Applied Mechanism Design for Social Good

CMSC828M – Fall 2016 – University of Maryland

Scribe: Katherine Scola

Overview of Lectures 6 and 7: Security Games

The overarching topic of these two lectures is leader-follower games, and security games. Leader-follower games (also known as Stackelberg games) are just games in which one player, the leader, must commit to a strategy before the other players chose an action. Security games area special case of leader-follower games that involve a defender and an attacker. The defender will naturally want to use their available resources to defend as many targets as possible, while attackers want to find the undefended targets and attack them. These games have many real-world applications, such as deciding on the locations of airport checkpoints, patrol routes in harbors, and scheduling Federal Air Marshals. In these notes we will see a more detailed overview of Stackelberg and Security games, plus there practical research papers on the topic.

Details

0.1 Introduction to Stackelberg and Security Games - John Dickerson

In the past few lectures, we have only considered simultaneous play games. In these games, two players would play their actions at same time (ie. two drivers going straight or diverging), with no prior knowledge of the other player's actions. Leader-follower games are fundamentally different from these simultaneous play games in that there are sequential actions.

In leader-follower games, also known as Stackelberg games, a player designated as the leader commits to acting in a specific way, and then the other player, designated the follower, observes the leader's mixed strategy before making a decision. Just like a simultaneous play game can have a Nash equilibrium, it is possible for leader-follower games to has have a Stackelberg equilibrium, which simply refers to a situation in a Stackelberg game where neither player wants to deviate.

For example, consider, a game with the following payoffs:

	Left	Right
Up	1,1	3,0
Down	0,0	2,1

If this were a simultaneous play game, then a pure-strategy Nash equilibrium would be (Up, Left). It is clear that once that position is reached, neither player would want to deviate unless the other player also deviated. However, consider what would happen if the row player were allowed to announce their pure strategy first. The row player could announce that they will always play Down. Then, assuming the column player was rational, they would chose to play Right to maximize their payoff. In this way, if the pure-strategy Stackelberg equilibrium is different to the pure-strategy Nash equilibrium, and the row player can actually increase their payoff by acting as the leader.

However, just like Nash equilibria, Stackelberg equilibria can be either pure-strategy or mixed strategy. To see this, consider what would happen if instead of announcing a pure strategy, the row player decided to announce that they would play Up 49% of the time, and that they would play Down 51% of the time. Then, the column player's best move would by to always play Right. This is still a Stackelberg equilibria.

In economics, the leader is sometimes referred to as the "market leader". In this setting there is usually a firm that has some advantage (such as a technological breakthrough or the ability to buy all assets at the low price before the market can adjust) that allows them to move first. By committing to a strategy early on, this firm induces a Stackelberg game, and can effectively force the other player's to act in a way that benefits the leader. However, since we do not model things like the significant cost of Research and Development, uncertainty over market demand, and the initial marketing costs, occasionally in practice the second player has an advantage, called Second-Mover Advantage.

One interesting question to ask in relation to Stackelberg games would be when there exists a computationally tractable way to find equilibria. Although it turns out that in general finding Stackelberg equilira of a game can be NP-Hard, a solution can be found in polynomial time for some specific cases.

In the case of 2-player, zero-sum games, it doesn't matter whether or not the the row player declares their strategy ahead of time. Regardless of the order of play, they will be trying to minimize the maximum expected utility to the column player. Conversely, if the column player will be trying to maximize the minimum expected utility of the the row player. This means that the minimax strategy is not only a Nash Equilibrium, but a Stackelberg equilibrium as well. Because we already established in lecture four that Nash equilibria of two player zero-sum games can be found in polynomial time via an LP, it is clear that Stackelberg equilibria could be found in the same way.

In the case of 2 player general-sum games, the process for finding a Stackelberg equilibria is a bit more complicated, but still possible in polynomial time. The way this is done is by solving the following LP for every possible column c^*:

$$\begin{array}{lll} \text{Maximize} & \sum_r p_r u_R(r, c^*) & \\ \text{st.} & \sum_r p_r = 1 & \\ & \sum_r p_r u_C(r, c^*) \geq \sum_r p_r u_C(r, c) & \forall c \\ & p_r \geq 0 & \forall r \end{array}$$

Where p_r is a variable that represents the probability the row player (leader) will play row r, $u_R(r, c)$ is a variable that represents the utility the row player receives if (r, c) is played, and $u_C(r, c)$ is a variable that represents the utility the column player receives if (r, c) is played,. Notice that we are trying to maximize the row player's utility, subject to both column optimality, and distributional constraints. After running all the consecutive LPs, all the row player has to do is chose the strategy from the LP with the highest objective.

So, for example, if we were trying to calculate the leader's best move for the game described earlier, we could set the $p_{up} = x$ and $p_{down} = y$. Then, the two LP (corresponding to the left and right columns respectively) would be:

$$\begin{array}{ll} \text{Maximize} & 1x + 0y \\ \text{st.} & 1x + 0y \geq 0x + 1y \\ & x + y = 1 \\ & x \geq 0 \\ & y \geq 0 \end{array} \qquad \begin{array}{ll} \text{Maximize} & 3x + 2y \\ \text{st.} & 0x + 1y \geq 1x + 0y \\ & x + y = 1 \\ & x \geq 0 \\ & y \geq 0 \end{array}$$

Interestingly, if we allow commitment to mixed strategies, it is always weakly better to commit. This is not the case if we only allow pure strategies, though. To see why, consider a game of rock, paper, scissors. If the leader of a game of rock, paper, scissors is forced to announce a pure strategy, then the follower will always be able to win the game by playing that action that beats the leader's announced action. However, if the leader is allowed to announce a mixed strategy, than the leader can announce that they will play each action with probability $\frac{1}{3}$, which will give the same expected utility as the leader would have received if they did not announce their strategy ahead of time.

In Bayesian 2-player, general-sum games, as long as the follower has only one type θ, we can still solve for the leader's optimal action. Like before, the leader will have to solve an LP for every column c^*:

$$\begin{array}{lll}
\text{Maximize} & \sum_\theta \pi(\theta) \sum_r p_{r,\theta} u_{R,\theta}(r, c^*) & \\
\text{st.} & \sum_\theta \pi(\theta) \sum_r p_{r,\theta} u_c(r, c^*) \leq \sum_\theta \sum_r p_{r,\theta} u_c(r, c) & \forall c \\
 & \sum_r p_{r,\theta} = 1 & \forall \theta \\
 & p_{r,\theta} \geq 0 & \forall r, \theta
\end{array}$$

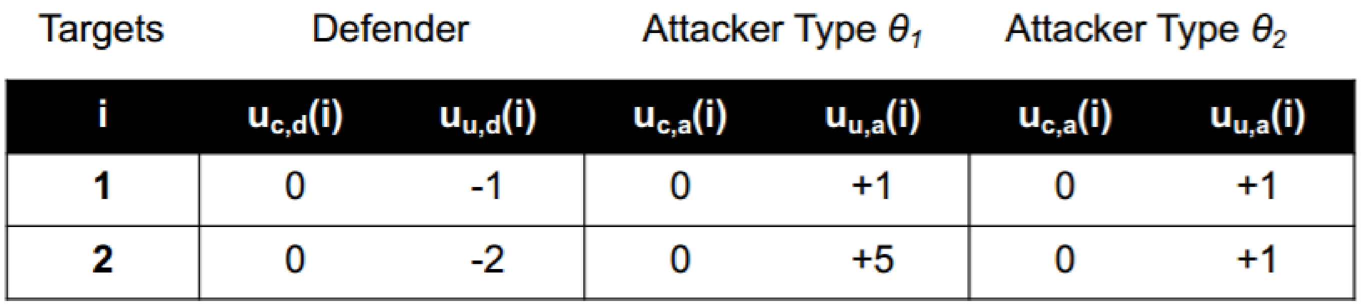
Where $\pi(\theta)$ represents the probability that the leader will be of type θ. In order to have an ex-ante guarentee for this LP, it is important to have some sort of meaningful probability distribution over types. The other variables are the same as they were for the non-Bayesian case, except that they are also defined with respect to the leader's type.

Although we have shown polynomial time algorithms to solve some special cases of Stackelberg games, in general it is NP-Hard to compute. Some models that are NP-Hard to solve for include 2-player general-sum Bayesian games with one-type leader, 2-player general-sum, with general number of types, and any N-player game for $N > 2$.

Of particular interest for these lectures are a special type of Stackelberg game called Stakelberg security games. In security games, the terminology "leaders" and "followers" is usually replaced by defenders and attackers. This is because the leader (defender) is interested in protecting some set of targets, while the follower (attacker) is interested in attacking those targets. A security game can be defined by a three tuple (N, U, M), where N is the set of n targets, U is the utilities associated with the defender and attacker, and M is all subsets of targets that can be simultaneously defended by the defender's limited deployment of resources.

Since the defender might have several resources that allow them to defend (such as more than one security guard), we say a schedule $S \subset 2^N$ is the set of targets defended by a single resource r in the defender's solution. We also say an assignment function $A : R \to 2^S$ is the set of all schedules a specific resource can support. Furthermore, we use the notation $u_{c,d}(i)$ and $u_{u,d}$ to denote the utility that the defender receives when target i is attacked while covered or uncovered respectively. $u_{c,a}(i)$ and $u_{u,a}(i)$ represent the same utilities for the attacker.

As an example, consider a game that involves two targets, 1 and 2. The attacker has a resource R that can be used to used to defend on of the two targets. The defender knows their own utilities, but is unsure which of two types the attack will be. The the game could be represented in a table like this:



Targets	Defender		Attacker Type θ_1		Attacker Type θ_2	
i	$u_{c,d}(i)$	$u_{u,d}(i)$	$u_{c,a}(i)$	$u_{u,a}(i)$	$u_{c,a}(i)$	$u_{u,a}(i)$
1	0	-1	0	+1	0	+1
2	0	-2	0	+5	0	+1

In this example, we see that the defender values target 2 more, and the attacker either is indifferent between the two targets, or highly values target 2. What action would be best for the defender would depend on a number of factors, such as the probability that they believe the attacker might be one type over the other.

These security games have a lot of real world applications, as seen in the next three sections. If we are allowed to tie break in favor of the defender (called a strong Stackelberg equilibria), then a solution always exists. This is a fair assumption to make in practice, because the defender usually has ways of "nudging" the adversary in their favor. However, as the next three sections make clear, computation and uncertain can be practical problems that make finding optimal solutions difficult.

0.2 "Computing Optimal Randomized Resource allocations for massive security games" - Neal Gupta

This first paper deals with problems surrounding the equilibria of large security games. As in other security games, the defender has m resources that they must use to cover n targets, with $m < n$. The defender can commit to a mixed strategy, and after the attacker observes the probabilities for each coverage set, the attacker will chose a pure strategy. Then the utility that the defender gets from target t if it is attacked would be $U^c_\Theta(t)$ if t was covered, or $U^u_\Theta(t)$ if t was uncovered. Similarly, the utilities that the attacker would receive for a covered or uncovered target t would be $U^c_\Psi(t)$ and $U^u_\Psi(t)$ respectively.

For this model, we assume that the players are risk neutral, and that there is only one type of follower. In theory this should be in P, but if the problem is large, we often still cannot solve it efficiently. The number of pure strategies can be unreasonably large for some instances of the problem because we must consider all possible subsets that could be attacked. For example, if there are 100 targets, and 10 resources, the the number of pure strategies that the attacker could chose from is around 17 trillion- a number that is much to huge to be practical.

Because of the large size of these security games, an extensive representation would be too big. Instead we will formulate a compact representation that will make the problem more manageable. We say that a defender commits to a mixed strategy $\Delta = (\delta_{12}, \delta_{13}, \delta 14, ...)$ such that for all indexes i and j, $0 \leq \delta_{ij} \leq 1$ and $\sum_{i,j} \delta_{ij} = m$. In general this will be length $\binom{n}{m}$, because the vector Δ will have to cover every possible allocation of resources by the defender. Them the attacker strategy can be given using an efficient algorithm, which given any mixed strategy Δ, computes the best target $\arg\max_{t \in \Gamma(\Delta)} U_\Theta(\Delta, t)$ such that $\Gamma(\Delta) = \{t : t \in \arg\max U_\Psi(\Delta, t)$. The key insight here is that the only information needed to represent the defender strategy is the probabilities that a target is covered. With one attacker, any feasible assignment of resources that achieves a coverage vector C will be a Stackelberg Security Equilibrium that is weakly optimal.

The first way to solve for these games would be the ERASER method's basic formulation, which can be seen below:

$$\begin{array}{lllr}
\max & d & & \\
\text{st.} & \sum_{t \in T} a_t = 1 & & (1) \\
& \sum_{t \in T} c_t \leq m & & (2) \\
& d - U_\Theta(t, C) \leq (1 - a_t)Z & \forall t \in T & (3) \\
& 0 \leq k - U_\psi(t, C) \leq (1 - a_t) & \forall t \in T & (4) \\
& U_\Theta(t, C) = c_t U^c_\Theta(t) + (1 - c_t) U^u_\Theta(t) & & (5) \\
& a_t \in \{0, 1\} & \forall t \in T & (6) \\
& c_t \in [0, 1] & \forall t \in T & (7) \\
& c_t \leq m & \forall t \in T & (8)
\end{array}$$

Here constraints (1) and (6) ensure that exactly one attack occurs, constraints (2) and (7) ensure that all resources are used in a feasible way. Constraint (3) is a constraint only one the attacked target, and defines d. Constraint (4) is defines K to be an upper bound on the attacker's utility, and says that the attacker's utility must be at least k for the attacked target.

The next way to solve these types of large security games would be the ORIGAMI-Iterative Algorithm. It is an iterative algorithm with an LP similar to ERASER, except that it relies the additional assumption that attackers like it when attacks succeed, and defenders like it when attacks fail. This means that there is no marginal benefit for either player for adding coverage to targets outside of the attack set, and because of the tie-break rule, adding targets from the attack set to the set of covered targets will always benefit the defender. In practice this is almost always the case, so it makes sense to include it in our LP.

The MILP for ORIGAMI is as follows:

$$\begin{array}{lll}
\min & k & \\
\text{st.} & \sum\limits_{t \in T} c_t \leq m & \\
& U_\Psi(t, C) \leq k & \forall t \in T \\
& k - U_\Psi(t, C) \leq Z(1 - \gamma_t) & \forall t \in T \\
& U_\Psi(t, C) = c_t U_\Psi^c(t) + (1 - c_t) U_\Psi^u & \\
& \gamma_t \in \{0, 1\} & \forall t \in T \\
& c_t \in [0, 1] & \forall t \in T \\
& c_t \leq \gamma_t & \forall t \in T
\end{array}$$

Where γ_t represents and indicator variable for the attack set. The other variables and constraints follow roughly the same pattern as in the ERASER LP.

In the ORIGAMI algorithm, it can be showed that if $\hat{U}_\Psi(C) = x$, then $c_t \geq \frac{x - U_\Psi^u(t)}{U_\Psi^c(t) - U_\Psi^u(t)}$ for every target t with $U_\Psi^u(t) > x$. This is because if there was less coverage, the attacker would always attack and get strictly better utility.

The last way that the paper mentioned to solve large security games was the ERASER-C method. This method is a more realistic version of the basic ERASER method, which allows resources to have a set of schedules they can work with by giving each one it's own type. In practical applications of security games, such as scheduling the Federal Air Marshal Service (FAMS) to patrol flights, this is a reasonable assumption to make. An air marshal cannot be on multiple flights at the same time, and must follow a path that does not require them to jump instantaneously between cities. Each air marshal's type would be their starting location, and depending on that type they could patrol some schedule of flights that start there.

In the general problem, a naïve representation of this would be infeasibly large, because there could potentially be a huge number of schedules for each resource. However, in practical applications, it is often manageable. For example, the FAMS scheduling problem, schedules are only of length two.

The ERASER-C MILP is below:

$$\begin{array}{lll}
\max & d & \\
\text{st.} & \sum\limits_{t \in T} a_t = 1 & \\
& \sum\limits_{\omega \in \Omega} q_s & \forall s \in S \\
& \sum\limits_{s \in S} q_s M(s, t) = c_t & \forall t \in T \\
& \sum\limits_{s \in S} h_{s,\omega} C_a(s, \omega) \leq \mathcal{R}(\omega) & \forall \omega \in \Omega \\
& h_{s,\omega} \leq C_a(s, \omega) & \forall s, \omega \in S \times \Omega \\
& d - U_\Theta(t, C) \leq (1 - a_t) Z & \forall t \in T \\
& 0 \leq k - U_\psi(t, C) \leq (1 - a_t) & \forall t \in T \\
& a_t \in \{0, 1\} & \forall t \in T \\
& c_t \in [0, 1] & \forall t \in T \\
& q_s \in [0, 1] & \forall s \in S \\
& h_{s,\omega} \in [0, 1] & \forall s, \omega \in S \times \Omega
\end{array}$$

The only potential issue with this MILP is that it could give a solution that is not feasible using the schedule because of odd cycles. This can be fixed with some additional constraints, but with FAMS, the schedules of size 2 ensure that are are no odd cycles.

Finally, these algorithms have not just been analyzed theoretically- they have also been tested in practice. Experiments have shown that in many cases, these algorithms offer huge improvements in computational scalability over previous algorithms, and will offer additional ways for patrols to solve large security games.

0.3 "TRUSTS: Scheduling Randomized Patrols for Fare Inspection in Transit Systems Using Game Theory" - Kiran Javkar

This paper talks about a specific type of security game that takes place on public transit systems. In these transit systems, passengers are required to purchase tickets to ride, but since it is impossible to check every possible passenger's ticket, the patrol units must come up with some schedule for inspecting passengers and imposing fines for fare evasion. This setting is different from some other Stackelberg settings not only because of the large number of players, but because the defender will often have massive temporal and spatial constraints, so the patrol strategies have to be easy to execute.

The paper specifically focuses on the LA metro and TRUSTS, which stands for Tactical Randomization for Urban Security in Transit Systems. On the LA metro, the LA sheriffs department (LASD) sends uniformed patrols on board trains and stations to check tickets. The followers are the approximately 300,000 people who ride the LA metro system every day, and they can partially observe the LASD's strategy before deciding whether or not to buy a ticket. Both parties want to maximize the amount of money they have- the patrols by collecting as many fines as possible, and the passenger by avoiding fines. It is a zero-sum game because the utilities are shared- if the passenger is caught without a ticket, they will lose an amount of money equal to the amount the defender receives.

Formally, we model the train system as a directed transition graph $G = (V, E)$ according to the metro timetable. We let each vertex $v = (s, t)$ represent some station s at a time t. There is an edge from (s, t) to (s', t') if s and s' are consecutive stops for some train in the train schedule or $s = s', t < t'$, and there is no vertex from (s, t'') with $t < t'' < t'$. Each edge e has a patrol action duration l_e, and an effectiveness value f_e, which measures the likelihood of a passenger being inspected on that edge.

The LASD has γ deployable units, each with a patrol duration of at most κ hours. These patrols can be used either for on-train or in-station inspections, but the schedule they are given must feasible in the sense that they can travel from one part of their patrol to another in the required amount of time.

After the LASD choses a defense path, the passengers have a choice to buy as ticket at cost ρ, or risk paying the fine for fare evasion at cost τ. In order to encourage people to buy tickets, τ should be much larger than ρ. Each passenger has a type Λ determined by their path, chosen from the set of passenger types. Given a strategy of γ patrol units, the inspection probability for a rider of type $\lambda \in \Lambda$ would be $\min(1, \sum_{i=1}^{\lambda} \sum_{e \in P_i \cap \lambda} f_e)$.

It is clear from this formulation that this is Bayesian Stackelberg game with 1 type of leader, and a follower of multiple types. Just like other Bayesian Stackelberg games, we can try to find the leader's best move via LP. The basic formulation of the LP is:

$$\begin{array}{lll}
\max\limits_{x,u} & \sum\limits_{\lambda\in\Lambda} p_\lambda u_\lambda & \\
\text{st.} & u_\lambda \leq \min(\rho, \tau \sum\limits_{e\in\lambda} x_e f_e) & \forall\lambda\in\Lambda \\
& \sum\limits_{v\in V+} x_{(v+,v)} = \sum\limits_{v\in V-} x_{(v,v-)} \leq \gamma & \\
& \sum\limits_{(v',v)\in E} x_(v',v) = \sum\limits_{(v,v+)\in E} x_{(v,v+)} & \forall v\in V \\
& \sum\limits_{e\in E} l_e x_e \leq \gamma\kappa & \\
& 0 \leq x_e \leq \alpha & \forall e\in E
\end{array}$$

Where x_e captures whether or not there are patrols on a particular edge e, and u_λ is the expected amount paid by a rider of type λ. $\alpha \in [0,1]$ is used as an upper-bound on x_e, and limits the number of patrol units on each edge.

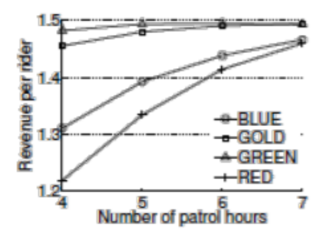
This basic formulation works on maximizing the expected total revenue from rides of type λ, subject to feasibility constraints such as conservation of flow. It does not solve the problem optimally, but this LP can be used as a way to overestimate the actual inspection probability, and thus the defender's utility. However, this LP has two fundamental issues. The first is that it fails to satisfy patrol length limit constraints. The second is that it could lead to patrol units frequently switching between trains or stations. This can be a problem because eerytime a officer needs to make a switch, they will have to stop what they are doing and get started on a different patrol, which could be inefficient. In addition, if an officer is scheduled to be on a train patrol, with frequent switching there is a chance an officer to miss their train while transferring, effectively wasting that resource.

To fix some of these problem, we can use an extended formulation of the LP, which was the main result of the paper. The extended LP is as follows:

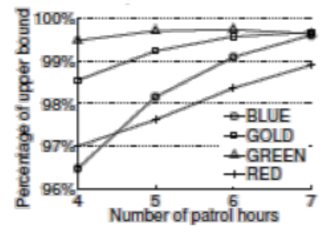
$$\begin{array}{lll}
\max\limits_{x,y,u} & \sum\limits_{\lambda\in\Lambda} p_\lambda u_\lambda - \beta \sum\limits_{e\in E} c_e y_e & \\
\text{st.} & u_\lambda \leq \min(\rho, \tau \sum\limits_{e\in\lambda} x_e f_e) & \forall\lambda\in\Lambda \\
& \sum\limits_{v\in V+} y_{(v+,v)} = \sum\limits_{v\in V-} y_{(v,v-)} \leq \gamma & \\
& \sum\limits_{(v',v)\in E} y_(v',v) = \sum\limits_{(v,v+)\in E} y_{(v,v+)} & \forall v\in V \\
& x_e = \sum\limits_{e'\in\Gamma(e)} y'_e & \forall e\in E \\
& 0 \leq x_e \leq \alpha & \forall e\in E
\end{array}$$

In this formulation, the term being subtracted from the objective function is used to penalize the switching edges. c_e will be 1 for switching edges, and 0 otherwise. It will still give an overestimate for the problem, because there is a possibility that a passenger's fine will be counted more than once by the LP. However, this provides a tighter upper bound on the optimal revenue for defenders than the basic formulations was, and is more realistic because it takes the real-world penalty for frequent switching to account.

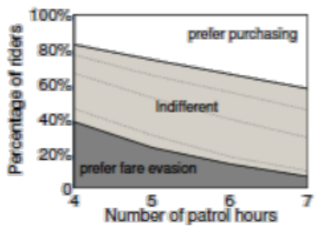
Finally, as can be seen in the original paper, it is clear that this works not only in theory, but also in practice. Experimentally, the LP was tested for four data sets, with one for each LA Metro Rail line, and the following graphs show a visual representation of these results.



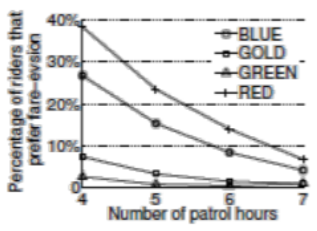
(a) Per passenger revenue of the computed mixed strategy.



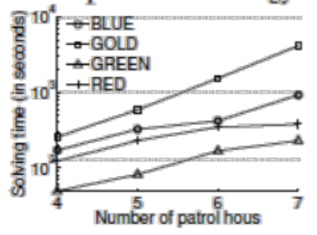
(b) Percentage vs. upper bound.



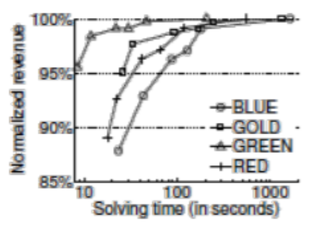
(c) Evasion tendency distribution of Red line.



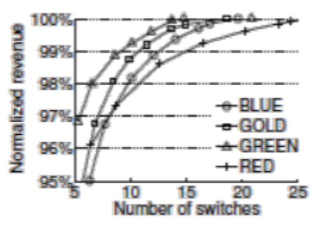
(d) Percentage of riders that prefer fare evasion



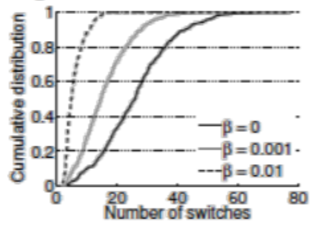
(e) Runtime of solving the LP by CPLEX.



(f) Tradeoffs between optimality and runtime.



(g) Tradeoffs between optimality and patrol preference.



(h) Cumulative probability distribution of the number of switches for the Red line.

One question that was discussed in class was the method that was used for to collect data on the percentage of people who evade tickets. In practice, the LASD most likely uses data from collected fines, however, this has the possibility of introducing bias into the figure. Exactly how they came up with the percentage, and how they might or might not correct for bias is unclear.

0.4 "Deploying PAWS: Field Optimization of the Protection Assistant for Wildlife Security" - Rock Stevens

PAWS is stands for Protection Assistant for Wildlife Security, and is an organization that works to try to defend against animal poaching. The battle between poachers and PAWS patrol units induces a Stackelberg security game with respect to animal densities, where the poachers are the attackers and patrol units are the defenders. A grid is drawn around the area that the animals live on, and each block of the grid is considered a target.

We start with the assumption that poaching is a zero-sum game, because both attackers and defenders value the endangered animals with equal utility. We say that the defender's strategy can be sored in a vector C, where each entry c_i of the vector is the probability that the resource is deployed at target i. Then, the utilities for the attacker and the defender, respectively, at each target are:

$$U_i^a = c_i U_{p,i}^a + (1 - c_i) U_{r,i}^a$$

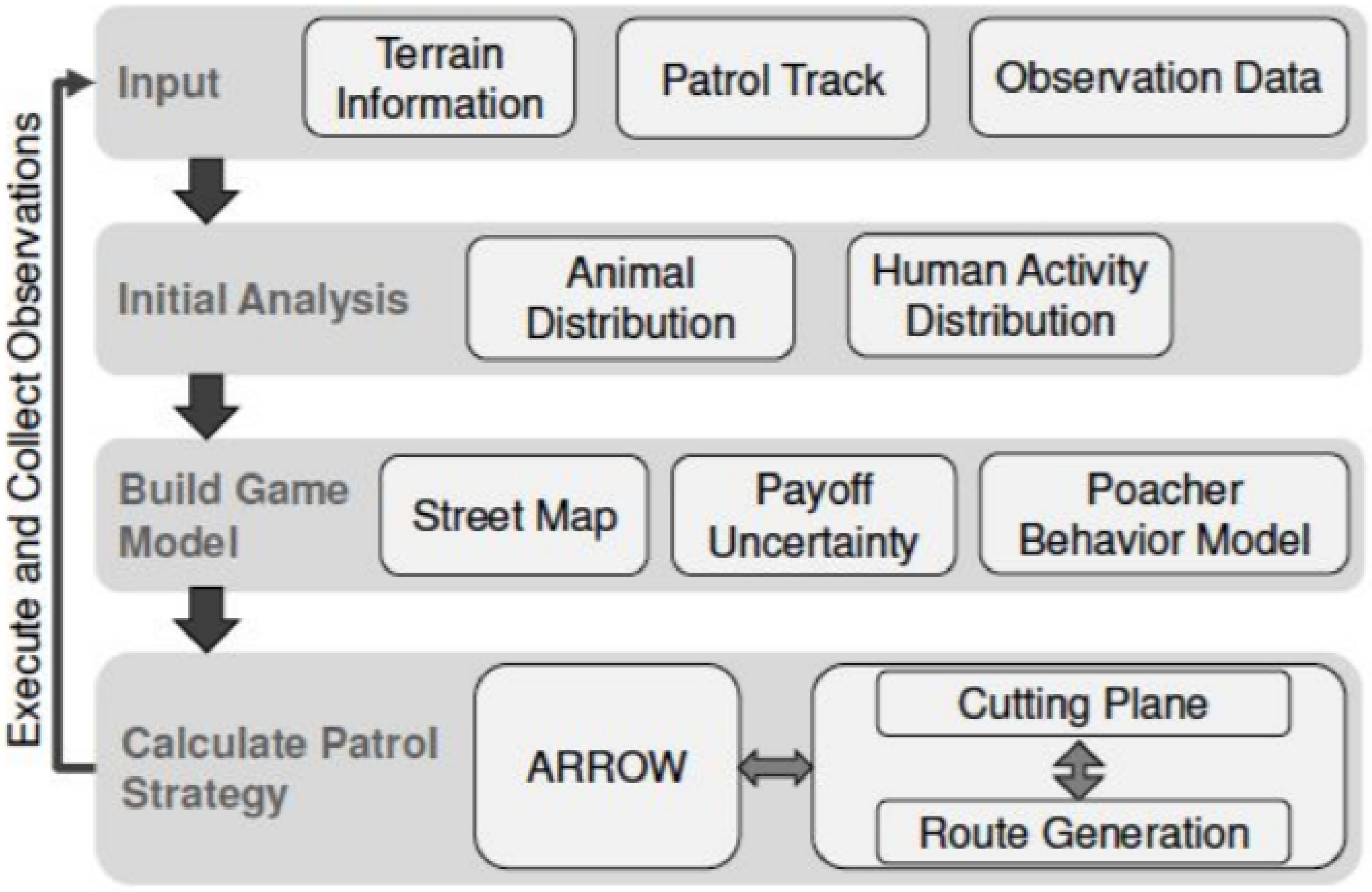
$$U_i^d = c_i U_{r,i}^d + (1 - c_i) U_{p,i}^d$$

Where $U_{p,i}^a$ is the attacker's penalty if i is defended, $U_{r,i}^a$ is the attacker's reward if i is undefended, $U_{r.i}^d$ is the defender's reward if i is defended, and $U_{p,i}^d$ is the attacker's if i is undefended. These rewards and penalties are determined by the animal density at target i.

Initially PAWs would calculate their best strategy by looking at the grid of animal densities, calculate every possible route that the patrols could takes, and then chose the best one. This has several flaws- including the obvious problem that the algorithm is exponential and therefore not scalable for larger versions of the problem. In addition, the PAWS algorithm assumed that there were known, fixed animal densities within the region they are trying to protect, which is an unreasonable assumption to make because animals move around, and defenders will not always have perfect information. It also ignore very practical concerns when designing routes, such as the topology of the land, and human scheduling constraints.

Now PAWS has improved their algorithms to fix some of the problems that the original algorithm had. Now before the algorithm decides on a path, it looks at each 50m x 50m space on the grid to determine which areas have passable topology, and marks key access points for the patrols to cross. After the key access points are marked, PAWS can locally connect the access points with weighted distance metrics that can be used to evaluate different paths. In addition, PAWS uses its topology data to make better estimates about where endangered animals will be, and uses the concept of minimax regret to handle the uncertainty they do have able where animals are.

Perhaps the most interesting update to computer scientists is that the PAWS algorithm is now scalable. Rather than just checking every possible path, it uses iterative, self adjusting algorithms to calculate patrol paths more accurately and efficiently. Roughly, the updated algorithm has four phases- input, initial analysis, building the game model, and calculating the patrol strategy. Then, as patrols execute their routes and collect observations, they feed that information back into the algorithm to adjust. Below is a flow chart that shows a more detailed representation of what PAWS does:



This algorithm is not only interesting theoretically, but it also worked well in practice. When tested on 4-5 day patrols with 3-7 patrollers, it seemed to be incredibly reliable. If PAWS can be widely put into practice, it could end up making a positive impact of the world, and save endangered animals from harmful poaching.

Although it wasn't covered in the paper, during class we also discusses some ways to potentially make PAWS better in the future. One way to do this is to consider more logistical constraints that were still not taken into account. For example, it might be useful to consider what how much food and water the patrols need to bring with them for each path, and factor that inot determining which paths are best. In addition, we could consider things like the availability of GPS signals within a region, since in certain arts of the terrain the GPS patrols use to navigate might have a higher rate of going out.

Another thing to consider is the possibility of re-evaluating the what is meant when we say an area is "defended". Currently PAWS 2.0 says that the area is defended if they have patrols cover at least 50% of the uncertainty area to look for snares and traps. In reality, this might not be an effective definition, because if poachers see a patrol unit removing a snare, they can just go back after the defenders have left and put up their traps again. In addition, in stead of covering 50% of the area, it might be more effective to say an area is defended if all traps and snares are removed.

Also, it might be useful to consider that attention is a finite resource for people. As patrol units go about the day, they will inevitably become tired and less attentive as time goes on. Perhaps in the future, algorithms could be developed that would take this into account, and find some way to make sure patrols aren't inattentive during more important parts of the route. Lastly, the weather of the area might affect how much patrol units could do, and it might be interesting to add that as an overlay.

After the talk, some interesting points were bought up. First, we discussed the assumption PAWS makes that poaching is a zero-sum game. The results in this paper do rely on the this assumption, and the point was made that in the real world that may or might not be true. The argument could be made that a successful snare has more value to a poacher than removing a snare has for a defender. Even if we assume that a snare or a trap is a guaranteed catch for the poachers, is it true that everyone values the animals in the same way? As PAWS stands now, however, it does rely on the assumption that poaching is zero-sum.

Another potential issues that was brought up was the cost of hiring patrol units. We will have to pay someone to actually go on the patrols that are decided on, but that cost is not explicitly included in the payoffs. However, the algorithm's scheduling constraints work to solve this problem in a slightly indirect way. Once it is decided how many days of patrol coverage can be afforded, the algorithm will work to optimize a route for that number of days.

Lastly, we discussed the poachers, and how that is taken into account in the PAWS algorithm. At some point, the poachers might give up on placing snares if the area is too well-defended, but that is not considered by PAWS. In the future, it might be interesting to also consider poachers as people with a finite amount of resources and attention. It is also interesting to consider the fact that defenders are only trying to stop the poaching rather than catch the poachers themselves. The algorithm would most likely be very differently if poachers could be caught immediately and taken away from the area.

References

[1] Fang, F., Nguyen, T. H., Pickles, R., Lam, W. Y., Clements, G. R., An, B., Singh, A., Tambe, M., & Lemieux, A. (2016). Deploying PAWS: Field optimization of the protection assistant for wildlife security. In *Proceedings of the Twenty-Eighth Innovative Applications of Artificial Intelligence Conference*.

[2] Kiekintveld, C., Jain, M., Tsai, J., Pita, J., Ordóñez, F., & Tambe, M. (2009, May). Computing optimal randomized resource allocations for massive security games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1* (pp. 689-696). International Foundation for Autonomous Agents and Multiagent Systems.

[3] Yin, Z., Jiang, A. X., Tambe, M., Kiekintveld, C., Leyton-Brown, K., Sandholm, T., & Sullivan, J. P. (2012). TRUSTS: Scheduling randomized patrols for fare inspection in transit systems using game theory. *AI Magazine*, 33(4), 59.