Genetic Programming and Evolvable Machines
https://doi.org/10.1007/s10710-020-09389-y

Adversarial genetic programming for cyber security: a rising application domain where GP matters

Una-May O'Reilly[1] · Jamal Toutouh[1] · Marcos Pertierra[1] · Daniel Prado Sanchez[1] · Dennis Garcia[1] · Anthony Erb Luogo[1] · Jonathan Kelly[1] · Erik Hemberg[1]

Received: 18 October 2018 / Revised: 18 September 2019

Abstract
Cyber security adversaries and engagements are ubiquitous and ceaseless. We delineate *Adversarial Genetic Programming for Cyber Security*, a research topic that, by means of genetic programming (GP), replicates and studies the behavior of cyber adversaries and the dynamics of their engagements. Adversarial Genetic Programming for Cyber Security encompasses extant and immediate research efforts in a vital problem domain, arguably occupying a position at the frontier where GP matters. Additionally, it prompts research questions around evolving complex behavior by expressing different abstractions with GP and opportunities to reconnect to the machine learning, artificial life, agent-based modeling and cyber security communities. We present a framework called *RIVALS* which supports the study of network security arms races. Its goal is to elucidate the dynamics of cyber networks under attack by computationally modeling and simulating them.

Keywords Genetic programming · Coevolutionary algorithms · Cyber Security

1 Introduction

The natural world yields many examples of adversarial advantages arising through evolution. These are testaments to biological arms races that have played out with complex dynamics, typically over macroscopic time scales. There is a wide range of

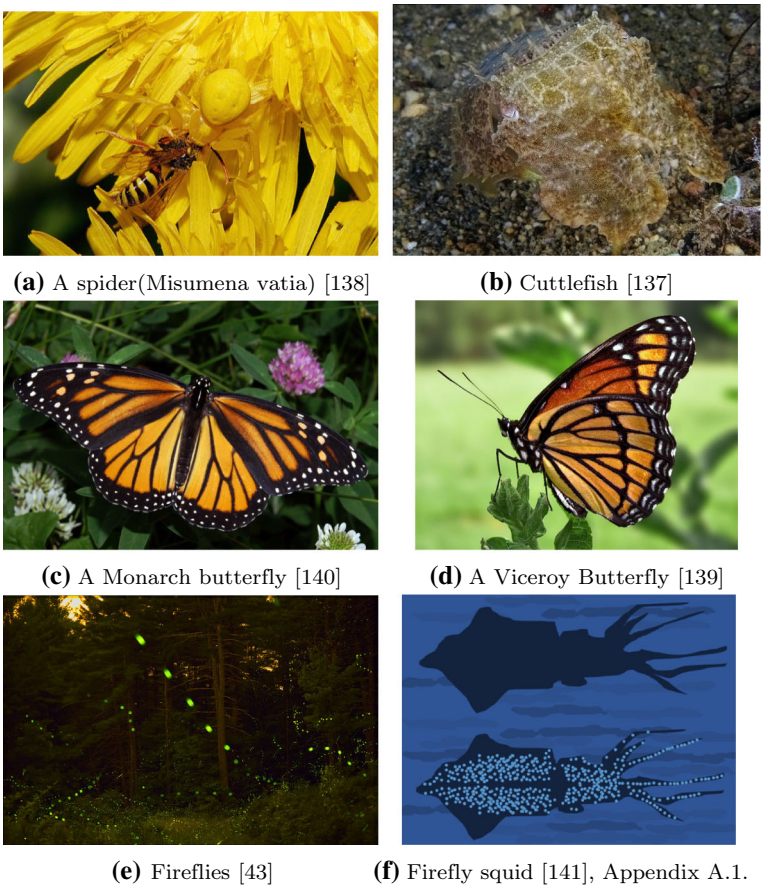
✉ Jamal Toutouh
toutouh@mit.edu

Una-May O'Reilly
unamay@csail.mit.edu

Erik Hemberg
hembergerik@gmail.com

[1] MIT CSAIL, Cambridge, USA

Published online: 02 April 2020



(a) A spider(Misumena vatia) [138] **(b)** Cuttlefish [137]

(c) A Monarch butterfly [140] **(d)** A Viceroy Butterfly [139]

(e) Fireflies [43] **(f)** Firefly squid [141], Appendix A.1.

Fig. 1 Adversarial defense and attack adaptions. Coloration adaptation—camouflage (**a**, **b**) and biomimicry (**c**, **d**). Bioluminescence defensive (**e**, **f**)

advantages evolved by predators and prey including examples of animal coloration—camouflage that is static such as the yellow crab spider (*Misumena vatia*) and dynamic such as the cuttlefish (see Fig. 1a, b), bioluminescence (e.g. mid-water crustaceans and fireflies, Fig. 1e, f), and mimicry (e.g. the Monarch and Viceroy butterflies, Fig. 1c, d).

Contemporary human-made ecosystems also yield arms races that arise from adversarial competitions. In the U.S. automotive industry, firms engage in an arms race to develop innovative new products ahead of the competition. The relationship between innovations and firm performance has been analyzed from a coevolutionary perspective [133] (see "Appendix A.2"). The *circular economy* and its related *cradle-to-cradle* idea describe manufacturer-customer relationships that cooperatively evolve while business ecosystems within the economy also evolve, though competitively [105]. In an example from taxation, tax shelters prey upon unintended lapses of the tax code

until defensive mediations address them. On occasion, however, these mediations either introduce new ambiguity or signal evaders to target a different weakness or inefficiency and the arms race continues [125,147].

One of the contemporary crucibles of adversarial activity and arms races is cyberspace. It is a complex, human-made ecosystem that spans networks (including the Internet), edge devices, servers, clouds, supercomputers. Each sub-ecosystem has sets of resources that collectively are a source of contention. Each includes human actors in circumstances where their software is proxy for their part in cyber adversarial engagements. Actors can have conflicting objectives. From a security perspective one population of actors have benign intentions and the other has malicious intentions. Benign "cyber actors" are conforming within social and legal norms. They may control data which they need to keep confidential or private. They may operate networks, or websites in order to conduct their enterprise's business—digital commerce. Different devices e.g. cellphones and laptops support everyday social and economic functions. Underpinning all these activities are computational hardware and software. Through attacks on these resources, malicious actors seek to disrupt, infiltrate and exfiltrate to steal, poison, extort so they can profit financially and advance their causes.

Focusing on network security, one pernicious kind of network attack is named Denial of Service (DOS). The goal of a DOS attack is to consume resources of a target so the target has none left to serve legitimate clients. At the extreme end of the range of volume are Distributed DOS (DDOS) attacks. These are extremely aggressive but more rare, likely due to the cost of setting them up. At a technical level, DDOS attacks are composed from a software toolkit. One set of software comprises multiple tactics that allow a network of compromised servers, i.e. a botnet, to be stealthily established. Another has tactics that enable, from a command and control location, a (human) controller to direct the bots to launch DOS attacks. These tactics can even react to defensive measures being deployed during an attack. The controller strategically selects the tactics and aims them at a target.

It is arguable that DDOS attacks are analogous to population members of a species, with tactical variation akin to biological variation. This population arguably undergoes evolutionary adaptation because ineffective attacks are not reused and effective ones undergo adaptation in order to circumvent defensive measures (i.e. reproductive selection and variation are present). With each new generation of attacks, the defensive measures themselves adapt or evolve then they face the next round of adaptation based on the toolkit. The attacker's general goal is to minimize its use of resources while maximizing its Denial of Service and the defender's general goal is to minimize the impact of the attack. In terms of dynamics, an escalating, adversarial arms race around engagements based on the conflicting objectives consequently emerges.

There is reliable and thorough documented evidence of DDOS attacks and counter measures that support an evolutionary interpretation, at a high level of abstraction [10]. MIRAI [130] is a notorious DDOS, receiving publicity for two high profile attacks: Krebs [24] and DYN [118]. From its inception to the present, MIRAI has adapted its tactics and targets to thwart defensive counter-measures, see Fig. 2. Figure 3 shows a Red Queen like DDOS dynamics—while evolution is churning beneath the surface, the number of attacks and their diversity of volume barely changes. Defensive gains are relative and transient.

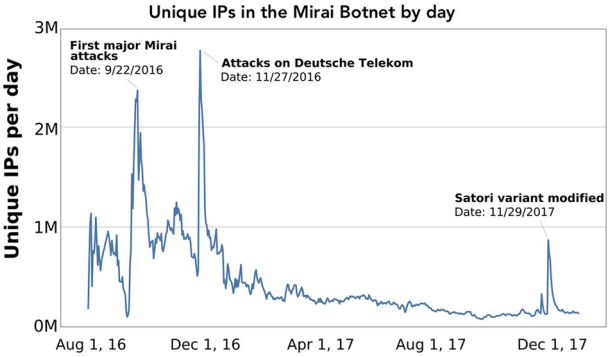


Fig. 2 Time series of size of bots (number of unique IPs of the compromised servers) in the MIRAI net. The size escalates prior to an attack. It plummets as the attack is repelled but adaptations allow it to sustain its size then regrow [6], modified for legibility [4]

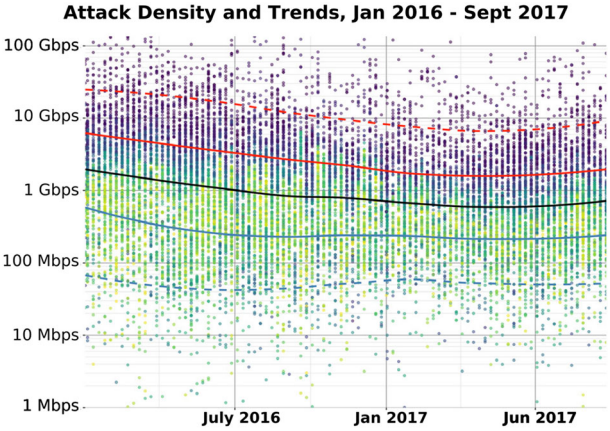


Fig. 3 Evolution is churning underneath the surface where we see DDOS attack densities over time, modified for legibility [5]

DOS attacks sit amongst an alarming array of other cyber attack types, such as Advanced Persistent Threats (APT) and Ransomware (see "Appendix A.3") that exhibit intelligent and agile attacks as well as attack versus defense arms race dynamics. Across the entire cyber ecosystem there is ceaseless evolution of kinds, arguably species, of attacks and co-evolving responses. Beyond their inconvenience, attacks risk lives, have financial costs, threaten businesses, impinge upon privacy and disrupt legitimate activity.

One way to improve defenses requires understanding better how an attacker behaves and how competing attacks and defenses react to one another. Feeding back information on *what could happen* is extremely valuable. It elucidates potential or past behaviors. For defensive design, knowing what could happen can help by *identifying unforeseen or particularly critical attacks*. Knowledge of arms race dynamics can inform the *adversarially hardening* of defensive designs *ahead of time*.

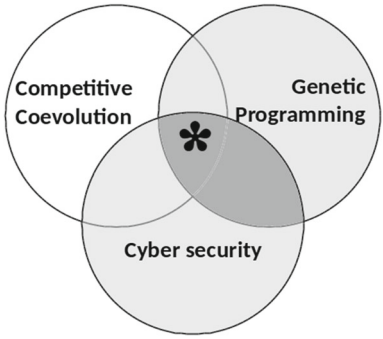


Fig. 4 Domain intersections of *Adversarial Genetic Programming for Cyber Security*

Understanding what could happen requires examining, actually playing out, or simulating, the dynamics of ongoing engagements between multiple parties with varying behaviors. Genetic programming (GP) can meet these requirements. It is ideally suited to represent the behavior of an adversary when it engages an opponent. Within a competitive evolutionary algorithm, GP representations can serve as the units of its selection and variation and GP operators can serve as the means of generating new solutions from existing ones.

Cyber security needs a way of exploring and executing attack versus defense engagements. GP is a paradigm that handles the performance selection and variation of units that *behave*! It can explore by directly manipulating executable functions without syntactic destruction or explicit program knowledge. It can express behavioral solutions drawn from a search space of hierarchical, varying length, executable structures and it has genetic operators that are able to blindly, at the representation level, generate novel syntactically correct solutions. Cyber security needs an expressively powerful way to describe abstracted attacks and defenses in a way that they are also able to actually compete against each other. GP can fulfill this because it can accommodate any abstraction that can be described as a set of *functions* and *terminals* or a computational language expressed by a grammar.

Combinations of adversarial evolutionary algorithms and GP matched with the rich problem domain of cyber security thus meld into an increasingly critical intersection with an agenda of compelling and still untapped scope, see Fig. 4. We name this topic *Adversarial Genetic Programming for Cyber Security*. It is *not* a new topic; it is the delineation of extant and current research approaches. We call attention to them because cyber security is urgent and escalating in challenge, and despite contributions to date, many novel approaches are required. At the 20 year mark of GP, when it is no longer necessary to show that GP works, it is timely to direct mature and nascent GP methods plus GP researchers in the direction of this critical domain with real world problems. Can search-based software engineering techniques gain traction on security exploits by considering adaptation at actual code level? Can GP enable the exploration of cyber space actors' goals and resulting plans, thereby addressing their intentions and cognitive reasoning? These questions and ones in between represent an opportunity for GP research that matters.

Paper Roadmap Having motivated *Adversarial Genetic Programming for Cyber Security*, in Sect. 2, we first succinctly cover GP and evolutionary algorithms (EAs). To consider adversarial evolution, we then describe competitive coevolutionary algorithms to remind readers of their complexity. We end the section by showing how GP and a coevolutionary algorithm can be combined. In Sect. 3 we survey prior evolutionary computation research into Artificial Intelligence (AI) and games (relevant because games are also competitive contexts), and software engineering (related because tests have been evolved to compete with program solutions to spur correctness). We then survey research around and within our central topic with Table 1 summarizing. Sections 2 and 3 inform the design motivations underlying a framework we present in Sect. 4. Named *RIVALS*, it combines competitive coevolutionary algorithms and GP for network security design. We describe three systems drawing on the framework and a component that extracts useful solutions to support the design decisions of a network defense designer. We draw to a close with a summary and a broad discussion of possible paths forward within this exciting, important paradigm.

2 Basis algorithms for adversarial evolution

Variable length genotypes, such as the hierarchical tree structures introduced by Koza [69], define large search spaces and improve the flexibility and power of Evolutionary Algorithms conducted by executable structure search [67,100]. GP is arguably defined by its distinctive structures that are variable length and executable plus its operators that preserve syntactic correctness while changing genotype size and structure. An Evolutionary Algorithm (EA) can evolve individual solutions in the form of executable structures, fixed length genotypes such as the bit strings used by Genetic Algorithms (GAs) [51] are inflexible.

Biological coevolution refers to the influences two or more species exert on each other's evolution [112]. A seminal paper on reciprocal relationships between insects and plants coined the word "coevolution" [40]. Coevolution can occur in the context of cooperation, where the species experience mutual benefit and adaptation, or in the context of competition, where the species negatively interact due to constrained resources that are under shared contention or a predator-prey relationship.

EAs usually abstract the evolutionary process while ignoring coevolution. To evaluate the quality of an individual, they apply an a priori defined fitness function and this function does not reflect the possible interactions between another individual or a dynamic environment. Optimality can be formulated by an absolute rather than relative objective. Coevolutionary algorithms extend EAs by basing the fitness of an individual on its interactions with other individuals or the environment. They mimic the coupled species-to-species interaction mechanisms of natural coevolution in order to solve a niche of search, optimization, design, and modeling problems [11,108,112,120].

Similar to the biology, coevolutionary algorithms are categorized as cooperative or competitive. Cooperative coevolutionary algorithms abstractly model the beneficial interaction between populations or in environments with time dependent factors [70]. They are frequently set up with multiple populations each solving a distinct sub-

problem of a larger problem. We continue this section by describing competitive coevolutionary algorithms in more detail as it is central to adversarial behavior.

2.1 Competitive coevolutionary algorithms

A basic competitive coevolutionary algorithm evolves two coupled populations, each with conventional selection and representation-aligned variation (crossover and mutation) operators which suit the representation of the population's genotype. One population comprises what is commonly called *tests* and the other *solutions*. We refer to individuals in a competitive coevolutionary algorithm generally as *adversaries*. In each generation, different competitions are formed by pairing up a test and a solution drawn from their respective populations. This couples the two population as they share a fitness evaluation component. We shall shortly describe different ways in which these competitions can be set up.

A test competes to demonstrate the solution as incorrect; this is its objective. The solution competes to solve the test correctly; this is its objective. In a security setting, it may be more apt to translate from *test* and *solution* to *attack* and *defense* as well as to refer to *engagements* rather than *competitions*. Fitness is calculated over all of an adversary's engagements. The dynamic of the algorithm, driven by conflicting objectives and guided by performance-based selection and random variation can gradually produce better and more robust solutions (i.e defenses) [112,120]. Generally, competitive coevolutionary algorithms suit domains in which there is no exogenous objective measure of performance but wherein performance is relative to others. These have been called *interactive* domains in [108] and include games and software engineering. In most domains a competition is often computationally expensive because it involves simulation or a complex testbed. We next describe how competition structuring addresses this challenge.

Competition structures For efficiency, the algorithm designer tries to minimize the number of competitions per generation while maximizing the accuracy of its fitness estimate of each adversary. The designer is able to control how competitions are structured and how many competitions are used to estimate the fitness of an adversary. Assuming one or both populations are of size N, two extreme structures are: *one-vs-one*, each adversary competes only once against a member of the opposing population, see Fig. 5a, d, and *all-vs-all*, each adversary competes against all members of the opposing population, see Fig. 5b, e. *One-vs-one* has minimal fitness evaluations[1] ($O(N)$) and strong competition bias. In contrast, *all-vs-all* has a quadratic number of fitness evaluations, yielding a high computational cost, $O(N^2)$ but weaker competition bias [120]. Other structures provide intermediate trade-offs between computation costs and competition bias, e.g. a *tournament* structure ranks individuals based on different rounds of peer competitions, see Fig. 5c.

In [89], adversaries termed hosts and parasites are placed on a $M \times M$ grid with a fixed neighborhood (size c) and one host and parasite per cell. The structure of the competitions is competition among all competitors in the neighborhood. Fitness

[1] Computational cost is shown for two populations.

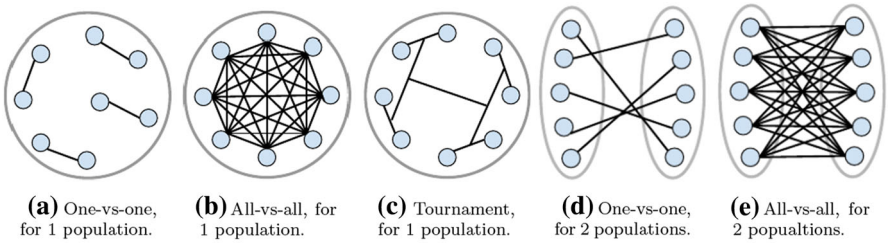
Table 1 Some representative related work to *cyber security*. It includes three different domains: games, security, and software, and three algorithmic solutions: GP (genetic programming), Comp Coev (Competitive coevolution), and combination of Comp Coev and GP

Reference	Algorithm	No. of Populations	Representation	Competition Structure	Fitness Scoring
Games					
Axelrod et al. [15,16]	Comp Coev GA	One	Bit strings	Tournament	Maximum expected utility
Crawford-Marks et al. [35]	Comp Coev GP	Two: players and smart-balls	Trees	One versus subset	Maximum expected utility
Harper [55]	Comp Coev GP	One	Variable integer vector	One versus a subset	Maximum expected utility
Keaveney and O'Riordan [63]	Comp Coev GP	One	Trees	One versus subset	Maximum expected utility
Lim et al. [76]	GP	One	Trees	Playing games	Pareto optimality
Luke et al. [81]	Comp Coev GP	One	Trees	One versus one	Maximum expected utility
Miles et al. [88]	Comp Coev GA	One	Bit strings	One versus one	Maximum expected utility
Security					
Garcia et al. [48]	Comp Coev GP	Two: defender and attacker	Integers and BNF grammar	NA	Multiple comparison
Hemberg et al. [56]	Comp Coev GP	Two: defender and attacker	Integers and BNF grammar	All versus All	Pareto optimality
Hingston and Preuss [57]	Comp Coev GA	Four: two of learners and two of tests	Pairs of real numbers	One versus all	Best-worst case
Kewley and Embrechts [64]	Comp Coev GA	Eight: four friendly and four enemy forces	Vector of four parameters	One versus a subset	Maximum expected utility
McDonald and Upton [86]	Comp Coev GA	Two: red and blue teams	Vector of parameters	One versus all	Force Exchange Ratio
Ostaszewski et al. [102]	Comp Coev AIS	Two: detectors and anomalies	Vector of parameters	One versus subset	Maximum expected utility

Table 1 continued

Reference	Algorithm	No. of Populations	Representation	Competition Structure	Fitness Scoring
Rush et al. [113]	Comp Coev EA	Two: defenders and attackers	Not specified	One versus subset	Maximum expected utility
Service and Tauritz [119]	Comp Coev GA	Two: system hardenings and system faults	Bit strings: unified power flow controller installation	One versus a subset	Nash equilibrium
Suarez-Tangil et al. [129]	GP	One	Trees: intrusion detection rules	Risk assessment	Maximum expected utility
Winterrose and Carter [145]	GA	One: strategies	Bit strings: finite states machine	Fixed scenarios	Success against the defender
Software engineering					
Arcuri and Yao [13]	Comp Coev GP	Two: programs and tests	Trees	One versus all	Nash equilibrium
Adamopoulos et al. [3]	Comp Coev GA	Two: programs and tests	Sequences of mutant programs and tests	One versus all	Maximum expected utility
Oliveira et al. [97]	Comp Coev GA	Two: programs and tests	Sequences of mutant programs and tests	One versus all	Pareto optimality
Wilkerson and Tauritz [142]	Comp Coev GP	Two: programs and tests	Trees and lists	One versus a subset	Maximum expected utility

The columns relate to properties described in Sect. 2.1, algorithm class, number of populations, representation of the individuals, competition structure and how fitness is assigned to an individual



(a) One-vs-one, for 1 population. **(b)** All-vs-all, for 1 population. **(c)** Tournament, for 1 population. **(d)** One-vs-one, for 2 populations. **(e)** All-vs-all, for 2 popualtions.

Fig. 5 Competition structures between adversaries/competitors

evaluations are reduced to $O(Mc^2)$ by this. An adversary has an outcome for each competition. We next discuss how these outcomes can be united to assign it a fitness score.

Fitness assignment At each generation an adversary needs to be assigned a fitness score derived from some function of its performance outcomes in its competitions. For example, the function can average the performance outcomes or use the maximum, minimum, or median outcome [15]. Other approaches have been defined depending on the specific problem domain, e.g. [9,27,44,76,81,135]. A more formal approach, using the *solution* and *test* perspective, describes fitness assignments as *solution concepts* [108]. Solution concepts include: best worst case, maximization of expected utility, Nash equilibrium, and Pareto optimality.

Coevolutionary algorithms are more complex and challenging to direct toward some expected outcome than a single population EA. We next explain the pathologies they exhibit and describe accepted remedies for them.

Pathologies and remedies The interactive aspect of fitness implies that a competitive coevolutionary algorithm lacks an *exact* fitness measurement. While EAs use a fitness function that allows an individual to be compared and globally ranked, in coevolutionary algorithms two members of the same population, during selection, may be imprecisely compared when they did not compete against the same opponents. Adding more imprecision, regardless of the function that computes a fitness score, an adversary's score may change if it later competes against different opponents. These properties imply that any ranking of individuals is a noisy estimate. This estimation can lead to competitive coevolutionary algorithms *pathologies* that limit the effectiveness of their arms race modeling [108]. Furthermore, pathologies can arise from competitive imbalance, particularly when the behavior space of one population is not the same as that of the other. This asymmetry is considered and addressed in [98]. Pathologies include: *disengagement*—occurring when opponents are not challenging enough, eliminating an incentive to adapt, or when opponents are too challenging and progress becomes impossible [28], *cyclic dynamics*—generally appearing in transitive domains (A beats B, B beats C and C beats A) as oscillation of the evaluation metric [58], *focusing* or *overspecialization*—arising when an adversary evolves to beat only some of its opponents, while neglecting the others [26], and *coevolutionary forgetting*—occurs when an adversary's ability to beat an opponent evolves away [23].

In order to remediate these pathologies, assorted methods have been proposed [22,23,26,28,42,58,70,144]. These center on archives or memory mechanisms, maintaining suboptimal individuals in the population, using a neighborhood spatial structure, and/or monitoring the progress of the algorithm.

With the basics of GP and competitive coevolutionary algorithms introduced, we are now able to present a combined competitive coevolutionary and GP algorithm.

2.2 GP and adversarial evolution

Combining GP and competitive coevolutionary algorithms enables *cyber security* arms races to be replicated by evolving executable adversaries. An example of an alternating GP competitive coevolutionary algorithm [14] is shown in Algorithm 1. The adversaries (attacker/defender) are coupled and evolve in an alternating manner. First, the adversaries are initialized. Then, at each generation, a new attacker population, $\mathbf{A}_t$, is selected, mutated, recombined and evaluated against the current defense population, $\mathbf{D}_{t-1}$. Based on the evaluation the attackers are replaced. Then, control reverts to the defender population so it can evolve to conclude the generation. This algorithm allows more attack evolution than defensive evolution, reflecting what can happen in real cyber systems. A note of caution about this approach in the context of algorithms, rather than reality, is offered by [87] where it is observed that moderate environmental variation across generations is sufficient to promote the evolution of robust solutions.

Algorithm 1 Example of alternating GP Competitive coevolutionary algorithm
Input:
T: number of generations $\mathcal{L}$: Fitness function, $\mathcal{F}$: Functions, $\mathcal{T}$: Terminals
μ: Mutation probability, ξ: Crossover probability, N: Population size

$\mathbf{A}_0 \leftarrow [\mathbf{a}_{1,0}, \ldots, \mathbf{a}_{N,0}] \sim \mathcal{U}([\mathcal{F}, \mathcal{T}])$ ▷ Initialize minimizer(attacker) population
$\mathbf{D}_0 \leftarrow [\mathbf{d}_{1,0}, \ldots, \mathbf{d}_{N,0}] \sim \mathcal{U}([\mathcal{F}, \mathcal{T}])$ ▷ Initialize maximizer(defender) population
$t \leftarrow 0$ ▷ Initialize iteration counter
repeat
  $t \leftarrow t + 1$ ▷ Increase counter
  $\mathbf{A}_t \leftarrow select(\mathbf{A}_{t-1}))$ ▷ Selection
  $\mathbf{A}_t \leftarrow mutate(\mathbf{A}_t, \mu))$ ▷ Mutation
  $\mathbf{A}_t \leftarrow crossover(\mathbf{A}_t, \xi))$ ▷ Crossover
  $\mathbf{a}'_*, \mathbf{d}'_* \leftarrow \arg\min_{\mathbf{a} \in \mathbf{A}_t} \arg\max_{\mathbf{d} \in \mathbf{D}_{t-1}} \mathcal{L}(\mathbf{a}, \mathbf{d})$ ▷ Best minimizer
  if $\mathcal{L}(\mathbf{a}'_*, \mathbf{d}'_*) < \mathcal{L}(\mathbf{a}_{N,t-1}, \mathbf{d}_{N,t-1})$ **then** ▷ Replace worst minimizer
    $\mathbf{a}_{N,t-1} \leftarrow \mathbf{a}'_*$ ▷ Update population
  end if
  $\mathbf{A}_t \leftarrow \mathbf{A}_{t-1}$ ▷ Copy population
  $t \leftarrow t + 1$ ▷ Increase counter before alternating to maximizer
  $\mathbf{D}_t \leftarrow select(\mathbf{D}_{t-1}))$ ▷ Selection
  $\mathbf{D}_t \leftarrow mutate(\mathbf{D}_t, \mu))$ ▷ Mutation
  $\mathbf{D}_t \leftarrow crossover(\mathbf{D}_t, \xi))$ ▷ Crossover
  $\mathbf{a}'_*, \mathbf{d}'_* \leftarrow \arg\min_{\mathbf{a} \in \mathbf{A}_t} \arg\max_{\mathbf{d} \in \mathbf{D}_t} \mathcal{L}(\mathbf{a}, \mathbf{d})$ ▷ Best maximizer
  if $\mathcal{L}(\mathbf{a}'_*, \mathbf{d}'_*) > \mathcal{L}(\mathbf{a}_{N,t}, \mathbf{d}_{N,t-1})$ **then** ▷ Replace worst maximizer
    $\mathbf{d}_{N,t-1} \leftarrow \mathbf{d}'_*$ ▷ Update population
  end if
  $\mathbf{D}_t \leftarrow \mathbf{D}_{t-1}$ ▷ Copy population
until $t \geq T$
$\mathbf{a}_*, \mathbf{d}_* \leftarrow \arg\min_{\mathbf{a} \in \mathbf{A}_T} \arg\max_{\mathbf{d} \in \mathbf{D}_T} \mathcal{L}(\mathbf{a}, \mathbf{d})$ ▷ Best minimizer
return $\mathbf{a}_*, \mathbf{d}_*$

There are some open challenges when applying *Adversarial Genetic Programming for Cyber Security*. Some of them are inherent in the use of EAs and adversarial evolution, such as the subjective solution evaluation, i.e., individuals interact with different opponents and the fitness is based on these, so it only provides a relative ordering [107,109,110]. Other amplified challenges are the evaluation cost, since the individual's fitness is calculated based on its interaction with other individuals which might require high computational costs [17,66,78,79,89]; Finally, combining GP and adversarial evolution generates complex models that complicate the algorithm operator and parameter selection.

This section has described algorithms that form the foundation of computational adversarial evolution. In the next section we describe prior Evolutionary Computation work that use GP, competitive coevolutionary algorithms or a combination of them in the domains of AI and Games, security, and Software Engineering.

3 Related work

We now turn our attention to prior work relating to the theme of adversarial contexts. We organize by application domain, starting with domains which are competitive in nature but not cyber security: AI and games (Sect. 3.1) and Software Engineering (Sect. 3.2). Most examples take technical approaches that use competitive coevolutionary algorithms or GP. They provide essential illustrations of how the algorithms can be used as building blocks. Table 1 catalogs the related work by domain, algorithm, representation, competition structure, and fitness scoring. Figure 6 illustrates how they intersect thematically.

3.1 AI and games

We find related work in AI and games because many games are competitive and game playing is a fertile research ground for investigating adversarial learning. We embrace a broad definition of game—encompassing board games, e.g. [106], video game playing, e.g. [63,121] and social science games e.g. [15].

One of the first competitive coevolutionary algorithms was used to find efficient strategies for the Iterated Prisoner's Dilemma [15,16]. This seminal project used a single population and it structured its strategy competition by running a tournament among the population members. A single population is typical for symmetric games, i.e. games where each player has the same set of moves (behavioral repertoire) and objectives. We find symmetric board game projects that use single population competitive coevolutionary algorithms to evolve Tic-Tac-Toe, Othello and Backgammon players. The Tic-Tac-Toe project represented players with GP [9] while both Othello and Backgammon projects used temporal difference learning, which is a gradient-based local search method [71,106,123,131,132]. A subsequent work used symmetric competitive coevolution in games and demonstrated impressive results through self-play [30]. A noteworthy system at the intersection of competitive coevolutionary algorithms and GP used a single population of agents cooperating as a team to com-

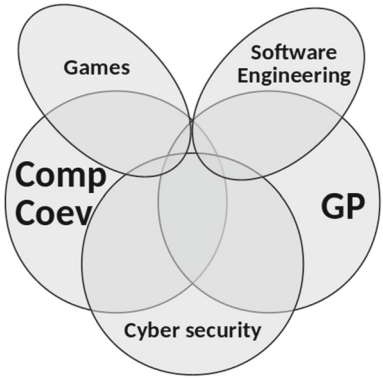


Fig. 6 Venn diagram showing the intersections of domains in *Adversarial Genetic Programming for Cyber Security* related work

pete for the "RoboCup97" soccer competition [81]. Soccer, being symmetric, would allow the system to potentially train against itself. In [122] the authors designed an evolutionary strategizing machine for game playing and other contexts. In [124], the authors coevolved strategies to solve Rubik's Cube where the test population evolved the Rubik's Cube configurations while the learner population evolved GP individuals to solve the Cube configurations. An ablation study demonstrated the contribution of competitive coevolution over random replacement of Cube configurations.

Cyber security adversaries are arguably asymmetric. In asymmetric games where opponents have different objectives and move sets, we find multi-population competitive coevolutionary algorithms. These include a competitive coevolutionary algorithm with evolutionary strategies that evolves Pac-Man versus Ghost Team players [27] and a PushGP system coevolving player versus smart-ball for the game "Quidditch". The latter is noteworthy for its intersection of a competitive coevolutionary algorithm and GP.

In some asymmetric games one adversary is external to the learning system and quite complex. This drives single or multi-population coevolutionary algorithm systems. A competitive coevolutionary algorithm and GP study compared using a single population to nine populations when evolving controllers for a car racing game [135]. The multi-population approach generally produced better controllers. Similarly, real time strategy games (which provide challenging AI problems [72,95]) have applied a single-population competitive coevolutionary algorithm and GP to develop automated players that use a progressive refinement planning technique [63]. Other work [88] coevolved real time strategy players by representing the AI agents with Influence Maps [38]. A real-time Neuroevolution of Augmenting Topologies (rtNEAT) approach evolved neural networks in real time while a game was played [128]. Behavior Trees (BT) were introduced to encode formal system specifications [7], but have also represented game AI behavior in commercial games [94]. GP evolved BT controllers for "Mario" [103] and "DEFCON" [76] games. Another study used Grammatical Evolution to generate Java programs to compete in "Robocode". The players were represented by programs [55] and a spatial and temporal competitive coevolutionary algorithm and GP were used.

3.2 Software engineering and software testing

Software testing called *fuzzing* arguably started with [54] which used grammars to generate program inputs for tests. This transitioned to direct automated random testing. One relevant security example is [50] that searched for bugs in a security protocol. Later, constraint based automatic test data generation was used to generate tests to verify if software variants were safe from malicious manipulation [39].

For similar purposes, the search based software engineering community uses evolutionary computation. Coevolution is used in SBSE with GA representations, see e.g. [3,8,97]. The community has used competitive coevolutionary algorithms and GP to coevolve programs and unit tests from their specification, e.g. [13,14,142]. Another study [18], used coevolution to distinguish correct behavior from incorrect.

3.3 Cyber security

We now turn to the domain of cyber security. There are a number of examples where, without any computational modeling or simulation, cyber security dynamics are described as evolutionary or as arms races. According to [143], *cyber security* is the task of minimizing an *attack surface* over time. The attack surface is the portion of a system which has vulnerabilities. Attackers attempt to influence the system's nominal state and operation by varying their interactions with the attack surface by stealth and non-compliance (see "Appendix A.4"). *Cyber security* perimeter protection (e.g. a firewall) is a battle fought and lost. Attackers are able to now gain access and defenders must resort to strategies that assume the attacker is present but hidden. One tactic to detect their presence is deception. For this a "honeypot" that entraps an attacker by imitating its target is commonly deployed. This arms race escalates *ad infinitum* as attackers then anticipate what the defenders have anticipated (e.g. the honeypots) and so on.

Retrospective security event reports also document coevolution. For example, [53] is an empirical study of malware evolution. Arguments for employing nature-inspired technologies for *cyber security* that mention how biological and ecological systems use information to adapt in an unpredictable world include [34,45,60,85,114].

A selected set of cyber security research that takes technical approaches is described in the Security subsection of Table 1. This set include works that we would not consider Adversarial Genetic Programming for Cyber Security: one work uses a genetic algorithm and neither coevolution or GP [145] and six others use are adversarial in nature, i.e. they use coevolution, but not GP [57,64,86,102,113,119]. The remaining three projects fit within the topic of Adversarial GP [48,56,129]. The research within this set, along with other related work, can also be distinguished by what specific application it addresses in the domain of cyber security:

Moving target defense (MTD) techniques seek to randomize components to reduce the likelihood of a successful attack, reduce the attack lifetime, and diversify systems to limit the damage [41,96]. A MTD study investigates an adaptable adversarial strategy based on Prisoners Dilemma in [145]. Strategies are encoded as binary chromosomes

representing finite state machines that evolve according to GA. The study has one adaptive defender population in GA and few fixed scenarios.

Network defense investigation is studied with the coevolutionary agent-based network defense lightweight event system (CANDLES) [113] a framework designed to coevolve attacker and defender agent strategies with a custom, abstract computer network defense simulation. The *RIVALS* network security framework, elaborated in Sect. 4 supports three studies into respectively DDOS, deceptive and isolation defense.

Self-adapting cyber defenses are for example the Helix self-regenerative architecture [74]. Helix shifts the attack surface by automatically repairing programs that fail test cases using Software Dynamic Translation, e.g. Strata [117]. Another example of automated fault analysis and filter generation within a system is named "FUZZBUSTER" [91,92]. Like Helix, FUZZBUSTER is designed to automatically identify software vulnerabilities and create adaptations that protect those vulnerabilities before attackers can exploit them. Both FUZZBUSTER and Helix use a GP system called GenProg [136] for automatically fixing code.

Physical infrastructure defense is studied in terms of how network components can be made resilient in [119].

Anomaly detection attempts to discern network or other activity behavior that is out of the ordinary or that differs from normal. In the auto-immune system computational paradigm, normal has been characterized as "self" [46]. Anomaly detection has been studied as a one class learning problem by [102] who use the artificial immune systems paradigm and coevolution for search. Attempts to build GP anomaly detectors have mostly assumed labeled data sets and they evolve a classifier that labels outputs as normal or not. They frequently encounter a class imbalance issue. This is explicitly addressed in [126]. Later work adopted an explicitly Pareto archive formulation of competitive coevolution [75].

Vulnerabilty testing Vulnerability testing involves testing a system with modifications to known exploits or attack vectors. Examples of developing mimicry attacks to test for vulnerability are found in [61,62]. The approach used the alarm signal for coevolution of a GP exploit generator. Moreover, GP was limited to instructions that were in legitimate applications, forcing GP to search for exploits described in terms of instructions used by legitimate applications. Others have also used GP to coevolve port scans against the 'Snort' intrusion detector with the objective of evolving port scans that demonstrate holes in the IDS. See for example [73]. Vulnerability testing for malware in mobile applications using coevolution appears in [25].

Malware detection has seen GP used to evolve novel PDF malware to automatically evade classifiers [148] and to study malware in the form of return-oriented program evolution [47].

Intrusion detection Most examples of intrusion detection are addressed as a multi-class classification problem where activity such as network traffic must be labeled as normal, Denial of Service, botnet or something else. Multi-class classification has been studied using GP without coevolution. For example, [129] optimizes intrusion detection rules. Others study botnet detection with GP under streaming data label budgets and class imbalance [65]. With both intrusion detection and anomaly detection, one challenge is obtaining a dataset that truly reflects the network or any other cyber

environment. For example, a widely used dataset known at KDD'99 has been criticized for its artificially generated normal data, which did not encompass the diversity in real normal behavior, see [77,84] for more details. It is also challenging to label and maintain datasets, see [49] as an example. The persistent nature of these challenges provides additional motivation for *RIVALS*.

Battle management has historically noteworthy work by [12] which focuses on security in battle management. It was novel in considering semi-autonomous control of several intelligent agents; plan adjustment based on developments during plan execution; and the influence of the presence of an adversary in devising plans. Similar but contemporary work on a computational military tactical planning system uses fuzzy sets and competitive coevolutionary algorithm with a battlefield tactics simulator for decision support [64].

Red teaming is a technique utilized by the military operational analysis community to check defensive operational plans by pitting actors posing as attackers (a red team) against the defense (a blue team). This activity has evolved from human teams to teams of programmers overseeing automated attacks tactics and defensive measures. See [37] and studies [1,93] trying to automate the exercise to use less manpower, or use it more efficiently with agent-based models [19,57,86,146,149,150].

The next section Prior work in AI and games, cyber security and software engineering domains is helpful in showing how competitive coevolutionary algorithms and GP can be used in adversarial contexts. It also provides a modest number of examples of GP combined with a competitive coevolutionary algorithm. We now proceed to present a detailed example of research within *Adversarial Genetic Programming for Cyber Security*. Named *RIVALS*, it is a software framework for studying network security. *RIVALS* addresses an aspect of central importance to the 2016 DARPA grand challenge in cyber security named "The World's First All-Machine Hacking Tournament" [37]. This aspect that adaptation of both sides of a cyber security battle ground must be anticipated and that eventually posture reconfiguration needs to be fully automated.

4 RIVALS

Consistent with the *Adversarial Genetic Programming for Cyber Security* paradigm, our goal is to study the dynamics of cyber networks under attack by computationally modeling and simulating them. Ultimately we aim to provide defenders with information that allows them to anticipate attacks and design for resilience and resistance before their attack surfaces are attacked. We exploit competitive coevolutionary algorithms and GP for these purposes within a framework named *RIVALS* [101,104,111].

Conceptually, the framework consists of three elements: *adversaries*, *engagements and their environments*, and *competitive coevolution*. These fit together as two connected modules—one executing the coevolutionary algorithm and executing the engagements, and the other executing the engagements with the competitive coevolutionary algorithm directing attack and defense engagement pairings, see Fig. 7.

We now proceed to describe each of these elements and *RIVALS*' use cases.

Fig. 7 High level view of *RIVALS* framework. The left hand side component is a competitive coevolutionary algorithm that evolves two competing populations: attacks and defenses. The right hand side component from a computational perspective is a modeling or simulation environment. The environment is initialized with a mission and can be reset each time it is passed an attack–defense pairing to run an engagement. It first installs the defense, then it starts the mission and the attack. It evaluates the performance of the adversaries relative to their objectives and returns appropriate measurements

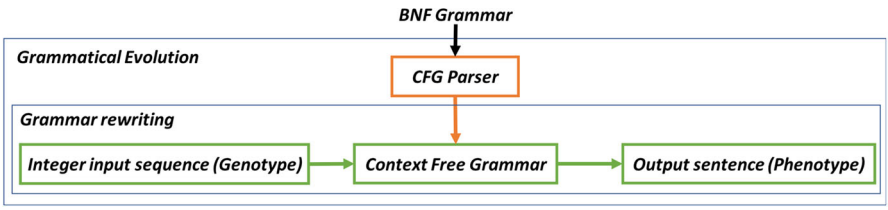


Fig. 8 Grammatical evolution string rewriting. A context free grammar rewrites an integer sequence (genotype) to an output sentence (phenotype)

4.1 Elements of *RIVALS*

Adversaries The system consists of two adversarial populations—*attacks* and *defenses*. The primary objective of an *attack* is to impair a mission by maximizing disruption of some resource. The primary objective of a *defense* is to complete a mission by minimizing disruptions. Both attack and defense can have secondary objectives based on the cost of their behavior. We design attack and defense behavior by defining grammars that can express the different variations of attacks and defenses. An attack or defense is a "sentence" in the grammar's language. We implement a rewriting process ("generator") to form sentences from the grammar [99], see Fig. 8. The sentences (i.e. attacks and defenses) are functions that are executable in the layers of the engagement environment.

The grammar has Backus Naur Form (BNF) and is parsed to a context free grammar structure. The (rewrite) rules of the generator express how a sentence, i.e. attack or defense, can be composed by rewriting a start symbol. The adversaries are represented with variable length integer vectors. The generator decodes these vectors to control its rewriting. As a result, different vectors generate different attacks or defenses. For different use cases, it is only necessary to change the BNF grammar, engagement environment and fitness function of the adversaries. This modularity, and reusability of the parser and rewriter are efficient software engineering and problem solving advantages. The grammar additionally helps communicate to the designer how the system works and its assumptions, i.e. threat model. This enables conversations and validation at the domain level with experts and increases the confidence in solutions from the system.

Engagements and the engagement environment An engagement is an attack on a defense. Engagements take place in an *engagement environment* which is initialized with a mission to complete and a set of resources such as network services that support the mission. The defense is first installed in the environment and then, while the mission runs, the attack is launched. The scenario (mission and resources) and attacks are then executed. Engagements have outcomes that match up to objectives; they are phrased in terms of mission completion (primary objective) and resource usage (second objective). Implementation-wise, the engagement environment component can support a problem-specific network testbed, simulator or model. Mod-sim is appropriate when testbeds incur long experimental cycle times or do not abstract away irrelevant detail.

Coevolutionary algorithms *RIVALS* maintains two *populations* of competing attackers and defenders. It calculates the fitness of each population member (in both attack and defense populations) by assessing its ability to successfully engage one or more members from the adversarial population, given its objective(s). It also directs selection and variation. *RIVALS* [104,111] utilizes different coevolutionary algorithms to generate diverse behavior. The algorithms, for further diversity, use different "solution concepts", i.e. measures of adversarial success. However, engagements are often computationally expensive and have to be pairwise sampled from two populations at each generation, a number of enhancements enables efficient use of a fixed budget of computation or time.

RIVALS' compendium Solely emulating or simulating cyber arms races is not sufficient to practically inform the design of better, anticipatory defenses. In fact, competitive coevolution poses general challenges when used for design optimization. The following ones in *RIVALS* make it difficult to present a designer with clear information derived solely from multiple simulation runs [111,115]:

1. Attacks (and defenses) of different generations are not comparable because fitness is based solely on the composition of the defense (attack) population at each generation. So no clear champion emerges from running the algorithm.
2. From multiple runs, with one or more algorithms, it is unclear how to automatically select a "best" attack or design.

To this end, *RIVALS* provides an additional decision support component, named `ESTABLO` [111,115], see Fig. 9. At the implementation level, the engagements and results of every run of any of the system's coevolutionary algorithms are cached. Later, offline, `ESTABLO` filters these results and moves a subset to its *compendium*. To prepare for the decision support analysis, it then competes all the attacks in the compendium against all defenses and ranks them according to multiple criteria, e.g. maximum-expected utility, best-worst case. For the defensive designer, it also provides visualizations and comparisons of adversarial behaviors to inform the design process.

Use cases The *RIVALS* framework supports use cases—investigations into arms races of a particular cyber network context. They each interface with the framework through a set of domain specific information. For each use case, grammars define the behavioral space of the adversaries, objectives define the goals of the adversaries for

Fig. 9 Overview of the ESTABLO framework used by *RIVALS* for decision support through selection and visualization by using a compendium of solutions from coevolutionary algorithms

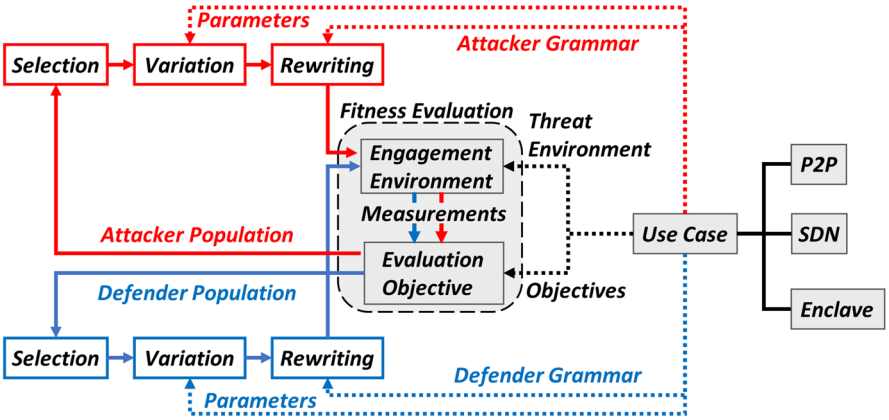


Fig. 10 *RIVALS* framework component decomposition showing interface with use cases

scoring fitness and the threat environment describes its engagement environment. For each use case, different parameters to control the coevolutionary algorithm are also available. Fig. 10 depicts the high level decomposition of the framework and shows how a use case interfaces with it. Table 2 presents the domain specific information for three use cases. By name, these are:

– *DIFFRACTED* Defending a peer-to-peer network against Distributed Denial of Service (DDOS) attacks [48] (Sect. 4.2)
– *AVAIL* Defenses against device compromise contagion in a segmented enterprise network [56] (Sect. 4.3), and
– *DARK-HORSE* Deceptive defense against the internal reconnaissance of an adversary within a software defined network [104] (Sect. 4.4)

The following sections elaborate on these use cases.

4.2 *DIFFRACTED*: dos attacks on peer-to-peer networks

A peer-to-peer (P2P) network is a robust and resilient means of securing mission reliability in the face of extreme distributed Denial of Service (DDOS) attacks. *DIFFRACTED* [48], assists in developing P2P network defense strategies against DDOS attacks. Attack completion and resource cost minimization serve as attacker objectives. Mission completion and resource cost minimization are the reciprocal defender objectives. DDOS attack strategies are modeled with a variety of behavioral languages.

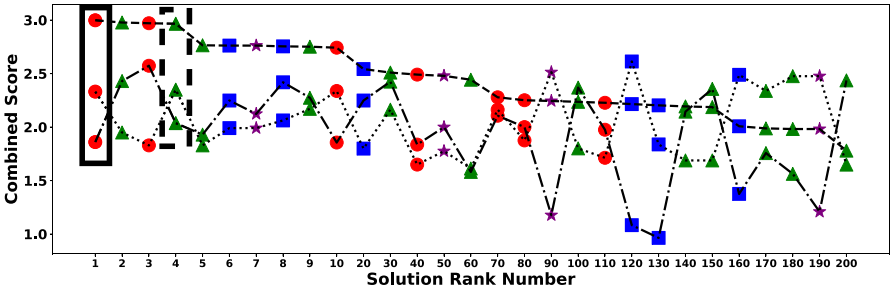


Fig. 11 The x axis shows a sorted subsample of attackers (note, the top 10 are shown and then every tenth) and the y axis shows the ranking score. The ranking is done on the scores from the compendium. The values for the same run and unseen test sets are shown on separate lines. The algorithm used to evolve the attacker is shown by the marker and the color. The attacker in the box with the solid line is the top ranked solution from the Combined Score ranking schemes. The solution in the dashed box is the top ranked solution from the Minimum Fitness ranking scheme

A simple language e.g. allows a strategy to select one or more network servers to disable for some duration. Defenders choose one of three different network routing protocols: shortest path, flooding and a peer-to-peer ring overlay to try to maintain their performance. A more complex language allows a varying number of steps over which the attack is modulated in duration, strength and targets. It can even support an attack learning a parameterized condition that controls how it adapts during an attack, i.e. "online", based on feedback it collects from the network on its impact. Defenders have simple languages related to parameterizations of P2P networks that influence the degree to which resilience is traded off with service costs. A more complex language allows the P2P network to adapt during an attack based on local or global observations of network conditions.

An example of attackers from ESTABLO on a mobile resource allocation defense used in *DIFFRACTED* [115] is shown in Fig. 11. The mobile asset placement defense challenge is to optimize the strategic placement of assets in the network. While under the threat of node-level DDOS attack, the defense must enable a set of tasks. It does this by fielding feasible paths between the nodes that host the assets which support the tasks. A mobile asset is, for example, mobile personnel or a software application that can be served by any number of nodes. A task is, for example, the connection that allows personnel to use a software application.

4.3 *AVAIL*: availability attacks on segmented networks

Attackers often introduce malware into networks. Once an attacker has compromised a device on a network, they spread to connected devices, akin to contagion. *AVAIL* considers *network segmentation*, a widely recommended defensive strategy, deployed against the threat of serial network security attacks that delay the mission of the network's operator [56] in the context of malware spread.

Network segmentation divides the network topologically into *enclaves* that serve as isolation units to deter inter-enclave contagion. How much network segmentation is helpful is a tradeoff. On the one hand, a more segmented network provides

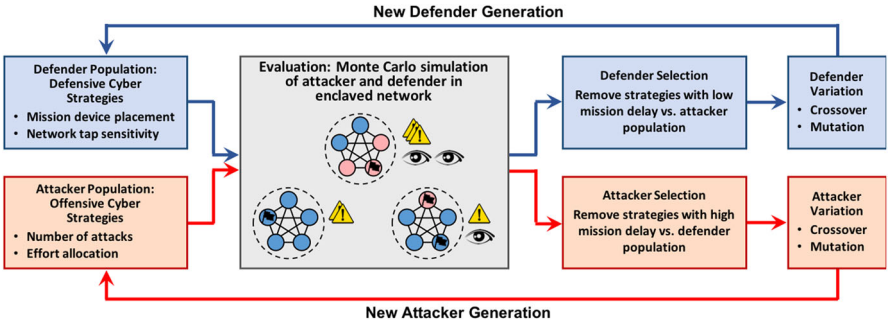


Fig. 12 Flow of the competitive coevolutionary algorithm used to evaluate defensive network enclave configurations (blue boxes) and contagion attacks (red boxes) in *AVAIL* [56]. A Monte Carlo simulation of device compromise contagion in a segemented network is used to assign the fitness of the adversaries. *AVAIL* uses maximum expected utility as a fitness score based on the mission delay

lower mission efficiency because of increased overhead in inter-enclave communication. On the other hand, smaller enclaves contain compromise by limiting the spread rate, and their cleansing incurs fewer mission delays. Adding complexity, given some segmentation, a network operator can monitor threats and utilize cleansing policies to detect and dislodge attackers, with the caveat of cost versus efficacy.

AVAIL assumes an enterprise network in carrying out a *mission*, and that an adversary employs *availability attacks* against the network to disrupt it. Specifically, the attacker starts by using an exploit to compromise a vulnerable device on the network. This inflicts a mission delay when a mission critical device is infected. The attacker moves laterally to compromise additional devices to further delay the mission. Figure 12 shows *AVAIL*.

AVAIL employs a Monte Carlo simulation model as its engagement environment. Malware contagion of a specific spread rate is assumed. The defender decides placement of mission devices and tap sensitivities in the pre-determined enclave segmentation. The attacker decides the strength, duration and number of attacks in an attack plan targeting all enclaves. For a network with a set of four enclave topologies, the framework is able to generate strong availability attack patterns that were not identified a priori. It also identifies effective configurations that minimize mission delay when facing these attacks.

4.4 *DARK-HORSE*: internal reconnaissance in software defined networks

Once an adversary has compromised a network endpoint, they can perform network reconnaissance [127]. After reconnaissance provides a view of the network and an understanding of where vulnerable nodes are located, attackers are able to execute a plan of attack. One way to protect against reconnaissance is by obfuscating the network to delay the attacker. This approach is well suited to software defined networks (SDN) such as those deployed in cloud server settings because it requires programmability that they support [68]. The SDN controller knows which

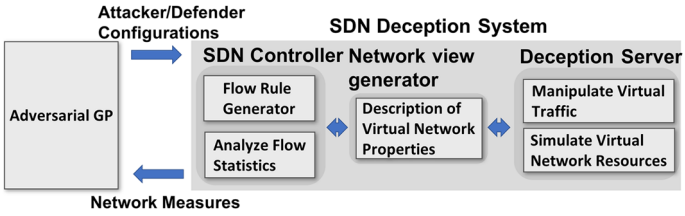


Fig. 13 *DARK-HORSE* fitness evaluation (left side) has a defensive SDN system based on the POX SDN Controller that creates different network views and manipulates traffic, and an attacker that performs NMAP scans in a `mininet` simulation. The scan results are passed into a fitness function that assigns fitness values for the adversaries' configurations. The search in the competitive coevolutionary algorithm uses maximum expected utility of the detection time to assign a fitness value

machines are actually on the network and can superficially alter (without function loss) the network view of each node, as well as place decoys (honeypots) on the network to mislead, trap and slow down reconnaissance. *DARK-HORSE* is shown in Fig. 13.

One such multi-component deceptive defense system [2] foils scanning by generating "camouflaged" versions of the actual network and providing them to hosts when they renew their DHCP leases. We use this deception system and `mininet` [134] within the framework as an engagement environment. This allows us to explore the dynamics between attacker and defender on a network where the deception and reconnaissance strategies can be adapted in response to each other [104].

A deception strategy is executed through a modified `POX` SDN controller. A reconnaissance strategy is executed by an NMAP scan [82]. The attacker strategy includes choices of: which IP addresses to scan, how many IP addresses to scan, which subnets to scan, the percent of the subnets to scan, the scanning speed, and the type of scan. The defender strategy includes choices of: the number of subnets to set up, the number of honeypots, the distribution of the real hosts throughout the subnets, and the number of real hosts that exist on the network. Fitness is comprised of four components: how fast the defender detects that there is a scan taking place, the total time it takes to run the scan, the number of times that the defender detects the scanner, and the number of real hosts that the scanner discovers. Through experimentation and analysis, the framework is able to discover configurations that the defender can use to significantly increase its ability to detect scans. Similarly, there are specific reconnaissance configurations that have a better chance of being undetected.

4.5 *RIVALS* summary

We summarize *RIVALS* use cases with Table 2 and its architecture with Fig. 10. The *RIVALS* framework is one example of Adversarial Genetic Programming for Cyber Security, where the domain of network security is one of many possibilities in *cyber security*. Possible future steps are elaborated in the next section.

Table 2 How *RIVALS* components are configured to express specific use cases

Component	Use case		
	Robustness versus denial	Deception versus reconnaissance	Isolation versus contagion
NAME	*DIFFRACTED*	*DARK-HORSE*	*AVAIL*
Threat environment	P2P versus DDOS	Pox SDN versus Nmap	Enclave versus Malware
Evaluation	ALFA sim	Mininet	High level Sim
Objective	Mission disruption	Detection speed	Mission delay
Behavior: attacker	Node/Edge impairment	Scanning parameters	Strength and duration
Behavior: defender	Network settings	Honeypots	Network taps and device placement
Adaptivity	Yes	No	No
Text description	Section 4.2	Section 4.4	Section 4.3

5 Taking stock and moving forward

Summary Considering the paper from bottom to top, we described three use cases: *DIFFRACTED*, *AVAIL*, and *DARK-HORSE*. For 3 specific network security contexts, they investigate unique threat environments, objectives and behaviors for their adversary species, and varying capacities to adapt during an attack on a mission. They draw upon the *RIVALS* framework in which adversarial arms races are the activity of interest. *RIVALS* is an example of *Adversarial Genetic Programming for Cyber Security* and is motivated by a call to arms to work on automated defensive reaction to waves of attacks. *RIVALS* pushes towards this goal by considering how *both attack and defense* adapt to each other. While we identified a breadth of applications to cyber security in the set of EAs we surveyed, we found just a modest number of extant *adversarial GP* that represented approaches for behavior investigation and two population dynamics, the latter combining competitive coevolution with GP. The body of prior work relevant to cyber security from AI and games and software engineering more generally informs the reader as to how competitive coevolutionary algorithms and EAs can be used as building blocks in different adversarial contexts. Finally, at the top, we argue that *Adversarial Genetic Programming for Cyber Security* is compellingly worthy of future community attention because of its potential to solve a critical and growing set of challenges in cyber security and because of how well GP and adversarial evolution match the technical nature of the challenges.

Future directions To end, we consider what future research questions and directions of study, stemming from the domain of cyber security, are prompted by *Adversarial GP*.

We forecast explorations driven by the wide array of cyber security scenarios will arise. There is a plethora of attack surfaces. We have mentioned some in Sect. 3.3, however, there are others such as networked cyber physical systems, ransomware, gadgets and malware. For example, how can GP help examine the adversarial cyber security of a power grid or self-driving cars? How could ransomware's next mutation be

predictable? Can GP leverage code repositories and gadgets to discover new malware? Social engineering is currently an Achilles heel for security measures because it preys on human nature and once privileged access has been obtained, it is much harder to discern an attacker. Insider threats similarly disguise attacks. How can GP inform insider threats and social engineering?

One challenge for the GP community is to become adequately conversant in cyber security topics (though no member needs to be conversant in all of them). Collaborations with cyber security researchers seem advisable, offering additive and synergistic benefits. It will be important to identify the most appropriate level of abstractions from which to design GP languages or function and terminal sets. It will be imperative to develop metrics of success and paths to technology transfer and solution deployment. At a practical level, undoubtedly new modeling and simulation modeling platforms will be needed. They will need to be general purpose (somewhat like *RIVALS*) and specific. Undoubtedly some will need to be scaled due to the computational cost of executing a mission.

We forecast the opportunity for extensions and innovations in GP and coevolutionary algorithms for cyber security. Different patterns and rates of evolution will be observed across different security scenarios. The opening to develop new algorithmic models of these phenomena is exciting. GP evolves behavior but many different expressions of behavior remain unexplored. Developing the new behavioral evolutionary capabilities that will be needed is both challenging and motivating.

We also forecast that research into *Adversarial Genetic Programming for Cyber Security* will push bridge building to other research areas. One impetus will be the quest to describe at a finer granularity what is happening during cyber evolution. For this, it may be advisable to consider evolution through the lens of individuals, such as how Artificial Life (ALife) [83] models individuals. What cyber security scenario should be studied with ALife? How can agents in a cyber security ALife system be represented by evolvable GP behaviors? What if a GP defender "died" after an attack and attackers starve as they fail to penetrate defenses? How would dynamics at this scale offer different insights? What would a simulation of the DDOS attack ecosystem reveal about the progression of botnet sizes if intra-species competition was examined? What can ALife studies into ecosystem dynamics around intra-species competition and cooperation reveal?

Arguably, adversarial GP is a form of agent-based modeling (ABM [90]) where agents are GP executable structures. Some ABM systems model more complex domains while the agents have simpler strategy spaces. Are there ABM investigations such as those of financial markets, crowd control, infectious diseases or traffic simulation that suggest innovative transfer to Adversarial GP and cyber security? Both ALife and ABM have examples that consider the spatial dimension of a domain. Competitive coevolutionary algorithms have spatial models. Interesting new models lie at these intersections.

Another area with potentially valuable interaction is Machine Learning (ML). Statistical machine learning relies on retrospective training data to learn a model capable of generalizing to new unseen data drawn from the same distribution as the retrospective data. In contrast, *Adversarial Genetic Programming for Cyber Security* is not data driven and this conveys it a niche. However, what can be transferred from the

studies that have started to bridge supervised learning and test-based co-optimization [107]? Adversarial classification [36] as a game between classifier and adversary also appears in ML. As well, Generative Adversarial Networks (GANs) [52] derive generative models from samples of a distribution using adversarial neural network training. Coevolution has been applied to improve the robustness and scalability of these systems [116]. Competitive coevolution has also been combined with multi-layer perceptron classifiers with floating point representation [29].

Studies of adversarial coevolution in cyber security based on data that classify attacks using machine learning methods exist, e.g. see a pro-active defense for evolving cyber threats and a predictive defense against evolving adversaries in [31–33]. In adversarial Machine Learning the attacker manipulates data in order to defeat the Machine Learning model [20,59,80]. Can coevolution be used to anticipate continual adversarial evasion and guide model hardening?

Game theory considers equilibria and paths to equilibria. What is an equilibrium in cyber security? It could be the complete wipeout of one side of the adversarial equation. Or, it could be the point where an attacker chooses not to target a defense because it is less expensive to go elsewhere. It could be a Nash equilibrium where neither attacker or defender has a better tactical position to go to unless the other simultaneously changes also. One direction should investigate how these game theoretic notions can inform the highly empirical, algorithmic systems of Adversarial GP.

We believe the future directions are not enumerable because the likelihood of new adaptations by adversaries will never be zero. To date, defenses expose far more attack surface than they can protect and attackers only need to find one crack to penetrate. To this end, the emerging importance of *Adversarial Genetic Programming for Cyber Security* is not likely to fade. We trust this justifies our call to arms.

Acknowledgements This was supported by the CSAIL CyberSecurity Initiative. This material is based upon work supported by DARPA. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements. Either expressed or implied of Applied Communication Services, or the US Government. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 799078.

Appendix A

A.1 Firefly squid bioluminescence defense

When firefly squid are seen from below by a predator, the bioluminescence helps to match the squid's brightness and color to the sea surface above.

A.2 A coevolutionary perspective of the U.S. automotive industry

Talay, Calantone, and Voorhees presented an study that explicitly terms competitive interactions between firms "Red Queen competition", in which gains from innovations are relative and impermanent [133].

A.3 Advanced persistent threats and ransomware

Some DOS attacks includes Advanced Persistent Threats (APT) and Ransomware. The first ones have multiple stages starting at external reconnaissance then moving to intrusion (e.g. social engineering or use of zero day exploits), laterally moving malware, command and control direction to data exfiltration and, finally, self-erasure. Ransomware, which largely preys upon unpatched systems and which exploits anonymous payment channels enabled by Bitcoin, has also recently become more frequent.

A.4 Cyber security attack categorization

Examples of attacks, classifications and taxonomies can be found at https://cwe.mitre.org/index.html. One categorization is: (A) Advanced Persistent Threats, "lurking" threats from resourceful persevering adversaries. (B) Denial of Service Attack, defense resource limitation and exposure, means of penetrating systems. (C) Identity theft, e.g. impersonating users. Attacks are also characterized by their stages on a timeline. Another characterization is based on the attacker identity, from individuals to organized criminals and nation states, and what resources they access, see [21] for details.

References

1. H.A. Abbass, The art of red teaming, in *Computational Red Teaming*, ed. by H.A. Abbass (Springer, Berlin, 2015), pp. 1–45
2. S. Achleitner, T. Laporta, P. McDaniel, Cyber deception: virtual networks to defend insider reconnaissance, in *Proceedings of the 2016 International Workshop on Managing Insider Security Threats* (2016), pp. 57–68
3. K. Adamopoulos, M. Harman, R.M. Hierons, How to overcome the equivalent mutant problem and achieve tailored selective mutation using co-evolution, in *Genetic and Evolutionary Computation—GECCO 2004* (Springer 2004), pp. 1338–1349
4. Akamai, Akamai's State of the Internet/Security Report—Q1 2017 report. Technical report, Akamai Technologies, Inc. (2017). https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q1-2017-state-of-the-internet-security-report.pdf
5. Akamai, Akamai's State of the Internet/Security Report—Q3 2017 report. Technical report, Akamai Technologies, Inc. (2017). https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q3-2017-state-of-the-internet-security-report.pdf
6. Akamai Technologies, State of the internet quarterly security reports (2017). https://www.akamai.com/us/en/about/our-thinking/state-of-the-internet-report/global-state-of-the-internet-security-ddos-attack-reports.jsp
7. J.C. Alex, Behavior trees for next-gen game AI, in *Game Developers Conference, Lyon, France* (2007), pp. 3–4
8. S. Anand, E.K. Burke, T.Y. Chen, J. Clark, M.B. Cohen, W. Grieskamp, M. Harman, M.J. Harrold, P. McMinn et al., An orchestrated survey of methodologies for automated software test case generation. J. Syst. Softw. **86**(8), 1978–2001 (2013)
9. P.J. Angeline, J.B. Pollack, Competitive environments evolve better solutions for complex tasks, in *Proceedings of the Fifth International Conference (GA93), Genetic Algorithms* (1993), pp. 264–270
10. M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J.A. Halderman, L. Invernizzi, M. Kallitsis, et al., Understanding the mirai botnet, in *26th USENIX Security Symposium (USENIX Security 17)* (2017), pp. 1093–1110
11. L.M. Antonio, C.A.C. Coello, Coevolutionary multi-objective evolutionary algorithms: a survey of the state-of-the-art. IEEE Trans. Evolut. Comput. (2018). https://doi.org/10.1109/TEVC.2017.2767023

12. C. Applegate, C. Elsaesser, J. Sanborn, An architecture for adversarial planning. IEEE Trans. Syst. Man Cybern. **20**(1), 186–194 (1990)
13. A. Arcuri, X. Yao, Coevolving programs and unit tests from their specification, in *Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering*, ACM (2007), pp. 397–400
14. A. Arcuri, X. Yao, Co-evolutionary automatic programming for software development. Inf. Sci. **259**, 412–432 (2014)
15. R. Axelrod, *The Evolution of Cooperation*, vol. 10 (Basic Books, New York, 1984)
16. R. Axelrod et al., The evolution of strategies in the iterated prisoner's Dilemma, in *The Dynamics of Norms*, ed. by C. Bicchieri, R. Jeffrey, B. Skyrms (Cambridge University Press, Cambridge, 1987), pp. 1–16
17. A.G. Bari, A. Gaspar, R.P. Wiegand, A. Bucci, Selection methods to relax strict acceptance condition in test-based coevolution, in *2018 IEEE Congress on Evolutionary Computation (CEC)* (IEEE, 2018), pp. 1–8
18. E. Barr, M. Harman, P. McMinn, M. Shahbaz, S.I. Yoo, The oracle problem in software testing: a survey. IEEE Trans. Softw. Eng. **41**, 507–525 (2015)
19. D. Beard, Enhancing Automated Red Teaming with Monte Carlo Tree Search (2011)
20. B. Biggio, F. Roli, Wild patterns: ten years after the rise of adversarial machine learning. ArXiv preprint arXiv:1712.03141 (2017)
21. D. Bodeau, R. Graubart, *Characterizing Effects on the Cyber Adversary: A Vocabulary for Analysis and Assessment* (The MITRE Corporation, Bedford, MA, 2013)
22. J.C. Bongard, H. Lipson, Nonlinear system identification using coevolution of models and tests. IEEE Trans. Evol. Comput. **9**(4), 361–384 (2005)
23. R. Boyd, Mistakes allow evolutionary stability in the repeated prisoner's Dilemma game. J. Theor. Biol. **136**(1), 47–56 (1989)
24. Brian Krebs, Akamai on the Record KrebsOnSecurity Attack. https://krebsonsecurity.com/2016/11/akamai-on-the-record-krebsonsecurity-attack/ (2016). Accessed October 10, 2018
25. R. Bronfman-Nadas, N. Zincir-Heywood, J.T. Jacobs, An artificial arms race: could it improve mobile malware detectors? in *2018 Network Traffic Measurement and Analysis Conference (TMA)* (IEEE, 2018), pp. 1–8
26. A. Bucci, Emergent geometric organization and informative dimensions in coevolutionary algorithms, Ph.D. thesis, Brandeis University (2007)
27. A.B. Cardona, J. Togelius, M.J. Nelson, Competitive coevolution in MS, Pac-Man, in *2013 IEEE Congress on Evolutionary Computation* (2013), pp. 1403–1410
28. J. Cartlidge, S. Bullock, Combating coevolutionary disengagement by reducing parasite virulence. Evol. Comput. **12**(2), 193–222 (2004)
29. M. Castellani, Competitive co-evolution of multi-layer perceptron classifiers. Soft. Comput. **22**(10), 3417–3432 (2018)
30. K. Chellapilla, D.B. Fogel, Evolution, neural networks, games, and intelligence. Proc. IEEE **87**(9), 1471–1496 (1999)
31. R. Colbaugh, K. Glass, Proactive defense for evolving cyber threats, in *2011 IEEE International Conference on Intelligence and Security Informatics (ISI)* (IEEE, 2011), pp. 125–130
32. R. Colbaugh, K. Glass, Predictive defense against evolving adversaries, in *2012 IEEE International Conference on Intelligence and Security Informatics (ISI)* (IEEE, 2012), pp. 18–23
33. R. Colbaugh, K. Glass, Moving target defense for adaptive adversaries, in *2013 IEEE International Conference on Intelligence and Security Informatics (ISI)* (IEEE, 2013), pp. 50–55
34. Crandall, J.R., Ensafi, R., Forrest, S., Ladau, J., Shebaro, B.: The ecology of malware, in *Proceedings of the 2008 workshop on New Security Paradigms* (ACM, 2009), pp. 99–106
35. R. Crawford-Marks, L. Spector, J. Klein, Virtual witches and warlocks: a quidditch simulator and quidditch-playing teams coevolved via genetic programming, in *Late-Breaking Papers of GECCO-2004, the Genetic and Evolutionary Computation Conference. Published by the International Society for Genetic and Evolutionary Computation* (2004)
36. N. Dalvi, P. Domingos, S. Sanghai, D. Verma, et al. Adversarial classification, in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM, 2004), pp. 99–108
37. DARPA, The World's first all-machine hacking tournament. http://archive.darpa.mil/cybergrandchallenge/ (2016). Accessed October 10, 2018

38. M.A. DeLoura, *Game Programming Gems*, vol. 2 (Cengage Learning, Boston, 2001)
39. R. DeMilli et al., Constraint-based automatic test data generation. IEEE Trans. Softw. Eng. **17**(9), 900–910 (1991)
40. P.R. Ehrlich, P.H. Raven, Butterflies and plants: a study in coevolution. Evolution **18**(4), 586–608 (1964)
41. D. Evans, A. Nguyen-Tuong, J. Knight, Effectiveness of moving target defenses, in *Moving Target Defense*, ed. by S. Jajodia, A. Ghosh, V. Swarup, C. Wang, X. Wang (Springer, Berlin, 2011), pp. 29–48
42. S.G. Ficici, Solution concepts in coevolutionary algorithms. Ph.D. thesis, Brandeis University (2004)
43. Flickr, Fireflies brighter (2014). https://www.flickr.com/photos/antoniseb/14325795079/in/gallery-flickr-72157645552049011flickr. Picture taken by Jay Cross—License: CC-BY-SA-2.0
44. D. Fogel, *Blondie24: Playing at the Edge of Artificial Intelligence* (Elsevier, Amsterdam, 2001)
45. R. Ford, M. Bush, A. Bulatov, Predation and the cost of replication: New approaches to malware prevention? Comput. Secur. **25**(4), 257–264 (2006)
46. S. Forrest, S.A. Hofmeyr, A. Somayaji, T.A. Longstaff, A sense of self for unix processes, in *Proceedings 1996 IEEE Symposium on Security and Privacy* (IEEE, 1996), pp. 120–128
47. O.L. Fraser, N. Zincir-Heywood, M. Heywood, J.T. Jacobs, Return-oriented programme evolution with roper: a proof of concept, in *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (ACM, 2017), pp. 1447–1454
48. D. Garcia, A.E. Lugo, E. Hemberg, U.M. O'Reilly, Investigating coevolutionary archive based genetic algorithms on cyber defense networks, in *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '17* (ACM, New York, NY, USA, 2017), pp. 1455–1462
49. S. Garcia, M. Grill, J. Stiborek, A. Zunino, An empirical comparison of botnet detection methods. Comput. Secur. **45**, 100–123 (2014)
50. P. Godefroid, N. Klarlund, K. Sen, Dart: directed automated random testing, in *ACM Sigplan Notices*, vol. 40, pp. 213–223. ACM (2005)
51. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st edn. (Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1989)
52. I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples. ArXiv preprint arXiv:1412.6572 (2014)
53. A. Gupta, P. Kuppili, A. Akella, P. Barford, An empirical study of malware evolution, in *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International* (IEEE, 2009), pp. 1–10
54. K.V. Hanford, Automatic generation of test cases. IBM Syst. J. **9**(4), 242–257 (1970)
55. R. Harper, Evolving robocode tanks for evo robocode. Genet. Progr. Evol. Mach. **15**(4), 403–431 (2014)
56. E. Hemberg, J.R. Zipkin, R.W. Skowyra, N. Wagner, U.M. O'Reilly, Adversarial co-evolution of attack and defense in a segmented computer network environment, in *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (ACM, 2018), pp. 1648–1655
57. P. Hingston, M. Preuss, Red teaming with coevolution, in *2011 IEEE Congress on Evolutionary Computation (CEC)* (2011), pp. 1155–1163. https://doi.org/10.1109/CEC.2011.5949747
58. G.S. Hornby, B. Mirtich, Diffuse versus true coevolution in a physics-based world, in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation, Vol. 2* (Morgan Kaufmann Publishers Inc., 1999), pp. 1305–1312
59. L. Huang, A.D. Joseph, B. Nelson, B.I. Rubinstein, J. Tygar, Adversarial machine learning, in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence* (ACM, 2011), pp. 43–58
60. D. Iliopoulos, C. Adami, P. Szor, Darwin inside the machines: malware evolution and the consequences for computer security. ArXiv preprint arXiv:1111.2503 (2011)
61. H.G. Kayacık, Can the best defense be a good offense? Evolving (MIMICRY) attacks for detector vulnerability testing under a 'black-box' assumption. Ph.D. thesis, Dalhousie University, Halifax (2009)
62. H.G. Kayacık, A.N. Zincir-Heywood, M.I. Heywood, Can a good offense be a good defense? Vulnerability testing of anomaly detectors through an artificial arms race. Appl. Soft Comput. **11**(7), 4366–4383 (2011)
63. D. Keaveney, C. O'Riordan, Evolving coordination for real-time strategy games. IEEE Trans. Comput. Intell. AI Games **3**(2), 155–167 (2011)

64. R. Kewley, M. Embrechts, Computational military tactical planning system. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **32**(2), 161–171 (2002). https://doi.org/10.1109/TSMCC.2002.801352
65. S. Khanchi, A. Vahdat, M.I. Heywood, A.N. Zincir-Heywood, On botnet detection with genetic programming under streaming data label budgets and class imbalance. Swarm Evolut. Comput. **39**, 123–140 (2018)
66. H.S. Kim, S.B. Cho, An efficient genetic algorithm with less fitness evaluation by clustering, in *Proceedings of the 2001 Congress on Evolutionary Computation*, pp. 887–894 (2001)
67. K.E. Kinnear, W.B. Langdon, L. Spector, P.J. Angeline, U.M. O'Reilly, *Advances in Genetic Programming*, vol. 3 (MIT Press, Cambridge, 1999)
68. K. Kirkpatrick, Software-defined networking. Commun. ACM **56**(9), 16–19 (2013)
69. J.R. Koza, *Genetic Programming II, Automatic Discovery of Reusable Subprograms* (MIT Press, Cambridge, MA, 1992)
70. K. Krawiec, M. Heywood, Solving complex problems with coevolutionary algorithms, in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion* (ACM, 2016), pp. 687–713
71. K. Krawiec, M.G. Szubert, Learning n-tuple networks for othello by coevolutionary gradient search, in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11* (ACM, 2011) pp. 355–362
72. R. Lara-Cabrera, C. Cotta, A.J. Fernández-Leiva, A review of computational intelligence in RTS games, in *2013 IEEE Symposium on Foundations of Computational Intelligence (FOCI)* (2013), pp. 114–121
73. P. LaRoche, N. Zincir-Heywood, M.I. Heywood, Evolving TCP/IP packets: a case study of port scans, in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications* (IEEE, 2009), pp. 1–8
74. C. Le Goues, A. Nguyen-Tuong, H. Chen, J.W. Davidson, S. Forrest, J.D. Hiser, J.C. Knight, M. Van Gundy, Moving target defenses in the helix self-regenerative architecture, in *Moving Target Defense II*, ed. by S. Jajodia, A.K. Ghosh, V.S. Subrahmanian, V. Swarup, C. Wang, X.S. Wang (Springer, Berlin, 2013), pp. 117–149
75. M. Lemczyk, M.I. Heywood, Training binary GP classifiers efficiently: a pareto-coevolutionary approach, in *European Conference on Genetic Programming* (Springer, 2007), pp. 229–240
76. C.U. Lim, R. Baumgarten, S. Colton, Evolving behaviour trees for the commercial game DEFCON, in *European Conference on the Applications of Evolutionary Computation* (Springer, 2010), pp. 100–110
77. R. Lippmann, J.W. Haines, D.J. Fried, J. Korba, K. Das, The 1999 Darpa off-line intrusion detection evaluation. Comput. Netw. **34**(4), 579–595 (2000)
78. P. Liskowski, K. Krawiec, Non-negative matrix factorization for unsupervised derivation of search objectives in genetic programming, in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference* (ACM, 2016), pp. 749–756
79. P. Liskowski, K. Krawiec, Online discovery of search objectives for test-based problems. Evol. Comput. **25**(3), 375–406 (2017)
80. D. Lowd, C. Meek, Adversarial learning, in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* (ACM, 2005), pp. 641–647
81. S. Luke et al., Genetic programming produced competitive soccer softbot teams for robocup97. Genet. Program. **1998**, 214–222 (1998)
82. G. Lyon, Nmap network scanner. https://nmap.org/ (2018). Accessed July 6, 2018
83. C.M. Macal, M.J. North, Tutorial on agent-based modelling and simulation. J. Simul. **4**(3), 151–162 (2010)
84. M.V. Mahoney, P.K. Chan, An analysis of the 1999 Darpa/Lincoln laboratory evaluation data for network anomaly detection, in *Recent Advances in Intrusion Detection*, ed. by G. Vigna, C. Kruegel, E. Jonsson (Springer, Berlin, 2003), pp. 220–237
85. W. Mazurczyk, S. Drobniak, S. Moore, Towards a systematic view on cybersecurity ecology. ArXiv preprint arXiv:1505.04207 (2015)
86. M.L. McDonald, S.C. Upton, Investigating the dynamics of competition: coevolving red and blue simulation parameters, in *Proceedings of the 37th Conference on Winter Simulation*, pp. 1008–1012 (2005)
87. N. Milano, J.T. Carvalho, S. Nolfi, Moderate environmental variation across generations promotes the evolution of robust solutions. Artif. Life **24**(4), 277–295 (2019)

88. C. Miles, J. Quiroz, R. Leigh, S.J. Louis, Co-evolving influence map tree based strategy game players, in *IEEE Symposium on Computational Intelligence and Games, 2007. CIG 2007* (IEEE, 2007), pp. 88–95
89. M. Mitchell, Coevolutionary learning with spatially distributed populations, in *Computational Intelligence: Principles and Practice*, ed. by G.Y. Yen, D.B. Fogel (Springer, Berlin, 2006)
90. N. Moran, J. Pollack, Effects of cooperative and competitive coevolution on complexity in a linguistic prediction game, in *Artificial Life Conference Proceedings*, Vol. 14 (MIT Press, Cambridge, 2017), pp. 298–205
91. D.J. Musliner, S.E. Friedman, J.M. Rye, T. Marble, Meta-control for adaptive cybersecurity in fuzzbuster, in *2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)* (IEEE, 2013), pp. 219–226
92. D.J. Musliner, S.E. Friedman, J.M. Rye: Automated fault analysis and filter generation for adaptive cybersecurity, in *Proceedings of the 6th International Conference on Adaptive and Self-Adaptive Systems and Applications* (2014)
93. A.B. Nettles, The president has no clothes: the case for broader application of red teaming within homeland security. Technical report, DTIC Document (2010)
94. M. Nicolau, D. Perez-Liebana, M. O'Neill, A. Brabazon, Evolutionary behavior tree approaches for navigating platform games. IEEE Trans. Comput. Intell. AI Games **9**(3), 227–238 (2017)
95. M. Nogueira-Collazo, C.C. Porras, A.J. Fernández-Leiva, Competitive algorithms for coevolving both game content and AI. A case study: planet wars. IEEE Trans. Comput. Intell. AI Games **8**(4), 325–337 (2016)
96. H. Okhravi, T. Hobson, D. Bigelow, W. Streilein, Finding focus in the blur of moving-target techniques. Secur. Priv. IEEE **12**(2), 16–26 (2014). https://doi.org/10.1109/MSP.2013.137
97. A.A.L. de Oliveira, C.G. Camilo-Junior, A.M.R. Vincenzi, A coevolutionary algorithm to automatic test case selection and mutant in mutation testing, in *2013 IEEE Congress on Evolutionary Computation* (2013), pp. 829–836
98. B. Olsson, Co-evolutionary search in asymmetric spaces. Inf. Sci. **133**(3–4), 103–125 (2001)
99. M. O'Neill, C. Ryan, *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*, vol. 4 (Springer, Berlin, 2003)
100. U.M. O'Reilly, P.J. Angeline, Introduction to the special issue: Trends in evolutionary methods for program induction. Evol. Comput. **5**(2), v–ix (1997)
101. U.M. O'Reilly, H. Erik, An artificial coevolutionary framework for adversarial AI, in *Adversary-Aware Learning Techniques and Trends in Cybersecurity, AAAI Fall Symposium* (2018)
102. M. Ostaszewski, F. Seredynski, P. Bouvry, Coevolutionary-based mechanisms for network anomaly detection. J. Math. Modell. Algorithms **6**(3), 411–431 (2007)
103. D. Perez, M. Nicolau, M. O'Neill, A. Brabazon, Evolving behaviour trees for the Mario AI competition using grammatical evolution, in *European Conference on the Applications of Evolutionary Computation* (Springer, 2011), pp. 123–132
104. M. Pertierra, Investigating coevolutionary algorithms for expensive fitness evaluations in cybersecurity. Master's thesis, Massachusetts Institute of Technology (2018)
105. A. Petrlic, Circular economy: a coevolutionary perspective on diversity. uwf UmweltWirtschaftsForum **24**(2), 253–260 (2016)
106. J.B. Pollack, A.D. Blair, Co-evolution in the successful learning of backgammon strategy. Mach. Learn. **32**(3), 225–240 (1998)
107. E. Popovici, Bridging supervised learning and test-based co-optimization. J. Mach. Learn. Res. **18**(38), 1–39 (2017)
108. E. Popovici, A. Bucci, R.P. Wiegand, E.D. De Jong, *Coevolutionary Principles* (Springer, Berlin, 2012), pp. 987–1033
109. E. Popovici, A. Bucci, R.P. Wiegand, E.D. De Jong, Coevolutionary principles, in *Handbook of Natural Computing*, ed. by G. Rozenberg, T. Back, J.N. Kok (Springer, Berlin, 2012), pp. 987–1033
110. E. Popovici, E. Winston, A framework for co-optimization algorithm performance and its application to worst-case optimization. Theor. Comput. Sci. **567**, 46–73 (2015)
111. D. Prado Sanchez, Visualizing adversaries—transparent pooling approaches for decision support in cybersecurity. Master's thesis, Massachusetts Institute of Technology (2018)
112. C.D. Rosin, R.K. Belew, New methods for competitive coevolution. Evol. Comput. **5**(1), 1–29 (1997)

113. G. Rush, D.R. Tauritz, A.D. Kent, Coevolutionary agent-based network defense lightweight event system (CANDLES), in *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference* (ACM, 2015), pp. 859–866
114. R.D. Sagarin, T. Taylor, Natural security: how biological systems use information to adapt in an unpredictable world. Secur. Inform. **1**(1), 14 (2012)
115. D.P. Sanchez, M.A. Pertierra, E. Hemberg, U.M. O'Reilly, Competitive coevolutionary algorithm decision support, in *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (ACM, 2018), pp. 300–301
116. J. Schmiedlechner, A. Al-Dujaili, E. Hemberg, U.M. O'Reilly, Towards distributed coevolutionary gans. ArXiv preprint arXiv:1807.08194 (2018)
117. K. Scott, J. Davidson, Strata: a software dynamic translation infrastructure, in *IEEE Workshop on Binary Translation* (2001)
118. Scott Hilton: Dyn Analysis Summary of Friday October 21 Attack. https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/ (2016). Accessed October 10, 2018
119. T. Service, D. Tauritz, Increasing infrastructure resilience through competitive coevolution. New Math. Nat. Comput. **5**(02), 441–457 (2009)
120. K. Sims, Evolving 3d morphology and behavior by competition. Artif. Life **1**(4), 353–372 (1994)
121. M. Sipper, Evolved to Win. Lulu.com (2011)
122. M. Sipper, Y. Azaria, A. Hauptman, Y. Shichel, Designing an evolutionary strategizing machine for game playing and beyond. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **37**(4), 583–593 (2007)
123. R.E. Smith, Co-adaptive genetic algorithms: an example in othello strategy, in *Proceedings of the Florida Artificial Intelligence Research Symposium, 1994* (1994)
124. R.J. Smith, M.I. Heywood, Coevolving deep hierarchies of programs to solve complex tasks, in *Proceedings of the Genetic and Evolutionary Computation Conference* (ACM, 2017), pp. 1009–1016
125. Son of Boss: Son of boss—Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Son_of_Boss (2018). Accessed October 10, 2018
126. D. Song, M.I. Heywood, A.N. Zincir-Heywood, Training genetic programming on half a million patterns: an example from anomaly detection. IEEE Trans. Evol. Comput. **9**(3), 225–239 (2005)
127. A. Sood, R. Enbody, Targeted cyberattacks: a superset of advanced persistent threats. IEEE Secur. Priv. **11**(1), 54–61 (2013)
128. K.O. Stanley, B.D. Bryant, R. Miikkulainen, Real-time neuroevolution in the nero video game. IEEE Trans. Evol. Comput. **9**(6), 653–668 (2005)
129. G. Suarez-Tangil, E. Palomar, J.M. de Fuentes, J. Blasco, A. Ribagorda, Automatic rule generation based on genetic programming for event correlation, in *Computational Intelligence in Security for Information Systems*, ed. by A. Herrero Cosio, E. Corchado Rodriguez (Springer, Berlin, 2009), pp. 127–134
130. Symantec Security Response: Mirai: what you need to know about the botnet behind recent major DDoS attacks. https://www.symantec.com/connect/blogs/mirai-what-you-need-know-about-botnet-behind-recent-major-ddos-attacks (2016). Accessed October 10, 2018
131. M. Szubert, W. Jaskowski, K. Krawiec, Coevolutionary temporal difference learning for othello, in *2009 IEEE Symposium on Computational Intelligence and Games* (2009), pp. 104–111
132. M. Szubert, W. Jaśkowski, K. Krawiec, On scalability, generalization, and hybridization of coevolutionary learning: a case study for othello. IEEE Trans. Comput. Intell. AI Games **5**(3), 214–226 (2013)
133. M.B. Talay, R.J. Calantone, C.M. Voorhees, Coevolutionary dynamics of automotive competition: product innovation, change, and marketplace survival. J. Prod. Innov. Manag. **31**(1), 61–78 (2014)
134. M. Team, Mininet—realistic virtual SDN network emulator. http://mininet.org/ (2018). Accessed July 6, 2018
135. J. Togelius, P. Burrow, S.M. Lucas, Multi-population competitive co-evolution of car racing controllers, in *2007 IEEE Congress on Evolutionary Computation* (2007), pp. 4043–4050
136. W. Weimer, S. Forrest, C. Le Goues, T. Nguyen, Automatic program repair with evolutionary computation. Commun. ACM **53**(5), 109–116 (2010)
137. Wikimedia Commons: Cuttlefish changing color. https://upload.wikimedia.org/wikipedia/commons/thumb/1/1c/Cuttlefish_color.jpg/636px-Cuttlefish_color.jpg. Picture taken by Nick Hobgood—License: CC BY-SA 3.0

138. Wikimedia Commons: Misumena vatia with wasp (1998). https://en.wikipedia.org/wiki/File:Misumena.vatia.beute.wespe.1771.jpg#filelinks. Picture taken by Olaf Leillinger—License: CC-BY-SA-2.0/DE and GNU FDL
139. Wikimedia Commons: Viceroy butterfly (2005). https://commons.wikimedia.org/wiki/File:Viceroy_Butterfly.jpg. License: CC BY-SA 3.0. Subject to disclaimers
140. Wikimedia Commons: Monarch in may (2007). https://en.wikipedia.org/wiki/Monarch_butterfly#/media/File:Monarch_In_May.jpg. Created: 29 May 2007 By Kenneth Dwain Harrelson—License: CC BY-SA 3.0
141. Wikimedia Commons: Bioluminnescence in ocean organisms (2014). https://en.wikipedia.org/wiki/Bioluminescence#/media/File:Squid_Counterillumination.png. Picture taken by Chiswick Chap—License: CC BY-SA 4.0
142. J.L. Wilkerson, D. Tauritz, Coevolutionary automated software correction, in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, GECCO '10* (ACM, New York, NY, USA, 2010), pp. 1391–1392
143. G. Willard, Understanding the co-evolution of cyber defenses and attacks to achieve enhanced cybersecurity. Warfare **14**, 17–31 (2015)
144. N. Williams, M. Mitchell, Investigating the success of spatial coevolution, in *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation* (ACM, 2005), pp. 523–530
145. M.L. Winterrose, K.M. Carter, Strategic evolution of adversaries against temporal platform diversity active cyber defenses, in *Proceedings of the 2014 Symposium on Agent Directed Simulation*, p. 9. Society for Computer Simulation International (2014)
146. B.J. Wood, R. Duggan, et al. Red teaming of advanced information assurance concepts, in *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, Vol. 2, pp. 112–118 (IEEE, 2000)
147. D. Wright Jr., Financial alchemy: how tax shelter promoters use financial products to bedevil the IRS (and how the IRS helps them). Ariz. St. LJ **45**, 611 (2013)
148. W. Xu, Y. Qi, D. Evans, Automatically evading classifiers, in *Proceedings of the 2016 Network and Distributed Systems Symposium* (2016)
149. J. Yuen, Automated cyber red teaming. Technical report, DTIC Document (2015)
150. F. Zeng, J. Decraene, M. Low, S. Zhou, W. Cai, Evolving optimal and diversified military operational plans for computational red teaming. Syst. J. IEEE **6**(3), 499–509 (2012). https://doi.org/10.1109/JSYST.2012.2190693