Fighting m-criminals with Bayesian Stackelberg games

Nimalan Mahendran

April 22, 2009

Abstract

Bayesian Stackelberg games have been used to compute optimal patrol strategies for the security of the Los Angeles International Airport (LAX) by modelling the security situation as a 1-leader, 1-follower, Bayesian game where there is one leader type and an arbitrary number of follower types. A previous approach, named DOBSS, computes the Stackelberg solution for this game by solving a Mixed Integer Linear Program (MILP). This paper shows that the action-graph game representation would be helpful in speeding up the MILP algorithm and also make it possible to scale the MILP approach to n-followers, each with an arbitrary number of follower types, as long as the followers are unable to affect each other by their actions.

1 DOBSS and LAX

Game-theoretic analysis is particularly well-suited to security domains, given the self-interested and strategic nature of crime and crime fighting. The adversarial scenario can be modelled as a game and then useful questions can be answered by computing the appropriate solution concept. However, there are many adversarial scenarios where computing the appropriate solution concept is intractable. The nature of the solution concept and the lack of structure in the adversarial scenario are factors that contribute to the intractibility.

[PPM+08a] describe an adversarial scenario where a single crime fighter must guard against the threat of criminals by patrolling areas under its protection. The crime fighter's patrolling capabilities are limited, so it must choose a limited subset of areas to patrol, based on the utility it places on protecting certain areas. The criminals base their choice of targets on the patrol strategy that they perceive the crime fighter to be following and on the utlity offered to them by engaging in crime in certain areas.

[PPM+08a] propose a model for the patrol scenario as a Bayesian game with one crime fighter and one criminal, where the crime fighter has only one type, but the criminal could have an arbitrary number of types. An action for the crime fighter in this Bayesian game is to select an area to patrol. An action

for the criminal in this Bayesian game is to select an area to strike. These actions can be randomized over to generate mixed strategies. These mixed strategies are interpreted as patrol and strike policies, where the probability that an action is played in a mixed strategy is exactly the proportion of times that the corresponding area will be patrolled or struck in a patrol or strike policy [PPM+08a].

The criminal is able to observe the patrol policy of the crime fighter and strike with knowledge of this patrol policy. Hence, the crime fighter takes on a leader role and the criminal takes on a follower role. The moves in the Nash equilibrium solution of a Bayesian game are simultaneous, but the moves in this leader-follower scenario are not. The leader-follower scenario can be modelled by the Stackelberg solution [SC73], where the follower selects its optimal strategy after exactly observing the leader's strategy.

There are certain conditions that must hold in the Stackelberg solution [CS06]. The leader must commit to its strategy and cannot change it in light of the follower's strategy choice. The possession of this ability by the leader to commit to a strategy must also be common knowledge.

[PPM+08a] present a method for computing the Stackelberg solution of the crime fighting scenario by expressing it as a Mixed-Integer Linear Program (MILP), an NP-hard optimization problem with algorithms that work well in practice. This method is called the Decomposed Optimal Bayesian Stackelberg Solver (DOBSS). DOBSS was then applied to the ARMOR system that is used for security scheduling at the Los Angeles International Airport.

The Stackelberg solution is presented in Section 2. An explanation of why it is must easier to find a Stackelberg solution to the 1-leader, 1-follower, n-follower-type game than it is to find a Nash equilibrium is given in Section 3. A compact and computationally efficient game representation known as action-graph games and its algorithm for computing expected utility is surveyed in Section 4. The motivation behind using action-graph games in this paper is to be able to exploit structure in the crime fighter's and the criminal's payoffs to speed up DOBSS and to extend the solution beyond the 1-criminal case. An very high-level overview of branch-and-cut MILP solving algorithms is given in Section 5 to illustrate the importance of computing expected utility efficiently. Section 6 proposes a method of using action-graph games to exploit structure in the payoffs to enable solutions of the m-criminal case, where the criminals cannot influence each other through their actions.

2 The Stackelberg solution

The Stackelberg solution to a game is one where one of the players assumes the role of the leader and all other players assume the role of the follower. The leader commits to a strategy and then the followers play the game with knowledge of the leader's strategy. The leader's position might seem like a disadvantaged one, but in fact it is shown in [SC73] that the leader will do no worse by playing a Stackelberg strategy than it would by playing according to the corresponding

Nash equilibrium.

The intuition behind the proof of this claim is that the leader is able to use its commitment to a strategy as a means of selecting the the best response of the follower that is the most agreeable to the leader. A Nash equilibrium is simply a choice made in the same manner, expect that the additional constraint that the leader's strategy choice be a best response to the follower's best response, is also imposed.

The following game,

	c	d
a	2, 1	4, 0
b	1, 0	3, 2

, taken from [PPM$^+$08a], illustrates the Stackelberg solution and the advantage of being a leader. The row player is the leader and the column player is the follower. It can be verified that the only Nash equilibrium is (a, c), by iterated removal of strictly dominated strategies. This Nash equilibrium results in a payoff of 2 for the leader. If the leader commits to playing the pure strategy b, its payoff will be 3, because the follower will maximize its utility by playing d. If the leader commits to playing the mixed strategy $[a : 0.5, b : 0.5]$, its payoff will be 3.5, because the follower's best response is to play d [PPM$^+$08a].

Stackelberg games are solved by computing the best strategy for the leader to commit to.

3 The relative ease of computing the Stackelberg solution

The task for the follower, given the leader's strategy choice, is a single-agent linear programming problem. Therefore, although the leader's utilities are expressed as $u_l(s_l, s_f)$, the dependence of u_l on s_f is unnecessary, as s_f is determined by s_l. The leader's task is to find the optimal strategy s_l, where the expected utility under s_l is $u_l(s_l)$ and the evaluation of $u_l(s_l)$ involves the solution of the single-agent linear programming problem for the corresponding follower strategy s_f[PPM$^+$08a].

The single-agent nature of the choice of s_f, given s_l, is the main reason that solving for the leader's optimal Stackelberg strategy is much easier than solving for a Nash equilibrium.

3.1 Non-Bayesian Games

It is much easier to understand the difference in tractibility in computing the Stackelberg solution as opposed to the Nash equilibrium if the analysis is restricted to 2-person non-Bayesian games.

The Nash equilibrium for 2-person non-Bayesian games can be expressed as a Linear Complementarity Problem (LCP) and solved by the Lemke-Howson algorithm [SLB09]. The LCP, a feasibility program with no objective function,

is given by

$$\sum_{k \in A_l} u_l(a_l^j, a_f^k) s_f^k + r_l^j = U_l^*, \qquad \forall j \in A_l$$

$$\sum_{j \in A_l} u_f(a_l^j, a_f^k) s_l^j + r_f^k = U_f^*, \qquad \forall k \in A_f$$

$$\sum_{j \in A_l} s_l^j = 1, \sum_{k \in A_f} s_f^k = 1$$

$$s_l^j \geq 0, s_f^k \geq 0, \qquad \forall j \in A_l, \forall k \in A_f$$

$$r_l^j \geq 0, r_f^k \geq 0, \qquad \forall j \in A_l, \forall k \in A_f$$

$$r_l^j s_l^j = 0, r_f^k s_f^k = 0, \qquad \forall j \in A_l, \forall k \in A_f \qquad \text{(LCP.1)}$$

where Equation LCP.1, known as the complementarity conditons, are the only non-linear terms [SLB09]. The presence of the complementarity conditions makes solving the LCP much harder than solving a linear program [SLB09].

3.1.1 The MILP Approach of DOBSS

The non-Bayesian Stackelberg game with 1 leader and 1 follower is solved in [PPM+08a] by solving the following Mixed-Integer Quadratic Program (MIQP)

$$\max_{s_f, s_l, u_f^*} \sum_{j \in A_l} \sum_{k \in A_f} u_l(a_l^j, a_f^k) s_l(a_l^j) s_f(a_f^k)$$

$$\text{subject to } \sum_{j \in A_l} s_l(a_l^j) = 1, \sum_{k \in A_f} s_f(a_f^k) = 1$$

$$0 \leq u_f^* - EU_f(s_l, a_f^k) \qquad \forall k \in A_f \qquad \text{(MIQP-1.1)}$$

$$u_f^* - EU_f(s_l, a_f^k) \leq (1 - s_f(a_f^k))M \qquad \forall k \in A_f \qquad \text{(MIQP-1.2)}$$

$$s_l(a_l^j) \geq 0 \qquad \forall j \in A_l$$

$$s_f(a_f^k) \in \{0, 1\} \qquad \forall k \in A_f \qquad \text{(MIQP-1.3)}$$

$$u_f^* \in \mathbb{R}$$

where $EU_f(s_l, a_f^k) = \sum_{j \in A_l} s_l(a_l^j) u_f(a_l^j, a_f^k)$ is the expected utility gained by the follower playing the pure strategy a_f^k and M is a large constant. u_f^* represents the utility of the follower's best pure strategy response to s_l, where the optimality of u_f^* is enforced by constraint MIQP-1.1.

DOBSS simplifies the solution of the non-Bayesian Stackelberg game by searching only for pure strategy best responses for the follower, as indicated by constraint MIQP-1.3. This is a valid restriction because all pure strategies that are in the support of a mixed strategy best response for the follower will have the same expected utility as the mixed strategy itself and the leader is unable, by its commitment to the leader strategy s_l, to exploit the fact that the follower is playing a pure strategy in a Stackelberg solution.

The MIQP was derived in [PPM+08a] by analyzing the optimality conditions of the single-agent linear programming problem faced by the follower, given the leader's strategy. Complementary slackness conditions are important optimality conditions that state that any action in the support of the follower's strategy must be a best response pure strategy to the leader's strategy. These complementary slackness conditions, like the complementarity conditions from the LCP, would make the solution of the resulting Stackelberg game intractable. The restriction to only consider pure strategy best responses allows the complementary slackness conditions to be simplified into constraint MIQP-1.2, turning the problem into a MIQP. Constraint MIQP-1.2 is inactive when $s_f(a_f^k) = 1$ and forces the equality

$$u_f^* - EU_f(s_l, a_f^k) = 0$$

when $s_f(a_f^k) = 0$, and hence is equivalent to the complementary slackness conditions when $s_f(a_f^k) \in \{0, 1\}$.

The MIQP can be expressed as a Mixed-Integer Linear Program (MILP) by replacing the quadratic objective function with a linear one through a change of variables [PPM+08a]. The resulting MILP can then be solved by general-purpose MILP solvers, although the MIQP will continue to be the basis of discussion in this paper, to ease the notation.

The MIQP is a tractable solution for a single follower type, but the Bayesian Stackelberg game used to model the security scenario in [PPM+08a] must handle an arbitrary number of follower types.

3.2 Bayesian Games with one leader type and n follower types

[PPM+08a] show that the Bayesian Stackelberg game with n follower types decomposes into a convex combination of n Bayesian Stackelberg games with 1 follower type for each game by proving that the objective function and constraints in both cases are equivalent. Hence, the MIQP for solving the Bayesian

Stackelberg game with n follower types $T = \{t_1, \ldots, t_n\}$ is given by

$$\begin{aligned}
&\max_{s_f, s_l, u_f^*} \sum_{t \in T} p^t \sum_{j \in A_l} \sum_{k \in A_f} u_l(a_l^j, a_f^k) s_l(a_l^j) s_f^t(a_f^k) \\
&\text{subject to } \sum_{j \in A_l} s_l(a_l^j) = 1 \\
&\qquad \sum_{k \in A_f} s_f^t(a_f^k) = 1 && \forall t \in T \\
&\qquad 0 \leq u_f^* - EU_f^t(s_l, a_f^k) && \forall k \in A_f, \forall t \in T \\
&\qquad {u^t}_f^* - EU_f^t(s_l, a_f^k) \leq (1 - s_f^t(a_f^k))M && \forall k \in A_f, \forall t \in T \\
&\qquad s_l(a_l^j) \geq 0 && \forall j \in A_l \\
&\qquad s_f^t(a_f^k) \in \{0, 1\} && \forall k \in A_f, \forall t \in T \\
&\qquad {u^t}_f^* \in \mathbb{R} && \forall t \in T
\end{aligned}$$

where the quantities involving the follower f must now also be indexed by the follower's type t, where p^t is the probability that the follower will be of type t.

The MIQP for solving the Bayesian Stackelberg game with n follower types can be converted into a MILP by the same change of variables formula and solved by general-purposes MILP solvers.

[PPM+08a] have shown that DOBSS outperforms other methods for solving Bayesian Stackelberg games with 1 leader, 1 follower and n follower types. DOBSS and algorithms for computing Nash equilibria, such as the Govindan-Wilson method and the simplicial subdivision method, have a common bottleneck: the computation of expected utility for an agent playing an action, given the mixed strategies of all other agents.

4 Action-graph games and the expected utility problem

Action-graph games (AGGs), presented in [JLB08] and [SLB09], exploit the underlying structure in non-Bayesian games to compute the expected utility for an agent playing an action, given the mixed strategies of all other agents.

4.1 Representing games with action-graphs

The AGG representation can represent any normal-form game [JLB08], but can represent them much more compactly when the following kinds of structure exist in the agents' utility functions, defined by when they exist,

Strict independence This kind of structure exists when only the identities of two agents are needed to determine whether they can affect each other's utilities through their actions.

Context-specific independence This kind of structure exists when both the identities of and the actions taken by two agents are needed to determine whether they can affect each other's utilities through their actions.

Anonymity This kind of structure exists when only the actions taken by two agents are needed to determine whether they can affect each other's utilities through their actions.

Normal form games represent the utility functions of each agent as a huge lookup take that returns the agent's utility for a full action profile, and hence grows exponentially in the number of agents.

Action graph games represent the utility functions of all agents as a graph $G = (A, E)$, where A is the set of all distinct actions. An agent's utility for an action $a \in A$ is fully determined by the number of agents that choose action a and the number of agents that choose each action $a' \in \nu(a)$, where $\nu(a)$ is the neighbourhood of a in G. The utility of any agent choosing action a in an action graph is $u^a(c(a), c^a)$, where $c(a)$ is the number of agents that chose action a and $c^a \equiv \big(c(a'), a' \in \nu(a)\big)$ is a tuple representing the number of agents that chose each action $a' \in \nu(a)$ [JLB08].

AGGs can be used to represent any normal-form game, despite the restricted form of each action node's utility function. A normal-form game with no structure can be represented by an AGG that has a separate node for every action in every agent's action set. An action node for agent i, a_i, has every action node of every other agent in its neighbourhood and hence $u^{a_i}(c(a_i), c^a) = u^{a_i}(a_i, a_{-i}) = u_i(a_i, a_{-i})$ and u^{a_i} degenerates into an exponential-size lookup table for node a_i.

AGGs, in addition to being space-efficient, also enable the time-efficient computation of expected utility.

4.2 Computing expected utility with action-graphs

DOBSS and algorithms for finding Nash equilibria rely heavily on efficiently computing the expected utility of a particular action a_i for an agent i, given the mixed strategy profile σ_{-i} for all other agents, $V^i_{a_i}(\sigma_{-i})$ [JLB08]. The direct algorithm for calculating this expected utility is

$$V^i_{a_i}(\sigma_{-i}) = \sum_{a_{-i} \in A_{-i}} u_i(a_i, a_{-i}) p(a_{-i}|\sigma_{-i})$$

$$p(a_{-i}|\sigma_{-i}) = \prod_{j \neq i} \sigma_j(a_j)$$

where a_j is agent j's action in action profile a_{-i} [SLB09]. This direct algorithm is exponential in the number of agents [JLB08].

Expected utility can be much more efficiently computed for games with structure when the AGG representation is used. Expected utility computation in AGG exploits strict and context-specific independence through projection and exploits anonymity through a dynamic programming approach [JLB08].

4.2.1 Strict and context-specific independence

The utility function for action a_i has the form $u^{a_i}(c(a_i), c^a)$, implying that the value of $V^i_{a_i}(\sigma_{-i})$ does not depend on the states of the action nodes that are outside $\nu(a_i)$. Therefore, the action nodes in $A \setminus \nu(a_i)$ are mapped to a dummy node $\emptyset$ [JLB08]. Assuming for simplicity that all of the agents' action sets have the same size α and that there are q agents, the expected utility computation is a sum over $O(\alpha^{q-1})$ terms in the normal-form game, but is a sum over $O((|\nu(a_i)|+1)^{q-1})$ terms in AGG representation [JLB08].

If $|\nu(a_i)|$ is small, this is amounts to a large speedup over normal-form games.

4.2.2 Anonymity

The expected utility computation, as presented so far, still takes exponential time in the number of agents. The exponential time complexity results because each utility $u^{a_i}(c(a_i), c^{a_i})$ must be weighted by the probability $p(c(a_i), c^{a_i})$ that $c(a_i)$ agents could be playing a_i and c^{a_i} agents could be playing in each respective node in $\nu(a_i)$ [JLB08].

Assume for simplicity that exactly q' agents, not including agent i, could play a_i and any action in $\nu(a_i)$. Then the number of different configurations of a_i and $\nu(a_i)$ that $p(c(a_i), c^{a_i})$ would have to be computed for, without taking anonymity into account, would be $O((|\nu(a_i)|+1)^{q'})$, but the number of different configurations would be only be $O(q'^{|\nu(a_i)|+1})$, if anonymity were taken into account [JLB08].

[JLB08] provide a dynamic programming algorithm to compute every value $p(c(a_i), c^{a_i})$ efficiently by making use of anonymity.

4.3 Can action-graph games be used to speed up DOBSS?

DOBSS involves the execution of a MILP solver on a program equivalent to the MIQP described previously. The expected utilities in the MIQP have the following form

$$\begin{aligned} EU_l(s_l, s_f) &= \sum_{t \in T} p^t \sum_{j \in A_l} \sum_{k \in A_f} u_l(a_l^j, a_f^k) s_l(a_l^j) s_f^t(a_f^k) \\ &= \sum_{t \in T} p^t \sum_{j \in A_l} u_l(a_l^j, a_f^{k^t}) s_l(a_l^j) \\ EU_f^t(s_l, a_f^k) &= \sum_{j \in A_l} u_f(a_l^j, a_f^k) s_l(a_l^j) \end{aligned}$$

where $a_f^{k^t}$ is the pure strategy selected by a follower of type t. The ability of AGGs to exploit anonymity does not provide any gains in efficiency for DOBSS because only two players are involved.

Patrolling is a task that occurs in a spatial domain, with equivalent action sets for the crime fighter and the criminal, so it is very likely that an action node's neighbourhood is much smaller than the total number of actions, and

hence a speedup should be provided by the ability of AGGs to exploit context-specific independence. However, since the follower only needs to choose pure strategies, the expected utility computations only involve summations over the leader's action set.

Therefore, it seems as though the AGG representation cannot offer significant speedups in expected utility computation for DOBSS, but a deeper look at the workings of a broad class of MILP-solving algorithms reveals a different story.

5 Mixed-integer linear programming and the importance of efficient expected utility computation

A mixed-integer linear program (MILP) has the form

$$\begin{aligned} \min_x \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \in (\mathbb{Z}^{n-p}, \mathbb{R}^p) \end{aligned}$$

and where $p = n$, this corresponds to a linear program (LP) [FM05]. An LP relaxation of a MILP is obtained by removing all constraints in the MILP that require variables to be in $\mathbb{Z}$.

MILPs are closely related to LPs, as illustrated by the pseudocode given in [FM05] of a broad class of MILP solvers known as cutting-plane methods.

Algorithm 1 Cutting-plane method to solve $MILP$

```
k = 0
LP^0 is the LP relaxation of MILP
x̃^0 be the solution of LP^0, obtained by an LP solver
while x̃^k ∉ (ℤ^{n−p}, ℝ^p) do
    Find a cutting plane a_cut^T x = b_cut, defined as a linear inequality that is
    satisfied by all feasible solutions to MILP, but not by x̃^k.
    Add a_cut^T x = b_cut to LP^k to obtain LP^{(k+1)}
    k = k + 1
    Solve LP^k for x̃^k with an LP solver
end while
return x̃^k
```

It is known that linear programming can be solved in polynomial time [FM05], so the cutting-plane method uses the LP relaxation to get a suitable starting point $\tilde{x}^0$ and then attempts to satisfy the integer constraints by adding cutting planes and re-running the LP solver on the new LP until all integer constraints are satisfied. Hence, every single LP that must be solved in the cutting-plane method begins with a solution where a variable that should be an

integer is actually a real number. Therefore, the expected utility computations at each stage will involve mixed strategies for the follower, instead of just pure strategies, as previously assumed.

It seemed that the ability of AGGs to capture context-sensitive independence would not be useful in the MILP solver in Section 4.3, but a closer analysis of a particular MILP solving method, the cutting-plane method, reveals that this ability could actually result in substantial speedups in DOBSS.

6 Solving the m-criminal problem

The ability of AGGs to capture anonymity structure cannot be exploited in the 1-leader, 1-follower Bayesian Stackelberg game with n follower types. The 1-leader, m-follower Bayesian Stackelberg game with n follower types for each follower would be just as hard to solve as a Nash equilbrium for a Bayesian game with m-followers with n follower types, because once the leader has committed to its strategy, the m followers are left to reason against each other in the resulting Bayesian game.

However, if it is assumed that the n followers are unable to affect each other through their actions, the resulting problem should still be tractable. This is because each follower the resulting m-follower Bayesian Stackelberg game only needs to solve a single-agent linear optimization problem, as the single follower did in Section 3.

The resulting MIQP, with m followers, where $F = N - \{l\}$ is the set of followers is

$$\max_{s_f, s_l, u_f^*} \sum_{t \in T} p^t \sum_{j \in A_l} \sum_{a_{-l} \in A_{-l}} u_l(a_l^j, a_{-l}) s_l(a_l^j) \prod_{f \neq l} s_f^t(a_f^k)$$

$$\begin{aligned}
\text{subject to } & \sum_{j \in A_l} s_l(a_l^j) = 1 \\
& \sum_{k \in A_f} s_f^t(a_f^k) = 1 && \forall t \in T, \forall f \in F \\
& 0 \leq u_f^* - EU_f^t(s_l, a_f^k) && \forall k \in A_f, \forall t \in T, \forall f \in F \\
& {u^t}_f^* - EU_f^t(s_l, a_f^k) \leq (1 - s_f^t(a_f^k))M && \forall k \in A_f, \forall t \in T, \forall f \in F \\
& s_l(a_l^j) \geq 0 && \forall j \in A_l \\
& s_f^t(a_f^k) \in \{0, 1\} && \forall k \in A_f, \forall t \in T, \forall f \in F \\
& {u^t}_f^* \in \mathbb{R} && \forall t \in T, \forall f \in F
\end{aligned}$$

and now the evaluation of $\sum_{t \in T} p^t \sum_{j \in A_l} \sum_{a_{-l} \in A_{-l}} u_l(a_l^j, a_{-l}) s_l(a_l^j) \prod_{f \neq l} s_f^t(a_f^k)$ will be much faster in the AGG representation for games with context-sensitive independence and anonymity. DOBSS should consequently experience significant speedup.

References

[CS06] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *EC '06: Proceedings of the 7th ACM conference on Electronic commerce*, pages 82–90, New York, NY, USA, 2006. ACM.

[FM05] Armin Fugenschuh and Alexander Martin. Computational integer programming and cutting planes. In K. Aardal, George Nemhauser, and R. Weismantel, editors, *Discrete Optimization*, volume 12, chapter 2. Elsevier, December 2005.

[JLB08] Albert Xin Jiang and Kevin Leyton-Brown. Action-graph games. Technical Report TR-2008-13, University of British Columbia, September 2008.

[PPM+08a] Praveen Paruchuri, Jonathan P. Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. Playing games for security: an efficient exact algorithm for solving bayesian stackelberg games. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 895–902, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[PPM+08b] Praveen Paruchuri, Jonathan P. Pearce, Janusz Marecki, Milind Tambe, Fernando Ordez, and Sarit Kraus. Efficient algorithms to solve bayesian stackelberg games for security applications. In Dieter Fox and Carla P. Gomes, editors, *AAAI*, pages 1559–1562. AAAI Press, 2008.

[SC73] M. Simaan and J. B. Cruz. On the stackelberg strategy in nonzero-sum games. *Journal of Optimization Theory and Applications*, 11(5):533–555, September 1973.

[SLB09] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, New York, 2009.