Comparing Agents' Success against People in Security Domains*

Raz Lin[1] **and Sarit Kraus**[1,2]
[1] Department of Computer Science
Bar-Ilan University
Ramat-Gan, Israel 52900
[2] Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742 USA
{linraz,sarit}@cs.biu.ac.il

Noa Agmon, Samuel Barrett and Peter Stone
Department of Computer Science
The University of Texas at Austin
Austin, Texas, USA
agmon,sbarrett,pstone@cs.utexas.edu

Abstract

The interaction of people with autonomous agents has become increasingly prevalent. Some of these settings include security domains, where people can be characterized as uncooperative, hostile, manipulative, and tending to take advantage of the situation for their own needs. This makes it challenging to design proficient agents to interact with people in such environments. Evaluating the success of the agents automatically before evaluating them with people or deploying them could alleviate this challenge and result in better designed agents. In this paper we show how Peer Designed Agents (PDAs) – computer agents developed by human subjects – can be used as a method for evaluating autonomous agents in security domains. Such evaluation can reduce the effort and costs involved in evaluating autonomous agents interacting with people to validate their efficacy. Our experiments included more than 70 human subjects and 40 PDAs developed by students. The study provides empirical support that PDAs can be used to compare the proficiency of autonomous agents when matched with people in security domains.

Introduction

The ability of expert designed agents (EDAs) to interact successfully with people is critical in any domain they are deployed. Increasing security needs have heightened the demand for autonomous agents operating and interacting with people in security domains. In these domains, it is necessary to react to intentional threats from adversarial agents, often under significant uncertainty (e.g., agents performing perimeter/border patrol). These environments are characterized as zero-sum environments, in which the adversary can learn or observe the agent's strategy and use it to its advantage. The success of an agent in such environments can have far-reaching consequences on capital or human lives (Pita et al. 2009). Simple real-life examples of such domains may include a perimeter patrol environment in which guards try and detect penetrations or a predator-prey scenario (Benda, Jagannathan, and Dodhiawala 1985), in which predators are required to surround a prey.

Although agents' interactions in security domains have gained focus, existing algorithms have some notable drawbacks. Many of the drawbacks arise from the fact that many of these algorithms assume optimal behavior on the part of the adversaries, which rarely holds in real-world situations, especially when people are involved (Pita et al. 2010). Many applications are already deployed in real-world security domains. For example, Amigoni *et al.* (2009) state that in patrol environments current strategies for detecting intruders are not always efficient in providing the patroller with high utilities, and propose a game theoretic framework to find, what they claim to be, optimal-efficient patrolling strategies. Another example of successful agents are the ones used to determine the scheduling of security at the Pittsburgh International (PIT) and Los Angeles International (LAX) Airports and the Federal Air Marshal Service (Pita et al. 2009).

One way to compare agents' strategies that are destined to perform against human adversaries is by facing them against other autonomous agents that mimic the decisions taken by human adversaries. However, designing agents that model human behavior is a very difficult task, especially due to diverse behavior of people making it difficult to capture behavior patterns in a monolithic model. People tend to make mistakes, and they are affected by cognitive, social and cultural factors, etc. (Lax and Sebenius 1992). Thus, it is commonly assumed that people cannot be substituted in the evaluation process of agents designed to interact with people. Furthermore, when people are used for experimentation purposes the evaluation process becomes time consuming and costly, making it a very difficult task for researchers.

The use of peer designed agents (PDAs) – computer agents developed by human subjects – has been proposed for experimental usages to investigate decision making in a multi-player computer game by Grosz *et al.* (2004). Lin *et al.* (2010) suggested using PDAs to remove people from the evaluation loop of automated negotiators and thereby simplify the evaluation process. While few applications exist in the automated negotiations domain, security domains have been gaining attention and agents are already deployed in many real-time applications. Thus, the ability to compare these agents using PDAs is of great importance.

Alas, security domains are zero-sum environments. In

*This research is based upon work supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under grant number W911NF-08-1-0144 and under NSF grant 0705587.

perfect information zero sum domains which consist of multiple rounds, such as chess and checkers (8×8 draughts), autonomous agents play against people differently than the way they play against other autonomous agents in the same environment (Bushinsky 2010).

This brings us to the following questions. Can PDAs help evaluate agents for security domains, or will this evaluation break down in zero-sum environments due to adversaries adapting to the PDAs? Is the evaluation of agents by PDAs limited to bilateral negotiations? Does the large search space and need of domain knowledge make security domains like zero-sum games, where evaluation by PDAs is not valid? Or, does the smaller action space in security domains make them similar enough to negotiation domains to enable the use of PDAs for evaluating agents?

In this paper we provide results of extensive experiments involving more than 70 human subjects and 46 PDAs. The main contribution of our work lies in the successful demonstration of the applicability of PDAs for comparing EDAs' success in security domains. We focus on two security domain environments. These environments are characterized by small game trees, small sized action sets and not requiring too much domain knowledge on the part of the agents. In the first experiment, people penetrated an area guarded by a team of several robots patrolling around its perimeter. In the second experiment, people had to control predators with the aim of catching an agent controlling the prey. In each experiment, we compared the performance of the EDAs both against PDAs and people. This was done in order to understand whether PDAs can be used to compare EDAs' success against people. Note that throughout the paper, we do not investigate the performance of the PDAs, but rather use them to investigate the behavior of EDAs.

Related Work

Important differences exist between designing an autonomous agent that can successfully interact with a human counterpart and designing an autonomous agent that interacts with other autonomous agents. When dealing with zero-sum environments or competitive games, results from social sciences suggest that people do not follow equilibrium strategies (Erev and Roth 1998; McKelvey and Palfrey 1992).

Agmon *et al.* (2008) investigated the problem of multiple robots patrolling a closed area. In this problem, a team of robots repeatedly visit a target area to protect. The robots' task is to detect the penetrators, adversaries played by people, and catch them. Agmon *et al.* showed that the optimal algorithm for the patrolling task fails miserably when matched with people in this setting.

In addition to this, the evaluation of agents that interact with people is a cumbersome task. For example, Agmon *et al.* (2008) needed nearly 70 students to evaluate their agents, and even then they did not obtain statistically significant results in all cases. Pita *et al.* (2010) used more than 200 students to compare different models of the autonomous agents.

The use of PDAs has been extensively studied within the context of the Trading Agent Competition (Wellman et al. 2001). In TAC, one needs to design a trading agent that participates in auctions for a certain good item. While the use of PDA's within this domain demonstrates the benefits of a large set of PDAs for evaluation purposes of EDAs, in this case, the PDAs are actually a form of EDAs, as the agents submitted to the competition are generated by experts and researchers.

The Use of PDAs: The Methodology

It is important to understand the subtle considerations of the methodology for using PDAs in our research. In this section, we elaborate on the basis of using PDAs and on their design process. All PDAs, detailed design instructions, code skeleton, EDAs, domains, and results described in the paper can be downloaded[1] to allow replication of the results and extension to different domains and scenarios. In addition, Listing 1 summarizes the methodology procedures.

Recall that the objective of our paper is to demonstrate how peer design agents can be used for the purposes of evaluating expert design agents, and not as a mechanism for replacing the EDAs or comparing the PDAs with EDAs. The paper hypothesizes that PDAs can serve as people's proxies in the design-test-revise cycles of computational decision-making strategies. In addition, we do not claim that PDAs behave in a way that is similar to the behavior of individual people. Instead, we use them as a mechanism to compare EDAs' performance.

Ideally, the PDAs should be designed by a population that is as similar as possible to the people who will later be matched against the EDAs. In this paper, we use essentially identical populations. The designers need to be given a precise task of implementing a proficient autonomous agent for a given security domain, with explanations about the security domain, but withholding information concerning the EDAs themselves. Also, as the focus is on the design of peer designed agents, and not expert design agents, no prior knowledge of decision- or game-theory is required.

To minimize errors and facilitate implementation, using the same simulation environment as the people who are matched against the EDAs is encouraged, as well as providing them with skeleton classes and APIs, having all the necessary functionality. This will also allow them to focus on the strategy and behavior of the agent, and eliminate the need to implement the communication protocol. This will also allow them to use the simulation environment to test the strategies of their agents.

After the PDAs are submitted, a pre-test phase of experiments should be exercised to validate the correctness and soundness of the PDAs. PDAs that fail this phase (e.g., have debugging or compilation errors, such as 'crashing' or 'hanging', due to the different settings or variations upon which the experiments are run and of which the designers are unaware) can be returned to the designers in order to fix the errors. After gathering all PDAs, including the resubmitted ones, they can be matched against the EDAs. PDAs that still exhibit errors need to be removed from the repository and be discarded for all experiments.

[1] Link removed to preserve anonymity.

Note that while we use people for the design of PDAs, once the PDAs capture their strategy, the PDAs can be used for the evaluation and comparison of different EDAs. This enables the pre-testing, revision, and improvements of the EDAs before operational testing and deployment. This is in contrast to the process of evaluation with people which is costly, requiring time, effort, and personnel. Furthermore, another agent may need to be evaluated, or an additional set of experiments and different people may be required when the strategy of the agent seems problematic. On the other hand, the PDAs can simply be reused, though careful experimental methodology must be used to avoid overfitting to the PDAs. We now continue with the description of the problem, the security domains and the EDAs with which the PDAs were matched.

Listing 1 Building blocks of using PDAs.

```
Provide designer with the precise task and implementation details
Generate skeleton classes and simulation environment
Do not provide details about EDAs
Obtain PDAs from designers
Run pre-test to validate PDAs
for all PDAs do
    if PDA fails then
        Allow revision and fixing
    end if
    if PDA still fails then
        Discard PDA
    end if
end for
Run PDAs against EDAs
Compare EDAs success
```

Problem Description

We consider the problem of comparing agents' success against people in security domains. Multiple agents are interacting with each other with the aim of achieving a certain goal despite the presence of adversaries. Two distinct security domains were used in the research, and both domains are described below.

The Perimeter Patrol Environment

The first environment was motivated by security issues of perimeter patrol. In this environment, a team of k mobile agents is required to repeatedly visit a given circular path (the perimeter of some polygon) in order to detect penetrations that are controlled by an adversary (see Figure 1). The robots may execute a wide variety of patrol algorithms. The adversary, based on the knowledge it has on the patrolling robots, has to choose a spot to penetrate that will maximize its chances of penetrating without being detected. We assume that the penetration time of the adversary is not instantaneous, and lasts t time units at which time the robots can detect the adversary if they are located in the same spot.

In our environment, the PDAs and humans play the role of the attacker facing simulated robotic defenders (the EDAs) which patrol along the perimeter. The humans or PDAs observe the patrolling robots for several time steps (depending on the setting) before deciding through which point to penetrate.

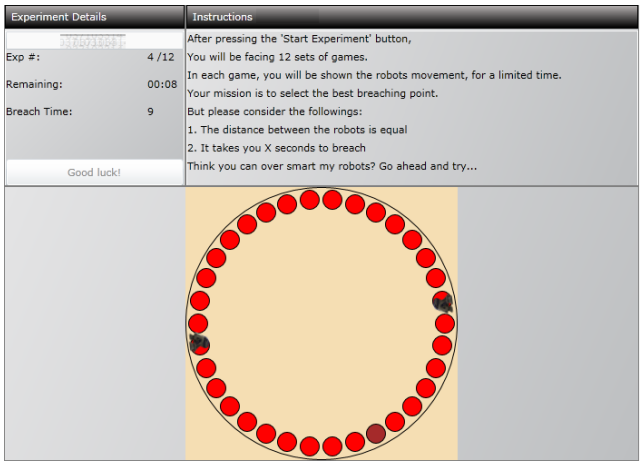

Figure 1: Multi-robot perimeter patrol game environment.

The patrol algorithms executed by the simulated robots are based on a nondeterministic optimal patrol framework described in (Agmon, Kraus, and Kaminka 2008). In this framework, all robots are placed uniformly along the perimeter and move in a coordinated manner, changing directions simultaneously (thus maintaining a uniform distribution throughout the execution). At each time step, the robots decide whether to continue straight (with probability p) or turn around (with probability $1 - p$). The choice of the p value is the essence of the patrol algorithm.

In this environment, we experimented with four EDAs which were adopted from (Agmon et al. 2009) and calculated their behavior according to the following algorithms:

- vMin whereby the robots maximize the probability of penetration detection in the weakest v segments (with several v values).
- vNeighbor whereby the robots maximize the probability of penetration detection in $v-$ neighborhood of the weakest segment (with several v values).
- MaxiMin, which maximizes the minimal probability of penetration detection along the perimeter, proven by (Agmon, Kraus, and Kaminka 2008) to be optimal against a full-knowledge adversary.
- MidAvg, a heuristic algorithm that averages between the p value of the optimal algorithms and the full and zero knowledge adversaries (MaxiMin and deterministic algorithms, respectively).

The adversarial behavior is assumed to be random wherever there is not enough knowledge to make knowledgeable decisions. Specifically, we assume that once the adversary has several locations through which she can penetrate and does not have enough knowledge to tell the difference between them, she will choose between these possibilities penetration spots at random. Algorithms vMin and vNeighbor take into consideration such cases, and optimize the robot's behavior under different assumptions on levels of uncertainty, i.e., number of possible spots. Therefore the actual

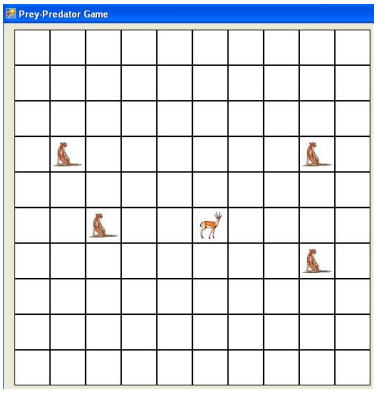

Figure 2: The predator-prey environment, represented by the gazelle and tigers, respectively.

patrol algorithm (the p value) adopted by the EDA varies, and depends on the assumed level of knowledge gained by the adversary on the patrolling robots.

The Predator-Prey Environment

The second security domain was motivated by the predator-prey scenario (Benda, Jagannathan, and Dodhiawala 1985). In this setting P predators are required to surround one prey. The game is played on a two-dimensional discrete grid of size $n \times m$, which is bounded and toroidal (e.g., one can move from the most right square to the most left square in the same row). The agents move simultaneously from their initial squares either up, down, left, or right, and agents cannot occupy the same position. The predators' goal is to surround the prey in all four directions (see Figure 2). All agents (both the predators and the prey) have perfect knowledge of the predators' and prey's positions.

In this environment, two EDAs were used. In the **ManhattanWalk** behavior, the prey calculates the minimum Manhattan Distance of the predators for every square on the board and moves to its adjacent square which has the maximum Manhattan Distance. The second EDA was **GuessManhattanWalk**, which also utilizes the Manhattan Distance as the previous one, but instead of employing it based on the current location of the predators, the algorithm tries to estimate the location of the predators in the next turn and then applies the Manhattan Distance. The estimation of the location of the predators is based on the prediction that each predator will choose to move to an adjacent square that is on the shortest path to reach the prey.

We proceed by describing the experiments conducted on these environments, their methodology and results.

Empirical Methodology

The experiments were conducted using the simulation environments described earlier. We ran an extensive set of simulations, consisting of more than 70 human players (40 and 35 people in the perimeter patrol and predator-prey environments, respectively) and 46 PDAs (36 and 10 PDAs in the perimeter patrol and predator-prey environments, respectively). The human players were mainly computer science undergraduate and graduate students, while a few were former students who are currently working in the high-tech industry. Each subject played in only one security domain. Prior to the experiments, the subjects were given oral instructions regarding the experiment and the environment. In the perimeter patrol environment, they were also handed additional explanation sheets describing the process of the game and explaining the parameters displayed to them along the game. A similar population was also used in the design of the PDAs.

The experiments comprised two sub-experiments. In the first sub-experiment, the human subjects played the role of an adversary while working against the EDAs in a web-based environment. In the second sub-experiment, we matched the PDAs with the EDAs. In the perimeter patrol environment, the people and PDAs played the role of the adversary that tries to penetrate through the simulated robots. A higher score was given to those who managed to penetrate successfully as an adversary more times than others. In the predator-prey environment, the people and PDAs played the role of the predators. A higher score was given to those who managed to surround the prey with the least number of moves.

Results and Discussion

The goal of the experiments was to analyze whether the strategy method of PDAs can be used to compare algorithms implemented for autonomous agents matched against people in security domains. Throughout this section, we also evaluate the significance of the results. The significance test was performed by applying the *Mann-Whitney U-test* on the results. The *Mann-Whitney U-test* is a non-parametric alternative to the paired t-test for the case of two related samples or repeated measurements on a single sample, suitable for data without normal distribution (like the data in our case).

Perimeter Patrol Environment Results

In the perimeter patrol environment, people and PDAs were matched against 4 types of EDAs: vMin, vNeighbor, MaxiMin and MidAvg, in different settings. The human players were instructed not to infer from one variation of the game to another. Twelve different settings were evaluated, which varied in reference to three characteristics:

- Distance between the robots (denoted by d, where $d = 8, 16$).
- Adversary's penetration time (denoted by t, where $t = 6, 9$, respectively). During this time, the adversary is stationary along the fence, and can be detected by a patrolling robot that comes across its location.
- Which EDA defenders the attackers had to confront (as described in the Problem Description Section).

In our experiment, when $d = 8$ and $t = 6$, we chose to examine two levels of adversarial uncertainty, as described earlier, for each type of vMin and vNeighbor EDAs, marked in our results by vMin1 and vMin2 for the EDA of type vMin, and vNeighbor1 and vNeighbor2 for the EDA of type vNeighbor (which represent 2 and 3 points of uncertainty, respectively).

As proven in (Agmon et al. 2009), in some cases (depending on the d and t values), the vMin and vNeighbor types of EDAs act the same, i.e., the chosen patrol algorithm for both of them coincide. Therefore, we decided to also experiment with the values $d = 16$ and $t = 9$, when this situation occurs. In this case we examined several implementations of the vMin EDAs, denoted in the results as vMin3, vMin4, vMin5 and vMin6, whereby these implementations differ in the level of assumed uncertainty in the adversarial choice.

As the attackers (people or PDA) choose the segment for penetration and the EDAs use probabilistic strategy to detect the penetration, we compute the *expected* probability of penetration detection. This computation is done based on a given EDA's policy and the subject's penetration choice. The result corresponds to the probability that the chosen segment will be visited by one of the EDAs during the t time units following the subject's choice. We then average this over all subjects' choices to obtain the expected probability of penetration detection. Figures 3 and 4 describe the *expected* probability of penetration detection given the players' choice of penetration locations for $d = 8, t = 6$ and $d = 16, t = 9$, respectively. The values of the x-axis represent the different EDA types (as described above). The values in the y-axis represent the expected probability of penetration detection, i.e., the expected ratio of adversarial penetrations detected by the EDAs. The dark, light and white bars represent the expected penetration detection when people, PDAs, or an optimal algorithm were involved, respectively.

Note that since the y-axis represents the expected probability of penetration detection by our defending agents, higher values mean better performance by our designed agents, and as result worse performance of the other side.

The results show that the best EDA for the (d, t) value pairs of $(8, 6)$ was the vNeighbor2 agent - both against people and PDAs with expected penetration rates of 0.63 and 0.52, respectively. The worst EDA – MaxiMin – could also have been predicted by matching it against PDAs as it performed worse both against people and PDAs (0.40 and 0.42, respectively). Similar results were obtained for the (d, t) value pairs of $(16, 9)$. The best EDA was the MidAvg with expected penetration rates of 0.37 and 0.40 for people and PDAs, respectively. The worst EDA was the vMin4 agent with expected penetration rates of 0.26 and 0.23 against people and PDAs, respectively. These trends were proven to be statistically significant ($p < 0.05$). The main results, though, are the consistencies between the performance of the PDAs and people against the EDAs, as reflected in Figures 3 and 4. For all twelve algorithms implemented in the EDAs, a similar trend was observed between the PDAs and people. The results demonstrate that performing better against the PDAs entails performing better against people, though the actual results may differ. To bolster our confidence in the strength and benefits of using PDAs we also matched the EDAs with an optimal algorithm. In contrast to the trend found between EDAs and PDAs, when investigating the results of the EDAs against the optimal algorithm, for all twelve algorithms we can see inconsistency of the results. That is, unlike using PDAs, one cannot use optimal agents to compare how proficient EDAs will be when matched against people.

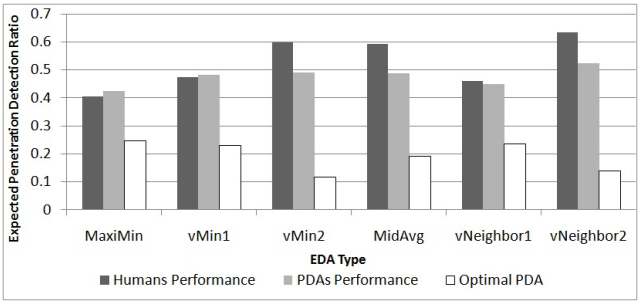

Figure 3: Results of the experiment for $d = 8, t = 6$. The bars represent the expected penetration detection ratio of the robots given the actual choices of the players.

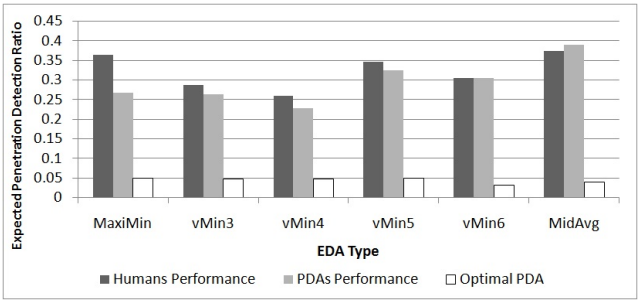

Figure 4: Results of the experiment for $d = 16, t = 9$. The bars represent the expected penetration detection ratio of the robots given the actual choices of the players.

Predator-Prey Environment Results

In the predator-prey scenario, 4 predators were required to surround one prey on a 10×10 board. Human players and PDAs were matched against different prey algorithms, as described in the Problem Description Section. Each PDA was run 100 times against each EDA in order to obtain unbiased results. Table 1 summarizes the results and compares the average time steps it took the PDAs and people to surround the prey in the experiments, when the prey followed either of the two implemented algorithm.

Comparing the PDAs behavior to that of people demonstrates that the **GuessManhattanWalk** EDA performs worse against both the PDAs and people, allowing them to surround it in less number of time steps (52.518 and 29.956 for the PDAs and people, respectively). Similarly, it took both the PDAs and people more time steps to surround the prey implementing the **ManhattanWalk** (109.605 and 43.879, respectively). The difference in the performance of the PDAs and the people against the EDAs is statistically significant ($p < 0.0045$ and $p < 0.046$, respectively). The results reveal the though **GuessManhattanWalk** is seemingly the more sophisticated (smarter) algorithm, it surprisingly does worse than the **ManhattanWalk** algorithm. Thus, the matching of the EDAs with PDAs was useful for predicting this counterintuitive result, and assist the designer in choosing which EDA to use.

	Prey Type	Predators' Type PDAs	People
(1)	**ManhattanWalk**	109.605	43.879
(2)	**GuessManhattanWalk**	52.518	29.956

Table 1: Results of the predator-prey experiment.

Unlike the perimeter patrol environment, in which it was shown that an optimal adversary exists (Agmon et al. 2008), it was infeasible to design an optimal adversary agent to match with the EDAs. This is due to the large branching factor of 5^4 and state space of 100^4 which made the computation intractable. Thus, in this domain, like many other possible domains, there is no option of comparing EDAs against an optimal agents. In those cases, the PDAs are not just a preferable option, but the *only* option.

Conclusions

This paper presented a novel investigation about the use of PDAs – computer agents developed by human subjects – in evaluation of agents in security domains. Evaluating expert designed agents is an obligatory process in validating the proficiency of the EDAs in their designated tasks. As EDAs are involved in more and more interactions with people, the evaluation process becomes increasingly difficult, as it requires time, effort, and personnel to evaluate the EDAs with people to assess their proficiency. We have demonstrated in two distinct security domains how peer designed agents can be used to compare the agents' proficiency.

The success in comparing an agent's proficiency based on interaction with PDAs is important. First, we showed that even though in other zero-sum environments the use of PDAs is not feasible, it is indeed applicable in the security domains environment. Second, once PDAs are obtained they are available for repeatedly testing, thus allowing researchers to apply the evaluation process at any given time. Moreover, their unbiased behavior allows them to be used repeatedly, with different designs of the tested agent, without affecting the results, as long as care is taken to avoid overfitting, e.g. by holding out some PDAs for one-time final testing.

In this research the population involved computer science undergraduate and graduate students (both in the design and the interaction). Our research demonstrated the success of comparing EDAs' using the PDAs methodology when the same population type is used. Obviously, in security domains, security experts (or hackers) are required to design the PDAs. Obtaining PDAs from this population can be achieved, for example, by capturing their strategy using programmers.

It is an interesting avenue for future work to explore to what extent the methodology extends to cases in which there is no access to the population of people (e.g. when they are enemies from a different culture and mindset). Future work also warrants careful investigation of the applicability of PDAs in domains that are more complex than bilateral negotiations and security domains, as well as larger domains (more than two agents). Such investigation will facilitate a better understanding of the differences in the characteristics of these domains, as well as the underlying assumptions behind the agents' design and interaction with people.

References

Agmon, N.; Sadov, V.; Kraus, S.; and Kaminka, G. A. 2008. The impact of adversarial knowledge on adversarial planning in perimeter patrol. In *AAMAS*.

Agmon, N.; Kraus, S.; Kaminka, G. A.; and Sadov, V. 2009. Adversarial uncertainty in multi-robot patrol. In *IJCAI*, 1811–1817.

Agmon, N.; Kraus, S.; and Kaminka, G. A. 2008. Multi-robot perimeter patrol in adversarial settings. In *ICRA*.

Amigoni, F.; Basilico, N.; and Gatti, N. 2009. Finding the optimal strategies for robotic patrolling with adversaries in topologically-represented environments. In *ICRA*, 819–824.

Benda, M.; Jagannathan, V.; and Dodhiawala, R. 1985. On optimal cooperation of knowledge sources. Technical Report BCS-G2010-28, Boeing Advanced Technology Center.

Bushinsky, S. 2010. Personal communication.

Erev, I., and Roth, A. 1998. Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibrium. *American Economic Review* 88(4):848–881.

Grosz, B.; Kraus, S.; Talman, S.; and Stossel, B. 2004. The influence of social dependencies on decision-making: Initial investigations with a new game. In *AAMAS*, 782–789.

Lax, D. A., and Sebenius, J. K. 1992. Thinking coalitionally: party arithmetic, process opportunism, and strategic sequencing. In Young, H. P., ed., *Negotiation Analysis*. The University of Michigan Press. 153–193.

Lin, R.; Kraus, S.; Oshrat, Y.; and Gal, Y. K. 2010. Facilitating the evaluation of automated negotiators using peer designed agents. In *AAAI*, 817–822.

McKelvey, R. D., and Palfrey, T. R. 1992. An experimental study of the centipede game. *Econometrica* 60(4):803–836.

Pita, J.; Jain, M.; Ordàñez, F.; Portway, C.; Tambe, M.; Western, C.; Paruchuri, P.; and Kraus, S. 2009. Using game theory for los angeles airport security. *AI Magazine* 30(1):43–57.

Pita, J.; Jain, M.; Tambe, M.; Ordàñez, F.; and Kraus, S. 2010. Robust solutions to stackelberg games: Addressing bounded rationality and limited observations in human cognition. *Artificial Intelligence* 174(15):1142–1171.

Wellman, M. P.; Wurman, P. R.; OMalley, K.; Bangera, R.; Lin, S.; Reeves, D.; and Walsh, W. E. 2001. Designing the market game for a trading agent competition. *IEEE Internet Computing* 5(2):43–51.