Received March 13, 2018, accepted April 18, 2018, date of publication April 30, 2018, date of current version June 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2831228

Multi-Agent Systems: A Survey

ALI DORRI[1,2], (Student Member, IEEE), SALIL S. KANHERE[1], (Senior Member, IEEE), AND RAJA JURDAK[2], (Senior Member, IEEE)
[1]School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia
[2]Data61, CSIRO, Brisbane, QLD 4069, Australia

Corresponding author: Ali Dorri (ali.dorri@unsw.edu.au)

The work of A. Dorri was supported by the Data61 Ph.D. Scholarship.

ABSTRACT Multi-agent systems (MASs) have received tremendous attention from scholars in different disciplines, including computer science and civil engineering, as a means to solve complex problems by subdividing them into smaller tasks. The individual tasks are allocated to autonomous entities, known as agents. Each agent decides on a proper action to solve the task using multiple inputs, e.g., history of actions, interactions with its neighboring agents, and its goal. The MAS has found multiple applications, including modeling complex systems, smart grids, and computer networks. Despite their wide applicability, there are still a number of challenges faced by MAS, including coordination between agents, security, and task allocation. This survey provides a comprehensive discussion of all aspects of MAS, starting from definitions, features, applications, challenges, and communications to evaluation. A classification on MAS applications and challenges is provided along with references for further studies. We expect this paper to serve as an insightful and comprehensive resource on the MAS for researchers and practitioners in the area.

INDEX TERMS Multi-agent systems, survey, MAS applications, challenges.

I. INTRODUCTION

In recent years, Distributed Artificial Intelligence (DAI) has received tremendous attention from academia due to its ability to address complex computing problems [1], [2]. DAI algorithms are classified into three categories, based on the fundamental methods used to solve the tasks, namely: parallel AI, Distributed Problem Solving (DPS), and Multi-Agent Systems (MAS). Parallel AI involves developing parallel algorithms, languages, and architectures for increasing the efficiency of classical AI algorithms by leveraging task parallelism [2]. DPS involves dividing a task into subtasks each of which is allocated to a node among a set of cooperative nodes, known as computing entities. Computing entities have shared knowledge or resources as well as predefined communications with other entities which in turn limits their flexibility [1].

Multi-Agent Systems (MAS), which are the main focus of this paper, consist of autonomous entities known as agents. Similarly to computing entities in DPS, agents collaboratively solve tasks yet they offer more flexibility due to their inherent ability to learn and make autonomous decisions. Agents use their interactions with neighbouring agents or with the environment to learn new contexts and actions. Subsequently, agents use their knowledge to decide and perform an action on the environment to solve their allocated task. It is this flexibility that makes MAS suited to solve problems in a variety of disciplines including computer science, civil engineering, and electrical engineering [3]. To develop MAS require addressing a diverse range of complex challenges such as coordination among agents [4], learning [5] and security [1].

Existing survey articles present a relatively narrow discussion on MAS, e.g., Zidan *et al.* [6] have focused on the fault detection challenge while Shamshirband *et al.* [3] elaborate on MAS applications in network security. However, researchers and practitioners (particularly those new to the area) need to have a broad understanding of MAS ranging from basic definitions, research challenges, application domains and evaluation methodologies. As agents can be applied in different disciplines, there is a need for a comprehensive field-agnostic resource to cover the aforementioned MAS concepts. By understanding the applications of agents in different disciplines, the reader will gain valuable insight into how they can put MAS into practice. Being aware of the open challenges is instructive to learn of potential pitfalls and limitations and also for identifying future research directions. A dive into the evaluation methodologies is necessary to appreciate the performance gains that can be achieved with MAS and the potential tradeoffs. Balaji and Srinivasan [7] discussed the salient features of MAS and outlined a limited number of challenges. However, they did not discuss

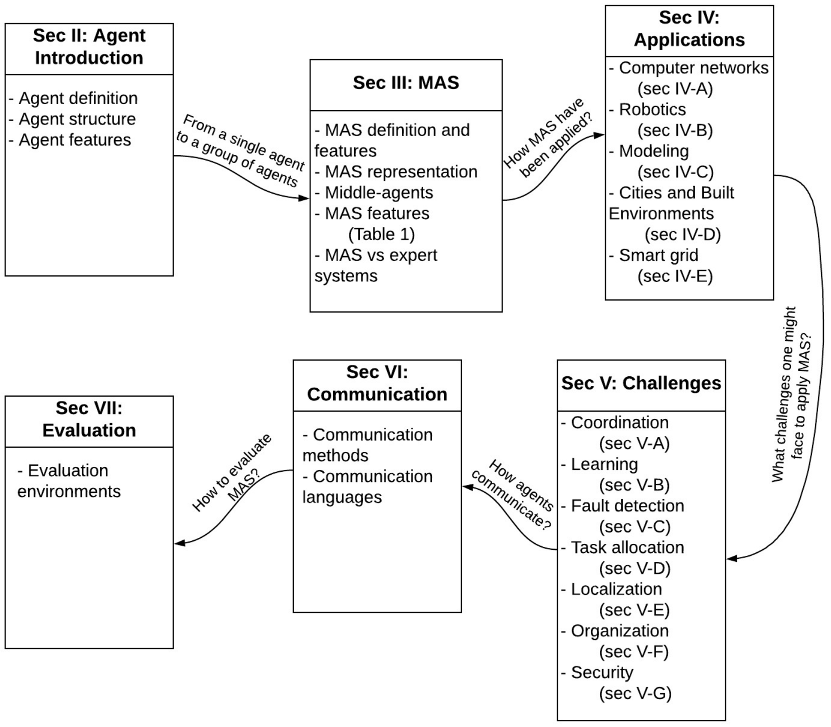


FIGURE 1. An overview of the paper.

applications, evaluations and a broader set of challenges including security and task allocation among others.

In this paper, we provide an all encompassing overview of MAS that will enable the reader to gain a thorough understanding of this broad discipline. To do so, we first define agents and outline their features to clearly differentiate MAS from other related concepts such as expert systems. Next, we present a taxonomy of the applications and challenges of MAS. Additionally, we discuss various approaches used to evaluate the performance of agent systems. Figure 1 depicts various topics covered in this paper section-by-section and also outlines the flow of ideas.

The rest of the paper is organized as follow: Section II provides an introduction to agents and their features. Section III introduces MAS and their salient features while Section IV covers MAS applications. Section V provides a comprehensive discussion on MAS challenges. Section VI discusses the key methods used for communications between different agents. Section VII provides a discussion on the methods for evaluating an agent system, and finally Section VIII concludes the paper.

II. AN INTRODUCTION TO AGENTS

In this section, we define agents and their key features. In the literature, there are multiple definitions for agents resulting from diverse application-specific features of agents. Russell *et al.* [8] defined an agent as "a flexible autonomous entity capable of perceiving the environment through the sensors connected to it." This definition has been corroborated by other researchers in the discipline, e.g., [9]. A different perspective was presented in [10], where the authors defined an agent as "an encapsulated computational system that is situated in some environment and this is capable of flexible, autonomous action in that environment in order to meet its design objective." These and most other definitions frame agents in the context of a specific field of study. However, the notion of agents is rather generic and can be broadly applied to many disciplines. Thus, we propose a generalized definition considering the fundamental abilities and features of agents:

Agent: An ***entity*** *which is placed in an* ***environment*** *and senses different* ***parameters*** *that are used to make a decision based on the goal of the entity. The entity performs the necessary* ***action*** *on the environment based on this decision.*

The above definition comprises four keywords which can be further elaborated:

1) *Entity:* Entity refers to the type of the agent. An agent can be a software, e.g. daemon security agents, a hardware component, e.g. thermostat, or a combination of both, e.g. a robot.
2) *Environment:* This refers to the place where the agent is located. The environment can be a network in the case of traffic monitoring agents, a software when the agent is monitoring the actions of software components, etc. An agent uses the information sensed from the environment for decision making. The environment has multiple features that affect the complexity of an agent-based system [1]:

- Accessibility: Accessibility refers to the accuracy with which agents can sense data from the environment. In an accessible environment, agents can sense accurate and up to date data from the environment. For instance, a network firewall agent may capture the entire traffic feed of a network. In contrast, in an inaccessible environment the agent senses noisy and/or incomplete data. Physical world, i.e., any event on earth, is an example of inaccessible environment as the sensors that capture the data introduce inherent noise which perturbs the observed (and thus measured) signal.
- Determinism: This refers to the predictability of the results of an action. In a deterministic environment the results are predictable, e.g. in an environment which can be modeled by finite state machine, the agent precisely knows the next state for each action. In non-deterministic environments, the outcome of an action is not entirely predictable as it may be influenced by other factors, e.g. in a game, the outcome depends on the actions of all participants.
- Dynamism: This refers to the changes that occur in the environment that are independent of the actions taken by the agents. An environment in which changes can only occur as a consequence of the action of the agents is considered to be static. In all other instances, an environment is considered to be dynamic. Recall that the decision making process of agents relies on the sensed information from the environment. When the environment changes, the previously sensed information may no longer be accurate. Thus, in a dynamic environment agents must detect a change in the state of the environment and update the sensed information which incurs more overhead compared to static environments where initial sensed information can be used for decision making during the lifetime of the agent.
- Continuity: This refers to the continuity or discreteness of the agent's environment. According to this feature, MAS environment is classified into two categorize: continuous and discrete. A continuous environment affects the agent's state through a continuous function, e.g., an agent that moves within a physical environment. A discrete environment constrains the agent to enter into a set of predetermined states, e.g. a mobile agent that measures its timestamped position as it moves through the environment.

3) *Parameters:* The different types of data that an agent can sense from the environment are referred to as parameters. For instance, the parameters for a soccer robot agent are the position and speed of the team members and opponents, and the position of the ball.

4) *Action:* Each agent can perform an action that results in some changes in the environment. For example, when a soccer robot kicks a ball the position of the ball changes. An agent can perform a set of discrete or continues actions. In a continues set of actions, the agent can perform unlimited actions, e.g. a soccer game. A discrete set of actions in contrast has a finite set of actions, e.g. an agent controlling a thermostat in a room.

The goal of each agent is to solve its allocated task with some additional constraints, e.g. a deadline. To achieve this aim, the agent first senses parameters from the environment. Empowered with this data, the agent can build up knowledge about the environment (discussed in Section V-B). An agent might also use the knowledge of its neighbors. This knowledge along with the history of the previous actions taken and the goal are fed to an inference engine which decides on the appropriate action to be taken by the agent. Figure 2 depicts the aforementioned steps.

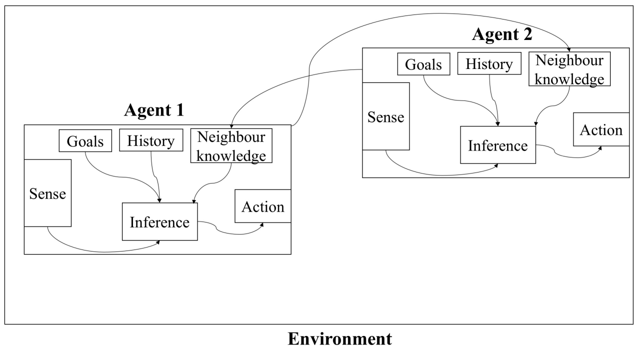


FIGURE 2. The structure of an agent.

The following features enable agents to have broad applicability and solve complex tasks [11], [12]:

- Sociability: Agents can share their knowledge as well as request information from other agents to improve performance in reaching their goals.
- Autonomy: Each agent can independently execute the decision making process and take appropriate action.
- Proactivity: Each agent uses its history, sensed parameters, and information of other agents to predict the possible future actions. These predictions allow agents to take effective actions that meet their goals. This ability implies that the same agent may take disparate actions when placed in different environments.

While an agent working by itself is capable of taking actions (based on autonomy), the real benefit of agents can only be harnessed when they work collaboratively with other agents. Multiple agents that collaborate to solve a complex task are known as *Multi-Agent Systems (MAS)*.

III. MULTI-AGENT SYSTEMS (MAS)

The salient features of MAS, including efficiency, low cost, flexibility, and reliability, make it an effective solution to

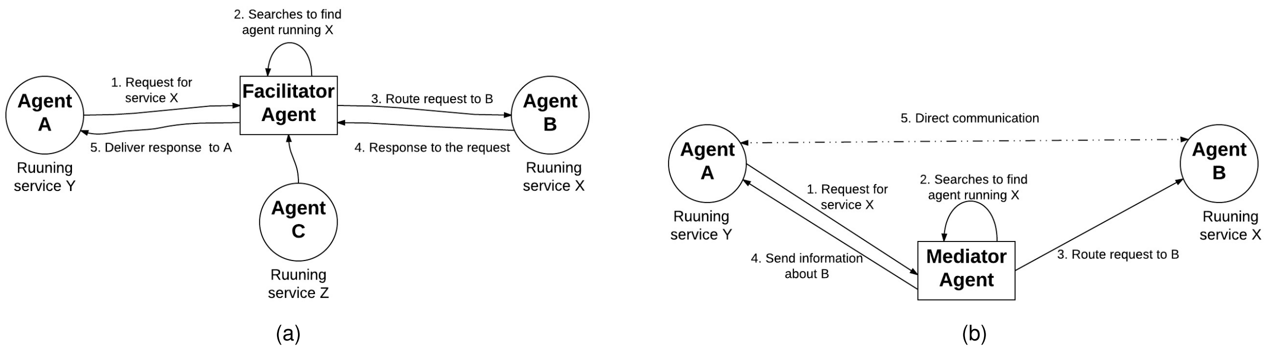


FIGURE 3. Implementation of a) Facilitator, b) Mediator.

solve complex tasks. Their efficiency stems from the division of labor inherent in MAS whereby a complex task is divided into multiple smaller tasks, each of which is assigned to a distinct agent [13]. Naturally, the associated overheads, e.g., processing and energy consumption, are amortized across the multiple agents, which often results in a low cost solution as compared to an approach where the entire complex problem is to be solved by one single powerful entity. Each agent can solve the allocated task with any level of pre-defined knowledge which introduces high flexibility [14]. The distributed nature of problem solving adopted in MAS also imparts high reliability. In the event of agent failure, the task can be readily reassigned to other agents.

To study MAS, agents and their relations are modeled using graphs. Graphs have been extensively used in computer science for modeling complex systems, e.g. social media [15], and analyzing them mathematically. When MAS are modeled as a graph, each vertex represents an agent and an edge between two vertices indicates that the two agents are communicating with each other (see Section VI). The actions taken by an agent may potentially change the relations between agents and thus change the structure of the graph. The final decision made by an agent applies to the corresponding graph that might change the edges or structure of the graph. Details of graphs relevant to MAS are beyond the scope of this paper and readers are referred to [16], [17] as complete resources.

In MAS, agents typically only have partial information as an agent mainly communicates with its direct neighbors. On the one hand, this reduces the communication overhead which in turn ensures scalability since the overheads remain relatively low as the number of agents increase. On the other hand, the time and communication overhead associated with finding an agent that provides a particular service increases. To reduce the outlined overheads for finding an agent, particularly for large-scale MAS, the notion of *middle agents* is introduced. Middle agents maintain a list of services offered by all agents. Any agent that is searching for a particular service first contacts a middle agent which directs it to the appropriate agent offering that service [18]. Depending on the implementation, middle agents can be classified as:

- Facilitator: As shown in Figure 3a, a facilitator acts as an intermediary between the agent sending the request (requester) and the agent providing the service (requestee). The facilitator routes the request to the appropriate agent. The response is sent back to the facilitator which relays it to the requester. As is evident, the facilitator becomes a bottleneck and a potential single point of failure [19]. To amortize the effects of a central facilitator, multiple collaborative facilitators are employed to respond to the requests [20]. This method requires the facilitators to communicate to remain synchronized and balance the load, i.e., requests, among themselves.
- Mediator: This implementation differs from the above in that the requester and requestee agents can communicate directly with each other as shown in Figure 3b. This reduces the load on the mediator agent as compared to the facilitator implementation.

TABLE 1. MAS features and categorizations that arise while considering each feature.

Features	Categories	References
Leadership	Leader-follow Leaderless	[21], [22] [23], [24]
Decision function	Linear Non-linear	[25], [26] [27], [28]
Heterogeneity	Heterogeneous Homogenise	[29], [30] [31]
Agreement parameters	First order Second order High order	[32] [33] [33]
Delay consideration	Time delay Without time delay	[34], [35], [33] [36]
Topology	Static topology Dynamic topology	[37] [38], [35]
Data transmission frequency	Time triggered Event triggered	[39] [34], [40]
Mobility	Static agents Mobile agents	[1] [41], [42]

A. MAS FEATURES

In the following we outline seven important features of MAS and discuss the different categorizations that arise while considering each feature. Table 1 proposes a summary of these features.

1) LEADERSHIP

Herein, we consider the existence of a leader, i.e., an agent that defines goals and tasks for the other agents based on one global goal. The presence or absence of such a leader can be used to categorize MAS as leaderless or leader-follow. In leaderless MAS, each agent autonomously decides on its actions based on its own goals. The decision of each agent is affected by the decision of other agents if agents collaborate to reach consensus (see Section V) on a particular feature. In contrast, in leader-follow, the leader agent defines actions for the rest of agents. Followers (i.e., other agents) communicate and share knowledge to find the position of the leader. The leader is either predefined or is collaboratively chosen by agents [43]. MAS can have a mobile leader or a group of agents acting as leaders. A mobile leader can move from one location to another (see discussion about mobility below). Thus, agents may need to track the position of the leader which incurs additional (communication and processing) overheads and delays. When there are multiple leaders, they collaboratively guide the followers by communicating with each other to make decisions.

2) DECISION FUNCTION

Herein, we categorize MAS based on the proportionality of the changes in the decision function output to its input changes. According to this, MAS are categorized as: linear and non-linear. In linear MAS, the decision of an agent is proportional to the sensed parameters from the environment, e.g. a thermostat agent turns off the heater when the temperature reaches a threshold. This feature makes linear agents easy to analyze mathematically. In non-linear MAS, the decision of the agent is not proportional to the sensed metrics due to the nonlinearity of the input to the decision making process, e.g., multiple aircrafts must agree on the pitch rate which is affected differently by earth arc and pitch angle [44].

3) HETEROGENEITY

Based on the heterogeneity of agents MAS can be divided into two categorize namely: homogeneous and heterogeneous. Homogeneous MAS include agents that all have the same characteristics and functionalities, while heterogeneous MAS include agents with diverse features.

4) AGREEMENT PARAMETERS

In some applications of MAS, agents need to agree on particular parameters known as metrics. Based on the number of metrics, MAS are classified as first, second, or higher order. In first order, agents collaborate to agree on one metric. For example, multiple warheads in a multi-warhead missile follow the same trajectory to destroy the targeted area [45]. In second order, agents must agree on two metrics, e.g., considering both position and velocity for spacecraft formation [46]. High-order MAS were first defined by the authors in [47]. Based on their definition, in high-order MAS agreement between agents is made when the metrics (either one or two) and their high-order derivatives converge to a common value. For example, in a bird flock, birds consider not only the position and velocities but also acceleration for agreeing on the direction of flying.

5) DELAY CONSIDERATION

Agents might face multiple sources of delay for performing tasks. For instance, delay in the communication media, e.g. wireless or wired, used by agents to exchange data, or delay in scheduling resources for each agent. Depending on whether the delays are substantial and relevant, MAS can be classified into two groups namely with delay or without delay. The former takes the delay sources into account, while the latter assumes that there are no delay sources. The latter instance is rather simplified as there is no communication and processing delay. However, most real world applications always experience non-trivial delays.

6) TOPOLOGY

Topology refers to the location and relations of agents. MAS topology can be either static or dynamic (also known as switching in the literature). In a static topology, the position and relations of an agent remains unchanged over the lifetime of the agent. In dynamic topology MAS, the position and relations of an agent changes as the agent moves, leaves or joins the MAS, or establishes new communications, i.e. relations, with other agents.

7) DATA TRANSMISSION FREQUENCY

Agents sense the environment and share the sensed data with other agents either in a time-triggered or an event-triggered manner. In the former, the agent continuously senses the environment, collects data, and in pre-defined time intervals sends all newly sensed data to other agents. In event-triggered MAS, the agent only senses the environment when a particular event occurs. Then, the agent sends the collected data to other agents.

8) MOBILITY

Based on their dynamicity, agents can be classified as static or mobile agents. A static agent is always located in the same position in the environment, while mobile agents can move around in the environment. A mobile agent can be hosted by other agents meaning that it uses the resources of other agents, monitors them, or senses the environment from the position of other agents to perform actions. For example, an Intrusion Detection System (IDS) agent moves between multiple servers (agents) in the network to analyze the server processes and communications and thus detect attacks.

B. DIFFERENTIATING MAS WITH SIMILAR SYSTEMS

Wooldridge [1], Zhao *et al.* [25], and Sadeghi *et al.* [48] compared MAS with expert systems and object-oriented programming language, two concepts that also involve decision making and sharing of knowledge. An expert system senses the environment and learns knowledge, then makes a decision to

TABLE 2. A study on the differences of MAS with expert systems and object-oriented programming.

	Metrics used for decision making	The way action is performed on the environment	With whom can communicate	Autonomy
MAS	Knowledge, Inputs, goal	Each agent can act according to the decision	Any agent in the network	Autonomous in sensing, making decision, and acting on the environment
Expert systems	Knowledge, Inputs	A controller is advised on the decision made and proper actions	A pre-defined list of entities	Autonomous in sensing the environment and making decision
Object-oriented programming	Inputs	The object can perform an action from a limited range of pre-defined actions	A pre-defined list of objects	Object has a pre-defined list of actions and inputs

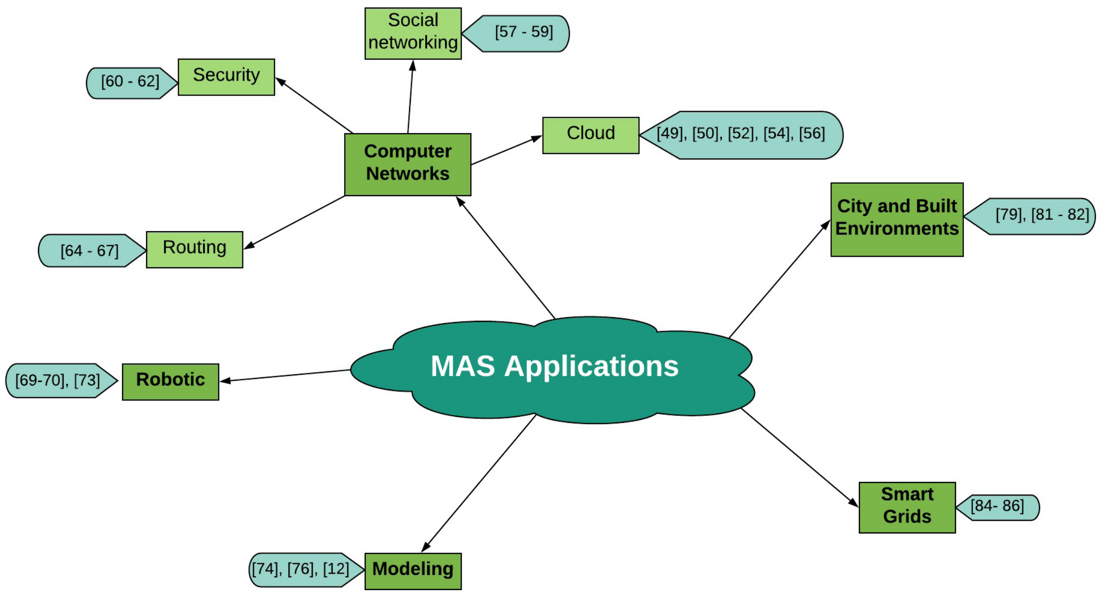


FIGURE 4. A summary of MAS applications.

solve a task. Unlike MAS where each agent can communicate with any other agent, an expert system can communicate and exchange data with pre-defined entities. Although both MAS and expert systems use a decision making function, the input differs which affects the final decision. In an expert system the decision is based on the sensed data from the environment and the knowledge of the expert system, while an agent also uses its goal. According to the decision, an expert system advises a controller to perform an action. The controller can reject the decision of the expert system as it is a separate system that uses other inputs as well. However, an agent directly acts on the environment after making a decision.

In object-oriented programming, an object (e.g. object A) can communicate (or share knowledge or resources) with a pre-defined limited set of objects by creating a public function. Other objects invoke the function to communicate with object A. When object A permits other objects to access one of its functions (by making the function publicly available), it cannot control the frequency with which the function is accessed. However, in MAS an agent can communicate with any node in the network and can control the frequency with which other agents request its resources. Objects have limited pre-defined inputs that can be passed from the main object, while agents use multiple inputs. A summary of differences between MAS, expert systems and object-oriented programming is given in Table 2.

In this section we outlined the key features of MAS. These features increase MAS applicability which is outlined in the next section.

IV. MAS APPLICATIONS

In this section, we present a taxonomy of MAS applications based on broad discipline: i) Computer networks (section IV-A), ii) Robotics (section IV-B), iii) Modeling (section IV-C), iv) City and built environments (section IV-D), and v) Smart grids (section IV-E). A summary of these applications is outlined in Figure 4.

A. COMPUTER NETWORKS

The complexity in computer networks significantly increases due to the emergence of new technologies and proliferation of Internet-connected devices. Agents are widely used to overcome this complexity. Due to the broad range of applications of MAS in networks, we further classify them into four sub-categorize:

1) CLOUD COMPUTING

Cloud computing enables ubiquitous access to the configurable system resources (e.g., CPU, GPU, and memory) and computing services (e.g., servers, databases, networking, and software) often over the Internet. Cloud computing uses *virtualization* as the underlying technology to provide service to the users. Using virtualization, a physical machine is shared among multiple customers as multiple *Virtual Machines (VMs)*, each of which emulates a distinct machine. Cloud computing has multiple advantages compared to the traditional approach wherein each user maintains dedicated resources for himself:

- Reduction of the monetary cost: The cloud users rent the resources they require which is managed by the cloud provider. This eliminates the need for buying and maintaining resources and thus reduces the monetary cost for the user.
- Reliability: The existence of multiple replications for each resource makes cloud computing resilient against resource failure and thus increases its reliability. In event of a resource, e.g., a CPU, failure its tasks can be offloaded to other similar resources.

The following are the key challenges that introduce complexity in cloud computing: managing the cloud resources and communication and accounting the usage of resources and/or services by each user [49], [50].

In [51] an agent-based framework is proposed for executing Bag-of-Tasks (BoT), i.e., a collection of independent tasks, in the cloud. A group of broker agents collect information about available resources and the cloud providers for each particular task in BoT, thus this group essentially are middle-agents discussed in Section III-A. Broker agents match each customer with an appropriately suited cloud provider. Then, the customer and the cloud provider negotiate to reach an agreement over the resources to be offered to the customer and the price that should be paid to the cloud provider. The allocation, deallocation, and execution time for a task is studied using simulations as well as the cost from the end users point of view. Simulation results show that the agent system has higher success in allocating tasks compared to Amazon EC2 cloud resource allocation method.

During task allocation, it is important to consider the load in each resource as overloading results in extensive delay in receiving services. Singh *et al.* [52] proposed an agent-based load balancing framework to balance the load in VMs. The cloud provider defines a policy to control the load on the VMs. The proposed method benefits from three agents namely: load agent, migration agent, and channel agent. The load agent ensures that the load on VMs matches with the defined policy by monitoring their load. To monitor the load on VMs, the load agent requests the channel agent to initiate a migrant agent. Migrant agents are mobile agents (see Section III-A) that move in the network and collect information about resources and VMs. This information is reported to the channel agent which controls transfer policy and selection policy. Then, the channel agent forwards the received information to the load agent. Based on the received information, the load agent balances the load between VMs by allocating incoming tasks to VMs with less load.

MAS are applied for resource monitoring [51], security [53], resource discovery [54], and automatic service management [55] in the cloud. A complete survey of MAS applications in the cloud can be found in [56].

2) SOCIAL NETWORKING

The popularity of social networks has increased exponentially with the growth in Internet users. A social network is comprised of actors, e.g. users, groups, and services [57]. The complexity of the social networks is derived from its dynamicity, i.e., large number of participants joining or leaving the network or establishing new connections with other participants, and its broad range of applications and services. MAS can be a potential solution to overcome the complexity of social networks.

Gatti *et al.* [58] proposed an agent-based method to predict user behavior, e.g. likes, posts, and follows, in social networks, e.g. Twitter. The authors proposed the use of multiple agents, i.e., actors, which are distributed in the social network to collect a dataset of the behavior of the users. The agents then perform topic and sentiment classification on the data of each particular user which is then used to build the user profile. Finally, the user profile is fed into a prediction system that predicts the future behavior of the user including likes, topics, replies, posts, and shares.

A social network needs not to be necessarily web-based. Any location that humans with similar interests gather together to interact and share information for a particular reason can be considered as a social network. Ma and Zhang [59] considered a school as a social network and applied MAS to aid the school managers in finding the relation between the distribution of fund to different school programs, e.g. sport, academic, and cultural activities, and school performance. Students, teachers, and school departments are represented as agents which collaboratively form the social network, i.e., the school. The interactions of each student with other students or teachers, the fund distribution policy from past years, and the performance of student in the current and past years are organized in a hierarchical structure. This structure is then fed into a learning function that evaluates the relation between the funds distribution policy and the academic performance of the students.

3) SECURITY

Security applications of MAS in networks have been studied since 2002 [60]. MAS are an effective solution to network security as they can proactively learn about and thus detect new security threats (see Section II).

An autonomous agent-based Intrusion Detection System (IDS) is proposed in [61]. The proposed IDS consists of five agents namely collective, detection, decision, response, and collaborative agents. The collective agent collects Simple Network Management Protocol (SNMP) and routing tables content from the network, and sends this data to the detection agent. The latter uses a misuse and anomaly detection engine to detect unusual packets or communications. The results of the detection engine are fed into a decision agent that decides if any malicious activity has occurred in the network and if so, it decides on a proper way to mitigate its affects. The decided action is then passed to a collaborative agent which is a mobile agent (see Section III-A) that delivers the decision to a response agent. The response agent enacts the appropriate action in the network.

Agent-based IDS is the main application of MAS in network security in the literature. However, MAS is also applied for other security applications. Sarika and Paul [62] proposed an agent-based security method to protect users against tabnabbing attack. In this attack, the attacker opens random websites on idle tabs of the target user's web browser. Two agents monitor open tabs and collect 5-tuple elements which are text, image, URL, title, and favicon, i.e. the webpage title icon. Then, agents compare the fingerprint of the actions running on the open tabs with the existing signatures of attacks to detect the tabnabbing attack. Simulation is conducted using JADE simulator (see Section VII). The simulation results demonstrate that using agents, false positive and false negative rates decreases compared to non-agent methods, leading to 91% accuracy in the attack detection.

It is likely that in the future MAS could also be used as a solution for complex security tasks such as trust and key management. Additionally, due to large scale and special features discussed in Section III-A, MAS can be used to overcome security challenges of Internet of Things (IoT). A comprehensive study on agent-based IDS is proposed in [3].

4) ROUTING

Routing refers to finding a path for packets from a source node towards a destination based on specific metrics, e.g. the number of hops between the source and the destination. Using agents for routing is among the first applications of MAS studied since 1998 [64]. MAS have been evolved since then as new challenges emerge for routing protocols.

Claes *et al.* [65] proposed a decentralized agent-based routing protocol for Vehicular Ad-hoc Networks (VANET) that is comprised of three groups of agents. The agents in the first group are hosted by the vehicles (see Section III-A) to monitor the vehicle parameters, e.g., speed, location, etc. The second group are hosted by the roadside infrastructures and monitor communications of the vehicles with each other and the roadside infrastructures. The third group create a virtual environment by modeling the vehicles, roadside infrastructures, and their communications, as a graph (see Section III). Ant colony algorithms are applied to the graph by agents to find a short route toward the destination. This method prevents congestion in an area by sending different routes to different vehicles. Implementation results prove that the agent-based method has shorter trip duration compared to other studied routing methods.

One of the key challenges in routing is the reliability of the selected route, i.e., ensuring that the route will not break frequently. Bendjima and Feham [65] proposed a reliable routing protocol for Wireless Sensor Networks (WSN). The proposed method, benefits from agents in forwarding packets and aggregating the sensors' data. Initially, the sink, i.e., the central controller in WSN, broadcasts a packet to find available resources of nodes for specific tasks which are used for sectoring the network. Then, the sink establishes mobile agents (see Section III-A) to collect and aggregate the data of sensors in multiple sectors. By tracking the sensors that each mobile agent visits, the sink initiates a sorted list of sensors based on their distance to the sink for each sector that is used as a reliable route toward each sensor. Simulation results, demonstrate lower energy consumption and packet drop rate in the agent-based routing protocol compared to conventional routing protocols.

In another attempt to address reliability, Manvi and Kakkasageri [67] proposed a reliable multicast routing protocol that consists of two groups of agents. The first group are mobile agents and move in the network to collect energy, bandwidth, mobility, and memory state of the nodes. This information is used to compute a Reliability Factor (RF) for each node. The RF is then passed to the second group of agents that establish a multicast backbone using the nodes with highest RF. Simulation results prove that the agent-based method has higher packet delivery ratio and reliability compared to methods that do not use agents.

B. AGENTS IN ROBOTICS

Using agents for robotics has been studied for over two decades with the first article published in 1996 outlining the the pros and cons of agents in robotics [68].

Cena *et al.* [69] argued that there exists two main challenges in robotics namely: (1) cooperation and coordination between robots, and (2) planning their movement trajectory. The authors then proposed a method that uses hardware and software agents (see Section II) to overcome the outlined challenges. Hardware agents refer to the physical hardware that makes up the robot, while software agents are decision making, path planning, task management, and communication agents. A hardware agent uses its sensors, e.g. cameras, to capture images of the environment. Then, a communication agent sends the images to an image processing agent. The latter processes images to find the location of the robot and obstacles in the environment. This information is then sent to the decision maker agent that finds a path with minimum obstacles toward the destination. The implementation results demonstrated that the proposed method could detect obstacles and find an optimized path (without any obstacle) to reach the destination.

Robots may be deployed in non-deterministic dynamic environments (see Section II) which increases the complexity of their decision-making. To study such complexity, an agent-based soccer robot is proposed by Duan *et al.* [70]. Agents (i.e., players) are grouped into teams. Agents in a team learn knowledge regarding the opponent team and possible actions by interacting with the environment, then they share learnt policies with other agents in their team. Reinforcement learning is used along with Probabilistic Neural Networks (PNN) to increase the accuracy of the final decision made by the agent. Implementation results show that agents predict the correct actions leading to an increased ball possession percentage (an important performance measure in soccer) in the agent team compared with the non-agent team.

Further applications of agents in robotics are discussed in [71]–[73].

C. AGENTS FOR MODELING COMPLEX SYSTEMS

Modeling complex dynamic systems is costly and incurs significant processing overhead due to the demand for powerful modeling platforms and high complexity. The flexibility, autonomy and scalability afforded by agents (see Section II) makes Agent Based Modeling (ABM) a low-cost and low-resource solution for modeling complex systems. ABM uses a rule-based methodology for modeling the environment in contrast with other modeling methods that use equations. The most important advantages of ABM are [74]: i) ability to be aggregated and combined with other modeling methods, ii) flexibility in assumptions for modeling a MAS, iii) flexibility in pre-defined knowledge as agents can gain knowledge by learning from the environment, iv) possibility of parallel execution which can speed up the modeling process, and v) ability to explore emergent behaviors due to agent proactivity.

Domínguez *et al.* [75] proposed an ABM to model supply chains. Each entity in the supply chain is modeled separately with its own policies and is able to define its own interactions with other entities. The proposed method comprise of two groups of agents, namely: planning and physical agents. Costumers and suppliers use six agents in the planning group to negotiate and reach an agreement over the price of the products. The agents in the planning group are as follows: i) demand fulfillment agent which manages the customer demand, ii) material resource planning agent which communicates with the producers and purchases products, iii) demand forecast agent that forecasts the demand of the customers based on the current and history of demands, iv) master planning agent which aggregates production planning, v) production planning agent which disaggregate the aggregated planning by the master planning agent, and vi) scheduling agent which schedules jobs in multiple agents. Three agents in the physical group perform physical tasks which are receiving and storing raw materials, manufacturing, and delivering products to the customer.

An ABM is proposed in [76] to study the usages of an Urban Distribution Center (UDC). In a smart city, the UDC centrally manages the delivery of products to control city congestion, pollution, delivery time, and reliability by optimizing and scheduling the routes used for delivering goods. In the proposed method, MAS are comprised of trucks, goods, costumers, UDC, and city vehicles. Unpermitted or overtime parked vehicles in parkings lead to parking problems for trucks and thus incur delay in delivering goods. The UDC agent considers this challenge while designing delivery paths to further reduce the delivery time. The proposed method minimizes the delivery cost for shop owners and maximizes the delivery profit for delivery companies compared to conventional management systems.

Based on arguments made in [12] the existing modeling environments for the power industry are not able to effectively model complex scalable power networks, e.g., a smart grid with a large number of different energy sources and consumers. ABM is an effective distributed modeling method for smart grids, that makes ABM an interesting research direction in this field [77]. Readers are referred to [73] and [77] for more detailed discussion on ABM.

D. AGENTS IN CITY AND BUILT ENVIRONMENTS

In recent years, using agents for managing cities and buildings has received tremendous attention from researchers. In a city, unorganized distribution of freight increases the cost, pollution, and congestion. Khayyat and Awasthi [79] proposed an agent-based method to address this challenge using six agents namely: RFIDG, retailer, supplier, carrier, network, and city agents. The RFIDG agent uses RFID tags to manage the supply of resources. To buy goods from a retailer or a supplier the customer sends its request to either the retailer or the supplier agent. On receiving the request, the retailer or supplier agent searches its database to find the requested goods. Next, the goods are sent to the carrier agent to be shipped to the customer. The network agent determines the optimal path that reduces congestion in the city for the carrier agent to deliver the goods. The city administrator agent informs both parties of a transaction (i.e., the supplier and the customer) about the relevant policies and rules, e.g, commerce and shipping goods.

Another fundamental challenge in cities is controlling and managing the transport system and traffic in growing metropolitan areas. Hager *et al.* [80] proposed an agent-based method to address this challenge. The proposed method considers parameters such as fare and people satisfaction for defining and analyzing the transport model. They use two groups of agents: travellers and vehicles. The agents in these groups share their knowledge about congestion and traffic which is used by other agents to decide on a low-congested short path toward their destination.

MAS is also applied for managing buildings. An agent-based method for heating management in buildings is proposed in [81]. The proactivity and flexibility of agents (see Section III-A) made them effective solutions for managing heating systems which are comprised of heterogeneous heating devices and sensors distributed around the building. A demand agent checks temperature in the building and passes the data to gas heater, buffer, and heat pump agents. The latter agents use the received data from the demand agent

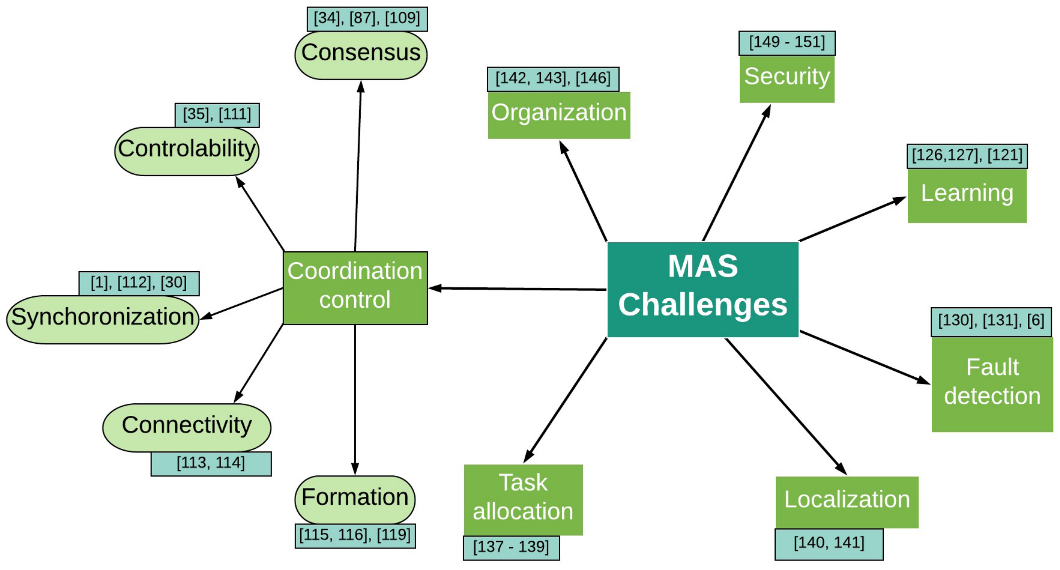


FIGURE 5. **A summary of MAS challenges.**

to balance the temperature of the building. The authors implemented the proposed method in an apartment and compared it with a centralized method as a benchmark. The implementation results show that the daily power consumption is significantly reduced using the agent method as compared to the centralized approach.

Further reading in this field can be found in [81] and [82].

E. AGENTS IN SMART GRIDS

In the literature, agents are used to address multiple challenges of smart grids including balancing the generated and the demanded energy, negotiating between the energy consumer and producer over the energy price, storing energy in the home storages, and energy restoration.

Nguyen and Flueck [84] proposed an agent-based service, i.e., energy, restoration for smart grids that supports the distributed energy storage. The system uses two agents namely: switching and distributed energy storage agent. The switching agent balances the energy load, and detects and then isolates faults, i.e., power outage due to a fault in some energy producers or energy distribution system. The energy storage agent provides energy for the grid depending on whether the grid is connected to the smart grid or is isolated. The proposed method improves the system loss by effective restoration and achieves dynamic islanding to restore the parts of the grid which are being disconnected (islanded).

The participants in smart grid are either energy producer or consumer. The energy producers aim to enhance their profit by selling energy with higher price. To achieve this aim, Vytelingum *et al.* [85] proposed an energy storage management method. Each energy producer (agent) has a storage device and aims to increase its profit by analyzing the price signals generated by other agents including customers and other energy producers. Each agent records the storage usage in a unique storage profile. The agent analyzes the storage profile using game theoretic methods to predict future usage and thus decide on whether to sell or store its produced energy.

Readers are referred to [86] for further reading of MAS applications in smart grids.

In this section we have provided a high-level discussion on MAS applications in different disciplines. The next section discusses key challenges in MAS.

V. MAS CHALLENGES

Although the salient features of MAS increase its applicability in multiple disciplines, significant research challenges need to be addressed which include: coordination between agents (section V-A), learning (section V-B), fault detection (section V-C), task allocation (section V-D), localization (section V-E), organization (section V-F), and security (section V-G). MAS challenges are typically application-specific. In this section, we outline the key challenges that apply to the vast majority of applications as summarized in Figure 5.

A. COORDINATION CONTROL

The action performed by each agent affects the environment and thus the decision made by other agents. Coordination control refers to managing agents to collaboratively reach their goals [14]. Multiple challenges arise from coordination including consensus, controllability, synchronization, connectivity, and formation which are outlined below:

1) CONSENSUS

In MAS consensus refers to achieving a global agreement over a particular feature of interest. Consensus has been widely studied in the literature [34], [87]. Multiple MAS features, outlined in Table 1, impact the consensus problem as they affect communication and collaboration between agents. Table 3 summarizes the feature-specific consensus approaches. This table also provides references for further study. We disregard MAS features that have no effect on the agent relationships compared to classical MAS, e.g., leaderless MAS.

TABLE 3. Feature-specific MAS consensus approaches.

MAS feature	Ref
Leader-follow	[88]–[90]
Event-triggered	[34], [91], [92]
Non-linear	[87], [89], [93]
Linear	[94]
Direct topology	[34], [36], [91], [95]
Dynamic topology	[93], [96], [97]
Time delay	[40], [98], [99]
Discrete time	[100]–[102]
Lossy network	[103], [104]
Communication delay	[105]
First order	[95]
Second order	[87], [88], [93]
High order	[106], [107]

The authors in [38] established a new consensus algorithm for both fixed and dynamic topology MAS. The authors mathematically prove that using their algorithm agents can reach an agreement on a particular feature of interest. Simulation results demonstrate that as time passes, the agent disagreement on the feature of interest reduces till they finally reach an agreement. The authors also defined a new type of consensus known as *average-consensus* wherein the agents reach a consensus over the average of the initial state of the particular feature of interest. In occasions where finding a unique parameter for agents to reach consensus either incurs significant (processing and communication) overhead or is impossible due to the diversity of agents and their features, average-consensus can be employed to make the agents consistent.

One of the well-known instances of consensus is flocking which refers to collective behavior of a number of agents to reach a group objective [108]. Flocking exists in nature, e.g. ant colony, bees, fish, and penguins [38]. In one of the pioneering papers on flocking [109], the authors proposed three main rules that lead to a flocking model. These are:

- Flock centring: attempt to stay close to nearby flockmates
- Obstacle avoidance: avoid collisions with nearby flockmates.
- Velocity matching: attempt to match velocity of nearby flockmates.

These three rules are known as cohesion, separation, and alignment, respectively. Flocking applications in engineering include: vehicular and transportation control, military missions, and mobile sensing [108], [110].

Achieving consensus has two main sub-challenges based on the features of the underlying MAS, namely:

- *Tracking*: In leader-follow MAS (see Section III-A) with one leader, agents should reach consensus over the position of the leader. This ensures that agents maintain their connection with the leader. This challenge is known as tracking [21].
- *Containment*: Containment is similar to tracking with the exception that MAS have more than one leader [111]. The leaders may limit the geographical position of agents to control the borders of their group. Additionally, leaders may connect to each other to exchange control data or the data produced by agents, e.g. sensed data from the environment.

2) CONTROLLABILITY

Controllability refers to the situation when MAS can be steered from an initial state to a specific state using certain regulations [35], [112]. Therefore, there is certainty in reaching a particular state by following specific steps. Two key metrics that affect MAS controllability are the degree of dynamism in the topology (see Section III-A) and the degree of determinism in the environment (see Section II). In a dynamic MAS, the topology changes periodically, affecting the links between agents and thus their collaboration. In non-deterministic environments the results of actions are not predictable, thus agents should decide on new actions after observing the result of their previous actions which incurs delay and limits the proactivity of agents. In the literature, controllability is largely achieved in a centralized manner, where designated leaders instruct followers to reach a particular goal. Controllability has multiple applications which includes: managing aircraft, vehicles, and robots [35].

3) SYNCHRONIZATION

Synchronization means the actions each agent performs are aligned in time with other agents [1]. Synchronization has a close relation to the consensus challenge that is nicely highlighted in [113] as: "consensus means agents reaching an agreement regarding a certain quantity of interest that depends on the state of all agents, while synchronization defines the correlated-in-time behavior among different agents." Increased heterogeneity among agents complicates synchronization. The reason is that homogeneous agents can be synced over a common feature, while for heterogeneous agents, synchronization should be achieved over multiple distinct features. Heterogeneous synchronization is also known as partial-state or output synchronization in the literature [30].

4) CONNECTIVITY

In some circumstances, agents demand permanent connection to each other, e.g. agents in leader-follow MAS should always be connected to the leader. This requirement introduces the connectivity challenge. The following challenges contribute to the complexity of the connectivity: i) mobility: the mobility of agents leads to frequent disconnections between agents and subsequent re-establishment of new connections, ii) noisy environments: any interference in the environment can interrupt the connection between two agents and thus require re-establishment of these connections, and iii) limited view of the MAS topology: as agents have limited view, positioning the agent to achieve maximum connectivity is challenging [114].

Based on connectivity, MAS can be classified into connected or connectionless. In connected MAS, permanent connectivity between agents is guaranteed, while in the connectionless model there is no such guarantee. Connectivity is widely applied in flocking and vehicular systems [115]. In these instances, the position of agents changes periodically and agents must always maintain their connectivity with each other.

5) FORMATION

In some instances, it may be necessary for a group of agents in a MAS to be organized in a particular structure and that this structure be maintained for a specific period of time (which could even be the entire lifetime of the MAS). For example, a group of Unnamed Airborne Vehicles (UAVs) can be requested to form a particular formation to find a specific item in the environment, or to sense multiple parameters from the environment. This challenge is known as formation challenge in the literature [116], [117]. Three main steps in formation are: 1) finding the most effective structure to be applied to all agents, 2) organizing agents based on the determined structure, and 3) maintaining the determined structure for a specific time. Heterogeneity of agents, limited view of the environment, and dynamicity of the MAS or environment makes the outlined steps highly challenging [118].

Liu and Geng [115] discussed the formation of agents for a finite time. Xia *et al.* [120] proposed a position aware formation method where a central agent which knows the position of all agents, controls the formation. Formation has a broad range of applications in military, disaster management, UAV control, and vehicular control [119], [121], [122]. A complete dicussion on formation and its applications can be found in [123].

B. LEARNING

In MAS, each agent autonomously decides on the appropriate action to reach to its goal based on multiple metrics (see Section III). Agents can leverage machine learning algorithms [124] to discover and forecast the changes in the environment and adapt to unforeseen situations [125] and thus form Multi Agent Learning (MAL) systems. The following challenges increase the complexity of adopting learning systems for MAS [5], [126], [127]: i) Processing and communication overhead of learning methods which consumes the resources of agents, ii) MAS environment may be dynamic. Thus, the agents must frequently sense updated information to be used by the learning machine, which in turn consumes a significant amount of agent resources, iii) The topology of MAS may change which requires reconnecting with neighbor agents, iv) Protecting agents against malicious agents which inject false information, and v) Scalability of the learning method for large scale MAS.

The agents can share their knowledge with their neighbors and achieve collaborative learning to reduce the effect of some of the challenges outlined above [128]. MAL is normally studied in the stylized setting provided by repeated games, e.g. Prisoners' Dilemma, Game of Chicken and Rock-paper-scisors [126]. Studying agents in the context of a game abstracts the basic concepts of MAL and directs the focus on MAL output by providing the final results of learning methods.

Two main machine learning methods used in MAL are Reinforcement Learning (RL) and Genetic Programing (GP). RL is a trial-and-error method where each agent changes its state and observes the reward or penalty it receives from the environment or other agents [5], [122], [126]. Each agent uses actions with positive effects repeatedly, while avoiding repeating actions with negative effects [127]. In RL, agents have no pre-defined knowledge or policies about the environment. In the literature, there are different reinforcement learning methods for agents such as Minimax-Q, the Friend -or-Foe Q-learning, Nash Q-learning, and Q-learning [122].

The second widely used learning method in MAL is GP which is a type of Evolutionary Algorithm (EA) in machine learning. In GP multiple computer programs are encoded as a set of genes that are evolved using EA [129]. As time passes, only the most effective genes survive and compete with other genes to approach the demanded solution [130].

C. FAULT DETECTION

Detecting and isolating faulty agents is a fundamental task as a faulty agent may infect other agents that it collaborates with [131]. Current methods for Fault Detection and Isolation (FDI) are mainly centralized, where a central agent aims to detect and then isolate faulty agents [132]. Centralized methods are suboptimal for large-scale and distributed systems such as MAS. This is attributed to the typical issues that plague centralized approaches including single point of failure and the potential for overwhelming a single agent (if many agents send requests to this agent). Consequently, FDI demands distributed solutions where all agents exchange data to detect and isolate faulty nodes [133].

A well-known approach for addressing FDI is the unknown observer. In this method, agents classify the inputs (i.e., the received signals or communications from other agents) into unknown and known groups. Next, the agents monitor the unknown input generators to detect faulty agents. The process of deciding on faulty agents is beyond the scope of this paper and is discussed with greater details in [130] and [131]. In [136] the existing FDI methods are classified into two categories based on the inputs used for identifying faulty nodes: a) without Embedded Residual Generator (ERG), and b) with ERG. In the latter category, the FDI uses new inputs from the environment to detect faults. In MAS with ERG, the results of the previous FDI is used in addition to the new inputs from the environment as the input of the FDI. Thus, the FDI always depends on the previous results which in turn decreases its robustness.

The current literature in FDI suffers from the following limitations [6], [132], [133]:

- The focus is mainly on homogeneous agents while in most applications agents are heterogeneous.
- Most existing methods demand high resource and/or data processing which might not be affordable by all agents.
- Only few works discuss isolating faulty agents. However, a detected but non-isolated faulty agent is able to send information to other agents, thus consuming resources of other agents.
- Most proposed solutions are centralized and thus not necessarily useful for MAS as discussed above.

A complete survey on FDI can be found in [6].

D. TASK ALLOCATION

Task allocation refers to the allocation of tasks to agents considering the associated cost, time, and (communication and processing) overhead. Task allocation can be centralized or decentralized [137]. Dos Santos and Bazzan [134] suggest a hybrid approach by organizing the agent system into multiple clusters. In each cluster one node (knows as cluster head) allocates tasks to the members of the cluster.

Two fundamental metrics for allocating tasks to agents are:

- *Agent talent*: This refers to the total number of resources of each agent. Agents are assigned tasks proportional to their resources. However, it is of great importance to consider the current load at an agent prior to allocating a new task to it. If the tasks allocated to an agent exceed the resources of the agent (i.e., the agent is overloaded), then the delay in receiving a response from the agent will increase significantly. To prevent overloading, load balancing is used to equally distribute the load between agents.
- *Agent position*: The position of an agent impacts the communication delay as well as overheads (e.g., the number of packets needs to be transmitted to communicate with other agents) [137], [139]. Thus, the agent position should be considered while assigning tasks to reduce the overhead. For instance, if task X demands resources owned by agents B and C, then it is preferred to allocate X to an agent that is located near both B and C [139].

Task allocation has diverse applications including allocating sensing tasks to heterogeneous agents and allocating rescue missions to ambulances [140]. A complete survey on task allocation is given in [137].

E. LOCALIZATION

Recall that each agent has limited view (only its neighbors) of the MAS topology. With this limited view, locating a particular agent, i.e. localization, can be challenging. An agent might be localized based on: i) having particular resources which is known as resource localization, ii) running specific services, or iii) owning a specific identity [141]. Most existing localization methods in literature are centralized. However, centralized methods would not necessarily scale for large scale MAS, thus raising the need for distributed solutions [141]. In a distributed solution, the agents exchange their neighborhood information, e.g., features, services, or identities, with other neighbors. Consequently, they can collectively build a global topology to localize any agent. However, this will increase communication overhead between agents. Additionally, there will be a significant delay in updating the information of agents in MAS. While a hybrid topology may be necessary to balance the overheads and centralization and distribution, determining the optimal level of distribution for localization remains an open issue.

The dynamicity of the agents clearly affects localization. Localizing dynamic agents demands more communication and computation resources compared to static agents. Agents can localize a dynamic agent by estimating its future position (or state) using its current speed or moving direction. This is known as state estimation [142]. Using state estimation, the agents can follow a mobile agent continuously which can potentially overcome the tracking challenge (see Section V-A.1).

F. AGENT ORGANIZATION

Organization refers to the way that agent communications and connections are defined. In certain approaches which are outlined below, agents can be grouped based on particular features such as goals, resources and services [143]. The prevalent approaches for organizing MAS are [144]–[146]:

- Flat: This is the most basic organization structure wherein all agents are regarded as equals. There is no designated leader and every agent communicates with its neighbors. Figure 6a illustrates this organization.
- Hierarchical: In hierarchical organization, agents have tree-like relations as shown in Figure 6b. Leaf agents, e.g., agents G and H in Figure 6b, communicate with other agents using their parents (agent B). Parents control their leaf agents, i.e., children, and may have their own parents. At the highest level, there is one agent known as root agent (agent A). Hierarchical organization may lead to delay or create a bottleneck, particularly at the root agent (or parents) as it is responsible for processing communications of all leaf agents. Based on the number of authoritative agents, i.e. those who have control over other agents, the hierarchical organization can be divided into two types namely simple and uniform. In the simple approach, the root agent has exclusive authority and controls all communications. In the uniform approach, there are more than one authoritative agents in the hierarchy meaning that in addition to the root, all or particular parents can also control their children.
- Holonic: In holonic organization, agents are organized in multiple groups which are known as holons based on particular features, e.g., heterogeneity or sensing ability of the agents in holon. Holons are then layered

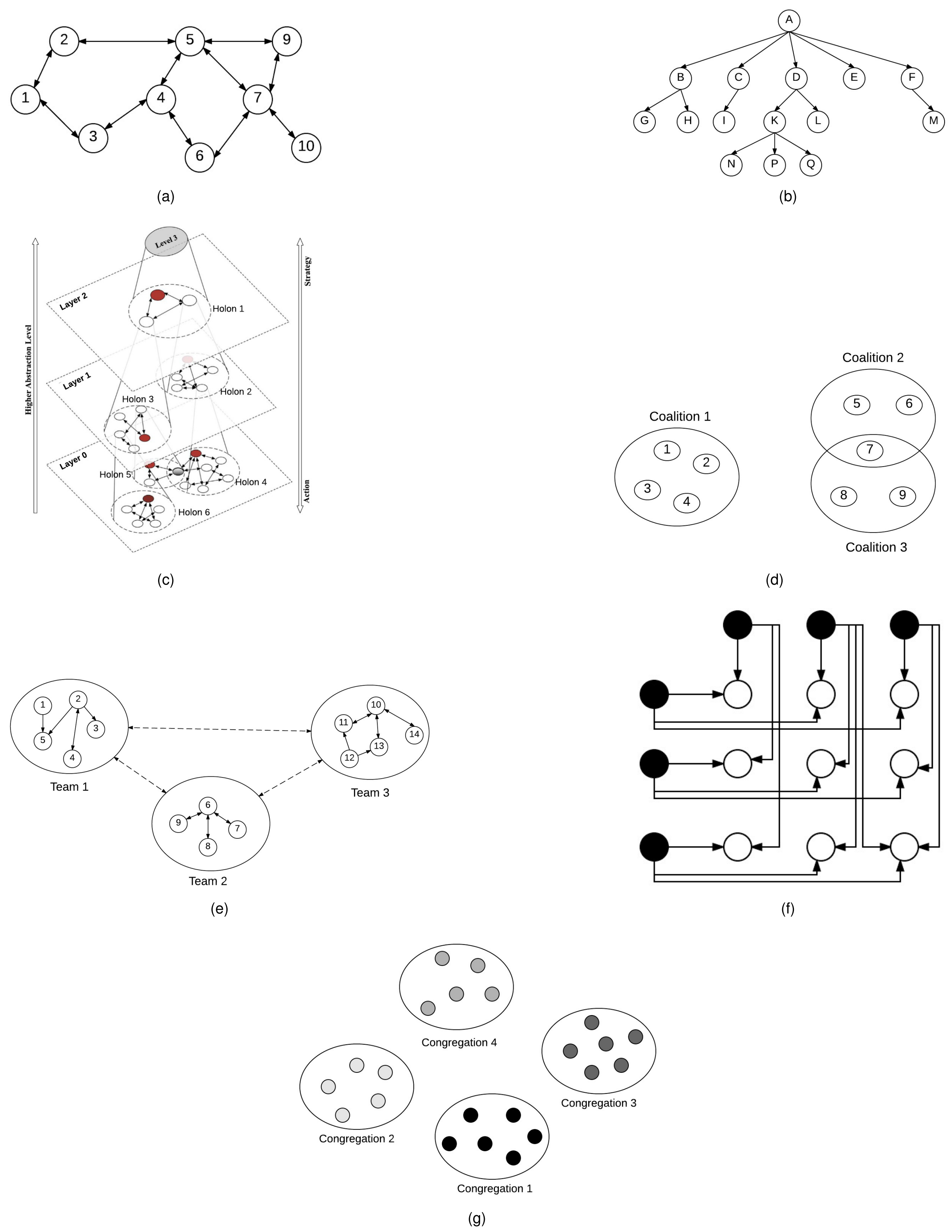


FIGURE 6. Fundamental organization methods employed in MAS, namely: a) Flat, b) Hierarchical, c) Holonic, d) Coalition, e) Team, f) Matrix, and g) Congregation.

in multiple layers as shown in Figure 6c. The agents can communicate with other agents in the same holon or in other holons in the same layer. Thus, in holonic organization an agent can be member of more than one holon in the same layer. For upper layer communications, a *head agent* is used which is selected among the most resource available agents in the holon. This organization is suited for MAS when each supper-agent (a member of a holon in upper layer) requires sub-agents (lower layer holon members) to solve a particular task collaboratively. Each sub-agent may also have the same requirement.

- Coalition: In coalition organization, agents are temporarily grouped based on their goal. Consequently, the agents can reach their own goal with lower (processing delay and communication) overhead compared to the organization where there is no such grouping. Figure 6d shows a coalition organization. Each agent can be part of more than one coalition, e.g., agent 7. By reaching their goal, the agents destroy the coalition. The internal organization of a coalition is normally flat; however, other organizations, e.g., hierarchical organization, can be used to further reduce overheads or apply control over agents. Finding and grouping agents with the same goal incurs processing and communication overhead on agents. Thus, there is a trade-off between the decreased overhead resulting from the coalition and the incurred overhead for finding agents with the same goal and forming them in coalition.
  This organization is suited when a collection of agents with similar goals exists in MAS which their collaboration associates them in reaching their goal. For example, a group of ambulances (agents) may gather together to rescue people during an earthquake. Forming coalition aids ambulances to reach their goal effectively as it distributes them fairly to cover wider area and thus rescue more injured people.
- Team: In team organization, the agents create a group (team) and define a group goal which differs with their own goal. Depending on the time required to reach the team goal, a team may be short-time or long-time. The agents in a team collaborate to reach the team goal. Figure 6e depicts a team organization. The team goal can be updated which leads to change in the responsibilities, roles and authorities of agents in the team. Each team can request information from agents in other teams to improve its own decision-making process. A team can have an internal organization (e.g. hierarchical) to improve the performance and efficiency in reaching the team goal. For instance, in a team of agents responsible for analyzing a city traffic, the agents create a hierarchical structure so that the traffic data can be integrated by the parents to reduce the communication overhead.
  The number of agents in a team (known as team size) is one of the key issues in the team organization. A larger team can sense more data from the environment; however, integrating the data and knowledge of multiple agents demands high processing. The final decision of the team is less challenging in small teams, however, the data used by small teams is limited. A trade-off between the decision-making overhead and the accuracy of the data should be considered while deciding the team size.
  Unlike coalition where agents are grouped to reach their own goal, in a team agents attempt reaching the team goal. Thus, this organization is suited when multiple agents attempt reaching the same goal.
- Matrix: In matrix organization each agent is administrated, i.e., managed, by at least two head agents (known as managers) as shown in Figure 6f. An example where this organization can be used is a company where each staff gives report to a product manager (that controls the product) and the functional manager (that controls the functions of a task). This organization is effective where agents are controlled by more than one leader (or manager).
- Congregation: In congregation organization agents in a location form a congregation to achieve their requirements that they cannot achieve alone. A farmer market can be considered as a human congregation where people gather to sell/buy what they need. Each agent can leave or join congregations but should be part of only one congregation at each point of time. The satisfaction of an agent in congregation, i.e., the degree in which an agent fulfills its requirements, depends on other agents in the congregation. A congregation should always have at least one member. Figure 6g shows congregation organization. This organization is affective where each agent requires the resources of other agents to achieve its goal or perform its tasks.

Readers are referred to [139] and [143] for further readings regarding agent organizations.

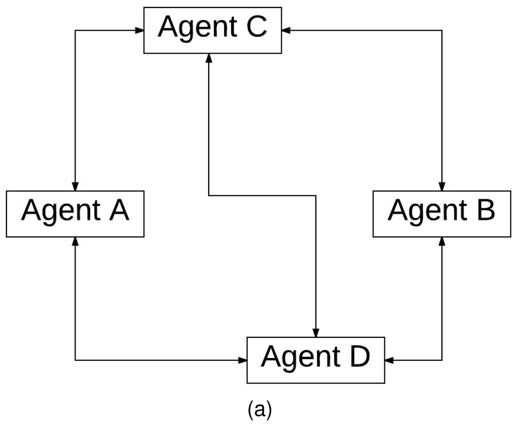
G. SECURITY

Security is highly challenging in MAS due to decentralization, sociability, and mobility [1]. The effects of these features are discussed below:

- Sociability: Agents use the information or knowledge that they acquire from either neighboring agents or the environment for the decision-making process. This makes an agent vulnerable against malicious entities that may share falsified data to impact the decision of an agent.
- Decentralization: With the lack of a central trusted authority, verifying the identity of agents and creating trust between agents become highly challenging.
- Mobility: A mobile agent might be affected by malicious agents. If so, it spreads false information to a growing number of agents that it encounters while moving. The other security threat of mobile agents is that it might attack the agent that it has migrated to it and consume its resources or read its data.

Key security requirements in MAS are [148], [149]:

1) Authentication: assures that each agent is the one that it claims to be.
2) Authorization: assures that each agent has the right to access what it requests for.



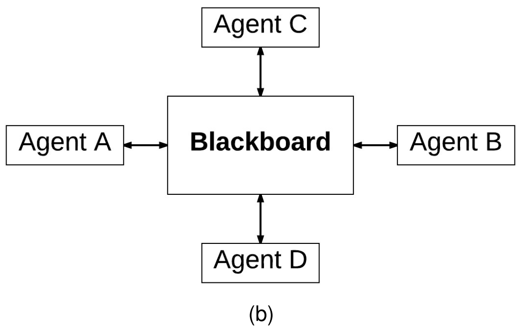


FIGURE 7. **Communication approaches in MAS: a) Message passing, b) Blackboard.**

3) Integrity: assures that a message has not being modified since generation.
4) Availability: assures that the services and resources are available to the authenticated and authorized agents when requested.
5) Confidentiality: assures that only the permitted agents can read particular data.

Distributed trust is widely used in MAS to enhance security whereby agents build trust in each other by observing and verifying their actions [150], [151]. In distributed trust, agents analyze the validity of the information they received from other agents. Agent A bestows more trust in agent B as it receives more truthful information from B [152].

This section outlined the key challenges in MAS. In the next section, we discuss the primarily communications methods in MAS.

VI. AGENT COMMUNICATION

Communication between agents has been studied for over 50 years [153]. Three widely used approaches for communication include:

- Speech act: In [153] John Austin, the pioneer researcher in speech act communication, identified that some utterance verbs or sentences, referred to as speech acts, change the physical environment, e.g. in the proper circumstances if a proper person says "I now make you man and wife," then this sentence affects the physical environment by defining new roles and conditions. An agent can act as a speaker (S) that produces utterance to change the beliefs of the hearer (H) [154]. Agents perform utterance by performing primary actions that are perceived as an utterance according to some language grammar [155]. Readers are referred to [1] for further reading.
- Message passing: In this method agents directly message each other as shown in Figure 7a. The agents use point-to-point or broadcast communication to talk to other agents. In the former, agent A can directly talk to agent B if it knows B's address. In broadcast communication model, agent A sends a message to all its neighbors. To ensure message interpretability, the agents in a communication must use an agreed structure which is further discussed below.
- Blackboard: In this communication method, agents can collaboratively share data with each other using a central repository called Blackboard as shown in Figure 7b. Each agent stores its data on the blackboard that is readable by other agents. To control the access of agents, the blackboard uses a control knowledge. Each agent can access multiple data defined in the control knowledge.

The message semantics are rather important to ensure that the agents communicating with each other have the same interpretation of the data exchanged. This is particularly challenging with heterogeneous agents. A simple example is when agent A sends temperature to agent B as 12°C, then the temperature should not being interpreted as $12°F$ by agent B. An Agent Communicate Language (ACL) aims to address the aforementioned challenge. An ACL provides a unique message format and ontology for all agents to communicate and interpret received messages. Balaji and Srinivasan [7] classified the ACL into two main categories namely procedural and declarative. In procedural ACLs, the communication between agents is modelled as a sharing of procedural directives. In declarative ACLs, declarative statements are used to specify definitions, assumptions, and assertions. The most important programming languages that support each of the mentioned categories are shown in Figure 8.

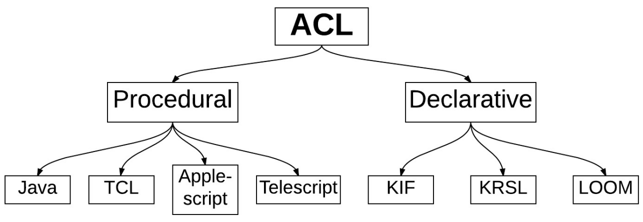


FIGURE 8. **ACL categories and their well-known programming languages.**

Foundation of Intelligent Physical Agents (FIPA) proposed a comprehensive ACL framework for agents which is now

TABLE 4. A summary of four MAS simulators.

Simulator	Programming language	Features	Ref
Repast	C++ and Java	Inherits diverse C++, Java libraries, graphical user-interface	[162]
MASON	Java	Discrete-event multiagent simulator, suitable for lightweight simulations, 2D and 3D visualisation	[163]
Netlogo	Scala with some parts of Java	Suitable for teaching and demonstration, each person can run a part of the program in a separate device and participate in the final MAS system	[164]
Anylogic	Java	Suitable for agent modelling, effective user interface, object-oriented structure, pre-built libraries and objects, one free educational version and expert versions needs to be purchased	[165]

widely used in most instantiations of MAS. Details of ACLs and their coding are beyond the scope of this paper and readers are referred to [7], [152], and [153] for more details.

VII. MODELING AND SIMULATION ENVIRONMENTS FOR AGENT-BASED SYSTEMS

This section outlines multiple modeling and evaluation methods used to analyze performance metrics, which vary depending on the MAS application and goal, of the designed agent-based system compared to the state-of-the-art. In the following we outline three fundamental evaluation methods:

- **Java Agent Development framework (JADE):** JADE is among the most widely used simulators in MAS. The popularity of JADE stems from the following features: i) it is Java based and benefits from third-party libraries, ii) it is written based on FIPA standard (see Sec VI), iii) it supports simulating distributed systems, iv) it has a graphical interface for designing MAS, v) it hides MAS complexity from the designer, vi) it is open-source, and vii) it can be linked to Matlab [158], [159]. The authors in [159] provide a complete instruction to MAS implementation using JADE.
- **GAMA**: GAMA is a modeling and simulation platform for building agent-based systems [160]. GAMA has a number of advantages including: i) it can be used to model/simulate MAS in any application, ii) it supports GAML, a high-level and intuitive agent-based language, that can be readily used to simulate MAS, and iii) it supports large scale MAS which are comprised of millions of agents.
- **Matlab:** Matlab is used to study the performance of MAS especially with respect to mathematical complex evaluations [161]. Additionally, Matlab is linkable to JADE for further studies on MAS performance [158].
- **Mathematical analysis:** MAS are representable using graphs (see Section III). Consequently, using mathematical analysis are employed to evaluate MAS performance.

The aforementioned evaluation methods are the most widespread methods to study MAS. However, there exists other infrequent evaluation methods which are discussed in Table 4.

The outlined evaluation methods are particularly designed for MAS. Due to wide applicability, particular evaluation methods which are used to analyze systems in a specific application can be employed to evaluate the performance of an agent-based system designed to address challenges in that application. For instance, to study the performance of an agent-based computer network, NS2 [166], which is a network simulator, can be employed for evaluation purpose. As another example, Sinalgo [167] can be used in distributed systems (similar to systems in Section IV-A).

VIII. CONCLUSION

In this survey, we proposed a high-level comprehensive discussion regarding diverse aspects of MAS which helps newcomers to grasp basic concepts of MAS, study existing applications in multiple disciplines, the challenges in developing MAS, and the methods to study MAS performance. We first provided a definition of agents and MAS and outlined their key features. We then discussed the main applications and challenges of MAS while introducing references for further studies. Next, we discussed communications between agents and concluded the paper by a discussion on evaluation methods to analyze the effectiveness of an agent-based system. We expect this article to serve as an insightful and comprehensive resource on MAS for researchers and practitioners in the area.

REFERENCES

[1] M. Wooldridge, *An Introduction to Multiagent Systems*. New York, NY, USA: Wiley, 2009.

[2] A. H. Bond and L. Gasser, *Readings in Distributed Artificial Intelligence*. San Mateo, CA, USA: Morgan Kaufmann, 2014.

[3] S. Shamshirband, N. B. Anuar, M. L. M. Kiah, and A. Patel, "An appraisal and design of a multi-agent system based cooperative wireless intrusion detection computational intelligence technique," *Eng. Appl. Artif. Intell.*, vol. 26, no. 9, pp. 2105–2127, 2013.

[4] A.-M. Zou, K. D. Kumar, and Z.-G. Hou, "Distributed consensus control for multi-agent systems using terminal sliding mode and Chebyshev neural networks," *Int. J. Robust Nonlinear Control*, vol. 23, no. 3, pp. 334–357, Feb. 2013.

[5] M. H. Bowling, "Convergence and no-regret in multiagent learning," in *Proc. NIPS*, 2004, pp. 209–216.

[6] A. Zidan *et al.*, "Fault detection, isolation, and service restoration in distribution systems: State-of-the-art and future trends," *IEEE Trans. Smart Grid*, vol. 8, no. 5, pp. 2170–2185, Sep. 2016.

[7] P. Balaji and D. Srinivasan, "An introduction to multi-agent systems," in *Innovations in Multi-Agent Systems and Applications*. Berlin, Germany: Springer, 2010, pp. 1–27.

[8] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, vol. 25. Egnlewood Cliffs, NJ, USA: Prentice-Hall, 1995, p. 27.

[9] L. C. Jain and D. Srinivasan, *Innovations in Multi-Agent Systems and Application*. Springer, 2010.

[10] D. Ye, M. Zhang, and A. V. Vasilakos, "A survey of self-organization mechanisms in multiagent systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 441–461, Mar. 2017.

[11] A. P. Garcia, J. Oliver, and D. Gosch, "An intelligent agent-based distributed architecture for smart-grid integrated network management," in *Proc. IEEE 35th Conf. Local Comput. Netw. (LCN)*, Oct. 2010, pp. 1013–1018.

[12] S. D. J. McArthur *et al.*, "Multi-agent systems for power engineering applications—Part I: Concepts, approaches, and technical challenges," *IEEE Trans. Power Syst.*, vol. 22, no. 4, pp. 1743–1752, Nov. 2007.

[13] H. Rezaee and F. Abdollahi, "Average consensus over high-order multiagent systems," *IEEE Trans. Autom. Control*, vol. 60, no. 11, pp. 3047–3052, Nov. 2015.

[14] L. Ma, H. Min, S. Wang, Y. Liu, and S. Liao, "An overview of research in distributed attitude coordination control," *IEEE/CAA J. Autom. Sinica*, vol. 2, no. 2, pp. 121–133, Apr. 2015.

[15] J. Qi, R. Vazquez, and M. Krstic, "Multi-agent deployment in 3-D via PDE control," *IEEE Trans. Autom. Control*, vol. 60, no. 4, pp. 891–906, Apr. 2015.

[16] R. Merris, "Laplacian matrices of graphs: A survey," *Linear Algebra Appl.*, vols. 197–198, pp. 143–176, Jan./Feb. 1994.

[17] C. Godsil and G. F. Royle, *Algebraic Graph Theory*, vol. 207. Springer, 2013

[18] H. F. Ahmad, "Multi-agent systems: Overview of a new paradigm for distributed systems," in *Proc. 7th IEEE Int. Symp. High Assurance Syst. Eng.*, Oct. 2002, pp. 101–107.

[19] Q. Liu, L. Gao, and P. Lou, "Resource management based on multi-agent technology for cloud manufacturing," in *Proc. Int. Conf. Electron., Commun. Control (ICECC)*, Sep. 2011, pp. 2821–2824.

[20] F. M. Al-Shrouf, "Facilitator agent design pattern of procurement business systems," in *Proc. 32nd Annu. IEEE Int. Comput. Softw. Appl. (COMPSAC)*, Aug. 2008, pp. 505–510.

[21] J. Fu and J. Wang, "Adaptive coordinated tracking of multi-agent systems with quantized information," *Syst. Control Lett.*, vol. 74, pp. 115–125, Dec. 2014.

[22] A. González-Pardo, P. Varona, D. Camacho, and F. de Borja Rodriguez Ortiz, "Communication by identity discrimination in bio-inspired multi-agent systems," *Concurrency Comput., Pract. Exper.*, vol. 24, no. 6, pp. 589–603, 2012.

[23] S. Li, H. Du, and X. Lin, "Finite-time consensus algorithm for multi-agent systems with double-integrator dynamics," *Automatica*, vol. 47, no. 8, pp. 1706–1712, Aug. 2011.

[24] D. Angeli and P.-A. Bliman, "Extension of a result by moreau on stability of leaderless multi-agent systems," in *Proc. 44th IEEE Conf. Decision Control*, Dec. 2005, pp. 759–764.

[25] Y. Zhao, G. Wen, Z. Duan, X. Xu, and G. Chen, "A new observer-type consensus protocol for linear multi-agent dynamical systems," *Asian J. Control*, vol. 15, no. 2, pp. 571–582, 2013.

[26] D. M. Zhang, L. Meng, X. G. Wang, and L. L. Ou, "Linear quadratic regulator control of multi-agent systems," *Optim. Control Appl. Methods*, vol. 36, no. 1, pp. 45–59, 2015.

[27] Z. Li, W. Ren, X. Liu, and M. Fu, "Consensus of multi-agent systems with general linear and Lipschitz nonlinear dynamics using distributed adaptive protocols," *IEEE Trans. Autom. Control*, vol. 58, no. 7, pp. 1786–1791, Jul. 2013.

[28] H. Du, Y. He, and Y. Cheng, "Finite-time synchronization of a class of second-order nonlinear multi-agent systems using output feedback control," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 6, pp. 1778–1788, Jun. 2014.

[29] Y. Kim and E. T. Matson, "A realistic decision making for task allocation in heterogeneous multi-agent systems," *Proc. Comput. Sci.*, vol. 94, pp. 386–391, Jan. 2016.

[30] S. Khodaverdian, "On the synchronization of linear heterogeneous multi-agent systems in cycle-free communication networks," in *Proc. 6th Int. Conf. Modelling, Identificat. Control (ICMIC)*, Dec. 2014, pp. 190–195.

[31] J. Vrancken and M. dos Santos Soares, "A real-life test bed for multi-agent monitoring of road network performance," *Int. J. Critical Infrastruct.*, vol. 5, no. 4, pp. 357–367, 2009.

[32] G. Miao and Q. Ma, "Group consensus of the first-order multi-agent systems with nonlinear input constraints," *Neurocomputing*, vol. 161, pp. 113–119, Aug. 2015.

[33] G. Wen, G. Hu, W. Yu, J. Cao, and G. Chen, "Consensus tracking for higher-order multi-agent systems with switching directed topologies and occasionally missing control inputs," *Syst. Control Lett.*, vol. 62, no. 12, pp. 1151–1158, 2013.

[34] L. Gao, X. Liao, H. Li, and G. Chen, "Event-triggered control for multi-agent systems with general directed topology and time delays," *Asian J. Control*, vol. 18, no. 3, pp. 945–953 2015.

[35] B. Liu, H. Su, R. Li, D. Sun, and W. Hu, "Switching controllability of discrete-time multi-agent systems with multiple leaders and time-delays," *Appl. Math. Comput.*, vol. 228, pp. 571–588, Feb. 2014.

[36] H. Du, S. Li, and S. Ding, "Bounded consensus algorithms for multi-agent systems in directed networks," *Asian J. Control*, vol. 15, no. 1, pp. 282–291, 2013.

[37] Z. Liu, X. You, H. Yang, and L. Zhao, "Leader-following consensus of heterogeneous multi-agent systems with packet dropout," *Int. J. Control, Autom. Syst.*, vol. 13, no. 5, pp. 1067–1075, 2015.

[38] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[39] G. Guo, L. Ding, and Q.-L. Han, "A distributed event-triggered transmission strategy for sampled-data consensus of multi-agent systems," *Automatica*, vol. 50, no. 5, pp. 1489–1496, May 2014.

[40] H. Li, C. Ming, S. Shen, and W. K. Wong, "Event-triggered control for multi-agent systems with randomly occurring nonlinear dynamics and time-varying delay," *J. Franklin Inst.*, vol. 351, no. 5, pp. 2582–2599, 2014.

[41] C. Wang, G. Xie, and M. Cao, "Controlling anonymous mobile agents with unidirectional locomotion to form formations on a circle," *Automatica*, vol. 50, no. 4, pp. 1100–1108, 2014.

[42] C. Wang, G. Xie, and M. Cao, "Forming circle formations of anonymous mobile agents with order preservation," *IEEE Trans. Autom. Control*, vol. 58, no. 12, pp. 3248–3254, Dec. 2013.

[43] Q. Song, F. Liu, H. Su, and A. V. Vasilakos, "Semi-global and global containment control of multi-agent systems with second-order dynamics and input saturation," *Int. J. Robust Nonlinear Control*, vol. 26, no. 16, pp. 3460–3480, 2016.

[44] H. Yang, B. Jiang, V. Cocquempot, and H. Zhang, "Stabilization of switched nonlinear systems with all unstable modes: Application to multi-agent systems," *IEEE Trans. Autom. Control*, vol. 56, no. 9, pp. 2230–2235, Sep. 2011.

[45] G.-S. Han, Z.-H. Guan, J. Chen, D.-X. He, and M. Chi, "Multi-tracking of first order multi-agent networks via self-triggered control," *Asian J. Control*, vol. 17, no. 4, pp. 1320–1329, 2015.

[46] W. Zhang, Y. Liu, J. Lu, and J. Cao, "A novel consensus algorithm for second-order multi-agent systems without velocity measurements," *Int. J. Robust Nonlinear Control*, vol. 27, no. 15, pp. 2510–2528, 2017.

[47] W. Ren, K. Moore, and Y. Chen, "High-order consensus algorithms in cooperative vehicle systems," in *Proc. IEEE Int. Conf. Netw., Sens. Control (ICNSC)*, Apr. 2006, pp. 457–462.

[48] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial Internet of Things," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2015, pp. 1–6.

[49] J. Bajo, F. De la Prieta, J. M. Corchado, and S. Rodríguez, "A low-level resource allocation in an agent-based cloud computing platform," *Appl. Soft Comput.*, vol. 48, pp. 716–728, Nov. 2016.

[50] J. Fiosina and M. Fiosins, "Density-based clustering in cloud-oriented collaborative multi-agent systems," in *Proc. Int. Conf. Hybrid Artif. Intell. Syst.*, 2013, pp. 639–648.

[51] J. O. Gutierrez-Garcia and K. M. Sim, "Agent-based cloud bag-of-tasks execution," *J. Syst. Softw.*, vol. 104, pp. 17–31, Jun. 2015.

[52] A. Singh, D. Juneja, and M. Malhotra, "Autonomous agent based load balancing algorithm in cloud computing," *Proc. Comput. Sci.*, vol. 45, pp. 832–841, Jan. 2015.

[53] K. Govinda and E. Sathiyamoorthy, "Agent based security for cloud computing using obfuscation," *Proc. Eng.*, vol. 38, pp. 125–129, Jan. 2012.

[54] R. Nikbazm and M. Ahmadi, "Agent-based resource discovery in cloud computing using bloom filters," in *Proc. 4th Int. eConf. Comput. Knowl. Eng. (ICCKE)*, Oct. 2014, pp. 352–357.

[55] F. Hou and X. Mao, "Cross-clouds services autonomic management approach based on self-organizing multi-agent technology," *Concurrency Comput., Pract. Exper.*, vol. 28, no. 11, pp. 3213–3237, 2016.

[56] K. M. Sim, "Agent-based cloud computing," *IEEE Trans. Serv. Comput.*, vol. 5, no. 4, pp. 564–577, Oct. 2012.

[57] Y. Jiang and J. C. Jiang, "Understanding social networks from a multiagent perspective," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 10, pp. 2743–2759, Oct. 2014.

[58] M. Gatti *et al.*, "Large-scale multi-agent-based modeling and simulation of microblogging-based online social network," in *Proc. Int. Workshop Multi-Agent Syst. Agent-Based Simulation*, 2013, pp. 17–33.

[59] L. Ma and Y. Zhang, "Hierarchical social network analysis using multi-agent systems: A school system case," in *Proc. IEEE Int. Conf. Syst., Man Cybern. (SMC)*, Oct. 2014, pp. 1412–1419.

[60] V. Gorodetski and I. Kotenko, "The multi-agent systems for computer network security assurance: Frameworks and case studies," in *Proc. IEEE Int. Conf. Artif. Intell. Syst. (ICAIS)*, Sep. 2002, pp. 297–302.

[61] L. Mechtri, F. D. Tolba, and S. Ghanemi, "Masid: Multi-agent system for intrusion detection in MANET," in *Proc. 9th Int. Conf. Inf. Technol., New Generat. (ITNG)*, Apr. 2012, pp. 65–70.

[62] S. Sarika and V. Paul, "AgentTab: An agent based approach to detect tabnabbing attack," *Proc. Comput. Sci.*, vol. 46, pp. 574–581, Jan. 2015.

[63] A. Dorri, "An EDRI-based approach for detecting and eliminating cooperative black hole nodes in MANET," *Wireless Netw.*, vol. 23, no. 6, pp. 1767–1778, 2017.

[64] G. Di Caro and M. Dorigo, "An adaptive multi-agent routing algorithm inspired by ants behavior," in *Proc. 5th Annu. Austral. Conf. Parallel Real-Time Syst. (PART)*, 1998, pp. 261–272.

[65] R. Claes, T. Holvoet, and D. Weyns, "A decentralized approach for anticipatory vehicle routing using delegate multiagent systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 364–373, Feb. 2011.

[66] M. Bendjima and M. Feham, "Multi-agent system for a reliable routing in WSN," in *Proc. Sci. Inf. Conf. (SAI)*, Jul. 2015, pp. 1412–1419.

[67] S. Manvi and M. Kakkasageri, "Multicast routing in mobile ad hoc networks by using a multiagent system," *Inf. Sci.*, vol. 178, no. 6, pp. 1611–1628, 2008.

[68] G. Dudek, M. R. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for multi-agent robotics," *Auto. Robot.*, vol. 3, no. 4, pp. 375–397, 1996.

[69] C. G. Cena, P. F. Cardenas, R. S. Pazmino, L. Puglisi, and R. A. Santonja, "A cooperative multi-agent robotics system: Design and modelling," *Expert Syst. Appl.*, vol. 40, no. 12, pp. 4737–4748, 2013.

[70] Y. Duan, B. X. Cui, and X. H. Xu, "A multi-agent reinforcement learning approach to robot soccer," *Artif. Intell. Rev.*, vol. 38, no. 3, pp. 193–211, 2012.

[71] J. Ota, "Multi-agent robot systems as distributed autonomous systems," *Adv. Eng. Informat.*, vol. 20, no. 1, pp. 59–70, 2006.

[72] P. Iñigo-Blasco, F. Diaz-del-Rio, M. C. Romero-Ternero, D. Cagigas-Muñiz, and S. Vicente-Diaz, "Robotics software frameworks for multi-agent robotic systems development," *Robot. Auto. Syst.*, vol. 60, no. 6, pp. 803–821, 2012.

[73] A. Soriano, E. J. Bernabeu, A. Valera, and M. Vallès, "Multi-agent systems platform for mobile robots collision avoidance," in *Proc. Int. Conf. Practical Appl. Agents Multi-Agent Syst.*, 2013, pp. 320–323.

[74] D. Helbing, "Agent-based modeling," in *Social self-Organization*. Berlin, Germany: Springer, 2012, pp. 25–70.

[75] R. Domínguez, S. Cannella, and J. M. Framinan, "SCOPE: A multi-agent system tool for supply chain network analysis," in *Proc. IEEE Int. Conf. Comput. (EUROCON)*, Sep. 2015, pp. 1–5.

[76] O. Wangapisit, E. Taniguchi, J. S. Teo, and A. G. Qureshi, "Multi-agent systems modelling for evaluating joint delivery systems," *Proc.-Social Behavioral Sci.*, vol. 125, pp. 472–483, Mar. 2014.

[77] P. Ringler, D. Keles, and W. Fichtner, "Agent-based modelling and simulation of smart electricity grids and markets—A literature review," *Renew. Sustain. Energy Rev.*, vol. 57, pp. 205–215, 2016.

[78] L. Chen, "Agent-based modeling in urban and architectural research: A brief literature review," *Frontiers Archit. Res.*, vol. 1, no. 2, pp. 166–177, 2012.

[79] M. Khayyat and A. Awasthi, "An intelligent multi-agent based model for collaborative logistics systems," *Transp. Res. Procedia*, vol. 12, pp. 325–338, Jan. 2016.

[80] K. Hager, J. Rauh, and W. Rid, "Agent-based modeling of traffic behavior in growing metropolitan areas," *Transp. Res. Procedia*, vol. 10, pp. 306–315, Jan. 2015.

[81] O. van Pruissen, A. van der Togt, and E. Werkman, "Energy efficiency comparison of a centralized and a multi-agent market based heating system in a field test," *Energy Procedia*, vol. 62, pp. 170–179, Jan. 2014.

[82] J. Cai, D. Kim, R. Jaramillo, J. E. Braun, and J. Hu, "A general multi-agent control approach for building energy system optimization," *Energy Buildings*, vol. 127, pp. 337–351, Sep. 2016.

[83] R. Yang and L. Wang, "Development of multi-agent system for building energy and comfort management based on occupant behaviors," *Energy Buildings*, vol. 56, pp. 1–7, Jan. 2013.

[84] C. P. Nguyen and A. J. Flueck, "Agent based restoration with distributed energy storage support in smart grids," *IEEE Trans. Smart Grid*, vol. 3, no. 2, pp. 1029–1038, Jun. 2012.

[85] P. Vytelingum, T. D. Voice, S. D. Ramchurn, A. Rogers, and N. R. Jennings, "Agent-based micro-storage management for the smart grid," in *Proc. 9th Int. Conf. Auto. Agents Multiagent Syst.*, 2010, pp. 39–46.

[86] G. H. Merabet *et al.*, "Applications of multi-agent systems in smart grids: A survey," in *Proc. Int. Conf. Multimedia Comput. Syst. (ICMCS)*, 2014, pp. 1088–1094.

[87] W. Yu, G. Chen, M. Cao, and J. Kurths, "Second-order consensus for multiagent systems with directed topologies and nonlinear dynamics," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 3, pp. 881–891, Jun. 2010.

[88] B. Zhou and X. Liao, "Leader-following second-order consensus in multi-agent systems with sampled data via pinning control," *Nonlinear Dyn.*, vol. 78, no. 1, pp. 555–569, 2014.

[89] X.-H. Wang and H.-B. Ji, "Leader-follower consensus for a class of nonlinear multi-agent systems," *Int. J. Control, Autom. Syst.*, vol. 10, no. 1, pp. 27–35, 2012.

[90] W. He, G. Chen, Q.-L. Han, and F. Qian, "Network-based leader-following consensus of nonlinear multi-agent systems via distributed impulsive control," *Inf. Sci.*, vol. 380, pp. 145–158, Feb. 2015.

[91] S. M. Noorbakhsh and J. Ghaisari, "Event-based consensus controller for linear multi-agent systems over directed communication topologies: A co-design approach," *Asian J. Control*, vol. 18, no. 5, pp. 1934–1939 2016.

[92] Z. Liu, Z. Chen, and Z. Yuan, "Event-triggered average-consensus of multi-agent systems with weighted and direct topology," *J. Syst. Sci. Complex.*, vol. 25, no. 5, pp. 1–11, 2012.

[93] S. Zhai and X.-S. Yang, "Consensus of second-order multi-agent systems with nonlinear dynamics and switching topology," *Nonlinear Dyn.*, vol. 77, no. 4, pp. 1667–1675, 2014.

[94] W. Hu, L. Liu, and G. Feng, "Consensus of linear multi-agent systems by distributed event-triggered strategy," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 148–157, Jan. 2016.

[95] G. Rohbogner, S. Fey, P. Benoit, C. Wittwer, and A. Christ, "Design of a multiagent-based voltage control system in peer-to-peer networks for smart grids," *Energy Technol.*, vol. 2, no. 1, pp. 107–120, 2014.

[96] X. Wang and G.-H. Yang, "Distributed reliable H_∞ consensus control for a class of multi-agent systems under switching networks: A topology-based average dwell time approach," *Int. J. Robus. Nonlinear Control*, vol. 26, no. 13, pp. 2767–2787, 2015.

[97] Z. Feng and G. Hu, "Passivity-based consensus for linear multi-agent systems under switching topologies," *Control Theory Technol.*, vol. 12, no. 3, pp. 304–316, 2014.

[98] J.-W. Yi, Y.-W. Wang, J.-W. Xiao, and Y. Chen, "Consensus in second-order Markovian jump multi-agent systems via impulsive control using sampled information with heterogenous delays," *Asian J. Control*, vol. 8, no. 5, pp. 1940–1949, 2015.

[99] J. Wu and Y. Shi, "Average consensus in multi-agent systems with time-varying delays and packet losses," in *Proc. Amer. Control Conf. (ACC)*, 2012, pp. 1579–1584.

[100] K. You and L. Xie, "Coordination of discrete-time multi-agent systems via relative output feedback," *Int. J. Robust Nonlinear Control*, vol. 21, no. 13, pp. 1587–1605, 2011.

[101] K. You and L. Xie, "Network topology and communication data rate for consensusability of discrete-time multi-agent systems," *IEEE Trans. Autom. Control*, vol. 56, no. 10, pp. 2262–2275, Oct. 2011.

[102] H. Zhao and J. H. Park, "Group consensus of discrete-time multi-agent systems with fixed and stochastic switching topologies," *Nonlinear Dyn.*, vol. 77, no. 4, pp. 1297–1307, 2014.

[103] Y. Zhang and Y.-P. Tian, "Maximum allowable loss probability for consensus of multi-agent systems over random weighted lossy networks," *IEEE Trans. Autom. Control*, vol. 57, no. 8, pp. 2127–2132, Aug. 2012.

[104] D. Ding, Z. Wang, D. W. C. Ho, and G. Wei, "Observer-based event-triggering consensus control for multiagent systems with lossy sensors and cyber-attacks," *IEEE Trans. Cybern.*, vol. 47, no. 8, pp. 1936–1947, Aug. 2017.

[105] X. Xu, L. Liu, and G. Feng, "Consensus of single integrator multi-agent systems with directed topology and communication delays," *Control Theory Technol.*, vol. 14, no. 1, pp. 21–27, 2016.

[106] W. Lu and T. Chen, "New approach to synchronization analysis of linearly coupled ordinary differential systems," *Phys. D, Nonlinear Phenomena*, vol. 213, no. 2, pp. 214–230, 2006.

[107] B. Tian, Z. Zuo, and H. Wang, "Leader–follower fixed-time consensus of multi-agent systems with high-order integrator dynamics," *Int. J. Control*, vol. 90, no. 7, pp. 1420–1427, 2017.

[108] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Trans. Autom. Control*, vol. 51, no. 3, pp. 401–420, Mar. 2006.

[109] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 25–34, Jul. 1987.

[110] M.-C. Fan, H.-T. Zhang, and M. Wang, "Bipartite flocking for multi-agent systems," *Commun. Nonlinear Sci. Numer. Simulation*, vol. 19, no. 9, pp. 3313–3322, 2014.

[111] S. Liu, L. Xie, and H. Zhang, "Containment control of multi-agent systems by exploiting the control inputs of neighbors," *Int. J. Robust Nonlinear Control*, vol. 24, no. 17, pp. 2803–2818, Nov. 2014.

[112] B. Liu, T. Chu, L. Wang, Z. Zuo, G. Chen, and H. Su, "Controllability of switching networks of multi-agent systems," *Int. J. Robust Nonlinear Control*, vol. 22, no. 6, pp. 630–644, 2012.

[113] S. Su, Z. Lin, and A. Garcia, "Distributed synchronization control of multiagent systems with unknown nonlinearities," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 325–338, Jan. 2016.

[114] H. Yu and P. J. Antsaklis, "Formation control of multi-agent systems with connectivity preservation by using both event-driven and time-driven communication," in *Proc. IEEE 51st Annu. Conf. Decision Control (CDC)*, Dec. 2012, pp. 7218–7223.

[115] L. Wang, X. Wang, and X. Hu, "Connectivity preserving flocking without velocity measurement," *Asian J. Control*, vol. 15, no. 2, pp. 521–532, 2013.

[116] Z. Lin, L. Wang, Z. Han, and M. Fu, "Distributed formation control of multi-agent systems using complex Laplacian," *IEEE Trans. Autom. Control*, vol. 59, no. 7, pp. 1765–1777, Jul. 2014.

[117] M. Jafarian, E. Vos, C. De Persis, A. J. Van Der Schaft, and J. M. Scherpen, "Formation control of a multi-agent system subject to coulomb friction," *Automatica*, vol. 61, pp. 253–262, Nov. 2015.

[118] Y. Q. Chen and Z. Wang, "Formation control: A review and a new consideration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Aug. 2005, pp. 3181–3186.

[119] Y. Liu and Z. Geng, "Finite-time formation control for linear multi-agent systems: A motion planning approach," *Syst. Control Lett.*, vol. 85, pp. 54–60, Nov. 2015.

[120] Y. Xia, X. Na, Z. Sun, and J. Chen, "Formation control and collision avoidance for multi-agent systems based on position estimation," *ISA Trans.*, vol. 61, pp. 287–296, Mar. 2016.

[121] H. Li, J. Peng, W. Liu, K. Gao, and Z. Huang, "A novel communication-aware formation control strategy for dynamical multi-agent systems," *J. Franklin Inst.*, vol. 352, no. 9, pp. 3701–3715, 2015.

[122] R. A. C. Bianchi, M. F. Martins, C. H. C. Ribeiro, and A. H. R. Costa, "Heuristically-accelerated multiagent reinforcement learning," *IEEE Trans. Cybern.*, vol. 44, no. 2, pp. 252–265, Feb. 2014.

[123] B. D. O. Anderson, B. Fidan, C. Yu, and D. Walle, "UAV formation control: Theory and application," in *Recent Advances in Learning and Control*. London, U.K.: Springer, 2008, pp. 15–33.

[124] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Mach. Learn.*, vol. 3, nos. 2–3, pp. 95–99, 1988.

[125] M. Bowling and M. Veloso, "Multiagent learning using a variable learning rate," *Artif. Intell.*, vol. 136, no. 2, pp. 215–250, Apr. 2002.

[126] D. Chakraborty and P. Stone, "Multiagent learning in the presence of memory-bounded agents," *Auto. Agents Multi-Agent Syst.*, vol. 28, no. 2, pp. 182–213, 2014.

[127] K.-S. Hwang, W.-C. Jiang, and Y.-J. Chen, "Model learning and knowledge sharing for a multiagent system with Dyna-Q learning," *IEEE Trans. Cybern.*, vol. 45, no. 5, pp. 978–990, May 2015.

[128] J. Ni, M. Liu, L. Ren, and S. X. Yang, "A multiagent Q-learning-based optimal allocation approach for urban water resource management system," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 1, pp. 204–214, Jan. 2014.

[129] P. Valencia, P. Lindsay, and R. Jurdak, "Distributed genetic evolution in WSN," in *Proc. 9th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, Apr. 2010, pp. 13–23.

[130] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, MA, USA: MIT Press, 1992.

[131] M. R. Davoodi, K. Khorasani, H. A. Talebi, and H. R. Momeni, "Distributed fault detection and isolation filter design for a network of heterogeneous multiagent systems," *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 3, pp. 1061–1069, May 2014.

[132] X. Liu, X. Gao, and J. Han, "Robust unknown input observer based fault detection for high-order multi-agent systems with disturbances," *ISA Trans.*, vol. 61, pp. 15–28, Mar. 2016.

[133] M. R. Davoodi, N. Meskin, and K. Khorasani, "Simultaneous fault detection and consensus control design for a network of multi-agent systems," in *Proc. Eur. Control Conf. (ECC)*, 2014, pp. 575–581.

[134] S. X. Ding, *Model-Based Fault Diagnosis Techniques: Design Schemes, Algorithms and Tools*. Springer, 2008.

[135] X. Liu, X. Gao, and J. Han, "Observer-based fault detection for high-order nonlinear multi-agent systems," *J. Franklin Inst.*, vol. 353, no. 1, pp. 72–94, 2016.

[136] S. X. Ding, "Integrated design of feedback controllers and fault detectors," *Annu. Rev. Control*, vol. 33, no. 2, pp. 124–135, 2009.

[137] N. K. Krothapalli and A. V. Deshmukh, "Distributed task allocation in multi-agent systems," in *Proc. IIE Annu. Conf.*, 2002, p. 1.

[138] D. S. dos Santos and A. L. C. Bazzan, "Distributed clustering for group formation and task allocation in multiagent systems: A swarm intelligence approach," *Appl. Soft Comput.*, vol. 12, no. 8, pp. 2123–2131, Aug. 2012.

[139] Y. Jiang and Z. Li, "Locality-sensitive task allocation and load balancing in networked multiagent systems: Talent versus centrality," *J. Parallel Distrib. Comput.*, vol. 71, no. 6, pp. 822–836, 2011.

[140] F. Rahimzadeh, L. M. Khanli, and F. Mahan, "High reliable and efficient task allocation in networked multi-agent systems," *Auto. Agents Multi-Agent Syst.*, vol. 29, no. 6, pp. 1023–1040, 2015.

[141] C. Lin, Z. Lin, R. Zheng, G. Yan, and G. Mao, "Distributed source localization of multi-agent systems with bearing angle measurements," *IEEE Trans. Autom. Control*, vol. 61, no. 4, pp. 1105–1110, Apr. 2016.

[142] M. A. Fallah, R. P. Malhame, and F. Martinelli, "Distributed estimation and control for large population stochastic multi-agent systems with coupling in the measurements," in *Proc. Eur. Control Conf. (ECC)*, 2013, pp. 4353–4358.

[143] B. Horling and V. Lesser, "A survey of multi-agent organizational paradigms," *Knowl. Eng. Rev.*, vol. 19, no. 4, pp. 281–316, 2004.

[144] A. Esmaeili, N. Mozayani, M. R. J. Motlagh, and E. T. Matson, "The impact of diversity on performance of holonic multi-agent systems," *Eng. Appl. Artif. Intell.*, vol. 55, pp. 186–201, Oct. 2016.

[145] A. Damba and S. Watanabe, "Hierarchical control in a multiagent system," in *Proc. 2nd Int. Conf. Innovative Comput., Inf. Control (ICICIC)*, Sep. 2007, p. 111.

[146] C. H. Brooks and E. H. Durfee, "Congregation formation in multiagent systems," *Auto. Agents Multi-Agent Syst.*, vol. 7, nos. 1–2, pp. 145–170, Jul. 2003. [Online]. Available: https://doi.org/10.1023/A:1024133006761

[147] E. Argente *et al.*, "Supporting agent organizations," in *Proc. Int. Central Eastern Eur. Conf. Multi-Agent Syst.*, 2007, pp. 236–245.

[148] Y. Jung, M. Kim, A. Masoumzadeh, and J. B. Joshi, "A survey of security issue in multi-agent systems," *Artif. Intell. Rev.*, vol. 37, no. 3, pp. 239–260, 2012.

[149] R. C. Cavalcante, I. I. Bittencourt, A. P. da Silva, M. Silva, E. Costa, and R. Santos, "A survey of security in multi-agent systems," *Expert Syst. Appl.*, vol. 39, no. 5, pp. 4835–4846, 2012.

[150] X. Wang, M. Maghami, and G. Sukthankar, "Leveraging network properties for trust evaluation in multi-agent systems," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol. (WI-IAT)*, vol. 2. Aug. 2011, pp. 288–295.

[151] H. Yu, Z. Shen, C. Leung, C. Miao, and V. R. Lesser, "A survey of multi-agent trust management systems," *IEEE Access*, vol. 1, pp. 35–50, 2013.

[152] G. Kołaczek, "Social network analysis based approach to trust modeling for autonomous multi-agent systems," in *Agent and Multi-Agent Technology for Internet and Enterprise Systems*. Berlin, Germany: Springer, 2010, pp. 137–156.

[153] B. J. Austin, V. Heine, and L. Sham, "General theory of pseudopotentials," *Phys. Rev. J. Arch.*, vol. 127, no. 1, p. 276, 1962.

[154] R. Kibble, "Speech acts, commitment and multi-agent communication," *Comput. Math. Org. Theory*, vol. 12, nos. 2–3, pp. 127–145, 2006.

[155] M. Colombetti and M. Verdicchio, "An analysis of agent speech acts as institutional actions," in *Proc. 1st Int. Joint Conf. Auto. Agents Multiagent Syst.*, 2002, pp. 1157–1164.

[156] S. Li and M. M. Kokar, "Agent communication language," in *Flexible Adaptation in Cognitive Radios*. New York, NY, USA: Springer, 2013, pp. 37–44.

[157] D. Juneja, A. Jagga, and A. Singh, "A review of fipa standardized agent communication language and interaction protocols," *J. Netw. Commun. Emerg. Technol.*, vol. 5, no. 2, pp. 179–191, 2015.

[158] Y. S. F. Eddy, H. B. Gooi, and S. X. Chen, "Multi-agent system for distributed management of microgrids," *IEEE Trans. Power Syst.*, vol. 30, no. 1, pp. 24–34, Jan. 2015.

[159] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems With JADE*. New York, NY, USA: Wiley, 2007, vol. 7.

[160] (2018). *GAMA*. Accessed: Apr. 1, 2018. [Online]. Available: http://gama-platform.org/

[161] D. Panasetsky and N. Tomin, "Using of neural network technology and multi-agent systems to preventing large-scale emergencies in electric power systems," in *Proc. 4th Int. Youth Conf. Energy (IYCE)*, Jun. 2013, pp. 1–8.

[162] (2017). *Repast*. Accessed: Apr. 19, 2017. [Online]. Available: https://repast.github.io/

[163] (2017). *MASON*. Accessed: Apr. 19, 2017. [Online]. Available: http://cs.gmu.edu/ eclab/projects/mason/, , [Online;

[164] (2017). *NetLogo*. Accessed: Apr. 19, 2017. [Online]. Available: https://ccl.northwestern.edu/netlogo/

[165] (2017). *AnyLogic*. Accessed: Apr. 19, 2017. [Online]. Available: http://www.anylogic.com/

[166] (2017). *NS2*. Accessed: Oct. 30, 2017. [Online]. Available: https://www.isi.edu/nsnam/ns/

[167] (2017). *Sinalgo*. Accessed: Apr. 19, 2017. [Online]. Available: http://www.disco.ethz.ch/projects/sinalgo/

ALI DORRI is currently pursuing the Ph.D. degree with the University of New South Wales, Sydney, Australia. He is also a Researcher with CSIRO, Australia. His research interests focus on enhancing security and privacy of the Internet of Things (IoT), smart vehicles, smart grids, energy management, smart cities, and healthcare. He is currently involved in optimizing and adopting blockchain for large-scale networks including IoT.

SALIL S. KANHERE (S'00–M'03–SM'11) received the M.S. and Ph.D. degrees in electrical engineering from Drexel University, Philadelphia. He is currently an Associate Professor with the School of Computer Science and Engineering, University of New South Wales, Sydney, Australia. He is also a Contributing Research Staff with Data61, CSIRO. His current research interests include Internet of Things, pervasive computing, crowdsourcing, and privacy and security. He has published over 150 peer-reviewed articles and delivered over 20 tutorials and keynote talks on these research topics. He is a Senior Member of the ACM. He was a recipient of the Humboldt Research Fellowship in 2014. He regularly serves on the organizing committee of a number of IEEE and ACM international conferences. He currently serves as an Area Editor for *Pervasive and Mobile Computing* and *Computer Communications*.

RAJA JURDAK (M'07–SM'11) received the Ph.D. degree in information and computer science from the University of California at Irvine, Irvine. He is currently a Senior Principal Research Scientist with CSIRO, where he leads the Distributed Sensing Systems Group. He is also an Honorary Professor with the University of Queensland and an Adjunct Professor with Macquarie University and James Cook University. He has over 120 peer-reviewed journal and conference publications, as well as a book *Wireless Ad Hoc and Sensor Networks: A Cross-Layer Design Perspective* (Springer, 2007). His current research interests focus on energy-efficiency, mobility, and security in networks. He regularly serves on the organizing and technical program committees of international conferences (DCOSS, RTSS, Sensapp, Percom, EWSN, and ICDCS).