Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Optimal Machine Strategies to Commit to in Two-Person Repeated Games*

Song Zuo and **Pingzhong Tang**
Institute for Interdisciplinary Information Sciences,
Tsinghua University, Beijing, China
zuos13@mails.tsinghua.edu.cn,kenshin@tsinghua.edu.cn

Abstract

The problem of computing optimal strategy to commit to in various games has attracted intense research interests and has important real-world applications such as security (attacker-defender) games. In this paper, we consider the problem of computing optimal leader's machine to commit to in two-person repeated game, where the follower also plays a *machine strategy*.

Machine strategy is the generalized notion of *automaton strategy*, where the number of states in the automaton can be possibly infinite. We begin with the simple case where both players are confined to automata strategies, and then extend to general (possibly randomized) machine strategies.

We first give a concise linear program to compute the optimal leader's strategy and give two efficient implementations of the linear program: one via enumeration of a convex hull and the other via randomization. We then investigate the case where two machines have different levels of intelligence in the sense that one machine is able to record more history information than the other. We show that an intellectually superior leader, sometimes considered being exploited by the follower, can figure out the follower's machine by brute-force and exploit the follower in return.

Introduction

The problems of computing game-theoretical solution concepts are of central importance to algorithmic game theory and AI system design that adopt these solution concepts. Examples abound, such as computation of Mixed-strategy Nash equilibrium (Chen and Deng 2006; Daskalakis, Goldberg, and Papadimitriou 2006), correlated equilibrium (Jiang and Leyton-Brown 2011) and Nash equilibrium in large extensive-form game (Sandholm 2010).

Recently, much work has focused on computing *optimal strategy to commit to* (aka. *Stackelberg leader strategy*) in two-person games (Stengel and Zamir 2004; Conitzer and Sandholm 2006; Letchford and Conitzer 2010; Conitzer and Korzhyk 2011). In such games, one player is able to convince the other that she can commit to a strategy, before the actual game starts. Clearly, in a two-person normal form game, being able to commit to an optimal mixed strategy never hurts, sometimes increases the player's utility, compared with Nash equilibrium utility. Optimal commitment finds important application in the real-world security domain (cf. (Tambe 2011) for a comprehensive introduction). A common scenario for such applications is to allocate a small number of defenders to protect a relatively large number of targets, each of whom can protect one target, from being attacked. The problem is then to compute an optimal strategy to commit to in the induced security game.

For the commitment to be credible, one practical way is to carry out this commitment repeatedly over time. In other words, what the players actually engage is a repeated game. As a result, what the leader should optimize is the commitment in a *repeated game* rather than the one-shot game.

In this paper, we consider the problem of computing optimal commitment in two-person repeated game. One straightforward observation is that the set of strategies in an infinitely repeated game is infinite and thus previous approaches in (Conitzer and Sandholm 2006; Letchford and Conitzer 2010; Marecki, Tesauro, and Segal 2012) do not apply. Yet, there is a subset of strategies in such games called *automata strategies* that are particularly sensible. Automata strategy has first been studied by Rubinstein (1986) to analyze repeated Prisoner's Dilemma and then studied extensively (Abreu and Rubinstein 1988; Banks and Sundaram 1990; Ben-Porath 1993; Neyman and Okada 2000) under the scope of "bounded rationality"[1]. Simply put, a state in the automata represents an equivalent class of histories, over which the player chooses the same action. State transition function of the automata takes the action profile from the previous round as input and transits the player to the next state. In other words, automata is an abstracted, compact representation of player's strategy in repeated games. We generalize this representation to the one allowing for infinite number of states, which we coin *machine strategies*.

We make the following contributions. We begin with the

*This work was supported by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the Natural Science Foundation of China Grant 61033001, 61361136003, 61303077, a Tsinghua Initiative Scientific Research Grant and a China Youth 1000-talent program.

[1]For introduction of automation strategy, refer to (Parkes and Seuken 2013, Chapter 3)

simple machines and then extend the techniques to general, possibly randomized machines. We give a concise linear program to compute the optimal leader's machine. The linear program involves operations over the convex hull formed by the set of utility profiles. We give two efficient implementations for this part of the linear program: one via deterministic implementation and the other via randomization.

We further consider a more interesting case where two players use machine strategies with different complexities, in the sense that one can record more history information than the other. We show that a superior leader can exploit the follower by explicitly computing the follower's machine configuration.

Preliminaries

We now recall some game-theoretical preliminaries.

Repeated games and machine strategies

Definition 1. *A two-person normal-form game is a tuple $G = (A, B, u)$, where*

- *$A = \{a_1, \ldots, a_m\}$ is the action set for player 1.*
- *$B = \{b_1, \ldots, b_n\}$ is the action set for player 2.*
- *$u : A \times B \to \mathbb{R}^2$ is the utility function for pure strategies.*

Let $S_1 = \Delta(A), S_2 = \Delta(B)$ be the sets of all probabilistic distributions over A, B and hence the sets of all mixed strategies for player 1 and player 2 respectively. The definition of utility function naturally extends to mixed strategies, i.e., for $s_1 \in S_1, s_2 \in S_2$, $u(s_1, s_2) = \mathbb{E}_{\alpha\sim s_1, \beta\sim s_2} u(\alpha, \beta)$.

Definition 2. *In a repeated game G^T, the same stage game G is played by the same players for T periods. We consider the case where the stage game is a two-person normal-form game and is infinitely repeated, i.e., $T = \infty$.*

One observation is that the set of strategies in an infinitely repeated game is infinite and thus previous approaches in (Conitzer and Sandholm 2006; Letchford and Conitzer 2010; Marecki, Tesauro, and Segal 2012) do not apply. Yet, there is a subset of strategies called *automata strategies* that are widely used (Osborne and Rubinstein 1994; Shoham and Leyton-Brown 2009; Parkes and Seuken 2013). We extend this notion to formulate the memory complexity measure of *all feasible strategies*.

Definition 3. *A machine strategy (or simply machine), M, is a tuple $M = (Q, q^0, \lambda, \mu)$, where Q is the set of states, $q^0 \in Q$ is the initial state, $\lambda : Q \to A$ for player 1 (or $\lambda : Q \to B$ for player 2) specifies the action for each state, and finally $\mu : Q \times A \times B \to Q$ is the state transition function.*

In each stage, M chooses the action prescribed by the current state, receives the action profile played in the last stage and transits to the next state according to the state transition function.

Notice that Q is not restricted to be finite, so we actually define all feasible strategies as machines. For machine strategies with finite state sets, we call them *automata strategies*.

Definition 4. *A machine strategy $M = (Q, q^0, \lambda, \mu)$ is an automata strategy, if Q is a finite set.*

The flexibility of allowing for infinite state set facilitates our study of complex machines that use states to record past plays. Let $\mathcal{M}$ be the set of all machine strategies for G^∞. We can partition $\mathcal{M}$ into sub-classes of machine strategies. Within each sub-class memory space (size of the state set) grows at the same "rate".

Definition 5. *Let $\mathcal{M}_{O(f(t))}$ be the set of machine strategies that may reach at most $2^{O(f(t))}$ different states in the first t periods of G^∞. Formally as follows*

$$\mathcal{M}_{O(f(t))} = \left\{ M \in \mathcal{M} : |Q^t| \leq 2^{O(f(t))} \right\}$$

where Q^t is recursively defined as follows,

$$Q^0 = \{q^0\}$$

$$Q^t = Q^{t-1} \cup \{\mu(q', a, b) : q' \in Q^{t-1}, a \in A, b \in B\}, t \geq 1$$

The O function here and o function in the next paragraph refer to the standard big and small O notations. By definition, M is an automata strategy if and only if $M \in \mathcal{M}_{O(1)}$. Restricting attention to the set $\mathcal{M}_{O(t)}$ in fact does not lose any generality since the memory space of size $O\big((mn)^t\big) = 2^{O(t)}$ is sufficient to record perfect history information.

However, if both machines' memory spaces do not grow fast enough, both of them only use finite many states through out the game. Consider the machine generated by $M_1 \times M_2$ (in analogy to Cartesian product of automata). If $M_1 \in \mathcal{M}_{O(f_1(t))}, M_2 \in \mathcal{M}_{O(f_2(t))}$ and $f_1(t) + f_2(t) \leq o(\log t)$, then the number of states of machine $M_1 \times M_2$ is $o(t)$. It means that the number of states for recording the play sequence grows slower than t. Therefore, the play will end up in a loop. It also implies that, there exist $M_1', M_2' \in \mathcal{M}_{O(1)}$ that can perfectly represent M_1, M_2.

We use the standard notion of limit-of-means utility for infinite repeated game.

Definition 6. *The limit-of-means utility over the infinite action sequence $(a^t, b^t)_{t=1}^\infty$ resulted by M_1 and M_2 is*

$$u^1(M_1, M_2) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} u(a^t, b^t)$$

Our final notion is called *security level* (Shoham and Leyton-Brown 2009, Chapters 3).

Definition 7. *Security level (aka. minmax value) is the maximum utility a player can ensure in the game regardless of the strategies of the other player. In other words, if the utility one gets is less than his security level, he always has incentive to deviate to some other strategy.*

Let $\sigma_l(G, \mathcal{M}_1, \mathcal{M}_2)$ be the security level of player $l \in \{1, 2\}$ in game G with $\mathcal{M}_1$ and $\mathcal{M}_2$ being the machine strategy spaces respectively.

We include two examples of machine strategy in the full version of this paper.

Folk Theorem for Machine Strategies

Before our results, we present an alternative Folk theorem as a useful lemma when two players are confined to machine strategies.

Lemma 1 (Folk Theorem for Machine Strategies). *If the utility profile (v_1, v_2) is a convex combination of $P = \{u(a,b) : a \in A, b \in B\}$ and dominates the security levels for both players, then it can be*

- *realized as a Nash equilibrium utility profile played by two machine strategies with growing memory space (i.e., $\exists M_1, M_2 \in \mathcal{M}_{O(\log t)}$ such that (M_1, M_2) is a Nash equilibrium and the utility $u^1(M_1, M_2) = (v_1, v_2)$),*
- *or ϵ-approximated by a Nash equilibrium utility profile played by two automata strategies for any given $\epsilon > 0$ (i.e., $\forall \epsilon > 0, \exists M_1, M_2 \in \mathcal{M}_{O(1)}$ such that (M_1, M_2) is a Nash equilibrium and $\|u^1(M_1, M_2) - (v_1, v_2)\|_2 \leq \epsilon$).*

Especially, if the utility profile (v_1, v_2) is further a rational combination of P, then it can be realized by a Nash equilibrium utility profile played by two automata strategies.

Here (v_1, v_2) is a convex and rational combination of P if there exists $k_{ab} \in \mathbb{N}$ for $a \in A, b \in B$ such that

$$(v_1, v_2) = \frac{\sum_{a \in A, b \in B} k_{ab} u(a,b)}{\sum_{a \in A, b \in B} k_{ab}}$$

For readers not familiar with Folk theorem, an explicit constructive proof can be found in the full version.

Remark 1. *A two-person normal-form game may not have a pure NE, so the utility profile that dominates the security levels for both players (aka. enforceable utility profile) may not be located in the convex hull of P. In this case, G^∞ does not have any NE. In contrast, if randomization is allowed, the existence of NE for G^∞ is guaranteed, which we will discuss in the last section.*

However, the existence of optimal commitment is independent with the existence of pure NE of G, (we will show the independence in the next section), so our results do not rely on any assumption on G.

Commitment

We study the two-person Stackelberg game between players who use automata strategies in this section and extend the results to machine strategies in next section. We prove the existence by explicit construction and give a linear algorithm for its computation.

Definitions

Consider G^∞ as a two-person Stackelberg game where the *leader* (player 1) first commits to a strategy M_1 and then the *follower* (player 2) chooses a best response strategy M_2 with respect to M_1. Given player 1 choosing M_1, player 2's problem is to solve the following optimization problem:

$$M_2^* = \arg\max_{M_2} u_2^1(M_1, M_2).$$

The goal of player 1 is to maximize his own utility, i.e., to solve the following optimizing problem:

$$M_1^* = \arg\max_{M_1} u_1^1\big(M_1, \arg\max_{M_2} u_2^1(M_1, M_2)\big).$$

Optimal automata strategy to commit to

In this section, we prove the base of our main results. We prove by construction that the optimal automata strategy to commit to exists and can be found in linear time.

Theorem 1. *For the infinite repeated game G^∞ under limit-to-means utility, there exists an optimal automata strategy for player 1 to commit to, i.e.,*

$$M_1^* = \arg\max_{M_1 \in \mathcal{M}_{O(1)}} u_1^1\big(M_1, \arg\max_{M_2 \in \mathcal{M}} u_2^1(M_1, M_2)\big)$$

Furthermore, it can be computed in linear time $O(mn)$.

Following previous observations, the utility profile, denoted by (v_1, v_2), of this game must be located in the convex hull of the set of all utilities reachable from pure strategy profiles $P = \{u(a,b) : a \in A, b \in B\}$, denoted by $\mathrm{Conv}(P)$, in order to be attainable by G^∞. Moreover, since the strategy of player 2, M_2, is the best response to M_1^*, v_2 must be no less than the security level of player 2 which can be guaranteed by player 2 alone. Notice that for any utility profile satisfying these two conditions (and is rational combination of P)[2], by the constructive proof of lemma 1, there exist M_1 and M_2 such that M_2 is the best response to M_1 and they together achieve the utility profile (v_1, v_2). (Despite the Nash equilibrium of G may not exist.) Let (a', b') and (a'', b'') be the utility profiles that reach the security levels for player 1 and player 2 respectively, i.e.,

$$b' = \arg\min_{b \in B} \max_{a \in A} u_1(a,b),\ a' = \arg\max_{a \in A} u_1(a, b')$$

$$a'' = \arg\min_{a \in A} \max_{b \in B} u_2(a,b),\ b'' = \arg\max_{b \in B} u_2(a'', b)$$

Hence the remaining problem is how to maximize the utility of player 1. The problem of player 1 is as follows,

$$\begin{aligned} \text{maximize} \quad & v_1 = u_1^1(M_1, M_2) \\ \text{subject to} \quad & (v_1, v_2) \in \mathrm{Conv}(P) \\ & v_2 \geq u_2^1(a'', b'') = v_2'' \end{aligned} \tag{1}$$

which is in fact a linear program (LP) of v_1 and v_2. Notice that the solution to this linear program is always non-empty, since solution $(v_1'', v_2'') = \big(u_1^1(a'', b''), u_2^1(a'', b'')\big)$ satisfies all the boundary conditions.

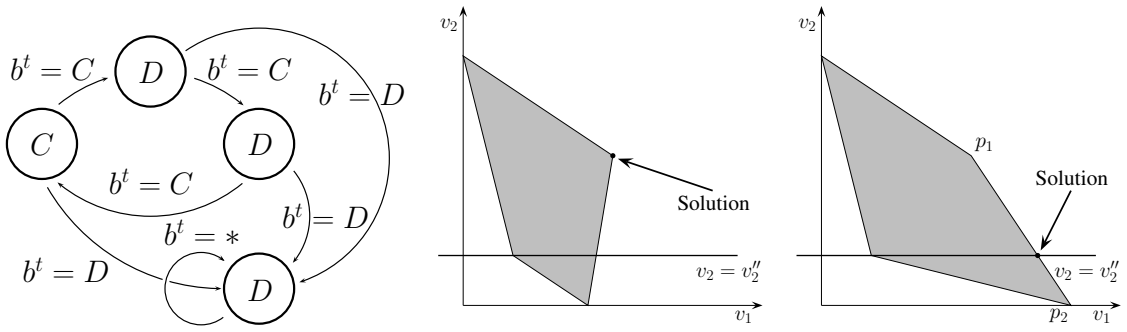
Proof. We give a constructive proof to theorem 1.

1. Solve linear program (1) and get the solution (v_1^*, v_2^*).
2. Construct M_1^*, M_2^* as we did in the proof of lemma 1 such that the utility profile of (M_1^*, M_2^*) is (v_1^*, v_2^*).
3. Output M_1^* as the final result.

The complexity will be discussed in the next subsection. □

A very useful observation is that the solution to the linear program (1) can be achieved at an extreme point of the polytope, i.e., either an element of P, or the intersection of line $v_2 = v_2''$ and the boundary of $\mathrm{Conv}(P)$.

[2] If it is irrational combination of P, this problem can be solved in an arbitrarily approximated sense.



(a) Example (b) Case 1 (c) Case 2

Figure 1: (a) Optimal commitment for player 1 in PD. (b)(c) Two types of solutions.

Example Consider an example where the stage game G is Prisoner's dilemma: $A = B = \{C, D\}$ and $u(C, C) = (3, 3), u(C, D) = (0, 5), u(D, C) = (5, 0), u(D, D) = (1, 1)$. Hence $v_2'' = 1$, and the linear program is

$$\begin{aligned} \max \quad & v_1 \\ \text{subj.t.} \quad & (v_1, v_2) \in \text{Conv}(\{(1,1),(0,5),(3,3),(5,0)\}) \\ & v_2 \geq v_2'' = 1 \end{aligned}$$

The optimal solution is $(v_1, v_2) = (13/3, 1)$, and the constructed result M_1^* is shown as figure 1 (a).

Implementing the LP

According to the proof of theorem 1, the bottleneck of solving the problem is to solve the specific linear program efficiently. Direct implementation of the linear program would induce a linear program of $O(mn)$ constraints. For more efficient implementations, we provide two algorithms. One is deterministic and the other is randomized.

A deterministic algorithm in $O\big(mn \log(mn)\big)$ The deterministic algorithm is simply to compute the convex hull of P in time $O\big(mn \log(mn)\big)$ (Graham 1972), find the intersection of the remaining inequality and the convex hull in time $O(mn)$, and finally get the optimal v_1 in time $O(mn)$.

A randomized algorithm in $O(mn)$ The randomized algorithm solves the linear program as follows.

Case 1: There exists element in P that is the solution. In this case, we are done (see figure 1(b)).

Case 2: Otherwise (see figure 1(c)) the solution must be the intersection point of line p_1p_2 and line $v_2 = v_2''$, where $p_1, p_2 \in P$ are on different sides of line $v_2 = v_2''$. Without loss of generality, suppose p_1 is above line $v_2 = v_2''$ and p_2 is below it. Then the first coordinate of p_2 is greater than p_1.

Let P_a be the set of elements in P above the separating line, and P_b be the set of elements in P below the separating line. Then the following *Algorithm 1* deals with this case.

Proof. **Correctness:** In each iteration, the algorithm either returns the solution when the "if" condition is satisfied, or removes at least one element, p_M, from $P_M \cup P_m$. Thus it always terminates. If we show that it never removes any element that might be the solution, it must return the solution.

Algorithm 1: Solving for case 2.

input : $\{P_M, P_m\} = \{P_a, P_b\}$
output: $\{p_1, p_2\}$
while *result is not found* **do**
  Let P_M be the larger set, P_m be the smaller set;
  Pick p_M from P_M uniformly at random;
  Find $p_m \in P_m$ s.t. P_m is to the left of line p_Mp_m;
  if P_M *is also to the left of line* p_Mp_m **then**
    return $\{p_1, p_2\} = \{p_M, p_m\}$;
  else
    Remove all the elements in P_M that are on or to the left of line p_Mp_m;

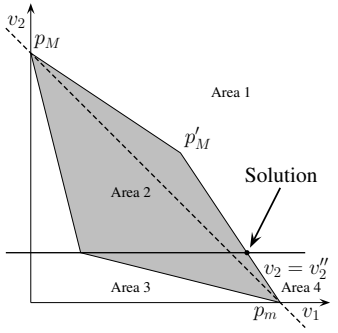


Figure 2: The plane is divided into 4 regions

Now we prove that the algorithm never removes "good" elements from P_a nor P_b. W.l.o.g., suppose $P_a = P_M$ has more elements than P_b. Lines p_Mp_m and $v_2 = v_2''$ divide the plane into four pieces (see figure 2). If the iteration ever reaches the removing step, there must be an element $p'_M \in P_a$ in area 1. Notice that p_m is selected such that there is no element in area 4. Therefore the final solution which keeps all the elements on or to the left of it must: i).go through area 1 and area 3; ii).keep p_m and p'_M on or to its left. Thus it never goes through area 2. In other words, removing elements in area 2 does not block the true solution.

Complexity: Let $h(n_1, n_2)$ be the expected running time

of *Algorithm 1* on input P_a, P_b, where $n_1 = |P_a|, n_2 = |P_b|$. Then we prove that $h(n_1, n_2) \leq c_1 n_1 + c_2 n_2 + c_0$ for sufficiently large constants c_0, c_1, c_2 by induction.

- When $n_1, n_2 \leq 3$, $h(n_1, n_2) = O(1)$.
- Suppose $\forall n_1, n_2 \leq N$, $h(n_1, n_2) \leq c_1 n_1 + c_2 n_2 + c_0$.
- Let $O(N) \leq c_1 N/2$ be the time for a single iteration.

$$\begin{aligned} h(N+1, N) &\leq O(N) + \frac{1}{N+1} \sum_{i=1}^{N+1} h(i, N) \\ &\leq O(N) + (N/2+1)c_1 + Nc_2 + c_0 \\ &\leq (N+1)c_1 + Nc_2 + c_0 \end{aligned}$$

- Notice that $h(n_1, n_2)$ is symmetric, then by the same argument $h(N+1, N+1) \leq (N+1)c_1 + (N+1)c_2 + c_0$.
- Therefore $h(n_1, n_2) \leq c_1 n_1 + c_2 n_2 + c_0 \leq O(n_1 + n_2)$.

So the expected time complexity is $O(|P|) = O(mn)$. □

Commitments for general machine strategies

As the succession of the previous section, we continue studying the Stackelberg game where two players are using general machines beyond automata. Especially, we focus on how fast memory space grows can make a difference in asymmetric intelligence games. We show that the space complexity significantly affects the dynamics between these two machines. In the next section, we apply these results to extend our main theorem to randomized case and most of the machine cases.

Optimal machine strategy to commit to

We first present our main theorem for machine strategies. Later we show by construction that the optimal machine strategy to commit to exists and can be found efficiently.

Theorem 2. *For the infinite repeated game G^∞ under limit-to-means utility, given that the machine strategy of player 2 is chosen from a subset $\mathcal{M}' \subset \mathcal{M}$, there exists an optimal machine strategy in $\mathcal{M}_{O(f(t))}$ for player 1 to commit to, i.e.,*

$$M_1^* = \arg \max_{M_1 \in \mathcal{M}_{O(f(t))}} u_1^1\big(M_1, \arg \max_{M_2 \in \mathcal{M}'} u_2^1(M_1, M_2)\big)$$

where $f(t)$ is an arbitrary function such that $\omega(1) \leq f(t) \leq O(t)$, and the subset $\mathcal{M}'$ is defined as follows[3]*,*

$$\mathcal{M}' = \Big(\mathcal{M}_{O(t)} \setminus \mathcal{M}_{O(\log t)}\Big) \cup \mathcal{M}_{o(\log f(t)/\log\log f(t))}$$

Notice that generally machines have infinite states, so usually cannot be described explicitly. However, the state transition function of our construction is *Turing computable* (see algorithm 2), so we can find the Turing machine that computes this function efficiently and use it as an alternative presentation of the actual machine.

[3] $\mathcal{M}'$ contains the powerful machines, which are too powerful to be exploited, and the weaker ones, which can be exploited by other machines with certain power.

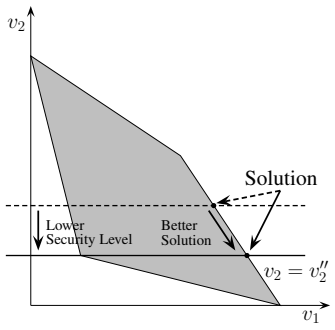


Figure 3: Lowering player 2's security level

Growing memory space enables machines to "record" increasingly more information as the game proceeds. Especially, compared with automata, machines with growing memory space (at least $O(\log t)$) can avoid looping forever.

However, the additional memory does not change the fact that the utility profile must lie in the convex hull, $\mathrm{Conv}(P)$. By lemma 1, if the utility profile lies in $\mathrm{Conv}(P)$ and dominates security levels of both players, it is achievable by Nash equilibrium. If it only dominates the security level of player 2, it is still achievable under Stackelberg game. Conversely if it is not in $\mathrm{Conv}(P)$ or it does not dominate the security level of player 2, it is never achievable under Stackelberg game, even if machine strategies were allowed.

In fact, it helps one to *lower the security level of the adversary* and *ensure a relatively high security level of the player*. Even though one can do nothing to the convex hull with advanced intelligence, being able to manipulate the security level is powerful enough to change the outcome of the game.

In other words, the extra memory does not help the players while they are still in cooperation, but ensures a higher utility once the cooperation is no longer sustained.

Manipulate the security levels

In this subsection, we discuss how to manipulate the security levels. An extreme way for one with its memory space growing sufficiently faster than the other one is to analyze its adversary's strategy and predict the future actions.

Lemma 2 describes how much extra memory is necessary for player 1 to suppress the security level of player 2.

Lemma 2. *For $\Omega(\log t) \leq f_1(t) \leq O(t)$, $\Omega(1) \leq f_2(t)$ such that $f_2(t) = o(\log f_1(t)/\log\log f_1(t))$*

$$\sigma_2(G^\infty, \mathcal{M}_{O(f_1(t))}, \mathcal{M}_{O(f_2(t))}) = \max_{b \in B} \min_{a \in A} u_2(a, b)$$

Recall that $\sigma_l(G, \mathcal{M}_1, \mathcal{M}_2)$ is the security level of player $l \in \{1, 2\}$ in game G with $\mathcal{M}_1$ and $\mathcal{M}_2$ being the machine strategy spaces for player 1 and player 2 respectively. If both use automata strategies, the security level of player 2 is

$$\sigma_2(G^\infty, \mathcal{M}_{O(1)}, \mathcal{M}_{O(1)}) = \min_{a \in A} \max_{b \in B} u_2(a, b)$$

which is greater than the quantity given in lemma 2.

In other words, with sufficient memory ability player 1 gains extra chance to increase his utility by reducing the security level of player 2. It is even more powerful in Stackelberg games, i.e., player 1 can further squeeze player 2's

utility to increase his utility (see figure 3). The utility profile in theorem 2 is calculated in the same way as theorem 1 except using security level given by lemma 2 when player 1 is able to exploit player 2. Finally, we explicitly construct player 1's machine that (when cooperation no longer sustained) achieves the security level specified in Lemma 2.[4]

Algorithm 2: Suppressing the security level of player 2.

input : Previous state $(t, \mathcal{H}, t_0, K, s, a^t, \tilde{b}^t)$ and previous action profile (a^t, b^t)
output: Next state $(t+1, \mathcal{H}, t_0, K, s, a^{t+1}, \tilde{b}^{t+1})$
$\mathcal{M}_K^* \leftarrow \mathcal{M}_K$;
foreach $\tilde{M}_2 \in \mathcal{M}_K^*$ **do**
  go though the record sequence $\mathcal{H}$;
  if $\tilde{M}_2$ *conflicts with* $\mathcal{H}$ **then** $\mathcal{M}_K^* \leftarrow \mathcal{M}_K^* \setminus \{\tilde{M}_2\}$;
if $\tilde{M}_2 = \emptyset$ *or* ($\tilde{b}^t \neq b^t$ *and* $s =$ paused) **then**
  $K \leftarrow \max\{k : \log mn \cdot (k+1) \cdot |\mathcal{M}_k| \leq f_1(t)\}$;
  $\mathcal{H} \leftarrow \emptyset$, $t_0 \leftarrow t$, $s \leftarrow$ recording;
  let a^{t+1} and $\tilde{b}^{t+1}$ be anything;
else
  let $\tilde{b}^{t+1}$ be the predicted action of M_2 by simulating the game between M_1 and $\tilde{M}_2$ from period t_0 to t;
  let a^{t+1} be the best response to $\tilde{b}^{t+1}$;
if $s =$ recording *and there is extra space* **then**
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{(a^t, b^t)\}$;
else
  $s \leftarrow$ paused;

Algorithm 2 describes the transition function of our constructed M_1 when the cooperation is not sustained. The state consists of current stage number t, record sequence $\mathcal{H}$, which begins at stage t_0, enumerating size K, recording state s, predicting action of player 2 $\tilde{b}^t$, and action of current state a^t. Initially, $K = 1$, $t_0 = t = 1$, $\mathcal{H} = \emptyset$, $s =$ recording. In each stage, M_1 enumerates all automata strategies in $\mathcal{M}_K$ (which contains those using at most K states) and predict M_2's action by anyone that never conflicts with record sequence $\mathcal{H}$. If everyone in $\mathcal{M}_K$ conflicts with $\mathcal{H}$ then M_1 increases the search space K and makes a new $\mathcal{H}$.

To sum up, as long as M_1 has enough memory, it almost correctly predicts M_2's action and plays the best response.

Randomness

As defined in definition 3, the action function and the transition function are both deterministic. A natural extension is to consider the machines with random inputs, so that they can change their states randomly and play mixed strategies.

To facilitate further discussion, we use superscript ^{r} to denote the randomness, i.e., M^r is a randomized machine, $\mathcal{M}^r$ is the collection of all randomized machines, and $\mathcal{M}^r_{O(f(t))}$ contains all machines that require at most $2^{O(f(t))}$ different states in period t.

[4]The correctness of the construction is shown in the full version.

	$\mathcal{M}_{O(g(t))}$	$\mathcal{M}^r_{O(g(t))}$	$\mathcal{M}^{(r)}_{O(\log t)}$
$\mathcal{M}_{O(1)}$	$\min_{a\in A}\max_{b\in B}$	$\min_{a\in A}\max_{b\in B}$	$\min_{a\in A}\max_{b\in B}$
$\mathcal{M}^r_{O(1)}$	$\min_{s_1\in S_1}\max_{b\in B}$	$\min_{s_1\in S_1}\max_{b\in B}$	$\min_{s_1\in S_1}\max_{b\in B}$
$\mathcal{M}_{O(f(t))}$	$\max_{b\in B}\min_{a\in A}$	$\max_{s_2\in S_2}\min_{a\in A}$	$\min_{a\in A}\max_{b\in B}$
$\mathcal{M}^r_{O(f(t))}$	$\max_{b\in B}\min_{a\in A}$	$\max_{s_2\in S_2}\min_{a\in A}$	$\min_{s_1\in S_1}\max_{b\in B}$

Table 1: Security levels of player 2

Since the randomized machine is able to use mixed strategies in stage games, the security level changes. Actually, equipped with randomness, the player who was well-understood by the adversary is able to guarantee better security level, since even if the other knows what the player is thinking, he still cannot predict the exact action. We present without proofs our results in table 1. Each entry specifies the security level of player 2 under different intelligences of both players. Formally, each entry quantifies $\sigma_2(G^\infty, \text{row}, \text{column})$ with $u_2(a,b)$ omitted. For example, the $\max_{b\in B}\min_{a\in A}$ in third row first column means $\sigma_2(G^\infty, \mathcal{M}_{O(f(t))}, \mathcal{M}_{O(g(t))}) = \max_{b\in B}\min_{a\in A} u_2(a,b)$, where $f(t)$ and $g(t)$ satisfies $\omega(1) \leq f(t) \leq O(t)$, $g(t) \leq o(\log f(t)/\log\log f(t))$. By minimax theorem,

$$\min_{a\in A}\max_{b\in B} u_2(a,b) \geq \min_{s_1\in S_1}\max_{b\in B} u_2(s_1,b)$$
$$= \max_{s_2\in S_2}\min_{a\in A} u_2(a,s_2) \geq \max_{b\in B}\min_{a\in A} u_2(a,b)$$

Allowing randomization has one more advantage, which is less related to this paper but of independent interest. As we mentioned previously, the intersection of $\text{Conv}(P)$ and utility profile that dominates the security levels might be empty for some games with deterministic players. However, if the players are equipped with randomness, the security levels move so that the intersection is guaranteed to be non-empty, i.e., the Nash equilibrium of the stage game.

Conclusion and Future Work

Computing Stackelberg leader strategy in various games has been shown to be both theoretically important and practically useful in settings such as security games. In this paper, we built a model of machine strategies with memory space increasing at different "rate". Then we studied the problem of computing the optimal machine strategies to commit to, and gave two efficient implementations to this problem. Finally we extended this result to machine strategies with asymmetric intelligence, i.e., randomness and growing memory space.

There are at least two exciting future directions. 1). To come up with a better algorithm for player 1 to "understand" player 2's strategy M_2, or prove that these machine strategies are unpredictable so that we are able to solve the problem for all machine strategies. ($M_2 \in \mathcal{M}_{O(g(t))}$ where $\Omega(\log t/\log\log t) \leq g(t) \leq o(\log t)$). 2). To extend asymmetric intelligence to the case where agents have asymmetric computational power.

References

Abreu, D., and Rubinstein, A. 1988. The structure of Nash equilibrium in repeated games with finite automata. *Econometrica: Journal of the Econometric Society* 1259–1281.

Banks, J. S., and Sundaram, R. K. 1990. Repeated games, finite automata, and complexity. *Games and Economic Behavior* 2(2):97–117.

Ben-Porath, E. 1993. Repeated games with finite automata. *Journal of Economic Theory* 59(1):17–32.

Chen, X., and Deng, X. 2006. Settling the complexity of two-player Nash equilibrium. In *In Proc. 47th FOCS*, 261–272.

Conitzer, V., and Korzhyk, D. 2011. Commitment to correlated strategies. In *AAAI*.

Conitzer, V., and Sandholm, T. 2006. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM EC (ACM-EC)*, 82–90. ACM Press.

Daskalakis, C.; Goldberg, P. W.; and Papadimitriou, C. H. 2006. The complexity of computing a Nash equilibrium. In *In Proc. STOC*, 71–78.

Graham, R. L. 1972. An efficient algorith for determining the convex hull of a finite planar set. *Information processing letters* 1(4):132–133.

Jiang, A. X., and Leyton-Brown, K. 2011. Polynomial-time computation of exact correlated equilibrium in compact games. In *ACM EC*, 119–126.

Letchford, J., and Conitzer, V. 2010. Computing optimal strategies to commit to in extensive-form games. In *ACM EC*, 83–92.

Marecki, J.; Tesauro, G.; and Segal, R. 2012. Playing repeated Stackelberg games with unknown opponents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 821–828. International Foundation for Autonomous Agents and Multiagent Systems.

Neyman, A., and Okada, D. 2000. Two-person repeated games with finite automata. *International Journal of Game Theory* 29(3):309–325.

Osborne, M. J., and Rubinstein, A. 1994. *A Course in Game Theory*. MIT Press.

Parkes, D., and Seuken, S. 2013. *Economics and Computation*. Cambridge University Press.

Rubinstein, A. 1986. Finite automata play the repeated prisoner's dilemma. *Journal of economic theory* 39(1):83–96.

Sandholm, T. 2010. The state of solving large incomplete-information games, and application to poker. *AI Magazine* 31(4):13–32.

Shoham, Y., and Leyton-Brown, K. 2009. *Multiagent Systems: Algorithmic, Game theoretic and Logical Fundations.* Cambridge Uni. Press.

Stengel, B. V., and Zamir, S. 2004. Leadership with commitment to mixed strategies. Technical report, London School of Economics.

Tambe, M. 2011. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press.