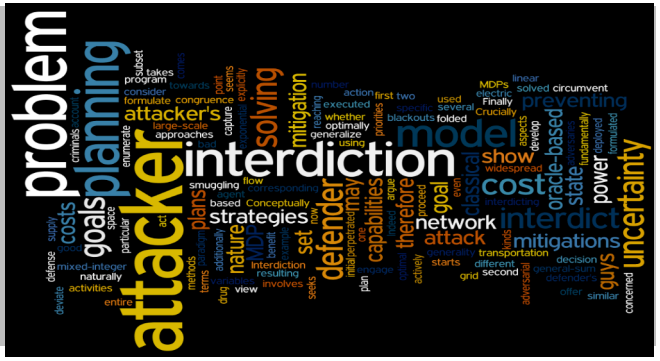
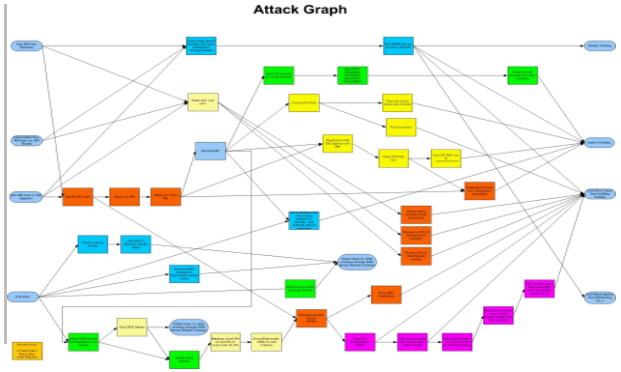
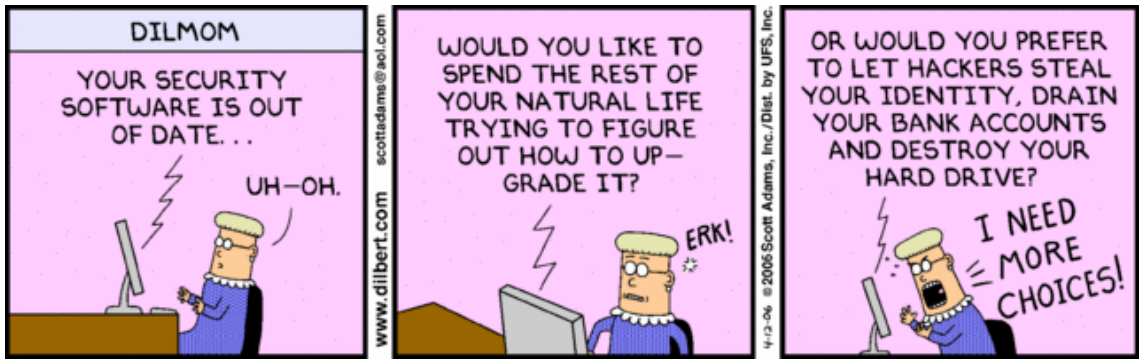
SAND2013-5730P

Exceptional service in the national interest

Sandia National Laboratories







Cyber Games

Yevgeniy [Eugene] Vorobeychik

Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY | NNSA Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

The Cyber threat

Sandia National Laboratories

- *Attackers are increasingly sophisticated, often have (direct or tacit) support of a state, very highly rewarded for success, and pay attention to what we are doing!*

- *Stuxnet/conficker*
  - *sophisticated combination of cyber attack tactics, including zero-days*
  - *Stunxnet: cyber-physical attack targeting critical infrastructure*

- *Corporate America loses between $100 billion to $1 trillion / year from theft of proprietary information (The Economist, Babbage blog, May 11, 2012)*

Conficker vs. The World

Sandia National Laboratories

- Conficker:
  - an infected machine generates 250 random domain names from 5 TLDs each day, and if one resolves, it is used to download the payload (seeded with date to synchronize)
- The World:
  - block all 250 domain names each day to counteract this
  - anti-virus signature, windows update
- Conficker:
  - increase number of TLDs to 8
  - blocks auto-update
  - attaches to removable media
- The World:
  - Conficker working group continues to shut down the generated domain names
- Conficker:
  - generate 50,000 random domain names from 8 TLDs

Security as Chess

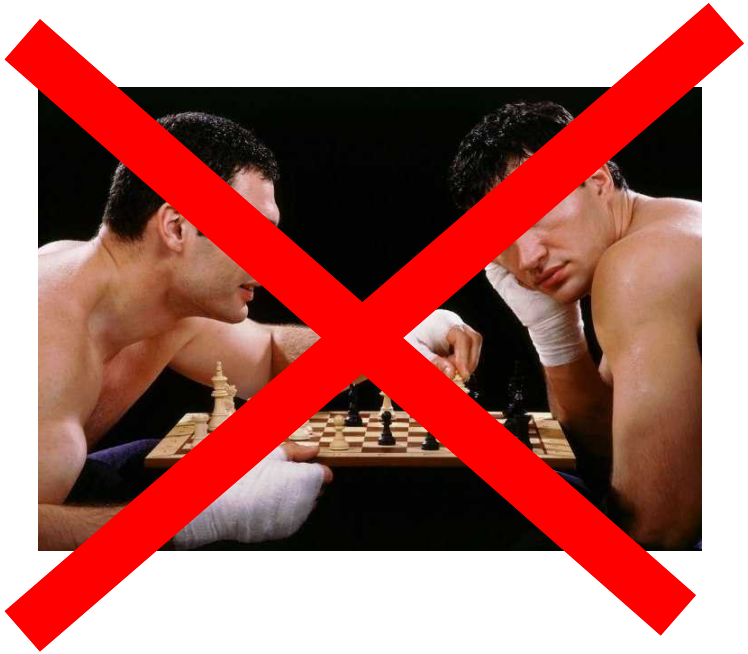
Adversarial: only one winner ("zero-sum")

Opponents must anticipate each other's moves

strategy should account for how the opponent will respond

Extremely complex: impossible to enumerate the set of all states in the game

How is Cyber NOT like Chess?



Chess has *complete information*:
- well-known rules
- clear what the opponent's objective is
  - *Cyber*: attacker' specific objectives are uncertain

Chess has *perfect information*:
- can observe all previous moves
  - *Cyber*: attackers can be difficult to detect

Chess has *no uncertainty* of any kind:
- consequences of moves are deterministic
  - *Cyber*: systems complex, non-deterministic

Chess has *only two players*
  - *Cyber*: many players

Game Theoretic Model

Sandia National Laboratories

(Non-Zero-Sum) Stackelberg Game

- *Defender moves first*:
  - *jointly chooses defense policies*
- *Attackers learn about the defenders' decisions*
  - *respond by choosing optimal attack policies*

Outline

- Complexity of Attack Space
  - Model attacker as a planning agent
  - Optimal plan interdiction problem
- Uncertainty about attackers (capabilities, goals); noise
  - Bayesian plan interdiction problem
  - Interdicting an attacker MDP
- Machine learning in adversarial settings

OPTIMAL INTERDICTION OF ATTACK PLANS

Game Theory for Physical Security

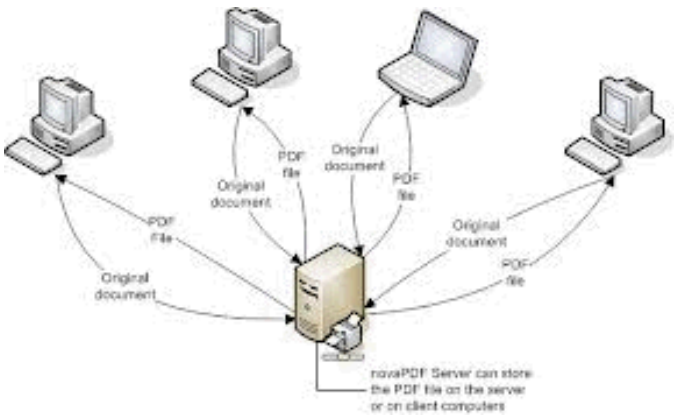
Sandia National Laboratories

- Defender protects N targets
- Attacker attacks a single target
  - maximizes expected utility given defense decisions

Fielded systems in LAX, US Coast Guard, Federal Air Marshall Service

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

Game Theory for Cyber Security



targets?

Sophisticated cyber attacks involve complex series of steps towards an attack goal

Cyber attacks (*Boddy et al.*):

1. Attacker sends an email message, spoofed to be from a colleague, with a new screensaver as an attachment.
2. Attachment is an executable that enables remote login, and captures and relays the user's password.
3. Attacker logs into the machine and executes a buffer overflow attack, gaining root (admin) privileges.
4. ...

Threat/Adversary Modeling

Sandia National Laboratories

- Basic picture in threat modeling and mitigation:
  - *Attacker*:
    - has a set of capabilities
    - can choose from a collection of "actions" (activities, exploits, attacks)
      - actions can have costs (time, likelihood of detection/capture, etc)
    - has a set of goals
      - goals may have different importance to the attacker/defender
      - may be content to achieve a subset of goals
    - constructs a plan (a sequence of actions) to achieve goals starting with capabilities
  - *Defender*:
    - *mitigations*: can block attack actions, initial capabilities, patch vulnerabilities
      - mitigations can be costly

Typical Approaches: Attack Graphs & Scenarios

Sandia National Laboratories

- A common way attacker decision process is modeled in practice (e.g., red teaming) is by using *attack graphs* and/or *scenarios*
- *Attack graphs:* a graph describing possible steps an attacker may take to achieve a goal
- *A scenario:* a sequence of actions that allows the attacker to achieve a goal starting from initial capabilities
  - For example: starting with a set of capabilities, how an attacker can exfiltrate Data X

Shortcomings of Attack Graphs/Scenarios

Sandia National Laboratories

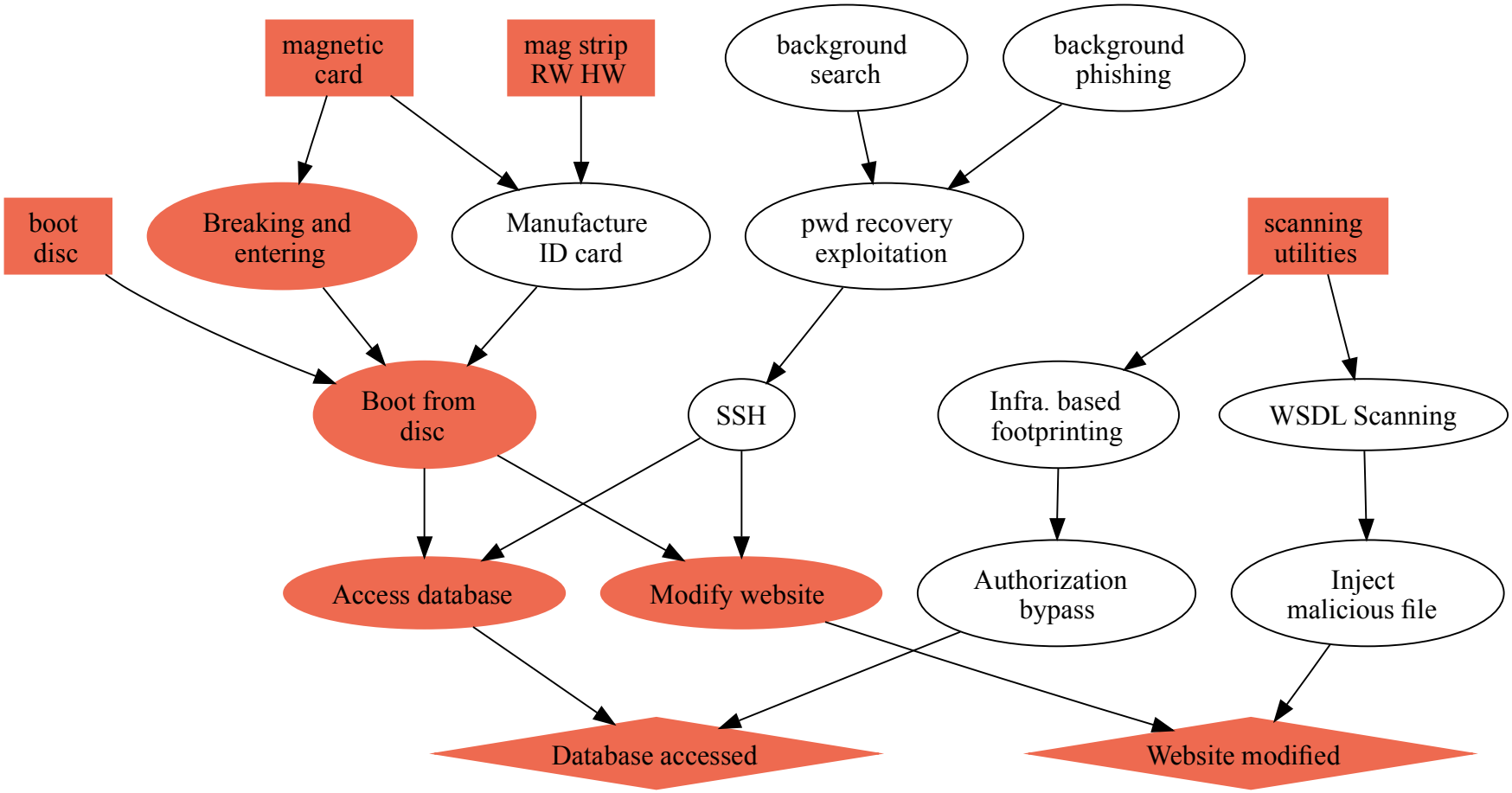
- Attack graphs become unwieldy very quickly
  - size exponential in the number of things the attacker can affect
  - 400 variables = 2^{400} states/nodes; age of universe, in seconds < 2^{30}
- Only a few attack scenarios can be crafted by hand; can miss the vast majority of possible attack graph paths

Modeling Attacker as a Planner

Sandia National Laboratories

- One of the main branches of AI is (formal/logic-based) planning
  - Model the planning problem using formal logic
  - Numerous heuristic planning tools
- (*Deterministic/Classical*) Planning problem:
  - The world = a set of logical variables = *state* (of the world)
  - Start: initial state of the world (variables that are **true** at time 0)
    - *attacker capabilities, network vulnerabilities*
  - Goal(s): variables (or boolean expressions) that the planner wants to satisfy
    - *attacker's goals (e.g., exfiltrate Data X)*
  - Actions: actions/steps a planner can take towards the goal
    - preconditions: variables that must be true for the action to apply
    - effects: variables that become true/false as a result of the action
  - A plan: a partial order of actions that achieve goals starting from the initial state

("Action") Attack Graph, and an Optimal Plan for an Example Scenario



Deterministic Plan Interdiction Problem

Sandia National Laboratories

DPIP: defender chooses an optimal subset of mitigations, accounting for attacker's best response plans

(optimal: maximizing defender's utility)

(utility = Value of goals – cost of mitigations)

Deterministic Plan Interdiction (DPIP)

Sandia National Laboratories

- DPIP_DP (deterministic plan interdiction decision problem): can the defender achieve target utility?
- Theorem: DPIP_DP is PSPACE-Complete
- Proof: by reduction from partial satisfaction planning
- Optimization: based on a known Integer Program for computing an optimal partial satisfaction plan (Vossen et al., 1999; Briel et al., 2004)
  - Maximize defender utility
  - subject to:
    - a plan is feasible/legal
    - a plan is the best plan for the attacker (*by comparison to all **feasible** plans*)

DPIP formulation

$$\max_{D_a, D_m, y_{a,t}, \delta_p} \quad \sum_{l \in L} V_l^D s_l - \sum_{m \in M} D_m C_m^D$$

s.t. :

$$\forall_a \quad D_a \leq \sum_m D_m A_{m,a}$$

$$\forall_{m,a} \quad D_a \geq D_m A_{m,a}$$

compute which actions are interdicted

$$\forall_{a,t} \quad y_{a,t} \leq (1 - D_a)$$

attacker cannot choose interdicted actions

$$\forall_{p,a} \quad \delta_p \geq D_a$$

$$\forall_p \quad \delta_p \leq \sum_{a \in p} D_a$$

identify non-interdicted plans

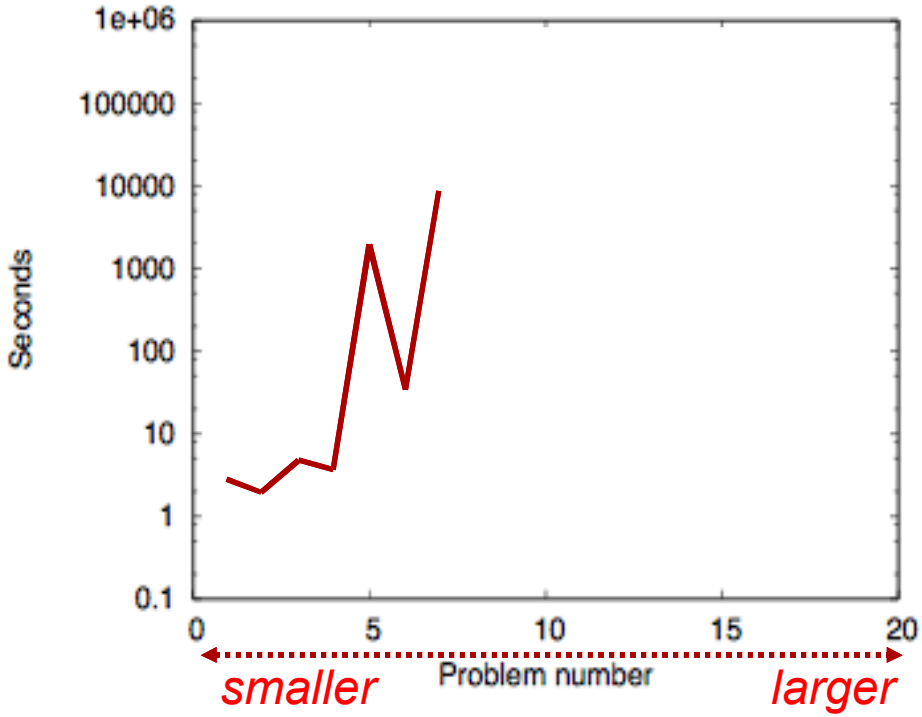
$$\forall_p \quad \sum_{l \in L} V_l^A s_l - \sum_{a,t} C_a^A y_{a,t} \geq U^A(p) - Z\delta_p$$

choose the best non-interdicted attack plan

plan feasibility constraints

DPIP (cont'd)

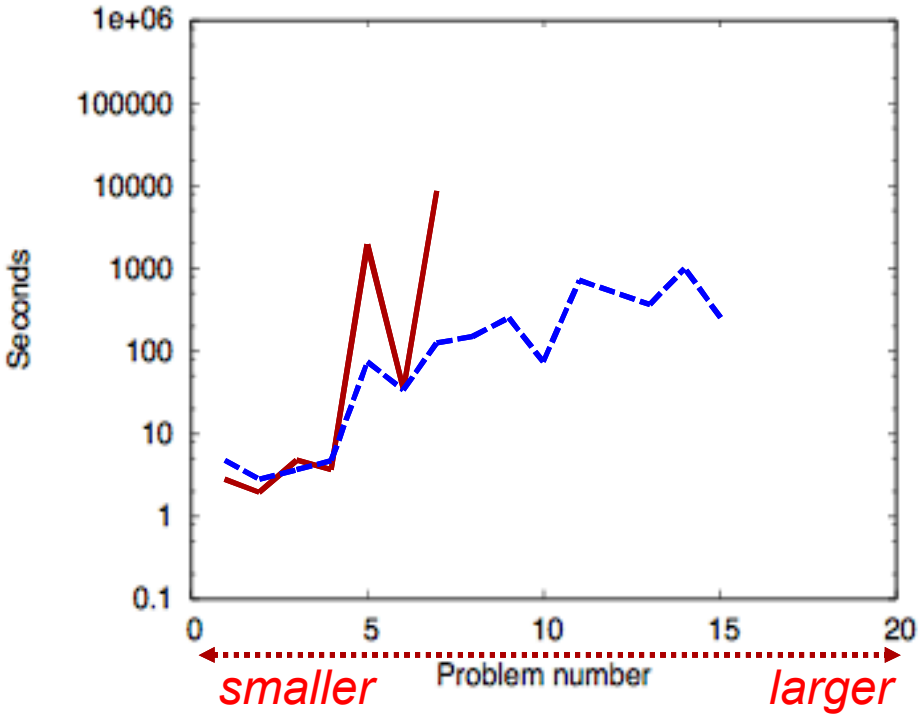
- IP: far too large (lots of feasible plans)
- Solution approach: constraint generation
  - *We "know" how to compute an optimal plan*
  - Iterate:
    - Compute optimal mitigations, given a subset of feasible plans
    - Compute a best response (plan) given mitigations
    - Use best response to generate a new set of constraints
    - Repeat (until convergence)



Problems from IPC

DPIP: Speeding Up

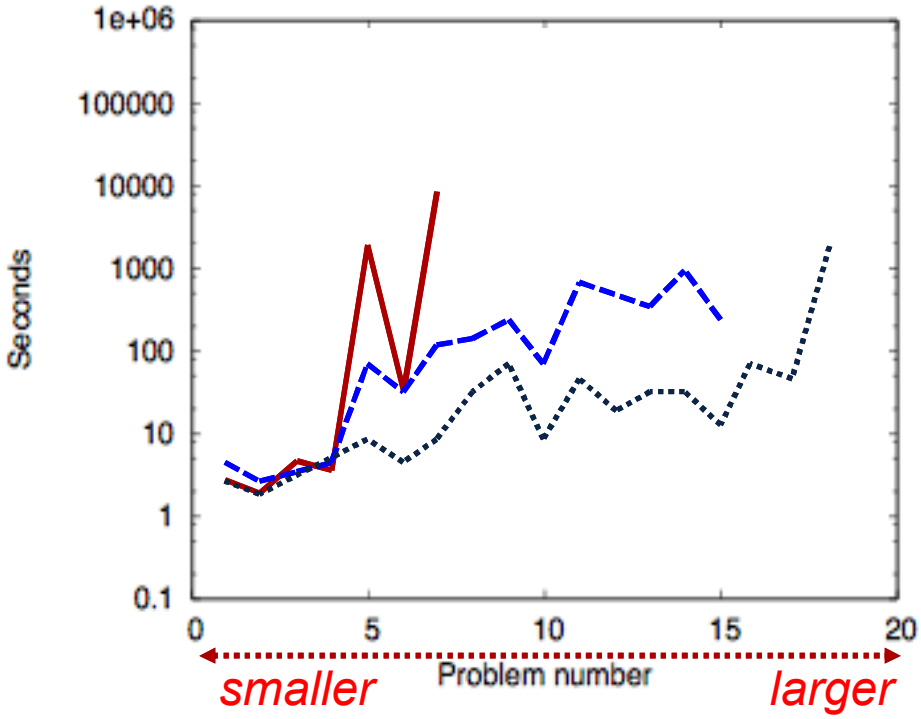
- Computing an *optimal* attack plan is quite hard
- Solution: set a time limit on the IP (usually lots of time is taken up to prove solution optimality)
- Verify optimality only at the very end
  - Still provably optimal solution for the defender



Problems from IPC

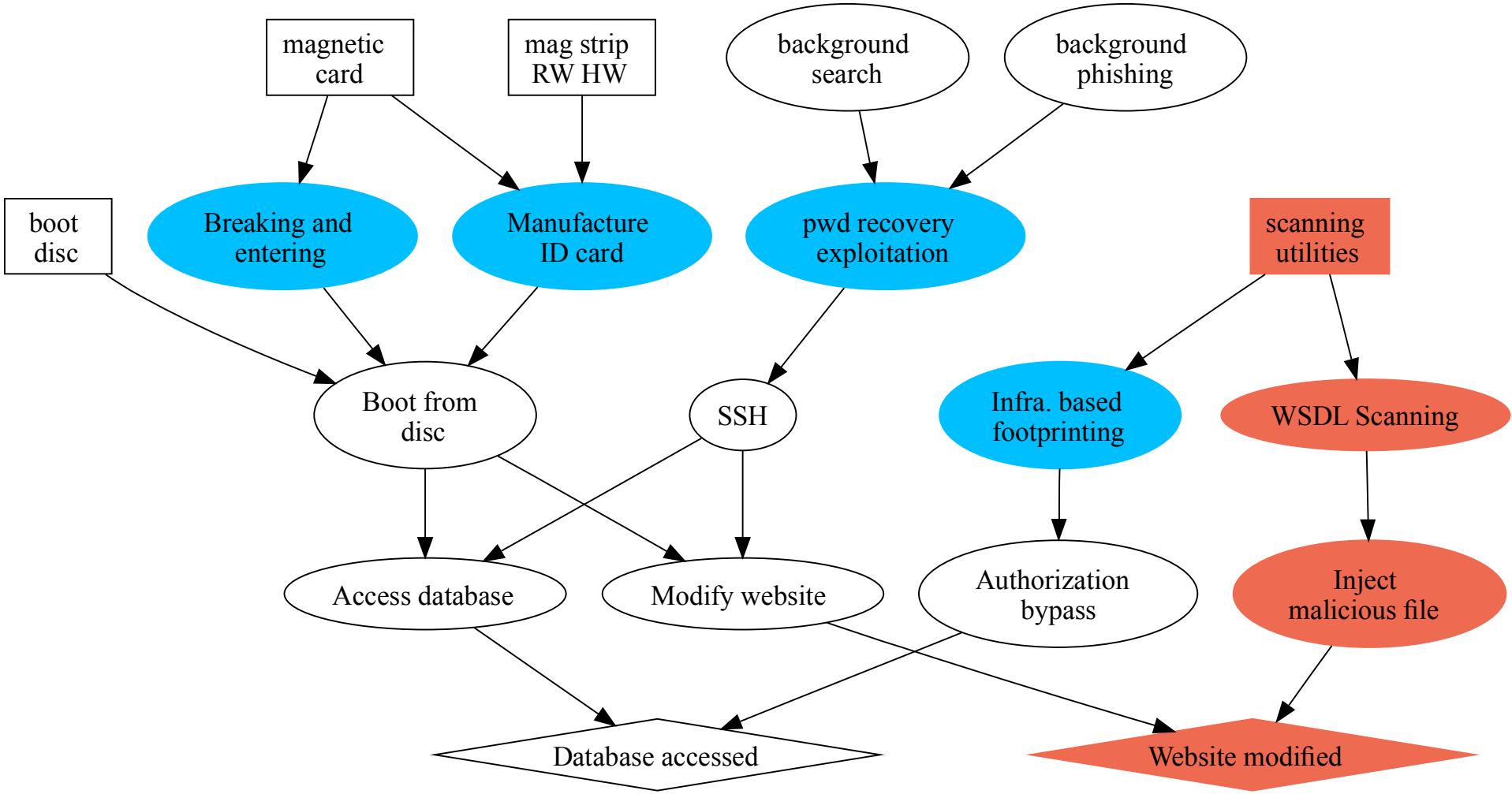
DPIP: speeding up more

- A better idea: leverage best heuristic planning tools from AI research community
  - SGPLAN5: state-of-the-art heuristic partial satisfaction planner; winner of IPC
- Still guarantee optimal solution by running the full IP in the end



Problems from IPC

Optimal Plan Interdiction Example



INCORPORATING UNCERTAINTY

What about uncertainty?

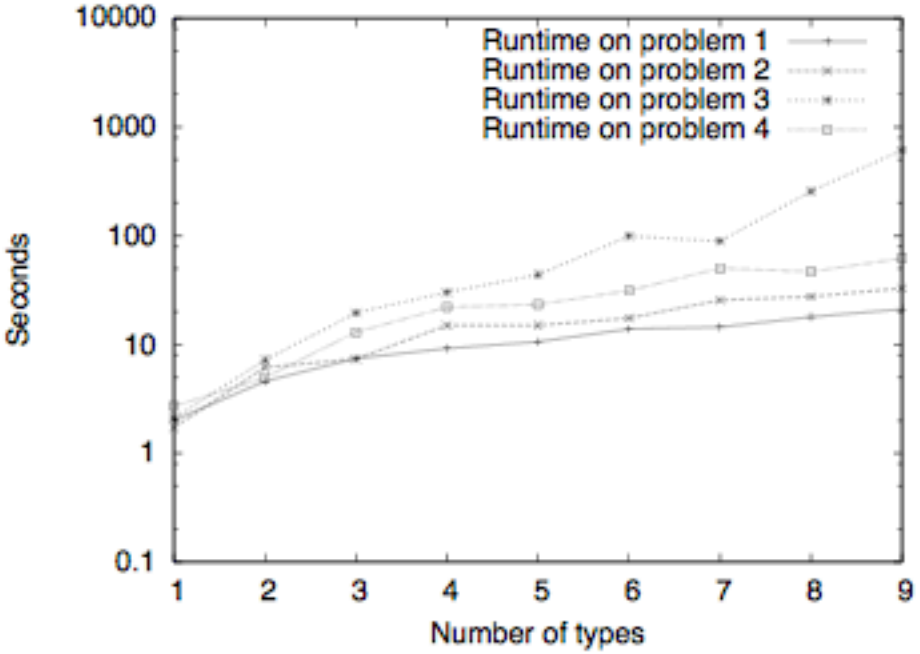
- DPIP assumes everything is certain/deterministic
- Usually uncertain about:
  - Attacker capabilities, goals
  - Execution and effects of actions
  - etc
- Questions:
  - Can we extend DPIP to accommodate all/some of this uncertainty?
  - If not, how should we model it?

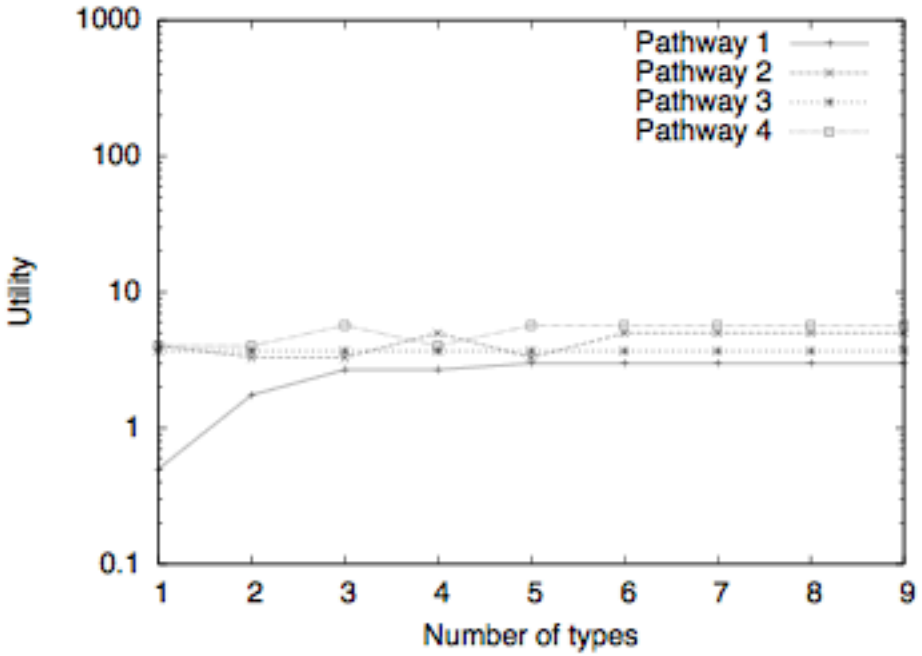
Uncertainty about capabilities/goals

- Can naturally model within the DPIP framework
  - Attacker has multiple types ("personalities"), each type corresponding to a set of capabilities/goals (*Bayesian plan interdiction problem; BPIP*)
  - Each type computes an optimal plan in response to mitigations
  - Objective now to maximize expected defender utility wrt distribution over attacker types

$$\sum_{\theta} \sum_{l \in L} p_{\theta} V^{D}_{\theta,l} s_{\theta,l} - \sum_{m \in M} D_m C^{D}_{m}$$

Experiments: BPIP





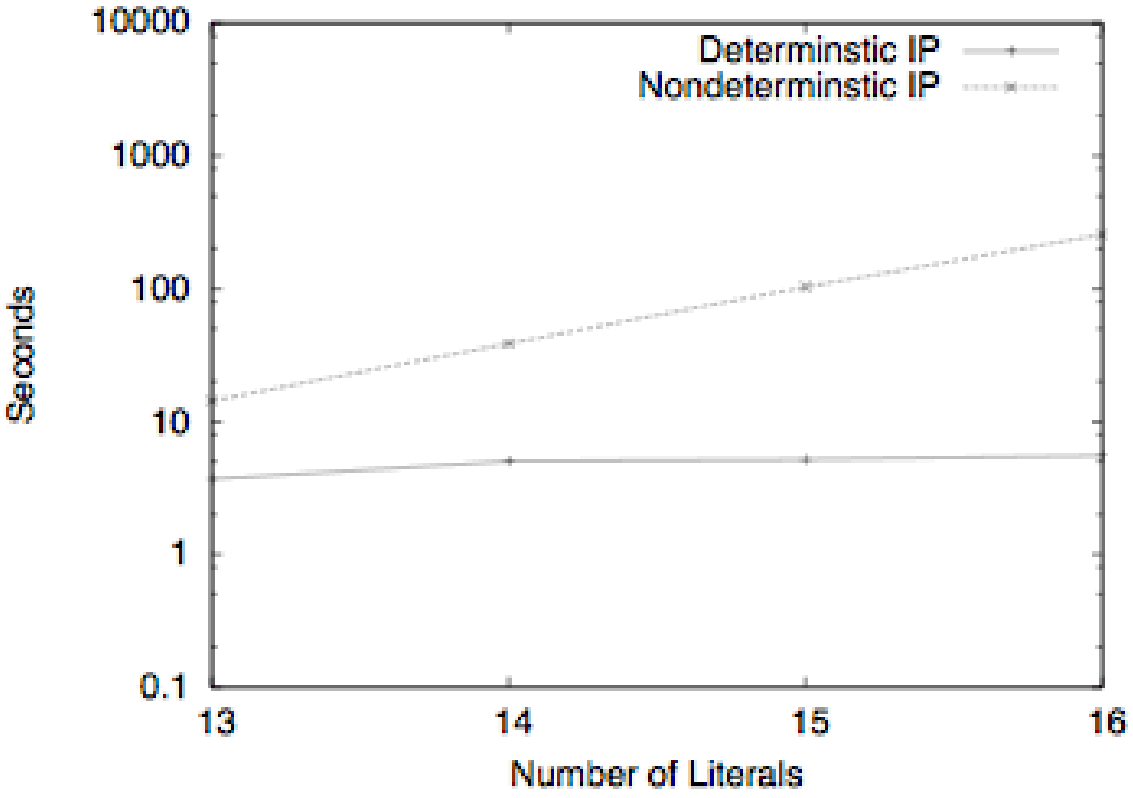
- Created an instance with 9 types, and aggregated (abstracted) types into sets to get fewer types
- Scales reasonably well, and only need relatively few types to get near-optimal solutions

Execution Uncertainty

Sandia National Laboratories

- *Special case*: only uncertain about whether an action succeeds or not, success is observable, and action can be repeated
  - Can be done using DPIP framework by computing expected number of tries for each action
- *Generalizing*: state is observable, but action may have non-deterministic effects
  - Example: “port scan” action (info you get is uncertain)
  - Attack planning problem is an MDP
  - MPD interdiction (MDPIP)
  - Leverage (dual) LP for computing optimal value in an MDP
  - *State space explosion*: must now explicitly represent all states in the system

Experiments with MDPIP



Can't scale beyond 15-20 literals

Much slower / poorer scalability than DPIP

But: far more general

Sandia National Laboratories

USING MACHINE LEARNING IN ADVERSARIAL ENVIRONMENTS

Traditional Machine Learning

Sandia National Laboratories

- Take a data set $\{(x_1,y_1),\ldots,(x_n,y_n)\}$, $(x,y) \sim \mathcal{D}$
- x is a feature vector
- Train a classifier, f(x) on data
- "Test" by classifying unseen instances $(x',y') \sim \mathcal{D}$
- Key assumption: both training and test data come from (roughly) the same distribution

- Alternative: online learning
- The opposite extreme: no distributional assumptions
- BUT: guarantees are asymptotic (for "large" sequences of x), and very weak; assume that labels y(x) are fixed for a given x

Machine Learning in Adversarial Environments

- Adversary deliberately manipulates features to bypass defensive measures
  - adversary previously (i.e., in $\mathcal{D}$) chose x (f(x) = bad), now chooses x' such that f(x') = good
  - clearly distribution is non-stationary
  - adversary effectively manipulates labels on instances (x' may previously have usually been good)
  - we care about *short-run impact*

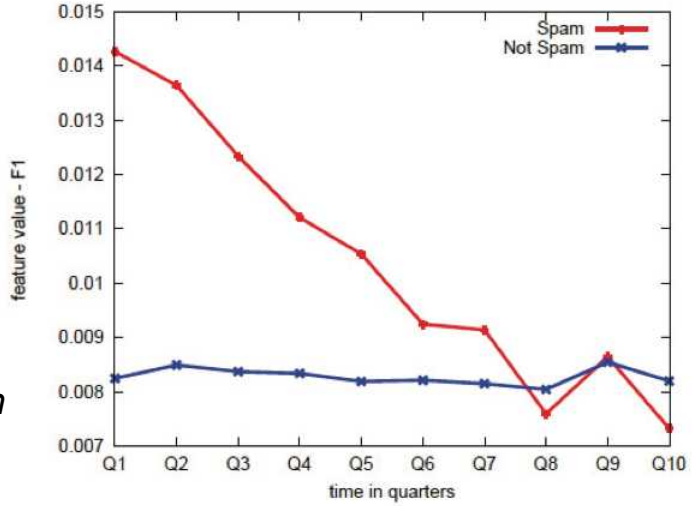


figure due to Rich Colbaugh and Kristin Glass, Sandia National Laboratories

Related Work

- Dalvi et al., 2004, Bruckner and Scheffer, 2009; 2011, Liu and Chawla, 2010, Colbaugh and Glass, 2013, etc
  - Adversary performs a linear transformation of the data (except for Dalvi et al)
  - Adversary aims to maximize total loss

Our Work

Sandia National Laboratories

- adversary can choose an arbitrary x
- adversary wishes to maximize expected utility from a successful attack (doesn't care about false positives)
- "defender" cares about false positives, as well as false negatives (Sommer and Paxson, 2010)
  - lost value to users
  - operational constraints: can only "inspect" so many cyber alerts
- many attackers
  - different attackers (or types) reflect different goals; why lump them into one bin?
- revealed preference
  - attacker previously used *x* as an attack; this reveals his preference about the most favorable means of attack
  - choice of *x'* must be as close to *x* as possible (change only a few feature values; similar to Dalvi et al. 2004 and Lowd and Meek, 2005)

Learning in a Box

Sandia National Laboratories

- Key insight: separate *learning* [about prediction based on current distribution of attacks] and *operational decisions* [using predictions made by learning and an adversary model]
  - Use learning as a black box to get *p(x)* = probability that *x* is generated by a malicious actor (e.g., use Naïve Bayes probabilities)
  - Optimize operational decisions based on *p(x)* and a *model of adversarial response*

Modeling Adversarial Response

- Let *x* be the preferred attack of some attacker
  - an email (feature vector) that embeds a malicious link
  - a sequence of steps to gain privileged access
- Suppose that *q(x)* is the probability that *x* will be blocked/*inspected*/generate an alert/etc
  - if $q(x) = 1$, *x* is not an effective means of attack
  - perhaps there is some other *x'* that achieves a similar goal, but has $q(x') < 1$?
  - *x' close to x are much more likely to have this property*
- Define $Q(x,x') = \exp\{-\delta \, ||x - x'||\}$
  - $\delta = 0$ => *doesn't matter which x' to use;* δ *large => can only achieve the objective with x*
- Utility of using *x'* is $\mu(x,x'; q) = V(x)\, Q(x,x')\, (1 - q(x'))$

Full model

- Consider two classes of adversaries (special cases of the model)
  - static ($\delta = \infty$), with probability P_S and dynamic ($0 < \delta < \infty$), w.p. P_A
  - static: those reusing already available tools/templates
  - dynamic: actively trying to get into the systems; highly sophisticated
- Defender's objective is to maximize utility, $U_D(q,p,X)$:
  - $U_D(q,p,X) = \Sigma_x (1-q(x))G(x) (1-p(x)) - p(x) (P_S v_S(x;q) + P_A v_A(x;q))$
  - $v_S(x;q) = V(x) (1-q(x))$
  - $v_A(x;q) = max_{x'} \mu(x,x'; q)$

Optimization Problem

$\max_q U_D(q,p,X)$

subject to:

$0 \leq q(x) \leq 1$

$v_S(x;q) = V(x)\ (1-q(x))$
$v_A(x;q) = max_{x'}\ \mu(x,x';\ q)$

Optimization Problem

Sandia National Laboratories

$\max_q U_D(q,p,X)$

subject to:

$0 \leq q(x) \leq 1$

$v_S(x;q) = V(x)\ (1-q(x))$
$v_A(x;q) = max_{x'}\ \mu(x,x';\ q)$

operational budget constraint

$\Sigma_x\ q(x) < c|X|$

Optimization Problem

Sandia National Laboratories

$\max_q U_D(q,p,X)$

subject to:

$0 \leq q(x) \leq 1$

$v_S(x;q) = V(x)\ (1\text{-}q(x))$
$v_A(x;q) = max_{x'}\ \mu(x,x';\ q)$

operational budget constraint

$\Sigma_x\ q(x) < c|X|$

key assumption: *X* (set of all feature vectors) is finite

Optimization Problem (LP)

Sandia National Laboratories

$\max_q U_D(q,p,X)$

subject to:

$0 \le q(x) \le 1$

$v_S(x;q) = V(x)\ (1\text{-}q(x))$
$v_A(x;q) = max_{x'}\ \mu(x,x';\ q)$

operational budget constraint

$\Sigma_x\ q(x) < c|X|$

key assumption: *X* (set of all feature vectors) is finite

(can either use a sample from distribution over x, or batch)

Relationship to “Standard” Learning

Sandia National Laboratories

- Typically people use a threshold on *p(x)* to classify good and bad

Relationship to "Standard" Learning

Sandia National Laboratories

- Typically people use a threshold on *p(x)* to classify good and bad
- **Proposition**: *suppose that there is no budget constraint and only static attackers. Then a threshold policy is optimal.*

Relationship to "Standard" Learning

Sandia National Laboratories

- Typically people use a threshold on *p(x)* to classify good and bad
- **Proposition**: *suppose that there is no budget constraint and only static attackers. Then a threshold policy is optimal.*

- **Proposition**: suppose that there are only static attackers, and a budget constraint. Then the following policy is optimal:
  - rank *x* according to *p(x)*
  - remove all *x* below threshold on *p(x)*
  - set *q(x) = 1* in descending order of *p(x)* until budget is exhausted

Experiments

Sandia National Laboratories

- Setup: TREC data set (50% spam, 50% ham), 2005-2008
- *Experiment 1*: train on 2005, test on ’05-’08, using *q(x)*
- *Experiment 2*: simulate utility-maximizing attacker
- *Experiment 3*: suppose our attacker model is wrong

Testing on Spam Data

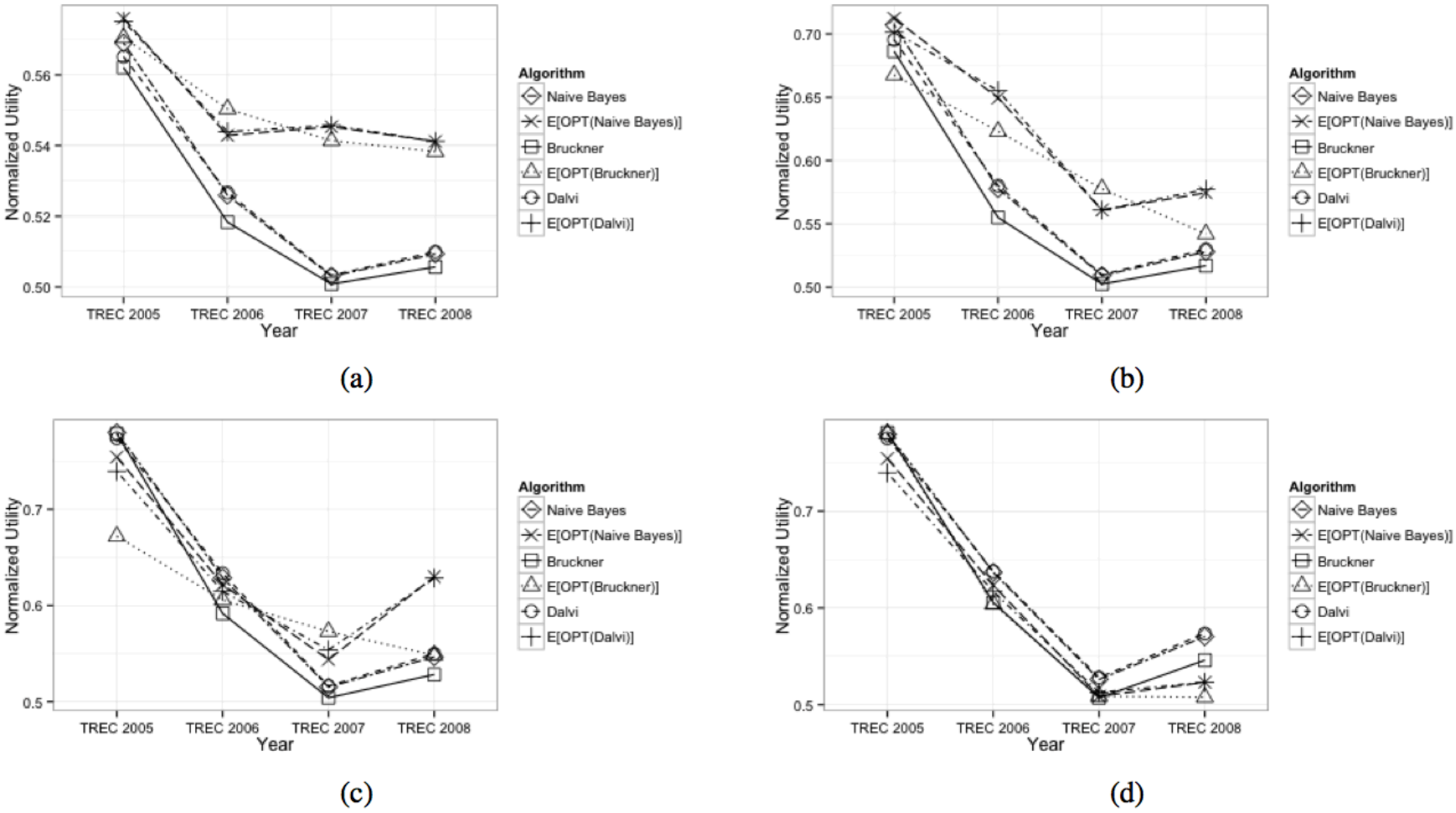
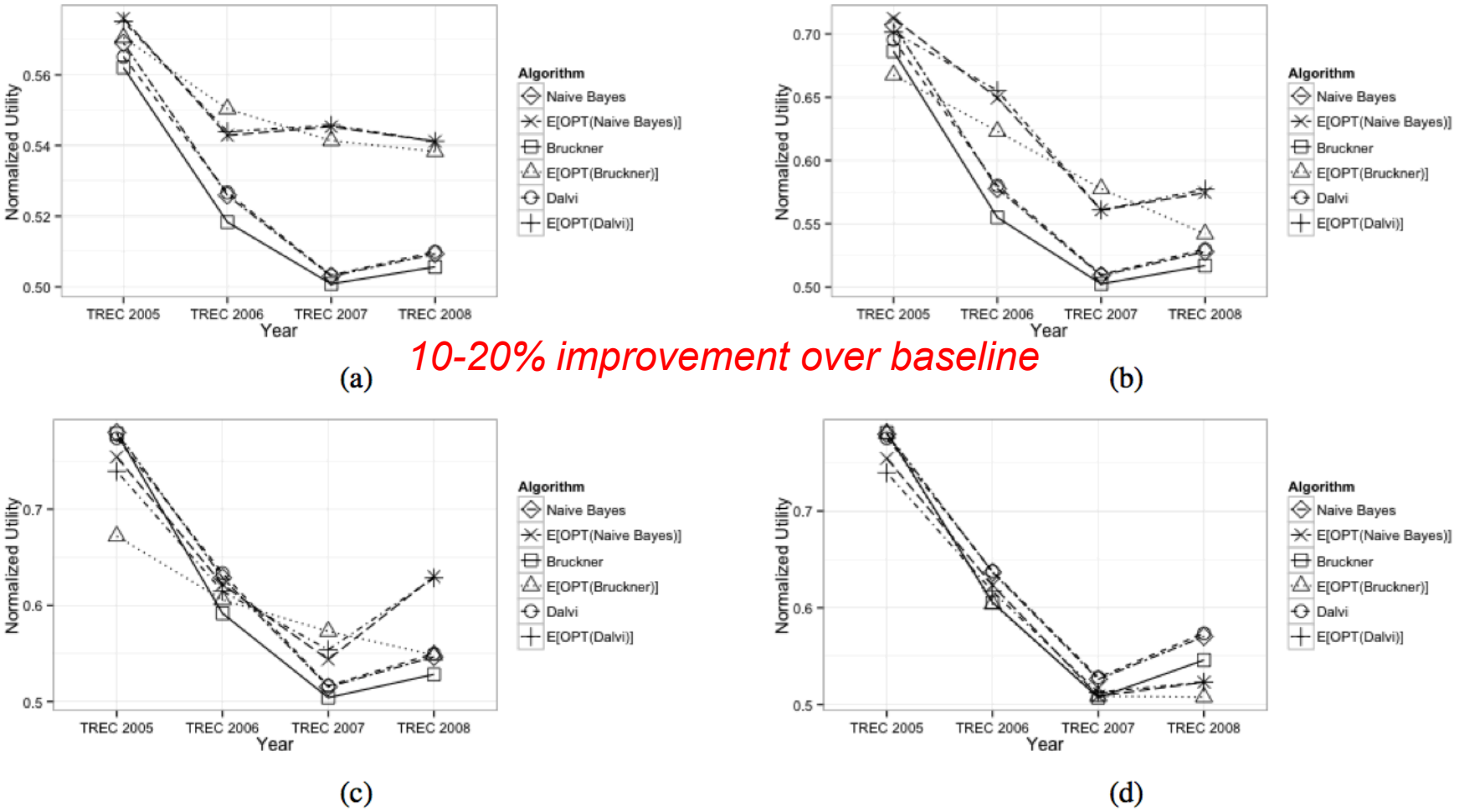


Fig. 2. Comparison of algorithms on TREC data, trained on year 2005, and tested on years 2005-2008. Our approach is labeled as $E[OPT(\cdot)]$, where the parameter is the classifier that serves as our $p(\vec{x})$. We use the following parameters: $\delta = 1$, $V(x) = G(x) = 1, P_A = 0.5$. (a) $c = 0.1$; (b) $c = 0.3$; (c) $c = 0.5$; (d) $c = 0.9$.

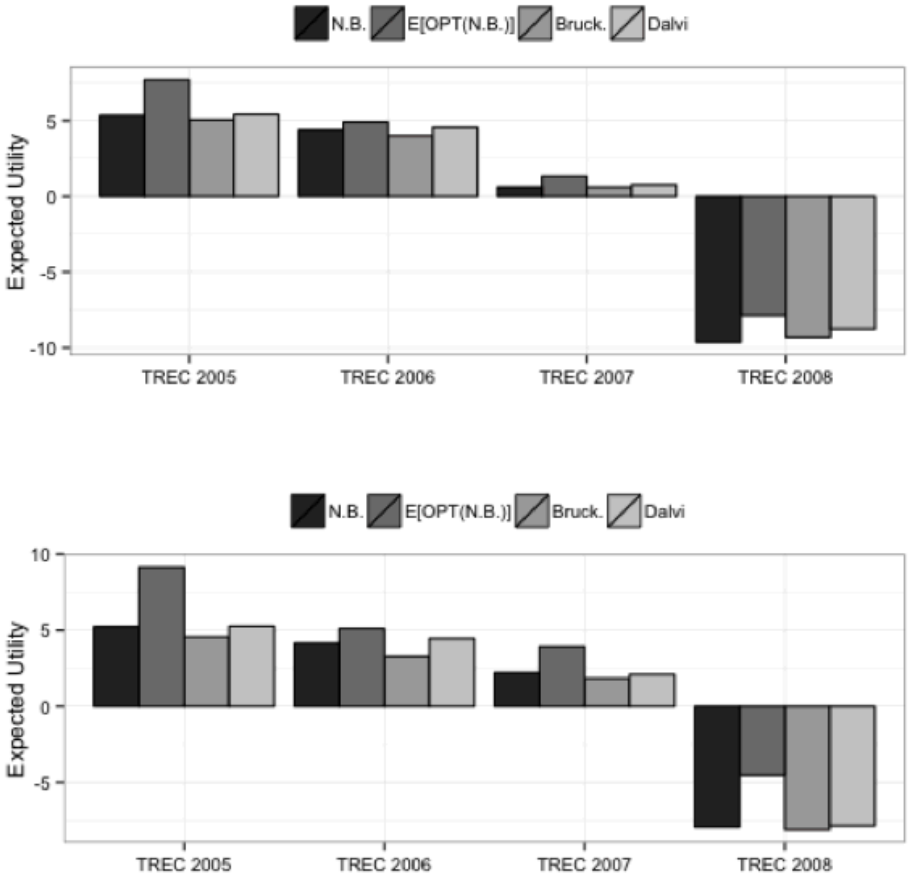
Testing on Spam Data



10-20% improvement over baseline

Fig. 2. Comparison of algorithms on TREC data, trained on year 2005, and tested on years 2005-2008. Our approach is labeled as $E[OPT(\cdot)]$, where the parameter is the classifier that serves as our $p(\vec{x})$. We use the following parameters: $\delta = 1$, $V(x) = G(x) = 1, P_A = 0.5$. (a) $c = 0.1$; (b) $c = 0.3$; (c) $c = 0.5$; (d) $c = 0.9$.

Against a utility-maximizing attacker

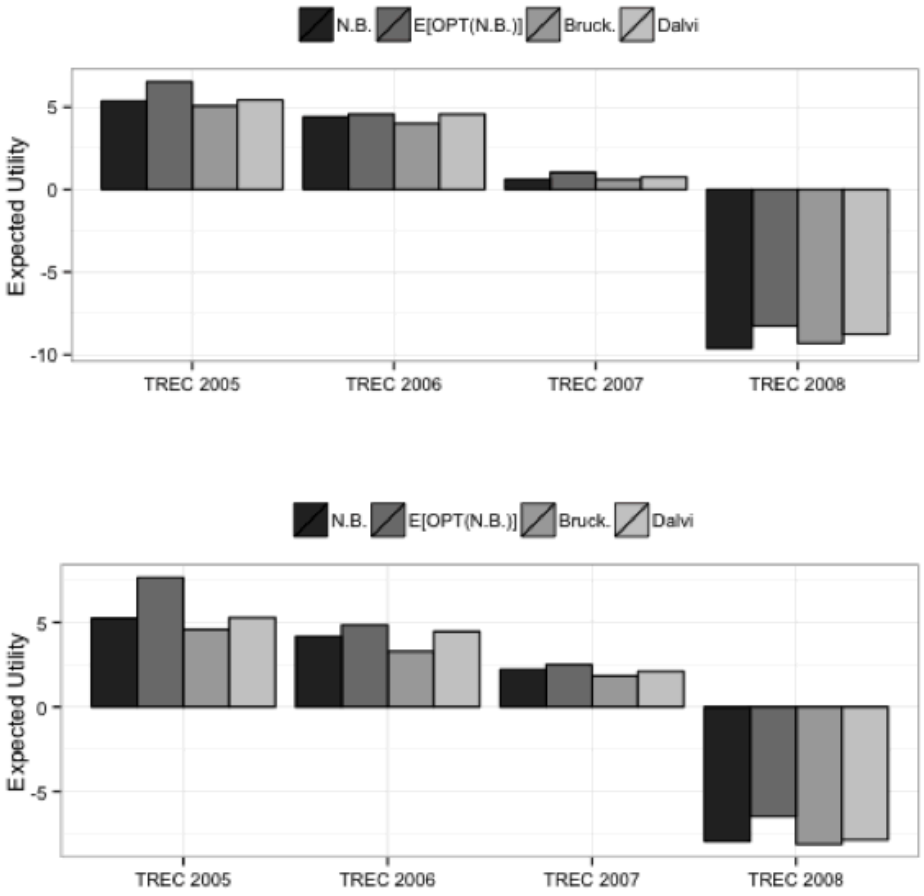


nearly 100% improvement over alternatives in some cases

even only using Naïve Bayes as the "baseline" p(x)

Fig. 5. The expected utilities, assuming $P_A = 1$ and that our attacker model is correct, where $p(x)$ is provided by Naïve Bayes; top: $c = 0.1$; bottom: $c = 0.3$.

Errors in attacker model parameters



assume error ~30%

almost no effect (robust to errors)

Fig. 8. The expected utilities, assuming $P_A = 1$, that our attacker model is correct, but allowing for errors in parameter estimates, and $p(x)$ is provided by Naïve Bayes; top: $c = 0.1$; bottom: $c = 0.3$.

Wrong attacker model

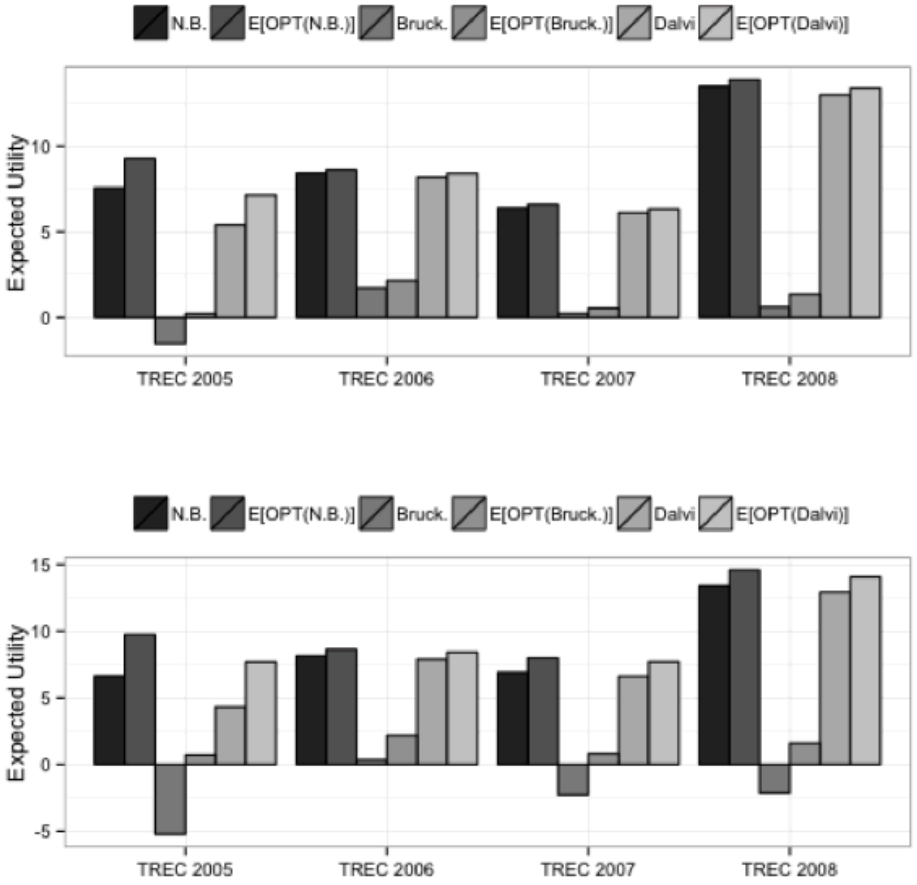


Fig. 10. The expected utilities, assuming $P_A = 1$ and that our attacker utility model is incorrect, and the actual utility decays for non-prefered input vectors according to Equation 8; top: $c = 0.1$; bottom: $c = 0.3$.

suppose actual attacker utility exhibits polynomial decay

almost no effect, still far better than other models

Summary

- Dynamic attacker model: use classical (AI) planning formalism to model an attacker
  - Can solve the plan interdiction problem at scale using Bender's
    - (also tried heuristics, which allow us to get similar scalability, but also use less memory)
  - Extension to uncertainty about attackers direct
  - Extension to execution uncertainty: scalability is a problem
- Adversarial machine learning
  - Model an attacker as a utility maximizer
  - Can compute optimal inspection/filtering policy (under budget constraint) using LP
  - Significantly outperforms alternative approaches

Future work

- Scale MDP interdiction problem (current research; using factored MDP representation, linear value function approximation, and state space abstraction techniques)
- Implement (current research; SANDMAN tool implements the threat model, user interface, and plan generation)
- AML approach: assumption of finite/sampled X problematic
  - look into alternative parametric function classes to approximate *q(x)*
- Relax assumption that the attacker "knows" *q(x)*
  - attacker is learning / reverse engineering the defense policy
  - introduces interesting ML theory questions

Collaborators

- Josh Letchford (formerly Duke University; now Sandia)
- John Ross Wallrabenstein (Purdue)