IEEE Open Journal of the Computer Society

Received 9 May 2020; accepted 1 June 2020. Date of publication 3 July 2020;
date of current version 24 August 2020. The review of this paper was arranged by Associate Editor Deze Zeng.

Digital Object Identifier 10.1109/OJCS.2020.3006807

Vision Paper: Grand Challenges in Resilience: Autonomous System Resilience through Design and Runtime Measures

SAURABH BAGCHI [1], VANEET AGGARWAL [2], SOMALI CHATERJI [2], FRED DOUGLIS [2] (Fellow, IEEE), ALY EL GAMAL [3], JIAWEI HAN [2] (Fellow, IEEE), BRIAN J. HENZ [4], HENRY HOFFMANN [5], SUMAN JANA [6], MILIND KULKARNI [5] (Senior Member, IEEE), FELIX XIAOZHU LIN [6], KAREN MARAIS [6], PRATEEK MITTAL [2], SHAOSHUAI MOU [2], XIAOKANG QIU [2], AND GESUALDO SCUTARI [6]

[1]Purdue University, Perspecta Labs, West Lafayette, IN 47907 USA
[2]University of Illinois at Urbana-Champaign, Urbana, IL USA
[3]Army Research Lab, Aberdeen, MD USA
[4]University of Chicago, Chicago, IL USA
[5]Columbia University and New York, NY USA
[6]Princeton University, Princeton, NJ 08544 USA

CORRESPONDING AUTHOR. SAURABH BAGCHI (e-mail: sbagchi@purdue.edu).

This work was supported in part by the National Science Foundation under Grant Number CNS-1845192 and in part by Army Research Lab, under Contract W911NF-20-2-0026.

ABSTRACT In this article, we put forward the substantial challenges in cyber resilience in the domain of autonomous systems and outline foundational solutions to address these challenges. These solutions fall into two broad themes: *resilience-by-design* and *resilience-by-reaction*. We use several application drivers from autonomous systems to motivate the challenges in cyber resilience and to demonstrate the benefit of the solutions. We focus on some autonomous systems in the near horizon (autonomous ground and aerial vehicles) and also a little more distant (autonomous rescue and relief). For *resilience-by-design*, we focus on design methods in software that are needed for our cyber systems to be resilient. In contrast, for *resilience-by-reaction*, we discuss how to make systems resilient by responding, reconfiguring, or recovering at runtime when failures happen. We also discuss the notion of adaptive execution to improve resilience, execution transparently and adaptively among available execution platforms (mobile/embedded, edge, and cloud). For each of the two themes, we survey the current state, and the desired state and ways to get there. We conclude the paper by looking at the research challenges we will have to solve in the short and the mid-term to make the vision of resilient autonomous systems a reality. This article came out of discussions that started at the NSF-sponsored Grand Challenges in Resilience Workshop held at Purdue in 2019 with the co-authors contributing to going into the depth of the issues and then this article.

INDEX TERMS Resilience, autonomous systems, resilience by design, resilience by reaction.

I. INTRODUCTION

We lay out our vision for resilience in autonomous systems and our view of the short-term and mid-term research challenges to realize the vision. Our view of resilience has two primary aspects.

1) *Resilience by design*: This is the aspect that designs and develops cyber systems so that they are resilient to a large set of quantifiable *perturbations*.
2) *Resilience by reaction*: This is the aspect that works at runtime when perturbations are incident on the cyber system and imbues the systems with the ability to "bounce back" quickly after a failure triggered by a perturbation.

Note, of course, that these two aspects of resilience are intertwined: systems can be *designed* so that they incorporate resilience by *reaction*.

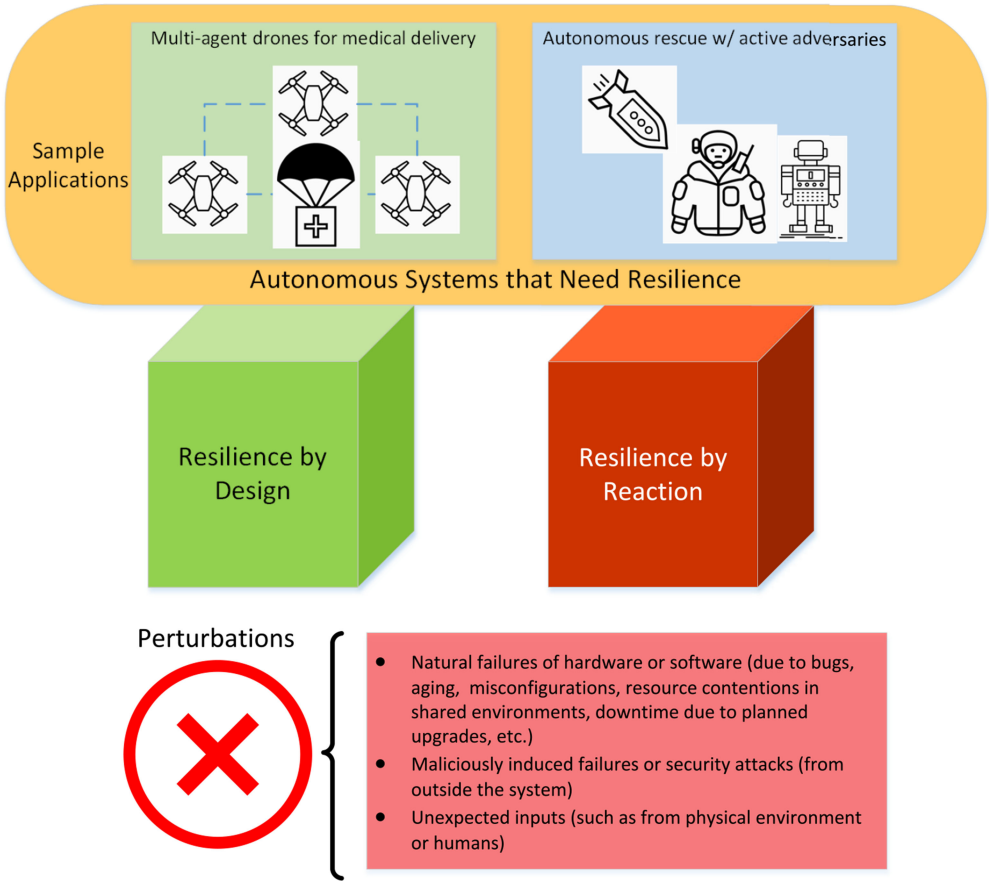


FIG. 1. High-level schematic of our vision for how resilience for autonomous systems can be structured. There are interplays between the two thrusts—resilience by design (offline techniques) and resilience by reaction (online techniques). We also show the target classes of perturbations that we consider in this article.

We also make specific the notion of **perturbations** that we want to deal with. These take three forms: (i) *natural failures of hardware or software* (due to bugs, aging, misconfigurations, resource contentions in shared environments, downtime due to planned upgrades, etc.), (ii) *maliciously induced failures* or security attacks (from outside the system), and (iii) *unexpected inputs* (our target class of autonomous systems will have to deal with the physical environment and will interface with humans, which will produce unpredictable data to which the system will need to adapt). The outcome of a perturbation that is not handled can be a hard failure (crash or hang) or a soft failure (*i.e.,* missing a deadline for a latency-sensitive application).

We will first introduce as application drivers two autonomous application scenarios where perturbations need to be handled. We will then discuss the resilience by design aspect and then the resilience by reaction aspect. For each, we will lay out the vision for the end state in 10 years. Then we will talk of the short-term and mid-term research challenges, side-by-side with the promising approaches being investigated today.

Fig. 1 shows a high-level schematic for the way we envision resilience in autonomous systems together with the universe of target perturbations.

II. AUTONOMOUS SYSTEMS AS APPLICATION DRIVERS

We first look at the state of resilience in two exemplar classes of autonomous systems. By autonomous we do not mean completely autonomous, rather those at varying levels of autonomy which involve some human involvement. We speculate at a desired degree of resilience against perturbations and use these as examples as broad motivation for the solution directions that we lay out in the rest of the article.

The first class of autonomous systems is drones being used to deliver essential medical supplies in hard-to-reach areas. We include in our purview interactions among multiple drones and among drones and the non-(Computer Science)expert humans responsible for their resilient operation. The second class of autonomous systems is rescue and relief by say an international humanitarian agency in the face of a natural or a man-made disaster. This involves ground and aerial sensors,

distributed inferencing from their inputs through processing at the edge as well as at the cloud.

II. DISTRIBUTED RESILIENCE IN MULTI-AGENT DRONES FOR MEDICAL DELIVERIES

Problem and Current State: The field of systems and control has recently been evolving from single monolithic system to teams of interconnected subsystems (or multi-agent networks). Because of the absence of centralized coordinator, algorithms for coordination in multi-agent networks (especially large-scale autonomous swarms) must be *distributed*, which achieve global objectives through only local coordination among nearby agents [19]. In order to guarantee all agents in the networks work as a cohesive whole, the concept of *consensus* naturally arises, which requires all agents in the network to reach an agreement regarding a certain quantity of interest [68], [92], [101]. A specific instantiation of this general idea of multi-agent systems is a **multiple drone system that is responsible for transporting essential supplies to a population affected by a natural disaster or medical supplies to a population where the surface transportation infrastructure is poor** [32], [113]. Some characteristics relevant to our discussion are that there are multiple cooperating agents involved, there is uncertainty in the physical as well as the cyber conditions (flying conditions may be variable and the network connectivity among multiple drones may be variable, as examples of the two kinds of uncertainty), and there is also human involvement, such as, to task the drones or to refill the supplies when the drone reaches a certain height above the ground at the home base.

Consensus is the basis of many distributed algorithms for computations [94], [146], [148], optimization [18], [24], [114], control [42], [48], [87] in multi-agent networks. The success of consensus-based algorithms relies on the assumption that all agents in the multi-agent swarm are cooperative, that is, each agent provides its own state value to its neighbor nodes and follows a common update protocol toward network objectives [101]. However, this particular autonomous system presents the salient challenge that it operates in an open and potentially adversarial environment, with exposure to a large and possibly unanticipated set of perturbations. On the one hand, distributed algorithms are inherently robust against individual node or link failures because of the absence of central controller; on the other hand, the strong dependence of distributed algorithms on local coordination raises a major concern of cyber attacks to the whole network through local attacks to one or more vulnerable agents [108]. Thus it is important to achieve *resilience* in order for autonomous multi-agent swarms to be used in critical applications like the current one.

There has been significant progress made in developing robust distributed algorithms by a combination of algorithmic and system-theoretic approaches in [17], [107], [141]. Further advancements will have to be made to handle hitherto unanticipated perturbations, to deal with the resource constraints of individual agents, to deal with time-varying characteristics such as link quality, and the possibility of multiple coordinated or uncoordinated perturbations. An entire new dimension arises due to the close interactions with humans—different patients may have different criticality requirements and these may change over short time periods, the level of cyber expertise of the human users both at the provider and at the consumer level will vary, and the time constants involved for some operations will be of human scale rather than cyber scale. Yet another dimension that needs significant research progress is scalability of these algorithms. While they have primarily been developed and evaluated under small world assumptions, they need to be re-designed or modified to operate at large scales, of the number of agents, of the distances (and latencies) involved and under the open-world assumption (new nodes can be added while the system is operational), etc.

II. COOPERATIVE AUTONOMOUS RESCUE WITH ACTIVE ADVERSARY

Operationalizing artificial intelligence (AI) for military applications often brings to mind either offensive or defensive operations such as breaching defenses or defending assets by intercepting projectiles [131]. Upon closer consideration, many of the challenges faced when automating these operations, such as dirty data, i.e. data with low signal to noise ratios, and sparse data, i.e. small training data sets [63], are also faced when performing civilian rescue operations. These resiliency challenges will be illustrated below with a small military rescue vignette and correlated with current Army research efforts and gaps including sensor fusion, autonomous coordinated swarms [112], and resource constrained computing. [67]

Consider for a moment a complex future operating environment [100] where military operations take place in the dynamic cyber and physical environments of large urban areas. AI algorithms often require training from large amounts of data for maneuver to include a priori knowledge of infrastructure including roads, buildings, and subterranean passageways. During military operations urban environments can change rapidly as buildings are destroyed and barriers to movement are erected, leaving little existing knowledge for aiding autonomous maneuver. In addition, instability of remaining structures is likely compromised so situational understanding (SU) must be gained on-the-fly, all while an adversary is actively employing anti-access and area denial (A2AD) capabilities. It is against this backdrop that future autonomous system will be called upon to locate, extract, and maneuver to safety either human teammates or other autonomous systems. Gaining SU in this environment will require fusing data from multiple sensing modalities over communications links that are unreliable at best, or denied at worst, requiring AI that achieves consensus on courses of action with incomplete information. Maneuvering through unstable structures requires

understanding of the physical world including solid body mechanics, material strengths and failure modes, and physical systems. The rescue of people or equipment is an operation that embodies these challenges for AI and autonomous systems to perform.

There are early research efforts to begin addressing many of these challenging areas including developing new sensor modalities, sensor fusion, and robotic perception. In order to complete the task illustrated above, to perform a military rescue operation, in the complex dynamic urban environment will require significant effort in many areas including the integration of these efforts into complex systems of systems. For instance, open challenges exist when sensor fusion occurs with asynchronous data, unavailable data, or active deception. Open challenges also exist for autonomous maneuver in dynamic environments, e.g., off-road driving, and when interaction with the physical environment is required such as moving obstacles or sliding a chair to remove an obstacle. Finally, communication and coordination of cooperating sensors and autonomous systems when traditional modes of communication such as radio frequency (RF) links are unavailable or denied is a long standing challenge that is compounded for this scenario by the uncertainty of the physical environment. Advancements that address these challenges will support civilian rescue operations such as in natural disasters without endangering additional human lives of rescuers and the compression of timelines for rescue operations.

III. RESILIENCE BY DESIGN

In this section, we discuss some design approaches to autonomous software systems that can make them resilient to the set of perturbations introduced earlier. For each we describe the problem context, some of the most promising techniques being researched today, and the desired end state and open challenges that we have to tackle to get there. This section includes discussion of attacks against building blocks of autonomous systems, resilient ML algorithms, immune-inspired resilient algorithms, program specification for resilience and keeping in mind human-in-the-loop.

We give in the side bar a distillation of the significant problems that we discuss in this section and the solution approaches. For the solution approaches, we categorize them as *Nascent* or *Developing*. The former means that there is little work so far but there is growing interest and some early promising results; the latter means there is a sizable body of work but it is growing and final answers are still in the future. By the nature of this article, none of the problems being discussed have fully mature solutions.

Problems:
1. Attacks against ML building blocks
2. Ease of generation and transferability of adversarial examples
3. Vulnerabilities of stochastic algorithms to corner cases
4. Synthesis of programs for resilience
5. Formal verification of correctness properties

Solution Approaches:
1. Whitebox testing and verification [N]
2. Optimization for resilience (in addition to accuracy) [D]
3. Evolutionary algorithms with resilience objectives [N]
4. Scalable verification techniques [D]
5. Verification with human-in-the-loop [N]

[N]: *Nascent;* [D]: *Developing*

A. ATTACKS AGAINST BUILDING BLOCKS OF AUTONOMOUS SYSTEMS

Problem Context: Deep learning algorithms have been shown over the past decade to be very successful in various image and speech processing applications (see e.g., [55]), and more recently for wireless communication systems (see e.g., [150] and [156]). These success stories suggest the applicability of deep neural networks in a ubiquitous fashion in the near future. However, for this to happen, such algorithms have to be designed while taking into consideration potential exposure to adversarial attacks, especially with their recently discovered vulnerabilities (see e.g., [20]). Furthermore, these adversarial attacks are effective even when the attacker can only perturb the test data, and even there only a small part of each data point, as in evasion attacks (see e.g., [15]). One of the early and most efficient attacks that have been proposed in the literature is the Fast Gradient Sign attack (FGS) [56]. FGS highlights the vulnerability of neural networks, as the adversarial perturbation takes place in the direction of the gradient of the loss function:

$$\widetilde{x} = x + \eta * sign\left(\nabla_x J(w, x, y)\right). \tag{1}$$

In (1), x refers to the original input sample and $sign(\nabla_x J(w, x, y))$ is the sign of the gradient of the employed cost function $J(w, x, y)$, which is a function of the input, x, the desired output, y, and the classifier weights, w. The parameter η is typically an l_p-bounded perturbation.

More stealthy attacks, that typically incur significantly higher computational cost, than the FGS attack have been proposed. Important examples are the evolutionary-algorithm-based attack introduced in [98], the feature-selection-based Jacobian Saliency Map attack introduced in [102], and the iterative Deep Fool and Carlini-Wagner (CW) attacks introduced in [91] and [35], respectively. In particular, iterative application of variants of the gradient sign concept were shown to present more effective attacks than the one step application of (1). This issue was closely analyzed in [74], where it was suggested that the one step FGS attacks leaks information about the true label. The classifier can then learn to perform very well on these adversarial examples by exploiting this

leaked information. We are inspired by such analysis as it relies on characterizing the flow of label information to assess the effectiveness of an adversarial attack. More generally, we believe that a principled approach to quantifying the value of knowledge at both the attacker and defender, as well as characterizing the flow of important information regarding the output of the machine learning models and input sample distribution, is missing in the literature, and crucially needed to reach a deeper understanding of the resilience of the foundational blocks of autonomous systems.

Current State: Recent work has also demonstrated the practical threat of adversarial examples in the context of real-world systems and constraints. For example, Papernot *et al.* [103] considered the scenario of a *black-box* threat model, in which the adversary does not have access to the details of the internal model structure or parameters, and also does not have access to the training data. Even in this challenging setting, the threat of adversarial examples persists. For example, it is possible for an adversary to locally train a model based on synthetically generated data, with associated labels obtained from interacting with the target model. The adversary can then use the locally trained model to generate adversarial examples, using standard techniques such as the FGS and CW attacks discussed above. This approach exploits the phenomenon of *transferability* of adversarial examples: with high likelihood, adversarial examples generated using the adversary's locally trained model successfully induce misclassification on the actual target model. Bhagoji *et al.* [14] explored an alternative attack approach for the black-box setting, which does not rely on transferability. Instead, the gradient term in (1) is replaced by an approximate estimate of the gradient which can be computed in a black-box manner by interacting with the target model.

Another thread of research has considered the question of designing physically-realizable adversarial examples for autonomous vehicles, whose effects persist in presence of real-world environmental constraints such as varying depth of perception, varying angle of perception, and varying brightness conditions. Standard attack techniques discussed previously produce adversarial examples that do not work well under such real-world conditions. Eykholt *et al.* [47] and Sitawarin *et al.* [125] modify the attack optimization problem to include such varying real-world conditions: their key insight is to incorporate a set of image transformations such as perspective transformations, image resizing, and brightness adjustment - as dataset augmentation techniques - while designing adversarial examples and evaluating their efficacy.

Desired End State and Ways to Get There: We note that machine learning driven autonomous systems need to be resilient not just to perturbed variants of the training or test data, but also to *unexpected* inputs, also known as *out-of-distribution examples*, that do not lie close to the training or test data. After all, applications such as autonomous vehicles operate in a dynamic environment and may naturally encounter objects that were not part of the training/test data. This observation has motivated a line of research on *open-world* machine learning, which augments conventional classifiers with another classifier for first deciding if the input is an in-distribution sample or an out-of-distribution sample [21], [79]. Approaches such as ODIN [79] rely on prediction confidence to make a determination about a sample being in-distribution or out-of-distribution. Detected out-of-distribution samples can simply be rejected. However, recent work by Sehwag *et al.* [119] has shown that existing open-world learning frameworks are not robust: an adversary can generate adversarial examples starting from out-of-distribution data to bypass detection. Here, instead of detecting whether the input sample is in-distribution or out-of-distribution, we detect whether each hidden layer, while processing the input sample, is in-distribution or out-of-distribution, as we have shown recently in [126].

Recently, there has been rapid progress in discovering effective defense strategies. However, most of these defense techniques are based on custom tailoring to the employed machine learning model, and the reported robustness would not hold, not only if the model's architecture change, but even if its parameters change due to new training data. For instance, the currently proposed approaches for defending against the FGS attack as well as iterative variants like the CW attack rely on simulating the attack. Furthermore, these defense strategies assume knowledge of the attack strategy while designing the defense, and provide little or no guarantees if the attacker decides to make, even very slight, changes to its strategy (e.g., choosing another model than the target model). Finally, while there is a vast literature on combating adversarial *intentional* machine learning attacks, there is little available work on designing the models to be resilient to unexpected inputs, that could result for example from unexpected behavior of other system components.

We believe that there are three key aspects regarding the challenge of designing resilient machine learning algorithms: **1: Perceptibility of the attack.** In the case of images, this is easily defined through human perception. For other applications, straightforward detection mechanisms should be used to judge how easy is it to detect the presence of an attack. **2: Value of knowledge in adversarial and uncertain environments.** For example, how to best exploit a finite-length secret key to protect a machine learning model. **3: Computational cost associated with attacking a target model.** Modern-day cryptography approaches rely on high computational cost associated with breaking their defense. Ideally, if the attacker has full knowledge of the system and the defense strategy, as well as an unlimited computational power, then there is no hope for resilience beyond the fundamental limits of the learning model; see e.g., [13] for the design of classifiers that consider such attacks.

B. RESILIENT ML ALGORITHMS

Current State Over the past few years, significant advances in ML have led to widespread adoption and deployment of ML in security- and safety-critical systems such as self-driving cars, malware detection, and gradually in industrial control

systems. However, ML systems, despite their impressive capabilities, often demonstrate unexpected/incorrect behaviors in corner cases for several reasons such as biased training data, overfitting, and underfitting of the models. In safety- and security-critical settings, an attacker can exploit such incorrect behaviors to cause disastrous effects like a fatal collision of a self-driving car. Even without an attacker trying to cause harm, a Tesla car in autopilot recently crashed into a trailer because the autopilot system failed to recognize the trailer due to its white color against a brightly lit sky.

Existing ML testing approaches rely mostly on manually labeled real-world test data or unguided ad-hoc simulation to detect such corner-case errors. But these approaches do not scale well for real-world ML systems and only cover a tiny fraction of all possible corner cases (e.g., all possible road conditions for a self-driving car). A promising and active line of research is to build a novel set of testing and verification tools for systematically finding such cases and ensuring security and safety of ML systems.

Our key insight is that most limitations of existing ML testing techniques result from their blackbox nature, i.e., they do not leverage an ML system's internal behaviors (e.g., outputs of intermediate layers) to guide the test input generation process. We have been developing whitebox testing and verification tools for performing static/dynamic/symbolic analysis of ML systems [109], [136], [143], [144]. These types of analyses have been used successfully for testing and verification of traditional software. However, the existing tools are not suitable for testing/verifying ML systems for the following reasons. First, traditional software logic is written by the developer while the logic of ML systems is inferred automatically from training data. Next, unlike most traditional software, ML systems tend to be highly non-linear. Finally, for some of these autonomous systems, it is a challenge to specify what is the expected behavior, considering the large number of possible interactions among the cyber, physical, and human elements. Therefore, support will be useful for specifying envelopes of desirable (and/or undesirable) behavior from these systems, which should then be decomposed into desirable or undesirable output states from each constitutent algorithm.

Over the last three years, we have been building new testing and verification tools to bring more rigor to DL engineering [78], [109], [136], [143], [144]. Given the challenges in specifying a full functional spec of DNNs, we design our tools to check transformation-invariant properties such as slight light condition change must not change the image class." We have explored different design tradeoffs between scalability, completeness, and soundness. Our tools have found thousands of corner-case errors in different DL systems including state-of-the-art image classifiers, object detectors, malware detectors, self-driving car software, and cloud computer vision systems built by Google, Amazon, IBM, and Microsoft. We have also been able to verify some of these DL systems on popular datasets. Our testing and verification tools often outperform other existing tools by orders of magnitude (5,000x on average). We are encouraged to see that the concepts and algorithms in our tools have already started to gain adoption by other research groups and the industry [33], [99], [155].

Desired End State and Ways to Get There: Despite the promising initial results are, there are many difficult open challenges that are not yet addressed. For example, existing DNN verification tools focus on verifying properties on a limited set of test samples with the hope that the guarantees achieved on individual samples generalize to unseen samples. One way to minimize such assumptions is to try to adapt existing specific testing and verification techniques (e.g., interval analysis, mixed-integer programming) to reason about distributions of inputs instead of individual inputs. Another interesting direction is to support a richer set of safety properties, different types of neural networks (e.g., RNNs), and different activation functions such as Sigmoid and tanh.

C. IMMUNE-INSPIRED RESILIENT ALGORITHMS

Immune-inspired algorithms fall in one of the following three sub-fields: *clonal selection*, *negative selection*, and *immune network algorithms*. These techniques are commonly used for clustering, pattern recognition, classification, optimization, and other similar ML domains. They are relevant to design of resilient systems because they are (under domain-specified assumptions) able to adapt to uncertain system conditions or unexpected inputs.

Often, these immune-inspired algorithms start with processes reminiscent of natural selection deploying the following steps: **initialization**, **selection**, **genetic operations** (encompassing crossover and mutation), and **termination**, which can occur when the algorithm has reached a maximum runtime or a set performance threshold. Each of these steps mimics a particular phase in the natural selection process. Further, in the selection phase of immune-inspired algorithms, there is a metric such as fitness function that measures how viable the solution is. Relevant to our discussion, the fitness function should incorporate metrics for resilience rather than just raw performance, *e.g.*, how does the transformation make the system more immune to new kinds of attacks. Alternately, reinforcement learning (RL) can enable the selection of the best fitness function and also confer resilience to the algorithms. This is especially helpful in the case of evolutionary algorithms such as genetic algorithms being used to solve difficult optimization problems where possible drawbacks are the long time to convergence and the possible convergence to a *set of fitness functions* on the Pareto frontier, rather than to a single optimal point.

Evolutionary algorithms: Evolutionary algorithms are useful because they can be used to search for resilient operating points of computing systems in a manner that does not incorporate strong assumptions about the behavior of the underlying system. There are essentially three very similar evolutionary algorithms: genetic algorithms (GA), particle swarm optimization (PSO), and differential evolution (DE), with GA more suitable for discrete optimization, while PSO

and DE being natural fits for continuous optimization processes. In general, for evolutionary algorithms, after initialization, the population is evaluated and stopping criteria are checked. If none are met, a new population is generated, and the process is repeated till the criteria are met. For increasing the resilience of these algorithms, diversity-aware variants of these algorithms have been proposed [51]. In most current applications, the optimization process occurs offline, however, some recent systems have evolved to combine offline training with further online adaptations, such as in our recent work on optimization of configuration parameters of database systems in the face of dynamic real-world workloads [82], [83]. Our systems combine offline training of the neural network with *online adaptation using time-efficient genetic algorithms* to search for discrete optimized state spaces. Such a design makes the system more agile to real-world variability, such as intercepting dynamic, fast changing workloads querying a database.

Desired End State and Way to Get There: For resilient system design, we would want static training and configuration of the system to be complemented with dynamic learning, even in the face of sparse data. Drawing inspiration from the immune system, one can think of the following characteristics for the algorithms that may confer real-world resilience: *stochasticity* (increasing the exploration space); *reinforcement learning*, which can contribute to emergent behavior without global coordination; *stigmergy* where the "agents" interact with the environment. Overall, these characteristics result in an emergent and probabilistic behavior, with redundancy and adaptivity in real-world settings, which are ideal for resilience.

D. PROGRAM SYNTHESIS FOR RESILIENCE

A resilient software system needs to be able to rapidly adapt itself for perturbations. This is essentially a complex and intellectually demanding programming task. Program synthesis, the process of automatically generating programs that meet the user's intent, holds promise to automate this programming task and significantly increase the level of autonomy. The last decade has seen tremendous progress in the efforts of program synthesis techniques, including the synthesis of SQL queries [31], cache coherence protocols [139] and network configurations [46], [133], or productivity software like Microsoft Excel [57], [58].

A major challenge for program synthesis is how to obtain a precise specification that reflects the programmer's goals. Toward addressing this challenge, programming-by-examples (PBE) has been an appealing technique [121], [122]. A PBE system is given a set of input-output examples and tasked to find a program whose behavior matches the given examples through iterative interactions with the user. Another promising technique is sketch-based synthesis [128], in which the programmer specifies a synthesis problem as a sketch or template, which is a program that contains some unknowns to be solved for and some assertions to constrain the choice of unknowns. While the combination of PBE and sketch-based synthesis has seen many successful applications [27], [70], [123], [124], [129], these techniques do not immediately allow the programmer to describe the resilience aspects of the target program, which are usually quantitative and optimization-oriented. For example, a critical assumption of PBE is that the user knows what the expected output is, at least for some sample inputs. In the resilience context, however, the programmer usually cannot quantitatively determine how resilient a program is. Similarly, providing a sketch of the desired resilient program is also challenging because the programmer may not know how a resilient program looks like. Therefore, automatic programming for resilient systems requires novel, user-friendly modalities for describing resilience-related objectives.

While there has been substantial effort in formally verifying hardware and software, these efforts have largely focused on functional correctness: ensuring that the system has the functionality you want. The problem is that verification is only as good as your specification: if your specification leaves details out (e.g., some functionality is unspecified), then formal guarantees do not address that aspect of your system at all. More importantly, *even if all the functionality of your system* is captured, specifications often do not consider *non-functional* aspects: timing, energy usage, interaction models, etc. These gaps in the specification leave open vulnerabilities. For example, underspecifying the timing information of the system may leave timing side channels open, allowing attackers to glean information about a system or even perturb its behavior. A specification that does not account for the interaction model of a system may not properly account for the ways that a human interacts with a system (e.g., not considering certain classes of inputs because the specification designer does not account for humans providing perverse inputs). In the context of this under-specification, no amount of verification can help provide resilience.

Of course, as specifications get more complex to account for all of the possible vulnerabilities and interactions, the *scalability* of formal techniques comes under pressure, and formal verification becomes considerably harder. Indeed, this means that in the presence of difficult verification tasks such as modeling systems that involve human interaction, practitioners often use highly simplified models of the system under inspection: easing the verification task by reducing the fidelity of the verification. While this "solves" the scalability problem, it leaves open the question of *how* to do this model simplification: a model must still capture enough behavior of the system for the formal guarantees to be meaningful. Deciding how to model systems, therefore, becomes a serious bottleneck for verification, and there have correspondingly been only a handful of studies of formal verification for human-in-the-loop systems.

Nevertheless, we identify verification of human-in-the-loop systems as an important challenge for the design of future systems, especially as more and more systems represent a collaboration between automation and humans. Consider, for example, airline auto-pilot systems that expect certain input behavior from humans but fail when the system enters an

unexpected regime that causes humans to apply unexpected control inputs.

Desired End State and Way to Get There: We propose a couple of promising directions to pursue in this space. First, can models be automatically developed for human-in-the-loop systems? Is it possible to automatically simplify a complex model (that might be automatically generated from observations of a system) to target particular desired properties such that humans can then intervene only to ensure that the model is faithful? As an example, consider observing the throttle inputs to a vehicle along with observing the motion of that vehicle to automatically infer a model relating the controls to the state of the vehicle. Second, can we tackle the scalability problem by *detecting simpler properties?* Rather than pursuing full formal specification and verification, it may be sufficient to make human-in-the-loop systems more robust by automatically identifying and flagging unexpected behavior (e.g., a conflict between the current state of an airplane and the types of control inputs being applied by a pilot), relying on the *presence* of a human in the loop to perform more fine-grained corrective action?

Constructing resilient systems with human in the loop further raises new challenges. The system designers should formally specify the *envelope* behaviors of humans and expected by humans. For instance, the design of a driver-assistance system may specify the maximum tolerable latency in recognizing objects to be 10 ms; an interactive image retrieval system may specify the minimum time that the user is given to annotate an image; an AI-based auto-completion code composer may specify the maximum human input rate is 50 program tokens per minute. With such explicit specifications of human behaviors, compiler and runtime may reason about system resilience by taking into human factors into account. For instance, they may assert if the human users are given enough think time to react to the detected anomaly, or if the human users are overwhelmed by the amount of training samples they have to annotate.

As the demands of resilience, and formal design principles for resilience, grow, we believe a key problem that should be tackled is scalability. How can larger systems be verified? How can more complex specifications be verified? Are there modular approaches to specification such that the verification task can be tuned to the particular property that we desire to address?

IV. RESILIENCE BY REACTION

Here we talk about the online measures to deal with perturbations to ensure that as much of the system functionality as possible is maintained. We structure our discussion in terms of the execution platforms (mobile, edge, cloud, or HPC) and the algorithms that are executing. That is, we consider what changes can be made to either the execution platform or the algorithms to ensure that the cyber system continues to operate in a resilient manner despite the occurrence of perturbations at runtime.

Like in the design section, we give in the side bar a distillation of the significant problems that we discuss in this section and the high-level solution approaches being developed. For the solution approaches, we categorize them as *Nascent* or *Developing*.

Problems:
1. Live reconfiguration or adaptation of distributed applications
2. Detecting anomalies in streaming textual data
3. Approximate computing while meeting resilience guarantees
4. Handling stragglers in distributed computation
5. Robust policies for adapting autonomous systems

Solution Approaches:
1. Live reconfiguration of distributed applications $\boxed{N}$
2. Monitoring and optimizing network middle boxes $\boxed{D}$
3. Stream data mining with soft real-time guarantees $\boxed{N}$
4. Flexible approximation and execution on a variety of platforms $\boxed{D}$
5. Straggler mitigation techniques being data-centric, heterogeneous, and proactive $\boxed{D}$
6. Specification of high-level resilience goals for autonomous systems $\boxed{N}$
7. ML + Control Theory to achieve these goals $\boxed{N}$

$\boxed{N}$: *Nascent;* $\boxed{D}$: *Developing*

A. TUNING CONFIGURATIONS OF DISTRIBUTED APPLICATIONS FOR RESILIENCE

Given the rise in data being generated by different sectors, especially in IoT and automation, scalable data engines, low-latency in-memory engines (e.g., Redis), and stream analytics engines, are on the rise. In the context of scalable data processing engines, we can consider the changing, unpredictable workload patterns as perturbations to the system, in the face of which the system will need to react. Without such reaction, the rate of servicing the requests will drop, and in some pathological cases, requests for services will be silently dropped.

Most static database configuration tuners, such as [43], [82], [142], tend to be "reactive" because they use optimization techniques for changing the configuration parameters *when* there is a change in the application characteristic, e.g., change in read-write ratios for database workloads. However, for global-scale, multi-tenant execution pipelines and data repositories, such as the metagenomics repository MG-RAST [26], [149], the workloads may be more dynamic and unpredictable, making it harder to "react" to workload changes on the fly. The goal for the reconfiguration is to maximize the system's performance, using metrics

such as the database's throughput and tail-latency. For such cases, recently, predictive configuration tuners that can work with dynamic workloads have been designed, such as the NoSQL database configuration tuner SOPHIA [83] and *Optimus Cloud* [81]. SOPHIA incorporates a workload predictor *and* a cost-benefit analyzer (CBA) in the optimization protocol that takes into account the cost of a reconfiguration (transient dip in throughput, possible transient unavailability of data) and the benefit (improved performance due to more optimized configuration). Subsequently, the system's configuration parameters are changed only when the CBA determines that the benefit outweighs the cost of reconfiguration. Then the system implements a graceful, decentralized scheme for the reconfiguration, so that data never becomes unavailable or is availability-aware, in sync with the organization's service-level agreement (SLA). In the context of analytics workloads running on cloud platforms, some current works, Selecta [73] and Cherrypick [4] can optimize the parameters of the cloud computing environment by predicting what will be optimal for the just-arrived application. Naturally, all this is predicated on accurate enough prediction of changing workload patterns.

End State and How to Get There: The continuing challenge remains to perform live upgrade of systems due to various events (changes in workload characteristics, data availability or consistency requirements, spatial migration of computing equipment), without degrading the data availability. An additional dimension to this problem is introduced by the use of cloud-hosted software, including some hybrid solutions, where part of the execution is on-premises and part on a remote cloud platform. The update could be to the configuration parameters, or more intrusively, to the software itself.

We can get there by learning from the long line of work on live upgrades of software systems, primarily focused on single-node software [64], [130]. We will have to bring in new distributed protocols that can progressively upgrade a distributed application, while keeping data continuously available and while respecting any end-user consistency requirement (SLA). It will be important to bring in a predictive component to such work, so that the reconfiguration can be initiated prior to the event, but in anticipation of it. Such proactive action can ensure high availability as well as decision as to whether the reconfiguration is beneficial at all. With respect to cloud deployments, cloud providers have mechanisms for data migration and VM migration but they have costs in terms of performance or simply $ costs. Therefore, the prediction can determine whether the benefit normalized by the $ cost or the transient performance impact is tolerable to the application.

B. DISTRIBUTED ENCLAVE DEFENSE USING CONFIGURABLE EDGES

Problem and Current State: In a geographically distributed system, the state of the network can change rapidly due to any of a number of factors. As mentioned in the introduction, perturbations can result from failures, overt attacks, or simply competition for resources. In most environments, there is no central arbiter with global knowledge of the state of the network. Instead, endpoints at the edges must infer network characteristics and adapt to changes to those characteristics. This adaptation can take many forms, including rerouting traffic, transforming content, and others.

There is a long history of adaptation on a case-by-case basis. For instance, Fox *et al.* described a mechanism for dynamic transcoding of web images into low-resolution forms [49]. Split TCP [10], [11] separates a TCP connection into one connection between a client and a proxy, and another connection between the proxy and a server; this allows the proxy to treat each part of the connection appropriately for its characteristics, such as high delay or loss. Middleboxes [120] are a generalization of this approach, interposing for various optimizations including Split TCP [76].

A holistic view that deals with dynamic changes to the network topology and workloads is more challenging. The DEDUCE[1] system from Perspecta Labs adds a "bump in the wire" between edges of a network and the WAN. The DEDUCE box is a middlebox that monitors system behavior and performs a number of optimizations. It splits connections for TCP and UDP , allowing it to transparently reroute via other DEDUCE edge nodes, change characteristics such as TCP congestion optimizations (e.g., from Cubic [62] to BBR [22], forward error correction, transcoding, and others. To do this, it needs a strong view of the state of the network [44], as well as the ability to evaluate ongoing network utility. It continually updates its plans [28] for how best to achieve its goals, which may be implicit (competing best-effort flows) or explicit (information about specific flows with deadlines).

Note that some aspects of its optimizations, such as rerouting, are similar to the functionality of the underlying IP network: IP can dynamically detect network outages and find new routes. The difference is that DEDUCE can apply a more holistic view, for instance rerouting one flow, consistently, along a particular path and a different flow along an alternate path. IP, by comparison, would intermix the packets along each path, leading to packet reordering and other performance implications.

End State and How to Get There: DEDUCE is an example of a set of cooperating middleboxes that operate in isolation from the rest of the network. That is, each DEDUCE endpoint can work with other DEDUCE endpoints to optimize traffic, but any traffic to non-participating edge networks is untreated. For the Internet to be truly resilient, techniques such as these should be more broadly adopted. This means for example that any communicating parties would be able to change their TCP congestion treatment dynamically (e.g., learning the best algorithm for a given situation [127]), add or remove forward error correction or other content-level treatments [80], adapt content to reduce bandwidth requirements [23], etc.

[1]DEDUCE stands for *Distributed Enclave Defense Using Configurable Edges.*

The ability to adapt traffic is only half the solution, however. The bigger challenge is in deciding *how* to adapt, in the presence of incomplete, distributed information. Network tomography [44] is still in relatively early stages, but the ability to gather and process dynamically changing state is crucial to network resilience. A crucial aspect of this processing is being able to distinguish between failures due to resource contention and those due to hardware outages. For instance, if losses are due to congestion, adding extra redundancy via FEC simply contributes to the congestion; if losses are due to a faulty router, redundancy may provide appropriate resilience. Similarly, the reaction to a denial of service attack may be different from the reaction to normal high traffic.

C. DETECTING ANOMALIES IN REAL TIME THROUGH TEXT MINING

In many scenarios, anomalies can also be uncovered through mining textual information (e.g., intruders may respond to system requests with naïve or threatening languages, a system may generate warning messages when detecting unusual situations, or people who observe something abnormal may signal alarms). It is thus critical to detect anomalies through text mining, in real-time. Preprocessing should be conducted beforehand by learning the text embedding space in typical situations with the distributions of phrases, topics, sentences, language features, and sentiments computed and stored, by using advanced text embedding methods developed recently, such as Word2Vec [88], Elmo [111], BERT [41], and JoSE [86].

End State and How to Get There: There is a need to develop stream data mining methods that can operate in real-time, e.g., they can calculate in an online manner, distribution of text elements and monitor the distribution closely. The language features (e.g., phrases, aspects, sentences, paragraphs, or topics) that substantially deviate from the typical ones can be considered as semantic outliers and should be detected and analyzed promptly by integration of text embedding and outlier detection analysis methods [157]. Alternatively, one may also use classification methods to train the system beforehand by collecting text messages in previously happened abnormal situations and go against the text happening in usual situations and such trained models can be used to signal anomalies on the fly. One can also integrate text classification and text outlier detection mechanisms to further enhance the quality of online anomaly detection.

D. APPROXIMATE COMPUTATION WITH RESILIENCE GUARANTEES

Problem and Current State: Many computations are inherently approximate—they trade off quality of results for lower execution time or lower energy. Approximate computing has recently emerged as an area that exposes additional sources of approximation at the computer system level, e.g., in programming languages, compilers, runtime systems, operating systems, and hardware architectures, thereby enabling us to re-define how we think about programs that implement novel solutions to an important class of problems. This has important implications for resilience because many demanding applications (such as, streaming video analytics) cannot run on resource-constrained devices (such as, IoT devices) because they exhaust the limited resources (memory, memory bandwidth, compute, IO, etc.). Therefore approximation has emerged as a potential technology, thus broadeing the domain of possible execution platforms for a wide variety of applications. One challenge of the area of approximate computing has been that the accuracy and performance of applying approximate system-level techniques to a specific application and input sets are hard to predict and control. Today this leads to too conservative choices for approximation [75], unacceptable quality outputs [9], [117], and even incorrect executions [116]. While the current approximate computing approaches show that the techniques have a lot of promise, making robust predictions about accuracy and performance is a key challenge to successful adoption of approximate computing in real-world applications.

The relevant current works in this space (approximation with resilience guarantees) answer the following three broad questions, in one or more of the domains of mobile applications, streaming video analytics, image processing, visualization of scientific computation, etc.

1) *When to approximate:* It searches for the period of the application's execution that is most productive to approximate. This is driven by early evidence that depending on when a specific technique is applied, there may be wide variations in time to convergence of the application or the quality of the output [90]. The granularity of the decision will be application specific and also subject to execution time constraints. Finer-grained monitoring and control are likely to lead to better performance-quality tradeoff, but with a law of diminishing returns. However, such monitoring and control come with their own overhead as well.
2) *How to approximate:* Any approximation technique typically accommodates one or more configuration settings, which captures how aggressively the approximation is done. For example, with a Neural Network-based video analytics query processing, we have to determine what is the optimal number of layers or the level of downsampling of the video frame. As another example, for a demanding computational genomics application, we have to decide how to segment the data and process in parallel for creating an SVM model in a distributed manner [54]. Consider that many applications in our target domains comprise pipeline of multiple software components or methods, each of which can benefit from one of several approximation techniques, and each technique comes with its configuration setting.
3) *How to approximate in input-aware manner:* It appears from some early evidence [75], [152] that in some important cases, the above two decisions have to be made in an input-aware manner. For example, if one is approximating a video stream and the stream consists of

relatively static scenes, more aggressive approximation can be applied than if it is a sports scene with fast movements. Particularly, since we want to bound the accuracy loss with approximation, it is important to take the input dependence into account. For our target domains, the characteristic of the input may change within the stream, requiring that the *when* and *how* decisions be revisited.

End State and How to Get There: There is the need to provide sound and practical techniques to approximate computation, under varied and unseen input data, while bounding the loss in accuracy and providing robust estimates for reduction in energy consumption and other resources. This will enable the grander vision off applications that can "flit" effortlessly between multiple execution platforms depending on the three axes of what is the capability of the platform, what is the resource demand of the computation (including approximation), and what is the cost of moving computation or data and orchestrating possibly distributed execution among the platforms.

There is the need to develop core algorithms to predict the impact of approximate computing on the accuracy and output quality. Further, we should create the models such that they take into account the input dataset and the state of the execution in deciding on the appropriate approximation configuration. The current approximate computing techniques are often inflexible and may miss profitable approximation opportunities or may mispredict the error rate. They are also fragile in the sense that their performance can fluctuate unacceptably under different input datasets. Fine-grained input-aware approximation algorithms that the community is developing has the potential to overcome these key challenges of approximate computing. Moreover, there is the need to show how to combine system-level and application-specific approximation techniques in the various target domains.

E. RESILIENCE TO STOCHASTIC TASK SIZES IN DISTRIBUTED COMPUTATION

Large scale computing jobs require multi-stage computation, where computation per stage is performed in parallel over a large number of servers. The execution time of a task on a machine has stochastic variations due to many contributing factors such as co-hosting, virtualization, hardware and network variations [30], [151]. A slow server can delay the onset of next stage computation, and we call it a *straggling* server. One of the key challenges in cloud computing is the problem of straggling servers, which can significantly increase the job completion time [53], [59]. Resilience to straggling servers is essential to counter the possibility of missing deadlines in job execution. This is relevant in autonomous systems of the kinds introduced earlier because there are (soft) timing requirements and many workloads execute on cloud computing platforms.

Current State: It has been observed that task execution times have significant variability, partly due to resource sharing by multiple jobs [39]. The slowest tasks that determine the job execution time are known as "stragglers". Some of the key approaches to mitigate the effect of stragglers are to have speculative execution which acts after the tasks have already slowed down [40] or proactive approaches that launch redundant copies of a task in the hope that at least one of them will finish in a timely manner [5], [7] or efficient scheduling strategies that aim to have smaller completion times with uncertain task execution times which do not change [2], [60], [61]. Some classic work in the context of parallel programs for multicore machines [16], [29] has shown how to do work stealing and task scheduling to minimize various objectives, including task completion time. When redundant tasks are launched on different servers, one approach is to perform an erasure-coding that provides significantly more flexibility as compared to replication. By having coding-theory based redundancy approaches, the user waits for any k out of the n servers to finish, while each server runs a smaller fragment of the task [3], [8]. Coding-theoretic techniques have been proposed to mitigate the effect of stragglers in gradient computation [118], [135], [154]. In [115], an approximate variant of the gradient coding problem is introduced, in which approximate gradient computation is done instead of the exact computation. A stochastic block code and an efficient decoding method for approximate gradient recovery are provided in [25].

Desired End State and Ways to Get There: Even though different approaches for straggler mitigation have been provided, efficient approaches require a holistic framework to understand the different design tradeoffs, including the completion time of the jobs, and the additional server costs spent for the jobs that will eventually not be completed. In order to come up with such a holistic framework, it would be essential to develop data-centric proactive approaches that leverage coding-theoretic and queuing-theoretic techniques. We note that deep reinforcement learning based approaches have been considered for scheduling jobs on the servers [1], [34], [110]. Reinforcement learning approaches with speculative execution to mitigate stragglers have been considered in [95]. However, such approaches do not consider multiple jobs, heterogeneous servers, coding-theoretic flexibilities, and approximate computing. Further, the allocation among different users must satisfy joint objectives, e.g., fairness, thereby needing decentralized, scalable solutions rather than centralized approaches [1]. Accounting for all these degrees of freedom significantly enlarges the design space and efficient approaches that explore the design space is an interesting problem.

F. ADAPTABILITY WITH RESILIENCE GUARANTEES

Current State: Computing systems must function effectively in dynamic environments where application workloads, available resources, and user requirements can all fluctuate in unpredictable ways. To handle these dynamics, system developers create *mechanisms* that enable the system to detect changes and then react to those changes. Unfortunately, the *policies* that govern how these mechanisms are applied are often ad hoc and heuristic based. These heuristics are developed by experts and tend to work very well on the system for which they were designed, but they are not robust to changes.

As an example, Samsung's scheduler for the Galaxy S9 smartphone has many heuristics governing when to change clockspeed and when to migrate a process between its fast, high-power cores and its slower, energy-efficient cores. The S9+ was anticipated to be an upgrade with higher performance and longer battery life due to improved processor design. However, product reviewers found that in practice, the S9+ exhibited lower performance and shorter life as scheduling heuristics tuned for the S9 produced poor results on the S9+ [50]. The S9/S9+ is one example demonstrating how fragile heuristic-based resource management can be, on complex, modern processor designs, and it demonstrates the need for more principled resource management.

Desired End State: Our goal is the design and development of a set of robust policies that govern how autonomous systems should react to unforeseen circumstances. These policies should be based on well-founded principles and come with clearly stated assumptions about the conditions under which they would be expected to work and the mechanisms available for the policies to be enforced. Furthermore, we advocate for autonomous systems that do not just react, but react to accomplish some high-level, user-defined goal. For example, such goals might be meeting a certain latency constraint with minimal energy or finding the most accurate model on an energy budget.

Ways to Get There: A first step to achieve the vision of adapting to high-level goals is to make those goals explicit in the program. In other words, there should be a (possibly domain-specific) language for describing a program's quantifiable behavior and the desired range of behavior that represents successful deployment. Furthermore, this language should also specify which behavior is a constraint—which must be respected for correct operation—and which is an objective—to be minimized or maximized subject to the constraints. In addition, this language should describe what system components can be changed to affect the goals. The idea of specifying high-level goals and mutable program components was a key part of the Self-aware computing project [66], language support for making this specification a first-class object appeared in the later Proteus project [12], and VStore [153], a video data store that respects encoding/decoding throughput constraint while maximizing storage efficiency.

Clearly defining goals is key to resilience by reaction, as it is only through the definition of goals that the system can observe they are not being met and react to restore correct operation. Of course, there is still the question of how to react, or how to map measurements of goals into appropriate settings for the mutable system components. We advocate a mixture of machine learning and control theory to achieve this mapping. Machine learning models are well-suited to capturing the complex tradeoff spaces that can arise in computing systems with competing goals and many mutable components. Control theory is well-suited to ensuring that constraints are met. Recent work demonstrates how the two can be combined to achieve the best of both approaches [89].

While recent research demonstrates that it is possible to build adaptive systems to meet goals, one major challenge is unaddressed: How can multiple, independent adaptive systems collaborate effectively, if developed independently by different stakeholders? For example, a mobile application might have goals in terms of responsiveness and image quality, while a mobile operating system might have goals in responsiveness and energy efficiency. If those systems are developed independently, they could easily make conflicting decisions, negating their potential benefits [65]. Thus, a common interface is likely necessary for specifying adaptive components and the goals they effect; a high-level negotiation mechanism are needed for coordinating their adaptations amongst different such components.

V. THE ROAD AHEAD

Here we look at the road ahead with a summary of the short-term and mid-term research and transition challenges.

1) *Creating resilient systems out of individually vulnerable components:* There will be increasing needs to build resilient systems out of components that are *not* individually resilient to the perturbations that the system will have to face. Potentially there will be a large number of such components composing the system. These components will be vulnerable due to innate design and implementation vulnerabilities, or due to unpredictable interactions with the external environment, either cyber or physical.
2) *Speeding up the cycle of design and generation of attacks and defenses against ML algorithms:* There will be a two-pronged need in this space of resilient ML—designing algorithms that are resilient by design to a well-quantified set of perturbations and speeding up the discovery of vulnerabilities in realized implementations of the ML algorithms. The speeding up will imply a partially automated process for discovery and patching of vulnerabilities in the ML applications, as was envisioned by the DARPA Cyber Grand Challenge competition [36].
3) *Synthesis of resilient programs by automated means:* There is a growing body of work on automatic synthesis of programs from specifications. We have to consider that the synthesized program meets well-quantified resilience guarantees. The automatic synthesis can take the form of full program synthesis or, what is more likely, augmentation of an existing program for the purpose of increasing its resilience. In this approach, we can take inspiration from non-traditional sources, such as, immune systems in biological organisms.
4) *Automated configuration of increasingly complex systems, for performance as well as for resilience:* This includes efforts to automatically navigate the large space of configuration parameters and determine the close-to-optimal settings within a reasonable time bound, perhaps even online. While there is a growing body of

work on automated configuration for performance, it will become important to perform such reconfiguration while meeting resilience goals, such as, server uptime and data availability.

5) *Use of compute power close to the client devices to increase the resilience of large-scale multi-tier systems:* This involves the use of edge computing resources for redundant execution, in addition to its traditional use for reducing latency of short-running queries. We will move to some autonomous systems whose algorithms can execute in parts in each of the three tiers of execution—client devices, edge computing devices, and cloud computing devices. The partitioning can happen flexibly, even at runtime, and can be done not just for performance but also for resilience. Different degrees of redundancy will be employed for different applications and at different tiers of the hierarchy.
6) *Expanding the scope of approximate computation and distributed computation to include resilience as a first-order principle:* Approximate computation will become an increasingly powerful means to execute demanding applications on resource-constrained devices, or where energy resource is at a premium. We need methods to approximate computation while still being able to provide resilience guarantees, likely probabilistic. The same applies to distributed computation, which will become increasingly relevant to scale up demanding ML applications. In such cases also, the loss in accuracy relative to the centralized computation must be bounded and quantified.

ACKNOWLEDGMENT

Any opinions, findings, and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] M. Agarwal and V. Aggarwal, "A reinforcement learning based approach for joint multi-agent decision making," 2019, *arXiv:1909.02940*.
[2] V. Aggarwal, M. Xu, T. Lan, and S. Subramaniam, "On the optimality of scheduling dependent mapreduce tasks on heterogeneous machines," 2017, *arXiv:1711.09964*.
[3] M. F. Aktas, P. Peng, and E. Soljanin, "Straggler mitigation by delayed relaunch of tasks," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 45, no. 2, pp. 224–231, 2018.
[4] O. Alipourfard *et al.*, "Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics," in *Proc. USENIX Symp. Networked Syst. Des. Implementation (NSDI)*, 2017, pp. 469–482.
[5] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Effective straggler mitigation: Attack of the clones," in *Proc. 10th USENIX Symp. Networked Syst. Des. Implementation (NSDI)*, 2013, pp. 185–198.
[6] B. D. O. Anderson, S. Mou, U. R. Helmke, and A. S. Morse, "Decentralized gradient algorithm for solution of a linear equation," *Numerical Algebra, Control Optimization*, vol. 6, no. 3, pp. 319–328, 2016.
[7] A. Badita, P. Parag, and V. Aggarwal, "Optimal server selection for straggler mitigation," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 709–721, Apr. 2020.
[8] A. Badita, P. Parag, and V. Aggarwal, "Sequential addition of coded tasks for straggler mitigation," in *Proc. IEEE Infocom*, 2020.
[9] W. Baek and T. M Chilimbi, "Green: A framework for supporting energy-conscious programming using controlled approximation," in *Proc. ACM Sigplan Notices*, 2010, vol. 45, pp. 198–209.
[10] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for mobile hosts," in *Proc. 15th Int. Conf. Distributed Comput. Syst.*, 1995, pp. 136–143.
[11] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Trans. Networking*, vol. 5, no. 6, pp. 756–769, 1997.
[12] S. *et al.*, "Proteus: Language and runtime support for self-adaptive software development," *IEEE Softw.*, vol. 36, no. 2, pp. 73–82, Mar. 2019.
[13] A. N. Bhagoji, D. Cullina, and P. Mittal, "Lower bounds on adversarial robustness from optimal transport," in *Conf. Neural Inform. Process. Syst. (NeurIPS)*, 2019.
[14] A. N. Bhagoji, W. He, B. Li, and D. Song, "Practical black-box attacks on deep neural networks using efficient query mechanisms," in *Proc. Comput. Vision - ECCV 2018-15th Eur. Conf., Munich, Germany, Sep. 8-14, 2018, Proc., Part XII*, 2018, pp. 158–174.
[15] B. Biggio *et al.*, "Evasion attacks against machine learning at test time," in *Proc. Joint Eur. Conf. Mach. Learning and Knowledge Discovery Databases*, 2013.
[16] R. D. Blumofe, C. F. Joerg, B. C. Kuszmaul, C. E. Leiserson, K. H. Randall, and Y. Zhou, "Cilk: An efficient multithreaded runtime system," *J. Parallel Distrib. Comput.*, vol. 37, no. 1, pp. 55–69, 1996.
[17] Z. Bouzid, M. G. Potop-Butucaru, and S. Tixeuil, "Optimal byzantine resilient convergence in uni-dimensional robot networks," *Theoretical Comput. Sci.*, vol. 411, no. 34, pp. 3154–3168, 2010.
[18] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.
[19] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control Robotic Networks*. (Applied Mathematics Series). Princeton, NJ, USA: Princeton Univ. Press, 2009.
[20] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2014.
[21] E. Cabana, R. E. Lillo, and H. Laniado, "Multivariate outlier detection based on a robust mahalanobis distance with shrinkage estimators," 2019.
[22] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *Commun. ACM*, vol. 60, no. 2, pp. 58–66, Jan. 2017.
[23] S. Chandra, C. S. Ellis, and A. Vahdat, "Application-level differentiated multimedia web services using quality aware transcoding," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 12, pp. 2544–2565, Mar. 2000.
[24] T. Chang, A. Nedic, and A. Scaglione, "Distributed constrained optimization by consensus-based primal-dual perturbation method," *IEEE Trans. Autom. Control*, vol. 59, no. 6, pp. 1524–1538, Jun. 2014.
[25] Zachary Charles and Dimitris Papailiopoulos. Gradient coding via the stochastic block model. *arXiv preprint arXiv:1805.10378*, 2018.
[26] Somali Chaterji, Jinkyu Koo, Ninghui Li, Folker Meyer, Ananth Grama, and Saurabh Bagchi, "Federation in genomics pipelines: Techniques and challenges," *Briefings Bioinf.*, vol. 20, no. 1, pp. 235–244, 2017.
[27] S. Chaudhuri, M. Clochard, and A. Solar-Lezama, "Bridging boolean and quantitative synthesis using smoothed proof search," in *Proc. 41st ACM SIGPLAN-SIGACT Symp. Principles Program. Lang.*, 2014, pp. 207–220.
[28] J. Chen, C. Fang, C. Muise, H. Shrobe, B. C. Williams, and P. Yu, "Radmax: Risk and deadline aware planning for maximum utility," in *Proc. Workshops at Thirty-Second AAAI Conf. Artificial Intell.*, 2018.
[29] Q. Chen and M. Guo, "Adaptive workload-aware task scheduling for single-isa asymmetric multicore architectures," *ACM Trans. Archit. Code Optimization (TACO)*, vol. 11, no. 1, pp. 1–25, 2014.
[30] D. Cheng, J. Rao, Y. Guo, and X. Zhou, "Improving mapreduce performance in heterogeneous environments with adaptive task tuning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 774–786, Mar. 2017.

[31] A. Cheung, A. Solar-Lezama, and S. Madden, "Optimizing database-backed applications with query synthesis," in *Proc. 34th ACM SIGPLAN Conf. Program. Language Des. Implementation*, 2013, pp. 3–14.

[32] CNBC. Zipline, which delivers lifesaving medical supplies by drone, now valued at $1.2 billion, May 2019. [Online]. Available: https://www.cnbc.com/2019/05/17/zipline-medical-delivery-drone-start-up-now-valued-at-1point2-billion.html

[33] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proc. 36th Int. Conf. Mach. Learn. PMLR*, vol. 97, pp. 1310–1320, 2019.

[34] D. Cui *et al.*, "A reinforcement learning-based mixed job scheduler scheme for grid or iaas cloud," *IEEE Trans. Cloud Comput.*, early access, Nov. 14, 2017, doi: 10.1109/TCC.2017.2773078.

[35] D. Wagner and N. Carlini, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2017.

[36] DARPA. Cyber Grand Challenge (CGC), 2016.

[37] L. Nunes, D. Castro, and J. Timmis, "An artificial immune network for multimodal function optimization," in *Proc. 2002 Congr. Evol. Comput.. CEC'02 (Cat. No. 02TH8600)*, 2002, vol. 1, pp. 699–704.

[38] L. N. de Castro and F. J. Von Zuben, "Ainet: An artificial immune network for data analysis," in *Proc. Data Mining: a Heuristic Approach*, 2002, pp. 231–260.

[39] J. D. and L. Andr. Barroso, "The tail at scale," *Commun. ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[40] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[41] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. 2019 Conf. North Amer. Chapter Assoc. for Comput. Linguist.: Human Lang. Technol., NAACL-HLT 2019, Minneapolis, MN, USA, Jun. 2-7, 2019*, pp. 4171–4186, 2019.

[42] F. Dorfler, M. Chertkov, and F. Bullo, "Synchronization in complex oscillator networks and smart grids," in *Proc. Nat. Academy Sci.*, 2013, vol. 110, no. 6, pp. 2005–2010.

[43] S. Duan, V. Thummala, and S. Babu, "Tuning database configuration parameters with ituned," in *Proc. VLDB Endowment*, 2009, vol. 2, no. 1, pp. 1246–1257.

[44] N. Duffield, "Simple network performance tomography," in *Proc. 3rd ACM SIGCOMM Conf. Int. Meas.*, 2003, pp. 210–215.

[45] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. MHS'95. Sixth Int. Symp. Micro Mach. Human Sci.*, 1995, pp. 39–43.

[46] A. El-Hassany, P. Tsankov, L. Vanbever, and M. Vechev, "Netcomplete: Practical network-wide configuration synthesis with autocompletion," in *Proc. 15th USENIX Symp. Networked Syst. Des. Implementation (NSDI 18)*, 2018, pp. 579–594.

[47] K. Eykholt *et al.*, "Robust physical-world attacks on deep learning visual classification," in *Proc. 2018 IEEE Conf. Comput. Vision Pattern Recognit., CVPR 2018, Salt Lake City, UT, USA, Jun. 18-22, 2018*, pp. 1625–1634, 2018.

[48] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1465–1476, 2004.

[49] A. Fox, S. D. Gribble, E. A. Brewer, and E. Amir, "Adapting to network and client variability via on-demand dynamic distillation," *ACM SIGOPS Oper. Syst. Rev.*, vol. 30, no. 5, pp. 160–170, 1996.

[50] A. Frumusanu, "Improving the Exynos 9810 Galaxy S9: Part 2 - Catching Up with the Snapdragon, Apr. 2018," Anandtech [Online]. available: https://www.anandtech.com/show/12620/improving-the-exynos-9810-galaxy-s9-part-2

[51] T. Gabor, L. Belzner, T. Phan, and K. Schmid, "Preparing for the unexpected: Diversity improves planning resilience in evolutionary algorithms," in *Proc. 2018 IEEE Int. Conf. Autonom. Comput. (ICAC)*, 2018, pp. 131–140.

[52] X. Z. Gao, S. J. Ovaska, X. Wang, and M-Y Chow, "Clonal optimization-based negative selection algorithm with applications in motor fault detection," *Neural Comput. Appl.*, vol. 18, no. 7, pp. 719–729, 2009.

[53] P. Garraghan, X. Ouyang, R. Yang, D. McKee, and J. Xu, "Straggler root-cause and impact analysis for massive-scale virtualized cloud datacenters," *IEEE Trans. Serv. Comput.*, vol. 12, no. 1, pp. 91–104, Jan. 2019.

[54] A. Ghoshal, A. Grama, S. Bagchi, and S. Chaterji, "An ensemble svm model for the accurate prediction of non-canonical microrna targets," in *Proc. 6th ACM Conf. Bioinformatics, Comput. Biol. Health Informat. (BCB)*, 2015, pp. 403–412.

[55] I. J. Goodfellow, A. Courville, and Y. Bengio, *Deep Learn.* 2016, Cambridge, MA, USA: MIT Press.

[56] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Intl. Conf. Learn. Representations*, 2015, pp. 1–15.

[57] S. Gulwani, "Automating string processing in spreadsheets using input-output examples," in *Proc. 38th Annu. ACM SIGPLAN-SIGACT Symp. Principles Program. Languages*, POPL '11, 2011, pp. 317–330.

[58] S. Gulwani, W. R. Harris, and R. Singh, "Spreadsheet data manipulation using examples," *Commun. ACM*, vol. 55, no. 8, pp. 97–105, Aug. 2012.

[59] Y. Guo, J. Rao, C. Jiang, and X. Zhou, "Moving hadoop into the cloud with flexible slot management and speculative execution," *IEEE Trans. Parallel Distributed Syst.*, vol. 28, no. 3, pp. 798–812, Mar. 2017.

[60] V. Gupta, B. Moseley, M. Uetz, and Q. Xie, "Stochastic online scheduling on unrelated machines," in *Proc. Int. Conf. Integer Program. Combinatorial Optim.*, 2017, pp. 228–240.

[61] V. Gupta, B. Moseley, M. Uetz, and Q. Xie, "Greed works online algorithms for unrelated machine stochastic scheduling," *Mathematics Oper. Res.*, vol. 45, no. 2, pp. 497–516, 2020.

[62] S. Ha, I. Rhee, and L. Xu, "Cubic: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008.

[63] B. Henz and T. Pham, "Army Research Laboratory essential research area: AI and ML (Conference Presentation)," in Thomas George, Achyut K. Dutta, and M. Saif Islam, editors, *Micro- and Nanotechnol. Sensors, Syst. Appl. X*, vol. 10639. International Society for Optics and Photonics, SPIE, 2018.

[64] M. Hicks and S. Nettles, "Dynamic software updating," *ACM Trans. Program. Lang. Syst. (TOPLAS)*, vol. 27, no. 6, pp. 1049–1096, 2005.

[65] H. Hoffmann, "Jouleguard: Energy guarantees for approximate applications," in *Proc. 25th Symp. Oper. Syst. Principles*, 2015, pp. 198–214.

[66] H. Hoffmann *et al.*, "Self-aware computing in the angstrom processor," in *Proc. 49th Annu. Des. Autom. Conf.*, 2012, pp. 259–264.

[67] M. S. Im, V. R. Dasari, L. Beshaj, and D. Shires, "Optimization problems with low swap tactical computing," 2019, *arXiv:1902.05070*.

[68] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 988–1001, Jun. 2003.

[69] D. Jakovetic, J. M. F. Moura, and J. Xavier, "Fast distributed gradient methods," *IEEE Trans. Autom. Control*, vol. 59, no. 5, pp. 1131–1146, 2014.

[70] J. Jeon, X. Qiu, J. Fetter-Degges, J. S. Foster, and A. Solar-Lezama, "Synthesizing framework models for symbolic execution," in *Proc. ICSE'16*, 2016, pp. 156–167.

[71] Z Ji and D. Dasgupta, "Real-valued negative selection algorithm with variable-sized detectors," in *Proc. Genetic Evol. Comput. Conf.*, 2004, pp. 287–298.

[72] J. Kim and P. J. Bentley, "Towards an artificial immune system for network intrusion detection: An investigation of clonal selection with a negative selection operator," in *Proc. 2001 Congr. Evol. Comput. (IEEE Cat. No. 01TH8546)*, 2001, vol. 2, pp. 1244–1252.

[73] A. Klimovic, H. Litz, and C. Kozyrakis, "Selecta: heterogeneous cloud storage configuration for data analytics," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2018. pp. 759–773.

[74] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," 2016, *arXiv:1611.01236*.

[75] M. A. Laurenzano, P. Hill, M. Samadi, S. Mahlke, J. Mars, and L. Tang, "Input responsiveness: using canary inputs to dynamically steer approximation," in *Proc. ACM SIGPLAN Notices*, 2016, vol. 51, pp. 161–176.

[76] F. Le, E. Nahum, V. Pappas, M. Touma, and D. Verma, "Experiences deploying a transparent split tcp middlebox and the implications for nfv," in *Proc. 2015 ACM SIGCOMM Workshop Hot Topics Middleboxes Netw. Function Virtualization*, 2015, pp. 31–36.

[77] H. J. LeBlanc, *Resilient Cooperative Control Networked Multi-agent Syst.*. Vanderbilt University, 2012.

[78] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," in *Proc. 2019 IEEE Symp. Secur. Privacy (SP)*, 2019, pp. 656–672, 2019.

[79] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *Proc. 6th Int. Conf. Learn. Representations, ICLR 2018, Vancouver, BC, Canada, Apr. 30 - May 3, 2018, Conf. Track Proc.*, 2018.

[80] B. Liu, D. L Goeckel, and D. Towsley, "Tcp-cognizant adaptive forward error correction in wireless networks," in *Global Telecommun. Conf., 2002. GLOBECOM'02. IEEE*, 2002, vol. 3, pp. 2128–2132.

[81] A. Mahgoub *et al.*,"OPTIMUS CLOUD: Heterogeneous configuration optimization for distributed databases in the cloud," in *Proc. 2020 USENIX Annu. Tech. Conf. (USENIX ATC 20)*, pp. 1–15, 2020.

[82] A. Mahgoub *et al.*, "Rafiki: A middleware for parameter tuning of nosql datastores for dynamic metagenomics workloads," in *Proc. 18th ACM/IFIP/USENIX Middleware Conf.*, 2017, pp. 28–40.

[83] A. Mahgoub, P. Wood, A. Medoff, S. Mitra, F. Meyer, S. Chaterji, and S. Bagchi, "SOPHIA: Online reconfiguration of clustered nosql databases for time-varying workloads," in *Proc. 2019 USENIX Annu. Tech. Conf. (USENIX ATC 19)*, 2019. pp. 223–240.

[84] C. M. Martinez, X. Hu, D. Cao, E. Velenis, B. Gao, and M. Wellers, "Energy management in plug-in hybrid electric vehicles: Recent progress and a connected vehicles perspective," *IEEE Trans. Veh. Technol.*, vol. 66, no. 6, pp. 4534–4549, 2016.

[85] H. Mendes, M. Herlihy, N. Vaidya, and V.K. Garg, "Multidimensional agreement in byzantine systems," *Distrib. Comput.*, vol. 28, no. 6, pp. 423–441, 2015.

[86] Y. Meng *et al.*, "Spherical text embedding," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2019.

[87] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods Multi-Agent Netw.*. Princeton University Press, 2010.

[88] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inform. Process. Syst. 26: 27th Annu. Conf. Neural Inform. Process. Syst. 2013. Proc. a Meeting held Dec. 5-8, 2013, Lake Tahoe, Nevada, United States.*, 2013, pp. 3111–3119.

[89] N. Mishra, C. Imes, J. D. Lafferty, and H. Hoffmann, "Caloree: Learning control for predictable latency and low energy," in *Proc. Twenty-Third Int. Conf. Archit. Support for Program. Languages Oper. Syst.*, 2018, pp. 184–198.

[90] S. Mitra, M. K. Gupta, S. Misailovic, and S. Bagchi, "Phase-aware optimization in approximate computing," in *Proc. 2017 IEEE/ACM Int. Symp. Code Gener. Optimization (CGO)*, 2017, pp. 185–196.

[91] S. M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2016.

[92] L. Moreau, "Stability of multi-agent systems with time-dependent communication links," *IEEE Trans. Autom. Control*, vol. 50, no. 2, pp. 169–182, Fab. 2005.

[93] S. Mou, Z. Lin, L. Wang, D. Fullmer, and A. S. Morse, "A distributed algorithm for efficiently solving linear equations and its applications (special issue jcw)," *Syst. Control Lett.*, vol. 91, pp. 21–27, 2016.

[94] S. Mou, J. Liu, and A. S. Morse, "A distributed algorithm for solving a linear algebraic equation," *IEEE Trans. Autom. Control*, vol. 60, no. 11, pp. 2863–2878, 2015.

[95] N. S. Naik, A. Negi, and V. Sastry, "Improving straggler task performance in a heterogeneous mapreduce framework using reinforcement learning," *Int. J. Big Data Intell.*, vol. 5, no. 4, pp. 201–215, 2018.

[96] A. Nedic and A. Ozdaglar, "Distributed sub-gradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, 2009.

[97] A. Nedic, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Trans. Autom. Control*, vol. 55, no. 4, pp. 922–938, 2010.

[98] A. M. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2015.

[99] A. Odena and I. Goodfellow, "Tensorfuzz: Debugging neural networks with coverage-guided fuzzing," 2018, *arXiv:1807.10875*.

[100] Department of the Army, "The operational environment and the changing character of warfare," Tech. Rep. TRADOC Pamphlet 525-92, Department of the Army, 10 2019.

[101] R. Olfati-Saber, J. L. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," in *Proc. IEEE*, 2007, vol. 95, pp. 215–233.

[102] N. Papernot, P. Mcdaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *IEEE Eur. Symp. Secur. Privacy*, 2016.

[103] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. Berkay Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. 2017 ACM Asia Conf. Comput. Commun. Secur., AsiaCCS 2017, Abu Dhabi, United Arab Emirates, Apr. 2-6, 2017*, 2017. pp. 506–519.

[104] H. Park and S. Hutchinson, "A distributed robust convergence algorithm for multi-robot systems in the presence of faulty robots," In *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2015, pp. 2980–2985.

[105] H. Park and S. Hutchinson, "An efficient algorithm for fault-tolerant rendezvous of multi-robot systems with controllable sensing range," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2016, pp. 358–365.

[106] H. Park and S. A. Hutchinson, "Fault-tolerant rendezvous of multirobot systems," *IEEE Trans. Robot.*," vol. 33, no. 3, pp. 565–582, Jun. 2017.

[107] F. Pasqualetti, A. Bicchi, and F. Bullo, "Consensus computation in unreliable networks: A system theoretic approach," *IEEE Trans. Autom. Control*, vol. 57, no. 1, pp. 90–104, 2012.

[108] F. Pasqualetti, F. Dorfler, and F. Bullo, "Attack detection and identification in cyber physical systems," *IEEE Trans. Autom. Control*, vol. 58, no. 11, pp. 2715–2719, Nov. 2013.

[109] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated whitebox testing of deep learning systems," in *Proc. 26th ACM Symp. Oper. Syst. Principles*, 2017.

[110] Y. Peng, Y. Bao, Y. Chen, C. Wu, and C. Guo, "Optimus: An efficient dynamic resource scheduler for deep learning clusters," in *Proc. Thirteenth EuroSys Conf.*, 2018, pp. 3.

[111] M. E. Peters, M. Neumann, L. Zettlemoyer, and Wen-tau Yih, "Dissecting contextual word embeddings: Architecture and representation," in *Proc. 2018 Conf. Empir. Methods Natural Lang. Process., Brussels, Belgium, Oct. 31 - Nov. 4, 2018*, 2018, pp. 1499–1509.

[112] B. Piekarski, B. Sadler, S. Young, W. Nothwang, and R. Rao, "Research and vision for intelligent systems for 2025 and beyond," 2016.

[113] National Public Radio. Medical Cargo Could Be The Gateway For Routine Drone Deliveries, Mar. 2018.

[114] S. Rahili and W. Ren, "Distributed continuous-time convex optimization with time-varying cost functions," *IEEE Trans. Autom. Control*, vol. 62, no. 4, pp. 1590–1605, Apr. 2017.

[115] N. Raviv, R. Tandon, A. Dimakis, and I. Tamo, "Gradient coding from cyclic mds codes and expander graphs," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4302–4310, 2018.

[116] M. Ringenburg, A. Sampson, I. Ackerman, L. Ceze, and D. Grossman, "Monitoring and debugging the quality of results in approximate programs," in *Proc. ACM SIGPLAN Notices*, 2015, vol. 50, pp. 399–411.

[117] M. Samadi, J. Lee, D. A. Jamshidi, A. Hormati, and S. Mahlke, "Sage: Self-tuning approximation for graphics engines," in *Proc. 46th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2013, pp. 13–24.

[118] S. Sasi, V. Lalitha, V. Aggarwal, and B. S. Rajan, "Straggler mitigation with tiered gradient codes," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4632–4647, Aug. 2020.

[119] V. Sehwag *et al.*, "Analyzing the robustness of open-world machine learning," in *Proc. 12th ACM Workshop Artificial Intell. Secur.*, 2019.

[120] V. Sekar, S. Ratnasamy, M. K. Reiter, N. Egi, and G. Shi, "The middlebox manifesto: enabling innovation in middlebox deployment," in *Proc. 10th ACM Workshop Hot Topics Netw.*, 2011, pp. 21.

[121] R. Singh, "Blinkfill: Semi-supervised programming by example for syntactic string transformations," in *Proc. VLDB Endowment*, vol. 9, no. 10, pp. 816–827, 2016.

[122] R. Singh and S. Gulwani, "Predicting a correct program in programming by example," in *Proc. Int. Conf. Comput. Aided Verification*, 2015, pp. 398–414.

[123] R. Singh, S. Gulwani, and A. Solar-Lezama, "Automated feedback generation for introductory programming assignments," in *PLDI*, 2013. pp. 15–26.

[124] R. Singh and A. Solar-Lezama, "Synthesizing data structure manipulations from storyboards," in *ESEC/FSE'11*, 2011, pp. 289–299.

[125] C. Sitawarin, A. N. Bhagoji, A. Mosenia, M. Chiang, and P. Mittal, "DARTS: Deceiving autonomous cars with toxic signs," *CoRR*, abs/1802.06430, 2018.

[126] K. S. Sivamani, R. Sahay, and A. E. Gamal, "Non-intrusive detection of adversarial deep learning attacks via observer networks," *IEEE Lett. Comput. Soc.*, vol. 3, no. 1, Jun. 2020.
[127] A. Sivaraman, K. Winstein, P. Thaker, and H. Balakrishnan, "An experimental study of the learnability of congestion control," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, 2014, vol. 44, pp. 479–490.
[128] A. Solar-Lezama, "Program sketching," *Int. J. Softw. Tools for Technol. Transfer*, vol. 15, no. 5, pp. 475–495, Oct. 2013.
[129] A. Solar-Lezama, L. Tancau, R. Bodik, S. Seshia, and V. Saraswat, "Combinatorial sketching for finite programs," in *ASPLOS'06*, 2006. pp. 404–415.
[130] C. A. N. Soules *et al.*, "System support for online reconfiguration," in *Proc. USENIX Annu. Tech. Conf., Gen. Track*, 2003. pp. 141–154.
[131] D. K. Spencer, S. Duncan, and A. Taliaferro, "Operationalizing artificial intelligence for multi-domain operations: a first look," in Tien Pham, editor, *Artificial Intell. and Mach. Learning for Multi-Domain Operations Appl.*, vol. 11006, pp. 1–10. International Society for Optics and Photonics, SPIE, 2019.
[132] S. Sridhar, A. Hahn, and M. Govindarasu, "Cyber-physical system security for the electric power grid," in *Proc. IEEE*, 2011, vol. 100, no. 1, pp. 210–224.
[133] K. Subramanian, L. D'Antoni, and A. Akella, "Genesis: Synthesizing forwarding tables in multi-tenant networks," in *Proc. 44th ACM SIGPLAN Symp. Principles Program. Lang.*, POPL 2017, 2017, pp. 572–585.
[134] S. Sundaram and B. Gharesifard, "Distributed optimization under adversarial nodes," *IEEE Trans. Autom. Control*, vol. 64, no. 3, pp. 1063–1076, Mar. 2019.
[135] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Int. Conf. Mach. Learn.*, 2017, pp. 3368–3376.
[136] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proc. 40th Int. Conf. Software Eng.*, 2018, pp. 303–314.
[137] J. Timmis and C. Edmonds, "A comment on opt-ainet: An immune network algorithm for optimisation," in *Proc. Genetic Evol. Comput. Conf.*, 2004, pp. 308–317.
[138] L. Tseng and N. H. Vaidya, "Asynchronous convex hull consensus in the presence of crash faults," in *Proc. ACM Symp. Principles Distrib. Comput.*, 2014, pp. 396–405.
[139] A. Udupa *et al.*, "TRANSIT: Specifying Protocols with Concolic Snippets," in *Proc. PLDI*, 2013, pp. 287–296.
[140] N. H. Vaidya, Iterative byzantine vector consensus in incomplete graphs. In *Int. Conf. Distrib. Comput. Netw.*, pp. 14–28, 2014.
[141] N. H. Vaidya, L. Tseng, and G. Liang, "Iterative approximate byzantine consensus in arbitrary directed graphs," in *Proc. ACM Symp. Principles Distrib. Comput.*, 2012, pp. 365–374, 2012.
[142] D. V. Aken, A. Pavlo, G. J. Gordon, and B. Zhang, "Automatic database management system tuning through large-scale machine learning," in *Proc. 2017 ACM Int. Conf. Manag. Data*, 2017, pp. 1009–1024.
[143] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, "Efficient formal safety analysis of neural networks," in *Proc. Adv. Neural Inform. Process. Syst.*, 2018, pp. 6367–6377.
[144] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, "Formal security analysis of neural networks using symbolic intervals," in *Proc. 27th* {USENIX} *Secur. Symp. ({USENIX} Secur. 18)*, 2018. pp. 1599–1614.
[145] X. Wang, S. Mou, and B. D. O. Anderson, "Scalable, distributed algorithms for solving linear equations via double-layered networks," *IEEE Trans. Autom. Control*, vol. 65, no. 3, pp. 1132–1143, Mar. 2020.
[146] X. Wang, S. Mou, and D. Sun, "Improvement of a distributed algorithm for solving linear equations," *IEEE Trans. Ind. Electronics*, vol. 64, no. 4, pp. 3113–3117, Apr. 2017.
[147] X. Wang, S. Mou, and S. Sundaram, "A resilient convex combination for consensus-based distributed algorithms," *Numer. Algebra, Control Optimization*, vol. 9, no. 3, pp. 269–281, 2019.
[148] X. Wang, J. Zhou, S. Mou, and M. J. Corless, "A distributed algorithm for least square solutions," *IEEE Trans. Autom. Control*, vol. 64, no. 10, pp. 4217–4222, Oct. 2019.
[149] A. Wilke *et al.*, "The mg-rast metagenomics database and portal in 2015," *Nucleic Acids Research*, 44(D1):D590–D594, 2015.
[150] X. Liu, D. Yang, and A. El Gamal, "Deep neural network architectures for modulation classification," in *Proc. Asilomar Conf. Signals, Syst., and Comput.*, 2017.
[151] Y. Xiang, T. Lan, V. Aggarwal, and Y.-F. R. Chen, "Joint latency and cost optimization for erasure-coded data center storage," *IEEE/ACM Trans. Netw. (TON)*, vol. 24, no. 4, pp. 2443–2457, 2016.
[152] R. Xu, J. Koo, R. Kumar, P. Bai, S. Mitra, S. Misailovic, and S. Bagchi, "Videochef: Efficient approximation for streaming video processing pipelines," in *USENIX Annu. Tech. Conf. (USENIX ATC)*, 2018, pp. 43–56, 2018.
[153] T. Xu, L. M. Botelho, and F. X. Lin, "Vstore: A data store for analytics on large videos," in *Proc. Fourteenth EuroSys Conf. 2019*, pp. 16. ACM, 2019.
[154] M. Ye and E. Abbe, "Communication-computation efficient gradient coding," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5606–5615.
[155] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems," in *Proc. 33rd ACM/IEEE Int. Conf. Automated Softw. Eng.*, 2018, pp. 132–142.
[156] X. Zhang, T. Seyfi, S. Ju, S. Ramjee, A. El Gamal, and Y. C. Eldar, "Deep learning for interference identification: Band, training SNR, and sample selection," in *Proc. IEEE Int. Workshop Signal Process. Advances Wireless Commun.*, 2019.
[157] H. Zhuang, C. Wang, F. Tao, L. M. Kaplan, and J. Han, "Identifying semantically deviating outlier documents," in *Proc. 2017 Conf. Empirical Methods Natural Lang. Process., EMNLP 2017, Copenhagen, Denmark, Sep. 9-11, 2017*, 2017, pp. 2748–2757.

SAURABH BAGCHI received the B.S. degree from the Indian Institute of Technology Kharagpur, Kharagpur, India, and the M.S. and Ph.D. degrees from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, all in computer science.

He is currently a Professor with the School of Electrical and Computer Engineering and the Department of Computer Science, Purdue University, West Lafayette, IN, USA. He is the founding Director of a university-wide resilience center at Purdue called CRISP (since 2017) and a co-lead on the WHIN center started 2018. His research interests are in dependable computing and distributed systems. He is proudest of the 21 Ph.D. students and 50 masters thesis students who have graduated from his research group and who are in various stages of building wonderful careers in industry or academia. In his group, he and his students have way too much fun building and breaking real systems.

VANEET AGGARWAL received the B.Tech. degree from the Indian Institute of Technology Kanpur, Kanpur, India, in 2005, and the M.A. and Ph.D. degrees from Princeton University, Princeton, NJ, USA, in 2007 and 2010, respectively, all in electrical engineering.

He is currently an Associate Professor with Purdue University, West Lafayette, IN, USA, where he has been since January 2015. He was a Senior Member of Technical Staff Research with AT&T Labs-Research, NJ (2010–2014), Adjunct Assistant Professor with Columbia University, NY (2013–2014), and VAJRA Adjunct Professor with IISc Bangalore (2018–2019). His current research interests are in communications and networking, cloud computing, and machine learning.

Dr. Aggarwal received Princeton University's Porter Ogden Jacobus Honorific Fellowship in 2009, the AT&T Vice President Excellence Award in 2012, the AT&T Senior Vice President Excellence Award in 2014, the 2017 Jack Neubauer Memorial Award recognizing the Best Systems Paper published in the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, and the 2018 Infocom Workshop HotPOST Best Paper Award. He is on the Editorial Board for the IEEE TRANSACTIONS ON COMMUNICATIONS, the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, and the IEEE/ACM TRANSACTIONS ON NETWORKING.

SOMALI CHATERJI received the Ph.D. degree in biomedical engineering from Purdue University, West Lafayette, IN, USA, winning the Chorafas International Award, College of Engineering Best Dissertation Award, and the Future Faculty Fellowship Award. She did her Postdoctoral Fellowship with the Department of Biomedical Engineering, University of Texas at Austin, where her work was supported by an American Heart Association award.

She is currently an Assistant Professor with the Department of Agricultural and Biological Engineering, Purdue University, where she specializes in developing algorithms and statistical models for genome engineering, precision health, and IoT. She followed this up with a Postdoctoral stint at Purdue Computer Science when she got her first NIH R01 on computational metagenomics. She is a technology commercialization enthusiast and has been consulting for the IC2 Institute at the University of Texas at Austin, since Spring 2014.

FRED DOUGLIS (Fellow, IEEE) received the B.S. degree from Yale University, New Haven, CT, USA, and the M.S. and Ph.D. degrees from U.C. Berkeley, Berkeley, CA, USA, all in computer science.

He is a Chief Research Scientist with Perspecta Labs, where he works on applied research in the areas of high-performance computing, network optimization, blockchain, and security. He is a member of the IEEE Computer Society Board of Governors. He served as the Editor-in-Chief for the IEEE INTERNET COMPUTING from 2007 to 2010 and has been on its editorial board since 1999. He is also on the editorial boards for the IEEE TRANSACTIONS ON COMPUTERS and *ACM Transactions on Storage*. He formed the IEEE-CS Technical Committee on the Internet, chairing it from 1997 to 2000, and previously chaired the TC on Operating Systems from 1996 to 1998.

ALY EL GAMAL received the B.S. degree in computer engineering from Cairo University, Giza, Egypt, in 2007, the M.S. degree in electrical engineering from Nile University, Giza, Egypt, in 2009, and the M.S. degree in mathematics and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 2013 and 2014, respectively.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. His research interests include information theory and machine learning.

Dr. El Gamal has received a number of awards, including the Purdue Seed for Success Award, the Purdue CNSIP Area Seminal Paper Award, the Purdue Engineering Outstanding Teaching Award, the DARPA Spectrum Collaboration Challenge (SC2) Contract Award and Preliminary Events 1 and 2 Team Awards, and the Huawei Innovation Research Program (HIRP) OPEN Award. He is currently serving as an Associate Editor in the area of Machine Learning and AI for Wireless at the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, and as a reviewer for the American Mathematical Society (AMS) Mathematical Reviews.

JIAWEI HAN (Fellow, IEEE) received the B.S. degree from the University of Science and Technology of China, and the Ph.D. degree from the University of Wisconsin-Madison, all in computer science. He is currently Michael Aiken Chair Professor with the Department of Computer Science, University of Illinois at Urbana-Champaign, Champaign, IL, USA. He received ACM SIGKDD Innovation Award (2004), IEEE Computer Society Technical Achievement Award (2005), IEEE Computer Society W. Wallace McDowell Award (2009), and Japan's Funai Achievement Award (2018). He is a fellow of ACM and served as the Co-Director of KnowEnG, a Center of Excellence in Big Data Computing (2014–2019), funded by NIH Big Data to Knowledge (BD2K) Initiative and as the Director of Information Network Academic Research Center (INARC) (2009–2016) supported by the Network Science-Collaborative Technology Alliance (NS-CTA) program of U.S. Army Research Lab.

BRIAN J. HENZ received the B.S. degree in manufacturing engineering from St. Cloud State University, St. Cloud, MN, USA, the M.S. degree in mechanical engineering from the University of Minnesota, Minneapolis, MN, USA, and the Ph.D. degree from the University of Maryland, College Park, MD, USA.

He is a Research Scientist for the U.S. Army Research Laboratory (ARL) in the Computational and Information Sciences Directorate at Aberdeen Proving Ground, Maryland. He is currently serving as a Science and Technology Advisor for the Network CFT (Cross Functional Team). His research interests are in the development of physics-based models of U.S. Army systems of interest, and distributed computational methods. He has published papers on a wide range of computational science efforts including the development of scalable multi-scale/multi-physics computational methods for manufacturing processes, reactive atomistic modeling of metal nanoparticles and multi-scale mobile ad-hoc network simulation/emulation. Prior to joining the Network CFT, he was the ARL Senior Campaign Scientist for Computational Sciences and co-led the development of the AI/ML Essential Research Program.

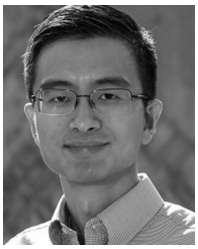
HENRY (HANK) HOFFMANN received the B.S. degree in mathematical sciences from the University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, and the S.M. and Ph.D. degrees in electrical engineering and computer science from MIT, Cambridge, MA, USA.

He is currently an Associate Professor with the Department of Computer Science, The University of Chicago, Chicago, IL, USA. He received the PECASE award in 2019 and a DOE Early Career award in 2015. His research explores the design and implementation of self-aware and adaptive computing systems.

SUMAN JANA is currently an Associate Professor with the Department of Computer Science and the Data Science Institute, Columbia University, New York, NY, USA. His primary research interests are at the intersection of computer security and machine learning. His research has received six best paper awards, a Google faculty fellowship, an NSF CAREER award, and an ARO young investigator award.

MILIND KULKARNI (Senior Member, IEEE) received the bachelor's degree from North Carolina State University, Raleigh, NC, USA, and the Ph.D. degree from Cornell University, Ithaca, NY, USA.

He is currently an Associate Professor with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA, where he is a University Faculty Scholar and a member of the Purdue Center for Programming Principles and Software Systems (PurPL). His research interests focus on developing languages, compilers and systems that can efficiently and effectively exploit locality and parallelism in complex applications on complex computation platforms. He received the NSF CAREER award in 2012 and the Department of Energy Early Career Research Award in 2013 for his work on optimizing irregular applications. He was awarded the Presidential Early Career Award for Scientists and Engineers in 2016. He has received numerous departmental, college, and university awards recognizing his teaching. He is a member of the ACM.



FELIX XIAOZHU LIN received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, and the Ph.D. degree in CS from Rice University, Houston, TX, USA.

He is currently an Assistant Professor with the Department of ECE, Purdue University, West Lafayette, IN, USA. At Purdue, he leads the Xroads systems exploration lab (XSEL) to accelerate and safeguard important computing scenarios. He and his students measure and build systems software with diverse techniques, including novel OS structures, kernel subsystem design, binary translation, and user-level runtimes. See http://felixlin.org for more information. He was the recipient of ASPLOS best paper award (2014), NSF CRII award (2015), Google Faculty Award (2016), and NSF CAREER award (2019).

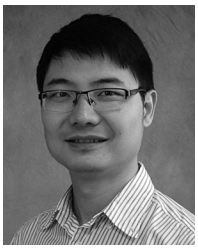
KAREN MARAIS received the B.Eng. degree in electrical and electronic engineering from the University of Stellenbosch, Stellenbosch, South Africa, the B.Sc. degree in mathematics from the University of South Africa, Pretoria, South Africa, the master's degree in space-based radar from MIT, Cambridge, MA, USA, and the Ph.D. degree from the Department of Aeronautics and Astronautics, MIT, Cambridge, MA, USA, in 2005. Prior to graduate school, she worked in South Africa as an Electronic Engineer.

She is currently an Associate Professor with the School of Aeronautical and Astronautical Engineering, Purdue University, West Lafayette, IN, USA. Her research interests focus on developing ways to deliver value through reliability, safety, and sustainability. She is currently working to learn how we can design and operate complex systems to work with human nature, and on data-driven approaches for improving aviation safety. She is the author or co-author of several technical publications, including 20 journal papers, two reports for the National Academies, and two book chapters. She received an NSF CAREER award in 2014.

PRATEEK MITTAL is currently an Associate Professor of Electrical Engineering with Princeton University, Princeton, NJ, USA, where he is also affiliated with Computer Science and the Center for Information Technology Policy. He is interested in the design and development of privacy-preserving and secure systems. A unifying theme in his research is to manipulate and exploit structural properties of data and networked systems to solve privacy and security challenges facing our society. His research has applied this distinct approach to widely-used operational systems, and has used the resulting insights to influence system design and operation, including that of the Tor network and the Lets Encrypt certificate authority, directly impacting hundreds of millions of users. He is the recipient of Princeton Universitys E. Lawrence Keyes, Jr. award for outstanding research and teaching, the NSF CAREER award, the ONR YIP award, the ARO YIP award, faculty research awards from IBM, Intel, Google, Cisco, Facebook, and multiple award publications.

SHAOSHUAI MOU received the Ph.D. degree from Yale University, New Haven, CT, USA, in 2014. After working as a Postdoctoral Associate with MIT, he joined Purdue University, West Lafayette, IN, USA as an Assistant Professor with the School of Aeronautics and Astronautics in 2015.

His research interests include distributed algorithms for control/optimizations/learning, multi-agent systems, UAV collaborations, perception and autonomy, resilience and cyber-security.



XIAOKANG QIU received the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 2013.

He is currently an Assistant Professor of Electrical and Computer Engineering with Purdue University, West Lafayette, IN, USA. Before starting with Purdue in 2016, he was a Postdoctoral Associate with the Massachusetts Institute of Technology. His current research interests lie in the algorithmic aspects of software verification and synthesis, and their applications to networking and accelerator programming. He is a member of the Purdue Center for Programming Principles and Software Systems (PurPL) and leads the Computer-Aided Programming (CAP) group at Purdue.

GESUALDO SCUTARI received the Electrical Engineering and Ph.D. degrees (both with Hons.) from the University of Rome La Sapienza, Rome, Italy, in 2001 and 2005, respectively.

He is the Thomas and Jane Schmidt Rising Star Associate Professor with the School of Industrial Engineering, Purdue University, West Lafayette, IN, USA. His research interests include continuous and distributed optimization, equilibrium programming, and their applications. Among others, he was the recipient of the 2013 NSF CAREER Award, the 2015 Anna Maria Molteni Award for Mathematics and Physics, and the 2015 IEEE Signal Processing Society Young Author Best Paper Award. He serves on the Editorial Board for the *SIAM Journal on Optimization* and the IEEE TRANSACTIONS ON SIGNAL PROCESSING (Senior Area Editor).