Knowledge-Driven Wireless Networks with Artificial Intelligence: Design, Challenges and Opportunities

Haris Gacanin, Senior Member, IEEE
Nokia Bell Labs, Belgium

Abstract — This paper discusses technology challenges and opportunities to embrace artificial intelligence (AI) era in the design of wireless networks. We aim to provide readers with motivation and general methodology for adoption of AI in the context of next-generation networks. First, we discuss the rise of network intelligence and then, we introduce a brief overview of AI with machine learning (ML) and their relationship to self-organization designs. Finally, we discuss design of intelligent agent and its functions to enable knowledge-driven wireless networks with AI.

Index Terms — Artificial intelligence, wireless, optimization.

I. INTRODUCTION

We are now several years into explosion of machine learning (ML) in wireless networks, used to enrich decision-making by finding structures in data – knowledge discovery (i.e. “means to extract models that describe the user behavior and/or network performance”). With new designs of wireless networks complexity and dynamicity increases, network resources are scattered and diversity of network elements increases [1]. Such designs are expected to support high network adaptability with respect to such dynamic environment (e.g. sensing and perception of network performance and user behavior) and different network services [1].

Design of learning agents with ML techniques in wireless networks improves adaptability in dynamic environments by enriching decision-making [2] – [8]. Consider these examples, which have interesting challenges: supporting massive Internet-of-Things applications (i.e., a massive number of devices, sensors and actuators) give rise to problem of dynamic network planning; broadband indoor and outdoor wireless leads to problems with real-time radio resource management; ultra-reliable communications require low latency and high availability to support critical applications, where network should support real-time adjustments on latency and reliability in the orders of 99,999999%. Current strategies enhance decision-making by knowledge discovery, but do not enable autonomy by knowledge manipulation, where past experiences derived from interactions with environment are exploited to build new knowledge enabling full network autonomy. Building and exploiting knowledge combined with enhanced decision-making strategies paves a way toward autonomous self-X (self-configuration, self-optimization, self-healing, etc.) network space.

Autonomous wireless networks require full awareness of its complex and dynamic environment with the agent designed, not only through learning, but through broader disciplines of artificial intelligence (AI) such as perception, reasoning, acting, optimization and planning. This enables paradigm shift from knowledge-discovery toward knowledge-driven operation of all-the-time in the real-time optimized user experience. The contributions of this paper are:

• This paper presents methodology and motivation for design of autonomous – knowledge-driven – wireless network agent operation;

• The selection of AI discipline affects agent design choices. We point out conceptual differences between AI and ML to understand how the disciplines of AI build and exploit knowledge versus learning/decision-only approaches supported by ML techniques;

• We discuss practical design guidelines for AI-driven self-organization with its innovative features synthesizing reasoning with decision-making by knowledge management.

II. KNOWLEDGE-DISCOVERY IN WIRELESS NETWORKS

Noteworthy self-X survey in multi-tier networks is presented in [5]. The authors discuss up to date research concepts including different learning and decision-making techniques (Genetic algorithms, Swarm intelligence, Neural networks, Fuzzy systems, Markov and Bayesian games) to improve network efficiency. Another comprehensive survey on ML techniques in self-X space is presented in [3].

Work in [1] provides helpful discussions on applications of AI to network management and orchestration. A conceptual example of high-level reinforcement learning (RL) framework for traffic-aware energy management has been presented. The framework is described as abstract layer interacting with network elements (i.e. radio access network, virtual nodes, etc.) using open-source interfaces. Inspiring, cognition-based network (COBANET) is proposed to automate network operations at the system level with abstracted learning-architecture functions for the network virtualization problems. The proposed approach develops around the application of deep learning and probabilistic generative models for system-level learning and reconfiguration. To address collaboratively-deployed virtual network elements the concept of Future Intelligent Network (FINE) framework is introduced in [7]. The
The field of AI provides methods to design agent to autonomously interact with environment in a way that humans consider intelligent – includes all the characteristics of human cognitive abilities, e.g., planning, perceiving, reasoning, learning, and problem solving. Today, an applied AI (i.e., henceforth AI) is used to perform a range of human cognitive abilities with focus on learning and decision-making driven by human defined rules and constraints before “actual learning” – today mostly known as ML and predictive analytics.

AI thus defines a framework for knowledge manipulation (building new and exploiting already gained knowledge) through disciplines such as perception, reasoning (specifying what needs to be done, but not how) and acting [9], [13]. ML techniques, within the field of AI, enable machines to learn with little or no guidance at all. In principle, ML evolved from the study of pattern recognition and computational learning theory. Unlike the broad field of AI focusing on intelligent agent design, ML defines a learning function to enable environment-inspired predictions or decisions.

ML techniques are considered in communication systems to support decision-making in self-X features such as mobility load balancing, mobility robustness optimization, coverage and capacity optimization, inter-cell interference coordination, random access channel optimization, cell outage detection to name just a few [3]. For more detailed taxonomy on ML techniques in communication networks we point interested readers to [4] – [6] and references therein.

B. Handling Multi-Data Sources in Complex Environments

Wireless networks are highly heterogeneous with multi-vendor and multi-standard data sources leading to design challenges of open data interfaces. The correct type of data should be collected in correct intervals and amount (e.g., too fast or too slow collection, too many or too few parameters may not be good depending on a use case). For example, in dynamic environments with interference infrequent collection limits proper root-cause analysis [10]. This is because data has its own life-time while today it is managed by deterministic collection rules and requires more intelligent data manipulation function [11].

Unlike large-scale data analytics [2], for operation and management in real-time, only a set of parameters would be relevant for collection across different network layers and devices to optimize the network performance [10]. This is because in optimization of wireless networks we resolve a problem in limited geographical area, but most likely the problem is partially moved to or enhanced in adjacent areas. The data collection should be done in a non-intrusive fashion to avoid service interruptions or unnecessary signaling embedded within communication protocols [10].

Thus, AI should control and select data sources depending on the use case and user location.
Past experiences retains its internal state to encode beliefs about its environment unless it is designed with some built-in knowledge. The agent by its own experience, where initially the agent acts randomly, is autonomous to the extent that its behavior is determined without indication of the agent’s success. In this context, the relationship is called a condition-action rule written as: if percept condition then action. Previously described agent types represent the learning element as supervised, unsupervised and reinforcement learning. Learning tasks are characterized by the feedback given to the agent as supervised, unsupervised and reinforcement learning.

The agent program runs on architecture to generate mapping function \( f : P \rightarrow A \), maps from percepts \( P \) to actions space \( A \), where the percepts provide information about environment without indication of the agent’s success. In this context, the agent is autonomous to the extent that its behavior is determined by its own experience, where initially the agent acts randomly unless it is designed with some built-in knowledge. The agent retains its internal state to encode beliefs about its environment and ability to modify its beliefs by perception, reasoning and acting. The agent actions depend on the prior knowledge about the state of agent and its environment, interactions with the environment (including percepts of current state of environment and past experiences of previous actions and percepts), the goals or preferences over states of the world and abilities regarding actions that agent can carry out.

Next, we briefly outline basic types of agent programs [9], [13].

A. Agent types

Simple reflex agent reacts on the current percept of environment creating relations to select the action. Such a relationship is called a condition-action rule written as: if percept condition then action. This is the simplest form of agent, but limited to built-in knowledge by designer. The agent performs well in fully observable environment, where the correct decisions can be made based on current percept. Thus, the agent is blocked if current percept is not provided by designer. This is because the built-in knowledge maps directly from percepts to actions.

Model-based reflex agent retains an internal state (that depends on the percept history) to track not yet observed environment. The update of state requires to hardcode the information how the world evolves independently of the agent and the information how actions taken affect the world. This information about the world is called a model of the world. The challenge is to determine the current state exactly from a partially observable environment so that the agent makes the best guess of the state.

Goal-based agent is given a goal describing desirable situations as “happy” and “unhappy”. The agent combines information about a goal with the model of environment to select actions that achieve the goal. In complex environments, where sequence of actions is necessary, search and planning subfields of AI are developed to find action sequences that achieve the goal. Decision-making in this case is not condition-action rule. This is because we consider a result and satisfaction of a taken action with respect to given goal. In comparison with previous agents the goal based agent is more flexible since it’s behavior can easily be changed by specifying different goal.

Utility-based agent handleless decision-making in partially observable and stochastic environment. The agent utilizes a more general performance measure for comparison between different states according to exactly how happy they would make the agent. The agent assigns a utility of performance measure to any given sequence of environment states. The utility distinguishes between more and less desirable ways of reaching a goal. Similarly, like goal-based agents the utility-based agent has advantages in terms of flexibility and learning. The agent is efficient to address conflicting goals by utility function that specify the appropriate trade-off. If the agent is given several goals, none of which can be achieved with certainty, utility provides a way to weight a likelihood of success against the importance of the goals. Thus, the utility-based agent chooses the action that maximizes the expected utility (i.e., the average utility the agent expects to derive) given the probabilities and utilities of each action outcomes.

B. Learning Agent

Learning agent operates in initially unknown environments and depend on three main segments: learning elements, performance element and optimizer (i.e. problem generator) as illustrated in Fig. 3(a). The learning is a process to modify some (or each) of the above segments (bringing them closer in agreement) by the available feedback to improve the overall performance measure as an objective criterion for success of an agent’s behavior. The agent is successful based on the measure how well it performs for unobserved percepts. Most common learning tasks are characterized by the feedback given to the agent as supervised, unsupervised and reinforcement learning.

The learning element is responsible for making improvements by provided feedback on how the agent is doing. It determines how the performance element should be modified to do better in the future. Design of a learning element is affected by the following:

- which segment of the agent is to be improved;
- what prior knowledge the agent already has;
- what representation is used for the data and the components;
- what feedback is available to learn from.

The performance element is responsible for selecting an action. Previously described agent types represent the performance element. The performance element uses

---

**Figure 2.** Agent function and interaction with environment.

**IV. AUTONOMOUS AGENT DESIGNS**

AI models the network by environment and intelligent agent called the world as illustrated in Fig. 2. The agent is composed of an architecture and an agent program as illustrated in Fig. 2(a). The architecture is a computing device having integrated sensors and actuators supporting the agent program.

The agent program runs on architecture to generate mapping function \( f : P \rightarrow A \), maps from percepts \( P \) to actions space \( A \) where the percepts provide information about environment without indication of the agent’s success. In this context, the agent is autonomous to the extent that its behavior is determined by its own experience, where initially the agent acts randomly unless it is designed with some built-in knowledge. The agent retains its internal state to encode beliefs about its environment and ability to modify its beliefs by perception, reasoning and acting. The agent actions depend on the prior knowledge about the state of agent and its environment, interactions with the environment (including percepts of current state of environment and past experiences of previous actions and percepts), the goals or preferences over states of the world and abilities regarding actions that agent can carry out.

Next, we briefly outline basic types of agent programs [9], [13].

A. Agent types

Simple reflex agent reacts on the current percept of environment creating relations to select the action. Such a relationship is called a condition-action rule written as: if percept condition then action. This is the simplest form of agent, but limited to built-in knowledge by designer. The agent performs well in fully observable environment, where the correct decisions can be made based on current percept. Thus, the agent is blocked if current percept is not provided by designer. This is because the built-in knowledge maps directly from percepts to actions.

Model-based reflex agent retains an internal state (that depends on the percept history) to track not yet observed environment. The update of state requires to hardcode the information how the world evolves independently of the agent and the information how actions taken affect the world. This information about the world is called a model of the world. The challenge is to determine the current state exactly from a partially observable environment so that the agent makes the best guess of the state.

Goal-based agent is given a goal describing desirable situations as “happy” and “unhappy”. The agent combines information about a goal with the model of environment to select actions that achieve the goal. In complex environments, where sequence of actions is necessary, search and planning subfields of AI are developed to find action sequences that achieve the goal. Decision-making in this case is not condition-action rule. This is because we consider a result and satisfaction of a taken action with respect to given goal. In comparison with previous agents the goal based agent is more flexible since it’s behavior can easily be changed by specifying different goal.

Utility-based agent handleless decision-making in partially observable and stochastic environment. The agent utilizes a more general performance measure for comparison between different states according to exactly how happy they would make the agent. The agent assigns a utility of performance measure to any given sequence of environment states. The utility distinguishes between more and less desirable ways of reaching a goal. Similarly, like goal-based agents the utility-based agent has advantages in terms of flexibility and learning. The agent is efficient to address conflicting goals by utility function that specify the appropriate trade-off. If the agent is given several goals, none of which can be achieved with certainty, utility provides a way to weight a likelihood of success against the importance of the goals. Thus, the utility-based agent chooses the action that maximizes the expected utility (i.e., the average utility the agent expects to derive) given the probabilities and utilities of each action outcomes.

B. Learning Agent

Learning agent operates in initially unknown environments and depend on three main segments: learning elements, performance element and optimizer (i.e. problem generator) as illustrated in Fig. 3(a). The learning is a process to modify some (or each) of the above segments (bringing them closer in agreement) by the available feedback to improve the overall performance measure as an objective criterion for success of an agent’s behavior. The agent is successful based on the measure how well it performs for unobserved percepts. Most common learning tasks are characterized by the feedback given to the agent as supervised, unsupervised and reinforcement learning.

The learning element is responsible for making improvements by provided feedback on how the agent is doing. It determines how the performance element should be modified to do better in the future. Design of a learning element is affected by the following:

- which segment of the agent is to be improved;
- what prior knowledge the agent already has;
- what representation is used for the data and the components;
- what feedback is available to learn from.

The performance element is responsible for selecting an action. Previously described agent types represent the performance element. The performance element uses
performance standard to distinguish the part of percept as a reward (or penalty) providing direct feedback on the quality of the agent’s behavior.

The optimizer is responsible for suggesting exploratory actions that will lead to new and informative experiences. This is because the performance element would keep selecting the “best” actions given what it knows. The optimizer enables agent to explore and select suboptimal actions in the short run, while discovering better actions in the long run. It is obvious that agent programs may have variety of components represented in different ways, so there appears to be great variety of learning methods. Finally, the critic tells the learning element how well the agent is doing with respect to a fixed performance standard outside of the agent.

C. Reinforcement Learning

RL agent envelops all of AI such that the agent should learn to perform successfully in observed environment. The agent learns a transition model for its own actions and uses a reward/penalty to learn an optimal or near-optimal policy for the environment as illustrated in Fig. 3(b) [9]. RL is model-free iterative learning technique with goal to reach optimal policy. The agent can be designed as passive or active learner. The former considers that the agent’s policy is fixed and the task is to learn the utility states leading to learning of a model of environment. The later considers that the agent must learn what to do by exploring to experience its environment and learn how to behave in it.

RL agent is mostly designed as utility-based agent, reflex agent or Q-learning agent. The agent learns utility state or an action-utility (Q-)function giving the expected utility of taking a given action in each state. Unlike the former two, Q-learning agent does not need a model of environment and it can compare expected utilities for its available choices without needing to know their outcomes. However, due to lack of the model the agent cannot look ahead and does not know where their actions lead. This is major learning limitation of this type of agents.

Thus, in dynamic and complex network environments RL agent is highly suitable for active learning, where the acts to obtain suitable examples from which to learn. In active learning, the agent “guesses” which examples would be suitable to learn from and take actions to probe these examples (see Sect. V-B.6).

V. KNOWLEDGE-DRIVEN-OPERATION WITH AI

Figure 4 illustrates the conceptual functions and enablers for autonomous networking. The agent needs to be fully aware of different network and service functions, while taking decision and actions in real-time based on predicted service sessions. From the agent’s perspective, the sessions are fully observable environments, but the network and user environments are partially observable. Here, AI with ML techniques is the major enabler that potentiate agent to learn from its experiences supporting reasoning and decision-making.

A recent attempt has been made successful to shift from the concept of knowledge-discovery toward knowledge-driven (autonomous) operation in [12]. The authors design autonomous agent to address the problem of self-deployment. The idea is to manipulate with knowledge by retaining and reusing past experiences in knowledge base to reason out an optimization strategy and search for new actions. In [15], the authors present domain knowledge driven RL agent design with Q-learning to solve self-optimization problem with joint channel association and location optimization.

A. Implementation of AI

The AI can be implemented in different forms such as rule-based system (RBS), ontology-based system (OBS) and case-based reasoning (CBR), among others [9, 13]. The RBS comprises a set of rules with predefined actions created by experts in the network domain. Similarly, OBS applies logic based reasoning for the domain attributes. Both RBS and OBS
require explicit domain knowledge to define the relations between rules and actions or objects. This is unlike the CBR that relies on the system memory (i.e. knowledge base (KB)) to build the knowledge using observations about previous actions and their impact on the network. A RL also benefits from such reasoning system which significantly speeds up learning of unknown environment and improves agent efficiency.

The intelligent agent perceives its environment roughly through a sequence of sensing, reasoning and acting to build its own knowledge and use it in the future actions. Thus, good actions, e.g. that achieve target quality-of-service, can be reused directly in the future when similar network conditions are sensed, while bad actions, e.g. that create coverage holes, will be considered to refine the searching strategy of the agent. The agent applies the following four stages on the KB [9]:

- Retrieve the most relevant case, in the KB, to the currently sensed information;
- Reuse the retrieved case or relative experience to solve the sensed problem;
- Revise the KB by updating the actions or learning fitness values of the stored cases;
- Retain the new cases (i.e. experiences) in the KB to be used in the future.

The framework illustrated in Fig. 5 implements the above four stages by the following design functions.

B. Network Self-organization

Self-organization is an autonomous process where a system’s structure and functionality at the global level emerge from interactions among the “lower-level” subsystems of the system without any external or centralized control [10]. The subsystems interact in a local context either by means of direct communication or environmental observations without reference to the global pattern. A summary of the self-organization features and their relationships with below presented AI functions is given in Table I.

The agent presented in Fig. 5(a) provides full flexibility to design each of the functions independently depending on the implementation. Such agent could be considered for single-agent problems such as network planning for example. The illustration in Fig. 5(b) presents the second approach, where RL is considered to update policies in the learning of a model [15]. The design functions of the framework are KB, sensing, perception, reasoning, decision making, optimization and learning with design implementation in [12]. Next, we describe autonomous agent design functions.

1) Knowledge Base

Knowledge represents the facts about an environment acquired through experience that can be used to solve problems in that environment. Knowledge is represented in agents in the form of “sentences” (used here as technical term) in a knowledge representation language that are stored in a KB (see example in [15]. KB is the representation of all the knowledge that is stored. (e.g. perceived network performance and expected quality-of-service after applying each action plan). The agent retrieves the most relevant case from the KB and reuses the corresponding action to solve the current problem (e.g. a simple case is a triplet of problem, action and learning coefficient [12]). A problem is a vector referring to sensed measurements that perceive the current network performance state. For each stored problem, an action can be performed which is essentially setting a new operational parameter in the network. After the action is applied, the learning coefficient of this action is calculated depending on the degree of how much (or does not) improves network performance. For example, the coefficient may be defined as a ratio between the achievable throughput and the requested user service demand, representing the quality-of-service satisfaction criteria.

2) Sensing

Sensing function is responsible for collection of network measurements via programmable interfaces (e.g. remote management protocols or interfaces defined by vendor software development kit). Examples of measurements in wireless network are modulation and coding scheme, load estimation,
channel state information (user terminal measurements), operating frequency, received power level, channel busy time, re-transmissions, failed packets, sent packets, user application and location, to name just a few. As discussed in Sect III.B, we note that the measurements collection strategy is selected depending on the use case by the perception function to perceive the environment, describe its current state of the wireless system and/or assess previous actions [12].

3) Perception

Measurements are transformed to different sensors by the perception function translating the measurements from each network element into system variables describing the state of the system. The example of such variables for self-deployment use case are: deployment decision variables (e.g. location allowed or not), network association variables (e.g., user association with an access point) and performance variables (e.g., user’s achievable throughput or demand) [12]. The percepts (i.e. performance indicators) are calculated for each radio interface of the element based on two successive sensing samples to detect the network and user state (e.g. unsatisfied user demand).

The perception function runs real-time network state monitoring to detect when the current configuration becomes sub-optimal and signals to evaluate the current operational state of the network. Examples of perception sensors are user battery status, mobility, coverage/signal quality indicator, achievable throughput to name just a few.

4) Reasoning

AI can be implemented as CBR with KB, where the percepts are stored and used to solve a new problem. Reasoning manipulates with entries of the KB to produce an action by identifying similarity between cases in KB and newly observed percepts. In the case that achievable throughput is perceived as lower than the user’s demand, the reasoning function should compare the current percept with the previously experienced situations in KB.

Reasoning function searches for the best action in response to perceived state of the agent. The current percept is compared to all the stored case in the knowledge base to calculate a similarity factor (e.g. Euclidean distance) [12]. In the case of high similarity with retrieved case having successful action in the past the action is reused to solve the current problem. Thus, the agent behaves not purely by reflex, but on built-in representation of knowledge in KB (i.e knowledge-based agents). This is called deterministic reasoning, but other approaches based on probabilistic reasoning may be considered [9].

5) Decision-making

Decision-making function evaluates actions by ability to meet the goal under the current and future percepts [9]. The agent stores previous action plans and through learning evaluates their quality through direct or indirect feedback (i.e learning fitness). The function checks both the similarity of cases and the fitness of the retrieved case(s). For example, if an observation is not true then the KB holds no matching case and two scenarios are possible and (i) a new case must be retained in KB or (ii) the best matching case has a suboptimal action that should be recomputed [12]. The optimization function is triggered to calculate a new action that will be executed and stored in the KB. We note that decision-making addresses the maximization of expected utility in episodic or sequential decision problems [9]. Some of techniques for decision-making are Markov decision processes (MDPs), game and optimization theory.

6) Optimization

Given the agent function, the general idea behind optimization is to tune some of the agent components that are left unspecified to produce the required behavior. Optimization (i.e. problem generator) is devising a search plan of action to achieve goal by finding the best hypothesis within action space [9]. Thus, the optimization function defines searching strategy for new actions. The optimization search direction for new actions can be implemented through exploitation and exploration [12], [15]. Exploitation greedily optimizes the network metrics within a limited search space that appears to be promising (already experienced knowledge in KB). Exploration aims at discovering (mostly sub-optimal) new search spaces that may lead to more promising (near-optimal) solutions than the currently exploited solution set. For example, dynamic programming methods have been used in AI to solve optimization problems by storing intermittent actions so that they can be re-used. Examples of some actions are reconfiguration of adaptive modulation and coding, frame size, power and channel adjustment, antenna parameters management, scheduling, handover, routing, access point location to name just a few.

7) Learning

Learning is a property of the agent to improve its behavior based on experience, e.g. such that it can do more, it can do things better and/or it can do tasks faster. Learning function is acquiring knowledge based on the observed states after applying the action enriching the knowledge used in reasoning function. By learning we gain experience and enrich built-in knowledge, update risks and rewards of actions, and reselect goals. The learning function controls update of cases the entries in the KB by improving the fitness accuracy of each action [12]. When the user is involved in providing direct or indirect feedback, semi-supervised learning may be adopted. Strictly supervised learning techniques are not applicable due to the lack of full knowledge about the environment (i.e. network environment, traffic demand, true customer satisfaction, etc.) leading to non-realistic models. Some learning methods are support vector machines (SVMs), neural networks (NNs), meta-heuristic algorithms, fuzzy logic, genetic algorithms, hidden Markov models (HMMs), Belief networks and multi-agent learning.

We note that although learning problem is mainly about finding the best model that fits the data the learning does not stand apart from the rest of AI. Beyond the fitting of data there are many issues such as domain knowledge representation; how when and what type of data to collect; and how to exploit learned experiences to improve agent functions.
VI. CONCLUSION AND RESEARCH DIRECTIONS

Unlike knowledge-discovery approaches supported by data analytics, this paper presented a vision of knowledge-driven wireless operations by joint application of AI disciplines such as sensing, perception, reasoning, learning, decision-making and optimization. Firstly, we discussed the rise of network intelligence and complexity as major drivers toward adoption of AI. We presented basic designs for intelligent agents and finally, we discussed how to utilize AI disciplines to design autonomous wireless network.

We envisioned several research challenges as follows. Due to ultra-dense network deployments, the optimization function should consider multi-objective design strategy such as adversary learning, where reasoning function needs to consider other agents in a multiagent environment. The exploration of probabilistic reasoning and inference for network diagnostics based on belief or neural networks is an interesting challenge. An efficient design of knowledge base would be necessary in large-scale ultra-dense deployments where single or multiple instances of KB would be required. Transfer learning, where reusing of knowledge across different (physical) environments, is an open issue. Finally, integration of localization and user behavior data with AI framework may lead to improved user experience.

REFERENCES

[1] R. Li, Z. Zhao, X. Zhou, G. Ding, Y. Chen, Z. Wang and H. Zhang, “Intelligent 5G: When Cellular Networks Meet Artificial Intelligence,” IEEE Wireless Communications Magazine, pp. 175 – 183, October 2017.

[2] J. Pérez-Romero, O. Sallent, R. Ferrús, R. Agustí, “Knowledge-based 5G Radio Access Network planning and optimization,” 2016 International Symposium on Wireless Communication Systems (ISWCS), 20-23 Sept. 2016, Poznan, Poland.

[3] P. V. Klaine, M. A. Imran, O. Onireti and R. D. Souza, “A Survey of Machine Learning Techniques Applied to Self-Organizing Cellular Networks,” IEEE Communications Surveys & Tutorials, Vol. 19, No. 4, Fourth quarter 2017.

[4] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen and L. Hanzo, “Machine learning paradigms for next-generation wireless networks,” IEEE Wireless Communications Magazine, pp. 98 – 105, April 2017.

[5] X. Wang, X. Li and V. C. M. Leung, “Artificial intelligence-based techniques for emerging heterogeneous network: state of the arts, opportunities and challenges,” Special section on artificial intelligence enabled networking, IEEE Access, Vol. 3, August 2015.

[6] M. Zorzi, A. Zanella, A. Testolin, M. De Filippo De Grazia and M. Zorzi, “Cognition-Based Networks: A New Perspective on Network Optimization Using Learning and Distributed Intelligence,” Special section on artificial intelligence enabled networking, IEEE Access, Vol. 3, pp.1512 – 1530, 2015.

[7] G. Xu, Y. Mu and J. Liu, “Inclusion of Artificial Intelligence in Communication Networks and Services,” Special Issue ICT Discoveries, ITU Journal, No. 1, October 2017.

[8] G. Cao, Z. Lu, X. Wen, T. Lei and Z. Hu, “AIF: An artificial intelligence framework for smart wireless network management,” IEEE Communication Letters, accepted for publication, 2017.

[9] S. Russell and P. Norvig, “Artificial Intelligence: A modern approach,” Prentice-Hall, Englewood Cliffs, vol. 25, p. 27. 1995.

[10] H. Gacanin and A. Ligata, “Wi-Fi SON: Challenges and Use Cases,” IEEE Communication Magazine, Network & Service Management Series, Vol. 55, No. 7, pp. 158 - 164, July 2017.

[11] S. Bi, R. Zhang, Z. Ding and S. Cui, “Wireless Communications in the Era of Big Data,” IEEE Communications Magazine, October 2015.

[12] R. Atawia and H. Gacanin, “Self-Deployment of Future Indoor Wi-Fi Networks: An Artificial Intelligence Approach,” 2017 IEEE Global Communication Conference, 04-08 December 2017, Singapore.

[13] D. Poole and A. Mackworth, *Artificial Intelligence: Foundations of Computational Agents*, second edition, Cambridge University Press, 2017.

[14] H. Gacanin and M. Wagner, “Artificial Intelligence Paradigm for Customer Experience Management in Next-Generation Networks: Challenges and Perspectives,” IEEE Network Magazine, accepted for publication [arXiv:1805.06254], Accessed on June 16th, 2018.

[15] E. Perenda, S. Karunaratne, R. Atawia, H. Gacanin, “Self-X Design of Wireless Networks: Exploiting Artificial Intelligence and Guided Learning,” arXiv:1805.06247v1, Accessed on June 16th, 2018.
| Self-organization feature | Characteristics | AI function | Models |
|--------------------------|----------------|-------------|-------|
| Self-configuration       | System configuration setup either on initial deployment of depending on the current critical situation in terms of network operations: cell coverage and deployment, neighbor cell list, authentication, maintenance updates, etc. Capability to maintain systems and devices depending on pre-defined system configuration. | Currently deterministic feature provided per network by operator’s auto-configuration server or placed locally on memory as system configuration backup. Some ML models are applicable to automatically configure a set of parameters per cell to optimize local-policy. | K-means clustering |
| Self-optimization        | Deterministic (human rule-based) system checks with automated optimization of the local operation parameters according to global objectives: quality of service, capacity/bandwidth, coverage, etc. | Perception/Reasoning: deterministic - belief states are determined by logical formulas, e.g. classification. Optimization – constrained (convex) optimization functions not adaptable to network updates. | Dynamic programing, SVM, HMM, regression, RL, NN |
| Self-healing             | Autonomous (machine-based) system checks and methods for adapting configurations the system-of-systems: network and user’s location-based updates. | Perception/Reasoning depending on a use case deterministic or probabilistic - belief is quantified as likely/unlikely or multi-class. Optimization – reinforcement learning, heuristics/meta-heuristics due to problem complexity supported by learning. Decision-making supported by learning. | NN, HMM, SVM, Bayes networks |
|                          |                            | Learning – supervised, semi-supervised, unsupervised and reinforcement learning. | SVMs, ANNS, meta-heuristic algorithms, fuzzy logic, genetic algorithms, hidden Markov models, Belief networks, multi-agent learning |