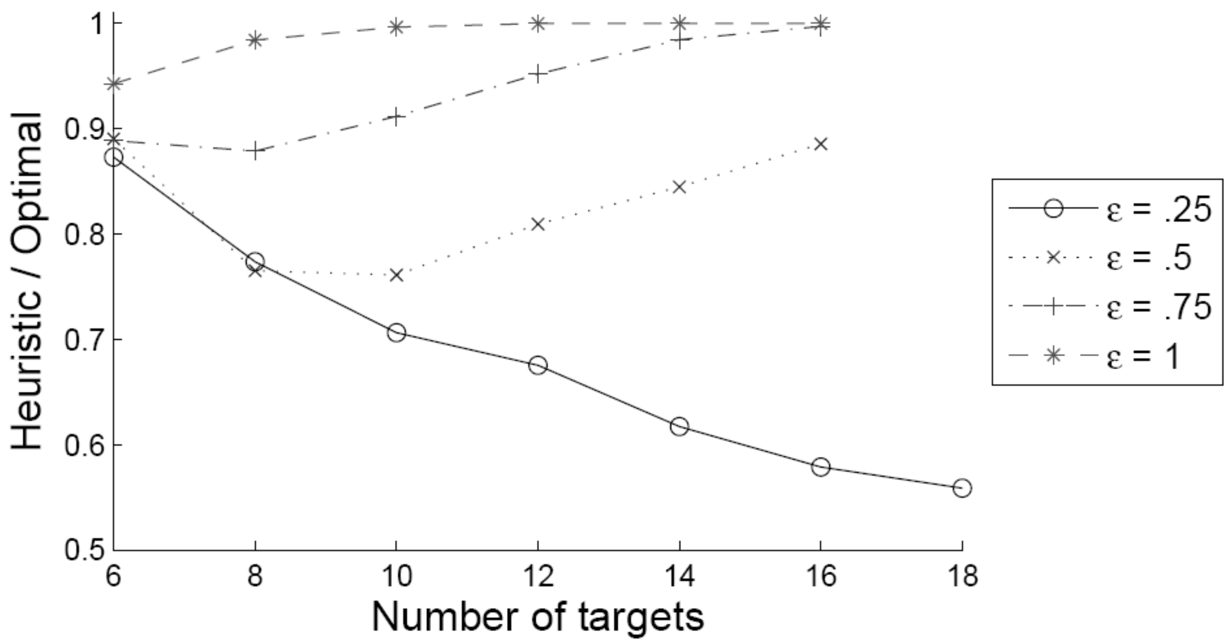
		$\|T\|$						
		6	8	10	12	14	16	18
ε	.25	0,07	0,34	1,91	11,54	82,26	439,92	**4068,8**
	.5	0,07	0,38	4,04	53,14	536,7	**4545,4**	≥ 5000
	.75	0,09	0,96	11,99	114,3	**935,74**	≥ 5000	≥ 5000
	1	0,14	1,86	17,46	143,05	**1073,**	≥ 5000	≥ 5000

- The edge density is a critical parameter. The more dense the graph, the more difficult to build the game.

		$\|T(s)\|$		
		5	10	15
m	2	-	17,83	**510,61**
	3	-	33	**769,3**
	4	0,55	35,35	**1066,76**
	5	0,72	52,43	**1373,32**

Building the Game (some numbers)

- Comparison with an heuristic sub-optimal algorithm.



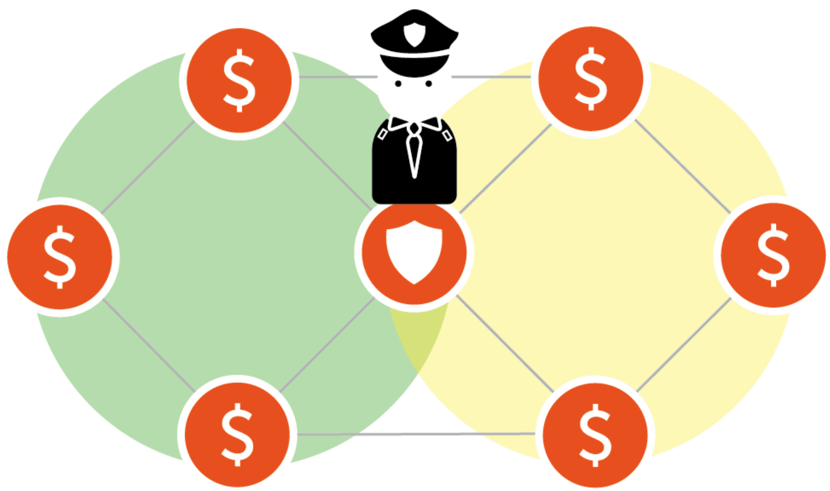
- Good news: the heuristic method seems to perform better where we the exact algorithm requires the highest computational effort

The Patrolling Game

- Solving the signal response game gives the Defender’s strategy on how to react upon the reception of a signal
- Patrolling game: what to do when no signal is received?
- It’s a Leader-Follower scenario: the Attacker can observe the position of the Defender before playing (we can solve it easily)
- What is the equilibrium patrolling strategy in the presence of an alarm system?

The Patrolling Game

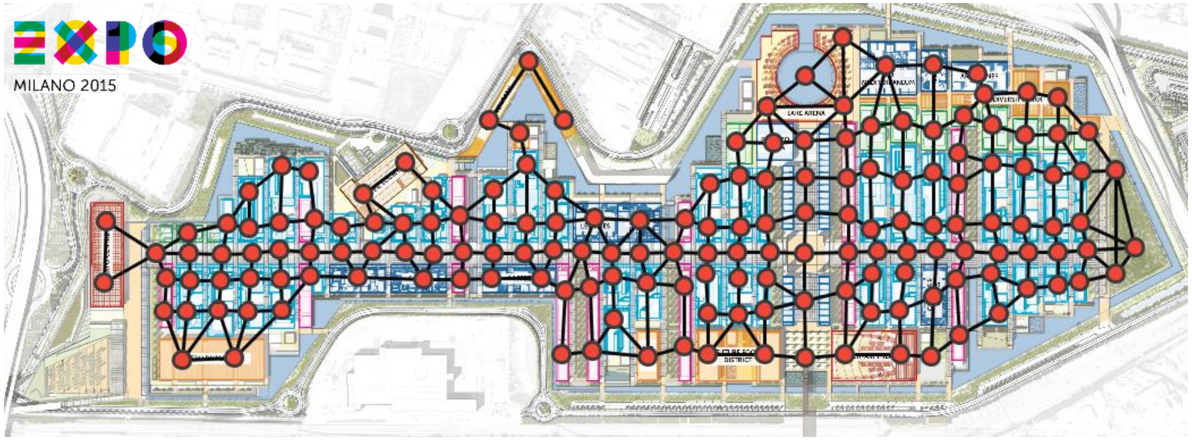
- Surprising result:
  - if the alarm system covers all the targets
  - if no false positive are issued
  - if the false negative rate below a certain threshold



- The equilibrium patrolling strategy is not to patrol! The Defender places at the most "central" vertex of the graph and waits for something to happen.
- If we allow false positives and arbitrary false negatives, things become much more complicated.

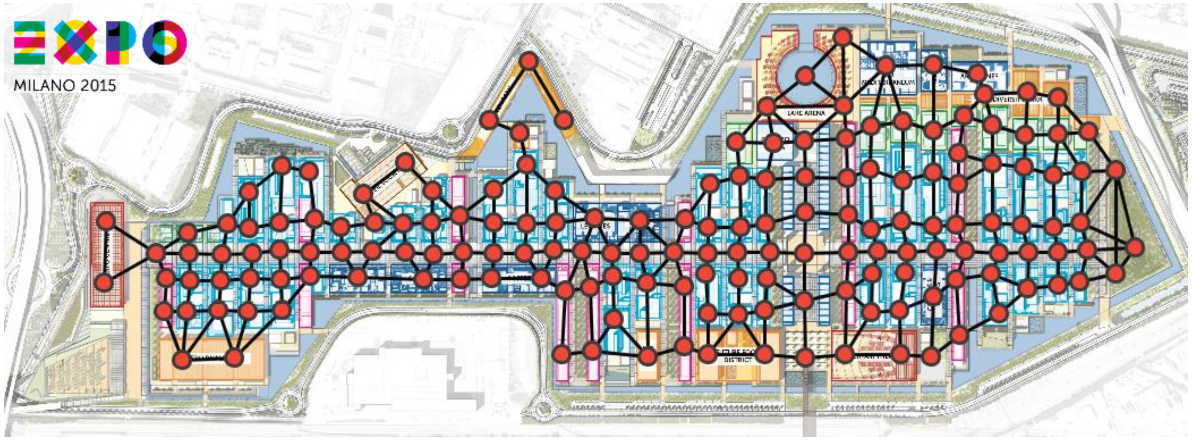
A real case study

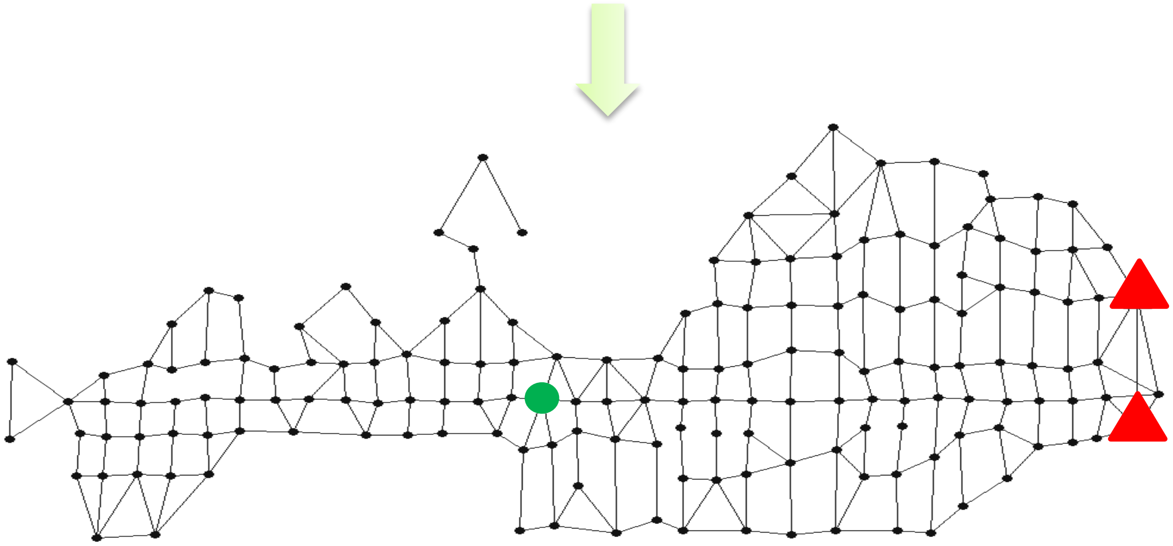
A real case study



Values and penetration times derived from public data of the event

A real case study





Values and penetration times derived from public data of the event

An application in cyber security

- Service S: composed by software models M1, M2, …, Mn
- Each module Mi represents a conceptually stand-alone component of the service which is executed on the client machine and can be replaced independently
- V(Mi) is the value of a software model
- T(Mi) is the expected corruption time
- We can update Mi, paying a cost (and vanishing any ongoing corruption effort)
- Updates can be observed