Hypergame Theory applied to Cyber Attack and Defense

James Thomas House[a], George Cybenko[b]

[a]Dartmouth College, Hinman Box 8000, Hanover NH 03755; james.t.house@dartmouth.edu
[b]8000 Cummings Hall, Dartmouth College, Hanover NH 03755; gvc@dartmouth.edu

ABSTRACT

This work concerns cyber attack and defense in the context of game theory – specifically hypergame theory. Hypergame theory extends classical game theory with the ability to deal with differences in players' expertise, differences in their understanding of game rules, misperceptions, and so forth. Each of these different sub-scenarios, or *subgames*, is associated with a probability – representing the likelihood that the given subgame is truly "in play" at a given moment. In order to form an optimal attack or defense policy, these probabilities must be learned if they're not known a-priori.

We present hidden Markov model and maximum entropy approaches for accurately learning these probabilities through multiple iterations of both normal and modified game play. We also give a widely-applicable approach for the analysis of cases where an opponent is aware that he is being studied, and intentionally plays to spoil the process of learning and thereby obfuscate his attributes.

These are considered in the context of a generic, abstract cyber attack example. We demonstrate that machine learning efficacy can be heavily dependent on the goals and styles of participant behavior. To this end detailed simulation results under various combinations of attacker and defender behaviors are presented and analyzed.

Keywords: hypergame theory, game theory, maximum entropy, hidden Markov models, machine learning

1. INTRODUCTION AND DEFINITIONS

Hypergame theory, introduced in the late 1970s by Bennet and Dando [1], is an extension of classical game theory intended to help deal with the differences that may exist between players – different levels of expertise, knowledge, deception, and perceptions of available options and actions.

Part of the process of solving a hypergame scenario involves the assignment or learning of probabilities for each subgame. Once this is accomplished – and given further assumptions about an opponent's likely perception – we can find the optimal playing policy.

Where applicable, we will follow conventions and nomenclature similar to those of Vane & Lehner. [2]

1.1 Basic Mechanisms of Hypergames

First let's establish some of the background necessary to give an example (figure 1).

The qualitative nature of this scenario is as follows:

- The row player (referred to as "Row"), attempting to attack/infiltrate a computer network, is able to employ three conventional attack methods (x, y, z). Row also has the option of using a relatively new exploit, which he believes fewer systems will be able to defend against. This new attack yields smaller net payoffs when successful (perhaps because in using it, some of its novelty and future utility is lost), but being harder to mitigate, also has milder losses for the attacker when it is not successful. We assume that all of these attacks are such that they cannot be defended against without human

Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense IX, edited by Edward M. Carapezza, Proc. of SPIE Vol. 7666, 766604 · © 2010 SPIE · CCC code: 0277-786X/10/$18 · doi: 10.1117/12.852338

$p_a = 0.65$			CMS 5x3 (a):	$c_{21} = 0.3062$	$c_{22} = 0.3287$	$c_{23} = 0.3651$	×	×	×	×
	$p_b = 0.25$		CMS 5x3 (b):	×	×	×	$c_{14} = 0.2377$	$c_{15} = 0.4074$	$c_{16} = 0.3549$	×
		$p_c = 0.10$	CMS 5x7 (c):	$c_{01} = 0$	$c_{02} = 0$	$c_{03} = 0$	$c_{04} = 0.1295$	$c_{05} = 0.3441$	$c_{06} = 0.2415$	$c_{07} = 0.2849$
NEMS 5x3 (a) ↓	NEMS 5x3 (b) ↓	NEMS 5x7 (c) ↓		Jr monitors x	Jr, y	Jr, z	Sr monitors x	Sr, y	Sr, z	Decoy
$r_{21} = 0.3848$	$r_{11} = 0.1234$	$r_{01} = 0.0561$	Attack, method x	**-5**	**6**	**7**	**-9**	**6**	**4**	**-3**
$r_{22} = 0.2388$	$r_{12} = 0$	$r_{02} = 0$	Attack, method y	**8**	**-5**	**6**	**6**	**-9**	**9**	**-4**
$r_{23} = 0.3764$	$r_{13} = 0.2068$	$r_{03} = 0.1393$	Attack, method z	**8**	**5**	**-3**	**4**	**8**	**-7**	**-2**
$r_{24} = 0$	$r_{14} = 0.6698$	$r_{04} = 0.4404$	Attack, new exploit	**4**	**0**	**3**	**3**	**-1**	**4**	**0**
$r_{25} = 0$	$r_{15} = 0$	$r_{05} = 0.3642$	Disconnect, no attack	**0**	**0**	**0**	**-1**	**0**	**0**	**4**

Figure 1. Network attack hypergame

monitoring and intervention – or more to the point, we assume that the defense (whether human- or software-driven) can only monitor one attack method at a particular instant. The payoffs and losses are assumed to be different between the various attacks – and furthermore, there are some similarities / correlations between some attacks (for example, examining figure 1, we see that watching for attack y helps to mitigate an attack by the new exploit as well).

- The column player ("Column") can be thought of as the sysadmin on duty; his company decides on each iteration (according to the subgame probabilities) whether to assign its junior sysadmin (who is less effective in an overall sense, though not in every detail) or its senior sysadmin to monitor the computer system. The company also has a decoy computer at his disposal: an attacker who is able to break into the decoy will find only worthless files, and furthermore will leave behind evidence of his attempt. The decoy's downside, for Column, is that legitimate outside users are unable to access the "real" computer whenever the decoy is online. Since we could consider one of Row's objectives to be harming the company – whether useful files are stolen or not – there is a positive payoff to Row if he disconnects without attacking when the decoy is in place. The harm or cost in this case is the lost productivity to the company owning the network, without the benefit of catching an attacker to offset the cost.

- The three probabilities in the upper left corner, p_a, p_b, and p_c, represent the true probabilities (not known to Row) that Column plays within the corresponding subgame.* For example, $p_a = 0.65$ indicates that there is a 65% chance that Column has assigned the junior sysadmin to monitor the network – perhaps operating under the belief that an attack is unlikely at this time and that the senior sysadmin's higher cost is not justified, perhaps due to a fixed labor schedule, etc.

- r_{mn} and c_{mn} represent probabilities associated with Nash strategies for the corresponding subgames (in general, for some subgames, it may be the case that one of these variables is exactly 1, indicating a pure strategy).

- The three perceived games (two strict subgames and the full game) are described as follows:

*In this paper, *subgame* can be taken to refer to the full game; the use of the term is not meant in a strict sense. In other words, subgame C is a subset (but not a strict / proper subset) of the full game, and is in fact identical with the full game.

- In subgame A, which is in effect with probability p_a, the junior sysadmin is on duty – we consider a sub-matrix of *all five rows and the first three columns.*
- In subgame B the senior sysadmin is on duty, and has a slightly better understanding of the new exploit. He is therefore better able to mitigate the use of this attack; he is also more experienced in general and usually (though not always) defends better than Junior against the conventional attacks. Senior is unable to deploy the decoy computer, however, as it would disrupt the work of legitimate users – this requires the approval of his boss. Here we consider a sub-matrix of *all five rows and columns four through six.*
- In subgame C the *full game* is considered. Senior's boss is able to put either Junior or Senior on duty – in effect, able to operate with either's expertise, at will – and also is able to replace the "real" computer (Row's desired target) with the decoy.

- Players move simultaneously in the full game and all subgames (like rock-paper-scissors rather than chess).
- Note that though this example is narrated with human attackers and defenders, the same sort of considerations could apply in automated scenarios, and would essentially reduce to questions of software sophistication for each side.

2. NETWORK ATTACK SOLUTION

The NEMS distributions (Nash Equilibrium Mixed Strategy) and expected payoffs (value v) for each subgame are shown below:

2.1 Numerical Results

- Subgame A
  - Row's value and NEMS: $\begin{bmatrix} 0.3848 & 0.2388 & 0.3764 & 0 & 0 \end{bmatrix}^T$, $v = 2.9972$
  - Column's value and NEMS: $\begin{bmatrix} 0.3062 & 0.3287 & 0.3651 & 0 & 0 & 0 & 0 \end{bmatrix}^T$, $v = -2.9972$
- Subgame B
  - Row's value and NEMS: $\begin{bmatrix} 0.1234 & 0 & 0.2068 & 0.6698 & 0 \end{bmatrix}^T$, $v = 1.7253$
  - Column's value and NEMS: $\begin{bmatrix} 0 & 0 & 0 & 0.2377 & 0.4074 & 0.3549 & 0 \end{bmatrix}^T$, $v = -1.7253$
- Subgame C
  - Row's value and NEMS: $\begin{bmatrix} 0.0561 & 0 & 0.1393 & 0.4404 & 0.3642 \end{bmatrix}^T$, $v = 1.0103$
  - Column's value and NEMS: $\begin{bmatrix} 0 & 0 & 0 & 0.1295 & 0.3441 & 0.2415 & 0.2849 \end{bmatrix}^T$, $v = -1.0103$

Incidentally, the fact that we are dealing with finite two-player games and rational payoff matrices implies that the exact solutions to each of these games must be in terms of rational numbers. The additional fact that we are dealing with a zero-sum game means that these Nash equilibria also conicide with minimax / maximin strategy sets. [3]

2.2 Remarks on Equilibrium Strategies

- Expected utility for Row decreases, as expected (pardon the choice of words), when we go from subgame A to B or from B to C. This is consistent with the intuitive notion that Column's results ought not get worse when he has a superior set of moves available, or when he has a strictly larger set of moves available (i.e., when a non-empty set of moves is merged with an existing set, even if all the new moves are inferior).

- The Jr sysadmin is never used in Nash play for the full game C; nor is Row's option of attacking by method y. The fact that monitoring for y also happens to give Column's best result against the new exploit, however, means that it still has a prominent place in Column's full game strategy.

- The "new exploit" and "disconnect" options do not make an appearance in Row's strategies until the senior sysadmin enters the game. In fact, when we go from playing either A or B to playing the full game C, "disconnect" becomes the second-likeliest move, though it was never played in either subgames A or B.

3. LEARNING UNKNOWN SUBGAME PROBABILITIES

Under some circumstances, for some hypergames, Row can estimate the subgame probabilities p_i (p_a, p_b, and p_c in our example).

No claims of optimality are made for the following procedures, but what follows will serve to illustrate that with repeated play, Row can begin with no knowledge (and no initial guess, in the maximum entropy case) for subgame probabilities, and end with estimates for these quantities. Further, we assume that Row's understanding of the subgames is accurate (or at least is accurate in the mean, in the noisy payoff case), aside from his lack of knowledge of the probabilities with which Column occupies them. Last, we assume that within each subgame Column will play a Nash equilibrium strategy.

3.1 Maximum Entropy – General Procedure

1. For each subgame i, Row computes the NEMS for Column (call this strategy c_i, a column vector of probabilities). Compute $E_i = U_i \cdot c_i$ for each subgame, where U_i is the payoff matrix for subgame i. Each row $(E_i)_r$ of the vector E_i represents the expected utility Row would gain on playing that row.

2. Let n represent the total number of subgames. For each available row r, compute

$$\sigma_r^2 = \frac{1}{n-1} \sum_i^n \left[(E_i)_r - \mu_r \right]^2 \tag{1}$$

   where μ_r is the (equiprobable) mean for row r across the subgames (that is, $\mu_r = \frac{1}{n} \sum_i^n (E_i)_r$). We can use this variance as a measure of the "dispersion" induced, by Column's switching from one subgame to another, on the expected payoff for a pure strategy consisting of Row playing r on each iteration with certainty. Call $\hat{r}$ the row for which this variance is greatest (for our generic network attack example, this is the second row). If the variance is equal among two or more rows, choose from among those maximum-variance candidates that row which maximizes the expected utility against Nash play by Column.

   Stated less formally, we are finding the strategy for Row which allows us to most easily distinguish one subgame's expected payoff from the remaining subgames.

3. Play $\hat{r}$ against Column multiple times and record the utility received from each play of the game. Compute $S_{\hat{r}}$, the mean of these utilities.

4. Assert the maximum entropy principle to find an estimate for each p_i. This can be done with a numerical solver and the Lagrange multiplier method, and amounts to finding the p_i which maximize

$$-\sum_i \left[p_i \cdot \log_2\left(p_i\right)\right] + \lambda_1 \cdot \left(S_{\hat{r}} - \sum_i \left[p_i \cdot \left(E_i\right)_{\hat{r}}\right]\right) + \lambda_2 \cdot \left(1 - \sum_i p_i\right) \tag{2}$$

The second and third terms are constraints, and λ_1 and λ_2 are their Lagrange multipliers. [4]

3.2 Maximum Entropy – Simulation Results

Iterated play of this scenario was simulated in Matlab, and results follow:

3.2.1 Noiseless Payoffs

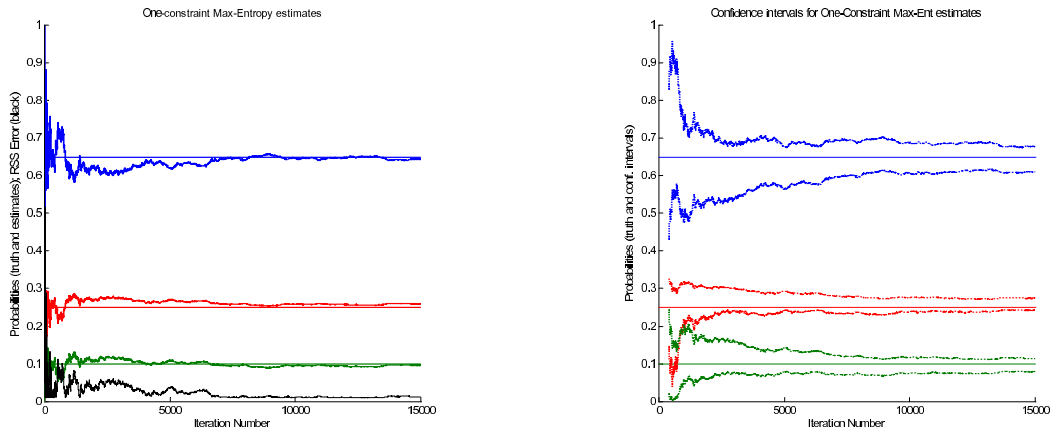

Figure 2. Max-entropy estimates, noiseless utility

The left part of figure 2 shows a typical result from 15,000 iterations of this game. The utilities are noiseless (that is, all entries in the payoff matrices are known with perfect predictability). The true probabilities (constant horizontal traces) are 0.65, 0.25, and 0.1. The total RSS error is the lowest trace in the plot. At the 15,000th step, the estimates are 0.644, 0.26, and 0.096, and have "settled" to within 0.05 RSS error by iteration 3134.

On the right are simultaneous multinomial confidence intervals ([5], [6]) at a 95% level. Though we do not always have unambiguous observations (unlike repeated die throws, for example), it is still reassuring to some extent to see the confidence intervals enclose the true probabilities here.

3.2.2 Noisy Payoffs

Next we run the same scenario, except that for every iteration of the game, the utility has a zero-mean uniform error added, no greater in magnitude than 100% of the nominal utility (figure 3). At the 15,000th step, the estimates are 0.635, 0.264, and 0.101, and have settled to within 0.05 RSS error by iteration 5205.

Although the maximum entropy method could perhaps be tweaked to better account for the utility noise [7], nonetheless we can see that it still works well without modification.

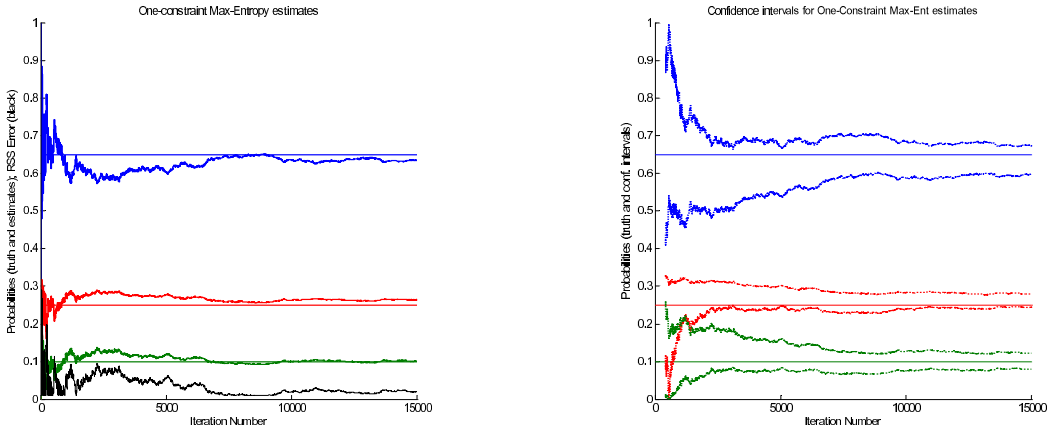

Figure 3. Max-entropy estimates, noisy utility

3.2.3 Pure Learning Strategy by Row vs. Nash Equilibrium Play

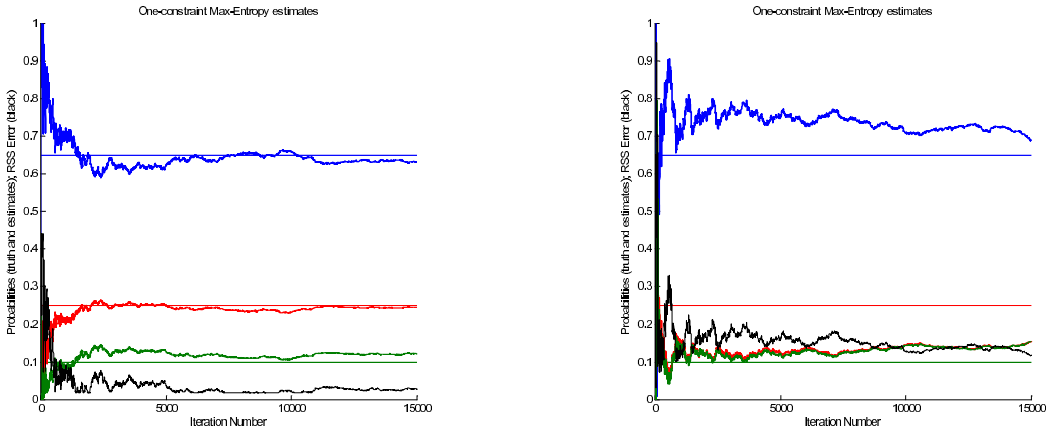

Figure 4. Row playing subgame A NEMS (l); subgame C NEMS (r)

Now we look at the difference in learning performance when Row plays his part of a Nash equilibrium (calculated on subgame A in the left figure; C in the right) rather than his pure "learning" strategy. Utility is noiseless here. At the 15,000th step, the estimates for the former case are 0.6338, 0.2451, and 0.1211, and have settled to within 0.05 RSS error by iteration 4942. For the latter case, we have 0.6904, 0.1550, and 0.1546, and the RSS error never falls below our arbitrary 0.05 threshold. In both cases, learning speed and accuracy are inferior to the results of the pure learning strategy of section 3.2.1.

3.3 Subgame Transitions as a Hidden Markov Model

The transitions from one subgame to another (or to the same subgame) can be modeled as a simple case of a hidden Markov model, as shown in figure 5. Here p_{ab} represents the probability of moving from subgame A to subgame B; q_{ac3} is the probability of emitting symbol c_3 on reaching subgame A, etc. To make comparisons with the maximum-entropy method straightforward, for the true probabilities we will have $p_{aa} = p_{ba} = p_{ca}$, $p_{ab} = p_{bb} = p_{cb}$, $p_{ac} = p_{bc} = p_{cc}$ (i.e. probabilities don't change when conditioned on current state, even though the HMM framework can certainly accomodate state conditionals). On building a history of emissions

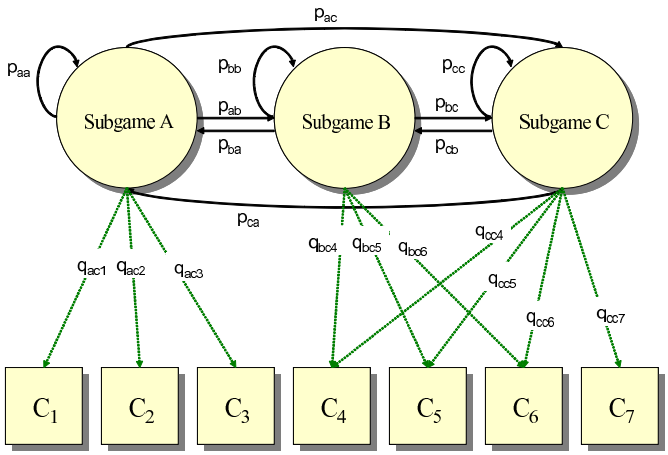

Figure 5. Subgame transitions and game results as an HMM

through repeated game play, we will use the Baum-Welch algorithm to estimate these transition probabilities, and hence the probability associated with each subgame. See Rabiner [8] for an excellent tutorial on HMMs.

3.3.1 Strategies for Learning the HMM Model

Here we would like to explore strategies for Row that are different from the maximum-variance "learning" strategy discussed above. This is because of the different way in which subgames are detected under the HMM framework.

Within a fixed row (which the Row player will know with certainty on each iteration, of course), we will treat the utility returned as the emission of the HMM. All else being equal, perhaps the sequence of observations which makes it easiest to distinguish between subgames is one in which each utility (observation) emitted corresponds with only one possible subgame. In general, however, there can be a great deal of subgame "overlap" in a hypergame – and our example here is not an exception. If we could make some rational assumptions to mitigate this overlap, our strategy would be more effective.

Happily, the assumption that Column plays the NEMS of the active subgame can help us on this point. For example, looking at the first row of our network attack scenario, the first three columns of subgame C are identical with the three columns of subgame A. However, when in subgame C, Column never plays these first three (he plays them with probability zero). Thus if we assume max-EU NEMS play by Column, the utilities from those first three columns can be associated strictly with subgame A.[†]

3.3.2 Minimax and Maximin Uniqueness – Learning vs. Obfuscation as a Nash Equilibrium

Though it is natural (and of primary importance) to analyze the performance of learning methods against optimal-expected-utility equilibrium play by Column, it also is worthwhile to ask: what if Column suspects that Row might have learning Column's behavior as his primary purpose? What sort of play might we see from Column in that case, and how might Row counter it?

One nicely intuitive approach would be to redefine / remap the payoff matrix such that those elements which can be expected to allow the quickest or most accurate learning will have the highest utility for Row. We could then calculate the Nash equilibrium strategies for this new payoff matrix, which would represent Row's attempt at best learning against Column's attempt at greatest obfuscation – these two would be mutual

[†]However, in section 3.3.2, we'll follow a different method for eliminating subgames – one more robust in the sense that it does not rely on the assumption of Nash play when associating each emission with a set of subgames. Eliminating subgames based on a particular strategy assumption remains a future direction we intend to explore.

best responses under the normal assumptions for any Nash equilibrium.[‡] Though Column cannot choose his subgame, he is certainly aware of which subgame he occupies, and can still choose to play within that subgame in such a way as to obfuscate the overall probabilities. So the immediate next step is to ask: how do we properly redefine the payoff matrix?

For both of these learning approaches, it is intuitive that Row should prefer as great a proportion of "unique" utility returns as possible. What we mean here, for example, is that if Row attacks by method x (row 1), and the noiseless utility he receives is -5 or 7 (but not 6), he can conclude with certainty that he was not playing against the Senior sysadmin (subgame B) on that iteration.[§] Note that he does not conclude he is playing subgame A, since subgame C might include a nonzero probability for these columns under a new learning/obfuscation NEMS (or some other strategy that is not the max-utility NEMS detailed previously). So we rely on the definitions of the subgames themselves to include or exclude Column's moves – we don't rely on zero entries in a mixed strategy distribution.[¶]

In contrast, if Row plays "Disconnect, no attack", then for any play by Column other than "Senior monitors x" and "Decoy", Row cannot draw any conclusion about which subgame (or column) is in play. We might therefore proceed in this manner: for each element in a given row of the full game payoff matrix, if that element's utility would rule out at least one subgame (that is, within at least one subgame it would be impossible for any strategy to result in that element's payoff), then assign some positive, nonzero utility to that element in some "learning payoff" matrix U_L. Note that with the overlap that can occur due to one subgame being identical with the full game (subgame C for our example), we do not require that one of these "unique" elements correspond with only one subgame – only that no such element correspond with *all* subgames. Figure 6 shows our new U_L matrix constructed in this manner – the value of each matrix element for this "metagame" is simply the number of subgames (0, 1, or 2) that can be ruled out – eliminated as possibilities.

To further expound on the justification for this remapping, consider that those elements which are nonzero do rule out, as impossible, at least one subgame – and since Baum-Welch has the Viterbi algorithm running repeatedly, calculating the most probable state sequence for various model assumptions, we would expect that those nonzero elements might introduce some level of certainty in the state sequence calculation at each point where those elements occur. It is difficult to imagine a scenario where such an introduction of certainty would *not* benefit Row's learning effort (but that is not to say, of course, that there does not exist some other mapping scheme which would be of greater benefit).

[‡]An important reminder – as part of the definition of a Nash equilibrium, an opponent's strategy is known, announced, or deducible. The "max dispersion" strategy detailed above is not proven to be suboptimal by the establishment of a Nash strategy here; the max dispersion strategy might simply be best applied to a different set of circumstances.

[§]Row will of course always know which strategy Row has played, so we are not looking for uniqueness over the entire matrix – only within each row.

[¶]One consequence of this is that for a game with N subgames, only those columns (if any) that exist only in that subgame which is equivalent with the full game can have an entry equal to $N-1$. All other entries must be strictly less. For our example here, we note that the rightmost column is present only in subgame C, and hence it is the only column with any entries equal to $3-1=2$.

$p_a = 0.65$			CMS 5x3 (a):	$c_{21} = \frac{1}{3}$	$c_{22} = \frac{1}{3}$	$c_{23} = \frac{1}{3}$	×	×	×	×
	$p_b = 0.25$		CMS 5x3 (b):	×	×	×	$c_{14} = \frac{1}{2}$	$c_{15} = \frac{1}{2}$	$c_{16} = 0$	×
		$p_c = 0.10$	CMS 5x7 (c):	$c_{01} = 0$	$c_{02} = 0$	$c_{03} = \frac{1}{2}$	$c_{04} = 0$	$c_{05} = \frac{1}{2}$	$c_{06} = 0$	$c_{07} = 0$
NEMS 5x3 (a) ↓	NEMS 5x3 (b) ↓	NEMS 5x7 (c) ↓		Jr monitors x	Jr, y	Jr, z	Sr monitors x	Sr, y	Sr, z	Decoy
$r_{21} = \frac{1}{3}$	$r_{11} = \frac{1}{2}$	$r_{01} = \frac{1}{2}$	Attack, method x	**1**	**0**	**1**	**1**	**0**	**1**	**2**
$r_{22} = \frac{1}{3}$	$r_{12} = \frac{1}{2}$	$r_{02} = \frac{1}{2}$	Attack, method y	**1**	**1**	**0**	**0**	**1**	**1**	**2**
$r_{23} = \frac{1}{3}$	$r_{13} = 0$	$r_{03} = 0$	Attack, method z	**0**	**1**	**1**	**1**	**0**	**1**	**2**
$r_{24} = 0$	$r_{14} = 0$	$r_{04} = 0$	Attack, new exploit	**0**	**1**	**0**	**0**	**1**	**0**	**1**
$r_{25} = 0$	$r_{15} = 0$	$r_{05} = 0$	Disconnect, no attack	**0**	**0**	**0**	**1**	**0**	**0**	**2**

Figure 6. Remapped metagame utility matrix U_L

The Nash equilibria for this re-mapped utility matrix come out as:

- Subgame A: Row plays $\left[\begin{array}{ccccc} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \end{array}\right]$; Column plays $\left[\begin{array}{ccccccc} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 \end{array}\right]$.[‖] (game value is $+\frac{2}{3}$ to Row)
- Subgame B: Row plays $\left[\begin{array}{ccccc} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \end{array}\right]$; Column plays $\left[\begin{array}{ccccccc} 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \end{array}\right]$. (game value is $+\frac{1}{2}$ to Row)
- Subgame C: Row plays $\left[\begin{array}{ccccc} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \end{array}\right]$; Column plays $\left[\begin{array}{ccccccc} 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \end{array}\right]$. (game value is $+\frac{1}{2}$ to Row)

One difficulty here is that since Row is uncertain which subgame is occupied at each iteration, he cannot be certain which of the above strategies to employ.[**] Helpfully, it is often the case (as above) that two or more of his strategies will be identical, though this need not be so in general.

We will call a Nash equilibrium strategy calculated on a metagame utility matrix such as figure 6 a *learning vs. obfuscation* equilibrium, or *learning* equilibrium for short.

We might interpret the value of a subgame here as a measure of potential quality / accuracy of learning when the equilibrium strategy set in question is played – then for each iteration where both Row and Column play the NEMS for subgame A, we could reason that Row has a greater expectation for learning than he would have when both Row and Column play the NEMS for subgame B or C (an expected $\frac{2}{3}$ subgames ruled out per iteration rather than $\frac{1}{2}$ ruled out). Also, there is naturally no reason why the expected payoff of this remapped game would not have meaning for any other sets of strategies. For example, calculating the value

[‖]This vector (and subgame B's vector) is expanded with zeros to be compatible with the size of the full game matrix; alternately we would write $\left[\begin{array}{ccc} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{array}\right]$ with the understanding that we mean it to apply to the 5x3 subgame A.

[**]In choosing which of his NEMS to employ on each iteration, one idea that is justifiable in a maximum-entropy sense would be to initially select the subgame A NEMS with probability $\frac{1}{3}$, the subgame B NEMS with probability $\frac{1}{3}$, and so forth. In general, when Row or Column has a choice between two or more equal-value Nash equilibria, one tie-breaker between those two might be to choose that strategy which gives the highest payoff on the normal (non-remapped) utility matrix.

under the remapped matrix, for both Row's and Column's max-EU equilibrium strategies, might give some indication of the learning potential of the normal (max-EU) equilibrium strategies. All that is needed is the equation $v = r^T \cdot U_L \cdot c$, with r and c being arbitrary mixed strategy distributions for Row and Column.

As a general comment, note that if the play is in a learning-vs-obfuscation equilibrium, it could be that such play offers an improvement (in learning) to Row compared to some other set of strategies; or it could instead be that such play offers an improvement (in obfuscation) to Column. Which way this goes is a characteristic of the game's payoff matrices and subgame definitions – an inherent attribute of each particular hypergame scenario.

3.4 HMM – Simulation Results

In figures 7 and 8 we show some example HMM results and figures of merit for several strategy pairings.

The entry *RSS (EP)* gives the RSS error using Baum-Welch when we begin with an equiprobable initial point (that is, we give the algorithm a starting guess of $\frac{1}{3}$ for each subgame's probability). *RSS (MaxEnt)* gives the error when we use the maximum entropy method's result as an initial guess for Baum-Welch. Given the sensitivity to the initial guess, it is not surprising that there is significant variation in the results. Most often, using the max-ent point as an initial guess gives noticeably better precision.

Note that the *learning value* entry in each cell is the value of the game calculated on U_L, but that this precise quantity cannot be known with certainty by Row. This is because each subgame probability determines to what extent the subgame contributes to this expected value, and these probabilities are exactly what we wish to estimate in the first place.

	Max-EU NEMS	Obfuscation NEMS
Pure learning strategy	RSS (EP) 0.1183 RSS (MaxEnt): 0.0796 Learning value: 0.7188	RSS (EP) 0.4076 RSS (MaxEnt): 0.3906 Learning value: 0.6083
Max-EU NEMS	RSS (EP) 0.2321 RSS (MaxEnt): 0.0600 Learning value: 0.6937	RSS (EP) 0.3724 RSS (MaxEnt): 0.4215 Learning value: 0.6083
Learning NEMS	RSS (EP) 0.1821 RSS (MaxEnt): 0.0571 Learning value: 0.6969	RSS (EP) 0.5218 RSS (MaxEnt): 0.2297 Learning value: 0.6083

Figure 7. HMM simulation results, 25k iterations

	Max-EU NEMS	Obfuscation NEMS
Pure learning strategy	RSS (EP) 0.0421 RSS (MaxEnt): 0.0316 Learning value: 0.7188	RSS (EP) 0.2356 RSS (MaxEnt): 0.2033 Learning value: 0.6083
Max-EU NEMS	RSS (EP) 0.1630 RSS (MaxEnt): 0.0940 Learning value: 0.6937	RSS (EP) 0.4614 RSS (MaxEnt): 0.4258 Learning value: 0.6083
Learning NEMS	RSS (EP) 0.1698 RSS (MaxEnt): 0.0604 Learning value: 0.6969	RSS (EP) 0.3874 RSS (MaxEnt): 0.2969 Learning value: 0.6083

Figure 8. HMM simulation results, 40k iterations

4. FUTURE WORK

- If we condition true subgame transition probabilities on the current, occupied subgame – we should expect that HMM methods will work well. Will the maximum entropy method still provide a worthwhile

initial guess?

- To what extent can we further remove Nash strategy assumptions about our opponent?
- Can we investigate time-varying utility? For example, a new exploit's utility should change as its novelty diminishes through repeated use.
- If we use the Nash play assumption to further rule out certain subgames, as explained previously, to what extent does our learning improve or degrade when that assumption is correct or incorrect?

5. CONCLUSION

Two independent methods for learning subgame probabilities during repeated play have been given. We have seen that Row's playing strategy significantly influences the accuracy of both methods, and that the two methods can be used in concert in such a way that they complement each other nicely. We have explored the notion of a learning-vs-obfuscation equilibrium, and have seen that at least for some hypergames, Column can hide his subgame probabilities to a great extent if he so chooses.

6. ACKNOWLEDGEMENTS

This research was partially supported by: Air Force Research Laboratory projects FA8750-10-1-0045 and FA8750-09-1-0174. Points of view in this document are those of the authors and do not necessarily represent the official position of the sponsoring agencies or the U.S. Government.

References

[1] P. G. Bennett and M. R. Dando, "Complex strategic analysis: A hypergame study of the fall of france," *Journal of the Operational Research Society* **30**, pp. 23–32, 1979.

[2] R. Vane and P. Lehner, "Using hypergames to increase planned payoff and reduce risk," *Autonomous Agents and Multi-Agent Systems* **5**, pp. 365–380, September 2002.

[3] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*, Cambridge University Press, 2002.

[4] A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes*, McGraw Hill, 4 ed., 2002.

[5] L. A. Goodman, "On simultaneous confidence intervals for multinomial proportions," *Technometrics* **7**, pp. 247–254, May 1965.

[6] B. J. R. Bailey, "Large sample simultaneous confidence intervals for the multinomial probabilities based on transformations of the cell frequencies," *Technometrics* **22**, pp. 583–589, November 1980.

[7] E. T. Jaynes, "On the rationale of maximum-entropy methods," *Proceedings of the IEEE* **70**, pp. 939–952, September 1982.

[8] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE* **77**, pp. 257–286, February 1989.