Visualizing Adversaries - Transparent Pooling
Approaches for Decision Support in Cybersecurity

by

Daniel Prado Sánchez

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

© Massachusetts Institute of Technology 2018. All rights reserved.

Author

Department of Electrical Engineering and Computer Science
May 25, 2018

Certified by

Una-May O’Reilly
Principal Research Scientist
Thesis Supervisor

Certified by

Erik Hemberg
Research Scientist
Thesis Supervisor

Accepted by

Katrina LaCurts
Chairman, Master of Engineering Thesis Committee
Visualizing Adversaries - Transparent Pooling Approaches for Decision Support in Cybersecurity

by

Daniel Prado Sánchez

Submitted to the Department of Electrical Engineering and Computer Science on May 25, 2018, in partial fulfillment of the requirements for the degree of Master of Engineering in Computer Science and Engineering

Abstract

Using coevolutionary algorithms to find solutions to problems is a powerful search technique but once solutions are identified it can be difficult for a decision maker to select a solution to deploy. ESTABLO runs multiple competitive coevolutionary algorithm variants independently, in parallel, and then combines their test and solution results at the final generation into a compendium. From there, it re-evaluates each solution, according to three different measurements, on every test as well as on a set of unseen tests. For a decision maker, it finally identifies top solutions using various metrics and visualizes them in the context of other solutions.

However, it can be difficult to decide on which coevolutionary algorithms to run individually or use in ESTABLO. A coevolutionary variant, POOLING, was then created using this same principle of combining multiple variants. POOLING runs competitive coevolutionary algorithm variants, combines their solutions after every generation, and seeds the next generation with the top solutions found.

ESTABLO (with POOLING as one of its variants) is demonstrated on multiple cyber security related problems. We found that using ESTABLO was beneficial to most problems as different variants dominated in different scenarios. We also found that POOLING was able to consistently produce individuals that performed well against adversaries and in the context of all of their peers.

Thesis Supervisor: Una-May O’Reilly
Title: Principal Research Scientist

Thesis Supervisor: Erik Hemberg
Title: Research Scientist
Acknowledgments

I would like to thank Erik Hemberg and Una-May O’Reilly for their help and guidance throughout this project. They were incredible mentors and I will be forever grateful. I would also like to thank Marcos Pertierra for his work on ESTABLE. He was always available to help and lend his support. I would also like to thank Jonathan Kelly for his work on SDN Deception and Abdullah Al-Dujaili for his work on ADHD. Thanks to their work we were able to expand the scope of our project. Finally, I would like to thank my family for their unending support and advice to get me to where I am today.
Contents

1 Introduction .......................................................... 15
  1.1 Research Questions .............................................. 16
  1.2 Contributions .................................................... 17
  1.3 Overview .......................................................... 17

2 Related Works ....................................................... 19
  2.0.1 RIVALS ......................................................... 19
  2.0.2 Decision Making .............................................. 20
  2.0.3 Coevolutionary Search ........................................ 20

3 Methods ............................................................... 21
  3.1 ESTABLO .......................................................... 21
     3.1.1 Step 1. Coevolutionary Search ............................ 22
     3.1.2 Step 2. Compendium Population ........................... 23
     3.1.3 Step 3. Evaluating the compendium ..................... 23
     3.1.4 Step 4. Solution Selection ................................. 25
     3.1.5 Step 5. Solution Justification ............................ 25
  3.2 POOLING Coevolutionary Algorithm .......................... 27
     3.2.1 Step 1. Single Coevolutionary Step ...................... 27
     3.2.2 Step 2. Top Solutions ..................................... 27
     3.2.3 Step 3. Recombine and Repeat ........................... 28
  3.3 User Interface ..................................................... 28
4 Experiments

4.1 Mobile Asset Placement (MAP) ........................................ 31
  4.1.1 Grammar ......................................................... 32
  4.1.2 Fitness Function .................................................. 32

4.2 ADHD ........................................................................ 33
  4.2.1 Grammar ......................................................... 33
  4.2.2 Fitness Function .................................................. 34

4.3 Adaptive ADHD .......................................................... 34
  4.3.1 Grammar ......................................................... 34
  4.3.2 Fitness Function .................................................. 35

4.4 Setup ........................................................................ 35

5 Results ............................................................................ 37

5.1 MAP .......................................................................... 38

5.2 ADHD ......................................................................... 39

5.3 Adaptive ADHD .......................................................... 42

6 Conclusion ...................................................................... 47
List of Figures

2-1 Three previously tested topologies used in RIVALS.................. 19

3-1 Overview of the ESTABLO framework for decision support through se-
lection and visualization by using a compendium of solutions from
coevolutionary algorithms........................................... 21

3-2 Compendium population in ESTABLO............................... 26

3-3 Flow of POOLING.................................................. 28

3-4 The ESTABLO UI can display phenotypic distances and data from any
ranking scheme file generated by the ESTABLO analysis............ 29

5-1 Legend for figures in this section........................................ 37

5-2 This figure shows the combined score of attackers that were produced
for the MAP problem. Only attackers generated from 3 runs are in-
cluded to aid in readability (runs chosen as they produced some in-
dividual that was ranked number 1 in atleast 1 ranking scheme). We
see that the top solutions’ scores increased when placed into the com-
pendium of solutions. This indicates robust attackers.................. 39

5-3 This figure shows the inverse pareto front of attackers that were pro-
duced for the MAP problem. Only attackers generated from 3 runs
are included to aid in readability (runs chosen as they produced some
individual that was ranked number 1 in atleast 1 ranking scheme). We
see that although there are not many fronts, a significant amount of
individuals are still dominated by the top front. This indicates strong
attackers................................................................. 39
5-4 This figure shows the combined score of defenders that were produced for the MAP problem. Only defenders generated from 3 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that the top solutions’ scores remained relatively the same when placed into the compendium of solutions. This indicates that defenders are consistent in how they perform against unseen adversaries.

5-5 This figure shows the inverse pareto front of defenders that were produced for the MAP problem. Only defenders generated from 3 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that there are many fronts and the top defenders are able to dominate a large portion of their peer population. This indicates strong defenders.

5-6 The top attacker is overlayed on this topology in red and the top defender in green.

5-7 This figure shows the combined score of attackers that were produced for the ADHD problem. Only attackers generated from 3 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that the top solutions’ scores dropped drastically when placed into the compendium of solutions. This indicates that attackers are not robust in the face of unseen adversaries.

5-8 This figure shows the inverse pareto front of attackers that were produced for the ADHD problem. Only attackers generated from 3 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that there are many fronts and the top attackers are able to dominate a large portion of their peer population.
5-9 This figure shows the combined score of defenders that were produced for the ADHD problem. Only defenders generated from 3 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that the top solutions' scores remained consistent when placed into the compendium of solutions. This indicates that defenders are robust in the face of unseen adversaries.

5-10 This figure shows the inverse pareto front of defenders that were produced for the ADHD problem. Only defenders generated from 3 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that there are many fronts and the top defenders are able to dominate a large portion of their peer population.

5-11 This figure shows the combined score of attackers that were produced for the Adaptive ADHD problem. Only attackers generated from 4 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that the combined score of attackers increased when placed into the compendium. This indicates robust attackers.

5-12 This figure shows the inverse pareto front of attackers that were produced for the Adaptive ADHD problem. Only attackers generated from 4 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that there are many fronts and the top attackers are able to dominate a large portion of their peer population. This indicates strong attackers.
This figure shows the combined score of defenders that were produced for the Adaptive ADHD problem. Only defenders generated from 4 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme).

We see that the combined score of defenders increased when placed into the compendium. This indicates robust defenders.

This figure shows the inverse pareto front of defenders that were produced for the Adaptive ADHD problem. Only defenders generated from 4 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that there are many fronts and the top defenders are able to dominate a large portion of their peer population. This indicates strong defenders.
List of Tables

4.1 Common coevolutionary parameters for each experiment. . . . . . . . 35

4.2 Specialized parameters for varying mission size experiments and fitness function evaluation on the four topologies. . . . . . . . . . . . . . . . 35

5.1 Number and ratio of attackers found in the top 25% of attackers ranked by combined score per coevolutionary variant for MAP. Here we can see that POOLING and Coev found the largest amounts and highest ratios of top attackers. This is over all runs (30 total runs). . . . . . . . 40

5.2 Number and ratio of defenders found in the top 25% of defenders ranked by combined score per coevolutionary variant for MAP. Here we can see that IPCA, POOLING, and rIPCA found the largest amounts and highest ratios of top defenders. This is over all runs (30 total runs). 41

5.3 Number and ratio of attackers found in the top 25% of attackers ranked by combined score per coevolutionary variant for ADHD. Here we can see that POOLING and Coev found the largest amounts and highest ratios of top attackers. This is over all runs (10 total runs). . . . . . . 41

5.4 Number and ratio of defenders found in the top 25% of defenders ranked by combined score per coevolutionary variant for ADHD. Here we can see that POOLING found the largest amount and highest ratio of top defenders. This is over all runs (10 total runs). . . . . . . . 41
### 5.5 Number and ratio of attackers found in the top 25% of attackers ranked by combined score per coevolutionary variant for Adaptive ADHD.

Here we can see that POOLING and rIPCA found the largest amounts and highest ratios of top attackers. This is over all runs (5 total runs).

### 5.6 Number and ratio of defenders found in the top 25% of defenders ranked by combined score per coevolutionary variant for Adaptive ADHD. Here we can see that no single coevolutionary variant is coming out on top. This is over all runs (5 total runs).
Chapter 1

Introduction

As more critical technologies and information are moved online, cyber attacks are becoming a common issue. As they increase in severity and frequency, it’s important that systems are secured against these forms of attack. If an attacker is thwarted by some defense, they may learn from this defense and find a different way to attack a system. The defender then needs to go back and come up with a way to defend against this new attack. Today, there are many systems that are unable to adapt to the onslaught of innovative and new attacks. Being able to reroute resources may be simple online but having to move physical resources, be it personnel or machines, may not be as dynamic in a network that is spread out across the country. In these sorts of situations, it is imperative to be able to select a solution that is robust even in the face of active attackers.

Coevolutionary algorithms allow us to simulate an arms race between two competing populations of solutions and tests and identify adversarially robust solutions [8, 12]. Using them involves two stages: Stage 1: Co-optimization of populations of competitors using multiple runs of different algorithm variants. Stage 2 Identification of a “best” solution from among the solutions of all the runs and algorithm variants.

When exploring a space like cybersecurity using coevolutionary algorithms, a large number of solutions will be found as multiple coevolutionary variants will be used and multiple runs will be executed. Given this expansive set of solutions it can become difficult to pick one solution to deploy. This is exacerbated by the following issues:
• solution fitness is only an estimate that is based on the specific testing context of the population against which the solution competed

• coevolutionary variants have different solution concepts and direct comparison can be problematic.

• they cannot be fused into one solution in most cases because the competitive context does not allow it, e.g. a network can not be replicated to support multiple network configuration solutions

1.1 Research Questions

This second stage of identification can be difficult as a large number of solutions might need to be considered. Another challenge with this stage of identifying a “best” solution is how to compare solutions that evolved separately. Each coevolutionary variant might prioritize different measures and as such might not be directly comparable.

These challenges lead to the following competitive coevolution research questions:

• How can a compendium of solutions and tests be brought to a “level playing field”, i.e. be made comparatively symmetrical, where the solutions have reliable fitness estimates and they can be fairly compared despite solutions being populated with different solution concepts and fitness estimates that are based solely on the context of a run?

• How can testing blind spots be found?

• How can a best solution be automatically selected and transparently justified, with auditability for deployment?

• How can these same principles of combining multiple variants be used to improve one coevolutionary algorithm?
1.2 Contributions

The contributions of this thesis are as follows:

- Implemented ESTABLO, a framework that can run multiple coevolutionary algorithms and will select a top solution among the compendium of solutions produced [10].

- Designed and implemented POOLING, a coevolutionary algorithm that runs multiple coevolutionary algorithms and subsequently evolves the top solutions found each generation.

- Designed and implemented a user interface to aid in decision support.

- Created and ran experiments on three distinct cyber security problems.

1.3 Overview

In Chapter 2 we discuss previous work that relates to the questions we attempt to answer in this thesis along with work that we build off of. Chapter 3 describes ESTABLO, POOLING, and a user interface that was used for aiding in decision support. In Chapter 4 we discuss the problems that we ran ESTABLO and POOLING on along with the parameters we used for each experiment. We then discuss the results of those experiments in Chapter 5. Finally, in Chapter 6 we discuss the overarching conclusions we drew from this thesis along with how this project could be extended in the future.
Chapter 2

Related Works

This chapter discusses previous work that pertains to coevolutionary algorithms and decision making.

2.0.1 RIVALS

Our work is a direct extension of the RIVALS [2] project, a cybersecurity project that aims to model real world attackers and defenders in a peer-to-peer network. RIVALS is itself a successor to STEALTH [3]. RIVALS was tested on three network topologies (Figure 2-1). In this work, we tested our system on those same topologies but present work on a new topology on which we tested ESTABLO.

![Diagram](image_url)

(a) Topology 1.  
(b) Topology 2.  
(c) Topology 3.

Figure 2-1: Three previously tested topologies used in RIVALS.
2.0.2 Decision Making

ESTABLO is related to a study of how to create practical optimal output mechanisms for co-optimization algorithms [9] and multiple criteria decision making (MCDM). MCDM centers on the preferences of a decision maker (DM). It can be categorized based on when preference expression occurs: a priori – before solution selection, a posteriori – after generating a Pareto set, and progressively – interactively during solution identification [1]. ESTABLO is intended to be used in an a-priori context.

2.0.3 Coevolutionary Search

Coevolutionary algorithms are well suited to domains that have no intrinsic objective measure, also called interactive domains [8]. Competitive coevolutionary algorithms are used when the quality of a potential solution to the problem is determined by its performance when interacting with some set of tests. Coevolutionary algorithms and cyber security has been investigated by [13] [11] [2].
Chapter 3

Methods

This chapter discusses the methods that were used to answer the research questions we posed. In Section 3.1, we propose ESTABLO, a framework that allows us to compare individuals produced by different coevolutionary algorithms and select a “best” solution according to a specified ranking scheme. In Section 3.2, we present POOLING, a coevolutionary algorithm that uses multiple coevolutionary algorithms and combines their top solutions every generation. These top solutions are then used for all of the variants being used as the initial population for the next coevolution.

3.1 ESTABLO

The steps of ESTABLO’s method are:

- Execute multiple coevolutionary search algorithm variants,
- Populate a *compendium* of solutions and tests,
- Evaluate all tests on the compendium solutions and measure them along the

![Diagram of ESTABLO framework](image)

Figure 3-1: Overview of the ESTABLO framework for decision support through selection and visualization by using a compendium of solutions from coevolutionary algorithms

21
product of two axes: 3 different test sets and 3 different solution concepts.

- Select the top solution using a combined ranking scheme of 3 different solution concepts.

- Explain the top solution relative through visualization and data to support the decision maker.

These steps are shown in Figure 3-1.

A population in a coevolving algorithm can be viewed as tests in one perspective and solutions in the opposite. Since we are interested in optimized tests as well as optimized solutions, we first conduct steps 3-5 with defenders as solutions and attackers as tests. Then we repeat steps 3-5 after swapping the roles of solution and test so that attackers are considered solutions and defenders as tests. Pseudocode for ESTABLO can be seen in Algorithm 1.

3.1.1 Step 1. Coevolutionary Search

One ESTABLO trial runs, in parallel, each of five coevolutionary search algorithm variants once and populates a compendium with their solutions and tests, see Figure 3-2. The variants are:

- **Coev**: simple alternating coevolutionary algorithm that uses expected utility solution concept by assigning the average fitness when evaluating solutions [2].

- **MinMax**: simple alternating coevolutionary algorithm that uses the best worst case solution concept by assigning the minimum fitness against tests when evaluating solutions [2].

- **IPCA**: uses the pareto optimality solution concept by maintaining an updating non-dominated archive of solutions [4].

- **rIPCA**: also uses the pareto optimality solution concept stores non-dominated archive of solutions but removes dominated adversaries [2].
• **POOLING**: uses multiple variants as described in Section 3.2

The solutions and test archives of IPCA and rIPCA are also populated. All algorithms are started from an experimental configuration, see Figure 3-2, that determines their resource budgets. Without loss of generality, in this paper ESTABL0 only stores the final population and archive of each run in its compendium. ESTABL0 conducts multiple trials.

### 3.1.2 Step 2. Compendium Population

The compendium \( C, C^{m \times n}, m, n \in \mathbb{N} \), is populated with all of the individuals in each population (solutions and tests, or in our example, defenders and attackers) from each run from each algorithm, where the total number of defenders and attackers is \( n \) and \( m \), respectively. Through indexing ESTABL0 is able to track the run and algorithm variant of any individual.

### 3.1.3 Step 3. Evaluating the compendium

The fitness of each solution is only an estimate based on the testing context in which it has been evaluated. For example, the defenders generated by IPCA have only been evaluated against the attackers generated by IPCA. In this step ESTABL0 improves this fitness estimate along two axes. First, it compete solutions against different test sets. The test sets are:

- Baseline: all tests of the solution’s run.
- Compendium: all tests.
- Unseen: An unseen set of non-evolved tests.

Second, it measures solutions with three different solution concepts. The solution concepts are:

- MEU: Maximized Expected Utility
• Best Worst Case: best worst case solution using the fittest test.

• Pareto optimality: the Pareto front of an individual solution.

Solutions vs All Tests Competitions

The compendium fitness estimation pairs are stored in a competition scoring matrix $S^{m \times n}$ with cells containing a tuple of size 2 where each entry is calculated by $s_{ij} = (f_a^{(i,j)}, f_d^{(i,j)})$, where $f_a$ is the attacker fitness function and $f_d$ is the defender fitness function.

$$S = \begin{pmatrix}
(f_a^{(1,1)}, f_d^{(1,1)}) & \cdots & (f_a^{(1,n)}, f_d^{(1,n)}) \\
\vdots & \ddots & \vdots \\
(f_a^{(m,1)}, f_d^{(m,1)}) & \cdots & (f_a^{(m,n)}, f_d^{(m,n)})
\end{pmatrix}$$

From $S$, ESTABLO computes the three measurements of fitness across the compendium.

Compendium Quality on Unseen Tests

Because we are concerned with the possibility of evolutionary blind spots in testing, ESTABLO creates a set of tests that are not in the compendium. This aims to enhance anticipatory indication of the a-posteriori performance of the solution. ESTABLO generates the unseen set of tests by random sampling (or enumerating if there are few enough combinations) all possible tests. Then it removes all that are already in the compendium. It then calculates the smallest symmetric difference between each remaining test and all tests in the compendium and uses this information to construct a frequency distribution sample based on difference. It then randomly draws from this distribution with a bias that favors tests of small and large differences, i.e. very or not similar to the compendium. The results of these draws becomes its unseen test set. ESTABLO then calculates the fitness measurements of solutions over the set.
3.1.4 Step 4. Solution Selection

ESTABLO next anticipates that the decision maker is working with only \textit{a priori} information to guide it in selecting a top solution. For this reason, it scores and ranks with 4 different solution ranking schemes and selects the top ranked solution in terms of combination score as the top solution: They are:

- \textit{Average Fitness (AF)}, based on expected utility solution concept,
- \textit{Minimum Fitness (MF)}, based on best worst solution concept,
- \textit{Pareto Front (PF)}, based on inverse Pareto front ratio \( \text{(Pareto Front ratio = (Pareto Front Number) / (Number of Pareto Fronts)} \),
- \textit{Combination (CS)}, based on the summed normalized average fitness, minimum, and inverse pareto front ratio.

In the event of a tie, each ranking scheme first uses \textit{Average Fitness}, then \textit{Minimum Fitness}, and then \textit{Pareto front} to break the tie. If multiple solutions are still tied after this, the top solution is chosen at random from among the tied solutions.

3.1.5 Step 5. Solution Justification

ESTABLO next anticipates that the decision maker must understand the merit of the top solution. To this end, it provides a set of visualizations and statistics, demonstrated in Section 5. The first plot type is dense with information: it is a series of sub-plots, one for each ranking scheme. Each subplot shows the rank ordered solutions for one ranking schema and their scores. Scores for the other kinds of solution measurements on the baseline and unseen test set are also visualized using different line types, symbols and colors. For the MAP problem, the phenotype of the solutions - the set of attack nodes or the asset allocations of the defender is overlaid on a graph-based visualization of the network. A distance heat map for phenotypic pairwise distance gives an indication of diversity. On all visualizations, the top solution is annotated along with top solutions of the other three ranking schemes. We also
implemented a user interface described in Section 3.3 that offers a more interactive interface to the ESTABLO plots.

We note that algorithm and solution designers can find ESTABLO’s visualizations very helpful though they would be considering them with different purposes and without the reference point of a selected solution.

Algorithm 1 ESTABLO

1: procedure ESTABLOCOEV(populations, generations, variants)
2:    compendium ← ∅
3: for variant in variants do  \( \triangleright \) run coevolution per variant
4:    pop′ ← Coevolution(populations, generations, variant)
5:    compendium ← compendium ∪ pop′
6: StoreCompendium(compendium) \( \triangleright \) store compendium for analysis
3.2 POOLING Coevolutionary Algorithm

The steps of POOLING’s method are:

- Execute a single coevolutionary step of multiple coevolutionary variants,
- Determine what the top solutions are,
- Recombine and repeat.

Pseudocode for POOLING can be seen in Algorithm 2.

3.2.1 Step 1. Single Coevolutionary Step

POOLING initially runs four coevolutionary search algorithm variants for one generation and populates a pool with their solutions, see Figure 3-3. The variants POOLING uses are:

- Coev
- MinMax
- IPCA
- rIPCA

This occurs for both attackers and defenders and a separate pool is set up for each.

3.2.2 Step 2. Top Solutions

As the solutions in the pools only evolved against the solutions generated by their respective variants, POOLING then compares the solutions in each pool against all of the solutions found. It does a similar process as ESTABLO by creating a competition scoring matrix and calculates the inverse pareto front ratio of each solution. POOLING then ranks each solution by the combined ranking scheme as described in Section 3.1.4.
From these ranked solutions, POOLING then takes the top $N$ solutions where $N$ is defined before the experiment begins. This is done to ensure the population size does not explode as this occurs with experiments where POOLING is allowed to run for many generations.

### 3.2.3 Step 3. Recombine and Repeat

Once the top solutions have been found, they are fed back into step 1 and this process repeats. It is important to note that every variant receives a copy of the same population at the beginning of step 1.

### 3.3 User Interface

Another tool that was implemented to aid in viewing these results and helping decision makers was a responsive user interface. The graphs generated by ESTABLO are static and it can be cumbersome to try and find properties about individuals like phenotype from the graphs. A user interface using JavaScript and a library called
Algorithm 2 POOLING

1: procedure POOLINGCOEV(populations, generations)
2:     \( t \leftarrow 0 \)
3:     best_individuals \( \leftarrow \emptyset \)
4:     while \( t < \) generations do \( \triangleright \) run for # generations
5:         pool \( \leftarrow \emptyset \)
6:         for variant in variants do
7:             pop' \( \leftarrow \) SingleCoevolutionStep(populations)
8:             pool \( \leftarrow \) pool \( \cup \) pop'
9:         populations \( \leftarrow \) RankPopulations(pool)
10:        populations \( \leftarrow \) ExtractBest(populations)
11:       best_individuals \( \leftarrow \) populations
12:      \( t \leftarrow t + 1 \)
13:  return best_individuals \( \triangleright \) Returns best solutions found

(a) ESTABLO UI showing combined rank- (b) ESTABLO UI showing phenotypic distance

Figure 3-4: The ESTABLO UI can display phenotypic distances and data from any ranking scheme file generated by the ESTABLO analysis.

ECharts was used. This UI allows the user to select files that contain the ranked individuals generated by ESTABLO’s analysis and a similar graph will be displayed. The UI allows users to select how many top solutions they want included, the choice between displaying AF or MF, and will display a solution’s phenotype when hovered over. This aids with simpler viewing and manipulation of ESTABLO’s output. An example of what the UI looks like can be seen in Figure 3-4.
Chapter 4

Experiments

In this chapter we discuss the experiments that were run with ESTABLO and POOLING after they had both been implemented. We first describe each problem that both ESTABLO and POOLING were evaluated on and then the parameters that were used for each kind of experiment.

4.1 Mobile Asset Placement (MAP)

The Mobile Asset Placement (MAP) problem we use is based on the problem described in [10] with a few modifications. There are a set of nodes simulating a peer to peer network that uses the Chord protocol to relay messages throughout the network. An attacker can simulate a DDOS attack by making nodes unavailable for the entire mission and defenders have a set of tasks that consist of a start and an end node that need to be connected (simulating sending messages between the start and end node). A defender succeeds if none of their tasks are disrupted.

The problem we tested ESTABLO and POOLING on, varies from RIVALS in the following ways:

- Attackers DDOS attacks last for the entire mission
- The only routing protocol used is Chord
- Defenders’ tasks don’t have a timeout
4.1.1 Grammar

The grammar we used for attackers is:

\[
<\text{Attacks}> ::= \text{DDOSAttack(<node>)} \\
| \text{DDOSAttack(<node>), <Attacks>}
\]

\[
<\text{node}> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15
\]

and the grammar we used for defenders is:

\[
<\text{list}> ::= [\text{Task1(<asset>, <asset>), Task2(<asset>, <asset>),} \\
| \text{Task3(<asset>, <asset>), Task4(<asset>, <asset>),} \\
| \text{Task5(<asset>, <asset>)}]
\]

\[
<\text{assets}> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15
\]

4.1.2 Fitness Function

The fitness function for attackers is defined as:

\[
f_a = \frac{n_{\text{failed}}}{n_{\text{tasks}}} - \frac{n_{\text{attacks}}}{C \cdot n_{\text{tasks}}}
\]

\(n_{\text{tasks}}\) is the total number of tasks, \(n_{\text{failed}}\) is defined as how many tasks the attacker was able to disrupt, and \(n_{\text{attacks}}\) is the number of attacks the attacker used, \(C = 1000\). This formula helps with incentivizing attackers to disrupt tasks but with as few attacks as possible.

The defender fitness function is as follows:

\[
f_d = \frac{n_{\text{successful}}}{n_{\text{tasks}}} - n_{\text{same_nodes}} - n_{\text{duplicate_tasks}}
\]

\(n_{\text{tasks}}\) is the same as before, \(n_{\text{successful}}\) is defined as the total successful tasks, \(n_{\text{same_nodes}}\) is the number of tasks such that the start and end node are the same (path to self), \(n_{\text{duplicate_tasks}}\) is the number of duplicated tasks. This function helps incentivize defenders to succeed at as many tasks as possible while penalizing approaches that use trivial tasks (same start and end node) or duplicate tasks. A duplicate task is
defined as a task with the same start and end node as another task (this excludes inverted tasks e.g. \texttt{Task(1, 2)} and \texttt{Task(2,1)} are not duplicates).

4.2 ADHD

In ADHD \cite{7}, there are attackers and defenders in a network. Attackers are trying to scan and exploit the network to move laterally throughout it. In this scenario, attackers are already inside the network and are attempting to discover other parts of the network.

Defenders attempt to slow down or prevent attackers from exploiting the network. They are able to create/destroy honeypots, reroute traffic, change resources, disconnect a host, create a subnetwork, and detect an attacker. All of its actions are designed to try and slow down attackers and mutate the network to make the attacker’s goal difficult to achieve.

Attackers and defenders have a budget that dictates how they can spend their resources in order to achieve their goals.

4.2.1 Grammar

The grammar for attackers is:

\[
\text{<attack_budget>} ::= \text{ADHDControllerSR(initial_budget=<budget>, controllers=[])} \\
\text{<budget>} ::= [\text{<digit>}, \text{<digit>}, \text{<digit>}, \text{<digit>}]
\]

\[
\text{<digit>} ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
\]

and the grammar for defenders is:

\[
\text{<defense_budget>} ::= \text{ADHDControllerSR(initial_budget=<budget>, controllers=[])} \\
\text{<budget>} ::= [\text{<digit>}, \text{<digit>}, \text{<digit>}, \text{<digit>}, \text{<digit>}, \text{<digit>}]
\]

\[
\text{<digit>} ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
\]
4.2.2 Fitness Function

An attacker’s goal is to scan as many hosts and execute as many successful exploits as possible. As the defender’s goal is to hinder the attacker, their fitness is negative of the attacker’s fitness.

4.3 Adaptive ADHD

This problem is an extension of the ADHD problem defined in the previous section. This problem varies by running multiple trials of ADHD but attackers and defenders have dynamic budgets. Individuals in this problem can modify their budgets after each trial using a defined controller that uses the previous budget and fitness.

4.3.1 Grammar

The grammar for the attacker is:

\[ <\text{attack\_budget}> ::= \text{ADHDControllerSR}(\text{initial\_budget}=<\text{budget}>, \text{controllers}=<\text{controllers}>) \]
\[ <\text{digit}> ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \]
\[ <\text{budget}> ::= [<\text{digit}>, <\text{digit}>, <\text{digit}>, <\text{digit}>] \]
\[ <\text{controllers}> ::= [<\text{controller}>, <\text{controller}>, <\text{controller}>, <\text{controller}>] \]
\[ <\text{controller}> ::= \lambda x, y: <\text{fcn}> \]
\[ <\text{fcn}> ::= <\text{logic}> \mid <\text{expr}> \]
\[ <\text{logic}> ::= <\text{logic}> \text{if } y > <\text{digit}> \text{ else } <\text{logic}> \mid <\text{digit}> \mid -<\text{digit}> \]
\[ <\text{expr}> ::= <\text{expr}> <\text{op}> <\text{expr}> \mid <\text{var}> \]
\[ <\text{op}> ::= + \mid - \mid * \mid / \]
\[ <\text{var}> ::= x[0] \mid x[1] \mid x[2] \mid x[3] \mid y \]

the grammar used for defenders is:

\[ <\text{defense\_budget}> ::= \text{ADHDControllerSR}(\text{initial\_budget}=<\text{budget}>, \text{controllers}=<\text{controllers}>) \]
\[ <\text{digit}> ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \]
\[ <\text{budget}> ::= [<\text{digit}>, <\text{digit}>, <\text{digit}>, <\text{digit}>, <\text{digit}>, <\text{digit}>] \]
Table 4.1: Common coevolutionary parameters for each experiment.

<table>
<thead>
<tr>
<th>Parameter Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tournament size</td>
<td>2</td>
</tr>
<tr>
<td>Parent archive probability</td>
<td>0.9</td>
</tr>
<tr>
<td>Crossover probability</td>
<td>0.8</td>
</tr>
<tr>
<td>Mutation probability</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 4.2: Specialized parameters for varying mission size experiments and fitness function evaluation on the four topologies.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Population Size</th>
<th>Runs</th>
<th>Fitness evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>20</td>
<td>30</td>
<td>100000</td>
</tr>
<tr>
<td>ADHD</td>
<td>20</td>
<td>10</td>
<td>10000</td>
</tr>
<tr>
<td>ADHD Adaptive</td>
<td>20</td>
<td>5</td>
<td>2000</td>
</tr>
</tbody>
</table>

<controllers> ::= [<controller>, <controller>, <controller>, <controller>, <controller>, <controller>]
<controller> ::= lambda x, y: <fcn>
<fcn> ::= <logic> | <expr>
<logic> ::= <logic> if y > <digit> else <logic> | <digit> | -<digit>
<expr> ::= <expr> <op> <expr> | <var>
<op> ::= + | - | * | /

4.3.2 Fitness Function

This extension uses the same fitness functions as defined in Section 4.2.2.

4.4 Setup

This section discusses the parameters used for each problem and what experiments were run.

The common parameters used for each experiment are shown in Table 4.1.

The specific parameters used for each experiment are shown in Table 4.2.
Chapter 5

Results

This chapter discusses the results for each experiment we ran. Namely, it talks about how the solutions that ESTABLO found stack up against one another. In these results we can see that it is beneficial to run multiple algorithms as no single variant dominates the rest on both the attacking and defending side in both experiments. Figure 5-1 contains the legend for the figures shown in this section.

For some experiments that had multiple runs, only the runs that produced individuals that were at the top of a ranking scheme were included for readability.

When looking at these results, we wanted to be able to answer if we could use ESTABLO to rank individuals produced by different variants, produce solutions robust in the face of unseen adversaries, and determine if a “best” could be selected. We also wanted to see how POOLING would perform against the variants that it used for its coevolution.

Figure 5-1: Legend for figures in this section
5.1 MAP

When looking at what ESTABLO’s analysis produced, we see some interesting trends. When looking at the attacker solutions when ranked with the combined ranking scheme, the variants that dominate are Coev and POOLING and the top solution was produced by Coev (Figure 5-2, Table 5.1). When looking at the combined scores among the entire compendium, solutions of the same run, and against the unseen set of individuals, for many of the stronger individuals there is a large gap in performance. The top individuals’ scores increased dramatically when compared against the entire compendium. This is promising as it shows individuals that are more robust in the face of adversaries they have not seen.

We can also see from Figure 5-3 that even though the top solutions ranked by inverse pareto front lie on the same pareto front, they can still completely dominate a fairly large amount of individuals. This is again promising as it shows that the top solutions are able to completely beat an entire group of solutions.

When looking at defender solutions ranked by the combined ranking scheme, we see that the strongest solutions are produced by rIPCA, IPCA, and POOLING and the top solution was produced by rIPCA (Figure 5-4, Table 5.2). Coev and Minmax are able to produce some top solutions but the high end of these solutions is mostly dominated by the other three variants. Here, there is very little variation on combined score between the compendium, same run, and unseen solutions. The scores are tightly packed which shows that these individuals performed very similarly under these different scenarios. This shows that defenders will most likely perform against unseen attackers similar to how they would against attackers they have encountered.

We also see from Figure 5-5 that defenders display the same behavior as attackers. Namely that the top solutions are able to dominate a large amount of individuals.

The phenotypes of the top attacker and defender (ranked by combined score) are overlayed over the topology used in these experiments in Figure 5-6. The attacker solution has opted to complete disconnect the network. It has successfully removed enough nodes to make any mission fail. The top defender solution seems to have opted
Figure 5-2: This figure shows the combined score of attackers that were produced for the MAP problem. Only attackers generated from 3 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that the top solutions’ scores increased when placed into the compendium of solutions. This indicates robust attackers.

Figure 5-3: This figure shows the inverse pareto front of attackers that were produced for the MAP problem. Only attackers generated from 3 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that although there are not many fronts, a significant amount of individuals are still dominated by the top front. This indicates strong attackers.

for shorter physical connection in the network. This might not directly translate to the minimum number of hops required when using the Chord protocol but it is interesting to note.

5.2 ADHD

When running ESTABLO on the ADHD problem, we found that the variants we used produced a similar amount of top attacker solutions (Figure 5-7, Table 5.3). In this problem though, the top solution was produced by rIPCA. The combined score of attackers does drop drastically when attackers are considered in terms of the com-
Table 5.1: Number and ratio of attackers found in the top 25% of attackers ranked by combined score per coevolutionary variant for MAP. Here we can see that POOLING and Coev found the largest amounts and highest ratios of top attackers. This is over all runs (30 total runs).

<table>
<thead>
<tr>
<th>Variant</th>
<th># Solutions</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coev</td>
<td>258</td>
<td>0.39</td>
</tr>
<tr>
<td>IPCA</td>
<td>49</td>
<td>0.07</td>
</tr>
<tr>
<td>MinMax</td>
<td>31</td>
<td>0.05</td>
</tr>
<tr>
<td>POOLING</td>
<td>228</td>
<td>0.35</td>
</tr>
<tr>
<td>rIPCA</td>
<td>89</td>
<td>0.14</td>
</tr>
</tbody>
</table>

Figure 5-4: This figure shows the combined score of defenders that were produced for the MAP problem. Only defenders generated from 3 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that the top solutions’ scores remained relatively the same when placed into the compendium of solutions. This indicates that defenders are consistent in how they perform against unseen adversaries.

This might show that attacker solutions found using ESTABLO are not robust in the face of a more varied population. We can also see from Figure 5-8 that there are a large amount of fronts and the highest front is fairly exclusive which shows that the top solutions found are able to outperform most of the population.

From Figure 5-9 and Table 5.4, we can see that POOLING is able to produce a large amount of the top solutions (it also found the top defender as well). We can also see that the top defenders’ combined score is more consistent between individuals of the same run and the compendium which shows they are more robust than the attackers that were found. We also see the same behavior as attackers in terms of fronts (Figure 5-10).
Table 5.2: Number and ratio of defenders found in the top 25% of defenders ranked by combined score per coevolutionary variant for MAP. Here we can see that IPCA, POOLING, and rIPCA found the largest amounts and highest ratios of top defenders. This is over all runs (30 total runs).

<table>
<thead>
<tr>
<th>Variant</th>
<th># Solutions</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coev</td>
<td>55</td>
<td>0.10</td>
</tr>
<tr>
<td>IPCA</td>
<td>138</td>
<td>0.25</td>
</tr>
<tr>
<td>MinMax</td>
<td>54</td>
<td>0.10</td>
</tr>
<tr>
<td>POOLING</td>
<td>157</td>
<td>0.28</td>
</tr>
<tr>
<td>rIPCA</td>
<td>156</td>
<td>0.28</td>
</tr>
</tbody>
</table>

Table 5.3: Number and ratio of attackers found in the top 25% of attackers ranked by combined score per coevolutionary variant for ADHD. Here we can see that POOLING and Coev found the largest amounts and highest ratios of top attackers. This is over all runs (10 total runs).

<table>
<thead>
<tr>
<th>Variant</th>
<th># Solutions</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coev</td>
<td>51</td>
<td>0.22</td>
</tr>
<tr>
<td>IPCA</td>
<td>27</td>
<td>0.12</td>
</tr>
<tr>
<td>MinMax</td>
<td>42</td>
<td>0.18</td>
</tr>
<tr>
<td>POOLING</td>
<td>79</td>
<td>0.34</td>
</tr>
<tr>
<td>rIPCA</td>
<td>34</td>
<td>0.15</td>
</tr>
</tbody>
</table>

Table 5.4: Number and ratio of defenders found in the top 25% of defenders ranked by combined score per coevolutionary variant for ADHD. Here we can see that POOLING found the largest amount and highest ratio of top defenders. This is over all runs (10 total runs).

<table>
<thead>
<tr>
<th>Variant</th>
<th># Solutions</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coev</td>
<td>30</td>
<td>0.16</td>
</tr>
<tr>
<td>IPCA</td>
<td>21</td>
<td>0.11</td>
</tr>
<tr>
<td>MinMax</td>
<td>28</td>
<td>0.15</td>
</tr>
<tr>
<td>POOLING</td>
<td>81</td>
<td>0.44</td>
</tr>
<tr>
<td>rIPCA</td>
<td>26</td>
<td>0.14</td>
</tr>
</tbody>
</table>
Figure 5-5: This figure shows the inverse pareto front of defenders that were produced for the MAP problem. Only defenders generated from 3 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that there are many fronts and the top defenders are able to dominate a large portion of their peer population. This indicates strong defenders.

Figure 5-6: The top attacker is overlayed on this topology in red and the top defender in green.

5.3 Adaptive ADHD

In the adaptive ADHD problem, we see that for attackers rIPCA and POOLING produced most of the top performing individuals in Figure 5-11 and Table 5.5 (POOLING also produced the top attacker). All of the attackers’ combined score increased from when they were compared among the run they were produced and the entire population of the compendium. This shows that these attackers are robust in the face of new adversaries that they may not have encountered. We also see that there are multiple fronts which shows the top adversaries outperforming the others which again hints of robustness (Figure 5-12).

While for defenders, there doesn’t appear to be a variant that is able to produce
Figure 5-7: This figure shows the combined score of attackers that were produced for the ADHD problem. Only attackers generated from 3 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that the top solutions’ scores dropped drastically when placed into the compendium of solutions. This indicates that attackers are not robust in the face of unseen adversaries.

Figure 5-8: This figure shows the inverse pareto front of attackers that were produced for the ADHD problem. Only attackers generated from 3 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that there are many fronts and the top attackers are able to dominate a large portion of their peer population.

higher quality individuals (Figure 5-13, Table 5.6). It is promising to note that the combined score of each defender did increase when placed in the compendium of solutions which indicates that these solutions are robust. Figure 5-14 also shows multiple fronts as we saw for attackers.
Figure 5-9: This figure shows the combined score of defenders that were produced for the ADHD problem. Only defenders generated from 3 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that the top solutions’ scores remained consistent when placed into the compendium of solutions. This indicates that defenders are robust in the face of unseen adversaries.

Figure 5-10: This figure shows the inverse pareto front of defenders that were produced for the ADHD problem. Only defenders generated from 3 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that there are many fronts and the top defenders are able to dominate a large portion of their peer population.

Figure 5-11: This figure shows the combined score of attackers that were produced for the Adaptive ADHD problem. Only attackers generated from 4 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that the combined score of attackers increased when placed into the compendium. This indicates robust attackers.
Figure 5-12: This figure shows the inverse pareto front of attackers that were produced for the Adaptive ADHD problem. Only attackers generated from 4 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that there are many fronts and the top attackers are able to dominate a large portion of their peer population. This indicates strong attackers.

Table 5.5: Number and ratio of attackers found in the top 25% of attackers ranked by combined score per coevolutionary variant for Adaptive ADHD. Here we can see that POOLING and rIPCA found the largest amounts and highest ratios of top attackers. This is over all runs (5 total runs).

<table>
<thead>
<tr>
<th>Variant</th>
<th># Solutions</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coev</td>
<td>11</td>
<td>0.09</td>
</tr>
<tr>
<td>IPCA</td>
<td>14</td>
<td>0.11</td>
</tr>
<tr>
<td>MinMax</td>
<td>19</td>
<td>0.15</td>
</tr>
<tr>
<td>POOLING</td>
<td>49</td>
<td>0.39</td>
</tr>
<tr>
<td>rIPCA</td>
<td>32</td>
<td>0.26</td>
</tr>
</tbody>
</table>

Figure 5-13: This figure shows the combined score of defenders that were produced for the Adaptive ADHD problem. Only defenders generated from 4 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that the combined score of defenders increased when placed into the compendium. This indicates robust defenders.
Figure 5-14: This figure shows the inverse pareto front of defenders that were produced for the Adaptive ADHD problem. Only defenders generated from 4 runs are included to aid in readability (runs chosen as they produced some individual that was ranked number 1 in at least 1 ranking scheme). We see that there are many fronts and the top defenders are able to dominate a large portion of their peer population. This indicates strong defenders.

Table 5.6: Number and ratio of defenders found in the top 25% of defenders ranked by combined score per coevolutionary variant for Adaptive ADHD. Here we can see that no single coevolutionary variant is coming out on top. This is over all runs (5 total runs).

<table>
<thead>
<tr>
<th>Variant</th>
<th># Solutions</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coev</td>
<td>24</td>
<td>0.19</td>
</tr>
<tr>
<td>IPCA</td>
<td>24</td>
<td>0.19</td>
</tr>
<tr>
<td>MinMax</td>
<td>26</td>
<td>0.21</td>
</tr>
<tr>
<td>POOLING</td>
<td>24</td>
<td>0.19</td>
</tr>
<tr>
<td>rIPCA</td>
<td>27</td>
<td>0.22</td>
</tr>
</tbody>
</table>
Chapter 6

Conclusion

In this thesis we explored ways in which to visualize solutions for coevolutionary algorithms and aid in decision support using ESTABLO. We also explored combining multiple coevolutionary algorithms into one coevolutionary algorithm to create POOLING.

Using these approaches we were able to find strong solutions and justify selecting them under various different problems. We also found that POOLING was able to find top solutions that would outperform solutions generated by the coevolutionary algorithms it was built upon.

For future work we want to apply ESTABLO and POOLING to other problems like software defined networks and see how they perform. We also want to extend being able to generate a set of unseen solutions to any grammar passed in (right now it can only do so for MAP). We also want to test ESTABLO and POOLING with more coevolutionary variants like the ones described in [6].
Bibliography


