An Overview of Machine Learning Approaches in Wireless Mesh Networks

Samurdhi Karunaratne and Haris Gacanin

Wireless mesh networks (WMNs) have been extensively studied for nearly two decades as one of the most promising candidates expected to power the high-bandwidth, high-coverage wireless networks of the future. However, consumer demand for such networks has only recently caught up, rendering efforts at optimizing WMNs to support high capacities and offer high QoS, while being secure and fault-tolerant, more important than ever.

Abstract

Wireless mesh networks (WMNs) have been extensively studied for nearly two decades as one of the most promising candidates expected to power the high-bandwidth, high-coverage wireless networks of the future. However, consumer demand for such networks has only recently caught up, rendering efforts at optimizing WMNs to support high capacities and offer high QoS, while being secure and fault-tolerant, more important than ever. To this end, a recent trend has been the application of machine learning (ML) to solve various design and management tasks related to WMNs. In this work, key ML techniques are discussed and past efforts applying them in WMNs are analyzed, while noting some existing issues and suggesting potential solutions. Directions are provided on how ML could advance future research. Recent developments in the field are also examined.

Introduction

IEEE 802.11 (Wi-Fi) network access has become so ubiquitous in recent years that one expects such connectivity everywhere, whether at home, in the workplace, at a restaurant, or on a plane. Due to their poor coverage and low quality of service (QoS) guarantees, single access point (AP) networks have failed to meet increasing broadband service requirements, resulting in a demand for multi-AP networks called wireless mesh networks (WMNs). However, WMNs are not restricted to Wi-Fi — they are used with many other wireless technologies including IEEE 802.15 (WPAN) and IEEE 802.16 (WiMAX).

A WMN generally consists of mesh gateways (MGs), mesh routers (MRs), mesh clients (MCs), and a set of wireless links among them. An MC can be regarded as a user device and is, in most cases, an endpoint of a flow of traffic through the network. The MCs are connected to a wireless backbone formed by the MRs. The MGs act as the points at which a WMN is connected to a wired infrastructure and, typically, to the Internet. Therefore, a network request originating at an MC would be transferred through its associated MR onto the wireless backbone, where it takes one or more hops to reach an MG before reaching the Internet (and vice versa).

Several factors affect the service (e.g., throughput, delay) experienced by an MC in a WMN, such as interference from other signals and contention due to simultaneous transmissions, to name just a couple. To obtain a demanded level of service, various design challenges like channel allocation, routing, resource allocation, and deployment strategy should be addressed, paying special attention to the intricacies that each problem entails.

Rule-based deterministic techniques that were initially introduced to solve these challenges produce satisfactory performance guarantees, but lack robustness in the face of an ever changing network environment [3, 8, 9]. Real-time optimization algorithms need to be adaptable to adjust themselves to recover from lost performance. Machine learning (ML) techniques are a fitting match to this description, as they can deduce the best decisions to be made by analyzing their growing database of past network statistics and performance data.

The objective of ML is to improve the performance of a system with a set of tasks by statistically analyzing the data it has gathered during the execution of previous tasks. ML techniques have been typically classified as supervised, unsupervised, and reinforcement learning [1, 15]. Supervised learning happens when the input data to a learner is already labeled with human-driven guidance. The input data to an unsupervised learning agent is unlabeled, so the learner must identify features or patterns in the dataset to label the data by itself. Reinforcement learning (RL) is another type of ML technique where the learner perceives its environment to incrementally conduct actions that try to maximize the cumulative value of a reward given in response to previous actions. Unlike supervised learning, where pre-labeled data are input to a learner, the supervision in RL is a reward given after an action is taken.

There has been increasing interest in the application of ML in wireless networks in general over the last decade [1], including in WMNs [3–14]. These attempts have been directed at optimizing various aspects of WMNs to improve user throughput, reduce end-to-end delay, or satisfy other QoS demands, while also trying to improve reliability and security. This article provides readers with a comprehensive overview of the application of different ML techniques in solving major functional design problems and handling management-level tasks in WMNs. These techniques are classified and expounded on to accentuate the problems in WMNs that could characteristically be solved by them. Conclusions and future directions are given at the end.

Digital Object Identifier:
10.1109/MCOM.2019.1800434

Samurdhi Karunaratne is with the University of Peradeniya; Haris Gacanin is with Nokia Bell Labs.

Applications of ML for Functional Design Problems in WMNs

When designing a WMN, various challenges determining the performance of the network need to be addressed; ML has aided in this by becoming an invaluable decision making tool. In each following subsection, a specific design problem is explored.

Routing

Routing is essentially deciding which route — among many possible ones — to take toward the destination at each intermediate MR along the path from source to destination. Historically, various routing metrics like expected transmission count (ETX), expected transmission time (ETT), and message integrity check (MIC) were used for routing in WMNs; other approaches have used techniques like heuristic-based algorithms, linear programming (LP), combinatorial optimization, and even metaheuristics like simulated annealing (SA) [14].

RL is characterized by a system that learns to make optimal decisions from the knowledge gathered by exploring its environment. At each step, it selects an action from a set of possible actions, and subsequently receives a reward from the environment corresponding to that action. Since the best such action is not known a priori, many different actions need to be tried out until the best action is learned (convergence). The learning agent can either select the best action so far (exploitation) or select an action randomly (exploration). In most practical applications of RL, the learning agent is biased, since the time required to learn the best action by simply selecting an action randomly during exploration is too high. Such an RL agent is depicted in Fig. 2. The bias represents some domain-specific knowledge used to guide the learning agent toward convergence.

RL lends itself nicely to the routing problem, as in each routing decision, the possible next hops to take toward a destination could be taken as the set of possible actions in that state. These routes could then be tried out in a fashion similar to trial and error until the best route is learned (the bias, in this case, could simply be the elimination of a few theoretically ineffective routes). It has become common practice to perform the learning in a distributed fashion [3–5], where each MR learns the best routing decisions to be made for itself, without considering other MRs. RL-based models have been extensively used to tackle the routing problem [3–5].

Q-learning: In Q-learning [2], there is a Q-value $Q(s, a)$ associated with performing action a at a state s that is updated each time that action is performed. At a given state, the action with the largest cumulative Q-value is considered the optimal action. Here, the compromise between exploration and exploitation may be made in different ways including simple greedy, ε-greedy, and soft-max [2]. Simple greedy action selection always exploits current knowledge to maximize immediate reward without sampling apparently inferior actions. The ε-greedy method behaves greedily most of the time, but with small probability ε, it randomly selects an action from among all actions with equal probability, independent of their Q-value estimates. In contrast, the Softmax method utilizes calculated action-selection probabilities to choose an action instead of picking an action at random. These probabilities are determined by ranking the Q-value estimates using a Boltzmann distribution.

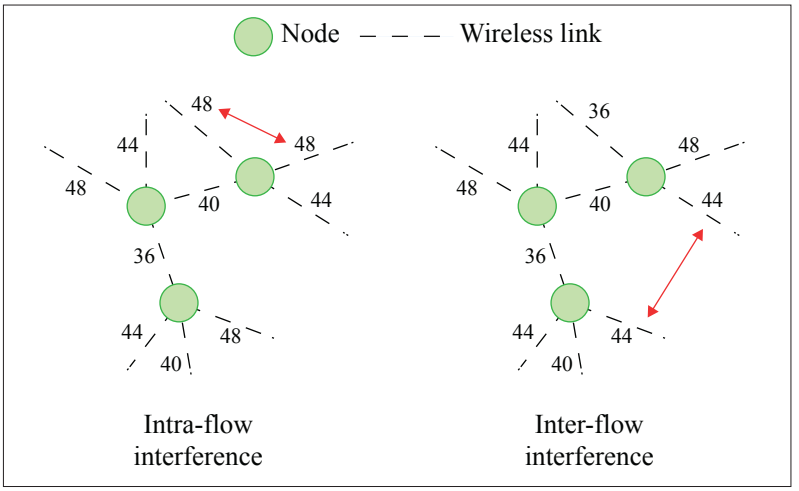

Figure 1. The channel assignment problem in a typical WMN.

The most common routing strategy is to guide (i.e., bias) the RL-agent by facilitating it to estimate the best path based on a rule-based mechanism using certain metrics or physical parameters. For example, in [3], the authors introduced a distributed algorithm called RLBDR where an RL-agent in each MR learns the best neighbor to send an incoming packet toward a given MG. While using an ε-greedy Q-learning strategy, each MR also makes use of theoretical estimates of the best path to the given gateway by calculating a parameter called path quality (PQ) for each possible path. This represents bias in the RL scheme. They also compared RLBDR with other deterministic routing schemes like MIX, ETX, nearest gateway routing, and gateway load-based routing [3]. RLBDR was shown to have much less mean delay and loss while providing significantly better throughput than all of these methods.

Learning Automata (LA): LA can also be classified as a type of RL: the environment of a single learning automaton can be described by a set of states $\mathcal{S}$, a set of possible actions $\mathcal{A}$, and a set of penalties (or rewards) $\mathcal{R}$ corresponding to each action. The automaton maintains a probability vector (ψ) which represents the probability that any action could be selected. Once an action is selected, if a penalty is received, the probabilities for all the other actions are increased and that for the selected action is decreased. One striking difference from Q-learning is that not only is the probability of the selected action affected in LA; every action is affected. LA are suited for distributed decision making in highly stochastic environments. They have been used extensively for WMN optimization tasks [4, 5, 9].

LA are also used for routing problems in a distributed fashion similar to Q-learning where a learning automaton installed at each MR considers the set of next hops as $\mathcal{A}$ and the set of destinations as $\mathcal{S}$, as illustrated in Fig. 4. A multicast routing protocol called Learning Automata Based Multicast Routing uses such LA installed on each interface of a node to build a multicast tree from minimal end-

An ANN usually consists of nodes called artificial neurons, which have connections between them known as edges. These edges typically have a weight that adjusts as learning proceeds. The weight is proportional to the strength of the signal at a connection. Each artificial neuron computes an output based on a nonlinear function of its inputs, which may be originating from other nodes or be external inputs.

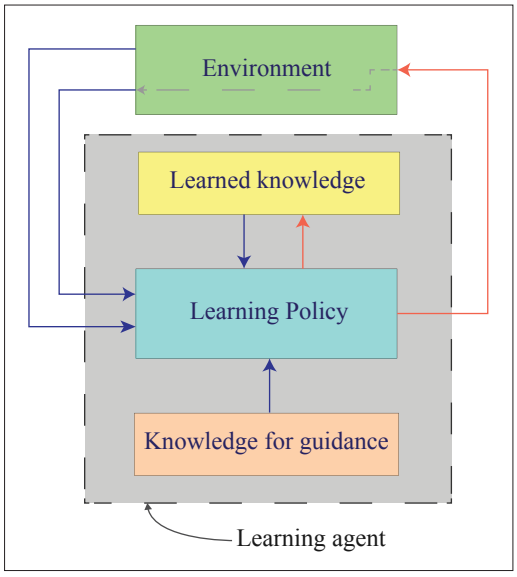

Figure 2. Reinforcement learning: the biased RL mechanism.

to-end delay paths between the source and each multicast receiver [4]. Then the LA optimize the initial tree to get a minimal interference tree. In [5], another multicast routing algorithm called Distributed Learning Automata-Based Multicast Routing Algorithm is proposed. It shrinks the action set of an MR by constructing a minimum Steiner connected dominating set iteratively, using LA distributed in each node. Also, the learned information is distributed among neighboring nodes to increase the convergence rate.

The delay of collecting feedback should be noted as one specific issue affecting the use of RL techniques like Q-learning and LA in WMN routing. In contrast to a design problem like channel allocation, routing decisions need to be made at a much greater frequency. Therefore, collecting feedback for each and every decision may not be feasible, primarily due to two reasons:

1. Increased control overhead that might result in link congestion
2. Delayed update of the database causing the approach to be less reactive (especially in highly dynamic environments)

One possible solution is to only give a single collective reward for a batch of consecutive actions instead of one for each action. Although less granular, this has the added benefit of mitigating oversensitivity of the RL-agent due to transient changes in the environment. This mechanism is portrayed in Fig. 3.

Artificial Neural Networks (ANNs): ANNs [15] have been developed to mimic the operation of a human brain, mostly to aid in recognizing nonlinear relationships in datasets. An ANN usually consists of nodes called artificial neurons, which have connections between them known as edges. These edges typically have a weight that adjusts as learning proceeds. The weight is proportional to the strength of the signal at a connection. Each artificial neuron computes an output based on a nonlinear function of its inputs, which may originate from other nodes or be external inputs.

A multicast routing algorithm using a type of ANN called Cerebellar Model Articulation Controller (CMAC) has been proposed to predict the probability of route and node disconnection (failures) to help select better routes [4]. The input space of the CMAC is quantized into discrete states called blocks, and memory cells will be associated with each state to store information (output) for that state. CMAC neural networks exhibit advantages like speedy learning and exceptional convergence properties.

Channel Allocation

The channel allocation problem deals with allocating channels to the wireless links among MRs in a WMN such that interference effects are minimized and channel utilization is maximized. For example, consider the scenario given in Fig. 1: four non-overlapping channels 36, 40, 44, and 48 (in the 5 GHz band) should be assigned among the given links to avoid the possibility of:

1. Intra-flow interference (between links of the same flow)
2. Inter-flow interference (between links of adjacent flows)

Past work has modeled the channel assignment problem as an edge coloring problem, vertex coloring problem, or max k-cut problem, solved using various techniques like heuristics, integer linear programming (ILP), and polynomial-time approximation scheme (PTAS) [14].

A typical use case of LA for channel allocation installs an automaton at each MR (or at each radio of an MR in a multi-radio scenario). The set of actions $\mathcal{A}$, in this case, is assigning each of the possible channels to the radio, while the state could be defined based on the current channel assignment. To this end, Leith and Clifford [6] proposed a self-managed LA-based algorithm that does not require any communication between MRs. Each automaton maintains and updates a vector ψ, which contains a probability corresponding to each channel that reflects its history of interference. If the current channel quality is above a certain threshold, the MR will continue to operate in it; otherwise, a channel is selected randomly based on the current value of ψ. They also theoretically proved that the convergence of their algorithm is guaranteed, provided that the channel assignment was feasible.

Bayesian Learning: Bayesian learning tries to calculate the posterior probability distribution of the target features of a testing object conditioned on its input features and the entire training dataset. An example of an object could be a wireless channel, while its features could be measurement data on its signal, noise, and interference levels measured at a radio operating on that channel at a particular MR. Starting with some guesses about the probability of an event occurring (prior probability), what happens (likelihood) is observed, and depending on what happens, the initial guess is updated. Once updated, the prior probability is called posterior probability. Bayesian learning is well suited for occasions where there is a limited number of data points and when outliers need to be handled well. Examples include maximum likelihood estimation (MLE) and maximum a posteriori estimation (MAP). MLE is a special case of MAP that uses a uniform prior distribution.

In the practical application of Bayesian models, Gibbs sampling provides a convenient way to

approximate posterior distributions. Assume we have random variables $\boldsymbol{S} = \{s_1, s_2, ..., s_n\}$. We start from the initial values $S^{(0)} = \{s_1^{(0)}, s_2^{(0)}, ..., s_n^{(0)}\}$, which can simply be taken from the prior distribution, and iteratively calculate $S^{(t)}$ from $S^{(t-1)}$ until sufficiently large t that makes $S^{(t)}$ appear as a sample of the true posterior distribution. The authors of [7] used this method by defining $\boldsymbol{S}$ as the set of states of the n MRs of a WMN, where each state S_i represented the channel assigned to the ith MR. Their objective was to find a set of channels $S^{(t)}$ that minimized the total interference received by all MRs. For this, they defined an energy function on each node where the energy depends on the channel assignment to that node. Then a Gibbs sampling procedure was conducted by which the network converges to a collection of states with minimum global energy — the optimal channel assignment.

***k*-Means Clustering:** *k*-means clustering [15] groups a set of unlabeled data consisting of n observations into a group of k clusters; each observation is assigned to the cluster to whose centroid it has the nearest Euclidean distance (the Euclidean distance is defined based on the features of the observations). The centroid of each cluster can be used as its label and is usually defined as the mean of the data points within that cluster. The most common algorithm for *k*-means clustering uses an iterative refinement technique.

One specific potential application of this technique related to channel assignment should be highlighted. Several algorithms using rule-based procedures cluster nodes to several groups with the purpose of treating channel allocation in a divide and conquer fashion [14]. A typical approach would assign the same channel to radios within the same cluster and present a methodology to assign channels to radios on the boundary between clusters to achieve inter-cluster connectivity. It is to be noted that *k*-means clustering could be used to intelligently cluster the set of MRs for this purpose. Most such approaches also require a cluster head to function; *k*-means clustering is naturally suited for this, as the centroids of the generated clusters could be used for this purpose (or some variation of it).

Network Deployment

Network deployment typically deals with placing the MGs and MRs at locations that are optimal to achieve maximal network performance. Even though many other optimization problems like routing and channel assignment assume a pre-defined placement of these nodes, the performance of their ultimate outcome depends on the initial physical arrangement of nodes. For example, a typical problem is figuring out the minimum number of MGs and the ideal location for them to be placed.

Metaheuristic techniques like simulated annealing (SA), genetic algorithms (GAs), and particle swarm optimization (PSO) have virtually become the de facto standard for intelligently solving MR and MG placement problems in WMNs. However, it must be noted that RL techniques like Q-learning and LA may still be worth exploring in this regard. A recent attempt has been made successfully at solving the MR placement problem where the idea of balancing the fronthaul and backhaul throughput of an MR was employed as a strategy [8]. Semi-supervised support vector machines (S3VMs), which are a variation of support vector machines (SVMs) that support unlabeled data, were used to identify throughput regions, while an exploration and exploitation strategy like RL was used as the learning strategy.

WMN problem	Objective	ML techniques used
Routing	The path with lowest cost to direct traffic from a source to destination	ANN [4], Q-Learning [3], LA [5], MDP [10]
Channel assignment	Assigning channels to radio(s) of nodes while minimizing interference	Bayesian learning [7], LA [6]
Network deployment	Placement of MGs and MRs to meet network demands like coverage	SVM [8]
Rate adaptation	Rate at which data is transmitted between each pair of nodes	LA [9]
Joint approaches	Solving multiple complementary problems	MDP [10]
Anomaly and intrusion detection	Detecting and alerting users about possible attacks	DT [11], SVM [13]
Integrity and fault detection	Identification of faults and/or changes in the network	PCA [12]

Table 1. Summary of WMN problems and corresponding ML techniques as solution tools.

Rate Adaptation

A WMN must expect a number of different flows of traffic at any given moment (one of the primary objectives of a WMN is supporting a higher number of simultaneous users than a single AP network). From the perspective of improving per-user throughput and fairness, it is vital that these transmissions between source-destination pairs happen concurrently. In conjunction with scheduling these different flows of traffic, the data rate at which they are transmitted is of importance as it ultimately impacts the inter-flow conflicts and hence throughput.

The rate adaptation problem also has characteristics that make it naturally attractive for RL-based solutions: let us take the Stochastic Automatic Rate Adaptation Algorithm (SARA) [9], for example. It deploys a stochastic learning automaton (SLA) at each MC of the WMN. The set of states $\mathcal{S}$ here is the set of potential receivers for a given MC i, which can be any other node of the network (the MG, an MR, or another MC). For each receiver j, the set of actions $\mathcal{A}$ is the set of possible transmission rates whereby data can be transmitted toward i from j. SARA sets equal probability for each rate in the beginning, chooses a data rate to transmit according to the current probability, and updates the probability vector according to subsequent throughput values achieved with respect to each rate. At convergence, the rate that provided the highest throughput will have the highest probability. In comparison with rule-based approaches like ARF and AARF, the throughput guarantees of SARA were shown to be far superior [9].

Joint Approaches

All the above design challenges are in fact sub-problems of the singular problem of designing a WMN where all of them are optimized to coexist and, more importantly, complement each other. In a real-life WMN, solutions to these prob-

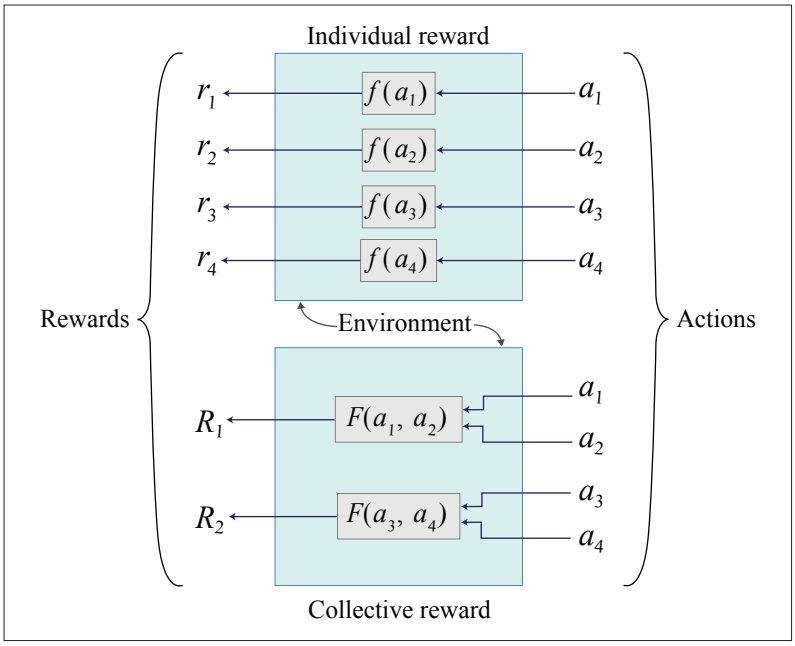

Figure 3. Combating delayed feedback in RL using collective rewards.

lems ultimately need to be used together, and even though they may perform well individually, they might act in detrimental ways to one another when deployed together. For example, the physical arrangement of the network restricts the improvements that could be made with channel assignment, which together with power control determines the connectivity. Decreased connectivity limits the number of possible routes in the network.

Markov Decision Process (MDP): An MDP [2] is just like a Markov chain, except the transition matrix depends on the action taken by the decision maker (agent) at each time step. The agent receives a reward, which depends on the action and the state. The goal is to find a function, called a policy, that specifies which action to take in each state, so as to maximize some function (e.g., the mean or expected discounted sum) of the sequence of rewards. One can formalize this in terms of Bellman's equation, which can be solved iteratively using policy iteration. The unique fixed point of this equation is the optimal policy.

In [10], Zhang *et al.* proposed a joint admission control and routing protocol that provides QoS guarantees in WMNs based on IEEE 802.16. The problem was modeled as a semi-Markov decision process (SMDP) and solved using a linear programming-based algorithm. The actions of the SMDP framework were whether or not to admit a user when a new or handoff connection request arrives, and to which route the incoming connection should be assigned. Multiple service classes were prioritized by imposing a different reward rate for each service class (service classes are defined in IEEE 802.16). The action chosen was based on the number of sessions of each class of traffic.

Applications of ML for Network Management in WMNs

When maintaining a WMN, it is critical to pay attention to certain management-level issues that may compromise the security, integrity, or expected performance level of the system. As network demands, computing protocols, and user expectations have become more and more complex over the years, ML has proved to be a vital instrument in developing tools to meet these challenges.

Anomaly/Intrusion Detection

Intrusion detection systems (IDSs) are used to alert users about possible attacks, ideally in time to stop an attack or mitigate the damage. They consist of three functions:

1. Event monitoring: The IDS must monitor some type of events and maintain the history of data related to these events.
2. Analysis engine: The IDS must be equipped with an analysis engine that processes the collected data to detect unusual or malicious behavior.
3. Response: The IDS must generate a response, which is typically an alert to system administrators.

Decision Tree (DT): DTs are learning trees where the internal (non-leaf) nodes represent decision conditions, and the leaf nodes represent a class or a feature of the input object (depending on whether a classification or a regression is being performed). By iterating down the tree, a final decision can be made. A number of different algorithms like Iterative Dichotomizer 3 (ID3) and its improved successor, C4.5, can be used to construct decision trees from class-labeled training tuples.

In [11], a cross-layer-based IDS is presented tht trains a normal profile from features collected from both the MAC layer and network layer. It includes four components: data collection, profile training, anomaly detection, and alert generation. Raw datasets are processed and loaded into the profile training module in which they used several classifiers like C4.5 (DT) and SVM (described below) for pattern learning. Finally, any observed behavior that deviates significantly from the profile is considered an anomaly, and an alert is triggered. The authors showed that their cross-layer-based IDS has a higher detection rate and lower false alarm rate than a standard network-layer-based IDS across a number of anomaly models.

Support Vector Machines: In an SVM [15], each data point is represented as an *n*-dimensional vector, and the goal is to construct hyperplanes that best separate the set of data points into classes. The best separation can be defined in many ways, one being the hyperplane that maximizes the distance to the nearest data point of any class.

The authors of [13] developed an SVM-based IDS. First, packet delivery ratio, packet arrival interval, and end-to-end delay statistics are gathered under normal and attack conditions. Then the normalized datasets corresponding to each of these features is input to the SVM to train the detection model. The trained model is then able to predict attack scenarios in test conditions; however, an alert threshold is set so as to minimize false positives before actually alerting the user in the response stage.

Integrity and Fault Detection

The inherent features of wireless communication such as interference, limited bandwidth, packet loss, dynamic obstacles, and propagation loss make WMNs unstable and somewhat unreliable

on certain occasions. As such, they may experience various failures (e.g., node or link failures), which may result in service interruption or degradation of performance. Hence, it is crucial to develop methods that can monitor the network and identify faults accurately and descriptively so that they can be remedied quickly.

Principal Component Analysis (PCA): PCA [15] takes a set of data and tries to reduce it into several principal components (PCs), which is a set of linearly uncorrelated variables. These components are ordered by the variance in the data that each component encapsulates, such that the first component has the highest variance. Being an exercise in finding maximal variance, prior normalization of input data is of utmost importance in PCA. It has applications in any problem where the number of variables is too large for a computationally feasible solution to be achieved and a smaller subset of those variables needs to be considered.

PCA has been used in the fault detection of WMNs in [12]. Here, they analyze the number of packets transmitted in l flows of data measured in p successive time intervals. After application of PCA on these flows, they are able to differentiate flows with high variance (abnormal flows), and hence identify faults. By developing an identification scheme that involves reverse-mapping the derived principal components back into the measurement space, they were not only able to reduce the number of false alarms but also to pinpoint the nodes causing the actual anomalies.

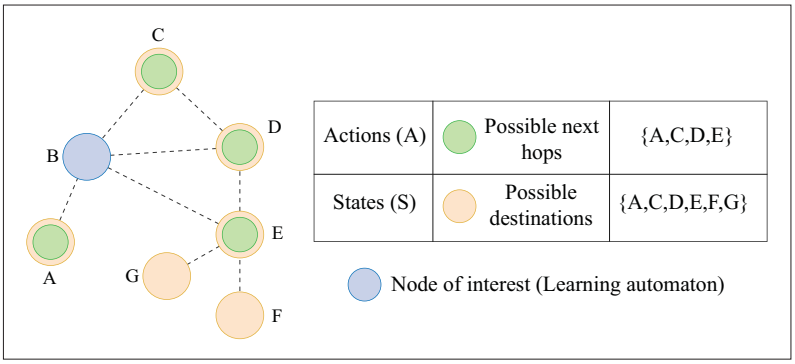

Figure 4. Typical LA-based solution to the routing problem.

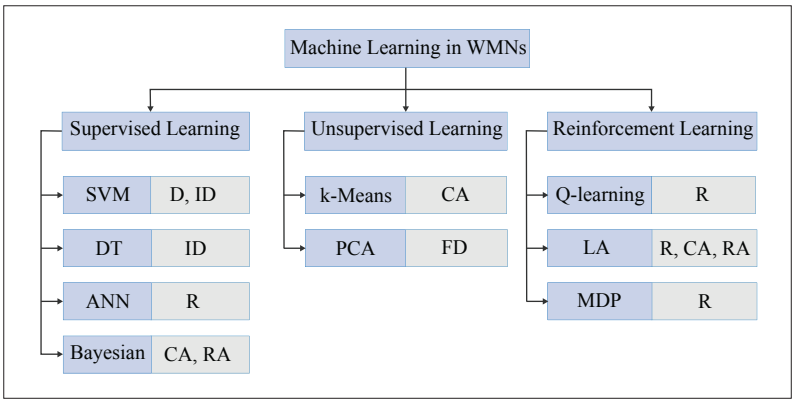

Figure 5. Use of different ML techniques in selected WMN applications: R: routing; CA: channel assignment; D: network deployment; RA: rate adaptation; ID: intrusion detection; FD: fault detection.

Conclusions and Future Directions

An overview of different design and management problems in WMNs has been presented, along with how different ML techniques have been applied to address them, as summarized in Table I and Fig. 5. The emphasis is on portraying how different WMN problems are formally abstracted and transformed into an ML problem. Some issues facing current applications are also discussed and potential solutions presented, highlighting ways in which ML could potentially be applied in future WMN research.

From this overview, it may be clear that RL-related techniques have been responsible for a large portion of the research efforts aimed at utilizing ML in optimizing design problems of WMNs. However, promising RL techniques such as temporal difference learning, Dyna-Q, and prioritized sweeping have been left unutilized in past research. These may provide better convergence guarantees and higher convergence rates than the already utilized methods like Q-learning and LA, which are very attractive features for a real-time stochastic system like a WMN.

Emerging techniques like deep Learning (DL) have a huge potential toward WMN optimization. For example, in a problem like channel allocation, different metrics derived from measurable physical parameters have been used so far to evaluate effects of overlapping channels. These metrics become even less useful in scenarios where multiple neighboring networks also interfere. It is more likely that the best metrics to evaluate such complex effects is nonlinear in nature and hard to deduce intuitively. Deep neural networks (DNNs), which are a common implementation of DL, are ideal for these types of nonlinear feature extraction tasks. As such, DNNs may be used, for example, to identify the features (metrics) that are best suited for consideration during channel assignment.

A note must also be made on cognitive radios (CRs). CR is a concept where the radios in a wireless network (not necessarily a WMN) intelligently manage and utilize the limited bandwidth spectrum. Although not all intelligent WMNs have CRs on them, CR-fortified WMNs can enable even existing intelligent optimization schemes to become better, and open the door to new ones. Therefore, future research should aim to realize the inherent symbiosis between these two technologies.

References

[1] C. Jiang *et al.*, "Machine Learning Paradigms for Next-Generation Wireless Networks," *IEEE Wireless Commun.*, vol. 24, no. 2, Apr. 2017, pp. 98–105.

[2] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 2012.

[3] M. Boushaba *et al.*, "Reinforcement Learning Based Routing in Wireless Mesh Networks", *Wireless Networks*, vol. 19, no. 8, 2013, pp. 2079–91.

[4] M. Jahanshahi and A. B. Talebi, "Multicast Routing Protocols in Wireless Mesh Networks: A Survey," *Computing*, 2014, pp. 1–29.

[5] J. A. Torkestani and M. R. Meybodi, "Weighted Steiner Connected Dominating Set and Its Application to Multicast Routing in Wireless MANETs," *Wireless Personal Commun.*, vol. 60, no. 2, Feb. 2010, pp. 145–69.

[6] D. J. Leith *et al.*, "WLAN Channel Selection Without Communication", *Computer Networks*, Jan. 2012.

[7] B. Kauffmann *et al.*, "Measurement-Based Self Organization of Interfering 802.11 Wireless Access Networks," *26th IEEE ICCC*, Barcelona, Spain, 2007, pp. 1451–59.

[8] R. Atawia and H. Gacanin, "Self-Deployment of Future Indoor Wi-Fi Networks: An Artificial Intelligence Approach," 2017 GLOBECOM, Singapore, 2017, pp. 1–6.

[9] T. Joshi *et al.*, "SARA: Stochastic Automata Rate Adaptation for IEEE 802.11 Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 19, no. 11, Nov. 2008, pp. 1579–90.
[10] S. Zhang, F. R. Yu and V. C. M. Leung, "Joint Connection Admission Control and Routing in IEEE 802.16-Based Mesh Networks," *IEEE Trans. Wireless Commun.*, vol. 9, no. 4, Apr. 2010, pp. 1370–79.
[11] X. Wang *et al.*, "Cross-Layer Based Anomaly Detection in Wireless Mesh Networks," *9th Annual Int'l. Symp. Applications and the Internet*, Seattle, WA, 2009, pp. 9–15.
[12] S. Hakami *et al.*, "Detection and Identification of Anomalies in Wireless Mesh Networks Using Principal Component Analysis (PCA)," *Int'l. Symp. Parallel Architectures, Algorithms, and Networks*, Sydney, Australia, 2008, pp. 266–71.
[13] E. A. Shams and A. Rizaner, "A Novel Support Vector Machine Based Intrusion Detection System for Mobile Ad Hoc Networks", *Wireless Networks*, vol. 24, no. 5, July 2018, pp. 1821–29.
[14] P. H. Pathak and R. Dutta, "A Survey of Network Design Problems and Joint Design Approaches in Wireless Mesh Networks," *IEEE Commun. Surveys & Tutorials*, vol. 13, no. 3, 3rd Quarter 2011, pp. 396–428.
[15] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*, Cambridge Univ. Press, 2014.

Biographies

Samurdhi Karunaratne (samurdhikaru@eng.pdn.ac.lk) is an undergraduate student at the University of Peradeniya, Sri Lanka expecting to graduate in 2019 with a B.Sc. in computer engineering. In 2017, he joined Nokia Bell Labs for a winter internship.

Haris Gačanin [SM] (haris.gacanin@nokia-bell-labs.com) received a Ph.D. in 2008 from Tohoku University, Japan, where he was an assistant professor until 2010. In 2010, he joined Alcatel-Lucent (now Nokia), where he works as department head in Bell Labs. His professional interest is design of autonomous communication systems. He is a Senior Member of IEICE with 200+ publications.