AGENP: An ASGrammar-based GENerative Policy Framework

Seraphin Calo*, Irene Manotas*, Dinesh Verma*, Elisa Bertino†, Mark Law‡, Alessandra Russo‡
*IBM Research
Yorktown Heights, NY
scalo@us.ibm.com, irene.manotas@ibm.com, dverma@us.ibm.com
†Purdue University
W. Lafayette, IN, USA
bertino@purdue.edu
‡Imperial College London
South Kensington Campus, London
mark.law09@imperial.ac.uk, a.russo@imperial.ac.uk

***Abstract*—Generative policies have been proposed as a mechanism to autonomously adapt to changes in a system to achieve the systems's goals with minimum to none human intervention. The combination of generative policies and learning mechanisms can help a coalition system to be more effective when working in a distributed, continuously transforming environment with a diverse set of members, resources, and tasks. Learning mechanisms based on logic e.g., Inductive Logic Programming (ILP), have several properties that make them suitable and attractive for the creation and adaptation of generative policies. Some of the ILP features include representation of the data and problems in a declarative way, the ability to learn a general model with a small number of examples, and with an existing background knowledge. Recently, Answer Set Programming (ASP) have been recognized as a powerful language for knowledge representation and reasoning for which ILP has been successfully applied to.**

This paper proposes AGENP, a ASGrammar-based GENerative Policy Framework for Autonomous Managed Systems (AMS) that aims to support the creation and evolution of generative policies by leveraging ILP. We describe the framework components, the required inputs, data structures, and mechanisms to support the refinement and instantiation of policies, identification of policy violations, and monitoring of policies and policy adaptation according to changes in the AMS and its context. We also present the main workflow for the global and local refinement of policies and their adaptation based on ASP and ILP concepts and tools, and present example scenarios where the AGENP framework can be useful for the self-generation of policies.

***Keywords*-policy-based management systems; generative policies; inductive logic programming**

I. Introduction

With the recent advances in Artificial Intelligence (AI), Autonomous Systems (AS) composed of intelligent, self-managed devices are on the rise [5, 15]. Traditional policy management tools need to evolve in order to be able to manage a myriad of complex situations where autonomous devices are operating, in isolation or collaboratively, to ensure that devices and systems take secure and appropriate decisions. Enabling policies to be autonomously adapted and created according to dynamic contexts, where resources availability and conditions change over time, could help autonomous managed systems to take timely decisions, while maintaining consistency and compliance with the system objectives.

Policy-based Management Systems (PBMSs) usually work with predefined policies that are provided by admin users, or experts on the system's operations, to ensure the proper system functionality. Policies definitions are usually not modified unless a human analyze them and pinpoint corrections or improvements, and new policies are only provided to the system by end users when new business objectives are identified. This manual approach for policy creation and modification is tedious and error prone [12]. Hence, approaches for guided and automatic policy generation have been proposed [12, 13, 17]. However, these approaches are either focused on static policies (i.e., predefined policies that do not change over time) or they rely on large amounts of data to be able to identify new policies, or required policies' transformations. To the best of our knowledge, there has not been approaches that try to learn a policy model to self-generate policies that capture common and optimal policy decisions of a managed system.

The Generative Policy-based Model (GPM), recently proposed in [14], proposes the concept of *generative policies* i.e., policies that are self-generated by the managed system (e.g., devices) to enable systems to determine their own behavior, allowing flexible policy definitions that make managed systems more autonomous. A generative policy model could be learned from an initial specification of policies of the managed system, and from the analysis of the system's operations, resource usage, and previous policy decisions. One learning approach that can be used for learning generative policy models is by leveraging Inductive Logic Programming (ILP) techniques. ILP is a learning approach that have been successful at learning representative models (e.g., programs) from examples and background knowledge [6]. ILP is an appealing symbolic learning ap-

proach because, contrary to other popular supervised learning approaches, it is a white-box or knowledge oriented method that provides easy to understand models [8], and does not require large amounts of labeled data to generate a good model of the underlying system's knowledge. This paper presents the ASGrammar-based GENerative Policy Framework (AGENP), a PBMS framework based on GPM and that is designed to learn a generative policy model via ILP with the goal of supporting autonomous management systems in self-generating and selecting optimal policies based on system's operation and history of policy decisions.

II. Generative Policy Scenarios

Various scenarios involving Autonomous Managed System could take advantage of the generative policies concept and a framework instantiation like AGENP. This section highlights some scenarios that we think would benefit from instantiations of the GPM in general.

A. Policies for Federated Learning Environments

Federated systems, i.e., systems where multiple organizations work together to share resources and collaborate to achieve common tasks, could benefit from using generative policies to manage its operations as shown in [1]. In the same way, a Federated Learning (FL) system, where the main objective is to learn a common Machine Learning (ML) model, requires the definition and management of policies for distributed learning and data/model fusion. These systems can use generative policies to manage and ensure the right operations of the FL system are in place. FL-related policies need to consider different characteristics of the FL system and its environment [16], such as the quality of the data and communications, the data arrangements among the different nodes, the type of ML algorithm, and the performance of different participating nodes, to decide how to distribute the data and learn a federated ML model. Given the wide diversity of learning environments and algorithms that can exist in FL systems, a framework like AGENP that helps to self-generate policies could provide great advantages for the proper functionality of the FL system by identifying over time which policies are optimal, and how policies can evolve to handle operations between different Autonomous Managed System (AMS) working together.

B. Policies for Autonomous Vehicles Operations

Autonomous transportation and systems for autonomous vehicles are becoming more common than before [4]. Policies that help to improve the user experience, i.e., focusing on personalization and context awareness by providing the ability for the vehicle to adapt to the behavior and desires of the users in a safe and secure way, are an important aspect of autonomous vehicles. Generative policies in this context can help autonomous vehicles to identify which personalization policies, based on the vehicle context information and user preferences, should be used during the vehicle operation based on the user, the vehicle status, and its environment. Also, generative policies can be used to pinpoint policies that define how more complex autonomous systems interact, for instance for situations where multiple autonomous vehicles are operating at the same time, interacting with each other.

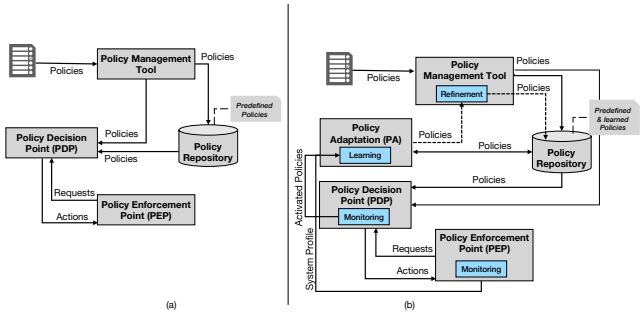

Figure 1. Policy-based Management Systems (PMS): (a) Standard PMS, (b) Evolved PMS.

III. Evolution of Policy-based Management Systems

A. Standard Policy-based Management Systems

Fig. 1 (a), shows a standard PBMS, such as the one defined by IETF/DMTF[1], where policy-based management tools create, edit, and control policies stored in a persistent policy repository. A Policy Decision Point (PDP) interprets policies in the policy repository and sends decisions to a Policy Enforcement Point (PEP) using an appropriate format. Then, the PEP executes an action, or set of actions, according to the decisions obtained from the PDP. Policies used in these systems are usually provided by administrator users or user experts on the system's functionality. In these PBMS, changes to policies do not occur unless an admin user analyzes the current policies and identifies those that need to be corrected or improved, and new policies are provided by a domain expert to the managed system when new business goals are identified.

B. Evolved Policy-based Management Systems

Changes in the internal and external conditions of the managed system can create scenarios where initial predefined policies cannot be used because such situations represent events and conditions not foreseen initially in the policies provided to the PBMS. Learning from the history of actions taken by the managed system, analyzing policies' decisions and policies' structure (i.e., event, conditions, actions), and improving policy definitions, would make the managed system to be prepared to make decisions for situations not initially anticipated, thus enabling its autonomous operation. Contrary to the standard policy-based

[1]https://tools.ietf.org/html/rfc2753

management system, the evolved policy-based management system includes support for *policy refinement* and *policy learning* to allow learning from past experiences, and from the continuous evolution of policies. Figure 1 (b) shows an overview of the evolved policy-based management system that includes the policy refinement and policy learning processes described in general below and in detail in Section V and Section VI.

Policy Refinement: The refinement process allows to find the best policy representation for policies used by the AMS. The policy refinement process continuously consolidate the set of policies, and their representations, according to the policy constraints initially stablished for the system, and based on decisions made by the managed system during its operation.

Policy Learning: Learning from previous policy decisions allows the managed system to identify new policies that can be used to handle unforeseen situations while complying with the system objectives and constraints, and without waiting for policies to be discovered and provided by an end user. The policy learning process learns a policy model that considers the refined policies, the decisions made by the system in the past, and the system context (e.g., system status and environment). The policy model is used to instantiate new policies that are predicted to be useful for the immediate and future system's operations. Section V describes in detail the structure of the GPM that is designed to support operations of autonomous managed systems, and Section VI describes the AGENP framework, which is an instantiation of the Generative Policy-based Model (GPM) using ILP as the learning approach.

IV. Learning Policy Models

We define the process of identifying the general policy characteristics i.e., objects, conditions, and actions, and their associated values, from a set of policies, and for a given policy domain type (e.g., access-control policy or network policy), as learning a policy model. A policy model can be represented as a grammar, or policy template, that can be used to instantiate policies for a given domain of an AMS. We propose inductive learning as a policy learning approach to capture a policy model for a set of policies, that can be used to refine, instantiate, and adapt policies according to the operations of a AMS.

A. Inductive Logic Programming (ILP)

In ILP the main task is to find an hypothesis, that together with given background knowledge, explains a set of observations (i.e., positive examples). ILP is appealing for learning tasks mainly because its ability to learn a general model with a small number of examples and with an existing background knowledge [9], and also because the learned model (or hypotheses) can be expressed in plan english making them easier for a human user to understand [11].

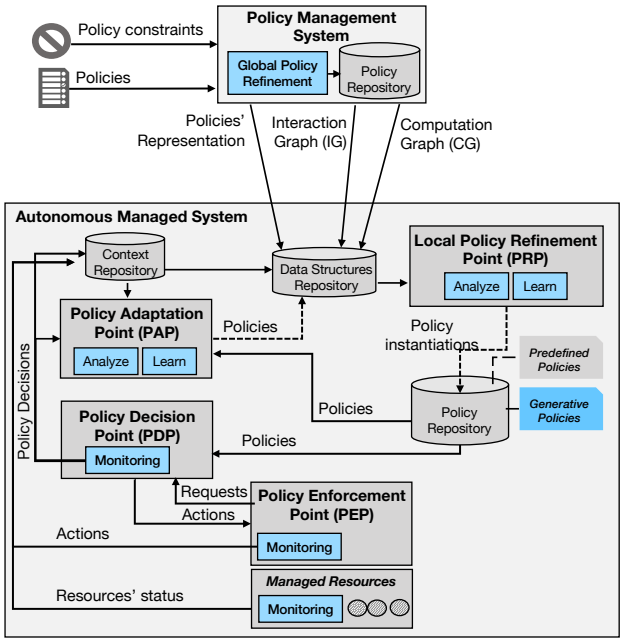

Figure 2. Overview of the Generative Policy Model Architecture

Answer Set Program (ASP) is a declarative programming paradigm based on logic programming, under answer set semantics, that have been used extensively in reasoning applications, and to support the development of inference engines. Recently, an ILP approach for ASP was proposed [9], which allows to learn inductive solutions as ASP. We propose using ASP, and Answer Set Grammar (ASG) as a low-level representation of policies, because ASP is a highly declarative paradigm for knowledge representation and reasoning, and it allows to specify policy constraints at the attribute level, enabling more flexible and dynamic policy specifications such as the ones required by the generative policy definition [3].

Next section presents a review of the GPM, and Section VI describes the design of the ASGrammar-based GENerative Policy Framework (AGENP) that uses ASP and ILP techniques to learn a generative policy model to instantiate policies according to the GPM model.

V. The Generative Policy Model (GPM)

This section presents a more detailed description of the GPM [14], including inputs and outputs of the model, and the components located in the management tool and in the managed system. Figure 2 shows an overview of the components for the policy-based management of autonomous managed systems in the generative policy model architecture.

A. Inputs and Outputs

The GPM takes as inputs *policy definitions* and *policy constraints*. Policy definitions are provided by admin users

to reflect the correct actions to take for different conditions of the managed system's operation. Policy constraints allow the AMS to establish bounds within which policies can be defined. Policy boundaries can be seen as regions of the search space of policy instantiations that are should never be explored. Policy constraints serve as a mechanism to bound the size of the policy search space. The outputs consist of the policy representations for the policies that have been found to be optimal given the conditions of the AMS operation, and the policy violations that have been found during the analysis and learning of policy decisions.

B. GPM Components

Beyond the traditional components of a PBMS, which include a PDP, a PEP, and a policy repository, the GPM follows the general architecture of the evolved PBMS described in Section III-B. The GPM includes two new components, a Policy Adaptation Point (PAP) at the AMS, and a Policy Refinement Point (PRP) at the management tool and at the AMS. Also, two additional repositories are defined for the AMS, the context information repository and the data structures repository, which are described below along with the new components.

1) Global Policy Refinement: Policy refinement refers to the process of interpreting the policy grammar, make modifications accordingly, e.g., to correct possible violations or inconsistencies, and generate policies according to the revised (modified) grammar. Policy refinement is done at the global level, i.e., in the management system, and at the local level, inside the autonomous managed system, in a Policy Refinement Point (PRP) as shown in Fig. 2. Global policy refinement allows the policy management system to consider policies that are within the prescribed bounds, and to obtain the right representation for the policies (i.e., grammar or template). In the case that the management system is collaborative i.e., multiple managed systems agree to collaborate to perform tasks together, then global refinement is in charge of defining the Interaction Graph (IG) and the Computation Graph (CG) that specify the roles and tasks that can be assigned to the different resources of the managed systems involved in the collaborative scenario.

2) Data Structures Repository: The data structures repository stores the policy representations (grammar), policy constraints, and interaction graph, along with their versions i.e., initial version specified by the admin user, and the subsequent versions generated by the AMS during its operation.

3) Context Information Repository: The context information repository holds information about the status of the management system. The monitoring of decisions in the PDP and the status of resources in the PEP is stored in this repository for future reference. The information keep in the context repository helps in the refinement and adaptation of policies by storing information that describes the system state and environment.

4) Policy Adaptation Point (PAP): The PAP component is in charge of analyzing the history of policies' decisions, and adapting the generative policy model so that it is representative of the types of policies used in the managed system. The PAP updates the policy representation (e.g., policy grammar or template) to include aspects of new policies that are considered to be useful to the managed system based on the analysis of policy decisions and context. Context information is obtained either from the monitoring of PDP and PEP, or from the context information repository.

Policy Representation Analyzer: This subcomponent of the PAP performs analytics over the policy decisions and resource management to inform the policy learner about conditions and actions that need to be considered to adapt the current policy representation. This component ensures that the policies being generated satisfy policy metrics (e.g., response time, confidence).

Policy Representation Learner: By using the information from the policy analyzer, the policy learner modifies the current policy representation to incorporate information that allow to instantiate policies that comply with the business objectives of the managed system as well as new policies that allow the managed system to behave autonomously.

5) Policy Refinement Point (PRP): At the local level, the refinement of policies inside the PRP uses the information from the global refinement, the context information repository, the policy repository, and the PAP to create a generative policy model that allows to instantiate policies that can be used by the managed system.

Policy Model Analyzer: This subcomponent of the PRP performs analytics over the policy representation obtained from the PAP, the context information, the current policy representation, and the data structures repository to identify the set of policy representations that can be used by the learner to create or update the generative policy model.

Policy Model Learner: With the results from the policy analyzer, the policy model learner creates or modifies the current generative policy model to incorporate information that enables the instantiation of generative policies. The policy model learner provides to the policy model analyzer the latest policy representation derived for the generative policy model.

VI. The ASGrammar-based GENerative Policy Framework (AGENP)

Based on the design of the GPM, we designed the ASGrammar-based GENerative Policy Framework instantiation that uses ILP mechanisms to learn the policy representations and the generative policy model to enable the self-generation of policies on AMSs. Following the same structure of the GPM, in Figure 3 we show an overview of the AGENP framework components using inductive learning mechanisms inside the PAP and PRP components, which are described in detail below.

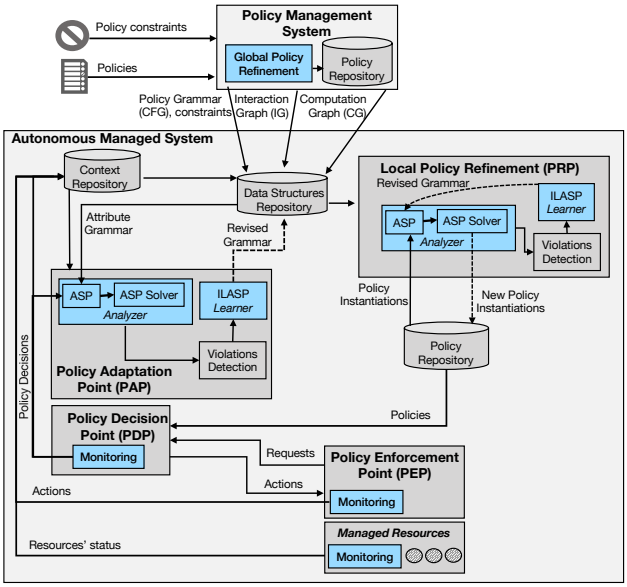

Figure 3. Overview of the ASGrammar-based GENerative Policy Framework (AGENP)

A. Inputs and Outputs

AGENP uses the policies as Context Free Grammar (CFG), the policy constraints, the IG and the CG as input to the AMS. The outputs include an attribute grammar and ASP that represent the policies for the AMS at a given point in time. The next section explains how this constructs are created inside the AGENP instantiation.

B. AGENP's Components Details

Here we describe the details of the GPM components that are instantiated in a specific way in the AGENP framework. Components that are not detailed here follow the same structure as it was defined in Section V.

Data Structures Repository: In the AGENP framework, this repository keeps track of the interaction graph, the policy grammars (CFG, and attribute grammars), and the policy constraints.

Policy Refinement Point (PRP): The analyzer in the PRP uses the information contained in the data structures repository to get the policy grammar needed by the Answer Set Program (ASP). If there does not exist an attribute grammar in the data structures repository, then the PRP first transforms the CFG to an attribute grammar. Then, the attribute grammar is translated to an ASG in the ASP subcomponent, using information from examples of policy instantiations, obtained from the context and policy repositories. Policy examples could also be acquired from an external trusted WikiHow-like repository containing policies from various related domains and scenarios as the Contextualised ASset Wiki (CASWiki) [2]. The ASP solver takes the ASG information and using a search algorithm finds the set of policy solutions and policy violations. Policy solutions are passed to an inductive learner for ASP, such as the Inductive Learning of Answer Set Programs (ILASP) [9], which both learns the ASP rules in the ASG and finds a representative model of the generative policies that can be used to instantiate policies, i.e., the 'revised' ASG. The generated 'revised' ASG is shared back to the ASP subcomponent. Once there has not been changes between consecutive versions of the grammar being generated the 'revised' attribute grammar is stored in the data structures repository. Policy instantiations, created from the ASG, are saved in the policy repository so that the AMS can use them for its operations. For a formal specification of how ASP can be learn, the reader is referred to [9].

Policy Adaptation Point (PAP): The policy adaptation process uses the latest attribute grammar from the policy data repository. History of policy decisions and context information, such as resources' status and availability, are used as examples and background knowledge by the analyzer, where positive examples can be policies with high quality (i.e., policies that are consistent, complete, minimal, relevant, and correct [7]). The attribute grammar is converted into an ASG in the ASP component and then it is passed to the ASP Solver to find the optimal set of policy solutions along with policy violations. Policy solutions are then passed to a learner for ASP, such as the ILASP2 learner [10], that is able to learn the constraints, in the form of ASP, for the generative policy model. Also, the learner returns a preference ordering for instances of the generative policy model. Both the preference ordering and the revised grammar are stored in the data structures repository.

VII. Related Work

Autonomic management is an area where policies are used to automatically manage a system under predefined conditions. In this area tools like Madison [12], a library for automatic policy generation for SELinux policies, have been developed to provide modules that guide the policy definition process, and modules that extract information from reference policy interface files to allow the creation of policy templates from system's accesses. However, to the best of our knowledge, none of the current autonomic management approaches for consider the evolution of policies according to the commons decisions made by the managed system. The closest work related to ours is the one done by Quiroz et al. that presents a framework for autonomic policy adaptation based on online analysis and characterization of system operation based on decentralized clustering techniques [13]. Our approach is different to theirs in that we use symbolic learning to capture common and optimal policies based on the system operation without relying on large amounts of data but on a small set of examples and background knowledge about the managed system policies, its components, and operation. Jabal et al. presents a review

of policy analysis approaches and tools. Policy examples for access control (e.g., role-based access control policies), and network (e.g., firewall policies, SDN policies) are presented, as well as the policy analysis process, and the frameworks and tools available [7]. As we have presented in this paper, different policy types and analysis tools can be leveraged by other instantiations of the GPM, and in the AGENP framework to analyze and select optimal polices' decisions for different AMS and scenarios.

VIII. Conclusions

We have presented the design of AGENP, a generative policy framework for autonomous managed systems based on ILP and ASP. AGENP enables the reasoning, refinement, and learning of policies autonomously using ILP as the mechanism to learn a generative policy model from a set of examples and background knowledge from the managed system. In AGENP, policies are represented as ASP to allow to express generated policies in a way easy to understand to human users, and facilitate the learning of policies' attributes and constraints via ILP. We described how existing algorithms and tools for learning ASP solutions can be leveraged in the AGENP's implementation to learn generative policies for autonomous managed systems. We presented two representative scenarios, including federated learning and autonomous vehicles, where generative policies and the AGENP instantiation can be useful. Future work includes empirical evaluation of the AGENP instantiation for different scenarios where generative policy generation can help a system to decide more autonomously about optimal policies, and to provide policies for unanticipated situations.

References

[1] E. Bertino, S. Calo, M. Toma, D. Verma, C. Williams, and B. Rivera. A cognitive policy framework for next-generation distributed federated systems: Concepts and research directions. In *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 1876–1886, 2017.

[2] E. Bertino, G. de Mel, A. Russo, S. Calo, and D. Verma. Community-based self generation of policies and processes for assets: Concepts and research directions. In *IEEE International Conference on Big Data*, pages 2961–2969, 2017.

[3] Seraphin Calo, Dinesh Verma, Supriyo Chakraborty, Elisa Bertino, Emil Lupu, and Gregory Cirincione. Self-generation of access control policies. In *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies, SACMAT*, pages 39–47, 2018.

[4] Riccardo Coppola and Maurizio Morisio. Connected car: Technologies, issues, future trends. *ACM Comput. Surv.*, 49(3):46:1–46:36, October 2016.

[5] C. L. Fok, M. Hanna, S. Gee, T. C. Au, P. Stone, C. Julien, and S. Vishwanath. A platform for evaluating autonomous intersection management policies. In *3rd International Conference on Cyber-Physical Systems*, pages 87–96, 2012.

[6] Sumit Gulwani, José Hernández-Orallo, Emanuel Kitzelmann, Stephen H. Muggleton, Ute Schmid, and Benjamin Zorn. Inductive programming meets the real world. *Commun. ACM*, 58(11):90–99, 2015.

[7] A. A. Jabal, M. Davari, E. Bertino, C. Makaya, S. Calo, D. Verma, A. Russo, and C. Williams. Methods and tools for policy analysis. *ACM Comput. Surv.*, 2018.

[8] Dimitar Kazakov and Daniel Kudenko. Mutli-agents systems and applications. chapter Machine Learning and Inductive Logic Programming for Multi-agent Systems, pages 246–270. Springer-Verlag New York, Inc., 2001.

[9] Mark Law, Alessandra Russo, and Krysia Broda. Inductive learning of answer set programs. In Eduardo Fermé and João Leite, editors, *Logics in Artificial Intelligence*, pages 311–325. Springer International Publishing, 2014.

[10] Mark Law, Alessandra Russo, and Krysia Broda. Learning weak constraints in answer set programming. *Theory and Practice of Logic Programming*, 15(4-5): 511–525, 2015.

[11] Mark Law, Alessandra Russo, and Krysia Broda. The complexity and generality of learning answer set programs. *Artificial Intelligence*, 259:110–146, 2018.

[12] K. Macmillan. Madison : A new approach to policy generation? In *SELnux Symposium*, 2007.

[13] Andres Quiroz, Manish Parashar, Nathan Gnanasambandam, and Naveen Sharma. Autonomic policy adaptation using decentralized online clustering. In *Proceedings of the 7th International Conference on Autonomic Computing*, ICAC, pages 151–160. ACM, 2010.

[14] D. Verma, S. Calo, S. Chakraborty, E. Bertino, C. Williams, J. Tucker, and B. Rivera. Generative policy model for autonomic management. In *IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation*, pages 1–6, 2017.

[15] Dinesh Verma, Graham Bent, and Ian Taylor. Learning neural network policies with guided policy search under unknown dynamics. In *Proceedings of the 9th International Conference on Advanced Cognitive Technologies and Applications*, COGNITIVE, 2017.

[16] Dinesh Verma, Seraphin Calo, and Greg Cirincione. Distributed AI and security issues in federated environments. In *Proceedings of the Workshop Program of the 19th International Conference on Distributed Computing and Networking*, Workshops ICDCN. ACM, 2018.

[17] Le Yu, Tao Zhang, Xiapu Luo, and Lei Xue. Autoppg: Towards automatic generation of privacy policy for android applications. In *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM, pages 39–50. ACM, 2015.