Joseph St. Cyr
Binghamton University – EECE580A: Cyber Physical System Security

Stackelberg Security Game Solved by ORIGAMI Algorithm

1. INTRODUCTION

The Optimizing Resources In GAmes using Maximal Indifference (ORIGAMI) project is an adjacent research topic which was spurred by a previous year's project as well as the paper found in reference [1]. The previous project was rooted in solving a non-cooperative Colonel Blotto Game with a single defender and two attackers. Dominating strategies were accounted for in the previous project to reduce the size of the game matrix input into the game solver. Although I believe it was implemented correctly, I don't believe it found enough dominated strategies given the game configuration parameters which provided only marginal performance increases. The run-times were in the order of several minutes per game. Taking the ORIGAMI algorithm concepts proposed in reference [1], I implemented a Stackelberg Security Game solver.

2. COLONEL BLOTTO GAME - DOMINATING STRATEGIES

When I first picked up the Colonel Blotto project, one of the immediate challenges I noticed was the readability of the code; There weren't very many useful comments in the code and many variables were single letters instead of having descriptive names. I spent a considerable amount going through each file in the Colonel Blotto project, renaming variables, adding file headers and comments, and making enhancements. This helped me understand fully what the code was doing and also developed my understanding of game theory in general. I also noticed that the x-axis of the figure that was being generated for the game was incorrect. The previous student(s) who worked on the Colonel Blotto project were plotting the defender's game resources (x-axis) against the game payoffs for each player (y-axis). This doesn't make much sense because each game payoff is a row vector of payoff amounts for each player (attacker1, attacker2, defender) at Nash equilibrium. Why would you want to plot each game payoff row vector with the defender's available resources for each game? You can see this in Figure 1 below.

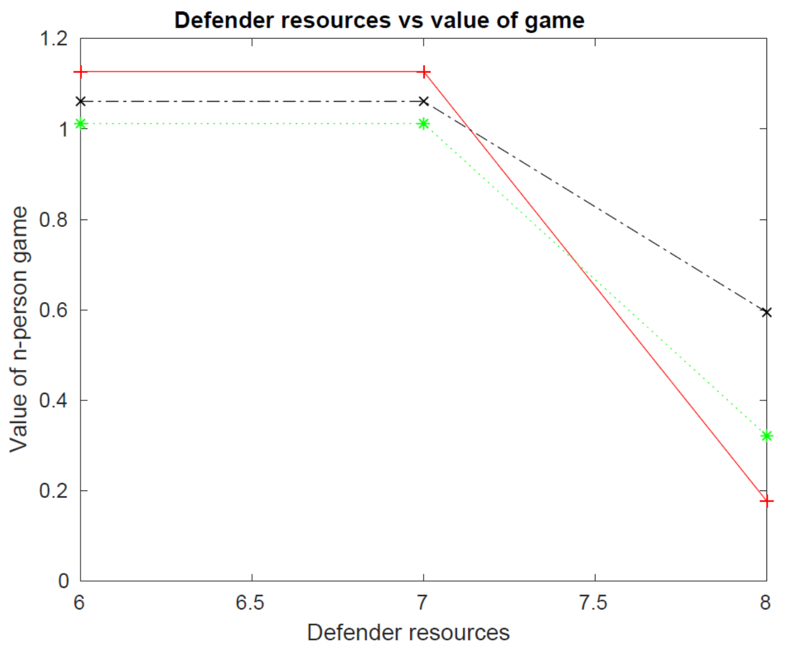

Figure 1. Original Colonel Blotto Game Figure - Junk

The original figure was modified to be more meaningful. The new figure can be seen in Figure 3 and Figure 3 below and depicts the games in a relevant manner.

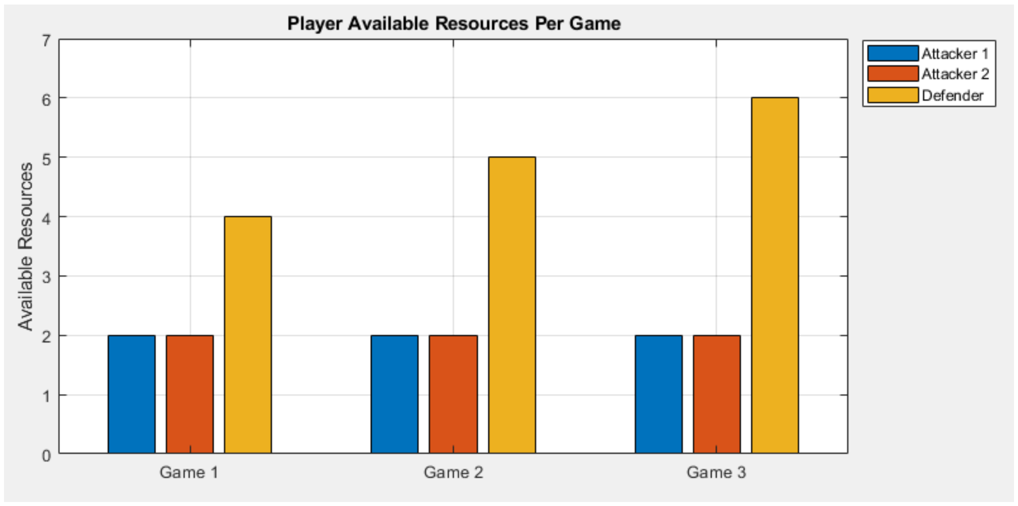

Figure 2. New Colonel Blotto Game Figure - Resources

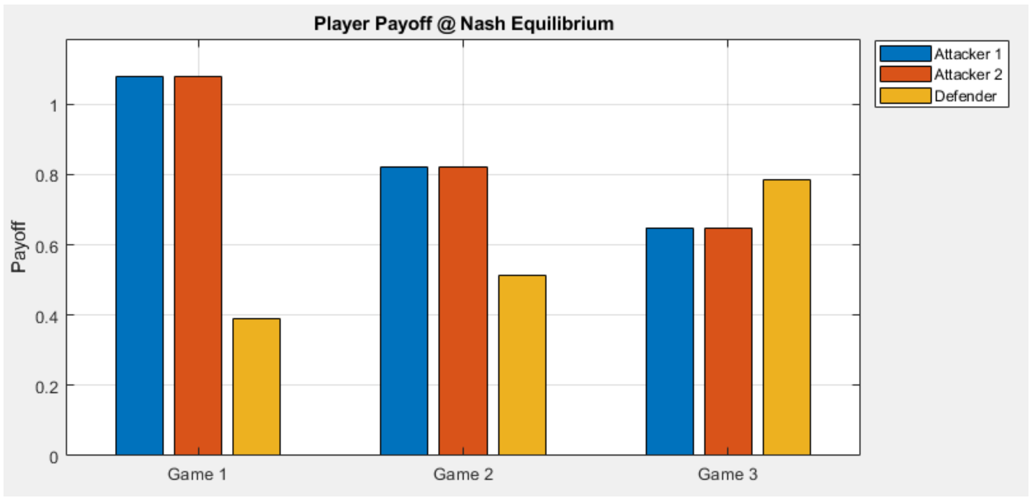

Figure 3. New Colonel Blotto Game Figure - Payoffs

Additionally, the elapsed time taken to solve each game was plotted in Figure 4 below.

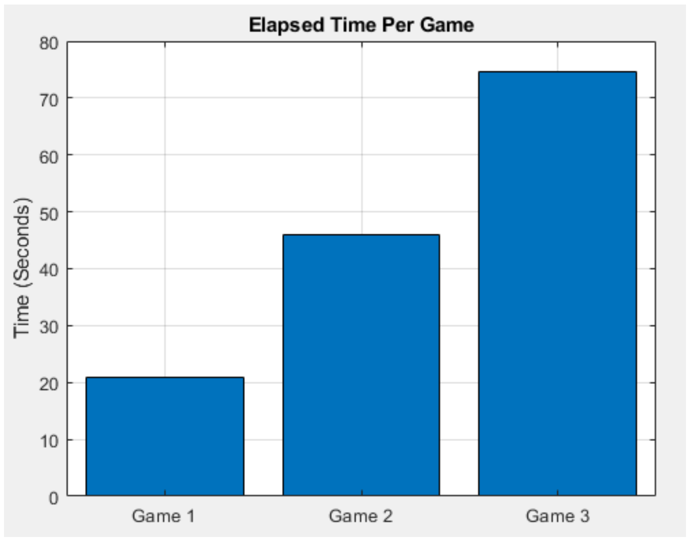

Figure 4. Colonel Blotto Game Elapsed Time Per Game

The work I completed this year on the Colonel Blotto project can be found in the public GitHub repository listed in section 4 - 1.a.

3. STACKELBERG SECURITY GAME - ORIGAMI

After reading the article in reference [1] and presenting on it in class I became interested in implementing the ORIGAMI algorithm proposed. The paper only provided ambiguous pseudo-code and contextual clues on how the algorithm worked. I spent several days (few hours each day) deciphering and developing an implementation. Ultimately, I was successful and the produced results are in alignment with what the paper intended. In summary, the ORIGAMI algorithm loops over a vector of available targets and allocates available defender resources to "cover" the targets based on attacker payoffs given a target is covered or uncovered. When the attack set is expanded, the coverage of each target is updated to maintain indifference between the attack set. In other words, the attacker is indifferent on which to attack because they all produce the same payoff. The solution maximizes the number of targets in the final attack set and maximizes the total coverage probability assigned to each target while maintaining indifference [1]. The results are based on execution of ten games. The game configurations are as follows in Table 1:

Table 1. Stackelberg Security Game: ORIGAMI – Configurable

```
% WHAT ARE THE AVAILABLE DEFENDER RESOURCES FOR EACH GAME?
%   [GAME_1; GAME_2; GAME_3; ... GAME_N]
defender_resources = [1;2;3;4;5;6;7;8;9;10];

% WHAT IS THE PAYOFF OF EACH TARGET FOR EACH PLAYER?
%   RESTRICT PAYOFF SUCH THAT...
%     defender_uncovered_payoff(TARGET) < defender_covered_payoff(TARGET)
%   AND...
%     attacker_uncovered_payoff(TARGET) > attacker_covered_payoff(TARGET)
%   FOR ALL TARGETS (1;2;3;...;n)
attacker_uncovered_payoff = [ 1; 2; 3; 4; 5; 6; 7; 8; 9; 10];
attacker_covered_payoff   = [-1;-2;-3;-4;-5;-6;-7;-8;-9;-10];

defender_uncovered_payoff = [-1;-2;-3;-4;-5;-6;-7;-8;-9;-10];
defender_covered_payoff   = [ 1; 2; 3; 4; 5; 6; 7; 8; 9; 10];
```

The results are as follows:

- Figure 5 shows the defender's resources per game. The defender gets one additional resource to allocate per game.

- Figure 6 shows the total number of available targets per game. Currently the project is only set up for a constant number of targets across all games.
- Figure 7 shows the defender's resource utilization at the strong Stackelberg Equilibrium (SSE). The data shows that seven resources are optimal for the defender to achieve the highest payoff with the most efficient utilization (smallest waste or resources).
- Figure 8 shows the player payoffs at the strong Stackelberg equilibrium (SSE). Notable data points are game five (five defender resources) where the defender and attacker payoffs are neutral. Another is game seven where the defender achieves the highest payoff with the fewest resources. In a strong Stackelberg equilibrium, the attacker selects the target in the attack set with maximum payoff for the defender [1].
- Figure 9 shows the elapsed time to calculate each game. Note that game eight and ten appear to be missing however that is not the case. The games compute very quickly and may be too short to quantify with the time precision available.

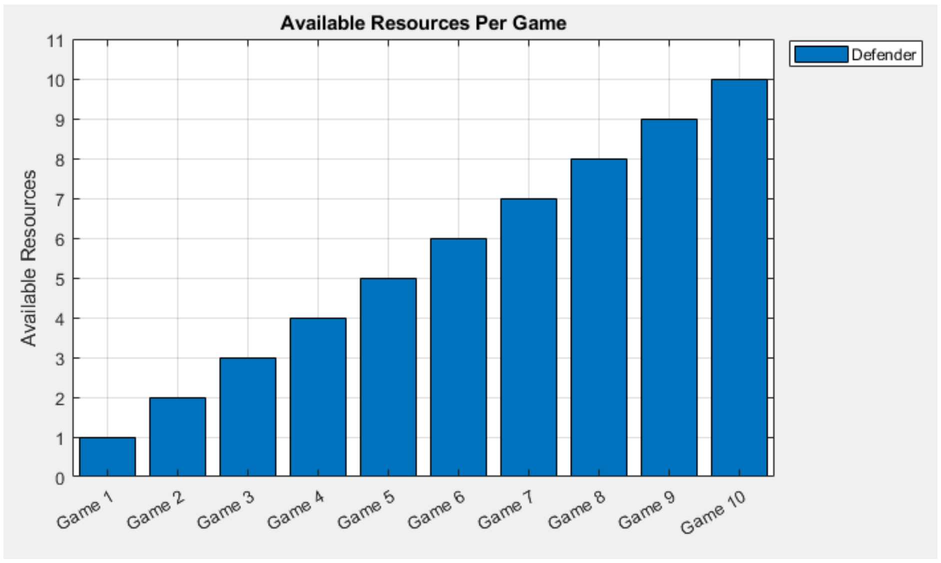

Figure 5. Stackelberg Security Game: ORIGAMI – Resources per Game

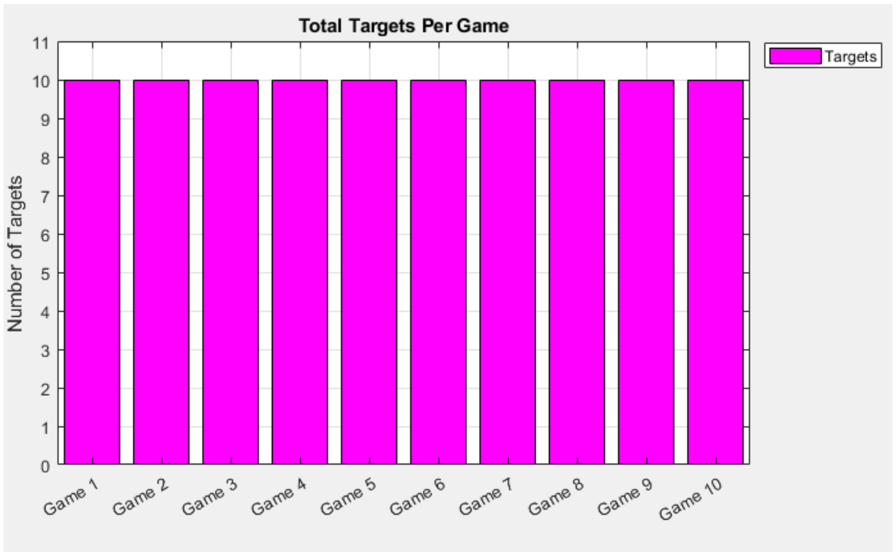

Figure 6. Stackelberg Security Game: ORIGAMI – Number of Targets per Game

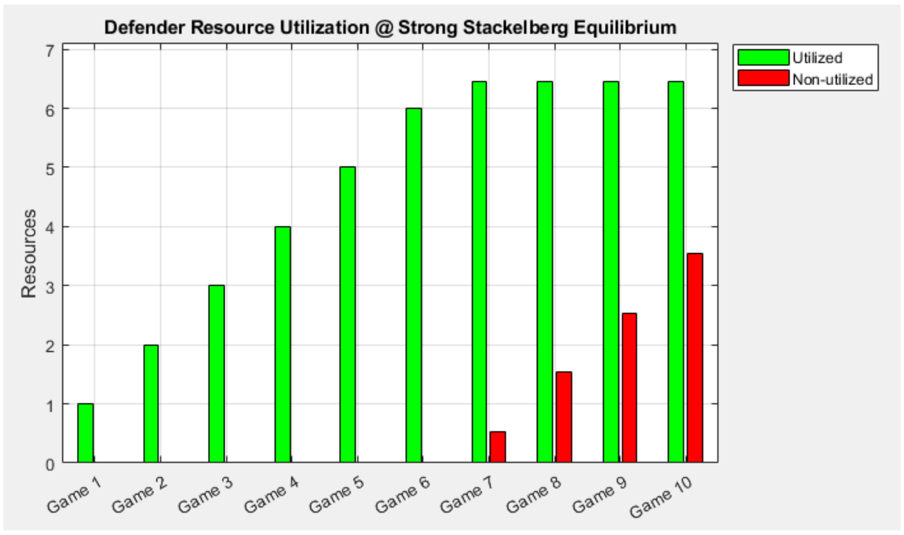

Figure 7. Stackelberg Security Game: ORIGAMI – Resource Utilization at SSE

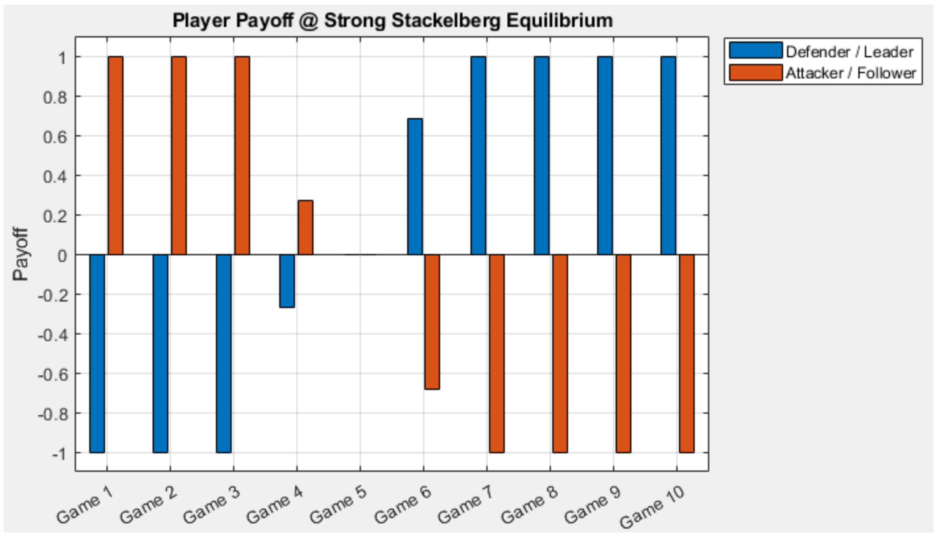

Figure 8. Stackelberg Security Game: ORIGAMI – Player Payoff at SSE

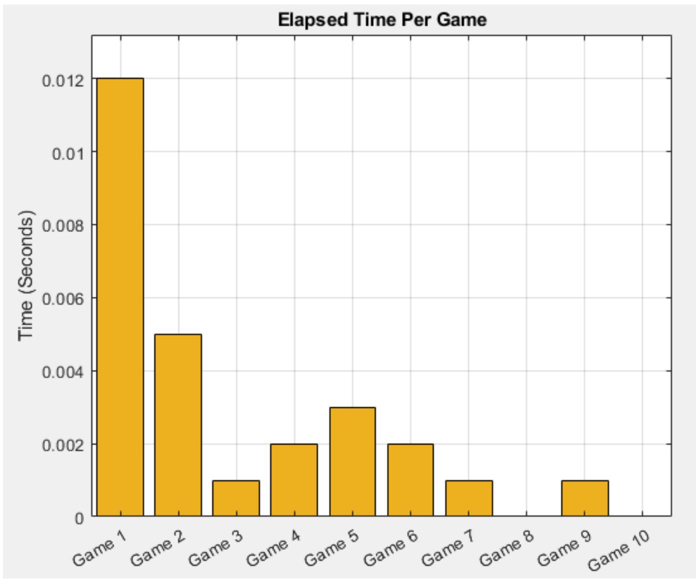

Figure 9. Stackelberg Security Game: ORIGAMI – Elapsed Time Per Game

The work I completed this year on the ORIGAMI project can be found in the public GitHub repository listed in section 4 - 2.a.

4. FUTURE WORK

There are several areas which future work can target. My recommendations on future work is as follows:

1. Make the number of targets per game dynamic. This will allow researchers to see how increasing targets (and keeping defender resources static) effects the player payoffs.
2. Plot additional figures of interest.
3. Test the algorithm with a large number of targets and resources to see if the performance matches what was reported in reference [1].
4. Apply the ORIGAMI algorithm to the Colonel Blotto game to reduce game calculation time. This will take some finesse due to the addition of cyber nodes, connections, multiple attackers, and multiple resources per target, however I believe it is achievable.

5. GITHUB

1. Colonel Blotto Game with Dominating Strategies
    a. https://github.com/SuPaSAINT/Dominating_Strategies_Game.git
2. Stackelberg Security Game Solved by ORIGAMI Algorithm
    a. https://github.com/SuPaSAINT/ORIGAMI.git

6. SYSTEM

Table 2. System Specifications

```
OS Name:                   Microsoft Windows 10 Home
OS Version:                10.0.18362 N/A Build 18362
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Workstation
OS Build Type:             Multiprocessor Free
System Manufacturer:       HP
System Model:              HP ENVY x360 Convertible 15m-dr1xxx
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 142 Stepping 12
                           GenuineIntel ~1803 Mhz
Total Physical Memory:     12,043 MB
Virtual Memory: Max Size:  14,475 MB
```

7. REFERENCES

[1] Kiekintveld, Christopher & Jain, Manish & Tsai, Jason & Pita, James & Ordóñez, Fernando & Tambe, Milind. (2009). Computing optimal randomized resource allocations for massive security games. AMAAS-2009 Conf. 2. 689-696. 10.1145/1558013.1558108.